David Hesketh Honours report

# 1.0 Contents

[1.0 Contents 1](#_Toc511567506)

[2. Introduction 2](#_Toc511567507)

[2.1 Glossary of Terms 2](#_Toc511567508)

[2.2 The Project 2](#_Toc511567509)

[2.3 Background 2](#_Toc511567510)

[2.3.1 The Problem/Rationale/Reasoning for project 2](#_Toc511567511)

[2.3.2 Accessibility in Gaming 3](#_Toc511567512)

[3.4 Ethical Considerations 3](#_Toc511567513)

[3.0 Literature Review 4](#_Toc511567514)

[3.1 Genre 4](#_Toc511567515)

[3.1.1 A Blind Legend 4](#_Toc511567516)

[3.1.2 Killer Instinct 5](#_Toc511567517)

[3.1.3 Blindscape 6](#_Toc511567518)

[3.1.4 Stealthfly 6](#_Toc511567519)

[3.2 Development engine 7](#_Toc511567520)

[3.2.1 Unity 3D 7](#_Toc511567521)

[3.2.2 Unreal Engine 4 8](#_Toc511567522)

[3.3 Development Platform 9](#_Toc511567523)

[3.3.1 Virtual Reality 9](#_Toc511567524)

[3.3.2 Computer 10](#_Toc511567525)

[3.3.3 Mobile 11](#_Toc511567526)

[4.0 Execution 12](#_Toc511567527)

[4.1 Development Solution 12](#_Toc511567528)

[4.1.1 Game Design Inspirations 13](#_Toc511567529)

[4.1.2 Game Genre and Control Scheme 13](#_Toc511567530)

[4.1.3 Notable Features for Accessibility 13](#_Toc511567531)

[2.3.1 Risk Assessment 13](#_Toc511567532)

# 2. Introduction

## 2.1 Glossary of Terms

Sight Loss – An individual is described as suffering from sight loss when their visual acuity is found to be 6/12 or worse as per the ‘State of the Nation Year in Review’ carried out by Specsavers in cooperation with the Royal National Institute for Blind People (Specsavers, 2017).

Project Sprint – The term used to mean a small but intensive period of work done to a short deadline.

Bugs – An error or flaw in the software system which causes the program to behave in unexpected ways or provide an incorrect result.

Binaural recording – A way of recording audio with 2 microphones arranged in such a way as to create a 3D sound effect for the listener.

Skill Ceiling – The point at which a player has mastered their ability to play a game and as such can improve no further.

Mods – Derived from the word “Modify”, to mod a game is to create new or change current content within an already existing game, this can include but is not limited to; levels, characters, objects and game modes.

HUD – Heads Up Display, a method of displaying information to a player visually as part of a games user interface.

## 2.2 The Project

The subject of this report is the design and development of a computer game with the specific aim of being easily accessible and enjoyable to those who suffer from sight loss. The intent behind this is to widen the reach of gaming to those who do suffer from this condition and to shine a light on the community of gamers with this disability.

## 2.3 Background

This project begun as an attempt to design and develop a game following specific constraints. There was the additional aim to show that the community of gamers suffering from sight loss need not be ignored in favour of greater graphical fidelity. To this extent a game has been designed and developed with those who suffer from sight loss as the key target audience. This game can also be played by those with full sight without difficulty.

### 2.3.1 The Problem/Rationale/Reasoning for project

People suffering from sight loss have difficulty playing games with visual effects designed for fully sighted users. Many developers do not take into account the possibility of those without sight playing their game and as such do not include several simple features which make many games much more accessible to those within this community. It is this is inherent lack of accessibility that is the problem within the games industry. This leaves a large market of potential players untapped. This market, treated by many developers as empty or even non-existent, is a wide and diverse one. It is possible that ignorance of developers toward this community is due to the fact that the majority of those suffering from blindness are 50 years of age or older (World Health Organisation, 2010) but the majority of those who play video games are under the age of 36 (Grubb, 2014). Despite this, the community of those who play video games without full access to their sight is a diligent one, taking to message boards and modding communities to help widen many game’s accessibility on their own. The response from many developers regarding this is a positive one, and these outlets can enact real change, as shown when a blind fighting game player raised concerns on the “Killer Instinct” forums (Yin-Poole, 2016). With this in mind, it should not be the responsibility of these gamers to ensure accessibility in the industry.

### 2.3.2 Accessibility in Gaming

The Video Game industry has been a growing economic powerhouse since its creation. From its simple origins in arcades where each play cost 25 cents to the increase of the total industry to a worth of 99.6 billion dollars as of the Newzoo 2016 global games market report (Newzoo, 2016). Despite what is suspected to be continued financial growth, there has been very limited development in the untapped market of accessibility in gaming. With approximately 360,000 people in the UK being registered as blind or partially sighted (1 in 30) as of 2014 according to the Royal National Institute of Blind People (RNIB). The number of those with sight loss is predicted to increase to over 2,250,000 as of 2020 as stated by the RNIB for reasons such as an aging population (Transversal, 2014). This increasing population is however comparatively underserved in the current games market. Audiogames.net shows only an available 606 audio games at the time of writing (Audiogames.net, 2017).

Companies such as Audiogames.net and AppleVis both of which provide gaming accessibility help to those with limited to no sight capabilities. AppleVis provides reviews of iOS games based on how accessible they are. Audiogames.net has compiled a list of games available and accessible to those with visual impairments, primarily in the field of the audio game genre. Being started by two lecturers, Richard van Tol and Sander Huiberts to provide an online information point for audio games. Companies like these are comparatively few and far between whereas companies that do not put accessibility at the forefront of their development model are far greater in number.

Development of audio games as a genre is not inherently more difficult than the development of any other genre of games. It does, however, require a paradigm shift. Many companies put cutting-edge graphical technology as their primary concern allow audio to be an afterthought. The required paradigm shift would put the audio development of a game to the forefront and push visuals to a more secondary consideration by comparison. Which up to this point in the mainstream games industry has not occurred.

### 3.4 Ethical Considerations

Participants who are over the age of 18 are required to take part in this project. Since this development and subsequent experimentation will require human participation, an ethics approval form will be required to be filled out, signed by the project supervisor and subsequently sent off to the ethical approval committee. Receiving ethical approval as early as possible will be essential for the ongoing success of this project.

Autonomy – Any participants who wish to leave at any time during the experiment are welcome to do so. All participants will be advised exactly what this research will be used for and what is required from them at time of request for participation. No one will be persuaded or coerced by researchers or outside influences during any meetings. All participants will be notified that they have the option to leave at any time. All who take part will be anonymous and be advised as such.

Confidentiality – All information relating to participants will be kept anonymous and will not be released. Information will exclusively be used for the project and will be destroyed at completion of project.

Special Needs – All special needs will be taken into account for any participants who require any additional support, such as those who have visual impairment requiring a researcher to guide them to the project area. Participants will be made aware that they have the option for this additional support.

Protected Vulnerable Groups – To ensure there is no conflict regarding protected vulnerable groups disclosure no one who is in receipt of any of the following will be asked to take part in this test; Registered Care Services, Community Care Services, Health and Welfare Services. To determine this, participants will be asked in advance and any who receive these services or refuse to answer will not be considered eligible for the test.

# 3.0 Literature Review

## 3.1 Genre

To determine what game should be developed for this project genre must be considered. The aspects of this which will be kept in mind are; ease of development, ease of portability, popularity and level of accessibility. To determine this, reviews will be read and the genre of which appears most frequently with highest ratings in terms of accessibility and developer familiarity with the genre in question. The reviews will be pulled from <http://game-accessibility.com>, <https://www.gamespot.com/reviews/>, <http://blog.aidis.org/>, <http://reviews.sightlesskombat.com/>, as these sites all provide reputable reviews. Several of the sites; game-accessibility.com, sightlesskombat.com and aidis.org all provide accessibility-based reviews with sightlesskombat.com being run by a blind competitive gamer.

In terms of genre popularity, shooting games have the highest rate of sales in the United States as of 2016 with 27.5% of all sales (Statista, 2018). The closest second, still according to Statista.com, was the vague genre of Action with 22.5% of all sales. The remaining 50% is split up as follows; Role-playing 12.9%, Sport games 11.7%, Adventure 7.8%, Fighting 5.8%, Strategy 4.3%, Racing 3.3% and Other with 4.1%. This would include all other genres of games not noted.

In terms of genre all are largely equally portable depending on which games engine is chosen. Unity 3D offers porting to a wide variety of systems including smart-phone and console as does Unreal Engine 4. The games development should however focus on a single system and then once fully developed porting should take place depending on control scheme and layout to allow access to more players.

### 3.1.1 A Blind Legend

A Blind Legend is an audio only Role-Playing Game in which you play as a blind knight on a quest to save his kidnapped wife. Mechanically the game uses his daughter, Louise, to lead the player through the game. To move through the game the touchscreen is used similarly to a joystick to control the player and direct them towards certain audio queues. The current health of the character is conveyed through to the player through the use of a consistent heartbeat sound which increases in pace as the player’s health reduces. The game uses gesture control to allow players to attack and defend against enemies and events. These gestures also have control over the ways the player perceives the game.

The sound design of the game is one of the primary driving features of it, as would be expected with it being an audio-only game. The audio itself has a strong link to the mechanics of the game and are used to enhance the users experience. For example, when the player raises their shield with a “pinch” gesture on the screen the audio becomes muffled in the headphones making it slightly more difficult to hear opponents. This does however, reduce the damage the player takes from context-based events such as attacks or falling rocks. Combat is managed through sound as well, with audio prompts notifying the player when they are about to be attacked to give them time to raise their shield or attack in advance of the opponent.

The reviews for this game are overall positive with it receiving a four point four out of five on google play (Google LLC, 2018) and a seven out of ten on steam (Valve Corporation, 2018), many of these reviews praise the game on accessibility and it is for this reason that mechanics from this game may be used in the development of this project. However, it is unlikely that the game developed will be a role-playing game as the genre offers comparatively minor market penetration whilst requiring a larger development team, due to the need for a story and/or script writer, a designer, a programmer and (under the assumption of a visual game) at least one artist. This would turn this project from a solo project to a group project increasing development time and requiring reliance on the work ethic of others.

### 3.1.2 Killer Instinct

Killer instinct is a popular fighting game featuring in many tournaments across the world. Most notably for this report, Killer instinct is an example of a game and development team who altered the game based on the feedback from a visually impaired user. This allowed the user, Ben (who goes by Sightless Kombat) to play the game competitively online and win regularly, reaching the “Killer” rank in January 2016 (Yin-Poole, 2016). This is part of a growing trend in development of games which shows developers listening to players’ calls for inclusivity.

The main draw behind the game for Ben was the fact that the music and sound design we interwoven so well into the game as compared to its competitors. The developers of the game have admitted that the sound design for the game is a primary factor in the development of Killer Instinct, being one of the things that makes it unique in the fighting game genre (Killer Instinct, 2015). The game uses a HUD volume slider to provide the player with additional audio cues which have been described as “crucial to playing the game without sight” (Yin-Poole, 2016). These cues include things such as the Knockdown Value Meter (a UI representation of a system put in place to prevent people from being able to execute infinite combos) and character specific prompts when the character “Cinder” uses a special move to set an opponent’s limbs on fire. The cues differ depending on whether it is the characters arms or legs that are currently aflame.

These features are lauded by the community as powerful examples of inclusivity and an interesting feature within the game. Players have attempted to play blindfolded and been able to due to the time taken to add this mechanic. The fact that the audio design was integral to the games development allowed quick implementation of these aspects of the game when concerns were voiced on the official forums. For this reason, should a fighting game be developed, audio cues will be implemented to allow players without sight to quickly determine incoming attacks, blocks and/or combos, should they be implemented.

### 3.1.3 Blindscape

Blindscape is another audio only game. This one is a story-based experience in which the protagonist commits crimes against the state and for this he has his sight taken from him. The game is played through the protagonist telling the players his story and his intentions as he goes. This provides an effective tutorial for the game. The narrator advises the player of his intention to leave the room, advising that he is in front of his door and this is the cue for the player to blindly tap the screen in search of the doorknob. Similar story prompts are provided throughout the game to complete minor puzzles, such as walking towards dripping pipe, opening a sewer hatch or pushing a small boat onto a river.

The game can be completed within 15 minutes and has no start menu, allowing the player to go straight into the experience with ease. The game explains the need for headphones by reading a pair of quotes played through the left and right ear. The game then goes straight into the story. The lack of a menu provides quick and easy entry into the game and allows player to experience it quickly.

The website game-accessibility.com rates the game highly on accessibility for those who are blind as it is an audio game. While this type of game would be fairly easy to develop, requiring only a storywriter and a programmer, it would not offer a significantly enough programming challenge as would be required by this project.

### 3.1.4 Stealthfly

Stealthfly is an audio only racing game. In this game you play a high-speed plane which has to avoid obstacles without the use of sight. This game is an excellent example of a normally heavily sight based game genre which has been cleverly adapted to allow users without sight to successfully play it.

The game uses different pitches of audio to provide the player with information on the location of obstacles. When the sound is higher the obstacles are on the right, when the sound is lower they are on the left (Game-Accessibility, 2018). The game allows players to play alone in an endless runner style or play against other players in a racing format.

This is a type of game which would make an excellent potential project which would fall within the scope of this project. Endless runner style racing games are a common quick programming challenge which are easy to develop but often require a variety of other features to flesh out the game. This has also shown that this genre of game can be comfortably adapted to allow easy use by those suffering from sight loss.

Based on research done the game to be developed will be of the fighting game genre which employs design techniques previously mentioned in audio games. The genre has a high market penetration by comparison to the short story style and endless runner style games, both of which would be categorised under “Other” in the previously mentioned Statista.com genre breakdown of sales in the United States. This in addition to the fact that the developer of this project has experience programming similar games and as such will be able to program and test the game quickly and efficiently.

## 3.2 Development engine

The engine for the development of this project must be carefully considered. Its adaptability portability must be considered prior to development to allow for maximum outreach. Language and peripheral support must be taken into account to ensure development continues with ease without the necessity of attempting to develop a new programming language or learn an entirely new syntax. Experience with the engine and knowledge of its inner workings should be considered to cut down on development time and allow a greater period for experimentation and evaluation of results. The available features of the chosen engine must be considered for the development of the project and, in the instance that a feature is missing, development time and importance level for that feature must be considered. Furthermore, accessibility of documentation is needed for error handling, should any occur. To this effect the popular development engines Unity 3D and Unreal Engine 4 will be reviewed for possible use.

### 3.2.1 Unity 3D

Unity 3D is a development engine designed for portability, accessibility, and ease of use. It supports the languages C# and JavaScript with the ability to write plugins to be read by the engine in C++. The core of unity itself is written in C++ with certain aspects of the editor being written in C#. This allows the engine to be very portable and gives it a shorter compile time.

Of the supported languages C# allows function overloading, JavaScript does not however which would reduce the customisability of the software to a small degree, however the requirement of function overloading is not a complete necessity as instead of using the process of function overloading a simple case of multiple function names can be used (Rongala, 2017). Unity 3D is also a particularly portable development environment allowing development for; virtual reality, mobile (Android and iOS), console, PC, MacOS, Linux, tvOS, WebGL, Samsung TV and PS Vita. This portability can provide a great deal of outreach to a variety of commercial devices, allowing further development into a commercial marketplace in the instance this project proves successful.

Unity 3D offers a standard set of classes for the development of games. These classes allow a simple and easy starting point for the development of game software. This development environment also provides a series of tutorials for all technology used and there is a large community of user made tutorials available online which allow the creation of a variety of games with minimal effort. These tutorials can act as a structure for the creation of software. However, this resource must not be overused to avoid any issues with plagiarism in the development of this game in an academic context.

Unity 3D provides 2 integrated physics engines, one for 2D physics and another for 3D physics. They both make use of rigidbody components (labelled as “Rigidbody 2D” in the case of 2D physics). The primary difference between the two physics engines is largely the addition of the extra dimension which allows the use of the Z axis (Technologies, 2017). This provides more design opportunities for the development of the project with the opportunity to have a 3D or 2D game. This opportunity must also be carefully considered for the development of the project in the instance Unity 3D is the chosen development engine.

In addition to an integrated physics engine Unity 3D offers the ability to provide 3D audio to help allow those without the use of sight identify the location of objects that create sound, reducing volume at distance and increasing as the player character gets closer to the source of the sound.

Unity 3D also provides a simple and easy to use analytics system which will allow recording of in-game analytic metrics such as play session duration and the level at which the player quit. These analytics work through an events system integrated into unity. Examples of this can be found in games such as the Android game Bright Void (Hesketh and Campbell, 2017).

### 3.2.2 Unreal Engine 4

Unreal Engine 4 is the primary game engine for the game development and publishing company, Epic Games. Known for games such as; Unreal Tournament, the Gears of War series, Bulletstorm and Paragon. All developed in the unreal engine, these games show an intuitive use of physics, lighting and mechanical techniques which the unreal engine provides. The engine itself is written in C++ as with Unity 3D allowing it to also have a high degree of portability and reduced compile time.

Unreal supports C++ as a development language as the basis of blueprint classes. The blueprint system is a visual scripting system to allow classes to be created in an innovative manner. This system uses visual blocks of code to show functions and references to variables (Games, 2017). This development method makes it easier for those in non-programming teams to create software for specific features within the game. This, however, is not necessarily a benefit in this project as this is development project is a programmer-specific project. Meaning the programming itself requires less simplification than in the case where designers would be required for the creation of features.

Unreal Engine 4 Also provides a variety of post-processing effects and a large animation toolset. However, due to the nature of the shortened development time and the simplicity of the visuals, these features become largely irrelevant and the learning curve on these features from a starting point of zero experience with them in the engine prohibit their use.

This Engine does, however, offer integrated support for Virtual Reality (VR) and Augmented Reality (AR) technologies. This makes the development of these applications quicker and easier than in most other engines. Since there are multiple virtual reality platforms under consideration for this development project, this integration has the potential to be greatly beneficial to the project’s development cycle in reducing the time required to integrate the VR aspect should that technology be chosen.

While this suite of development tools offers a large variety of high-end features, unfortunately, based on the feature list available (Games, 2017) there is no inbuilt analytics system at the time of writing. There is also a concern that the number of features available from Unreal 4 would be irrelevant and would clutter the development process. The lack of an inbuilt analytics recording system means that several metrics would need to be recorded manually, slowing development and experimentation process. For these reasons, coupled with the lesser amount of experience with the Unreal 4 Engine, the development project will use Unity 3D for the development and evaluation of this project. It is hoped that this will be a quicker and more effective development process as a result of this choice.

## 3.3 Development Platform

The development platform of this program must be carefully considered for a variety of reasons. This project must take into account the accessibility of the chosen device to ensure those suffering from sight loss can make use of the product. The cost and market popularity must be considered as a factor in the game’s ability to reach a target demographic. The ease of development of the project for the chosen platform and availability of documentation and reference materials must also be considered in order to ensure the pace of the development project continues briskly and without issue.

### 3.3.1 Virtual Reality

With the tremendous variety of virtual reality devices on the market, the choice between them can be a daunting task for many consumers, especially for those who do not necessarily benefit from the full effect of them. For that reason, this report will narrow the field of development devices by taking into account only the following virtual reality systems; HTC Vive, Oculus Touch, and the Google Daydream.

The HTC Vive offers the largest scale virtual reality experience with at the highest cost of the reviewed devices with a £599 price point (HTC Corporation, 2017). This device is also the most complicated device in terms of initial setup. The device comes equipped with a pair of base stations which HTC recommend be affixed to a wall. The Headset itself feeds into a “breakout box” via HDMI, USB 3.0 and 3.5mm headphone jack to provide audio to the headphones connected to the headset. Then comes a lengthy setup procedure involving synchronising the base stations and the software setup for either room scale or standing only (HTC Corporation, 2016). Comparatively, the Oculus Touch has a much simpler setup process. Two sensors and a headset are plugged into the computer via USB 3.0 cable, and then a quick software setup is done using the Oculus Software download. This is a much quicker and easier process however only offers room scale VR on an experimental basis with standing VR being the more commonly used on this system. This allows the user to more quickly and easily pick up and play with the Oculus Touch compared to the HTC Vive. However, of all the reviewed devices here, the Google Daydream has the simplest setup procedure. Dues to its requirement to be used with the Google Pixel mobile phone as opposed to a desktop setup this device simply requires the user to open the daydream app on their phone and follow the instructions to pair their device to the headset then insert the device into the headset. This makes it the simplest and easiest virtual reality device to set up, as there is no lengthy attachment procedure to attach a computer to the device and sensors to pair. This does, however, come with the downside that the Daydream offers no body tracking whatsoever meaning that design decisions within the project would be compromised.

The HTC Vive is often considered the top end of virtual reality hardware being the best-selling VR device of those reviewed, having sold 420,000 devices as of March 2017 (Ergürel, 2017). This would suggest that the Vive would have the greatest market share of VR devices as the Oculus Touch had only sold 243,000 devices and the Daydream having sold an estimated 260,000 headsets as of the final quarter of 2016 (Ergürel, 2017). This data would imply that to reach the greatest number of users within the general population the Vive should be developed for. This is reinforced in the partially sighted community by academic experiments previously done. Larger scale VR devices are generally the chosen development devices for those working with the blind as it allows easier development of cognitive maps of an area by those who do suffer from blindness (Merabet et al., 2012). Of the reviewed devices technically speaking the Google Daydream offers the largest scale VR as it is not confined to an area by sensors as with the Vive and Oculus. However, this benefit is offset by the lack of any form of body tracking making following the player more difficult within context.

The development cycle for a virtual reality game is largely the same as that of a conventional development project. In the instance of these three devices, the development for an Oculus and HTC Vive are largely the same with the chosen engine (Unity 3D) offering VR support in its latest incarnation. Allowing virtual reality support is as simple as checking the “VR Supported” box within the Unity editor and ensuring that the editor is set to build for PC, MacOS or Linux. In the case of the Google Daydream VR device, development is slightly more complicated despite still being integrated within Unity’s editor (Technologies, 2017) as it involves the integration of the Android SDK and an additional layer of debugging if there are any build errors involved.

Should the development of this project use a virtual reality device, then the chosen device for this purpose shall be the HTC Vive. With its high market share and ease of development within Unity in addition to the increased scale of the available play area, the Vive becomes the most appropriate device. The use of the Vive’s body tracking system should allow greater ease of design and the available peripheral integration should provide suitable complexity for the development portion of the project.

### 3.3.2 Computer

It is no secret that the section of the video gaming industry dedicated to computer games is a popular one, with its advocates fiercely defending the platform. There are three main competitors within the computer market in terms of operating system, those are; Windows, MacOS and Linux based operating systems. In addition to previous review criteria, reviewed computers and technical requirements shall consider requirements for the technologies previously reviewed and operating systems available for development on the Unity 3D engine. Luckily in this instance, Unity3D offers build options for PC, MacOS and Linux provided the developer has appropriate licences.

The accessibility of a desktop computer is second to none when it comes to allowing those with limited access to technology. The incredible variety of peripherals available for the computer user allows almost anyone to play games on a computer. With Apple’s MacOS computers being more geared towards proprietary software, the access to these kinds of peripheral devices are slightly more limited to on this operating system. However, in the instance of a windows operating system device most peripherals are quickly available. The same is true of many Linux based operating systems due to the open source nature of Linux. This means that if a peripheral is developed with drivers specifically for Windows or MacOS then there will often be a Linux equivalent available for download. In the instance that there are not drivers readily available one of the Windows or MacOS emulators such as ‘Wine’ can often used to properly run the required software for the device. In the case of this development project, the only required peripheral in the case of a computer-based game would be; a pair of headphones, a mouse, a keyboard and a monitor. Since the project targets those with visual impairment, a braille keyboard may be necessary. However, this would not be for the development project itself but instead would be for the subject of the experiment to more comfortably use the computer. All named devices are easily available for all reviewed operating systems and as such provides no weighting to the development of the project.

Linux operating systems are, due once again to their open source nature, free to download and use. This means that the cost of a Linux based computer is exclusively on the hardware cost. This, in turn, allows more people to more easily afford a computer. It may seem that this reduction in cost would mean that Linux as an operating system would have the highest market share among gamers. However, this is not necessarily the case as shown by the Steam Hardware and Software Survey (Valve Corporation, 2017). This monthly survey shows that as of October 2017 Linux based software made up only 0.35% of their user base, it is suspected that this is due to the lack of support from developers for Linux (Lyer, 2017). This is found similarly in MacOS based operating systems, again shown in the Steam Hardware and Software Survey (Valve Corporation, 2017) in this instance MacOS barely beats out Linux with only a 1.60% usage rate from steam users, the most popular of which being free after Apple has removed upgrade costs. The clear most popular operating system according to this survey is windows with 98.04% of Steam users using this operating system, for maximum market penetration the project would be developed to run on Windows 7 as 63.60% of Windows users on steam are running this operating system. While this is no longer available for purchase from Microsoft directly the most recent version of windows is available from the Microsoft website at £119.19 (Microsoft Corporation, 2017).

The availability of documentation for the development of the project is largely the same for each operating system due to the choice of the Unity 3D as a development engine. The prevalence of documentation for Unity 3D which supports all three operating systems means that each operating system can be developed for with ease in this instance. However, to develop for MacOS a Mac computer is required along with a licence which has an annual cost of $99 making it less favourable to develop for (Apple Inc, 2018).

Overall due to ease of development and higher market penetration this project will be developed for Windows. Since all reviewed operating systems are equally easy and available for development when using Unity 3D, market penetration becomes the only metric to bias the decision of which to use. As such Windows becomes the obvious development decision.

### 3.3.3 Mobile

Mobile games have been a rapidly emerging market within the games industry. Quickly becoming one of the most used devices for gaming, 42% of all game revenue has been made due to mobile (McDonald, 2017). For this reason, mobile must be considered for the development of this project. The two most popular mobile operating systems for the development of games are iOS and Android; developed by Apple and Google respectively. These operating systems will be reviewed as per the stated criteria to determine the most appropriate system for the development of this project should mobile be chosen as a development platform.

Both Android and iOS do both offer equal levels of hardware accessibility being operating systems for very similar types of device. Should development for this project take place using a smartphone or tablet device then certain hardware features should be considered. With the knowledge that an estimated 97% of smartphones make use of touchscreen technology (Allied Business Intelligence, 2011) non-conventional methods of control must be considered, as opposed to the traditional keyboard and mouse or controller. The development of this project, should it be done on mobile devices, must take into account the fact that those who are unable to see a touch screen may be unable to use a touchscreen. In this instance, the primary method of control would likely be the accelerometer or audio-explained touch prompts. This would allow those without site to more accurately use the device.

Android has the greatest market penetration of all mobile operating systems as of the second quarter of 2016 according to Gartner (Gartner, 2016). This report states that of all devices currently sold 86.2% have been Android devices. iOS is far from this with a 12.9% market share according to the same report. This shows that should the project be developed with Android in mind a much larger market demographic would be feasible than in the case of iOS.

With the development of this project taking place within the Unity 3D engine, ease of development has become much less of a concern for each device. Since Unity provides support for both reviewed systems, the development of either mobile operating system would be equally simple. However, once again, should the device be developed for the Apple-based product an annual 99$ licence fee would be required (Apple Inc, 2018). For this reason, should the project be developed for a mobile device, the Android operating system would be the targeted platform.

Based on all reviewed platforms and devices this project will be developed targeting the Android smart-phone**.** The reason for this is that, as previously shown, mobile phone games have surged in popularity in recent years with 42% of game revenue being made by the mobile platform. This coupled with the popularity of touch screen controls for those with sight loss as shown in the Genre section of this report would suggest that mobile is the best platform for development. The ability to target a larger number of users through development for android devices has also pushed the development towards android over iOS devices.

# 4.0 Execution

The purpose of this project is to encourage and exemplify the development of games which provide accessibility to those both with and without sight. The development of this project has been the primary focus of this report and its design will be explained in this section provide a full understanding of the development solution. The game will be designed with accessibility and ease of use in mind to ensure that any who wish to, sighted or non-sighted will be able to pick up and play it.

* Clearly **discuss**  original problem?
* Clearly **analyse** the features of the problem?

## 4.1 Design Solution

* Clearly **relate** those features to the literature review’s conclusions

The game will require careful design and development processes to ensure that the project will successfully answers the research question. The design process, as explained in this report, will take note of; gameplay inspirations, Genre, Intended Control Device, Intended Control Scheme and an overview of the game itself. The game overview will also explain how specific mechanics are used with the goal of accessibility. Provided this design is followed, the development section will explain how each mechanic and feature was implemented within the Unity Development environment. This will also ensure to make use of features specific to the mobile phone.

* **Indicate** where they support and or disagree with literature review’s conclusions

### 4.1.1 Game Design Inspirations

This game will be developed with a focus on simplicity and accessibility, pulling inspiration from reviewed games and building upon their mechanics. The hope is that by pulling the effective workings from the games reviewed the overall accessibility of the developed project will itself be higher. The game will not however be a facsimile of any of the reviewed games, in part to ensure there is no issues regarding plagiarism and partly to preserve the creative integrity of the project.

Killer Instinct – As noted in the previous section, Killer instinct utilises its HUD volume slider to provide audio cues to player which aids in playing the game without sight. This is an excellent feature and the development project will use a similar one to ensure that users without sight can comfortably tell when an attack is coming in or when one has been blocked or successfully hit.

Blind Legend – This game used a novel method of determining the health values of the player and the location of the opponent. The game used the noise of a heart rate to determine how much health the player has available, increasing in pace as the player loses health. This is a helpful method of showing players how much health they currently have as the heart rate is a ubiquitous method of communication. Often found in forms of media to show when a character is in danger, this will be considered for a method of showing player health. However, the specifics must be carefully considered as both players must be able to determine their own health without being distracted by the heartbeat of their opponents.

Kung Fury: Streets of Rage – Based on the short film “Kung Fury”, directed by David Sandberg, “Kung Fury: Streets of Rage” is a simplistic two button brawler in the vein of “One Finger Death Punch”. Waves of enemies attack from the left and right sides of the screen. If an enemy is within range of the player and the player presses the attack button corresponding to the side the opponent is on, the player will strike the opponent. This simple mechanic allows players to quickly and easily understand the core gameplay whilst not diluting enjoyment of the game. Taking inspiration from this, the developed game will use two button fighting controls with the aim of ensuring that players quickly and easily pick up the mechanics of the game. The controls used will not be the same as those of “Kung Fury: Streets of Rage” which holds exclusively attacking controls, as this will be a competitive game the controls will hold attack and block to allow you to reduce the damage coming in from an opposing player.

* **Explain** your ‘solution’, its approach, design & implementation/instrument

### 4.1.2 Game Genre and Control Scheme

The game to be developed for this project shall be a fighting game. In order to ensure fast paced gameplay and to reduce the likelihood of an inability to test, the game will make use of a local multiplayer environment as opposed to an online one. This in turn would mean that only one device is required per testing pair and allow a quicker turnover of results with better supervision.

Due to the use of a single device certain methods of control can be immediately discounted, namely; tilt control, voice control and camera-based control. These methods would not be feasible for use within a multiplayer game where both players are using a single phone. Considered control methods for this game are third party peripherals (such as the MOGA android mobile gaming system) and touch screen control. However, over the course of this project the focus has been on accessibility, with development taking into account ease of development for the project. Unfortunately, many peripherals require lengthy setup procedures involving the downloading of apps or the pairing of Bluetooth control devices. Due to this additional layer of abstraction between the player and the game, third party peripherals will no be considered as a control device. This leaves the only logical method of control to be the touch screen, not only because of its ease of use for development but also because almost all smartphone devices have access to a touchscreen (Allied Business Intelligence, 2011).

**Justify** your ‘solution’, its approach, design & implementation/instrument

### 4.1.3 Design Overview

* **Present** your ‘solution’, its approach, design & implementation/instrument

As is a staple in Audio Game development, the game will require earphones. However, it will not make use of them in a traditional manner. When playing the game, each player will wear one of the two earbuds. On start-up the game will provide a tutorial through either headphone. This tutorial will explain where on the screen the player needs to tap to attack or block. This will be done using stereo audio, the player with the right earphone in will only hear the information relating to their attack or block. The player with the left will hear the same information but relating to their own attack and block. The use of the headphones in this way negates the previously mentioned design issue regarding the use of audio to present health. The players will only be aware of their own health through the noise of a heartbeat which will steadily increase in pace and pitch as the player takes more damage.

For the game itself, spatial audio is used to allow players to be informed when if or when they are being attacked. Players will receive a prompt when they are being attacked in order to notify them of the opponent’s intent. However, players are not able to block when attacking and are not able to attack when blocking. The desire here, in terms of design, is that players will attempt to gauge based off their own health (communicated through a heartbeat) whether it will be more beneficial to attack and take a higher amount of damage or block and take less damage but risk losing their opening.

Players will not be made aware when their opponent is blocking until the opponent takes the damage from their attack. They will hear one sound corresponding to the opponent taking full damage and a different sound when a portion of the damage is blocked. When a player attacks they will hear a lower volume version of the attack sound, this is both a consequence of the spatial audio and to ensure that players will receive feedback. This will ensure that they are aware their attack has been registered.

The players will go through 3 rounds. The winner of the game will be determined at the end of these rounds. The player who has won more rounds will be considered the winner of the game and will hear audio in their earphone advising them of this and the player who has lost will hear audio advising them that they have lost.

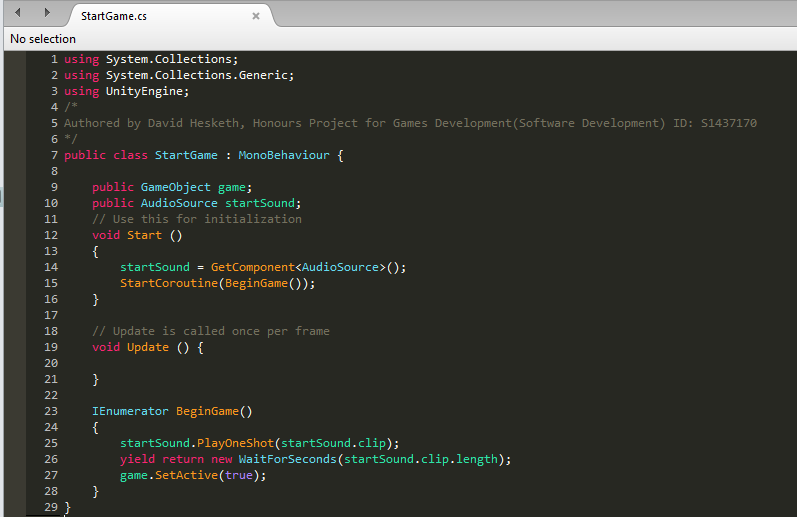
## 4.2 Development Process

This section of the report will be written in the past tense as it reviews the development of the project from point of view of its completion. The game itself has been developed according to the design brief above and will be reviewed with screenshots of code and explanations of the purpose of each function, coroutine and variable. All code used for the creation of this project and the project folder itself can be found here:

The following classes have been used through the creation of this project and will be listed in the order the players will encounter them; “StartGame”, “HeartRate”, “PlayerScript” and “RoundScript”. With the focus of development being on simplicity, the majority of the work is done in the “PlayerScript” class.

### 4.2.1 StartGame

The “StartGame” class is used to handle the processes required for the game prior to the player’s interaction. The class itself is a very simple one.



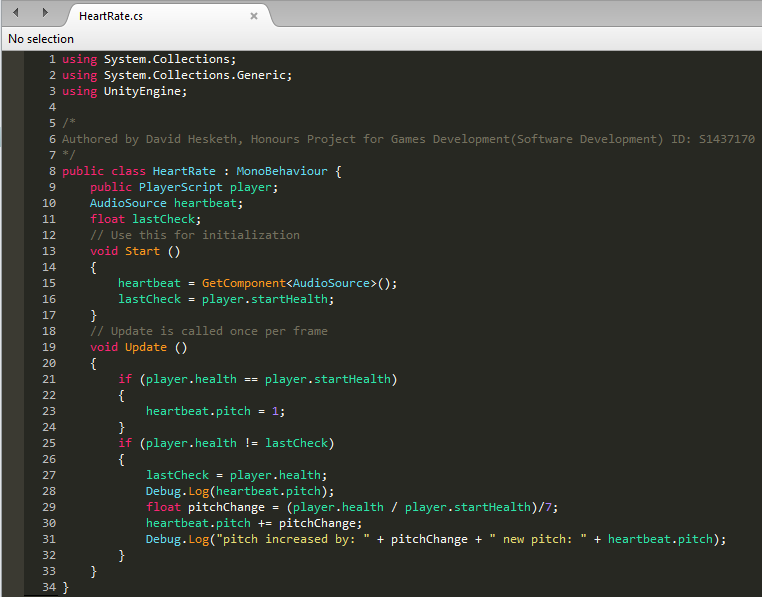
*Fig 1. Full StartGame Class Code*

The class holds two member variables. The first is a public “GameObject” identified as game. This “GameObject” holds the parent of the interactive elements within the scene. These are by default set to inactive. This ensures that players are unable to attack each other prior to being presented with the correct controls. The second variable is an “AudioSource”, identified as “startSound”. This holds the tutorial as its “AudioClip”. The reason this is not assigned in code is because the only clip used by this “AudioSource” is the tutorial clip.

In the “Start()” function the “AudioSource” is assigned as the attached component of that type and a “StartCoroutine()” is called to begin the tutorial. The coroutine then plays the tutorial audio and waits until the end of the clip to allow the players to begin the round. The player then hears the first heartbeat to indicate their health before hearing, a split-second later, the alert notifying them the first round is starting.

### 4.2.2 HeartRate

The “HeartRate” class manages the player’s perception of their level of health. It is another class which has very simple, easy to process code, with minimal interaction with other scripts.



*Fig 2. Full HeartRate Class Code*

This class holds three member variables. The first is a public “PlayerScript” identified as player. This holds the player whose health is being used as a reference. This variable is only used to pull the health value from so that the player is aware of their current level of health. The second is the attached “AudioSource” identified as “heartbeat”. This “AudioSource” holds the “HeartBeat” as its “AudioClip”. The final variable is a float called “lastCheck”. This variable is used to check if the players health has changed since the last frame.

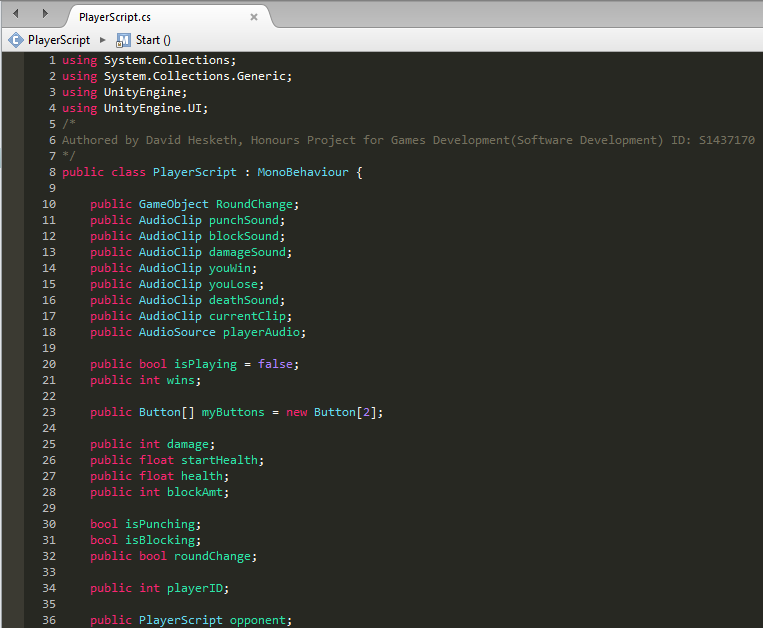
In the “Start()” function the “AudioSource” is set to the attached components and “lastCheck” is set to equal the player’s starting health. This is all the setup required initially.

The “Update()” method is called each frame and first checks if the player is still at full health. If they are, it sets the current pitch of the “heartbeat” to one, this is the base level or normal heart rate for the player to hear and shows them that they are at full health. This is implemented to reset the pitch when a new round starts. The code then checks if the player’s current health is different than that last time it changed. If it is, “lastCheck” is set to the players current health, a local variable is created called “pitchChange” which is set to equal the percentage difference between the current health and the starting health of the player divided by seven. The pitch variable contained within “hearbeat” is then altered by the amount denoted by “pitchChange”. This also serves to increase the speed of the “AudioClip” within “heartbeat”.

heartbeat has its “AudioClip” set in the Unity3D editor as well as its stereo pan which controls which ear the audio plays through. The clip is also set to loop in editor and this is why there is no code showing that process in Fig 2.

### 4.2.3 PlayerScript

The “PlayerScript” class handles the majority of the interactions between players. The player object itself is a “GameObject” which acts as a parent object for two buttons and the object holding the “HeartRate” script. Due to the fact that this class is much larger and more complex than the two preceding it, screenshots of the code in this class will be split up to show and subsequently explain relevant information



*Fig 3. PlayerScript Variable Declaration*

The variables used in the “PlayerScript” class are all required to ensure that the game properly functions. Each of these variables are member variables declared exclusively as needed to ensure minimal unnecessary code. Variables will be grouped by type and explained to ensure full understanding of the code.

GameObjects – RoundChange is the only GameObject used in this class. It is declared as public and assigned in the editor. This GameObject monitors the players’ health and starts the next round or ends the game when necessary.

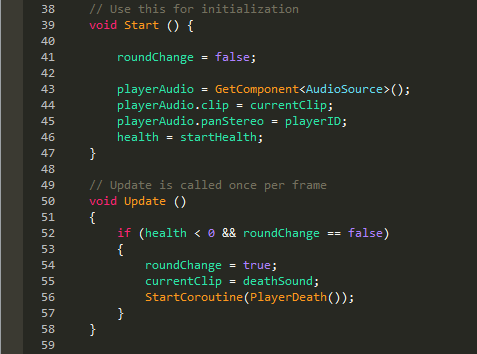
AudioClips – Each of these relate to a different sound used in the combat of the game. “punchSound”, “blockSound” and “damageSound” are used when attacking blocking damage and taking full damage respectively. “youWin” and “youLose” are played through the opponent’s earphone when they have won or lost. “deathSound” is played when the player’s health goes below zero. “currentClip” is whichever clip is going to be played and refers to the current active clip in the class.

Booleans – The Booleans (shortened in code to “bool”) used in this class are used to prevent certain actions taking place when a player is doing a specific action. “isPlaying” is used in the “RoundScript” to ensure that the player’s win or lose sound is not played more than once when the final round ends. “isPunching” is used to ensure that the player is unable to block whilst the player is using the attack button. “isBlocking” is used when the player takes damage to determine if the incoming damage is partially mitigated or not. “roundChange” is used when the player’s health goes below zero to ensure that the coroutine responsible for “deathSound” is not called more than once.

Integers – The integers (shortened to “int” in code) are used for player specific values, that are not divided by any other value at any point in class. “wins” is used to denote how many times the player has won a round. “damage” is used to show how much damage the player deals and to reduce the opponent’s health by this value. “blockAmt” is the amount that the player reduces incoming damage by. “playerID” is used to determine which player this script belongs to and by extension which earphone the sound will be played through, because of this the value is always either negative one or one.

Floats – These variables each relate to the current and starting health of the player. “startHealth” is used in the changing of the pitch of the players attached “HeartBeat” object to determine the amount to increase the pace and pitch of the sound by. “health” is the players current health and is used throughout the game to show when the player has died and by how much to change the pace and pitch of the player’s “HeartBeat” sound.

PlayerScripts – The only use of another “PlayerScript” in this class is the “PlayerScript” attached to the opponent. This is referenced when the player attacks the opponent to notify them that they are being attacked.



*Fig 4. PlayerScript Inherited Functions*

The functions used in the “PlayerScript” class handle how the class interacts with the game state. The standard classes, inherited from “MonoBehaviour” are “Start()” and “Update()”. These functions are called on initialisation and during each frame respectively.

When the players are set active the “Start()” function sets the default values for variables that will require it. Specifically; setting the “roundChange” value to false, the “playerAudio” to equal the attached “AudioSource” component and assigning its relevant variables as seen above. The fact that “playerAudio.panStereo” equals the value of “playerID” is the reason for “playerID” only ever equalling negative or positive one. This is because when “panStereo” is set to negative one it plays the audio through the left ear and when it is set to positive one it plays through the right ear. The reason these values are assigned in the “Start()” function is to ensure that they are not checked in the “Update()” prior to being assigned.

The “Update()” function is called each frame and in the “PlayerScript” class is used to check if the player’s “health” variable is lower than zero. “roundChange” is used to check if the if statement has been entered already. If the statement has yet to be entered “roundChange” is set to true and the “currentClip” is set to “deathSound” and then the player’s death coroutine is called.



*Fig 5. PlayerScript Non-Inherited Functions*

The non-inherited functions handle the combat of the game. These functions are; “StartPunch()”, “Blocking()”, “EndBlock()” and “BeingPunched(int dmg)”. Each of these largely work as the names would suggest.

“StartPunch()” is called when the player taps on the section of the screen to make their attack. It starts the “Punch()” coroutine, which will be thoroughly explained after the screenshot showing the coroutine information.

“Blocking()” is called when the players finger touches the section of the screen, it sets “isBlocking” to true. Its counterpart, “EndBlock”, is called when the player’s finger leaves that section of the screen.

“BeingPunched(int dmg)” is called by the opponent to show that the player is being attacked. This function takes in an integer as a damage value. It then declares a float as “healthChange”, this is used to determine how much the player’s current health value is changed by. If the player’s “isBlocking” boolean is true, then the “currentClip” is set to “blockSound” and health change is set to equal the incoming damage minus the damage reduced by the block. If the player’s “isBlocking” boolean is false, then “currentClip” is set to “damageSound” and “healthchange” is set to equal the incoming damage value without any reductions. The player then is forced to stop blocking and the health is reduced by the relevant amount. The relevant audio is then played to inform the opponent if they have hit the player for full damage or blocked damage.



*Fig 6. PlayerScript Coroutines*

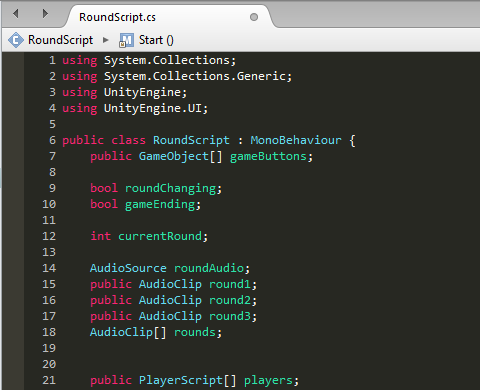
Coroutines handle the attacking and death of players. Whenever a player makes an attack, the “Punch()” coroutine is called by the “StartPunch()” function. When the players health hits zero the “PlayerDeath()” coroutine is called.

The “Punch()” coroutine is used to allow the players to attack. When called, it sets the previously mentioned “panStereo” variable to “playerID”. The coroutine then sets “isPunching” to true and sets “currentClip” to “punchSound”. Following this, the players buttons are then disabled to prevent the attacker being able to block and the relevant audo is played. This is to notify the opponent that they are about to take damage and give them the opportunity to block. The coroutine then waits for the length of “punchSound” to apply damage through a call to the opponent’s “BeingPunched(int dmg)” function. “isPunching” is then set to false and the buttons are turned back on to allow the player to react once again.

“PlayerDeath” is called in the frame where the players “health” variable is shown to be below 0, this coroutine plays the “currentClip” which has been set to “deathSound” by the “Update” function. The coroutine then waits for the duration of “currentClip”. This is a holdover from an earlier version of the combat system which had the players determining when to reset the game state, previously a function to do this was called after the “WaitForSeconds(float seconds)”. This caused several bugs and was eventually scrapped leaving the current architecture.

### 4.2.4 RoundScript

The “RoundScript” class handles the incrementing of the rounds in the game and ensure that the players are not able to attack each other before the round has started. The class also has control over the starting and ending of the game. This class will also be split into variables, functions and coroutines due to it requiring multiple screenshots to show all information.



*Fig 7. RoundScript Variable Declaration*

“RoundScript” uses a variety of variable types to manage the game. Each of these member variables are used in order to ensure the game can easily transition into, between and out of, each round. These variables will once again be grouped by type for ease of explanation.

GameObjects – A single array of GameObjects is declared and identified as “gameButtons”. This array holds all of the buttons used in the game by both players. It is used to set all of the buttons at the start of the game to active, this however occurs after the tutorial is listened to.

Booleans – These are both used to confirm if a process has already begun to occur to ensure that it does not get called more than once. In the case of “roundChanging”, this is used in “StartNewRound()” to ensure that the code contained within this function is not executed every frame that a player’s health is below zero and is instead only executed in the first frame. “gameEnding” is used to ensure that the win or loss sound for each player is only played once and the code within the coroutine which plays that sound is not called multiple times.

AudioSources – “roundAudio” is the only “AudioSource” declared in this class and is the “AudioSource” attached to the object holding this class as a component.

AudioClips – The AudioClips in this class each relate to the round to be announced. “round1”, “round2” and “round3” each relate to the rounds that their name suggests. The array of AudioClips identified as rounds is a three-position array which holds the different round AudioClips. This is referenced at the start of each round to play the relevant clip.

PlayerScripts – An array of PlayerScripts is declared and identified as “players”. This array holds both of the players and is used to compare health values for both players in the “Update()” function.



*Fig 8. RoundScript Inherited Functions*

As with all other classes shown, “RoundScript” inherits certain functionality from “MonoBehaviour”. Specifically, the “Start()” and “Update()” functions. Due to the fact that the object that holds this class is a child of the, “game”, “GameObject” found in the “StartGame” class, the “Start()” and “Update” functions are not called until the tutorial has been played.

The “Start()” function once again, sets the default values for relevant variables. It ensures that “rounds” and “roundAudio” are set correctly and the booleans are both set to false initially. The function also ensures all buttons are set to inactive and the coroutine to start the game is called.

In the “Update()” function, both players “health” are checked and if either are below zero, then the StartNewRound() function is called.



*Fig 9. RoundScript Non-Inherited Function*

The only non-inherited function used in “RoundScript” is the “StartNewRound()” function. This function, increments the current round and disables all buttons. If one player has more health than another, that player’s “wins” variable is incremented. If there has been less than three rounds, the class sets both player’s health to its starting value and calls the “StartRound()” coroutine. If three rounds have been played, the class checks to see which player has more wins and sets their win and loss audio respectively. It then checks if they player is playing their audio and if not, the audio is played and the “EndGame(float dur)” coroutine is called.



*Fig 10. RoundScript Coroutines*

Coroutines are used in the “RoundScript” class to start the first round, end the game and start each other round. The coroutines used to do this are “StartGame()”, “EndGame()” and “StartRound()” respectively.

“StartRound()” is called in the “StartNewRound()” function once the player’s health has been set to full. The coroutine plays the current round intro, waits for the audio to finish and then reactivates all of the buttons on the screen.

“StartGame()” is called in the “Start()” function and acts almost exactly the same as the “StartRound()” coroutine, however instead of playing whichever the current round intro is, this function only ever plays the first round intro.

“EndGame(float dur)” is called when the players have played through three rounds and have been advised of who won and who lost the round. The coroutine checks to see if the code has been executed and if it has not, waits for the players to hear their end game message and gives a ten second break before reloading the scene for the next game.

**Evaluation & Discussion approx 30% of word count**

### 2.3.1 Risk Assessment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Risk ID | Risk | Risk Severity | Risk Probability | Contingency Plan | Tools Used | Plan to Subvert Risk |
| 01 | Not enough participants to obtain meaningful data | High | Medium | Offer possible participants gifts such as pizza, fruit, juice etc. in order to encourage participation other students. | Participants, lecturers, email addresses | Contact lecturers and request that they send out emails requesting participants and invite students to participate |
| 02 | Game Not Developed Properly (excessively buggy/broken) | High | Low | Run consistent playtesting sessions prior to the experimental sessions in order to ensure bugs are quickly found and fixed. | Game Engine, Computer, Integrated development environment (IDE) | Risk cannot be fully subverted. Minimised through the full knowledge of the code and research of the techniques to be used coupled with time management. |
| 03 | No meaningful conclusions can be drawn from data gathered | High | Low | No contingency plan necessary as this would simply show invalid hypothesis | Computer with SPSS software installed | Successfully analyse data with SPSS |
| 04 | Participants drop out of experiment after agreeing to do so. | High | Low | Offer possible participants gifts such as pizza, fruit, juice etc. in order to encourage participation from other students. | Participants, lecturers, email addresses | Regularly contract prospective participants with reminders about dates and times scheduled |
| 05 | Game incomplete or unfinished | High | Low | None available as if the game is incomplete testing and further experimentation cannot be done | Game Engine, Computer, Integrated development environment (IDE) | Carefully plan stages and follow development timescale throughout the creation of the game. |

How can questions below be **demonstrated** in the report?

**Evaluate = attempt to form a judgement about, be specific about the basis for this judgement**

* What does the ‘solution’ its approach, design & implementation/instrument **contribute?** What are **key criticisms** of the solution its approach, design & implementation/instrument
* What are **the advantages/limitations** of the ‘solution’ its approach, design & implementation/instrument
* Do the results **identify** what is more or less important?
* Are the results **compared**  to other knowledge about the project?
* Is there comment on the **consequences** and **implications** of the results?

**Conclusions & Further Work approx 10% of word count**

* **Summarise** the main findings from the lit review, the execution, the evaluation & discussion chapters.
* **Explain** with evidence the final conclusions of your report. What do you want your reader to remember about the report?
* **Discuss** what else could take place to enhance, extend the project.
* Which direction could your project take in the future?

<http://www.rnib.org.uk/nb-online/video-computer-games-people-vision-impairment>

<http://www.rnib.org.uk/rnibconnect/technology/audiogames-and-playing-videogames-without-sight>

<http://game-accessibility.com/documentation/visually-impaired-gamers-where-to-go-what-to-play/>

<http://ifiction.org/>

<http://blog.aidis.org/audio-output-a-review-of-games-for-blindvi-gamers>

<http://www.eurogamer.net/articles/2016-03-29-meet-the-blind-gamer-with-a-killer-instinct>

Newzoo. (2016). *The Global Games Market 2016 | Per Region & Segment | Newzoo*. [online] Available at: https://newzoo.com/insights/articles/global-games-market-reaches-99-6-billion-2016-mobile-generating-37/ [Accessed 13 Oct. 2017].

Transversal, A. (2014). *How many people in the UK have sight loss? - RNIB - supporting blind and partially sighted people*. [online] Help.rnib.org.uk. Available at: https://help.rnib.org.uk/help/newly-diagnosed-registration/registering-sight-loss/statistics [Accessed 15 Oct. 2017].

Specsavers (2017). *The State of the Nation Eye Health 2017: A Year in Review*. [online] Royal National Institute of Blind People. Available at: https://www.specsavers-spectrum.com/wp-content/uploads/2017/09/RNIB-Specsavers-State-of-the-Nation-Report-2017.pdf [Accessed 15 Oct. 2017].

Audiogames.net. (2017). *AudioGames, your resource for audiogames, games for the blind, games for the visually impaired!*. [online] Available at: https://www.audiogames.net/list-games/listgames.php?mode=full [Accessed 15 Oct. 2017].

World Health Organisation (2010). *Global Data on Visual Impairments 2010*. [online] World Health Organisation, p.Page 5. Available at: http://www.who.int/blindness/GLOBALDATAFINALforweb.pdf [Accessed 15 Oct. 2017].

Grubb, J. (2014). *Gaming advocacy group: The average gamer is 31, and most play on a console*. [online] VentureBeat. Available at: https://venturebeat.com/2014/04/29/gaming-advocacy-group-the-average-gamer-is-31-and-most-play-on-a-console/ [Accessed 15 Oct. 2017].

TutorialsPoint (2017). *SDLC Waterfall Model*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm [Accessed 25 Oct. 2017].

TutorialsPoint (2017). *SDLC Agile Model*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/sdlc/sdlc\_agile\_model.htm [Accessed 27 Oct. 2017].

Rongala, A. (2017). *Benefits of C / C++ over Other Programming Languages*. [online] Invensis Blog. Available at: https://www.invensis.net/blog/it/benefits-of-c-c-plus-plus-over-other-programming-languages/ [Accessed 31 Oct. 2017].

Technologies, U. (2017). *Unity - Manual: Physics*. [online] Docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/PhysicsSection.html [Accessed 31 Oct. 2017].

Games, E. (2017). *Programming Guide*. [online] Docs.unrealengine.com. Available at: https://docs.unrealengine.com/latest/INT/Programming/index.html [Accessed 31 Oct. 2017].

Games, E. (2017). *Unreal Engine Features*. [online] Unrealengine.com. Available at: https://www.unrealengine.com/en-US/features [Accessed 3 Nov. 2017].

Hesketh, D. and Campbell, L. (2017). *Bright Void*. Glasgow: Whiteboard Games.

Ergürel, D. (2017). *The latest virtual reality headset sales numbers we know so far*. [online] Haptical. Available at: https://haptic.al/latest-virtual-reality-headset-sales-so-far-9553e42f60b5 [Accessed 4 Nov. 2017].

HTC Corporation (2017). *VIVE™ United Kingdom | Buy Vive Hardware*. [online] Vive.com. Available at: https://www.vive.com/uk/product/ [Accessed 4 Nov. 2017].

HTC Corporation (2016). *http://www.htc.com/managed-assets/shared/desktop/vive/Vive\_PRE\_User\_Guide.pdf*. [ebook] New Taipei City: HTC Corporation, pp.20-25. Available at: http://www.htc.com/managed-assets/shared/desktop/vive/Vive\_PRE\_User\_Guide.pdf [Accessed 4 Nov. 2017].

Stuff.com (2016). *Oculus Rift vs HTC Vive*. [online] Stuff. Available at: https://www.stuff.tv/my/features/oculus-rift-vs-htc-vive/setup-and-requirements [Accessed 5 Nov. 2017].

Merabet, L., Connors, E., Halko, M. and Sánchez, J. (2012). Teaching the Blind to Find Their Way by Playing Video Games. *PLoS ONE*, [online] 7(9), p.e44958. Available at: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0044958 [Accessed 9 Apr. 2017].

Technologies, U. (2017). *Unity - Google Daydream*. [online] Unity. Available at: https://unity3d.com/partners/google/daydream [Accessed 5 Nov. 2017].

Valve Corporation (2017). *Steam Hardware & Software Survey*. [online] Store.steampowered.com. Available at: http://store.steampowered.com/hwsurvey?platform=combined [Accessed 7 Nov. 2017].

Lyer, K. (2017). *The Best Operating Systems For PC Gaming*. [online] TechWorm. Available at: https://www.techworm.net/2017/02/best-operating-systems-pc-gaming.html [Accessed 7 Nov. 2017].

Microsoft Corporation (2017). *Your request appears to be from an automated process*. [online] Microsoft.com. Available at: https://www.microsoft.com/en-gb/store/d/windows-10-home/d76qx4bznwk4/1NT3 [Accessed 7 Nov. 2017].

McDonald, E. (2017). *The Global Games Market 2017 | Per Region & Segment | Newzoo*. [online] Newzoo. Available at: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/ [Accessed 7 Nov. 2017].

Gartner (2016). *Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016*. [online] Gartner.com. Available at: https://www.gartner.com/newsroom/id/3415117 [Accessed 7 Nov. 2017].

Allied Business Intelligence (2011). *97% of All Smartphones Will Have Touchscreens by 2016*. [online] Abiresearch.com. Available at: https://www.abiresearch.com/press/97-of-all-smartphones-will-have-touchscreens-by-20/ [Accessed 7 Nov. 2017].

Brennan, C. (2014). *'Video-less' 3D game made for blind players*. [online] BBC News. Available at: http://www.bbc.co.uk/news/technology-28757186 [Accessed 8 Nov. 2017].

Fingas, J. (2017). *Google built a spatial audio kit for games and VR*. [online] Engadget. Available at: https://www.engadget.com/2017/11/06/google-resonance-audio/ [Accessed 8 Nov. 2017].

AudioGaming (2016). *Technologies | AudioGaming*. [online] Audiogaming.net. Available at: http://www.audiogaming.net/technologies [Accessed 8 Nov. 2017].

Bauer, C., Hirsch, G., Zajac, L., Koo, B., Collignon, O. and Merabet, L. (2017). Multimodal MR-imaging reveals large-scale structural and functional connectivity changes in profound early blindness. *PLOS ONE*, [online] 12(3), p.e0173064. Available at: http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0173064&type=printable [Accessed 9 Nov. 2017].

Norman, J. and Bartholomew, A. (2011). Blindness enhances tactile acuity and haptic 3-D shape discrimination. *Attention, Perception, & Psychophysics*, [online] 73(7), pp.2323-2331. Available at: https://link.springer.com/content/pdf/10.3758%2Fs13414-011-0160-4.pdf [Accessed 9 Nov. 2017].

Huang, J., Yan, E., Cheung, G., Nagappan, N. and Zimmermann, T. (2017). Master Maker: Understanding Gaming Skill Through Practice and Habit From Gameplay Behavior. *Topics in Cognitive Science*, [online] 9(2), pp.437-466. Available at: http://onlinelibrary.wiley.com/doi/10.1111/tops.12251/pdf [Accessed 9 Nov. 2017].

Nealen, A. (2012). *Game feel, Principles of virtual sensations Controller mappings*.

van Gaal, W. (2017). *The Blind Gamer Playing 'Street Fighter 5' at a Pro Level*. [online] Motherboard. Available at: https://motherboard.vice.com/en\_us/article/nev47x/the-blind-gamer-playing-street-fighter-5-at-a-pro-level [Accessed 10 Nov. 2017].

Yin-Poole, W. (2016). *Meet the blind gamer with a Killer Instinct*. [online] Eurogamer.net. Available at: <http://www.eurogamer.net/articles>

Valve Corporation (2018), *A Blind Legend* [online] Store.steampowered.com. available at: <http://store.steampowered.com/app/437530/A_Blind_Legend/> [Accessed 30 Mar. 2018]

Google LLC (2018), *A Blind Legend* [online] play.google.com. available at: <https://play.google.com/store/apps/details?id=com.dowino.ABlindLegend&hl=en> [Accessed 30 Mar. 18]

Killer Instinct, 2015. *The Sounds of Killer Instinct: Season 2.* [online video] Available at: <https://youtu.be/IoMyR6LxT0M> [Accessed 31 Mar. 18]

Statista (2018). *U.S. most popular video game genres 2016 | Statista*. [online] Statista.com. Available at: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/ [Accessed 31 Mar. 2018].

Apple (2018). *Purchase and Activation - Support - Apple Developer* [online] Developer.apple.com. Available at: <https://developer.apple.com/support/purchase-activation/> [Accessed 02 Apr. 2018]

Game Accessibility(2018) *Stealthfly* [online] Game-accessibility.com. Available at: <http://game-accessibility.com/game/stealthfly/> [Accessed 02 Apr. 18]