AbyssScape – VR Dungeon Escape

Aisling Kelly-Brophy

N00183696

Supervisor: Catherine Noonan

Second Reader: Joachim Pietsch

Year 4 2021-22

DL836 BSc (Hons) in Creative Computing

Abstract

The aim of this project was to examine how such aspects as lighting and setting can create an immersive experience in virtual reality. Specifically, this was achieved through the creation of a dark and fantastical virtual reality experience, in which the player could solve challenges placed before them to escape their environment. Such virtual reality experiences have proven popular in the last few years; games such as *The Room VR: A Dark Matter* (FireproofGames, 2019) and *A Rogue Escape* (Armor Games Studios, 2021) have continued to attract positive reviews (Mairi, 2022). Furthermore, the fantasy genre, in particular that of medieval fantasy, has experienced a resurgence in the last ten years or so; thanks to the enduring popularity of the roleplaying game, *Dungeons & Dragons* (Wizards of the Coast, 1993). In light of this knowledge, it was decided, in examining these properties of immersion, that a virtual reality dungeon escape game would be created, utilising low-poly assets and stylised graphics.

The game was developed in the Unity engine, and coded using C#. Utilities such as Unity's inbuilt XR Toolkit were also used. The steps involved in making the game were background research, requirements gathering, and design; followed then by implementation and testing. Testing involved both functional testing and user testing. Results from these sessions highlighted areas for improvement, or for further development. Aspects to be considered for future development could include the expansion of the in-game environment, the provision of even more puzzles/challenges, and the implementation of more interface features, such as a pause menu.

Acknowledgements

I would like to extend my thanks to the many people who showed support and offered advice throughout the duration of this project. In particular, I would like to thank my project supervisor, Catherine Noonan, and my second reader Joachim Pietsch, for their help, communication, and overwhelming support throughout *AbyssScape*'s development.

I would also like to thank those who agreed to participate in my surveys, interviews, and usability testing. In particular, I would like to extend my heartfelt thanks to the members of IADT's Cosplay and RPG societies. Your knowledge and passion for games and interactive experiences has provided me with the much-needed motivation to complete such a large-scale project.

Lastly, I would like to extend my utmost thanks to family, friends, and anyone else who supported me in any way possible throughout this project. Your unwavering support and loyalty have seen me through the project's most difficult phases, and I cannot thank you enough.

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

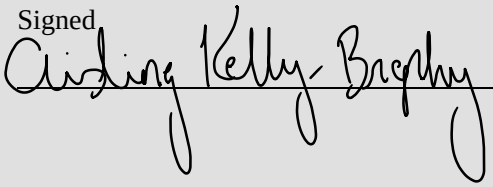
The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student :

Signed 

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction

AbyssScape is a virtual reality puzzle/escape room game in which the player, finding themselves trapped in a dark dungeon, must find a way to escape. It enables the player to locate various objects, combine their use in different ways, and solve puzzles before reaching the exit. The target audience for this game was envisioned as those already familiar with games and/or virtual reality, but its simplicity means that it could also be enjoyed by those who have never played before.

Virtual reality is a new and innovative area, not only in the realm of gaming, but in many others. Virtual reality technology has been used in areas such as real estate, tourism, and medicine (Thompson, 2017). Furthermore, Meta's discussion of a shared "metaverse" (Ravenscraft, 2021), a virtual world that closely mimics our own, has brought discussion and awareness of VR technology to the fore. Even within gaming exclusively, virtual reality applications continue to garner positive reviews, as they immerse their players like never before (Mütterlein, 2018). *AbyssScape* draws on the rising popularity of this niche in the market.

The game was developed in the Unity environment, and programmed using C# in the Visual Studio Code editor (see Implementation chapter). Unity is a beginner-friendly engine for developing 2D or 3D games, mobile apps, and more. Furthermore, it comes with its own proprietary XR Interaction Toolkit, which aids in the creation of VR applications.

The project was an individual undertaking. All research, design, implementation and testing was undertaken by a single individual. Though this meant more creative freedom, and less hindrance in fulfilling the project's vision, it also meant that the workload at times could seem quite daunting. Fortunately, project management tools such as Trello, coupled with careful note-taking, alleviated this burden, and as a result the project never felt too overwhelming.

There were several phases to the project: research, requirements, design, implementation, and testing.

Before the project could be developed, research into the area of immersive gaming, and in particular, virtual reality, was undertaken. A summary of these findings can be found in the Research section.

Requirements were also gathered before work commenced on the project. This involved researching similar games, their features, and what made them popular. Research into the features deemed most important in games and VR experiences was also completed during this phase.

During the design phase, sketches and wireframes of gameplay mechanics, user interfaces, and puzzles were drawn up. Diagrams depicting the "flow" of the game were also created. In addition, design elements such as font style, colour scheme, and UI assets were considered at this stage.

Once these concepts were decided upon, the implementation of the game could begin. Over the course of several fortnightly sprints, features such as environment layout, controls, and interactivity were continuously added.

Once a sufficiently working version of the game was complete, it went through a testing phase. Testing was split into two types: functional testing, in which each gameplay mechanic was scrutinized; and user testing, in which those unfamiliar with the game tested it for themselves. The functional testing carefully analyzed player input, to test if each mechanic performed its expected function. The user testing aimed to determine which features were most enjoyable, or if there were

any tasks that were unnecessarily difficult. Once completed, feedback was received from the test users, as well as suggestions on how to improve the game experience.

Each of the project phases has its own individual chapter, where it is discussed in more detail.

2 Research

Research into the area of design for immersive gaming was undertaken before commencing this project. It aimed to explore the design factors that create a sense of immersion, both as applied to the wider world and that of digital games. This research is outlined below.

2.1 Introduction

Immersion is, according to S. B. Schafer (2011), "a process of temporarily expanding consciousness into areas of the unconscious—something like hypnosis, but retaining consciousness as one does in lucid dreaming states".

Elaborating on this notion, humankind has always had a fascination with games. In fact, early examples of gaming tools, such as dice sets, have been unearthed dating back to three-thousand years ago (Kumar, Herger, & Dam, 2018). During the last fifty years or so, this fascination has only become more prevalent with the advent of digital games. But what exactly is it about games that continues to capture imaginations the world over? More specifically, how have games evolved to better involve their players, and immerse them in worlds unlike their own? What key concepts can be utilised to bring memorable experiences to players?

The aim of this chapter is to examine how different elements of design can be employed to provide immersive and memorable game experiences. To achieve this, key design concepts, and their psychological/emotional influence, will be examined; first as standalone elements, then as can be applied to game design. Then, the notion of "aesthetics" – both as applied to the everyday and to the world of games – will be explored. Finally, innovations in player immersion, current and future, will be explored.

2.2 Design Concepts

To understand the role of design in crafting memorable and immersive gaming experiences, it is important to first understand the importance of basic elements such as colour, shape, and sound. Questionable design choices, after all, can easily make or break a player's immersion.

2.2.1 Design Concepts in Everyday Life

This section will discuss the broader application of design elements, and the psychology behind design choices such as colour and shape. Though not specifically tied to the medium, many of these concepts can indeed be utilised when designing games.

2.2.1.1 Colour

According to the organisation DesignCloud, choosing harmonious colour schemes is one of the essential pillars of graphic design (DesignCloud, 2019). For example, consider a website with a jarring colour scheme – it will not attract an audience, and may in fact deter them. Conversely, consider a website with harmonious colours and a consistent theme. The latter will inevitably attract – and retain – a larger audience. In fact, according to the organisation Design Wizard, it is estimated that brand recognition can be increased by up to 80% through careful use of colour in logo design and marketing (Design Wizard, 2019).

In addition, each and every colour carries a psychological meaning (Design Wizard, 2019). For example, green is associated with nature, tranquillity, and health, whereas red may be associated

with strong emotions such as anger or desire. In terms of encouraging user engagement, blue is deemed the most popular colour choice; social media platforms such as Facebook and Twitter utilise the colour not only for brand recognition, but because it is commonly interpreted as a colour of friendliness, creativity, and reliability. Furthermore, it is inclusive; it ensures those who are red-green colour-blind can access the site's content (Cherry, 2020).

2.2.1.2 Lighting

Lighting can have a powerful psychological effect; exposure to different light sources and colours of light can induce various emotional states (Rossi, 2020). In her article, Camilla Rossi of the interior lighting company Karman states that when utilised suitably, lighting can induce in people feelings of "relaxation, intimacy, clear vision, excitement, and productivity". However, when used improperly, it can create a negative response, inducing instead feelings of "stress, sleepiness, melancholy, or anxiety".

Furthermore, lighting is a key element to the creation of atmosphere. In his publication, *Light Design and Atmosphere,* Tim Edensor puts forward the view that modern use of lighting in cities, such as in streetlights and advertisements, aid in creating their bustling, lively atmosphere. The popularity of tourist destinations, such as New York's Times Square or London's Piccadilly Circus, proves how such locations can be "atmospherically charged by illumination" (Edensor, 2015). It is clear that the visitor experience and immersion would not be the same without this vital element.

2.2.1.3 Shape

Shape has long served as a way for the human mind to organise and categorise the world around it. Consequently, many different psychological meanings have been assigned to both geometric and organic shapes over time. Circles, for example, have long been used as representations of the sun and moon, and, even where they are not, they still "take on some of the psychological and cultural baggage" of the celestial entities (Fussell, 2020). Other psychological allocations to shapes include the symbol of squares as stability and dependability, and triangles as dominance.

Organic shapes, on the other hand, feel familiar and comforting, owing to their presence in nature. As they are irregularly shaped, they may also be used to symbolise creativity and spontaneity (Fussell, 2020).

It is easy to see how these allocations and their deep-rooted psychological meanings can be used to provoke the viewer/audience's emotions. Such a connection could easily be used to immerse a potential user in an experience.

2.2.1.4 Sound

Sound design is just as crucial as graphics when it comes to providing an immersive experience. Fiona Thomas of Production Attic states that one might not notice the sound design behind videos and movies, but that the impact of it is certainly felt. Sound not only adds to the realism of settings, but also aids in world building, telling stories, and, like the other aforementioned design concepts, evoking emotion.

Clever sound design can, also, evoke emotions without the aid of accompanying graphics or film. According to Rev. Dr. Bradley D. Meyer of DesigningSound.org, setting and ambience can help in setting a scene, eg. a beach with gently rolling waves - one that perhaps reminds listeners of childhood vacations, or other such fond memories. Evocation of emotion through sound, without the "crutch" of visuals, can also be linked to empathy; for example, there is evidence that sounds representative of one's emotional state, such as crying or screaming, may provoke the same response as if it were accompanied by visuals (Meyer, 2016).

2.2.2 Design Concepts as Applied to Games

In this section, the application of fundamental design principles to digital games will be reviewed. Games work to immerse and engage their players not only through the use of the above elements, but by combining them to create memorable characters and environments.

2.2.2.1 Character Design

Combinations of the previously mentioned design concepts can be utilised to create engaging character designs. Characters need not be complex for players to find them relatable and/or aesthetically pleasing; in fact, one needs to look no further than mobile games such as Rovio's *Angry Birds* (2009) to learn how to design simple yet effective characters.

In his UX Design article, Stanislav Stankovic (2021) argues that the characters of *Angry Birds* offer important lessons in character design. The familiarity and simplicity of each design relies primarily on three important fundamentals of design – shape, colour, and emotion. The simple shapes and colours of the characters not only make them easy to identify, but, from a user experience perspective, make it evident the purpose for which each was designed. A triangle-shaped bird, for example, implies piercing ability; a large, heavy bird, destructive power; and a group of small, identical birds, a clustered attack (Stankovic, 2021). The clarity provided by each design means there is no confusion on the user's end; such would ruin the player's experience and sense of immersion.

2.2.2.2 Use of Colour

Colour can be used to denote many things in games – for example, signifying faction or highlighting the purpose of objects (DVNC Interactive, 2018). However, the main use discussed here will be the utilization of colour for both world building and emotional effect.

In the article by DVNC Interactive (2018), "Color Theory in Games – An Overview", it is mentioned that colour is an essential component to the observable world of a game. In Playdead's *Limbo* (2010), for example, the foreground is rendered in dark, black shades, whereas the non-interactable background elements are white and grey. Similarly, in Nintendo's *Super Mario Bros.* (1985), the background is a muted blue and largely empty, while interactable foreground elements and the player character are a warm orange-brown. This not only makes each interactable element immediately obvious, but also sets the scene and tone of the game.

Colour can also be used to easily differentiate characters, reducing confusion and increasing the affinity between player and game world. Returning to the *Angry Birds* example, Stankovic (2021) states that each character makes use of bright primary colours such as red, blue, yellow, and green. Not only are these colours eye-catching and easy to remember, they are also highly recognisable, even to children. In fact, the argument is made that the simplicity of the *Angry Birds*' designs mean that they are easily reproduced on paper, further promoting the brand, and deepening the emotional connection (Stankovic, 2021).

2.2.2.3 Environment Design

In addition to characters, creating believable, memorable environments is key to immersing players in a game's world.

Game worlds, Behnam Mehrafrooz (2020) of the Pixune organisation states, must feel authentic. Often, he claims, reality can be a solid foundation for building imaginative worlds upon. He goes on to state that even the art team of the acclaimed *Assassin's Creed II* (2009) spent time in Florence and Venice, sketching the environment around them, to then be able to craft a living, realistic game

world based on their findings. This results in an experience that feels “real and authentic” to players, even if they have never visited these places (Mehrafrooz, 2020).

It is often not enough for game environments to be realistic or “authentic”, however. Mehrafrooz goes on to state that environmental storytelling – the placement of non-interactable objects in a game’s environment – piques player’s curiosity and encourages them to further explore the game’s environment. An example given by Edwin McRae (2017) is that of a bloodstain in the empty corridors of *Alien: Isolation* (2014). Nothing happens if the player attempts to interact; it simply implies that there is danger ahead for the player. Adding such small touches aids in creating a fantasy world towards which the player develops real emotions (Mehrafrooz, 2020).

2.2.2.4 Sound Design

It is not enough for a game to have detailed visuals to be fully immersive – suitable audio must accompany game environments and characters if the experience of playing to be truly immersive.

The video game company Pearl Abyss’ Hwiman Ryu (2020) states that games can be enhanced by “impactful” sound design. Sound, he says, must be fun and exciting; even if realistic sound effects are your goal, it is often more effective to exaggerate them in some way. If they were exactly the same as the real world, the player would probably not feel excited about the game at all. More interesting sounds, however, are more noticeable and, as such, enhance the gameplay experience (Ryu, 2020).

Dustin Tyler (2021) of GameDesigning.org states that it takes playing a game on mute to realise how important sound is to the experience. Sound effects and music are there to make the game world, as well as its stories and characters, come alive. In fact, as well as aiding in immersion, a game’s soundtrack can evoke feelings of joy and nostalgia in the player, further developing the emotion felt towards the game (Tyler, 2021).

2.3 Aesthetics

“Aesthetics” refers to the philosophical study and appreciation of beauty and taste (Scruton, 2021). It is closely interlinked with the study and appreciation of the arts and of artistic value. More broadly, aesthetic experiences can be defined as having in common four factors: those of focus, intensity, unity, and coherence (Shelley, 2017).

In video games, aesthetics is defined as “the sensory phenomena that the player encounters in-game”, as well as the “aspects of digital games that are shared with other art forms” (Ribeiro, Rogers, Altmeyer, Terkildsen, & Nacke, 2020).

2.3.1 Aesthetics in Everyday Life

In this section, it will be laid out how the concept of aesthetics can be utilised to create atmosphere and the elicitation of an emotional response, both in games, and in other forms such as the spoken word.

2.3.1.1 Creating Atmosphere

It is hard to define what the concept of “atmosphere” is, exactly – though, as Greg Kasavin of Supergiant Games puts it, it can be described as a product of both theme and tone (Iyer, 2018). This can be applied to games, of course, but also, more broadly, to other experiences.

Games and player immersion go hand in hand – but, even before digital games could ever be conceptualised, people have always been seeking ways to make experiences more atmospheric. In

his article, “A Brief History of Immersion, Centuries Before VR”, P. T. Allen (2018) argues that, to an extent, even the spoken stories told by our ancestors worked to immerse the listeners in a world not their own. In fact, creation of mood and atmosphere is a crucial part to the storytelling process, even today. In written prose, atmosphere can be created through descriptions that evoke all five senses, thus enabling the reader to vicariously experience what it would be like to be present in the story (Berve, 2019).

This concept of atmosphere, and the human desire to experience a sense of immersion, also gave rise to impressive feats of art and design down through the ages. Examples of such include the impressive stained-glass windows of medieval churches and cathedrals, purpose-made to present “an immersive sense of otherworldliness” to their onlookers (Allen, 2018). Other examples of atmospheric experiences include festivals or Christmas markets. Here, crowds gather not only to purchase goods or attend events, but to simply take in the atmosphere created by the festive lighting (MK Illumination, 2020).

2.3.1.2 Emotional Design

Emotions are a normal part of everyday life. A reaction of fear, for example, might be felt when looking over a high banister railing. But again, these can be applied in many ways to enhance the user experience of an application (Komninos, 2020).

Various companies have already found out that employing an emotional aspect to their products helps in brand recognition and customer retention. In fact, as web designer Paul Jarvis (2014) states, “the best commercials sell a feeling or idea more so than an actual product”. But why is this method so effective? Simply put, it is because such an approach is enjoyable, memorable, and personal. It engages the user, keeping in their mind the company, product, or service (Jarvis, 2014).

2.3.2 Aesthetics as Applied to Games

In this section, the role of aesthetics in digital game worlds will be explored. The concepts of atmosphere, emotion, and appeal to players will be discussed, to examine how all can be combined into a singular captivating experience.

2.3.2.1 Atmosphere in Games

For Greg Kasavin, writer and designer of Supergiant Games’ *Bastion* (2011), atmosphere is important in that it constitutes the game’s “unique identity and feel” (Ribeiro et al, 2020). He goes on to state that atmosphere “creates immersion”, and ensures the game is “aesthetically coherent and creates the appropriate mood”.

Though games of many different genres can be described as atmospheric, and, consequently, fulfil their goals of immersing the player in their world, the discussion of such topics always returns to a prominent example – horror games. It is in these games that, arguably, the creation of the right atmosphere is most important. For example, in Supermassive Games’ *Until Dawn* (2015), the player knows, through their decisions and gameplay actions, that any wrong move could have dire consequences for the characters. Even leaving theme and setting aside, this works effectively to keep the player immersed and invested in the story, and evoke a feeling of unease and dread (James, 2019).

It is important to note that games of similar genre and/or play style can have vastly different atmospheres. A sci-fi game, for example, can be either humorous and quirky, or dark and gritty (James, 2019). As different atmospheres appeal to different types of players, it is important that designers consider exactly what sort of atmosphere they are looking to create, and why (James, 2019).

2.3.2.2 Emotions and Game Appeal

According to Stephane Bura (2008) of GameDeveloper.com, "players don't play to complete games… players play to feel emotions." Game design is not just about the technical aspects – it is "experience crafting for the purpose of emotion engineering". Emotions felt are not just a side effect of playing games – arguably, they are the very reason games are played.

So, if games are about the evocation and engineering of emotion, how does that tie into the provision of immersive experiences? If we refer once more back to the *Angry Birds* example, Stankovic (2021) goes on to say that the emotion and motivation behind the characters is also, aside from the aforementioned aspects, a key to its success. The title alone is enough to prompt anyone to wonder, "Why are these birds so angry?" It makes for an interesting opener, one to which the player must find the answer. Even the emotion behind the characters itself is relatable; everyone has felt anger sometimes. This level of relatability, therefore, prompts an empathetic response in the player, further drawing them in to the world of the game.

2.4 Future Innovations in Immersion

In this section, it will be discussed how current innovations in gaming technology – most notably, that of augmented and virtual reality (AR/VR) – can provide gaming experiences even richer in immersion than ever before.

2.4.1 Augmented Reality

Augmented reality is the imposition of virtual objects on the real-world environment. Players of augmented reality games still observe and interact with the real world, but through a device that enhances their reality by including virtual-only elements, such as a phone screen (Häger, 2017).

A case study for the popularity of AR, and indeed how it can be better utilised to provide immersive virtual experiences, is that of Niantic's *Pokémon Go*, released in 2016. The popularity of Nintendo's *Pokémon* franchise was no doubt reason for its overnight success; but its success did not come from that of its parent franchise alone. *Pokémon Go* enables users to locate and catch virtual creatures based on their device's location data – however, it was the feature which allowed players to use their phone's camera to "see" the monsters in "real life" that made the game so memorable. Though the app only makes limited use of the AR feature, it was undoubtedly one of its more prominent selling features (Bannerflow, 2016).

2.4.2 Virtual Reality

Even more immersive than augmented reality is virtual reality (VR).

An augmented reality (AR) application – though there may be virtual elements present – still requires the player to engage with the real world. Games such as *Pokémon Go* require that the player traverse their locality to progress. In virtual reality, however, the player does not see any trace of the real world at all. Through a headset such as the Oculus Quest or HTC Vive, the user engages with a completely computer-generated environment, right before their eyes.

The one fundamental element of a VR system is a computer-generated world that surrounds the participant, and that allows for head tracking (Slater, 2018). This head tracking mimics the natural human action of moving one's head to look at one's surroundings. Thus, it appears that the user "visits" the environment, rather than "sees" it (Mütterlein, 2018). If this is how players will interact with games in the future, it is easy to see how they will be able to feel completely immersed in the game world, as if they belong to it and not ours.

2.5 Conclusion

The concept of immersion has long fascinated humankind. In the world of game design, this is no different. Constant strides are being made in development of more immersive games, environments, and technologies.

Games, and indeed, any form of media, cannot be truly immersive without the utilisation of several important design concepts. Elements such as colour, shape, sound, and lighting can all be combined to provide the best experience possible for the end user.

In games, where immersion is paramount, this is even more important. Aspects such as character, careful use of colour, and environment/sound design are also important factors to note in the creation of engaging experiences.

Games will never stop being immersive. The technology to create even more immersive experiences is already making its way into the mainstream. The information presented here could be used, as an example, to create a virtual reality game, in which the player can feel truly included and enveloped.

3 Requirements

3.1 Introduction

In this chapter, the requirements for the application will be explored and discussed. Research for the application's development will involve searching for similar existing games, conducting interviews with prospective players, launching a user survey, and modelling personas based on the results. The requirements, both functional and non-functional, will then be listed, to get an overview of the most important features to include.

3.2 Requirements gathering

3.2.1 Similar applications

Three similar applications to the game I intend to develop are *I Expect You to Die, Conductor,* and *Obduction.* Each are VR adventure games that require the player to find clues, solve puzzles, and escape from a location or situation.

Application 1: *I Expect You to Die*

Figure 1: Screenshot from "I Expect You to Die". © Schell Games

I Expect You to Die is a Bond-esque secret agent puzzle adventure. The player, as an agent with telekinetic powers, must try and stop a nefarious weapons and pharmaceuticals corporation.

Advantages of *I Expect You to Die:*

- It is humorous and fun to play.
- The gameplay is not too difficult, so is accessible to a wide audience.
- It has an achievements system.
- Puzzles have multiple solutions, so can be solved differently upon replaying.
- Hidden content also increases replay value, as the player may not notice obscure details on first playthrough.

- It can be played from a seated position.
- It does not require an internet connection.

Disadvantages of *I Expect You to Die*:

- Puzzles require a lot of trial and error.
- The game's story is quite short.
- Retail price is quite high for the amount of content.

Application 2: *Conductor*

Figure 2: Screenshot from "Conductor". © Overflow

Conductor is a low-poly styled puzzle adventure in which the player must progress through a series of train stations. They are moved from station to station by steam locomotive, of which they must take charge and protect from enemy drones.

Advantages of *Conductor:*

- Has several different train stations, each with unique puzzles and objectives.
- A train transports you to the next station, which is an interesting mechanic in VR.
- Simple, low-poly style graphics make it less demanding on hardware.
- It is atmospheric and has a good soundtrack.

Disadvantages of *Conductor*:

- Puzzles are too easy.
- The game's story is quite short.
- Has some bugs, which can make the game unnecessarily difficult.
- There is no snap-turn ability with the controllers, so the player must always physically turn around to turn their character.
- Has little replay value.
- The ending is abrupt and unsatisfying.

Application 3: *Obduction*

Figure 3: Screenshot from "Obduction". © Cyan

Obduction is a game in which the player finds themselves transported to a strange alien world, which in places is oddly similar to Earth. They must explore, uncover clues, solve puzzles, and escape back home.

Advantages of *Obduction*:

- Has detailed, beautiful graphics.
- Puzzles contain a lot of attention to detail.
- Gameplay is longer than a lot of similar games.
- Game's world is large and well-developed.

Disadvantages of *Obduction*:

- Open environment means the game doesn't have the claustrophobic nature of most escape puzzle games.
- Game world is very large, and backtracking can be time-consuming.
- There are a lot of loading screens, even in the middle of puzzle-solving.
- The game doesn't give you any hints, so may be difficult for some players.
- Interactable items are not clearly highlighted.
- The ending is unsatisfying.

From the research undertaken, it is hoped that some of the key elements of these games and their progression can be incorporated into the application. It was also useful to note the negative points players had about these games, as it was made clear what to try and avoid.

3.2.2 Interviews

To further this research, interviews were conducted with prospective players. The interview questions asked first about their experience of games as a whole, then how each felt about VR games, and if they would play them. The interview transcripts can be found in the Appendix.

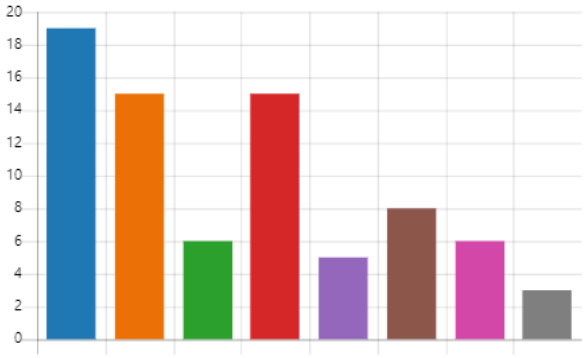
3.2.3 Survey

In addition to interviews, a questionnaire was created to explore the features that prospective players would most like to see in the VR game. Respondents were asked first if they play games, and if so, what they enjoy the most about them. They were then asked if they have ever used any VR applications or games, and if so, what they felt were the most important features. Participants who answered that they had never played a VR game, or used a VR application, were even asked if they would consider it at some point, and why.

4. If yes, what do you enjoy the most about games? (select top 3 most important features)

More Details

Gameplay	19
Story	15
Graphics	6
Characters	15
Ease of use	5
Replay value	8
Sound design	6
Other	3

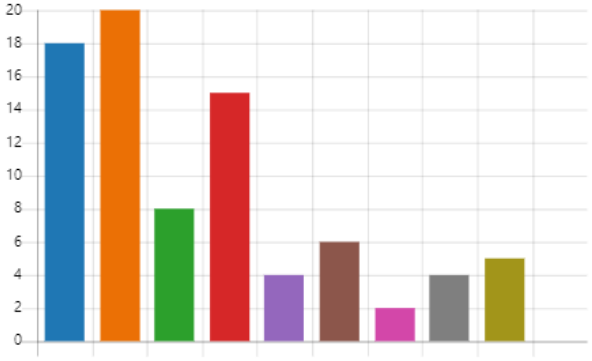


From the results, it became clear that the most important aspect of games, for most, is the gameplay. This was followed closely by story and characters.

7. What features do you think are the most important in a virtual reality app/game? (select top 3 features)

More Details

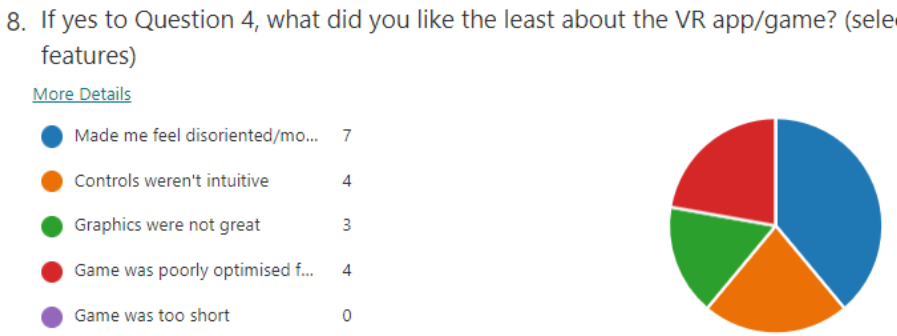
Comfort	18
Ease of use	20
Gameplay	8
Controls	15
Replay value	4
Graphics	6
Story	2
Characters	4
Sound design	5
Other	0



When it came to the most important features of a virtual reality (VR) game, the results were slightly different. This time, the most important features, according to the majority of respondents, were ease of use and comfort. These were followed closely by gameplay and controls. Surprisingly, gameplay ranked lower this time round, compared to the earlier question.

This illustrates a key difference between conventional gaming and VR gaming – that of usability/player comfort. When developing a standard video game, these are undoubtedly still of importance, but not in quite the same way. As virtual reality games involve the user directly participating in the game's environment, and partaking not in the "real" world, but a virtual space, their comfort while playing is of topmost priority.

In fact, many respondents answered that their least favourite aspect to VR games is that of disorientation/motion sickness. This was followed by poor controls and poor optimisation of the game for the target platform.



It became clear, from the results of the survey, that the top features to consider for an immersive VR game were ease of use, comfort, and intuitive controls.

3.3 Requirements modelling

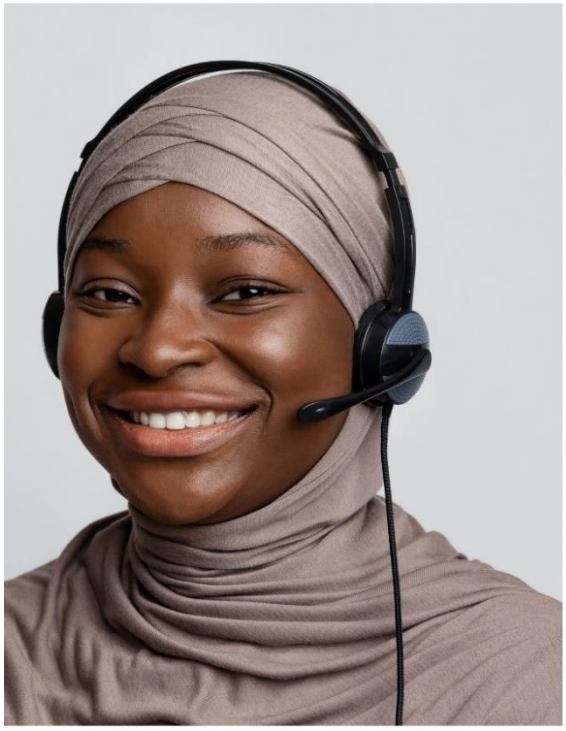
3.3.1 Personas

To further understand the needs and wants of prospective players, two personas were created. One represents someone who likes to use virtual reality as a tool for escapism, the other is a more typical "gamer".

Persona 1

Lili Romero

30 / NYC / Single / Tech and Games Enthusiast



PERSONALITY

Dedicated

Tech Savvy

Fun-loving

MOTIVATIONS

Networking

Convenience

Accessibility

GOALS

- discover games that are both fun and challenging
- use VR as an escape from everyday life

FRUSTRATIONS

- poor work/life balance
- inaffordable price of VR equipment
- dissatisfied with many mainstream games on the market

Persona 2

3.3.2 Functional requirements

1. *6 Degrees of Freedom (6DoF):* the player should be able to interact freely with the VR application, with minimal intrusion. They should be able to move their head, look around, and freely move their hands to interact with objects in the virtual space.

2. *Comfort:* the experience should be optimised in such a way that the player feels comfortable at all times. The experience must limit the likelihood of inducing disorientation/motion sickness in the player.

3. *Ease of Use:* the application must be easy for the player to use. There must not be any confusion as to whether or not objects or UI features are interactable.

4. *Controls:* the game must make use of the Oculus Quest 2's Touch controllers. It must utilise them in such a way that it feels natural for the player to interact with/grab objects in the VR world.

5. *Low-poly Graphics:* the application must be developed using low-poly graphics. This is not only a stylistic choice, but will reduce the amount of processing overhead and thus enable the application to run more smoothly.

6. *Puzzles:* the game should include puzzle elements. The player must search for specific objects/keys needed to progress, and must not be able to move on to the next area until the solution is found.

7. *User Interface (UI):* the application must include interactable menu screens. The player must be introduced to the game, be able to configure settings, and reset puzzles if they become stuck.

8. *Lighting:* while the overall tone of the game is dark and atmospheric, the game must include sufficient light that the player can clearly see their surroundings. The aesthetic choice must not hinder the player's visibility.

9. *Sound:* as the game is designed to be an immersive experience, it must make use of sound. This could be provided in the form of both suitable sound effects and background music.

10. *Atmosphere/Immersion:* as it takes place in a dark fantasy/gothic reminiscent setting, the game must evoke a tense, moody atmosphere. This should work to immerse the player in the environment, and provide a sense of urgency as they venture to escape the level.

3.3.3 Non-functional requirements

- Application must run at a minimum of 72 frames per second. Though this is not an essential requirement, it is the standard for all games/applications uploaded onto the Oculus store.

- Application must be an Android build (.apk). Though it can be built as a Windows executable (.exe) and run from a computer, it will not work correctly when loaded onto the headset.

- Application must be built specifically for the Quest 2 headset. While there are many VR headsets available, the Quest 2 is both the most affordable/accessible, and the only one available for development both within the college and at home.

- Application must be user-friendly. When in a virtual reality environment, it is easy for a player to get disorientated and/or motion sick. This can be mitigated through the inclusion of configurable settings, controller-enabled snap turning (as opposed to continuous joystick movement), and a gentle fade in/out of graphics when scenes are loaded or changed.

3.3.4 Use Case Diagrams

To consider the game flow, and actions the user should be able to take in-game, a Use Case Diagram had to be designed.

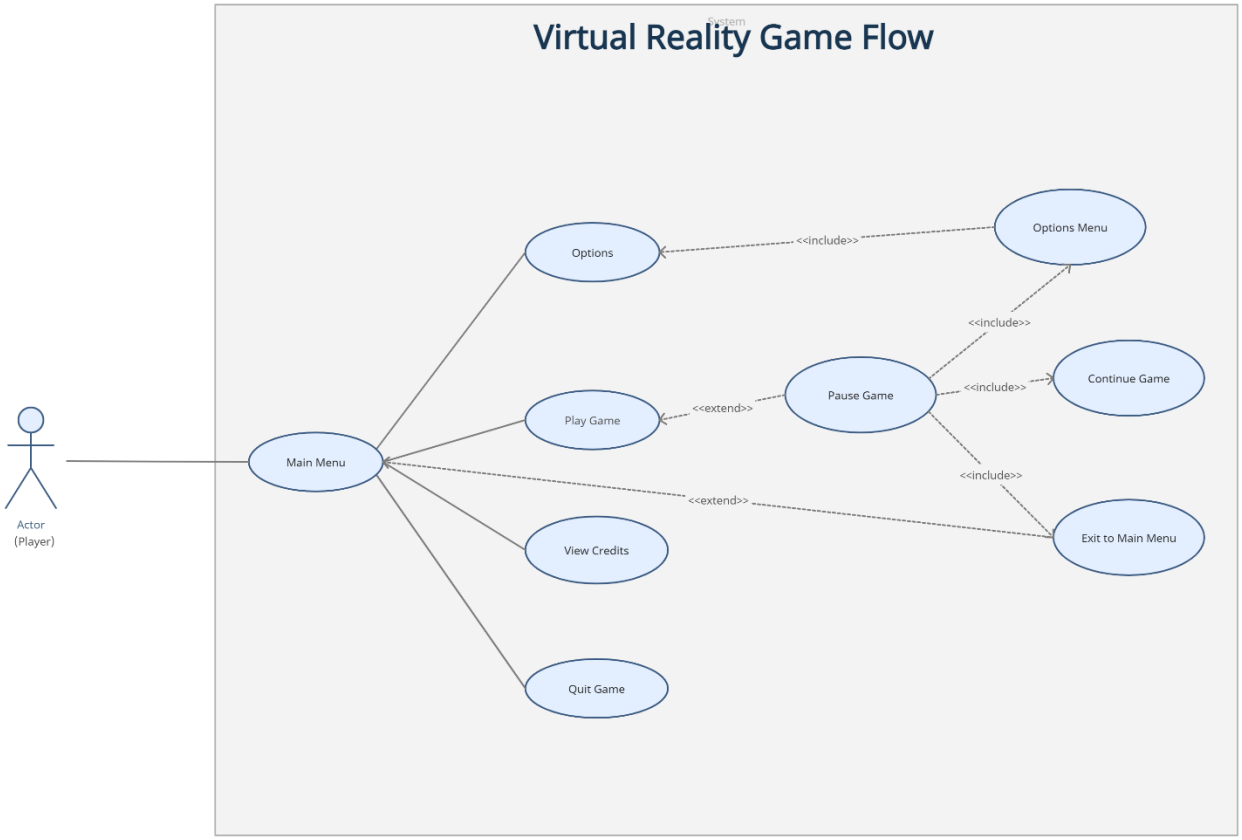


Figure 4: Diagram showing the processes undertaken by the player (actor) as they play the game.

The above diagram displays each scene/option the user progresses through as they play the game. From the main menu, they can select from the following buttons: Play Game, Options, Credits, or Quit Game. When they play the game, and hit the pause button, a pause menu shows up, containing the following: Continue Game, Options, or Exit to Main Menu.

3.4 Feasibility

It is intended for the application to be developed within the Unity game development engine, and targeted especially towards the Oculus Quest 2 headset. This will be feasible as there is access to these headsets both within the college and in a home environment. However, it may be necessary to develop the application primarily in the college lab, as the hardware would be better suited for the complex rendering of VR applications during development.

3.5 Conclusion

Various research methods were utilised to determine a both a prospective audience for *AbyssScape,* and to find out what such players would prioritise in a virtual reality game. As a result of this research, certain aspects of the game, such as ease of use and comfort level, were prioritised.

Analysis of existing games was carried out, and important features, such as aesthetic design and gameplay elements, were noted as reference points.

Interviews, surveys, and personas provided an even stronger idea of who would be most interested in playing *AbyssScape*. They also further illustrated the need for a suitable comfort level and ease of

gameplay, compared to non-VR games. It was discovered that there were among the highest priorities for players of VR games, compared to those of gameplay and story when questioned about traditional games.

4 Design

4.1 Introduction

AbyssScape was developed using the Unity game development engine, and programmed using the C# language within Visual Studio Code. This chapter details the design process behind the game, and includes a basic overview of the applications used in the process. Fundamental design factors, such as user interface design, font choice, colour schemes, user navigation, and game flow are all outlined below.

In the Program Design section, the technologies used to implement the game mechanics will be described, as well as a basic explanation of how they work. In the User Interface Design section, the design of the game, including layout, colour schemes and user interface elements, will be discussed. Similar examples will be referenced to convey how these were used during development to influence the design choices of *AbyssScape.*

4.2 Program Design

In this section, the applications used to develop the game will be outlined. Relevant examples of the interface of each will also be provided.

4.2.1 Technologies

4.2.1.1 Unity Engine

The application was developed using the Unity game development engine, and programmed using the C# language. Specifically, version 2020.3.25f1 was used. Versions 2020.1 and up are LTS (Long Term Support) releases of Unity. This means that they continue to be officially supported by the Unity creators and can easily be patched if bugs are found.

Unity is a very beginner-friendly game development engine. It is free to download and use, though a premium model is also offered. It provides a graphical interface and uses C# code natively, though Java, C/C++ and Lua scripting can also be used. Though Unity can also be used for 2D game development, it is tailor-made for 3D development. 3D game development can be carried out easily and intuitively through the provision of a “scene view” window. In this “scene view”, 3D assets can easily be created and placed within a three-dimensional game world, and adjusted as needed. This graphical interface also makes it easier for users to see the changes made to their code in real-time, or track errors as they arise.

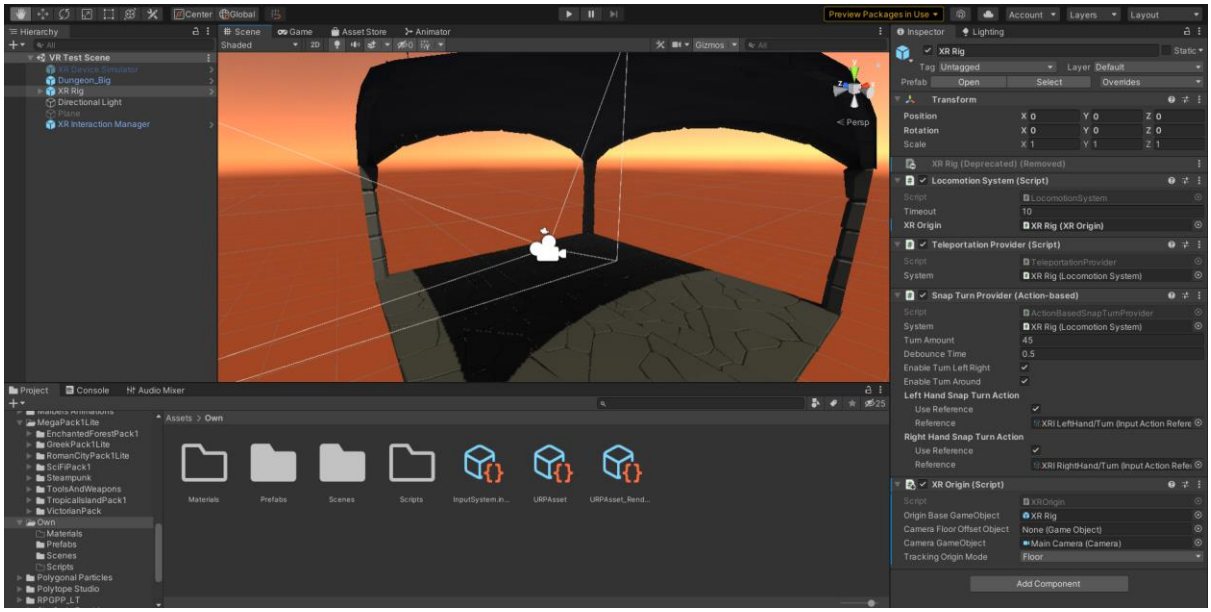


Figure 5: Unity main interface.

There exists an in-built programming feature in Unity called Visual Scripting, in which the user can click and drag gameplay functions/behaviours to create games without writing any code. This makes it even more accessible to those new to game development. However, this was not utilised during the development of this project; all scripting was written inside of Visual Studio Code.

Unity also supports a wide range of operating systems and hardware. It can be used to develop games or applications for Windows, MacOS, and Linux, as well as mobile platforms such as iOS and Android. It also supports development for VR platforms such as Oculus Quest, HTC Vive and Steam VR.

4.2.1.2 Visual Studio Code and C# (C-sharp)

Though other code editors are compatible with Unity, the one chosen for this project was Microsoft's Visual Studio Code. Visual Studio Code is a more simplified, streamlined version of Microsoft's Visual Studio, which makes it easy to focus purely on writing code. It is also extremely customisable and extensions can be added to aid the user in various areas, such as line auto-completion or formatting.

All scripts were written in C#, as it is natively supported by Unity and can be easily integrated into Unity projects. C# is an object-oriented language derived from C, and is quite similar in terms of syntax to C++. It is a high-level language, meaning it is developer-friendly and simple to debug and maintain, compared to low-level languages such as Assembly code. High-level languages, such as C#, can also be run on any platform, whereas low-level languages are entirely machine-oriented.

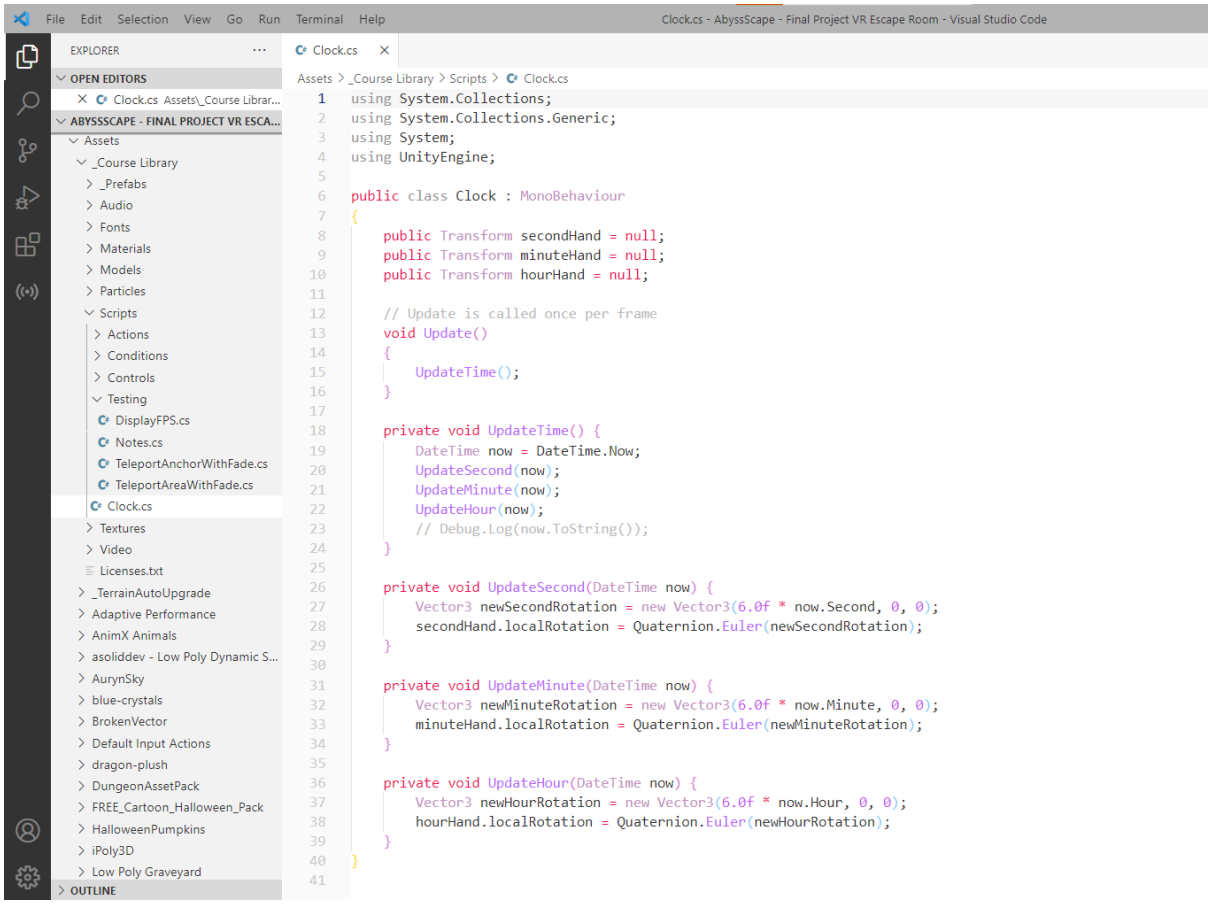


Figure 6: An example of the Visual Studio Code workspace.

4.2.1.3 Unreal Engine

Another game engine which could have been used is Unreal Engine. This is used across many triple-A game companies as it can be used to create very graphically impressive games. However, it would require a lot of independent learning to use. Due to the project's time constraints, it was simply not wise to spend extra time learning a new engine. Unreal Engine also relies more so on C++ than C#; again, this would have required the learning of a completely new language, which would be impractical given the time required to complete the project.

4.2.2 Structure of Unity

4.2.2.1 Controls

 Hand Tool – this is used to move around the game world. The shortcut key for this is Q.

 Move Tool – this is used to move game objects. The shortcut key for this is W.

 Rotate Tool – this is used to rotate game objects around their X, Y, or Z axes. The shortcut key for this is E.

 Scale Tool – this is used to change the scale of game objects. The shortcut key for this is R.

 Rect Tool – this is used to change the length/width of objects such as menu elements (buttons, panels, etc.). The shortcut key for this is T.

4.2.2.2 Hierarchy and Inspector

The Hierarchy panel (left) contains assets currently in the scene. From here, they can be deleted, have their order rearranged, and be selected to view in the Inspector (right). The Inspector shows the properties of the selected game object, such as name, tags, layer, position, and size. These values can then be changed in the Inspector.

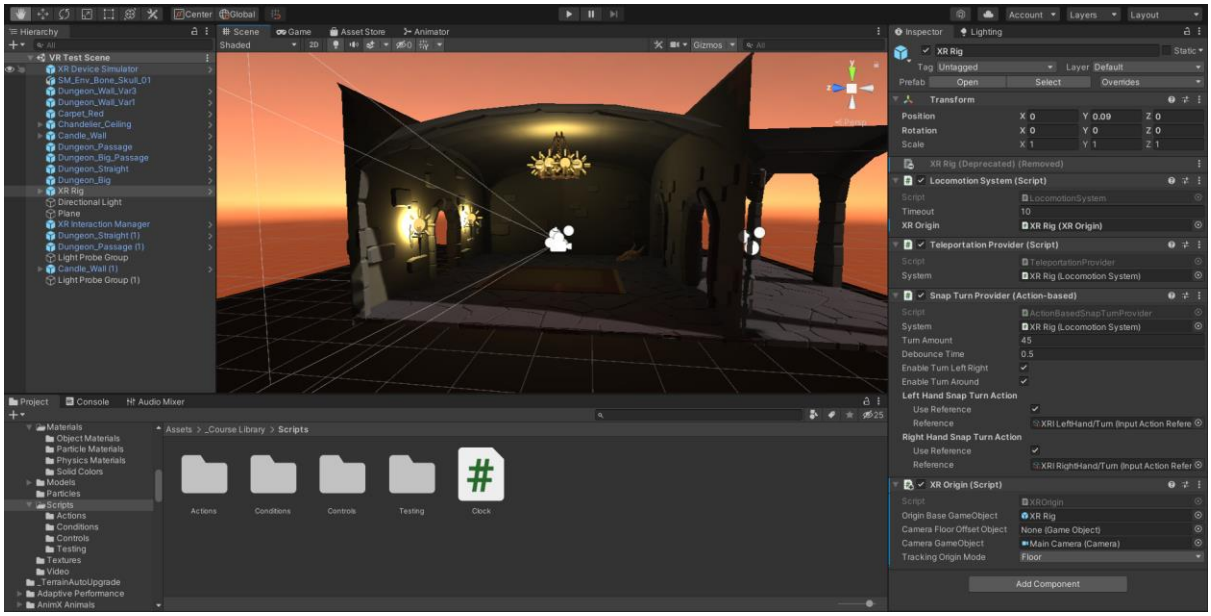


Figure 7: Unity's hierarchy can be seen to the left of the game scene panel. The Inspector is shown on the right.

4.2.2.3 Assets Folder

The Assets folder is the main project folder. It contains all the materials needed to create a game in Unity, such as scenes, models, prefabs, textures, materials, and scripts. These can also be arranged into separate subfolders if the user so wishes; this is not necessary, though it does make the location and selection of assets more streamlined. When an asset pack is imported from the Unity Asset Store, a new folder is created inside the project, containing the new assets.

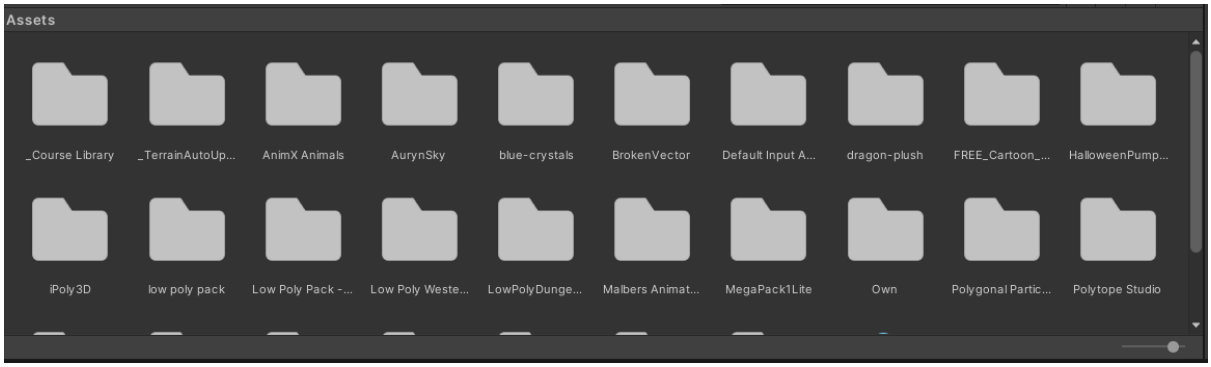


Figure 8: Example view of an Assets folder.

4.2.2.3.1 Scenes Folder

This folder contains all the scenes required in the build of the game. When it comes time for the game to be built, these scenes must be added to the build settings window to be used in the final product.

Figure 9: Scenes folder.

4.2.2.3.2 Prefabs Folder

This is the folder in which prefabricated objects (prefabs) can be stored. Prefabs are instances of objects that have preconfigured properties. When the developer wants to use a certain object again, they can simply create a prefab of the object, so its properties are retained. This creates an exact copy of the object. This is useful if the object needs to be used several times in the same scene, as it cuts down on the amount of time spent configuring each new instance in the Inspector.

Figure 10: Prefabs folder.

4.2.2.3.3 Scripts Folder

This folder contains all the C# scripts required to run the game. These scripts must be added to the relevant objects as a component in the Inspector window if they are to have any effect on the game.

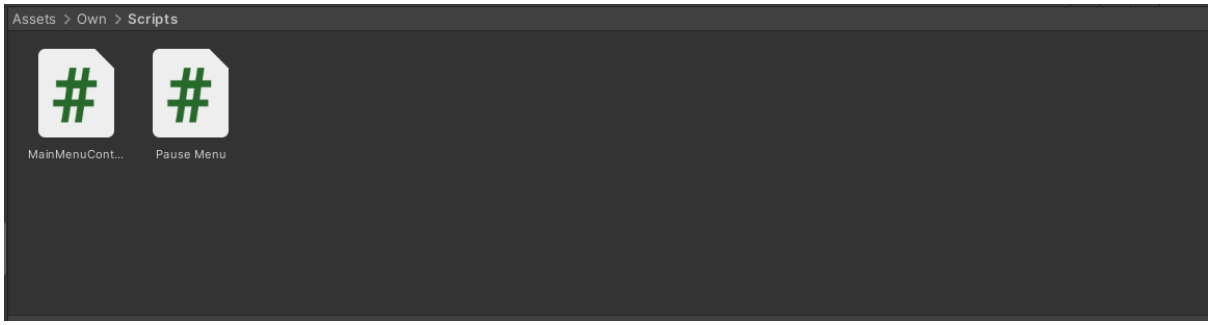


Figure 11: Scripts folder.

4.2.2.3.4 Materials Folder

This is the folder in which materials can be stored. Materials are textures which have been added to objects in the scene. When a 2D texture is added to a 3D object, a material is automatically created. Thus, it is fair to say that a material is a 3D version of that texture; it "wraps" around the mesh of a 3D object.

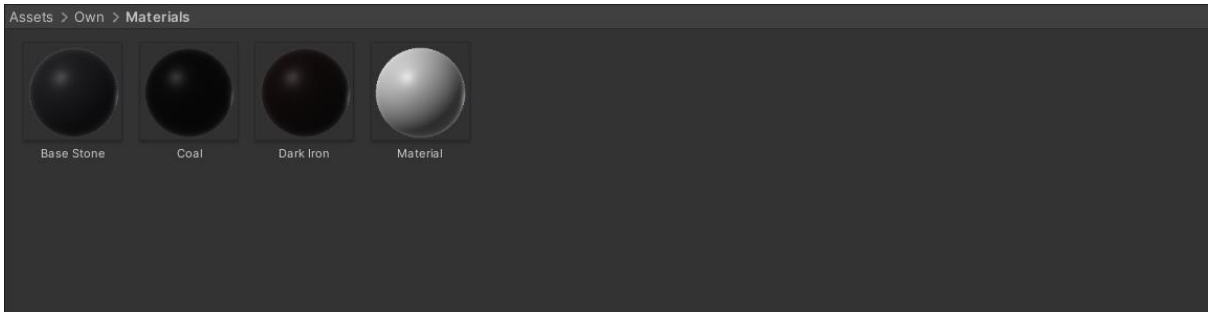


Figure 12: Materials folder.

4.2.3 Design Patterns

The world of the game was put together using a modular low-poly dungeon asset pack from the Unity Asset Store. Modular 3D assets are models that can be seamlessly placed together to create coherent level designs quickly and easily.

In the case of this game, tiles such as floors, walls, and doorways were placed in such a way that rooms could be formed. Over time these combined to form even more detailed structures. Once the overall structure of a room was complete, objects such as light sources and furniture could then be added. Modified versions of the downloaded assets could then be placed in the appropriate prefabs folder should they need to be used again.

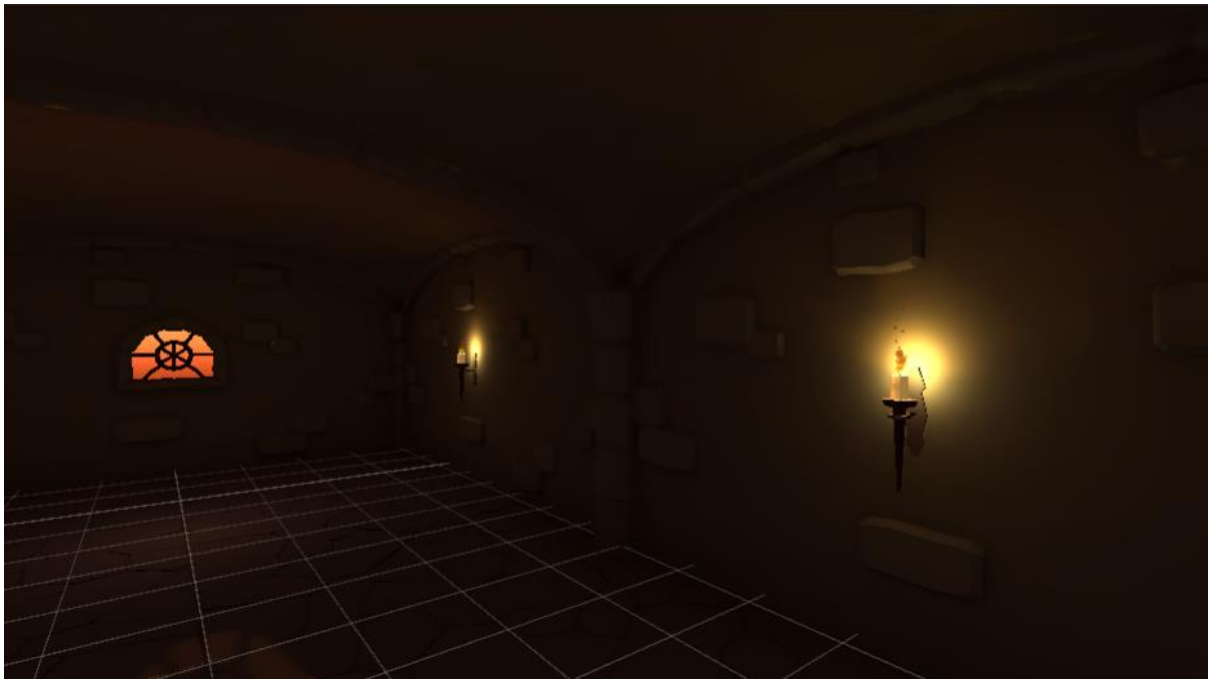


Figure 13: Room used in the Main Menu scene, showing how each wall, floor and ceiling tile can be placed together to form modular interiors.

4.2.4 Process design

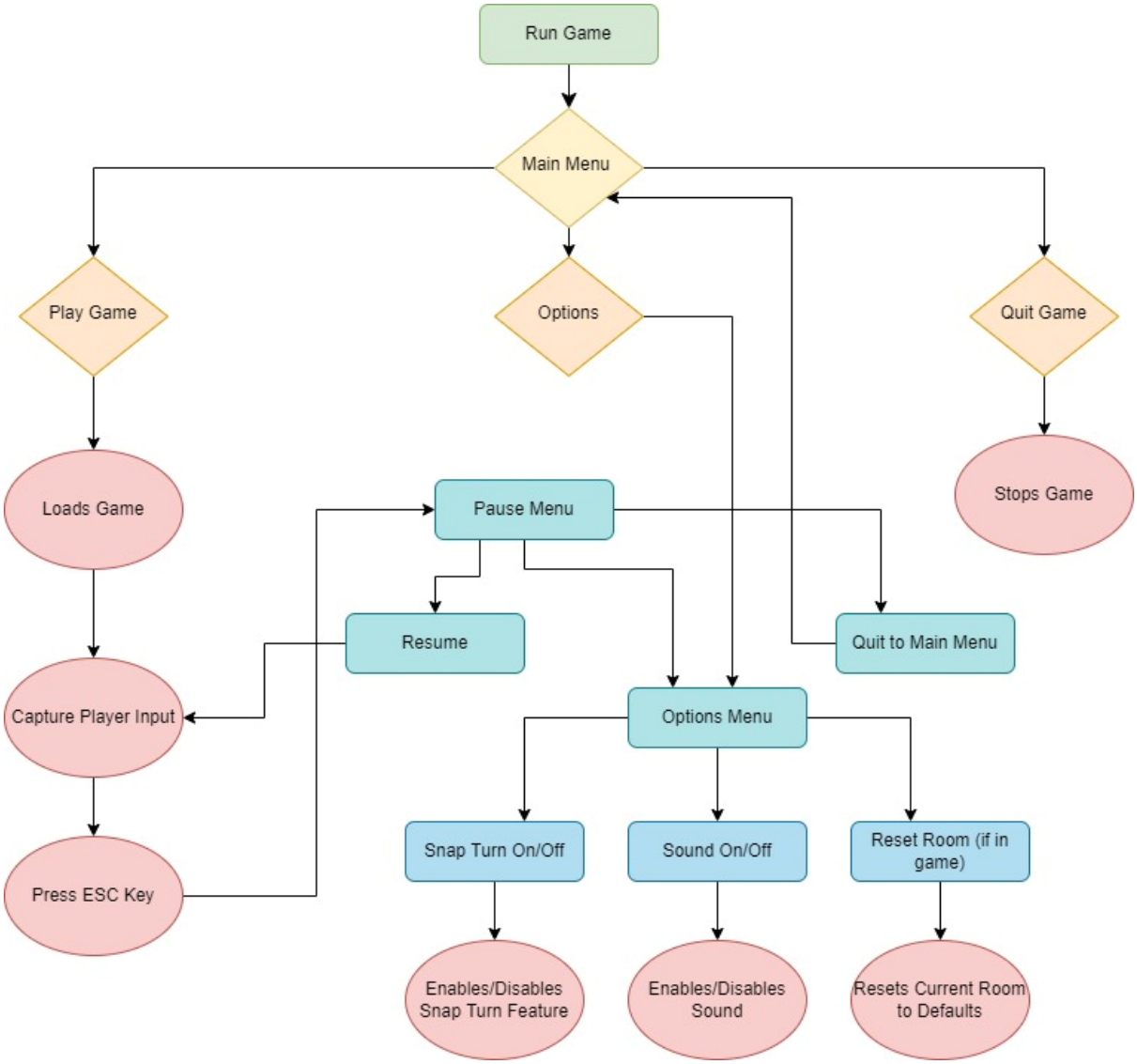


Figure 14: Main processes performed in the application.

4.3 User interface design

This section deals with the process behind the design of the game's user interface (UI). It also discusses design choices, such as fonts and colour palettes, and details why the final choices were made.

4.3.1 Wireframe

To conceptualise the main screens shown in the game, simple wireframes were drawn up. First, a main menu was drafted, showing the choices of Play, Options, and Quit. This main menu was envisioned as a menu "floating" above a 3D environment, similar to the game's level design, rather than a static screen. A menu panel present in a 3D world was thought to provide a better sense of immersion.

A sample view of how the game would look in first-person VR view was also drafted. The player's hands are visible to them at all times. With them, they can grab objects or perform other interactions such as opening doors.

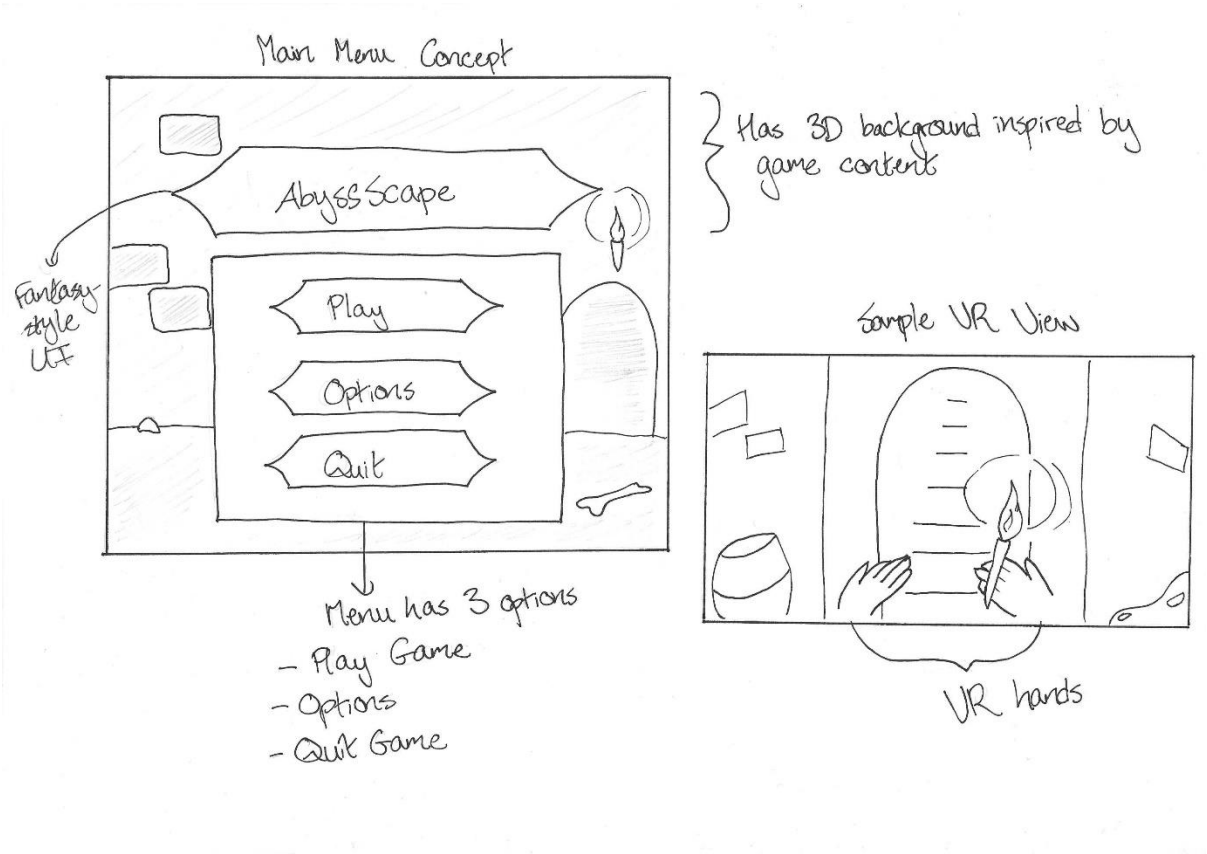


Figure 15: Wireframe showing a simple Main Menu and first-person view of the game.

4.3.2 Style Guide

Before any UI or menus could be designed/implemented, it was important to find an appropriate font and colour scheme for the game. The process behind this selection is shown below.

4.3.2.1 Fonts

For the purposes of this project, various medieval, gothic-style fonts were downloaded from the website dafont.com. These were then compared against each other; their strengths and weaknesses being evaluated before the final design was chosen.

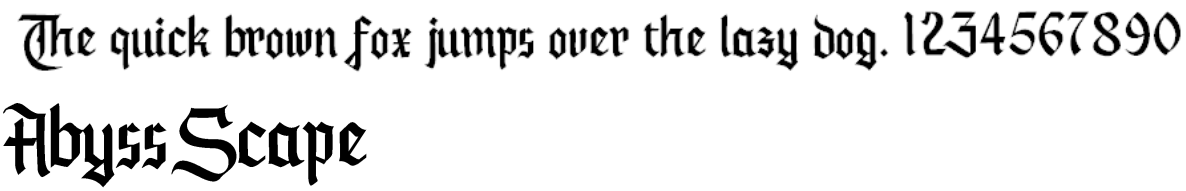
Font 1: *Dragon*, by Vunira Design (dafont.com)

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

ABYSSSCAPE

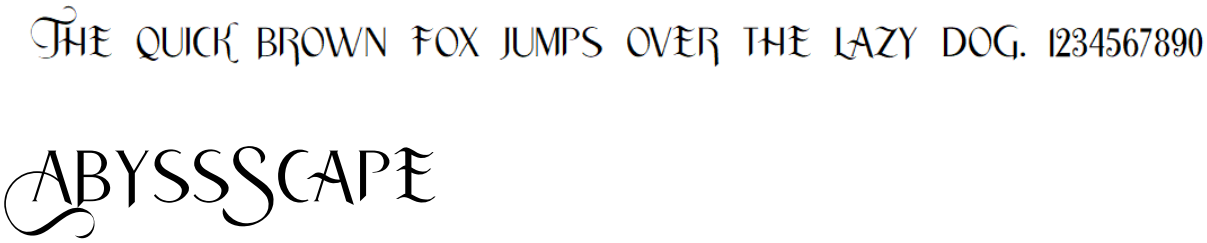
This font looks adequately fantastical and old-fashioned – both aspects the game sought to encapsulate. However, the font does not contain a matching set of numerals. It was not yet clear whether these would be needed; however, it was not chosen on account of this. It was also imagined that the decorative caps could cause readability issues among players.

Font 2: *Standrag*, by Taufiqurrohman Nadri (dafont.com)



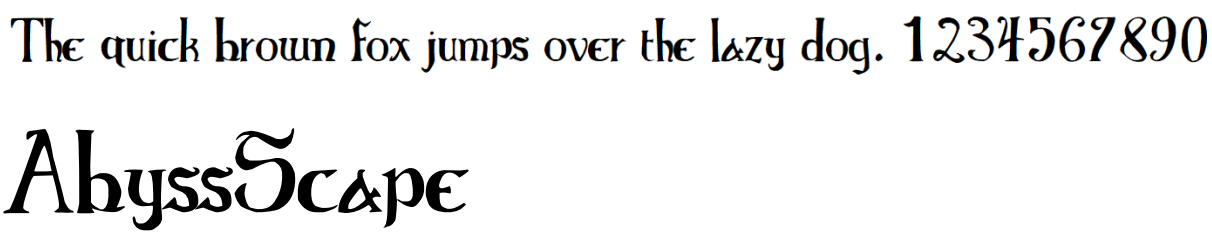
This font is reminiscent of the Gothic blackletter of old. For a game that takes place in a medieval fantasy dungeon, it certainly would not look out of place. However, it was felt that it lacked the "punch" that a game title and UI would need to grab the user's attention. There was also the issue that players might find the font hard to read.

Font 3: *Traditian*, by HansCo (dafont.com)



Though this font might have worked, it was decided against as, like the above example, it might have proven difficult for players to read. The fine strokes, coupled with the decorative accents, mean each letter looks different to a more "standard" typeface. By default, it is also a very small typeface; the size needs to be increased greatly for it to be easily read. Added to this was the fact that the numerals are all represented by narrow characters, which makes them hard to read when placed alongside each other.

Font 4: *Elementary Gothic Bookhand*, by Bill Roach (dafont.com)



This was the font that was decided on in the end. It is nicely decorative and fits with the theme, but at the same time, is legible enough so as to be understood by players. It was also used for a previous VR project, where it lent itself nicely to the theme – a dark, fantastical room, not too dissimilar to this game.

4.3.2.2 *Colour Scheme*

Figure 16: Colour palettes generated for use in UI features.

The website coolors.com was used to generate a colour palette for the user interface and menus of the application. The first palette was randomly chosen based on what would complement the tone of the game – mainly neutral, earthy colours. The second palette, however, was generated using colours directly from a screenshot of a scene. Colours such as that of the dark walls, the glowing torches, and the ominous orange sky were all selected.

Some of these colours did get used in the form of lighting and ambient features; however, a premade UI pack was downloaded for use in menus which did not contain most of these colours (see *UI Design* below for more detail).

4.3.2.3 UI Design

A prototype of the different menus encountered by the player was designed. First was the main menu, which lets the player choose between playing the game, configuring options, or quitting the application. A diagram of the main menu, and the options screen accessible from it, can be seen below.

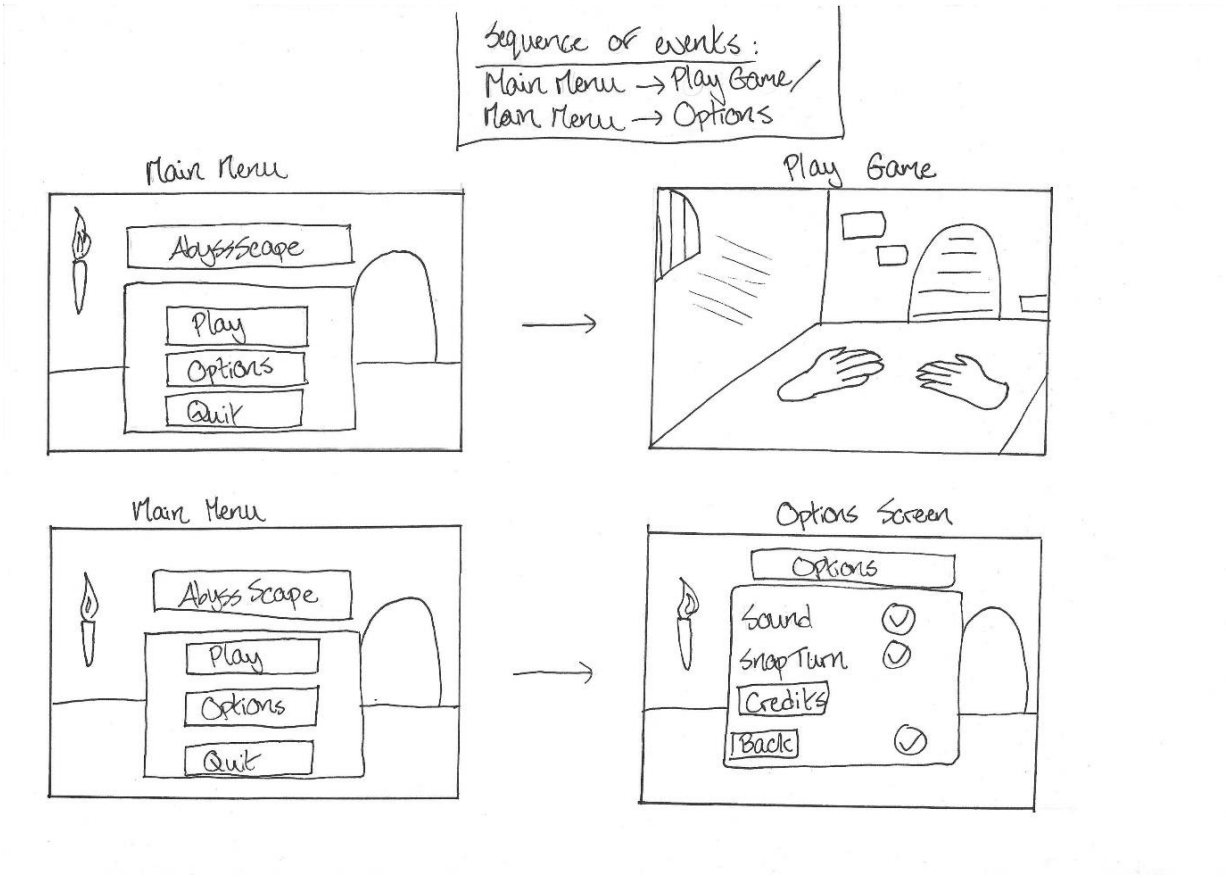


Figure 17: Main Menu and Options Screen processes.

When the player presses the pause button, the pause menu appears. From this, they can configure options, restart the game, or quit to the main menu. However, this was designed differently to the main menu screen. It was decided, in keeping with the game's provision of an experience as immersive as possible, that the game's pause menu should be implemented in the form of diegetic UI. Diegetic UI is a form of user interface in which the interactable elements are presented as part of the game's environment or characters, rather than an on-screen overlay – examples being the hologram inventory of *Dead Space*'s Isaac Clark, or the handheld map of *Far Cry 2* (W, 2018). This form of UI lets players perform the same actions as a standard interface, but with the added bonus of feeling completely immersed in the game world.

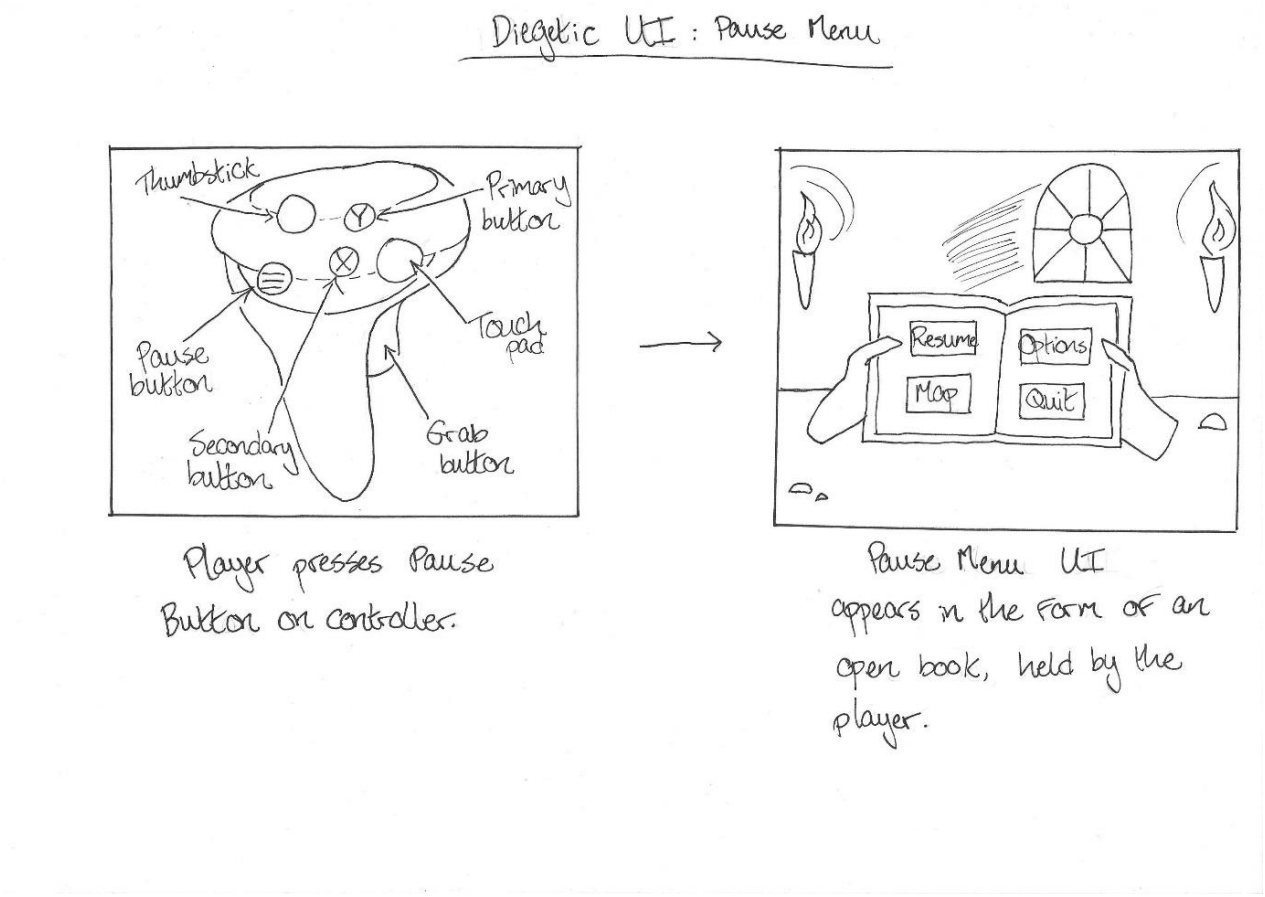


Figure 18: Diagram showing the process of pausing the game.

A package was downloaded from the Unity Asset Store to help with the UI creation and layout. It was felt that having pre-generated assets would speed up the tedious process of laying out UI elements from scratch. Thus, a UI set that both stylistically and thematically fit the game was purchased for the project.

Figure 19: The asset pack utilised for UI features, such as the Main Menu.

Due to time constraints, however, the pause menu was ultimately left out of the final product. Though it would undoubtedly be a good quality-of-life addition to the game, it was not felt that excluding it would negatively impact the game's core gameplay or performance.

4.3.3 Storyboard

Storyboarding is a vital part of the design process for any game. Developing further upon the ideas presented in the prototypes and wireframes, a storyboard was drawn out. This storyboard showed, in more detail, how the player would progress through the game, and which obstacles they would face along the way. It also showed what would happen when a certain action was performed, eg. when the pause button was pressed.

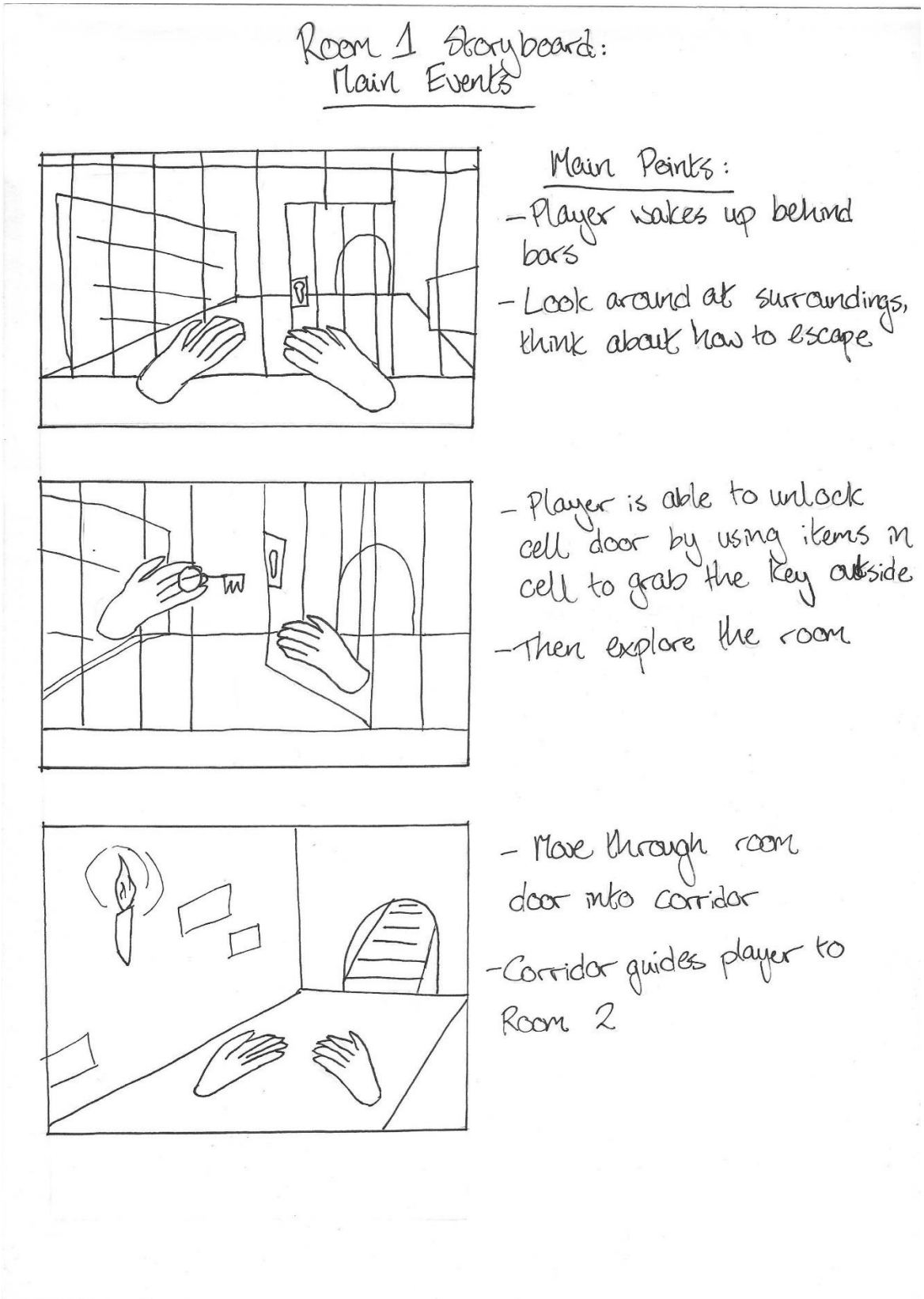


Figure 20: Simple storyboard detailing the player's first task.

The above storyboard details the main points of the first room encountered by the player. It shows them waking up behind bars, being able to escape their prison (with the help of some items found in

their cell) and, finally, being able to leave the room into the corridor beyond. From here, they must progress to the next room.

The process involved in escaping this room is shown in the below storyboard.

Figure 21: Storyboard detailing the player's second task.

4.3.4 Level Design

To get a loose idea of how the dungeon, with all its rooms, would be structured, a dungeon generator tool, available on the website https://donjon.bin.sh/5e/, was utilised. Though this site and its various tools are commonly used to generate ideas for the popular role-playing game, *Dungeons & Dragons* (Wizards of the Coast, 1993), it was felt that it would prove a useful resource when laying out the world of this game.

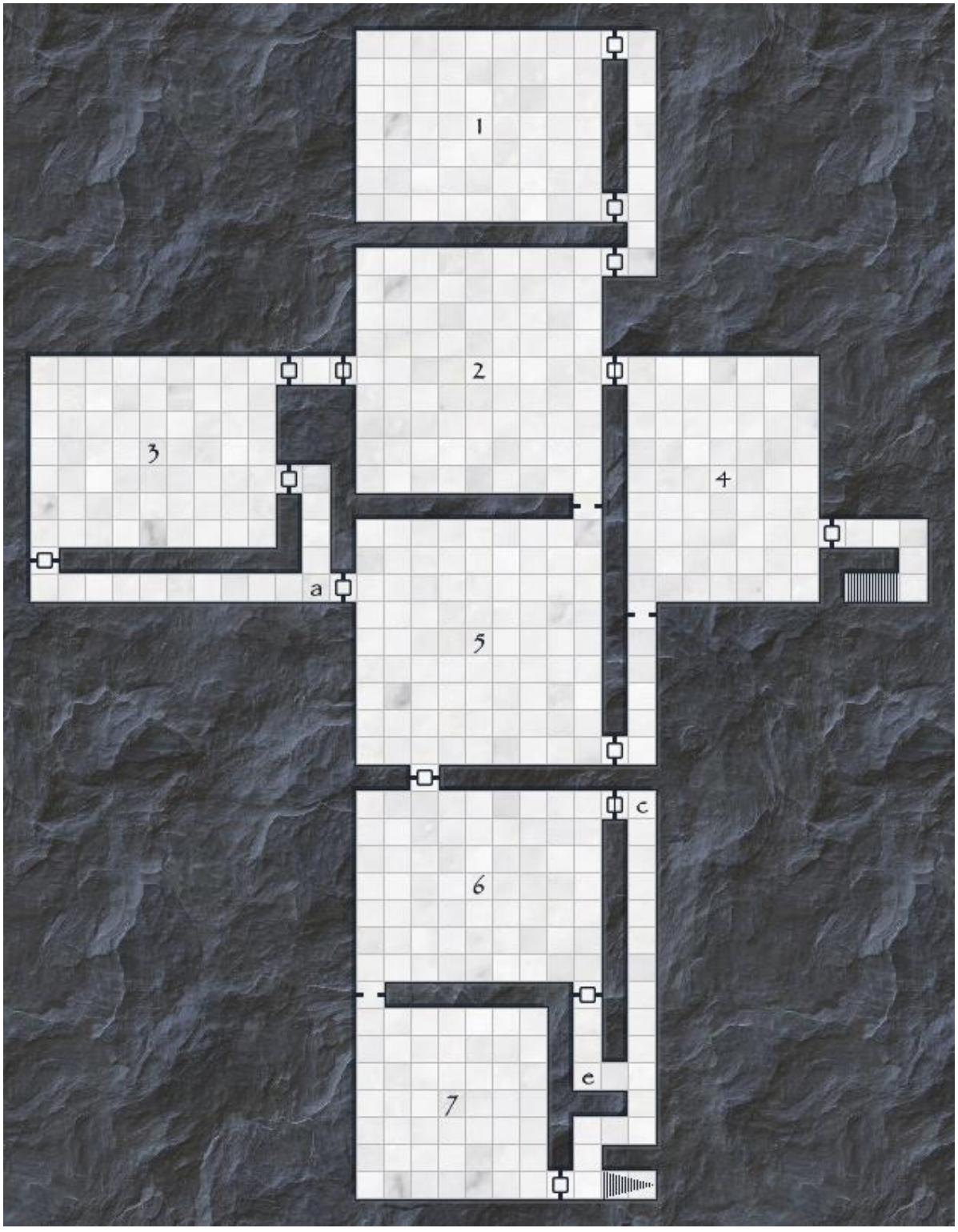


Figure 22: Dungeon map as a visual aid for the building of the game.

Unlike the above diagram, the final game was not intended to have seven rooms. Each would be quite large in and of itself, with a variety of tasks to complete to progress. The game was envisioned to only contain two to three rooms at maximum, with smaller corridors between. However, the room order, corridor placement, and overall layout of the generated example gave a good indicator as to how the player would navigate through the final game. It made the progression of the game much easier to visualise before refinements could be made.

A simple layout of each room could then be sketched out based on the above diagram.

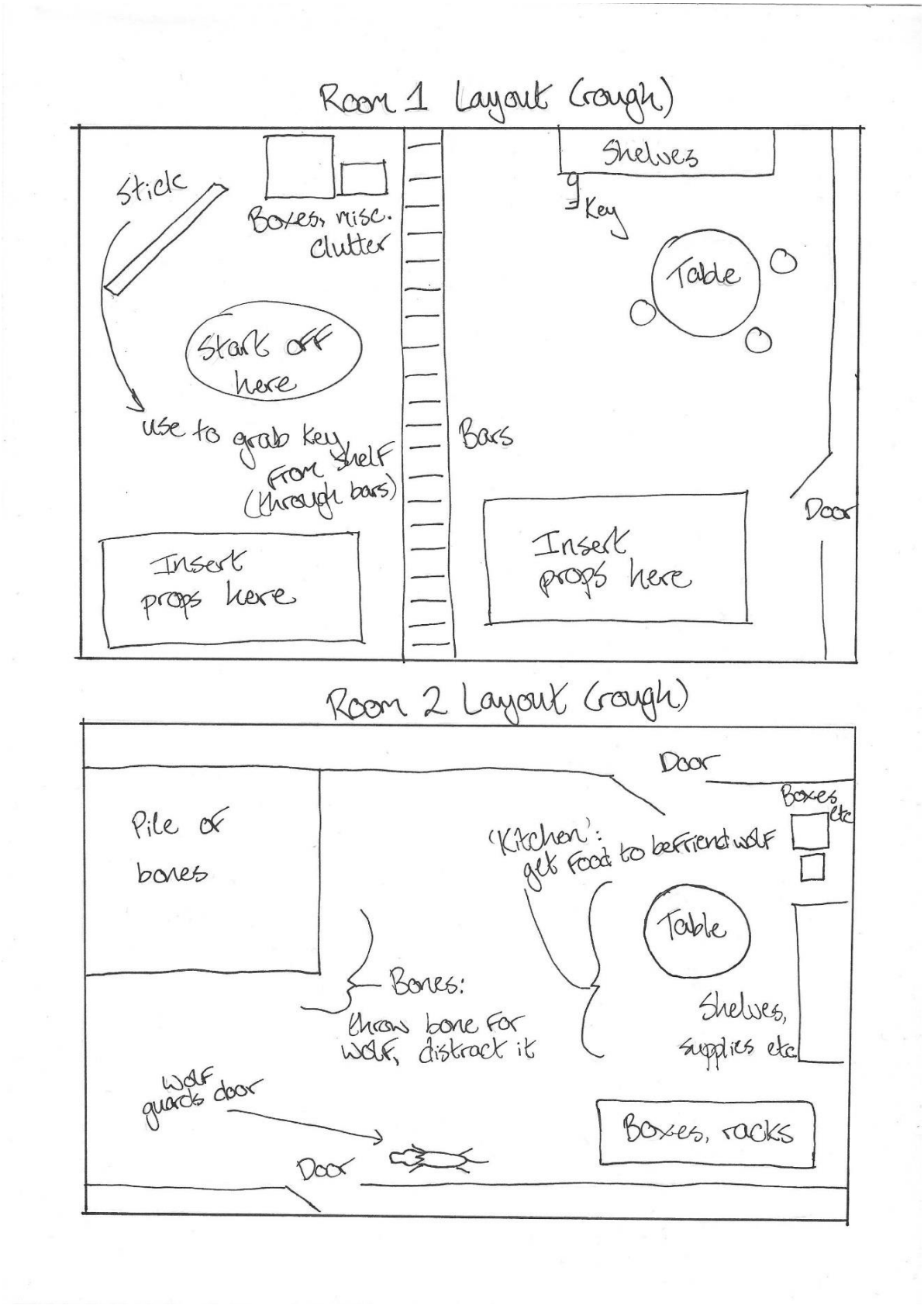


Figure 23: Bird's-eye view of two main rooms encountered by the player.

4.3.5 Environment

To envision the overall look and feel of the game environment, a mood board of ideas was created in Photoshop (photos taken from unsplash.com).

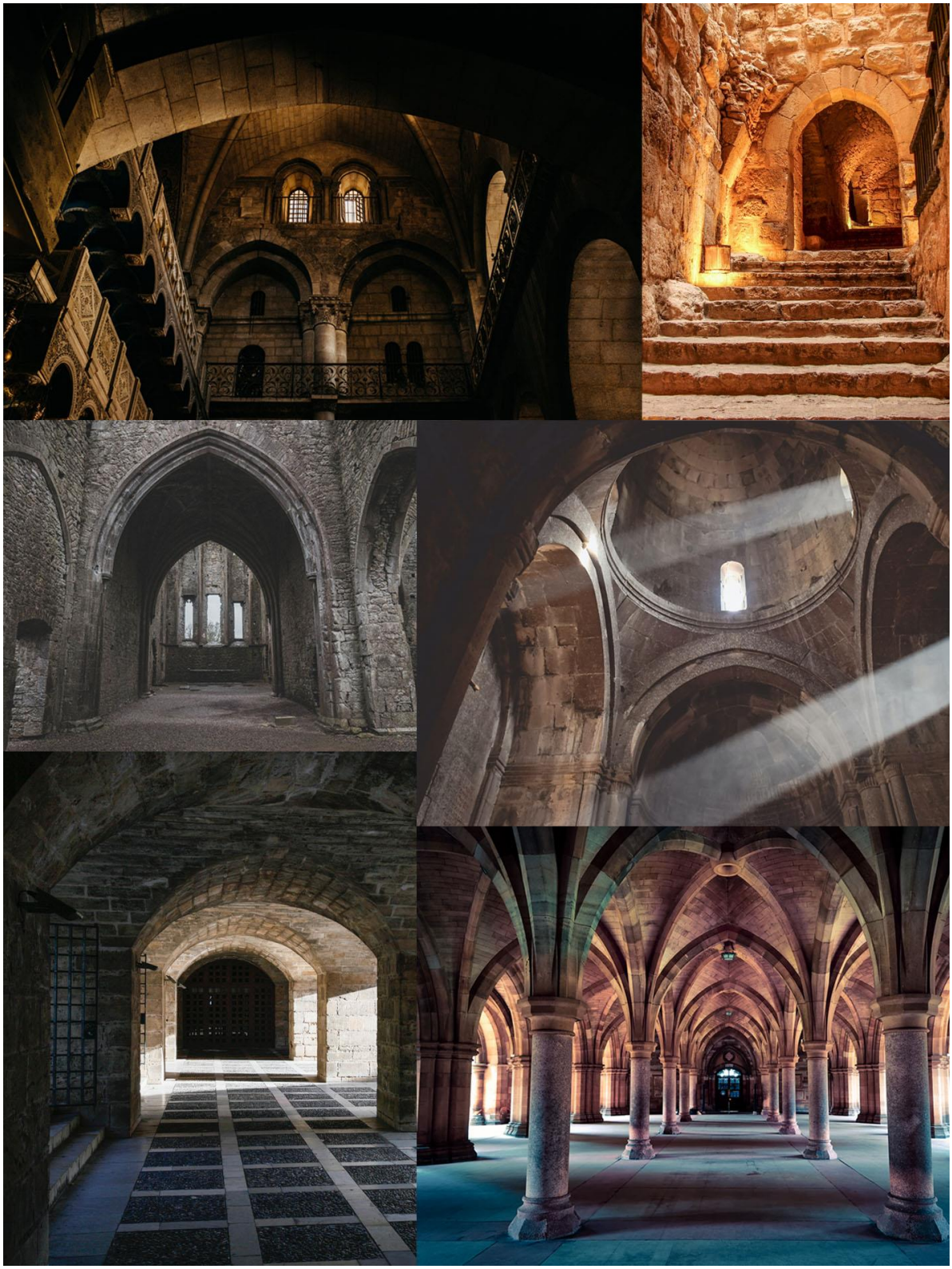


Figure 24: Mood board created to aid the game's visual development.

These images helped to envision the game environment as a tangible, extant place, rather than a fictional creation. The places in the photos also provided a great reference for how each part of the dungeon could be designed to look unique, with each room different to the last.

Though the final game levels had not yet been designed, a simple test scene was then created in Unity, both to create a simple prototype level, and to test the VR controls. Part of this sample scene can be seen below.

Figure 25: First sample room created.

Low-poly dungeon assets were imported from the Unity Asset Store and Sketchfab (sketchfab.com), and loosely placed together to get an idea of how each room could be built. It was also a useful exercise to see how lighting, skyboxes, and props could be used to set an appropriate tone for the game.

Later in the game's development, more puzzles/activities for the player to complete were drawn up. It was felt that the initial obstacles felt too rudimentary and might bore a more experienced player. The creative process, and ideas behind these new challenges, can be seen below.

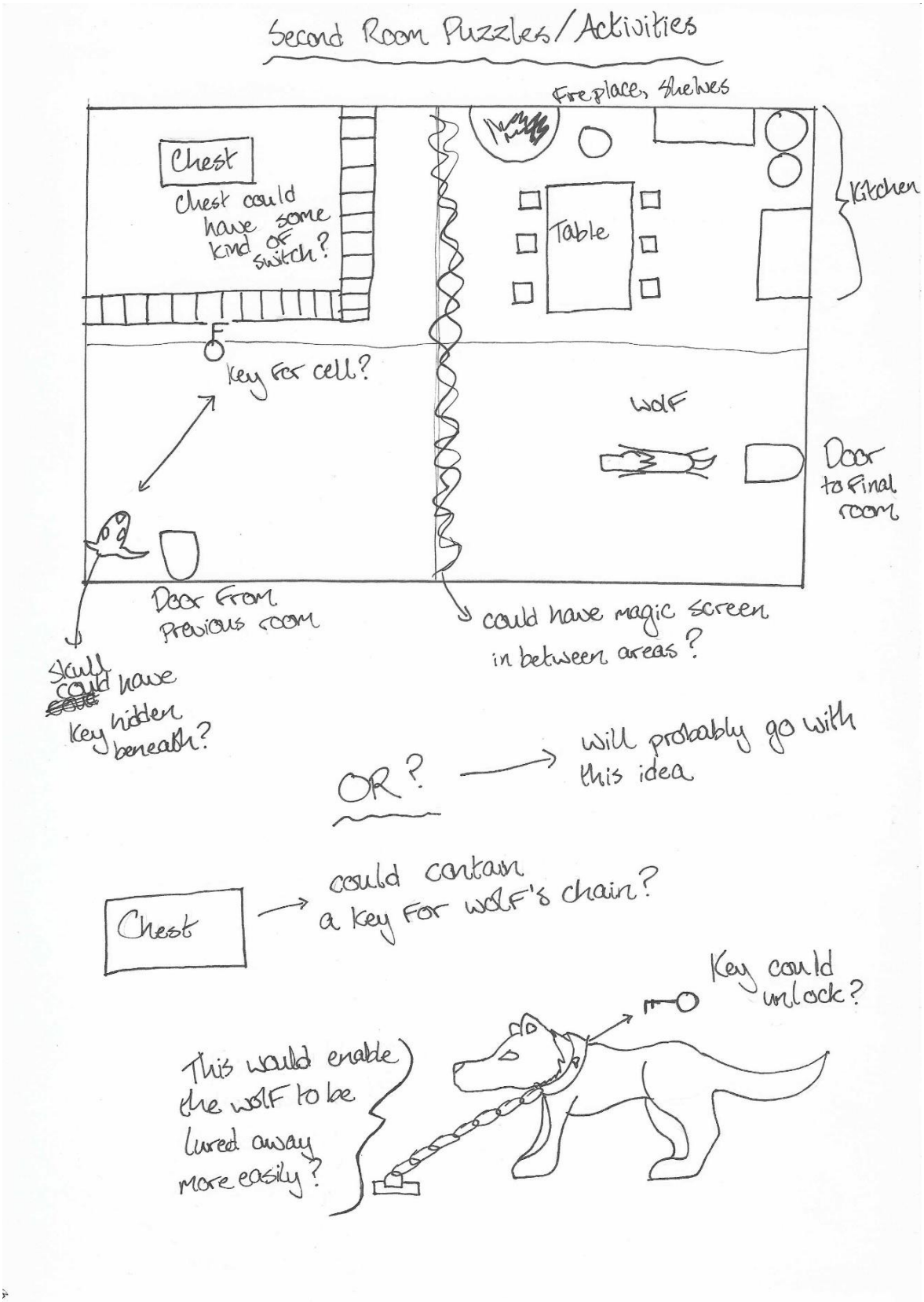


Figure 26: Continuous iteration on game's processes led to new challenges being designed for the player.

As can be seen above, the room was designed to include an extra set of challenges. It was no longer the player's goal to simply find a way around the guarding wolf, but they first had to figure out how to access the area which would allow them to do so.

This design was then iterated upon. Instead of having an extra object to find and use – a process already familiar to the player – it was decided that there should be another form of puzzle present in the room. The ideas for this are shown below.

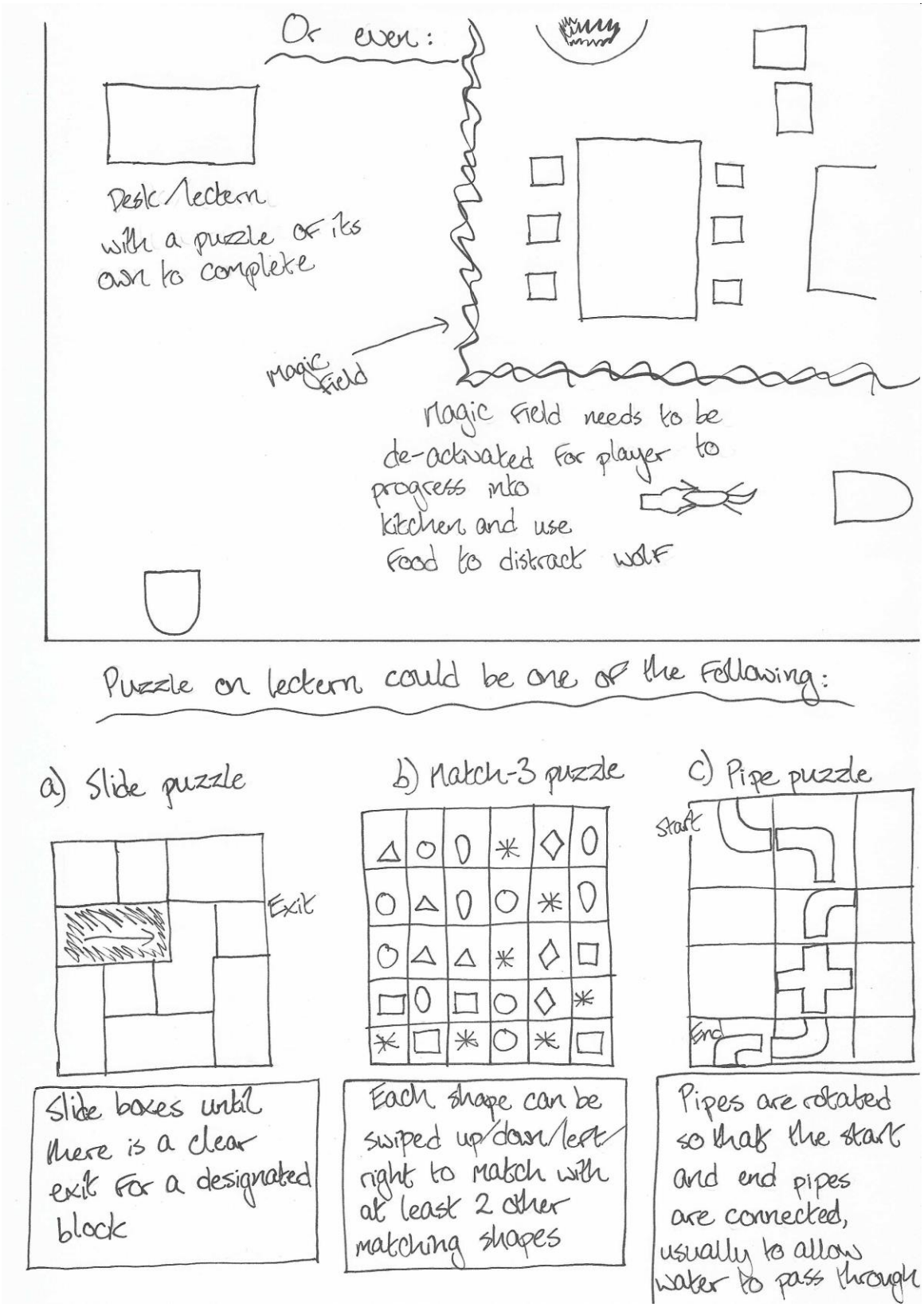


Figure 27: Further examples of new puzzles that could be presented to the player.

None of these puzzle concepts were implemented in the final product. However, the concept of solving a puzzle to access the next part of the room remained in place, with the chosen puzzle instead taking the form of a riddles quiz (see Implementation chapter).

4.4 Conclusion

There were a lot of important design aspects to take into consideration when designing the main components of *AbyssScape*. This chapter has discussed the design processes involved, including:

- The programs used to design and code the game
- How the player would navigate through the game
- Design ideation, and iteration
- Appropriate styling of elements such as fonts, colour scheme, and UI
- Storyboarding, level design, environment design, and puzzle ideation.

5 Implementation

5.1 Introduction

The application developed for this project is a virtual reality escape room game called *AbyssScape,* developed inside Unity 2020.3.25 and coded using C#. It also utilises Unity's inbuilt XR Interaction Toolkit. Unity is a game development engine that allows for the design, creation, and programming of many types of game, 2D or 3D. The programming language C# (C Sharp) is commonly used alongside Unity for programming of in-game assets; however other languages, such as Java or Lua, may also be used. The XR Interaction Toolkit enables the implementation of many VR features, such as the mapping of controls to specific XR controllers, or interaction with objects.

5.2 SCRUM Methodology

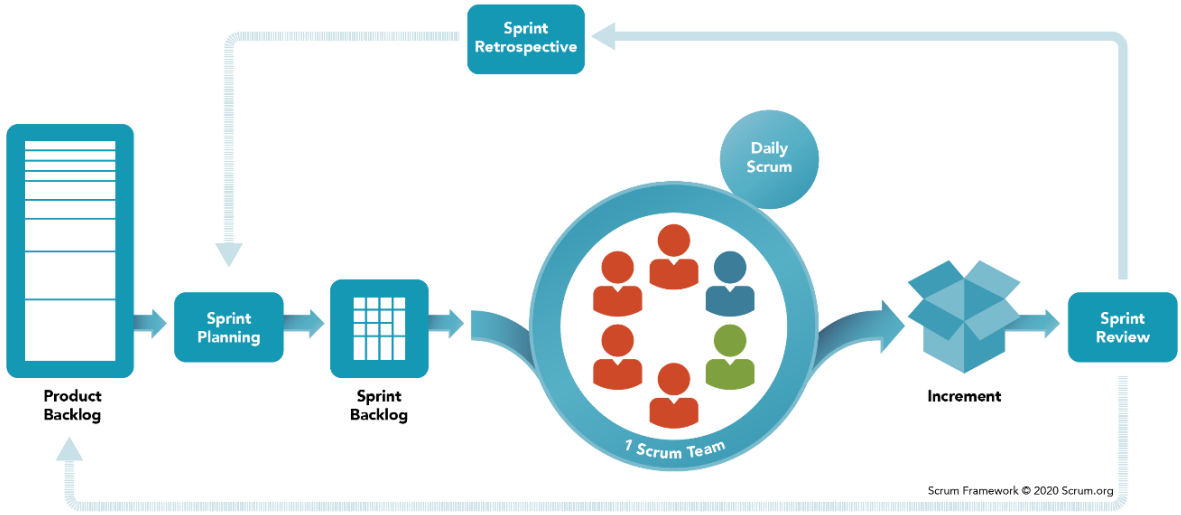


Figure 28: SCRUM methodology (source: https://www.scrum.org/resources/what-is-scrum)

The SCRUM methodology was used for the implementation phase of this project.

SCRUM is an agile development methodology, commonly used in software development. It works by first breaking down products into tasks, which are placed on a list and are then taken from the list in the order they are to be done. These tasks can then be handed to teams to work on within set timeframes known as "sprints".

The implementation phase for this project consisted of 8 sprints in total. These all took place in the second semester. Each sprint took place over a period of 2 weeks.

The requirements for the application were listed in a product backlog. Each item on the product backlog was broken down into a series of tasks which formed a sprint.

5.3 Development environment

The visual development of the game took place inside the Unity editor, specifically version 2020.3.25f, whereas all the code was written inside the Visual Studio Code environment.

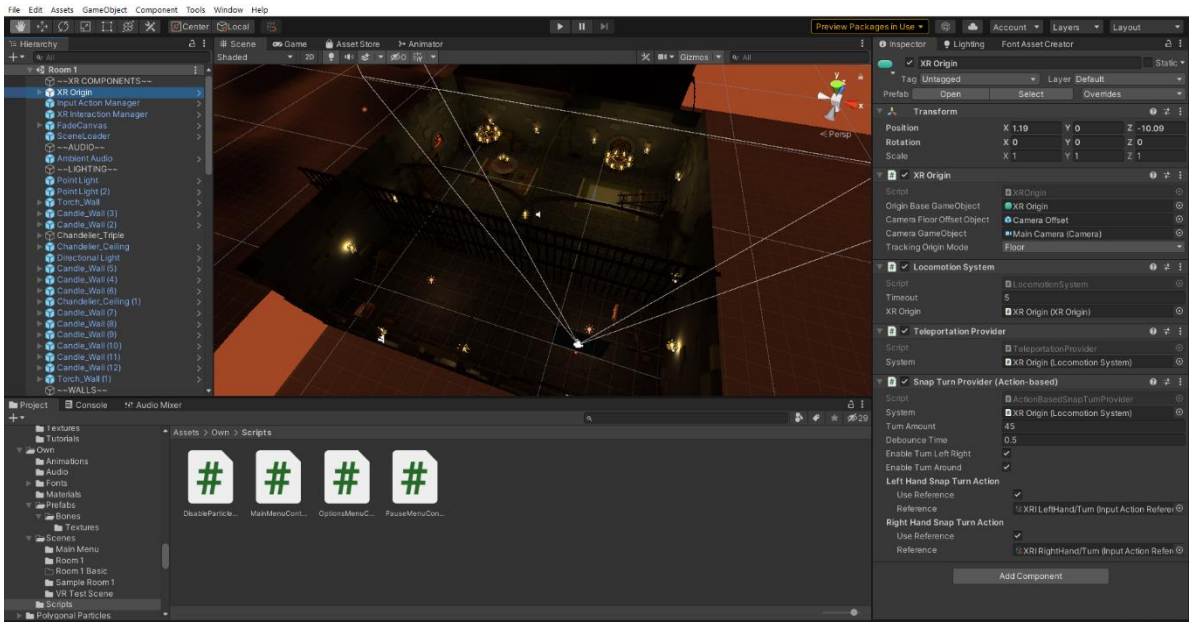


Figure 29: Unity interface.

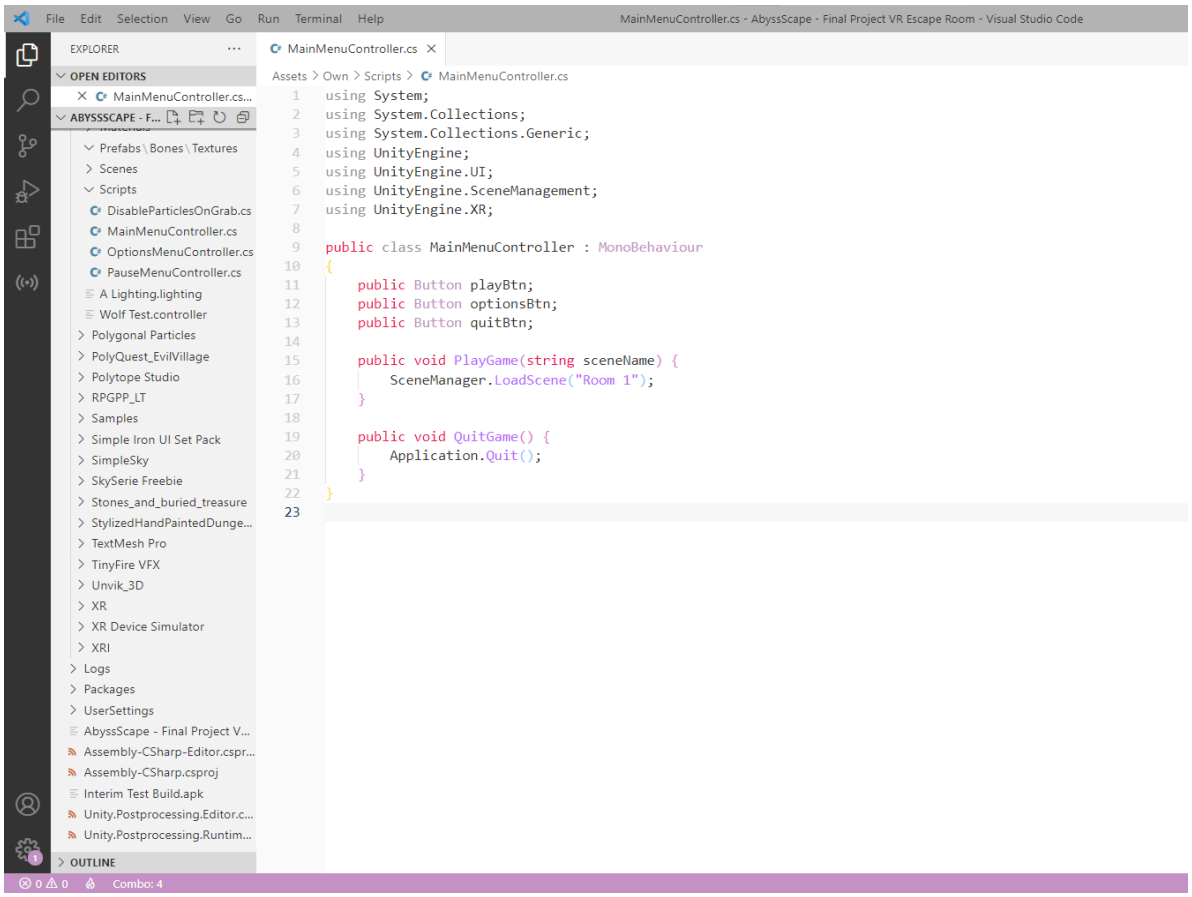


Figure 30: Visual Studio Code interface.

Though GitHub is a useful tool in software development, it was not utilised throughout this project – firstly, because this was an individual project, and there was no need for external collaboration;

secondly, because GitHub is not optimised for large files. Often, Unity projects, especially those containing 3D assets, can grow to quite a large file size, and GitHub has a limit of 100MB without the use of external tools, such as Git LFS (Large File Storage). It is also worth noting that work on this project was completed mainly from home, rendering the use of file-sharing and/or version control tools unnecessary.

5.4 Sprint 1

During Sprint 1, designing and coding were not paramount. Instead, the focus remained solely on research and requirements gathering. During this phase, project management tools, such as Trello, were also set up. More on these can be found under the Project Management section.

5.4.1 Research

The first item on the agenda was to conduct research on the chosen project area.

Virtual reality is a new and exciting, but also extensive, area. However, the concept of immersion in relation to games is nothing new. In fact, the very concept of immersion itself is as old as oral storytelling traditions.

A literature review on the sources curated and gathered for this research was carried out. This review explored the history of immersion, how it can apply to games, and how immersion can only continue to evolve with the advent of AR and VR.

More information on this can be found under the Research section.

5.4.2 Requirements Gathering

After the topic was researched, the next step was to find out what would be required to build an immersive VR game.

The first step was to find similar games/applications to that envisioned of *AbyssScape*. Games with a similar objective – solve puzzles and/or overcome obstacles to progress and escape – were researched. Three similar games were:

- *I Expect You to Die*
- *Conductor*
- *Obduction*

The mechanics, gameplay style, and even graphical style of each was examined to see what made them appealing to players.

More information can be found under the Requirements section.

5.4.3 Creating a Sample VR Room

To examine how the game would be built, and to test the components of the Unity XR toolkit, a very basic sample scene was built inside Unity. This room was built using assets downloaded from the Unity Asset Store. A modular low-poly dungeon pack was downloaded and imported into the

project. Assets from this package were utilised to put together a basic room that comprised of walls, halls, and doorways, as well as light sources such as torches. After testing out different skyboxes, one was chosen that was deemed suitable for the ominous, foreboding atmosphere the game intended to evoke.

Figure 31: Basic room concept inside Unity.

An XR Rig was also added to the centre of the room, and configured as necessary. This was tested with the Oculus Quest 2 headset, and was found to work as required.

5.4.4 Revisiting Previous Tutorials & Examples

To refresh the knowledge and understanding of the elements required to develop *AbyssScape,* previous Unity VR tutorials were revisited. Also revisited were previous VR examples, such as the interactive virtual room created for a previous assignment.

5.5 Sprint 2

During this sprint, further research was conducted. In addition, basic visualisations of gameplay and a user interface were drawn up and iterated upon.

5.5.1 Conducting Further Research

In addition to examining similar games, a survey was launched. This sought to obtain a clearer understanding of what players felt is the most important aspect of games. Respondents were also asked which aspects they found to be most important in virtual reality games, and if there were any aspects they disliked about virtual reality. Basic interviews were then conducted to reinforce the information collected.

Once results came in, a clearer picture of the important aspects to games, and in particular VR games, began to emerge. From this information, personas could then be created. Like the survey and interviews, these personas reinforced which aspects to prioritise during the game's development.

5.5.2 Beginnings of Design Phase

Before a start could be made on creating the game, the look and feel of the game, as well as the various screens encountered by the player, had to designed.

A layout concept and mood board were created to aid in the design process (see Design chapter).

Beyond the conceptualisation phase, designs for the various game screens were drawn up. The first of these was a main menu UI, as well as a basic overview of how the game would look in first-person VR view (see Design chapter).

5.6 Sprint 3

During this sprint, important design elements, such as storyboarding gameplay processes, level designs, and UI concepts, were finalised. A Main Menu system was also started on, though it would not be functionally programmed until later.

5.6.1 Level Design

It was conceptualised that there would be about two to three rooms in total in the dungeon. Thus, two basic rooms were designed as a starting point. These top-down views helped visualise which items to place in each room, what challenges the player would face, and how they would progress to the next room (see Design chapter).

5.6.2 Storyboarding

Before any of these gameplay features could be implemented, however, it was important to think about the flow of the game and its processes. Processes crucial to the core gameplay (such as locating objects, navigating obstacles) were drawn up in a storyboard format. These helped visualise how each process could be designed and coded (see Design chapter for more detail).

5.6.3 Main Menu UI

As the basic UI had already been designed, a Main Menu scene was created. A UI pack, stylised to resemble iron, was purchased from the Asset Store, and the buttons, frames, and panels contained within were used to lay out a series of menus.

Figure 32: The Main Menu.

In addition to this main menu, three other screens were implemented. These were the Options screen; the Help screen, accessible from the Options menu, which explains the game's controls; and, finally, an introductory screen, toggled when the player presses the "Play" button. This introductory screen provides some backstory to the player's situation, and why they must escape the dungeon in the first place.

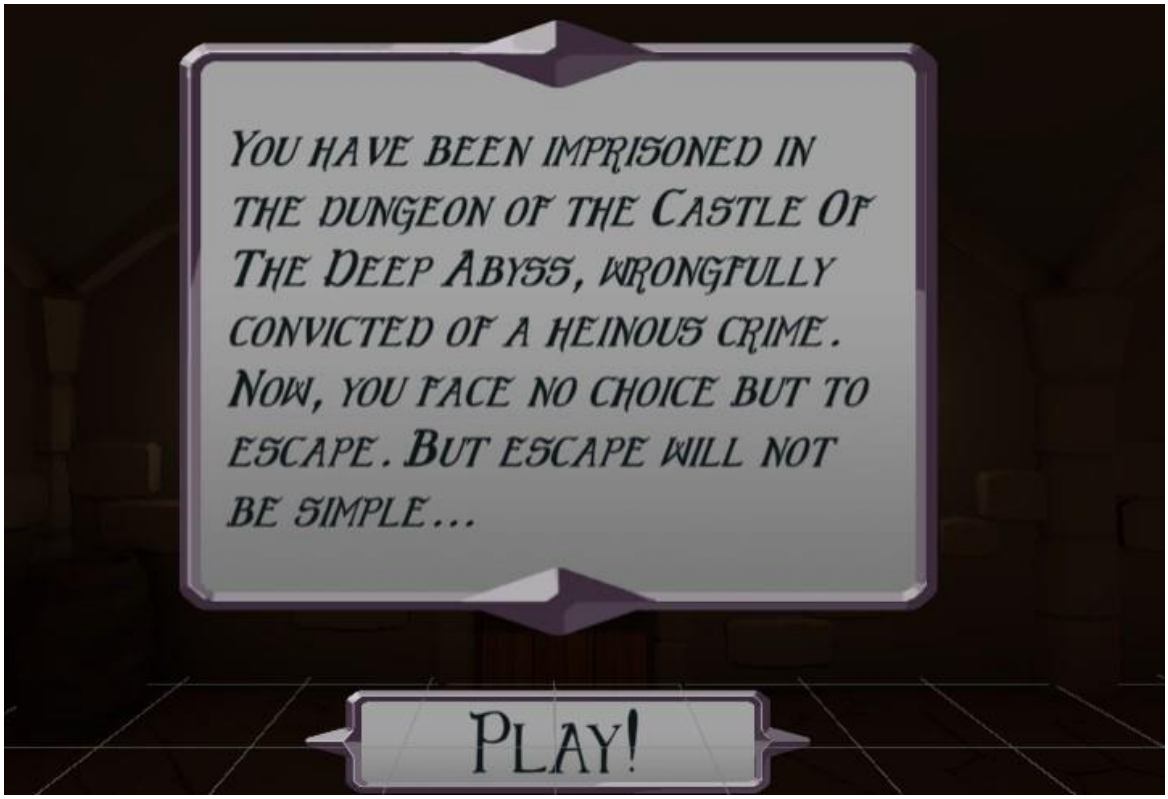


Figure 33: Intro screen, which appears when the player presses "Play" from the main menu.

Figure 34: Options menu.

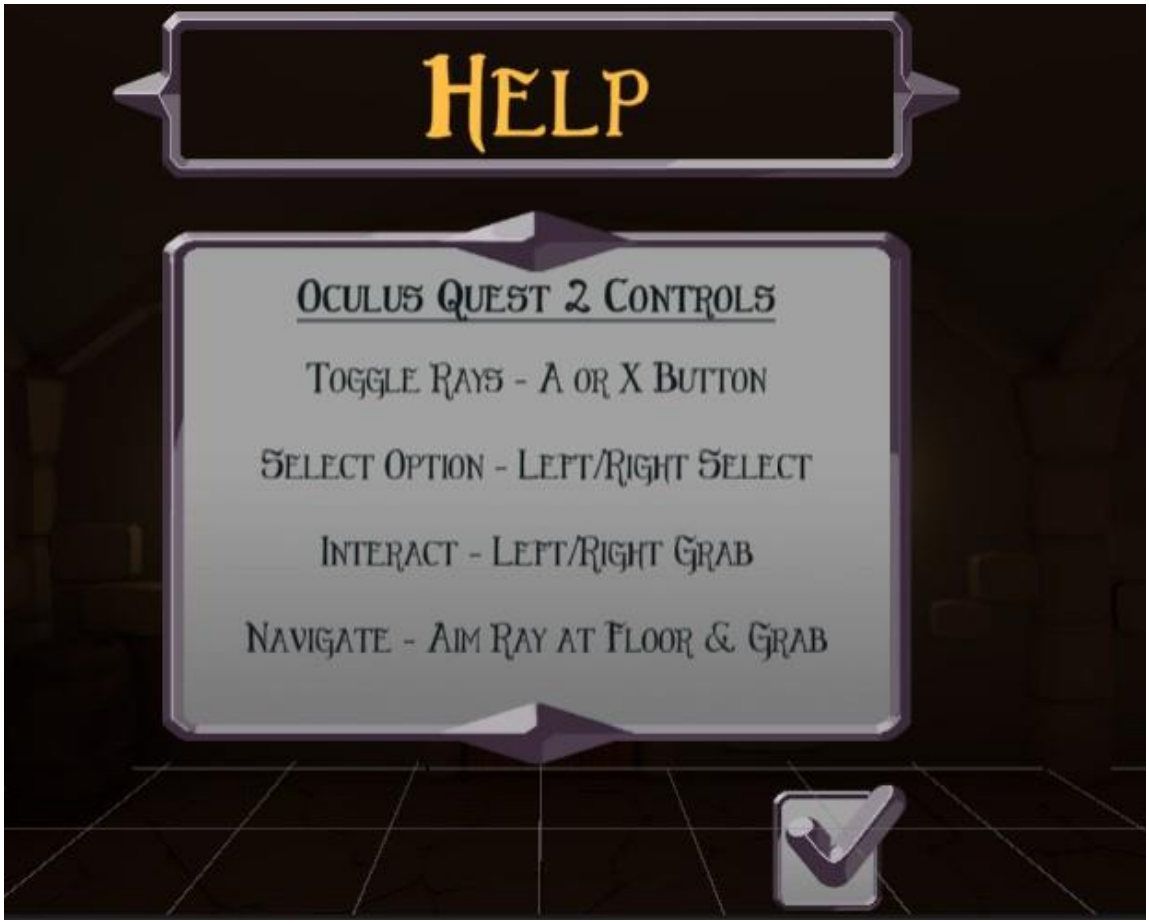


Figure 35: Help screen, explaining the game's controls.

Though the Main Menu itself had not yet been coded, each screen was toggled visible or invisible by the GameObject.SetActive() function inside Unity. When the player selected the Options button from the Main Menu, for example, the Main Menu would be rendered inactive, and the Options screen would become active. Conversely, if the checkmark or X button on the Options screen was selected by the player, the Options screen would be set to inactive, and the Main Menu would be set to active once more.

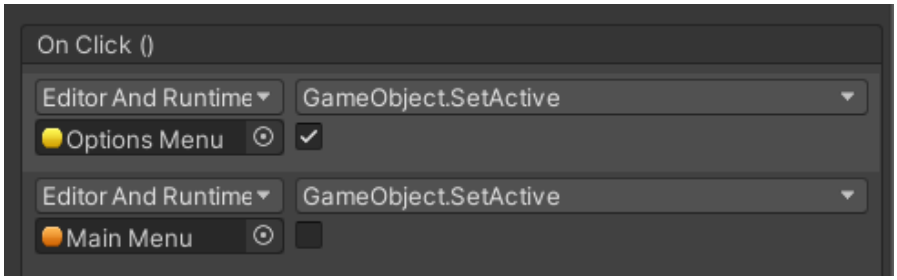


Figure 36: When the player selects Options, the Options menu becomes active and the Main Menu becomes inactive.

5.7 Sprint 4

The Main Menu, now laid out in full, had to be programmed for each button to perform its expected function. This sprint would also see construction beginning on the first room in the game.

5.7.1 Programming the Menu Functions & Repairing the XR Origin

To program each function of the main menu, a script called MainMenuController was created. This script contained functions for playing and quitting the game. (A function for calling up the Options menu was not needed, as this could be toggled in the Unity editor itself.)

MainMenuController.cs

Assets > Own > Scripts > MainMenuController.cs > ...

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.XR;

0 references
public class MainMenuController : MonoBehaviour
{
    0 references
    public Button playBtn;
    0 references
    public Button optionsBtn;
    0 references
    public Button quitBtn;

    //load the game scene
    0 references
    public void PlayGame(string sceneName) {
        SceneManager.LoadScene("Room 1");
    }

    //quit the application
    0 references
    public void QuitGame() {
        Application.Quit();
    }
}

```

Figure 37: The MainMenuController script, containing the functions required to load the game and to quit the game.

However, a roadblock occurred in the form of the Unity XR Rig not performing as required. The ray selectors, attached to the player's hands, could not be used to select any of these buttons. This was due to the inbuilt XR Toolkit needing to be updated. In the updated package, the XR Rig component was replaced instead with a component named "XR Origin". Though there looked to be no difference between the two, it was considered wise to remake the XR Rig from the ground up, by using this new XR Origin instead.

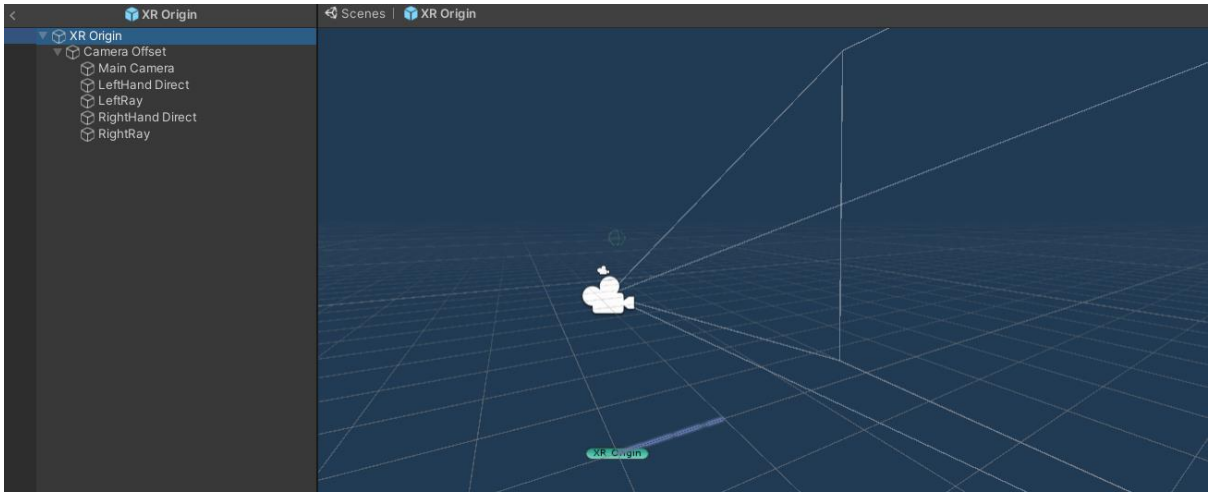


Figure 38: The XR Origin setup, with a Camera, Left Hand & Ray, and Right Hand & Ray.

This new XR Origin was created by first importing a premade Unity XR Origin object. This XR Origin contains a nested Camera Offset object, within which is the main camera that functions as the "eyes" of the player. With this set up, the position of both the left and right hand objects was copied from the original XR Rig, and new hand objects were added to the Camera Offset with these co-ordinates. New XR "ray" interactors were also added to these hands, again with the same properties as the previous ones.

Once the hands were created, Unity's XR Input controls had to be mapped to them. This would ensure that all the VR functionality of the Oculus Touch controllers could be utilised.

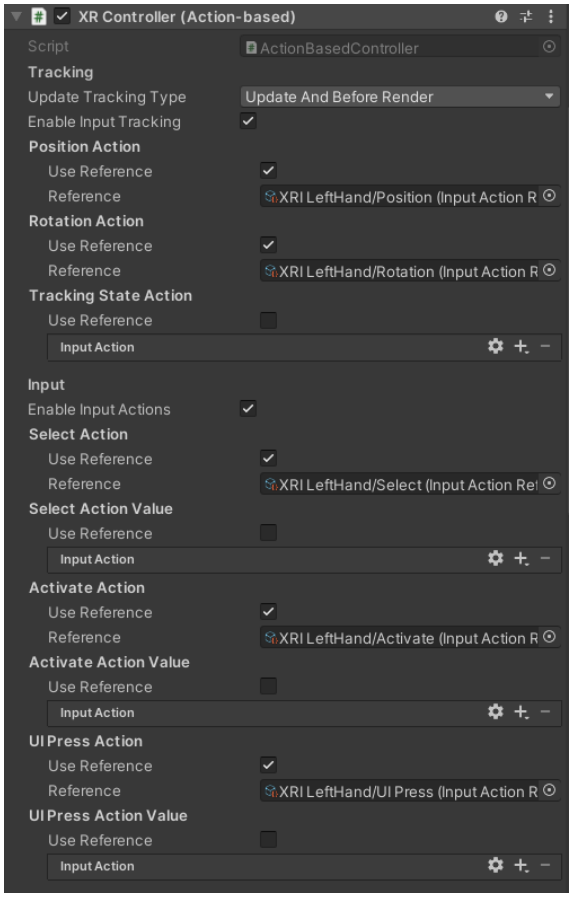


Figure 39: The XR Controller component, as shown on the Left Hand object.

The new XR Origin was then tested with the menu functionality. Thankfully, this time, it worked as planned. The rays, when toggled on by the player, were able to select each button. When Play was pressed, the scene transitioned to the game scene; when Options was pressed, the Options menu appeared; when Credits was selected, the Credits screen appeared; and when these screens were exited from, the Main Menu screen appeared once more.

The scripting of the options available was not completed until later; for now, other in-game features, such as building the game environment and implementation of obstacles, took precedence.

In addition, a short, looping piece of music was created inside MusicMaker for the Main Menu scene. MusicMaker is a program that enables the user to arrange premade music loops and, in doing so, to create film scores, soundtracks, songs, and more.

Once a suitable piece of music was composed, it was placed in the scene.

5.7.2 Building the Game Environment

Construction of the in-game level was also started at this point. Using a modular low-poly dungeon asset pack, walls, ceilings, and doors were placed together to create a singular, large room. Appropriate light sources, such as chandeliers and candles, were placed at several locations to prevent the room from appearing too dark. It was also at this point that various methods of lighting were tested. When developing for Android platforms, such as the Oculus Quest 2, setting light sources to Unity's "Mixed" or "Baked" settings is optimal, as it reduces the processing needed to render complex lighting. Consequently, all of the lights in the scene were set to "Mixed" mode, and the scene lighting was then generated through the baking of lightmaps.

Furniture items, such as a table and shelves, were also placed around the room to make it feel more authentic. In addition to this, the interactable objects required for the player to progress were placed in the relevant locations. Examples of these were a long plank of wood, with which the player could reach a key to unlock their cell, and a crystal ball that, when placed on its stand, would reveal a secret room behind an inconspicuous wall. It was ensured that each of these items contained a collider component, as well as Unity's "XR Grab Interactable" component. This is a script that comes with Unity's XR Interaction Toolkit. It enables the player to pick up items while playing in VR.

Figure 40: Long plank of wood inside cell. Scene lighting has been turned off for demonstration.

Figure 41: Crystal ball stand.

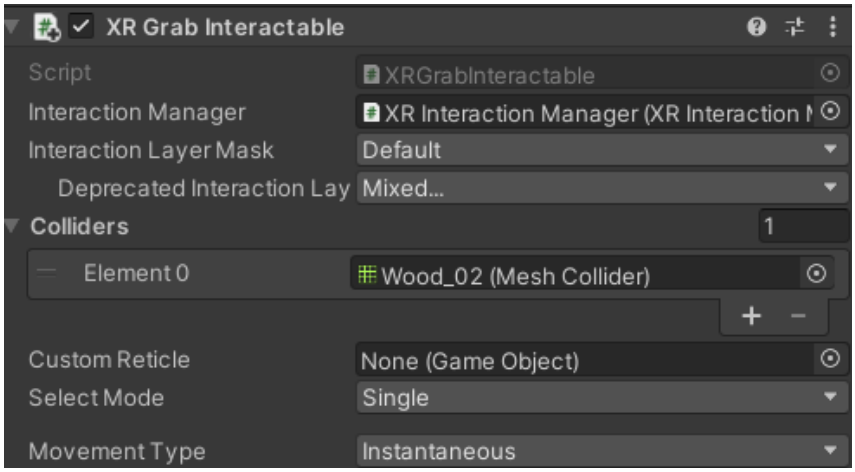


Figure 42: A snippet of Unity's inbuilt XR Grab Interactable component. This was placed on each interactable item and configured individually as required.

As the room remained quite dimly lit, however, it was imagined that these objects would not be too easy to find. To remedy this, a subtle particle effect was placed in the vicinity of each interactable object. This ensured that each object would be easier to locate, whilst not to be so obvious as to make the game too easy.

Figure 43: Interactable key. Again, scene lighting has been turned off for clarity.

Several static props were also placed around the room, to give the impression of a living, believable environment. These included a table, stools, crates, and shelves.

Figure 44: Sample shot of the first room, containing various props.

This in-game environment was constantly iterated upon and expanded throughout the game's development. Items were added, or removed, as was deemed fit.

5.7.3 Testing the Wolf Puzzle

It was estimated that, as it was a complex and animated 3D asset, the wolf set to guard the second door would take time to implement programmatically. Thus, it was critical to explore the asset's possibilities at this early stage, rather than later.

To see which animations would be needed, as well as the conditions required to transition between each, a test scene was created. This simple scene included only the wolf asset, a target for the wolf's AI programming, and a flat plane.

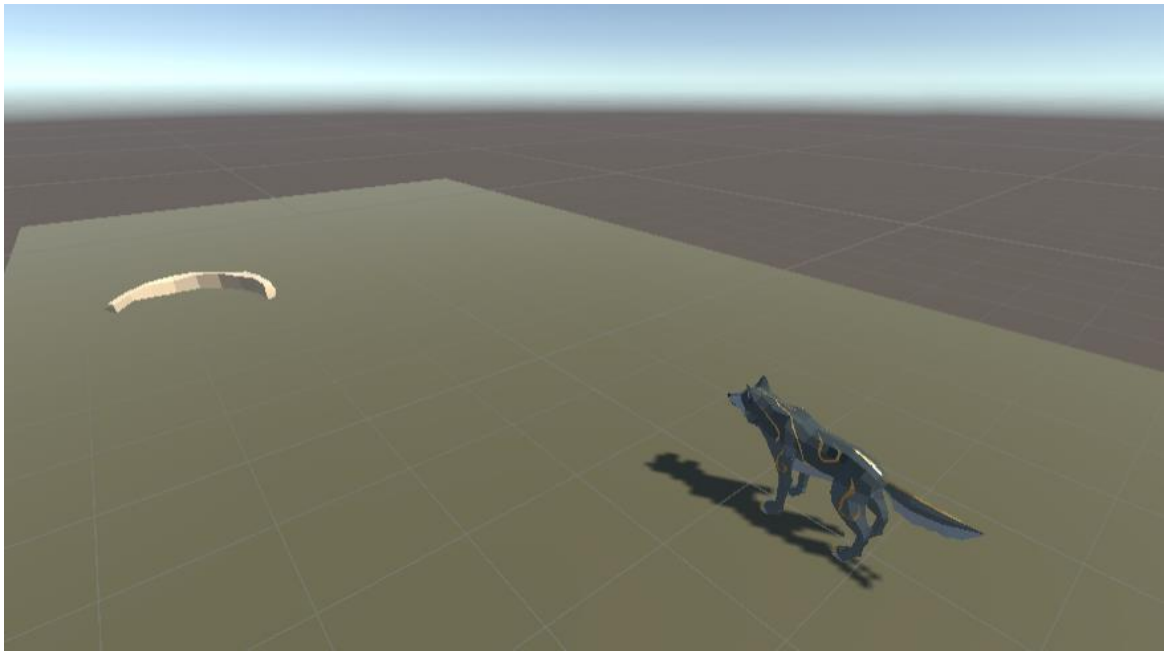


Figure 45: Wolf test scene.

To act as an AI agent, a Unity game object needs to contain a Nav Mesh Agent component. This is a component that can be used, for example, to create enemy AI that will advance on the player, or another target. In this case, the wolf was to be the agent; the bone, its target. A Nav Mesh Agent component was added to the wolf, as well as a Rigidbody component and relevant colliders.

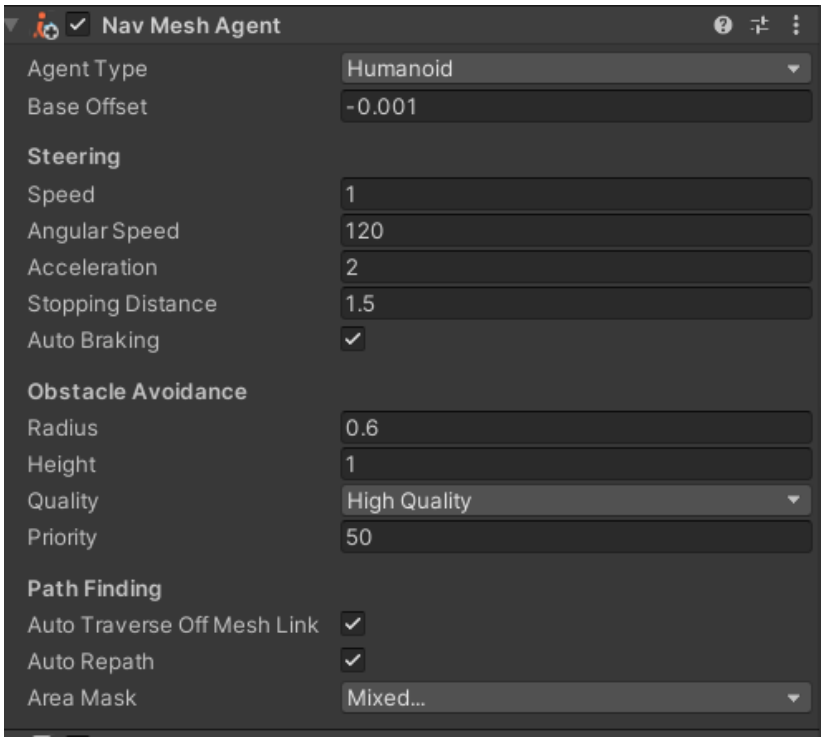


Figure 46: Unity's Nav Mesh Agent component, as it appears on the wolf.

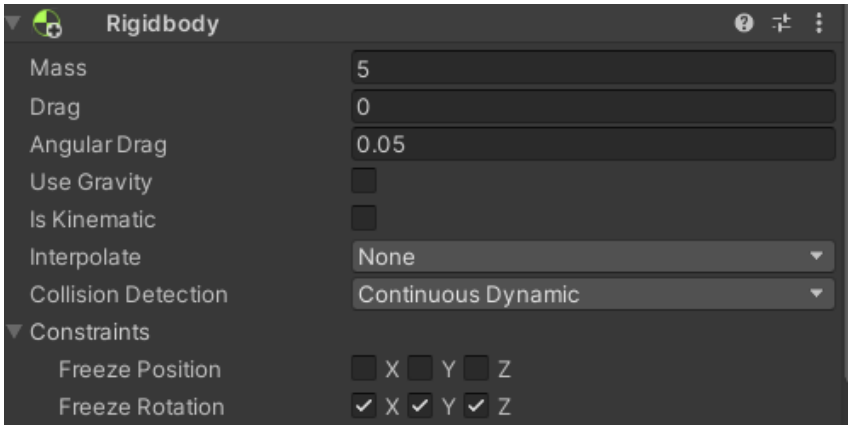


Figure 47: Unity's Rigidbody component, as it appears on the wolf.

These components also needed to be accessed through scripting if they were to function according to the desired behaviours.

```
void Start()
{
    anim = GetComponent<Animator>();
    rb = GetComponent<Rigidbody>();
    agent = GetComponent<NavMeshAgent>();
    agent.GetComponent<NavMeshAgent>().enabled = true;
    audio = GetComponent<AudioSource>();
    audio.enabled = true;

    target = GameObject.FindGameObjectWithTag("AI Target").transform;
}
```

When the game starts, each component is accessed through the C# “GetComponent” method, seen above. The target for the wolf, a variable created earlier on in the script, was also set. Through the use of the “FindGameObjectWithTag” method, the script configures any game object with the tag “AI Target” as an object the wolf will move towards.

An animator controller was set up containing the three necessary animations – an idle animation, movement, and eating. In addition, transitions were created between each of these states, containing conditions that could be triggered through code.

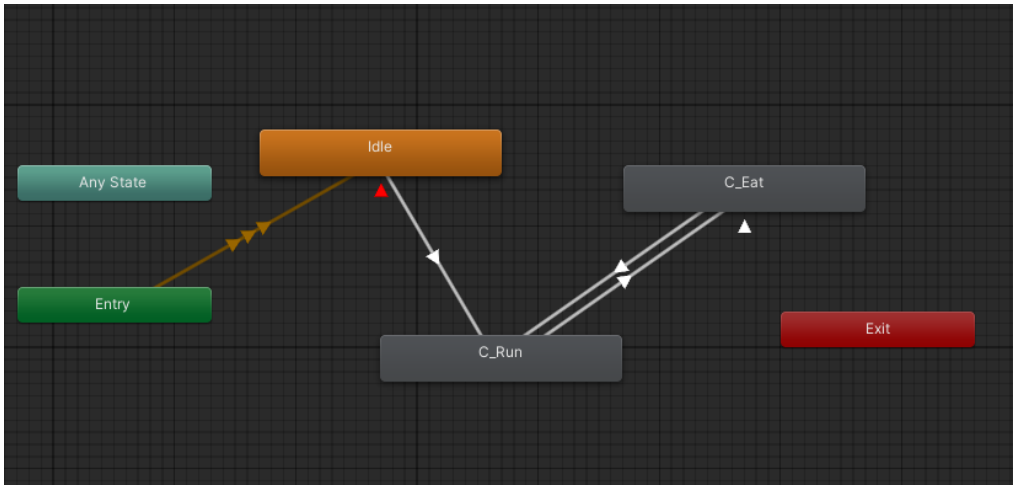


Figure 48: The animator controller used to configure the wolf's actions.

Three conditions, called “isIdle”, “isMoving” and “isEating”, were created, respectively. By default, the condition “isIdle” would be set to “true”, meaning that, by default, the wolf’s idle animation would play. However, if a certain condition was met – such as an object being placed within range – this state would be set to “false”. In its stead, the state “isMoving” would be set to “true”. Finally, if within a certain distance of its intended target (in this case, the rib bone), the wolf’s “isEating” condition would be met, rendering “isEating” to “true” and all other states to “false”.

This logic can be demonstrated more clearly through the code below.

```
void Update()
{
    float dist = Vector3.Distance(target.position, transform.position);

    if(dist <= lookRadius && dist <= 1.5) {
        audio.Stop();
        FaceTarget();
        agent.Stop();
        anim.SetBool("isEating", true);
        anim.SetBool("isMoving", false);
        anim.SetBool("isIdle", false);
    } else if (dist >= 1.5 && dist <= lookRadius) {
        audio.Stop();
        FaceTarget();
        agent.SetDestination(target.position);
        anim.SetBool("isMoving", true);
        anim.SetBool("isIdle", false);
        anim.SetBool("isEating", false);
    } else {
        agent.GetComponent<NavMeshAgent>().SetDestination(agent.transform.position);
        audio.Play();
        anim.SetBool("isIdle", true);
        anim.SetBool("isMoving", false);
        anim.SetBool("isEating", false);
    }
}
```

Figure 49: Code snippet of the "AI Wolf" script, showing the logic behind the wolf's actions.

First, a “lookRadius” had to be established. This is a spherical range in which the wolf would be able to “see” objects placed within; outside of which, the objects would have no effect on its behaviour.

```
public float lookRadius;
```

This variable was declared as “public” so it could be configured from within the Unity editor, for ease of testing.

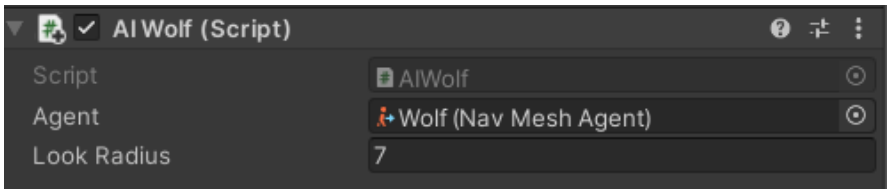


Figure 50: The "AIWolf" component, showing both the AI Agent and the Look Radius as editable fields.

In the Update function, which executes every frame, a floating point value for distance between the target and the wolf’s position is declared. Then, if the distance was to be less than, or equal to, the length of the “lookRadius”, as well as less than or equal to the stopping distance of the wolf’s Nav Mesh Agent component, the relevant boolean values would be triggered. Else, if the distance between wolf and target was to be greater than or equal to the stopping distance, the distance was still less than or equal to the “lookRadius”, and the wolf was not eating, a new destination would be set, and “isMoving” would once again be set to true – the wolf would advance towards the new target. If the wolf was not moving, and not eating, then the Nav Mesh Agent would cease moving forward, and the wolf would once again resume its idle animation.

5.8 Sprint 5

During this sprint, the in-game environment was expanded, the “socket” functionality of features such as locks was implemented, and a solution to the problem of the player’s XR Origin not transferring into the game scene was found. Furthermore, the options available to configure in the Options screen of the main menu were finally scripted.

5.8.1 Expanding the Game Environment

At this stage, the construction of the game environment was almost complete. New sections of the game world, such as the secret room and the second puzzle room, were completed. Once they were built, relevant props and interactable objects were placed within.

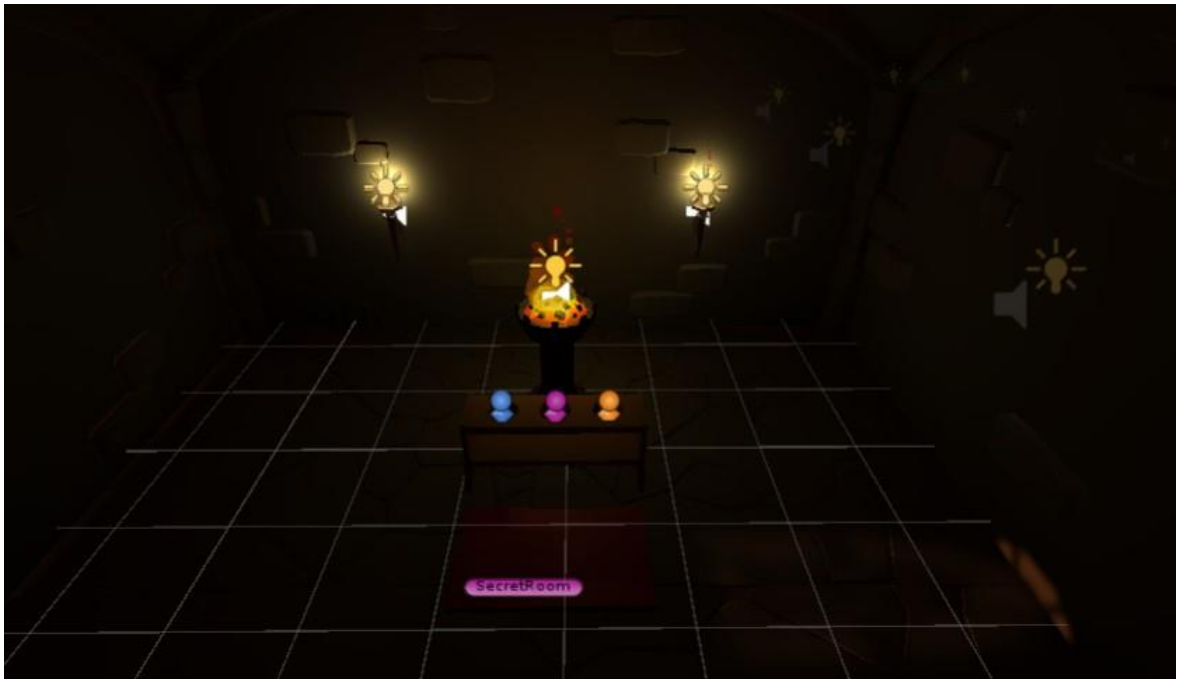


Figure 51: Secret room, triggered by placement of the crystal ball. The amphora stand contains socket interactors (see 5.8.2 below).

Figure 52: Important features of second room: kitchen area and guarding wolf.

New interactable items were also added as the world continued to expand. These included:

- Three gems, of different colours, which could be picked up and placed inside an amphora stand (see Figure 51) to trigger the removal of crystals blocking the doorway
- Food items, such as meat and bones, which could be used to lure the wolf
- A desk containing a puzzle (see 5.9.2).

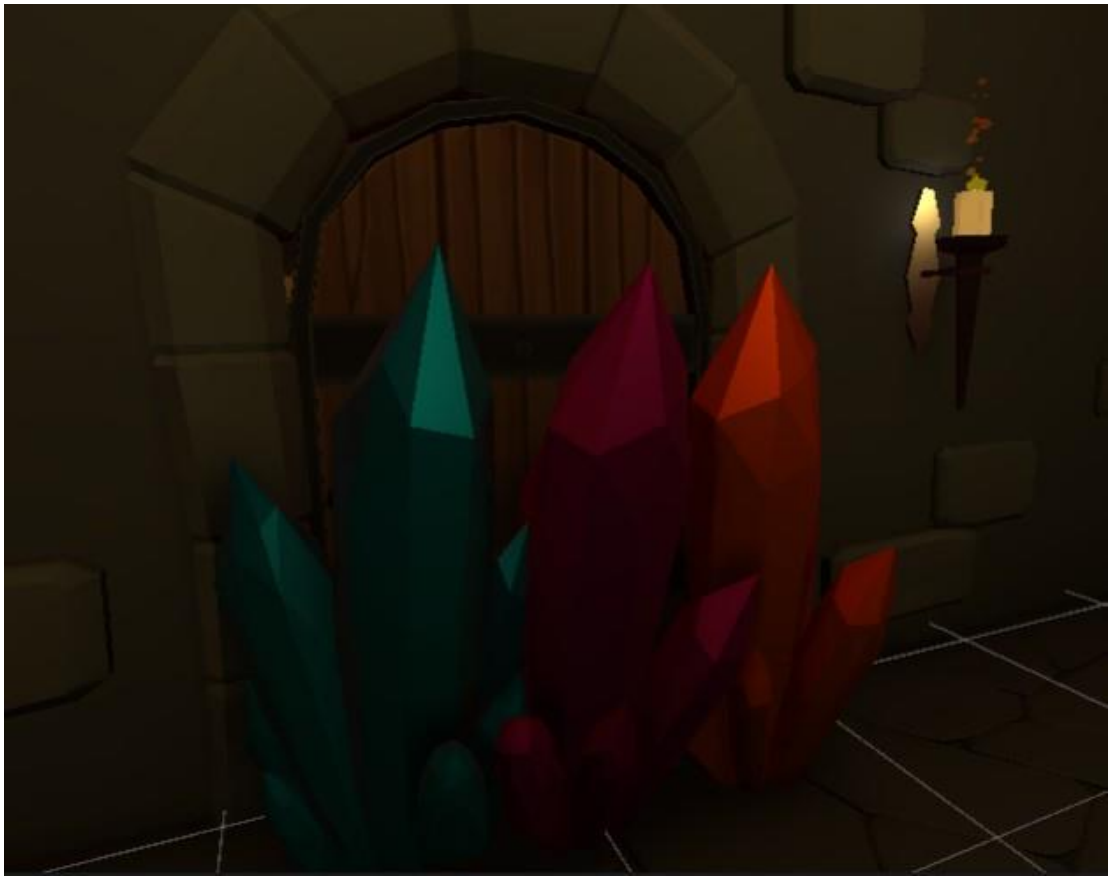


Figure 53: Crystals blocking doorway. The coloured gems around the room match these in colour to give the player a hint.

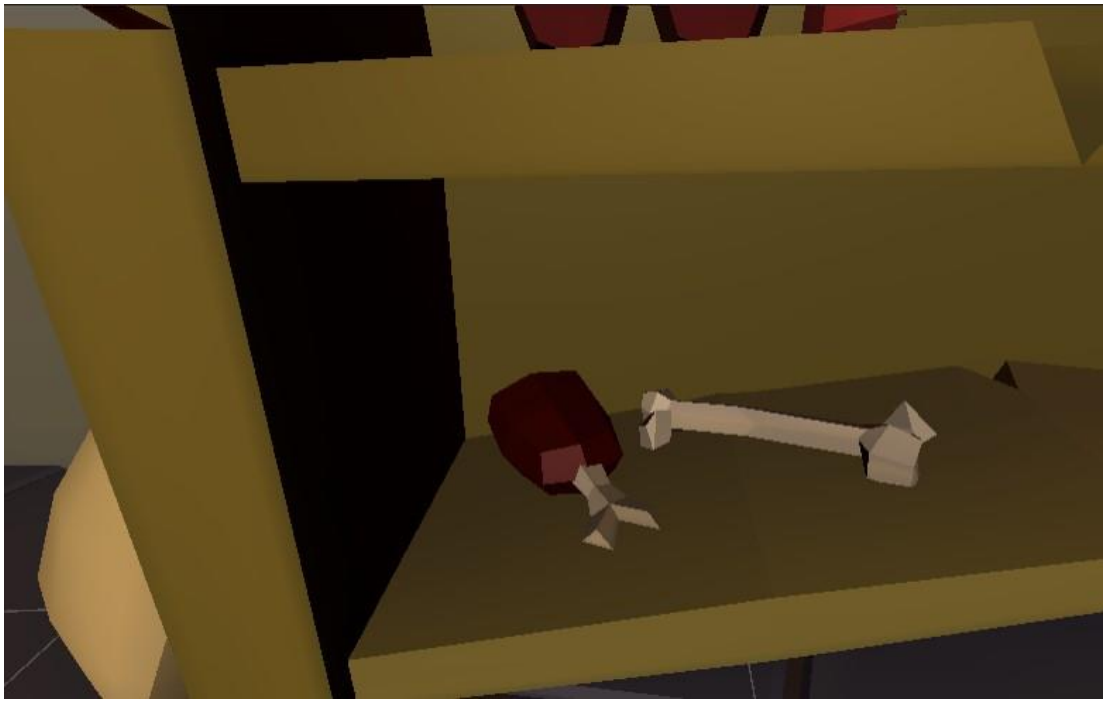


Figure 54: Food items for wolf. Scene lighting has been turned off.

Now that each item was in place, scripting of the important functions of the game, such as sockets and the AI wolf, could commence.

5.8.2 Implementation of Socket Functionality

One of the components of Unity's XR Interaction Toolkit is the "XR Socket Interactor" component. Like the XR Grab Interactable component, this is a script that ensures that an object can be placed in a precise location, for example on hooks or shelves. Often this "socket" is an empty game object, containing only a trigger collider and the Socket Interactor component. The object to be placed is given its own layer, and the socket interacts solely with this layer to achieve the desired result.

XR sockets were placed in the following locations:

- At the end of the wooden plank in the player's cell, to interact with the key outside
- On the cell door's lock, to interact with the key and consequently, open the door
- On the crystal ball stand, to ensure the crystal ball could be placed precisely on the stand
- The three holes of the amphora stand, in which the coloured gems could be placed.

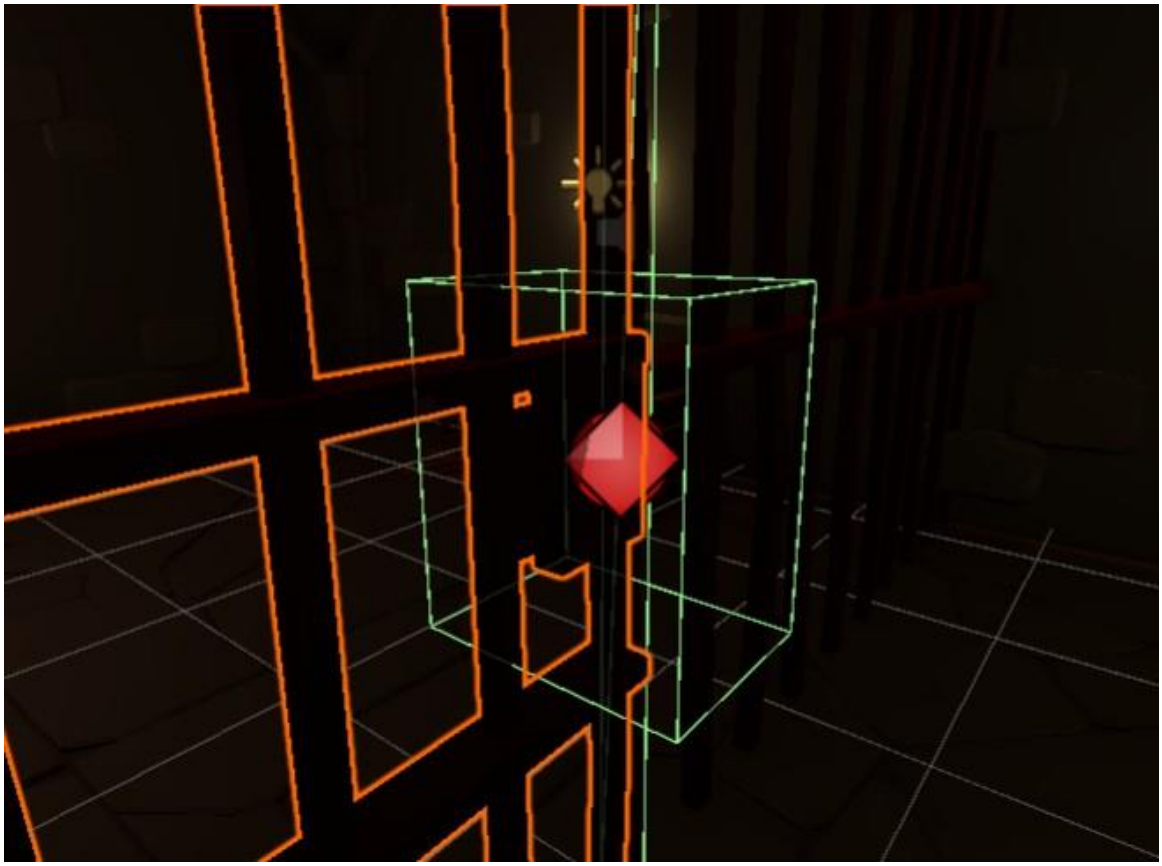


Figure 55: Lock socket (collider shown in green).

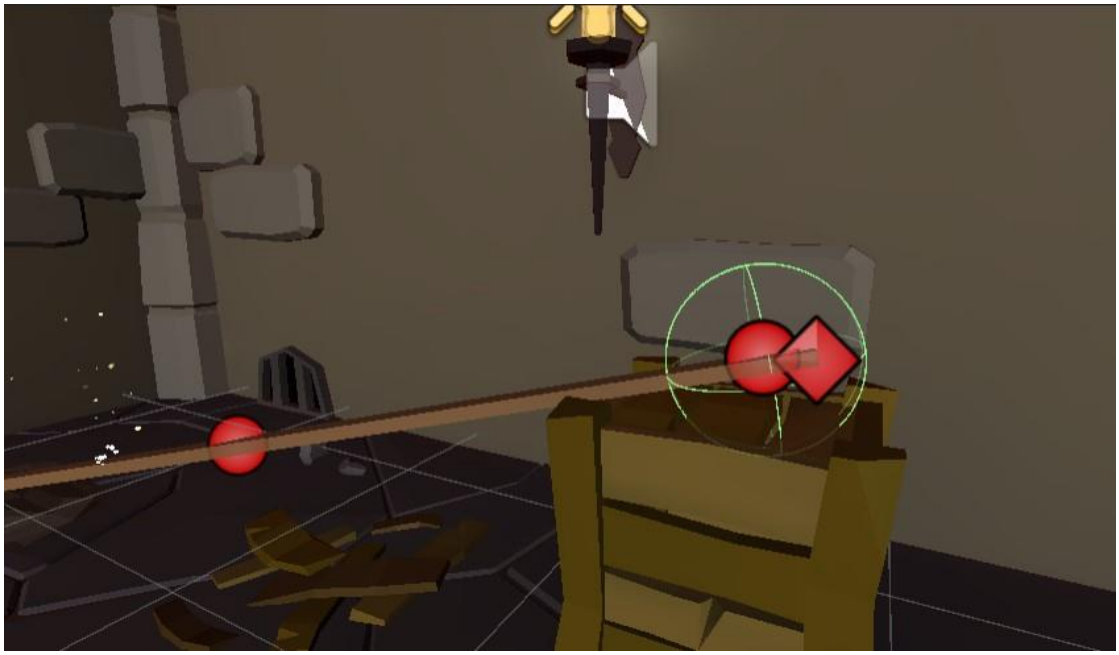


Figure 56: Socket for key attachment (green sphere collider). The player can use this to "hook" the key from the shelf outside.

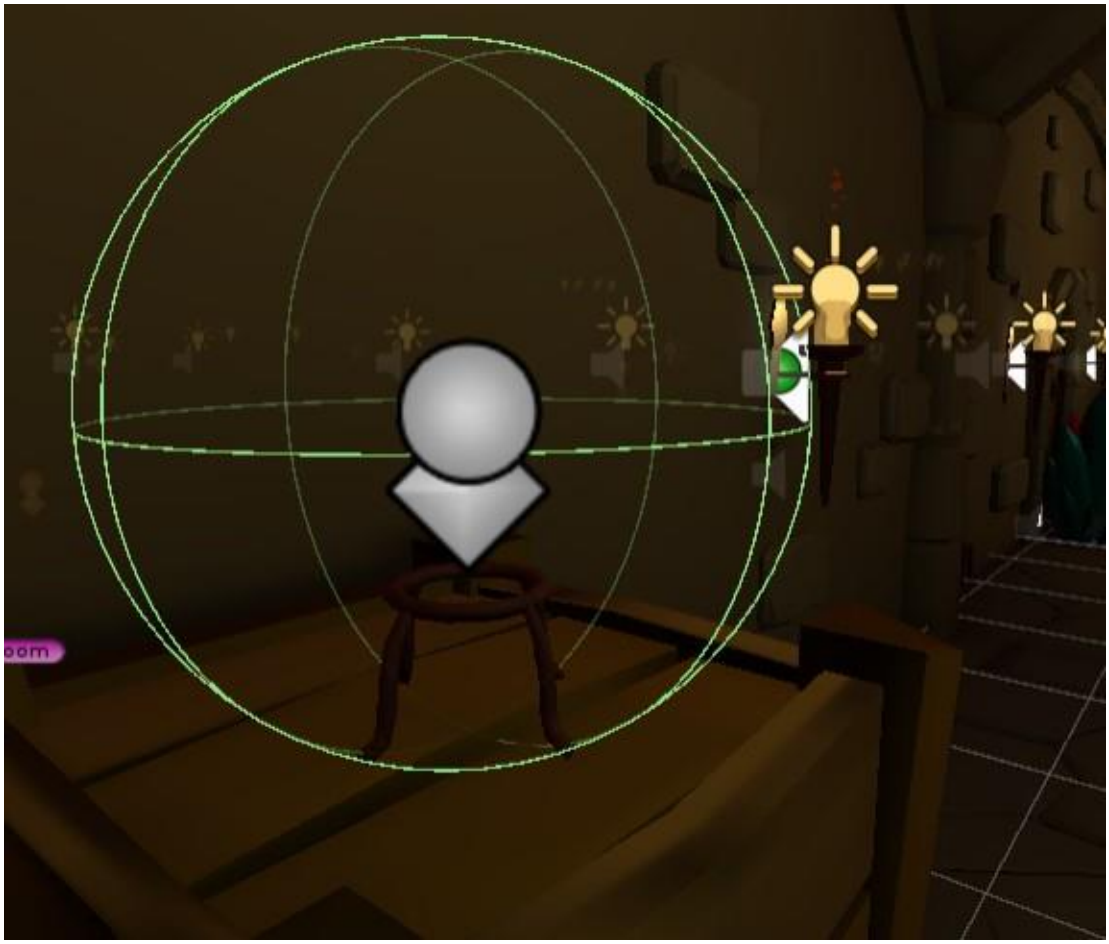


Figure 57: Crystal ball stand, with socket (green).

Figure 58: Stand for gems, with a socket for each (green).

Originally, the plan for the crystal ball socket interactor, as well as the amphora stand interactor, was that they would trigger an animation to play. These animations took the form of the relevant object sliding downwards to eventually disappear from view. However, during testing, the animations would not play, even as the right object was placed within the socket. A temporary fix for this was to simply set the moving objects to inactive upon the relevant socket interaction. It was decided that, as this was proven to work, the animations would be revisited later.

The animations were not utilised in the end, as it was felt that setting the game objects to inactive fulfilled the same purpose. In addition, objects that move at runtime must not be marked as “Static” in the Unity inspector window. While non-static objects are able to move, they are not affected by baked global illumination. This was of little importance to certain moveable objects, such as the doors; but it had the effect of making the moveable wall look different to the surrounding walls, rather than blending into the environment. It was felt that this made it look far too obvious and gave a jarring effect. Therefore, it was marked as “Static” so that it could be affected by the global illumination.

Figure 59: Static checkbox in the Unity inspector.

5.8.3 Animating the Cell Door

The door to the player's cell needed to be animated if it was to swing open and free the player.

In Unity, animations are controlled by a component known as the Animator Controller. Like in the wolf example (see Figure 48), an animator controller is an animation, or series of animations, which can be controlled using different sets of conditions. Animations (known as "states") can be dragged and dropped into these animator controllers, and either be triggered automatically, or only when certain conditions are met.

In the case of the door, the animation was to play automatically.

An animation was created, in which the rotation of the door was increased every frame, to give the illusion of it swinging outwards. Though the animation itself was not controlled by any conditions or triggers inside the animator, the Animator component was disabled so that the animation would not begin at the start of the game. Instead, the component was to be set to active once the key object was placed in the lock by the player. An audio source was also added to the door, once again set to play only when the key unlocked it.

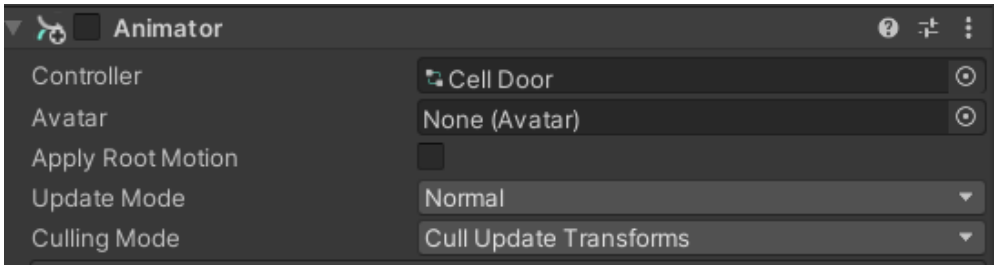


Figure 60: Door's Animator component. Notice how it is set as inactive.

Figure 61: Animator, and audio source, are enabled when key interacts with the lock socket.

5.8.4 Transferring XR Origin to Game Scene

To ensure continuity between the main menu and the game scene, the player's XR Origin had to be transferred smoothly between both.

One of the most common ways of achieving this through Unity scripting is through a method called DontDestroyOnLoad(). Normally, when a new scene is loaded, each object present in the previous is destroyed. However, with this method, objects can be loaded between scenes if so desired.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class DontDestroyPlayer : MonoBehaviour
{
    1 reference
    public GameObject obj;
    0 references
    private void Awake()
    {
        DontDestroyOnLoad(obj);
    }
}
```

The above code shows the script created for this purpose, DontDestroyPlayer. The variable denoting the object to be kept between scenes is declared as “obj”, and set to public so it can be selected directly inside the Unity inspector. On Awake() – before anything is updated – the program is instructed not to destroy the object set in the inspector. In this case, the player’s XR Origin was set to this value.

The XR Origin did indeed transfer between scenes – but with one small problem. When an object is transferred to another scene using the DontDestroyOnLoad() method, it is instantiated at the origin point (0,0,0). However, this point was not intended to be the spawn location. In the case of *AbyssScape*, instantiating the player at this point caused them to spawn in the middle of the room, rather than inside the cell as intended.

To combat this problem, it was first attempted to place the player at the correct starting position through code. A script called PlaceAtPosition was implemented.

```
public class PlaceAtPosition : MonoBehaviour
{
    3 references
    private GameObject SpawnPosition;
    4 references
    public static PlaceAtPosition instancedPlayer;

    0 references
    private Scene scene;

    0 references
    void Awake() {

        if (instancedPlayer == null) {
            instancedPlayer = this;
            DontDestroyOnLoad(instancedPlayer);
        }
    }

    // Start is called before the first frame update
    0 references
    void Start()
    {
        // instancedPlayer = GameObject.FindGameObjectWithTag("Player");
        SpawnPosition = GameObject.FindGameObjectWithTag("SpawnPoint");
        SpawnObjectAtPos();

    }

    1 reference
    public void SpawnObjectAtPos()
    {
        // if (scene.name == "Room 1") {
        //     instancedPlayer.transform.position = SpawnPosition.transform.position;
        // } else if (scene.name == "Victory Screen") {
        //     instancedPlayer.transform.position = SpawnPosition.transform.position;
        // }

        Instantiate(instancedPlayer, SpawnPosition.transform.position, SpawnPosition.transform.rotation);
```

This script set out to achieve the following:

- Declare variables for the player's SpawnPosition, the instanced player, and the current scene
- On Awake(), check to see if there is an instance of the player already in the current scene; if not, instantiate the player in the scene
- Find the GameObject tagged as "SpawnPoint", and allocate this value to the variable of SpawnPosition
- Call the SpawnObjectAtPos() method
- In the SpawnObjectAtPos() method, instantiate the instanced player, at the position and rotation of the SpawnPosition object.

However, this script did not perform as expected. No matter how many adjustments were made to this script, the player would continue to spawn at the origin point.

Though scripting a solution was seen as a more professional way to rectify this, a simple fix was implemented instead. This fix involved moving everything in the scene upwards along the Z-axis, consequently shifting the player's spawn location at (0,0,0) to the desired location within the cell.

Figure 62: Player's SpawnPosition.

It was undoubtedly not the most technical method of solving the problem, but as the project had a strict timeframe, it was a viable solution.

5.8.5 Coding the Options Functionality

Now that the main menu had been put together, it was time to code the options functionality.

In a previous VR example, the options configurable by the player were the ability to toggle the “snap turn” feature on or off, as well as the option to mute/unmute the background music. Those same functionalities were replicated here. To control these functionalities, two similar scripts, ChangeSpriteImage and ToggleMusic, were written, and attached to the Snap Turn button and Music button, respectively.

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.XR;

0 references
public class ChangeSpriteImage : MonoBehaviour
{
    0 references
    public Sprite buttonSprite;
    1 reference
    public Sprite newButtonSprite;
    1 reference
    public Button button;
    // private bool isChanged = true;

    0 references
    public void changeButtonImage() {
        button.image.sprite = newButtonSprite;
    }

}
```

The above script, ChangeSpriteImage, was utilised to change the image of the button's sprite when the button was pressed by the player. In addition to changing the image of the sprite, the XR Snap Turn Provider component attached to the XR Origin would be toggled off. This functionality, however, was configured through the Unity editor; it was simply the sprite image of the button that was targeted through the above code.

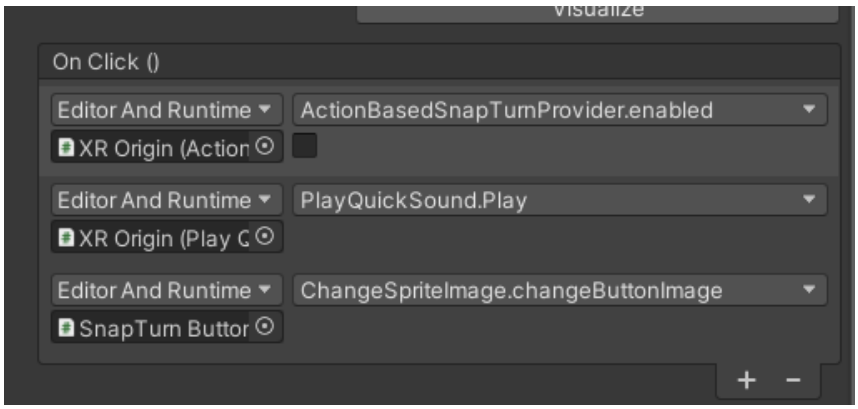


Figure 63: The Snap Turn button's functionality, as seen inside the Unity inspector.

The script used to control the Music button – ToggleMusic – was written similarly. This script also targeted the ambient music present in the scene, allowing the player to turn it off, or back on again.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.XR;
using UnityEngine.Audio;

0 references
public class ToggleMusic : MonoBehaviour
{
    0 references
    public AudioListener Listener;
    2 references
    private Sprite onImage;
    1 reference
    public Sprite offImage;
    3 references
    public Button button;
    2 references
    public AudioSource bgMusic;
    3 references
    private bool isOn = true;

    0 references
    void Start() {
        onImage = button.image.sprite;
    }

    0 references
    public void ButtonPressed() {

        if (isOn) {
            button.image.sprite = offImage;
            isOn = false;
            bgMusic.mute = true;
        } else {
            button.image.sprite = onImage;
            isOn = true;
            bgMusic.mute = false;
        }

    }
```

In this script, the “on” image – the image shown when the music is playing – is set as a private variable. This is because it does not have to be set in the Unity inspector, as it already has the same value as the starting sprite. The variables of “offImage”, “button”, and “bgMusic” could all be set from within the Unity inspector, while a Boolean value of “isOn” was created and declared as true by default.

At the start of the program, the “onImage” is set to the image of the sprite used in the button. Then, a public method called ButtonPressed() is constructed. Within this method is an if-else statement, which controls the music in the following way:

- First, if “isOn” equals true (which it does by default), the sprite image of the button is changed to the “offImage”.
- The value of “isOn” is set to false, and the background music is muted.
- If “isOn” equals false, ie. the music is not playing, the button sprite is changed back to that of the “onImage”, “isOn” is set to true once more, and the background music is unmuted.

A “Help” button, which, when pressed, would detail the control scheme of the game, was also included in the Options menu. However, it was felt that the Main Menu itself was a better location for this button, and so it was moved there instead.

When this button was pressed by the player, it would take them to a Help screen. Here, they would be able to read the controls used by the game.

Figure 64: Main Menu, showing the Play, Options, Help and Quit buttons.

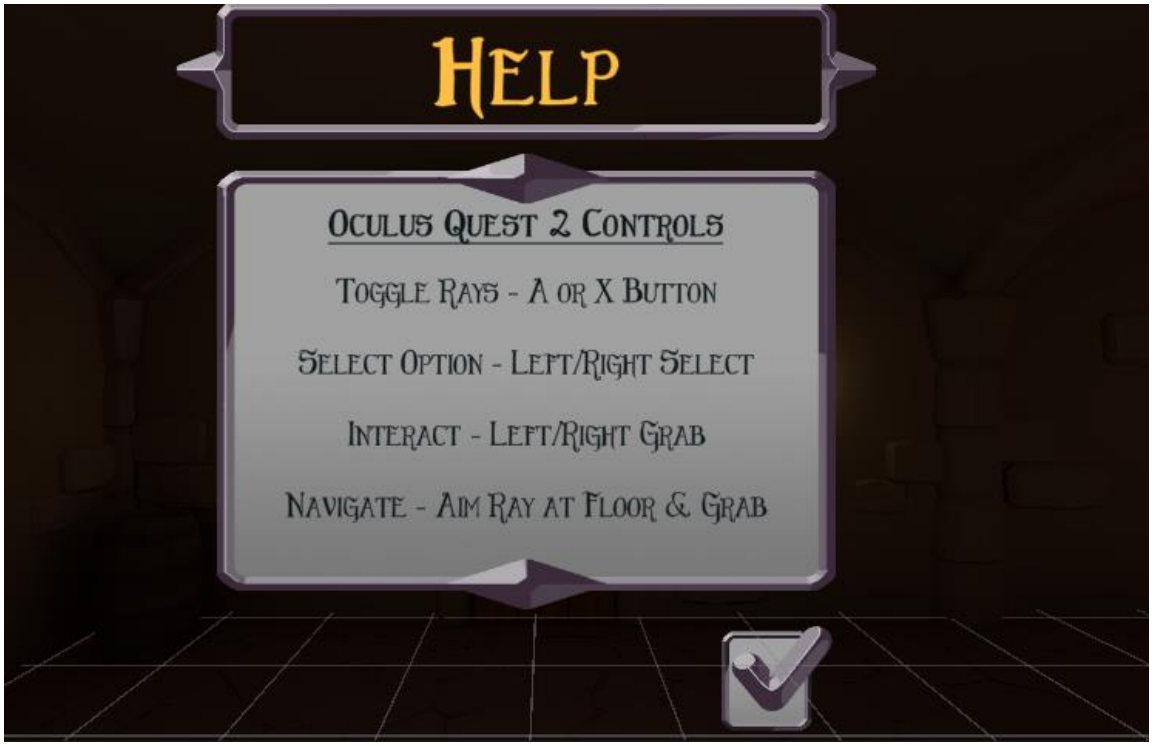


Figure 65: Help screen.

5.9 Sprint 6

The construction of the game scene entered its final stages. Throughout this sprint, remaining features, such as scripting of objects and extra puzzles/challenges for the player, were prioritised. The remaining doors were also configured to be openable, and a collider was placed on the player.

5.9.1 Implementation of Riddles Quiz

As simply locating and placing objects was deemed insufficiently challenging, a new puzzle was to be implemented.

This puzzle was to be activated by placing a book, hidden somewhere in the room, upon a lectern. This lectern contained a socket that, when the book was correctly placed within, would activate a puzzle for the player to complete before proceeding further.

Figure 66: Desk on which the book was to be placed.

The original idea for the puzzle was that of a match-3 game, in which the player would have to swipe tiles left, right, downwards, or upwards, to match it with at least two of the same kind. To try this idea, it was first brought into a test scene (below).

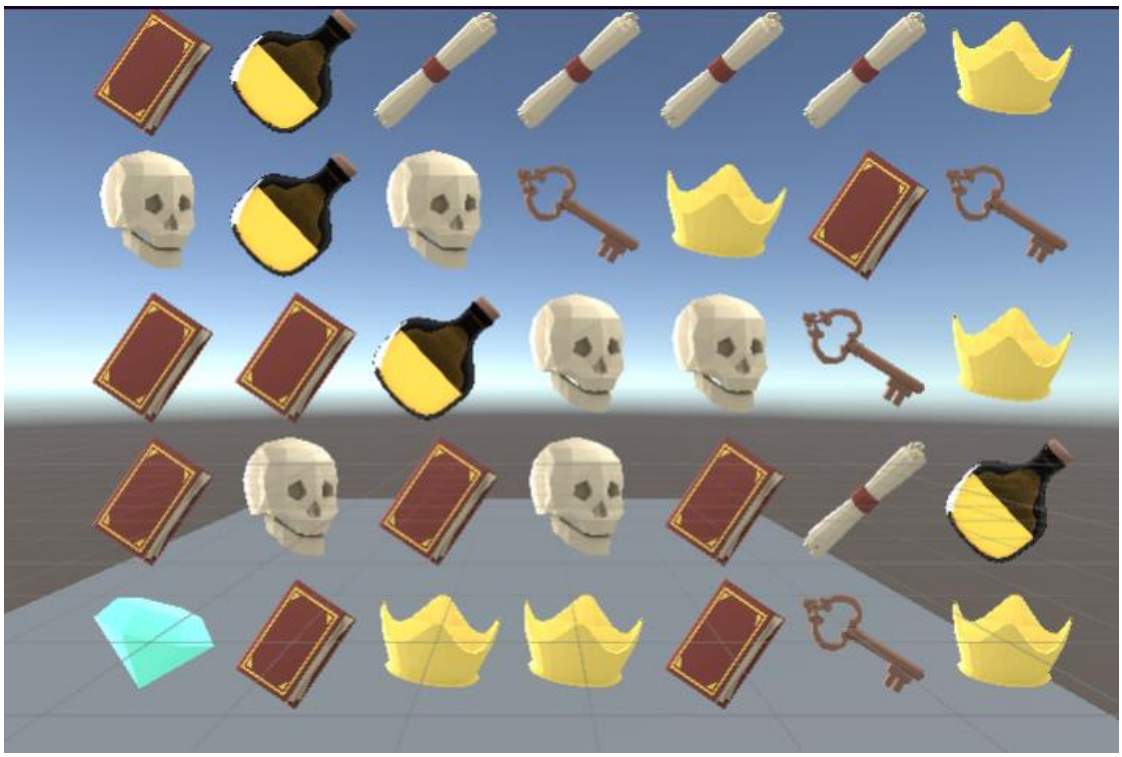


Figure 67: The match-3 test scene.

However, this idea was abandoned when it became clear that it would be too difficult to implement in VR. Games of the match-3 genre are commonly found on mobile platforms, as they are more suited to swipe controls. Implementation in VR, if it were to work, would feel unsatisfying in comparison. The mechanics of such a puzzle were also complex, involving many mathematical concepts such as trigonometry. Consequently, a different type of puzzle had to be implemented.

In keeping with the game's fantastical, medieval atmosphere, it was decided that, instead of the match-3 puzzle, a quiz in which the player must correctly answer three riddles was to be implemented. Not only was this more fitting, but it was also simpler logistically.

Upon activation of the book socket, a screen would appear to the player, prompting them to begin answering the riddles. When the player pressed the "Start" button, they would be taken to the first question. If answered wrongly, they would be taken back to the starting screen, where they could try again until giving the correct answer. Upon answering a riddle correctly, however, the screen would change to show the next one, and so on until all three were answered.

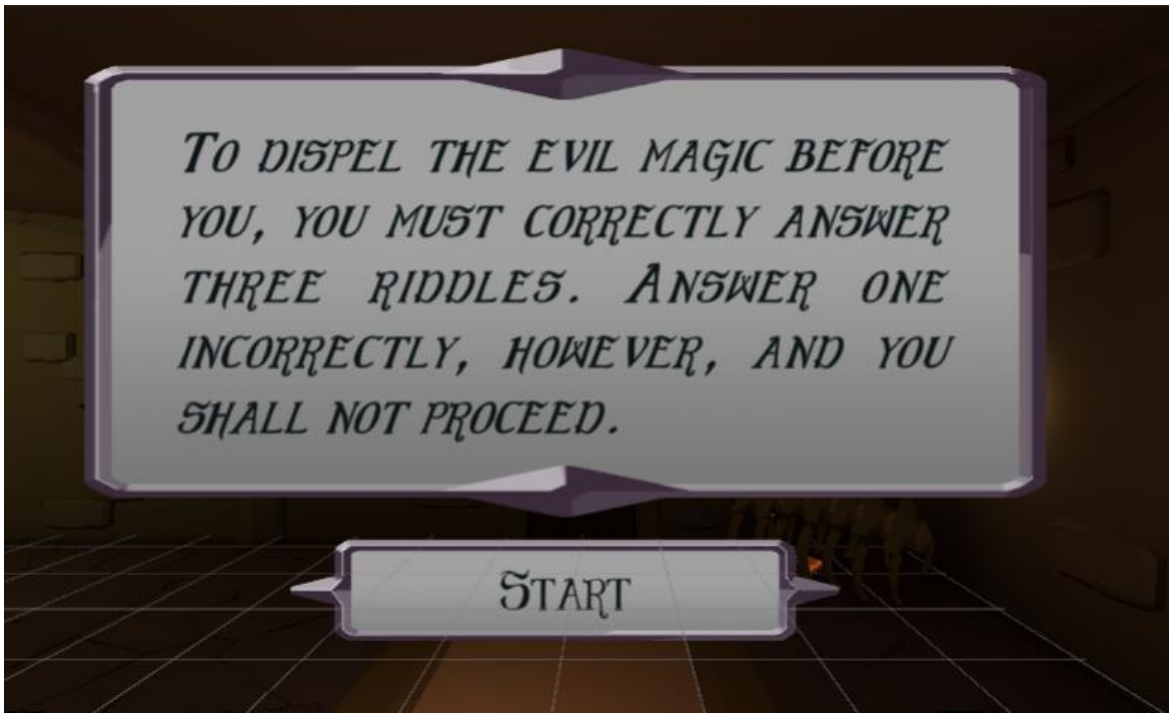


Figure 68: The Start screen for the riddles puzzle.

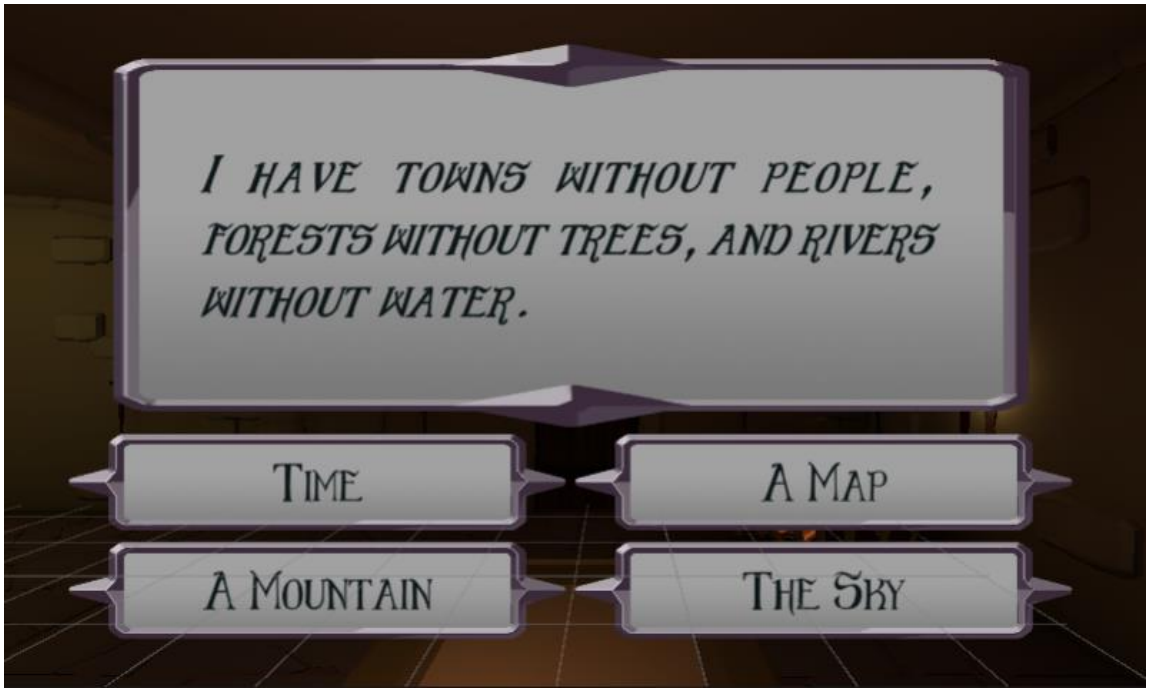


Figure 69: Example of a riddle on the Riddles canvas.

The riddles were set up using an empty game object, simply called “RiddleGameManager”. This object had a script attached called RiddleGameManager, which controlled which riddles were to be displayed to the player, as well as whether or not the player would proceed.

In addition, scripts called Questions and Answer were written. The Questions script contained the information needed by the RiddleGameManager to access each question, each set of answers, and each question’s correct answer; whereas the Answer script was used to detect whether or not a question was answered correctly.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;

[System.Serializable]
1 reference
public class Questions
{
    1 reference
    public string question;
    1 reference
    public string[] answers;
    1 reference
    public int correctAnswer;
}
```

Above is the Questions class, containing the public variables “question”, “answers”, and “correctAnswer”. This Questions class is marked as “Serializable”. In simple terms, this means that its

properties can be configured from directly within the Unity editor, eliminating the need to navigate back and forth between script and editor.

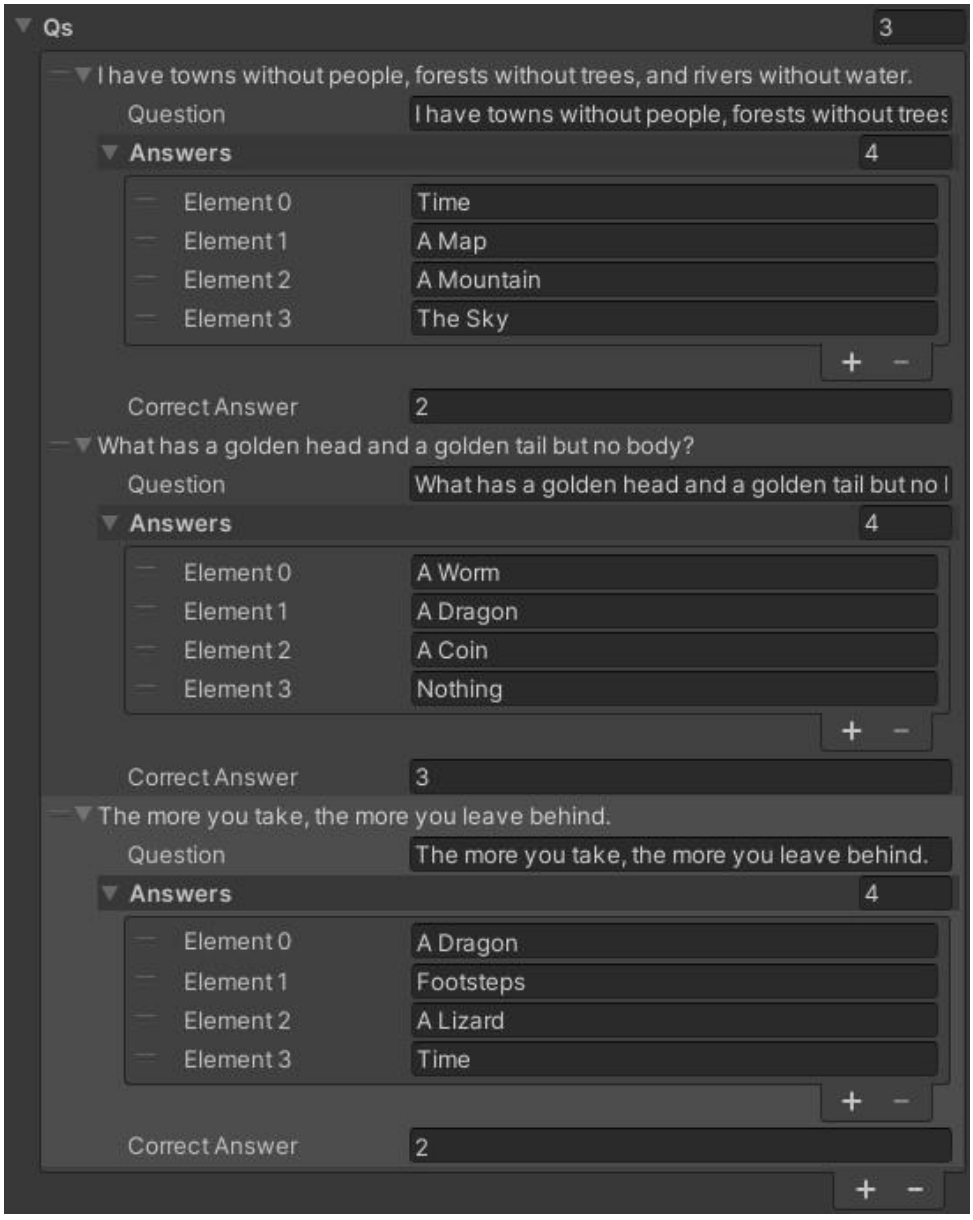


Figure 70: The questions and answers as they appear in the editor, as accessed from the RiddleGameManager.

The Answer class:

```
2 references
public class Answer : MonoBehaviour
{
    1 reference
    public bool isCorrect = false;

    2 references
    public RiddleGameManager riddleManager;

    0 references
    public void answer() {
        if (isCorrect) {
            Debug.Log("Correct Answer!");
            riddleManager.Correct();
        } else {
            Debug.Log("Wrong Answer!");
            riddleManager.IncorrectAnswer();
        }
    }
}
```

This class contains a boolean value for “isCorrect”, which is set to “false” by default. It also accesses the RiddleGameManager’s Correct() and IncorrectAnswer() methods, to track whether the player has answered correctly or not, and to change the result accordingly. In the answer() method (lowercase to avoid conflict with class name), what takes place is thus:

- If the player selects the correct answer, output “Correct Answer!” to the Unity console and execute the RiddleGameManager’s Correct() method.
- If the player answers incorrectly, output the message “Wrong Answer!” to the Unity console, and execute the RiddleGameManager’s IncorrectAnswer() method.

These methods can be seen below.

```
1 reference
public void Correct() {
    score += 1;
    qs.RemoveAt(index: currentQuestion);
    GenerateQuestion();
}

1 reference
public void IncorrectAnswer() {
    Debug.Log("You answered incorrectly. You will not proceed.");
    riddleCanvas.SetActive(false);
    startScreen.SetActive(true);
    book.transform.position = new Vector3(-1.50699997f, -0.0607459024f, 5.34800005f);
}
```

In the Correct() method, the score is incremented by one point (though a scoring system was never implemented in the end), the current question is removed, and the next question is generated,

through the GenerateQuestion() method (see below). If the IncorrectAnswer() method is called, then the canvas on which the riddle is displayed is set to inactive, and the start screen shows once more. (It was also originally intended that the book on the desk would revert to its previous position, but this was never tested as it was deemed unimportant.)

Should the player answer a riddle correctly, it would be replaced with a new one from the array.

```
2 references
void GenerateQuestion() {

    if (qs.Count > 0) {
        currentQuestion = Random.Range(0, qs.Count);

        questionText.text = qs[currentQuestion].question;

        SetAnswers();
    } else {
        Debug.Log("End of riddles!");
        GameOver();
    }

}
```

If the number of riddles answered was greater than zero, a random item out of the array would be presented on the canvas, and the relevant answers would be set. However, if the player reached the end, the GameOver() method would be called.

```
1 reference
public void GameOver() {
    magicFields[0].SetActive(false);
    magicFields[1].SetActive(false);
    riddleCanvas.SetActive(false);
}
```

At the end of the riddles challenge, both the magic fields (see below) and the canvas displaying the riddles would be set to inactive, leaving the player free to proceed.

To justify this puzzle's necessity, magic fields were placed around the kitchen-like area of the room. These had the effect of preventing the player from distracting the wolf before completing the puzzle contained within the desk. Each magic field comprised of an elongated cube object with the Mesh Renderer component toggled off, but the collider on. This in turn had the effect of creating an invisible forcefield around the kitchen area. A particle effect was applied as a child of these objects, to make it look magical, but also threatening.

Figure 71: Magic fields.

When all the riddles were answered correctly, the magic fields would dissipate, leaving the kitchen area open to the player.

This leads on to the next exercise: that of implementing the wolf's AI movements.

Figure 72: Wolf guarding door.

This process was carried out similarly to that of the test scene. However, this time there was one problem.

In the test scene, the target for the wolf to move towards was a static object. This time, though, it was to be an object picked up by the player. Setting the look radius large enough to include the food items was not an option, as this would cause the unwanted behaviour of the wolf automatically detecting them. The challenge here, consequently, was that of only increasing the wolf's look radius when the player selected the food object.

To achieve this effect, the AIWolf script was modified to include a new function, called setLookRadius(), that accepted the float value parameter of "rad". If the wolf's look radius was greater than or equal to zero, this "rad" value would be set as the new look radius.

```
0 references
public void setLookRadius(float rad) {
    //this will contain code to increment the lookRadi
    //it was noted during testing that the wolf only r

    //set the look Radius value in here instead of abo
    if (lookRadius >= 0) {
        rad = lookRadius;
    }

}
```

This value could then be set inside the Unity editor, when the relevant food object was picked up by the player.

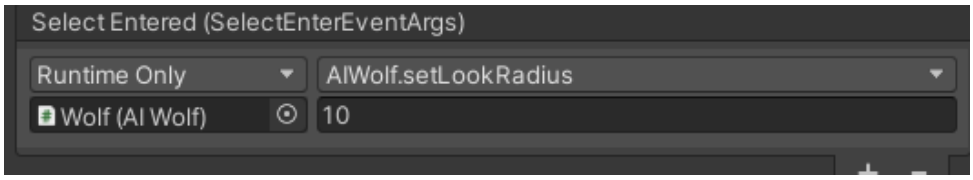


Figure 73: When the food object is selected, set the wolf's look radius to 10.

When tested in the editor, this produced the same effect as the test scene – however, problems were faced in the build (see chapter 6 for more details).

5.9.2 Configuration of Doors

For the user to proceed through the dungeon, the doors to each section had to be openable.

In Unity, there are several ways to achieve this. One of the most common ways to create interactable doors is to assign them a Hinge Joint component. Unity hinge joints work similarly to real ones – they are used to simulate the swinging open, or closed, of a door. Limits can also be set, to ensure that the door only opens a certain amount.

However, when hinge joints were tested with the doors of the dungeon, several problems were faced. The most common of these was the doors behaving erratically whenever a hinge joint was added. Fixes for this behaviour include increasing the mass of the doors' Rigidbody component, as well as the mass scale of the hinge joint itself.

These fixes, however, failed to produce results. Due to the strict timeframe of the project, a workaround was used instead, in the form of animations.

Like in the cell door example (see 5.8.3), an Animator component was placed on each door, and an animation was created that incremented the door's Y-axis rotation every frame. Again, this animation was only set to play once a certain condition had been fulfilled, such as correct placement of an object, or fetching a food item for the wolf.

The very last door was set up slightly differently. This one, instead of opening when a certain action was fulfilled, contained a large trigger collider that, when entered by the player's collider (see 5.9.3 below), would trigger it to open. Outside this door was another trigger collider, which, when entered by the player, would cause the scene to change to the Victory Screen (see 5.10.2).

Figure 74: The last door's Box Collider.

5.9.3 Adding a Collider to the Player

As it is not a requirement for most VR developments, the player's XR Origin did not originally contain a collider. However, for the player to trigger interactions, such as opening the very last door or transitioning to the last scene, a Collider component was added. This took the form of a narrow capsule collider, primitively resembling a humanoid shape.

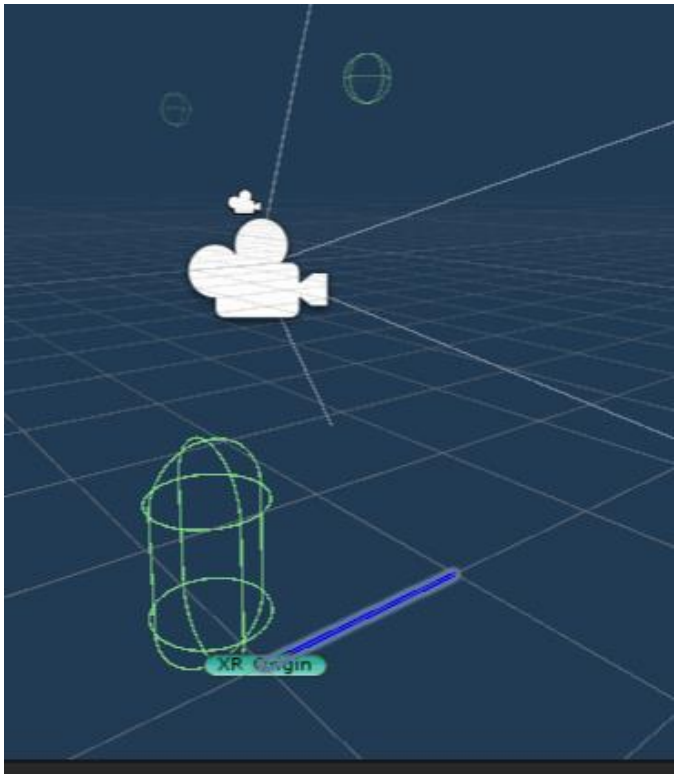


Figure 75: The player's Capsule Collider.

Now that the player had a “body”, they were able to activate triggers by stepping inside them.

5.9.4 Modifying the Plank/Key Interaction

Though they were some of the first tasks to be encountered by the player in-game, the plank and key interaction had not yet been tested.

Implementing the socket functionality for the wood to “hook” the key was not as simple as it initially seemed. With the other sockets, the object to be placed within was grabbed directly by the player. However, the socket on the wood plank required the key to snap to its position without first being picked up by the player.

Two approaches to this were considered.

The first was to attach a Fixed Joint component to both plank and key. A Unity fixed joint is a component that “sticks” two objects together. A script was implemented that would get the Rigidbody of both objects at runtime, thus “sticking” them to each other.

```
0 references
void Start()
{
    joint = obj.GetComponent<FixedJoint>();
    connectedObject = GameObject.FindGameObjectWithTag("Key");
}

0 references
public void Attach() {
    rb = connectedObject.GetComponent<Rigidbody>();
    joint.connectedBody = rb;
}
```

However, like the Hinge Joint component, this produced erratic results. The key did not snap to the wood's position; instead, whenever the plank was released from the player's hand, it would snap back into its original position.

The second approach was to simply increase the size of the key's Box Collider component, as well as that on the end of the plank. This would enable the player to knock the key off the shelf it rested on, and then push it along the floor to the cell.

Though it was not as user-friendly an approach as the former, it was considered the best option for the time.

5.10 Sprint 7

At this stage, a test build was considered for user testing. To ensure the app would run smoothly on build, optimisation of the game also commenced. This involved removal of superfluous objects and compression of lighting and textures. A Victory Screen was also implemented, which appeared when the player exited the final door of the dungeon.

5.10.1 Optimising the Game

It is important that, during Android or VR development inside Unity, certain features, such as textures or lighting, are compressed to reduce processing overhead.

Firstly, texturing of models used was compressed to a smaller size.

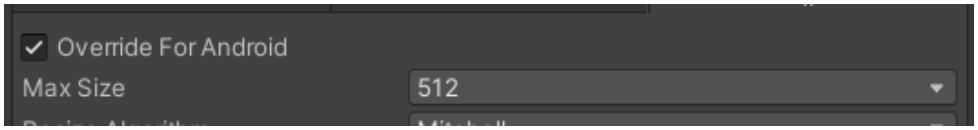


Figure 76: Most textures were compressed to 512 or 1024.

Then, lighting was considered. Though only baked and mixed lighting were used, which are already optimised for Android builds, there were further changes that could be made to ensure a smooth performance. Max lightmap size was set to 512, and the maximum number of lights rendered at any time in view was set to five, out of a possible eight.

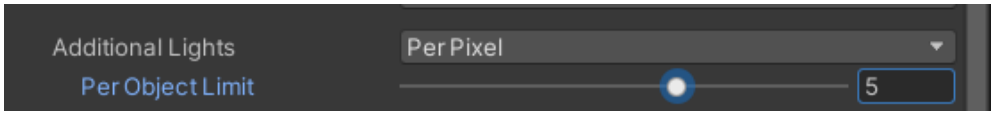


Figure 77: Only five additional lights can be rendered before they are culled from view.

Once textures and lighting were reduced in size, it was decided that superfluous props should be removed from the scene. Though certain props were retained for the creation of an atmospheric environment, those not likely to be seen by the player were deleted.

5.10.2 Implementing a Victory Screen

For completion of the game's objective to feel fulfilling to the player, a Victory Screen had to be designed and implemented.

As mentioned under 5.9.2, the trigger collider necessary to transition to a victory screen after exiting the last door was already in place. The only part remaining was to build the scene environment, as well as a UI display prompting the player to either play again, or quit the game.

It was also decided that this scene should have brighter, more hopeful lighting settings, in contrast to the infernal oranges and reds of the dungeon environs.

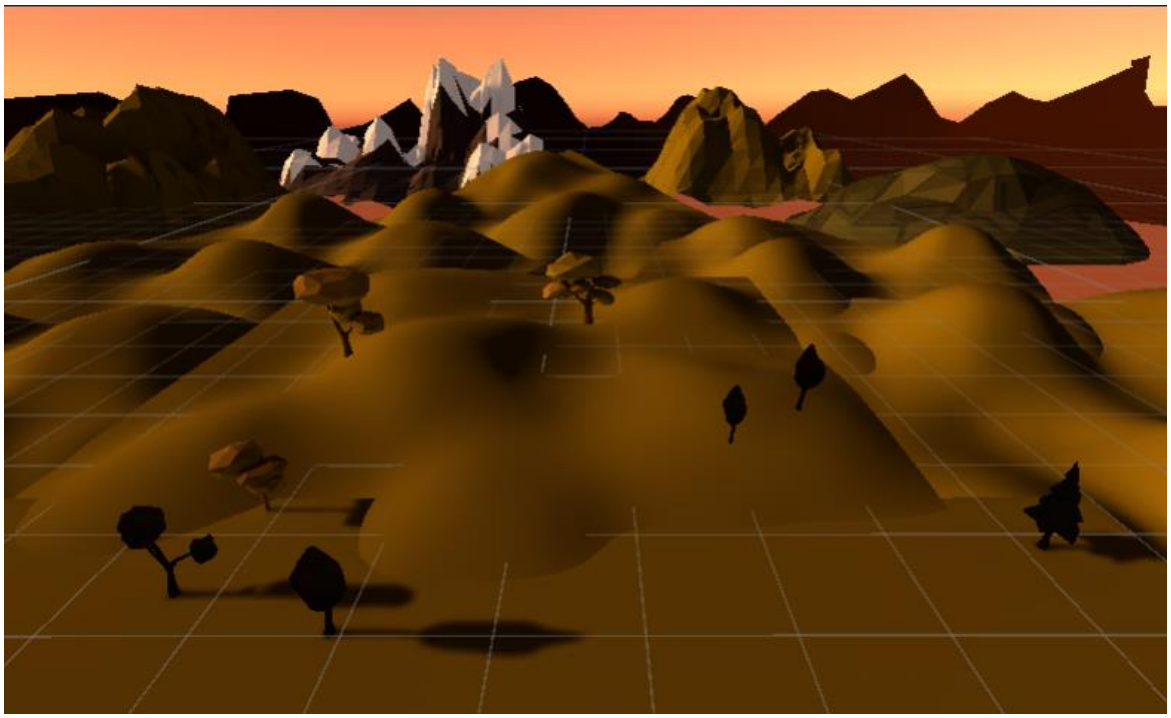


Figure 78: The background environment of the Victory Screen.

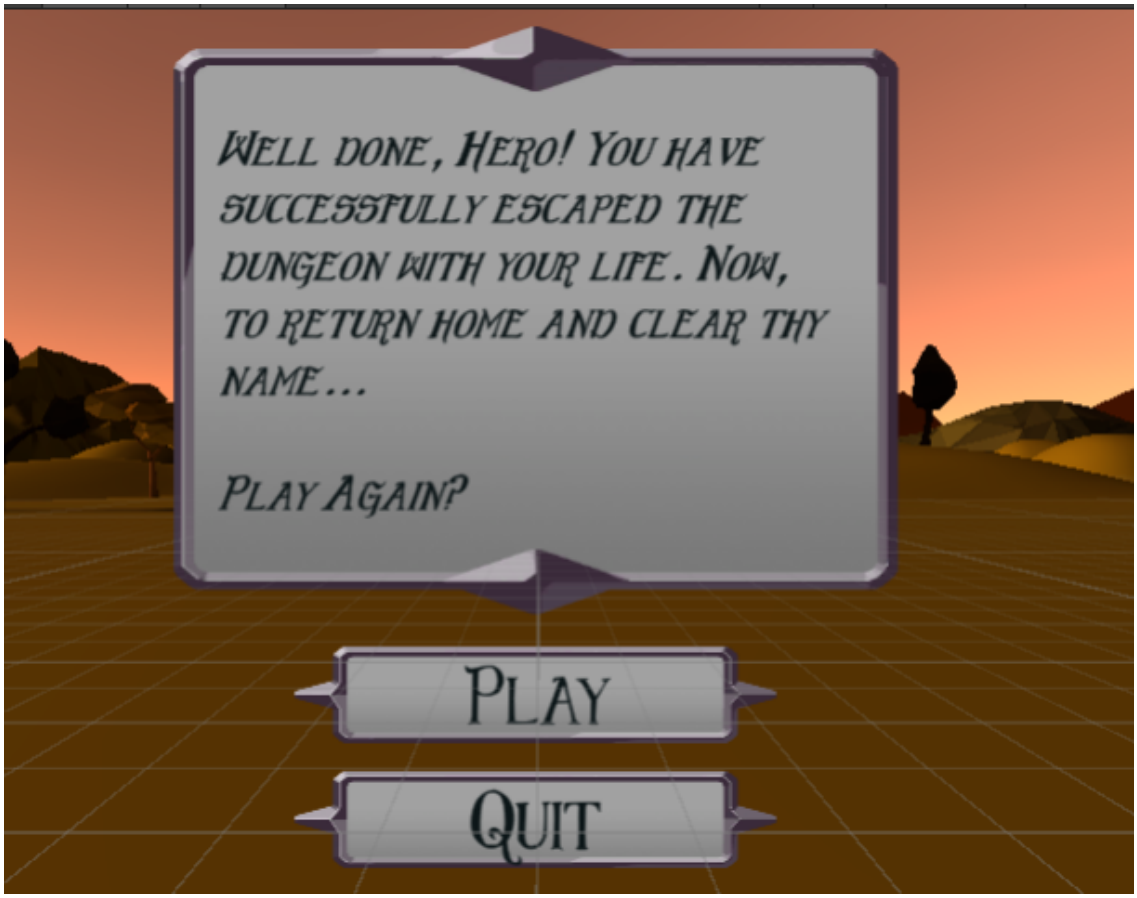


Figure 79: The Victory Screen UI.

5.10.3 Creating a Test Build

To conduct user testing, an alpha build of the game was made.

This process was not without its problems, however. Early builds made of the project had only resulted in the Quest headset crashing, preventing the application from loading. After some research, however, the root of the problem was found.

When building for Oculus devices, it is required to set the minimum target API to API Level 32. By default, Unity sets this API level value to "Android 6.0 'Marshmallow'". This was swiftly fixed.

Figure 80: API levels in build settings.

5.10.4 Functional and User Testing

Once the build problems were fixed, both functional and user testing were carried out (see chapter 6).

5.11 Sprint 8

Throughout this sprint, final optimisations and minor fixes were the priority. All of the important mechanics had been included at this stage of the project; all that remained was to ensure it was as functional as possible.

5.11.1 Colliders

There were multiple colliders in the scene which, as they were not convex in nature, the player could pass through unabated. If left alone, this risked many features, such as the cell bars, becoming redundant.

To remedy this, the mesh colliders of these objects were marked as convex. These colliders were present on both the cell bars and the magic forcefields.

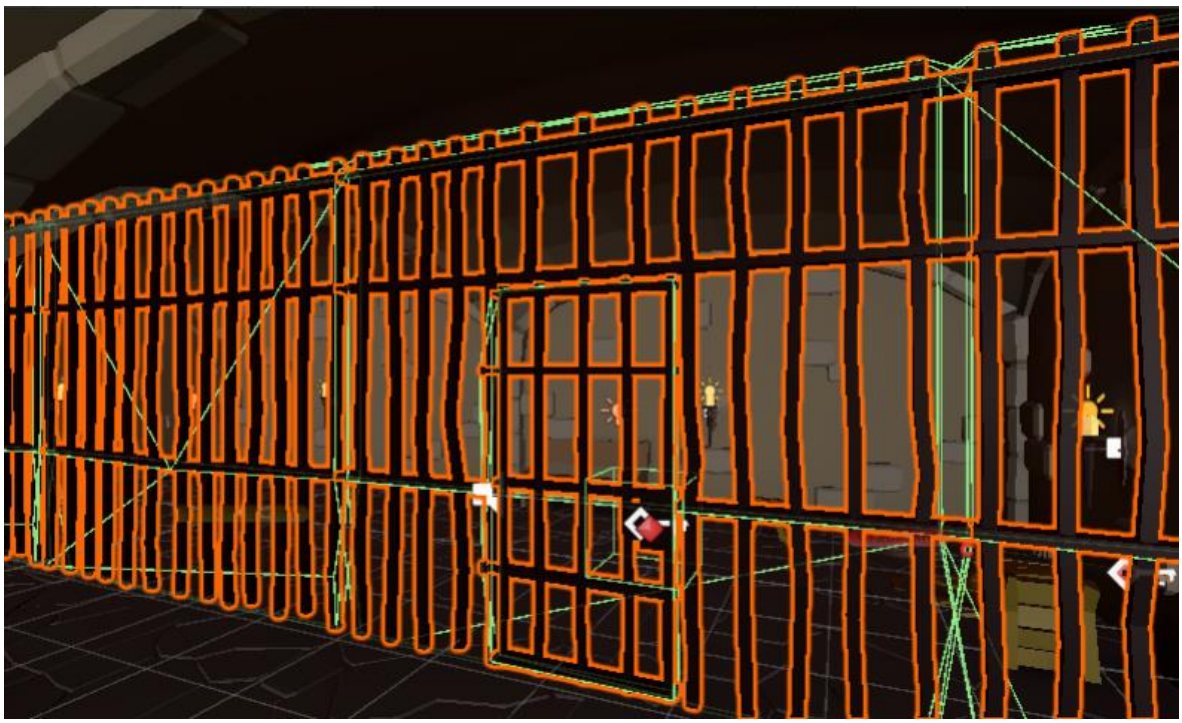


Figure 81: Convex collider on the cell bars and door (green).

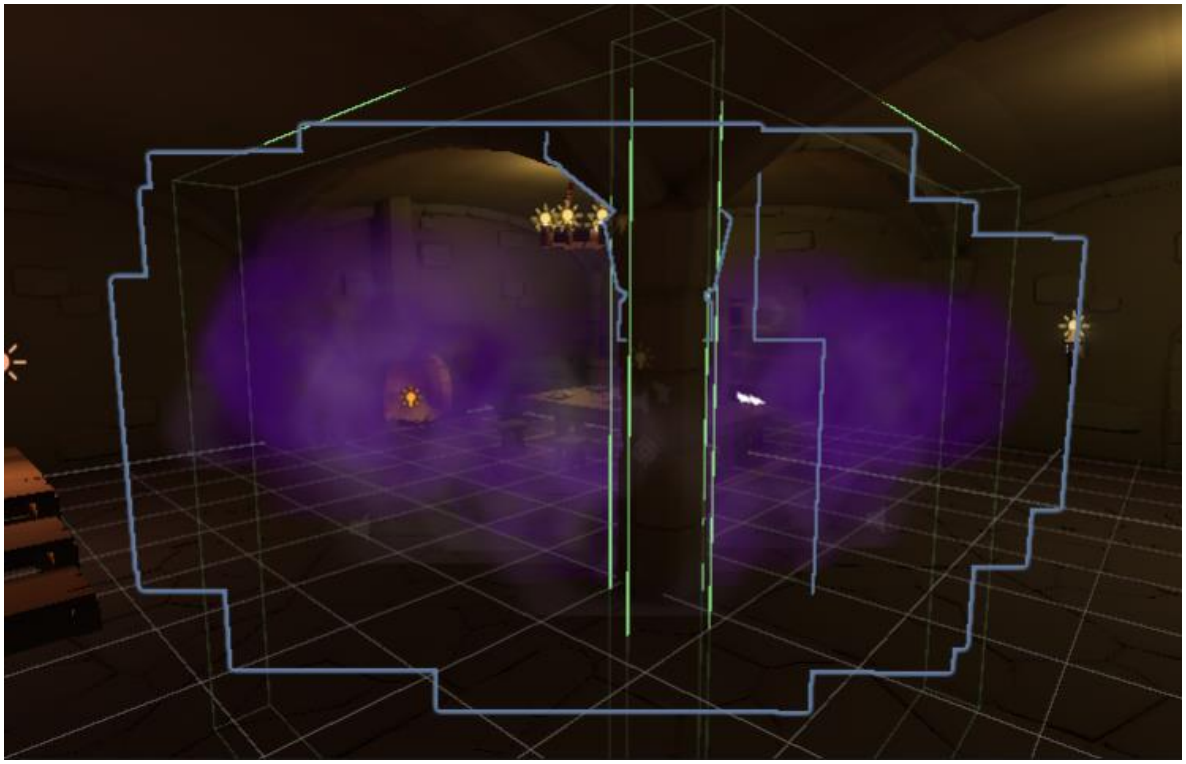


Figure 82: Convex colliders on magic fields.

5.11.2 Rotation of Attach Points

A frequent problem faced when implementing the XR Socket functionality is that of correct orientation. When an object is placed within its relevant socket, it is not always facing in the right direction. This is caused by the 3D model's pivot point and its native orientation.

A solution to this is to rotate the designated "AttachPoint" of the socket to the correct orientation, so that the object, when placed, will be positioned in the correct way.

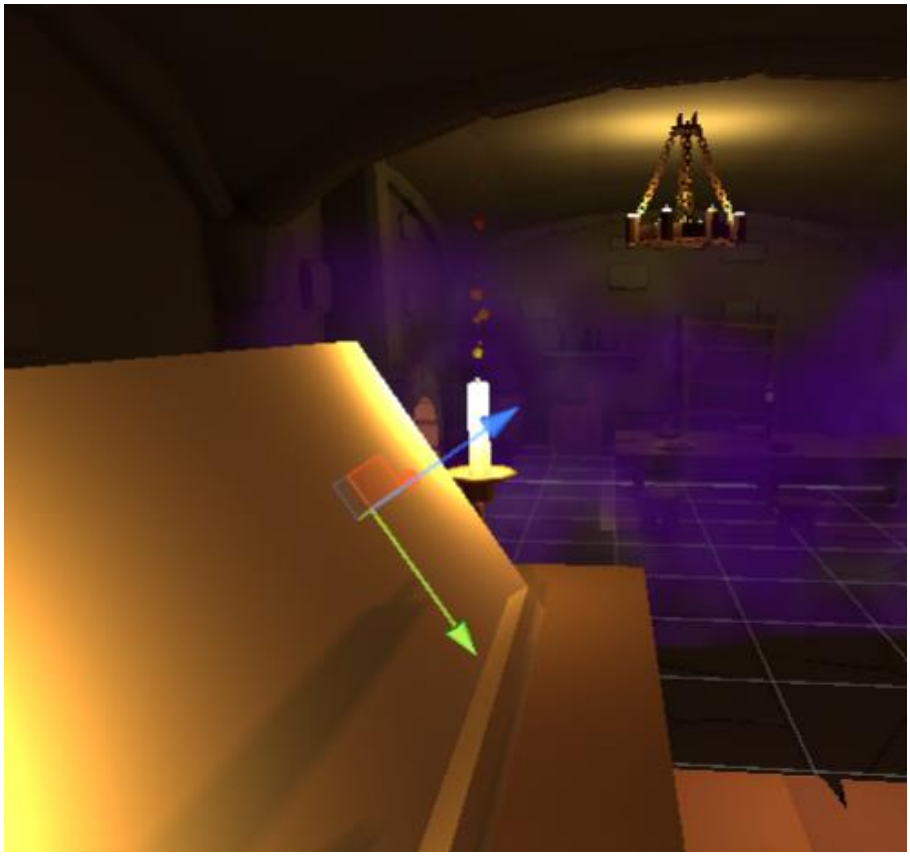


Figure 83: The movement toggles on the Attach Point for the book on the desk. The Z-axis is pointing outwards, so this is how the book will face.

Consequently, the "AttachPoint" of both the desk and lock sockets were rotated in the correct way.

5.11.3 Further Optimisations

Before a final build was made, further optimisations were undertaken. This involved ensuring all non-moving objects were marked as "Static", ensuring those that moved at runtime were not marked as "Static", and removing unnecessary physics components from objects that did not require them, eg. the animated doors. Objects that were previously interactable, but with which the user was not required to interact to progress, were also made static.

5.12 Conclusion

This chapter outlined the tasks performed during each fortnightly sprint, and the processes involved to bring the game's most important features to completion. There was a lot of work carried out over each sprint; but, nonetheless, a lot of knowledge was gained in the process. While the finished product still had potential for improvement, the features implemented most successfully were:

- The building of an atmospheric game environment
- Design and implementation of a main menu system
- Testing and implementation of wolf AI
- Transferring of player XR Origin from one scene to another
- Implementation of socket functionality

- Implementation of Riddles puzzle
- Configuration of triggers to open doors and move between scenes
- Configuration and optimisation of appropriate lighting.

6 Testing

6.1 Introduction

This chapter discusses the tests that were undertaken for *AbyssScape*. There were two types of tests: Functional Testing and User Testing. Functional Testing was undertaken to determine what worked, and what did not work, in the game; while User Testing was undertaken to determine the ease of use, and what the user can do in the game.

6.2 Functional Testing

The main goal of functional testing is to examine how functional the core game mechanics are, and to identify whether or not they respond correctly to user inputs. The functional testing for *AbyssScape* was divided into three parts:

- menus and user interface
- movement and controls
- in-game activities.

A technique known as “black box” testing is commonly used for such an undertaking; however, as this was an individual project, this approach was not taken.

6.2.1 Menu/User Interface

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	View the Help menu.	Press the Help button.	The Help menu opens and displays the controls.	The Help menu opened and displayed the controls.	Responsive, showed correct input.
2	Return to the Main Menu.	Press the OK button.	The Help screen closes and the Main Menu is again shown.	The Help screen closed, and the Main Menu was again shown.	Responsive, showed correct input.
3	View the Options menu.	Press the Options button.	The Options menu displays the toggles for the snap-turn and music functions.	The Options menu displayed the toggles for the snap-turn and music functions.	Responsive, showed correct input.
4	In the Options menu, toggle the Snap Turn function and Music off.	Press the buttons beside	Both the snap-turn function and	Both the snap-turn function and	Responsive, showed correct input.

		the “Snap Turn” and “Music” items.	the music are turned off.	the music were turned off.	(Note: the snap-turn function would not re-enable once turned off.)
5	On the Riddles canvas, select the Start button to begin the quiz.	Press the Start button below the text panel.	The Riddles canvas is enabled.	The Riddles canvas was enabled.	Responsive, showed correct input.
6	During the Riddles quiz, answer a riddle wrongly.	Select the wrong answer to a riddle.	The original text panel should reappear, and the user should be able to restart the question.	The original text panel reappeared, and the user was able to restart the question.	Responsive, showed correct input.
7	During the Riddles quiz, answer a riddle correctly.	Select the correct answer to a riddle.	The next riddle in the series should appear, until the last is reached. Then, the canvas should disappear completely.	The next riddle in the series appeared, until the last was reached. Then, the canvas disappeared completely.	Responsive, showed correct input.
8	On the Victory screen, select the Play Again button.	Select the Play Again button from the UI.	The UI should appear in front of the player, and the player should be able to play the game again.	The UI appeared too far away from the player, instead of directly in front; however, the player was able to play the game again.	Semi-functional.

6.2.2 Movement & Controls

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Teleport to a new location on the floor.	Toggle the rays using the Quest controllers' X or A buttons, then press the grab button.	The user is teleported to a new location in the room.	The user was teleported to a new location in the room.	Responsive, showed correct input.
2	Using the rays, grab an interactable object.	Toggle the rays using the Quest controllers' X or A buttons, then press the grab button.	The interactable object snaps to the location of the user's hand.	The interactable object snapped to the location of the user's hand.	Responsive, showed correct input.
3	Using only hands, grab an interactable object.	Place hand over an interactable object.	The interactable object snaps to the location of the user's hand.	The interactable object snapped to the location of the user's hand.	Responsive, showed correct input.

6.2.3 Activities

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Locate a plank of wood and use it to move a key to the cell door.	Grab the interactable plank and push the key from the shelf towards the cell.	The plank is picked up by the user, used to knock the key off the shelf, and slid across the floor to the cell.	The plank was able to be picked up by the user; however, it was not easy to manipulate the key with it. Furthermore, both plank	Responsive; however, not optimal and only semi-functional.

				and key would occasionally fall through the floor.	
2	Pick up key and place it in the lock.	Grab the key object, and place it in the lock socket attached to the cell door.	The key snaps to the lock and causes the door to open.	The key snapped to the lock and caused the door to open.	Responsive, showed correct input.
3	Locate a crystal ball stand and place the ball in it.	Pick up the crystal ball object, and place it in the crystal ball stand.	The crystal ball snaps to the stand's socket, and causes the wall behind it to disappear.	The crystal ball snapped to the stand's socket, and caused the wall behind it to disappear.	Responsive, showed correct input.
4	Locate a wooden stand, and three coloured gems to put in it. Put the coloured gems in the stand.	Pick up each of the gems, place them in the sockets of the stand.	The gems snap to their sockets, and cause the coloured crystals to disappear.	The gems snapped to their sockets, and caused the coloured crystals to disappear.	Responsive, showed correct input.
5	Locate a desk, and place a book on it.	Pick up the book object and bring it to the desk.	The book snaps to the socket on the desk.	The book snapped to the socket on the desk.	Responsive, showed correct input.
6	Remove the magic fields blocking the kitchen.	Answer the three riddles correctly.	The magic fields disappear.	The magic fields disappeared.	Responsive, showed correct input.
7	Use an appropriate food item to distract the wolf.	Pick up a food item and place it within range of the wolf.	The wolf moves towards the food item, and once close enough, begins to eat it.	The wolf occasionally moved towards the food source; however, most times, it remained in the idle pose, not detecting the food at all. Other times, it would	The food items were able to be handled by the player, however the wolf did not always respond as it should have. It would either remain in the idle pose, or only

				move towards food sources not selected by the player, or would only detect certain food items.	target certain items. Therefore, this feature remained semi-functional.
8	Go to the exit and step outside the door.	Move to the last room and out the door.	The Victory screen should trigger.	The Victory screen was triggered.	Responsive, showed correct input.

6.2.4 Discussion of Functional Testing Results

From the results of the functional testing, it could be concluded that most of the core mechanics of *AbyssScape* worked as intended. There were, however, areas for improvement, such as the wolf mechanic and the user interface of the victory screen showing up wrongly. It was decided that the wolf mechanic would continue to be worked on, as when it worked correctly, it formed an impressive part of the game experience.

All in all, however, the game functioned as intended.

6.3 User Testing

6.3.1 Introduction

As *AbyssScape* is a virtual reality game, the aims and uses of it would be different to a mobile app, website, or conventional game. However, there were still clear aims and use cases in mind when developing the project.

The game was primarily designed to cater towards those already somewhat familiar with VR, although it would also be simple enough for the inexperienced to play. It also is not particularly intensive. The game offers smooth controls, low-poly graphics, and minimal sound. Added to this is the fact that the Quest 2 headset is much more accessible and user-friendly than both its predecessors and competitors.

Above all, the game sought out to create an immersive puzzle experience, that could be enjoyed by any kind of player.

6.3.2 Ease of Learning vs. Ease of Use Testing

The game was not intended to primarily be an ease-of-learning exercise; however, with VR still failing to reach a mainstream audience, it was anticipated that most users would have little to no experience of VR environments. Therefore, certain tasks were designed to ease the players into the

game, such as learning how to move and pick up items. Other tasks, such as locating items and answering riddles, were designed to test how easy it was to perform these actions. They were also elements with which more seasoned gamers might be familiar; therefore, eliminating the need to "learn" such mechanics.

6.3.3 User Testing Tasks

The tasks presented to the users can be seen below.

6.3.3.1 Task 1

On the Main Menu screen, locate the "Help" button.

Press this button, read the controls panel, and exit the screen.

Then, locate the "Options" button. Press this button, find how to toggle music off, then back on.

6.3.3.2 Task 2

Play the game, and find out how to toggle rays and move around.

Then, locate an interactable plank of wood at the end of the cell. Find out how to pick it up.

Look outside the cell for a key object. Once located, find out how to move it.

When key is within reach, pick it up, and find how to unlock the door.

6.3.3.3 Task 3

Locate a crystal ball stand in the room, then the ball to put in it.

Once this is done, locate a wooden stand, then three coloured gems to put in it.

There is a hint in the room as to what order the gems should be placed; find out what it is.

6.3.3.4 Task 4

In the next room, locate a desk. Then, look around for a book to place on it.

Then, locate the "Riddles" canvas. Answer the riddles. (Do not worry if they are not answered correctly the first time – there are multiple attempts allowed.)

Locate a wolf guarding the next door, then find a suitable food item to distract it.

Once this is complete, find the exit.

6.3.4 Ideal Participants

The ideal participants of this game fell into two categories: those who played games, but had limited experience of VR; and those who had little to no experience of games or VR. It was felt that this was the best way to get measured, unbiased results.

For the following user testing sessions, two gamers, who fit these criteria, were approached about the activity, and gave their consent to testing the game. One of these was a gamer who had had prior experience with VR games, though not to a great extent; the other, a gamer with no prior experience, who had before only played non-VR games. Both, however, were familiar with the area of gaming as a whole; though VR games are noticeably different in many ways, this, too, would prove valuable.

6.3.5 Test Environment

User tests were conducted in person, as virtual reality is not suited to remote testing.

To carry out user testing, a build was first made of the application, and loaded onto the Oculus Quest 2 headset. The tester would then put on this headset, and play through the game, performing the tasks required (see 6.3.3).

The user testing was informal, with both a family member and a close partner agreeing to take part. The screen was not recorded, but both testers' observations were noted, and were consequently transcribed here.

Overall, the user testing took place in a casual, relaxed atmosphere, which made the test users feel at ease.

6.3.6 Analysis of Data and Recommended Design Changes

There were some very useful observations made during the user tests. Though the two testers were both people who enjoyed games, they both had limited experience of VR applications. This yielded interesting results, as not only were they able to draw on their knowledge of pre-existing games, but also how to make VR experiences better, especially for those new to the area.

Common observations during testing included:

- There was no reset button in the Options menu, and consequently no way to revert certain choices.
- Interactable objects sometimes clipped through the floor when dropped.
- The key was too hard to move to the cell door.
- The coloured gems looked too similar to the crystal ball, and could be easily mistaken for one another.
- Certain text options on the Riddles canvas overlapped each other, making it too easy to accidentally select the wrong answer.
- The wolf AI was not consistent. It would either not detect the items it should, and consequently not move; or, it would instead target the wrong items.
- The Victory Screen UI would not appear in front of the player.
- The haptic feedback in the controllers when objects were grabbed was not strong enough to be felt.

Feedback on the game was, however, overall positive. Testers responded by saying they would recommend the game to others. However, it was clear that several adjustments were necessary to provide an even more enjoyable experience.

Recommended design changes included the following:

- Including a reset button on the Options screen, to revert any changes made.
- Ensuring consistent physics on all interactable items.
- Refining the colliders on the text answers to the riddles, to avoid accidentally answering wrongly.
- Improving the wolf AI, so it would correctly target the item the player picked up, and move towards it.
- Fixing the position of the Victory Screen UI, so it appears in front of the player.
- Increasing the strength of the haptic feedback when objects are picked up.

6.3.7 Personal Reflection

Overall, the user testing process ran smoothly. At first, difficulties were faced, such as the test build of the game failing to load, and parts of the game not functioning as they should. Ultimately, though, these were of little hindrance to the experience, and instead presented learning opportunities. Once these difficulties were overcome, the tests proceeded as normal.

It was also refreshing to be able to carry out the tests in-person, rather than remotely. With previous projects, user testing was carried out over software such as Microsoft Teams. Though this testing yielded similar results, it was ultimately not as fulfilling as in-person testing. When carrying out user testing in-person, subtle cues such as body language can also be used to gauge interest and/or enjoyment. This is simply not possible during remote testing.

6.4 Testing Materials

Materials relevant to the testing process, such as pre- and post-test questionnaires and their responses, can be found in the Appendix.

6.5 Conclusion

This chapter has discussed the results of both functional and user testing. The functional testing ran smoothly, with all but a few key mechanics working as intended. Similarly, the user testing proved successful. Comments from the game's testers were taken on board as valuable suggestions, and overall user feedback indicated that the game was fun and enjoyable.

User testing and functional testing are both essential to an application's development, be it a game, mobile app, or website. The functional testing was useful in determining areas that could be improved upon in future, as was the user testing. Combined, they gave valuable insight into which areas could be iterated upon to provide an even more enjoyable, immersive game experience.

7 Project Management

7.1 Introduction

This chapter describes how the project was managed and how well it was managed individually. It shows the phases of the project, going from the project idea through to research, requirements gathering, the specification for the project, the design, implementation, and testing phases for the project. It also discusses Trello, GitHub and project developer's journals as tools which assisted in the project management.

7.2 Project Phases

7.2.1 Proposal

The initial proposal for the project was to create an interactive fantasy virtual reality environment. This idea was decided upon as creating games within Unity was considered a strong suit for the project developer. In addition, a virtual reality module had been completed in the previous term, resulting in an interactive application with a similar vision and overall aesthetic design. It was also decided that this project would utilise a different art style to the former. Once this idea was set in motion, it was proposed to the project supervisor.

7.2.2 Research

Before the project could commence, research was undertaken into the area of immersive gaming technologies, and how, with AR and VR, games are only becoming more immersive. During this phase, a literature review was conducted, which examined several sources. Each detailed important design aspects, how they contribute to positive experiences, and how they can be applied to games to provide the player with a memorable experience.

7.2.3 Requirements

Requirements gathering, though not difficult, required a lot of research into the area of VR games and applications. Though this was an area of great interest to the project developer, it is also undoubtedly a young and upcoming technological area. However, similar games were found and researched, the features contained within carefully analysed and noted for inclusion in the final project. In addition, a survey was launched, and interviews were conducted, both to determine a prospective audience, and to find what would entice players new to VR to play such a game. From these results, personas were then drawn up. These reinforced the requirements needed for the game to succeed in its goal.

7.2.4 Design

Wireframes of the game's user interface, as well as how it would appear to the player in VR, were drawn up. In addition, processes contained within, such as obstacles faced by the player, were storyboarded, to get an idea of the game's flow and navigation. As new puzzles and challenges were

imagined, and iterated upon, new storyboards and game screens were designed to aid in their implementation. UI assets, visual style, and colour schemes were also considered.

7.2.5 Implementation

During the implementation phase, the game was developed inside Unity, and the code for the gameplay tasks was written in Visual Studio Code. A lot of the code was reused from previous VR applications developed; however, several tutorials and online guides were also consulted, to speed up the process and learn how to implement tricky features.

There were some issues during this phase; mainly in the form of processes that were harder to implement than imagined. Some of these processes were eventually scrapped, and already existing aspects were instead improved. Others were only partially implemented, but in a way that ensured they were not game-breaking.

7.2.6 Testing

The testing process involved both functional testing, in which the core functions of the game were tested; and user testing, in which the game was tested by others. Functional testing helped in finding bugs and mechanics which were not working optimally, while user testing examined how others found the gameplay. Both sets of results provided a useful insight into areas for improvement, particularly should the game continue to be developed.

7.3 SCRUM Methodology

It was felt that working in sprints was highly conducive to the project development experience. It helped break down each task, or set of tasks, into manageable portions. This in turn ensured that the workload never felt overwhelming – each task felt achievable during the fortnightly timeframe. It also assisted in keeping the project focus on the most important tasks – if a task could not feasibly be achieved within the two weeks, it was deemed best to leave it be, and instead focus on another task.

Alternatively, the fortnightly timeframe of each sprint meant that some tasks could be completed quickly, leaving time to implement items on the project backlog. Again, though, if these took too long to implement, there were left out of the final product.

7.4 Project Management Tools

7.4.1 Trello

Trello is a tool that mainly utilises Kanban-style boards. These boards can be split into lists, which is useful for visualising tasks that are to be done, tasks currently in progress, and tasks completed. It can however be customised in whichever way the user sees fit.

Each item on the list, referred to as a card, can also be given unique colour-coded labels. This helps the user visualise, at a glance, which kind of task the card refers to. For example, a bug needing

attention might be given a red label, whereas a completed task might be labelled as green. Other useful features can also be added to cards, such as checklists and deadlines.

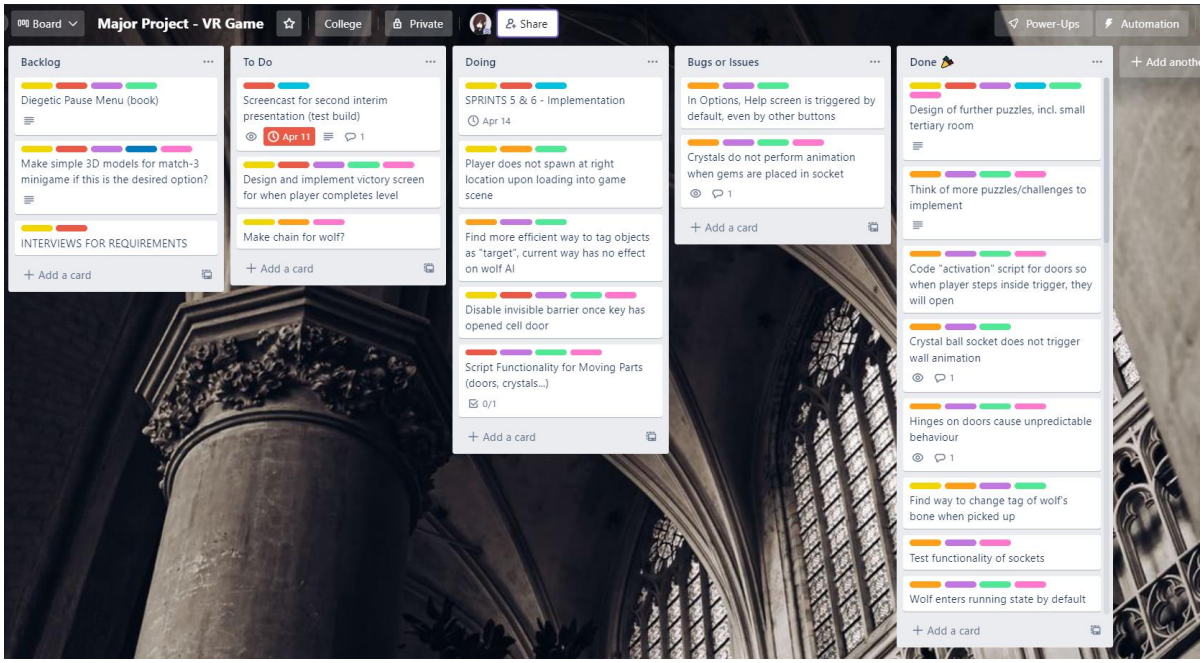


Figure 84: Project Trello board, showing a Backlog list, To Do list, Doing list, Bugs list, and Done list.

Using Trello was a positive experience. It helped the project to stay on track, even when there was a lot to be completed. Separating each task into its own card – sometimes with unique deadlines and/or checklists – reduced the workload into manageable chunks, which in turn made the project less overwhelming. The use of separate lists for tasks to be done, in progress, and completed also made it easier to visualise what remained to be done, and what was currently being done.

The Trello board could also be updated every week with new suggestions from the project supervisor. This ensured that each new suggestion was not forgotten about, and could be implemented in due time.

Overall Trello was a very useful tool, and one which would be utilised in future for other large projects.

7.4.2 GitHub

GitHub is a version management tool, commonly utilised in software development, which allows developers to collaborate on projects locally and/or remotely. Using GitHub, developers can “clone” a software folder, called a repository, from a remote server to their own device. Once changes are made to this local repository, the files can then be “pushed” to the remote repository. In this way, it is ensured that the repository hosted remotely is the latest version. From there, the repository, and its changes, can be “pulled” to a local device by a team member, and the cycle can continue.

This is a good system in theory, but it was not utilised for this project. Unfortunately, GitHub is not optimised for large files, such as those contained in Unity projects. As Unity projects contain a multitude of assets, they often grow to a large file size. GitHub is limited to a maximum upload size of 100MB, and in reality, many Unity projects are not this small.

Large File Storage (LFS) systems are available should a developer need to commit large files – but this was not necessary for this project. As this project was developed from home, there was no need to transfer files to a remote server.

7.4.3 Journal/Notes

Though there was no formal journaling of project ideas/reflections, notes were taken frequently throughout the course of the project. These notes included sprint deadlines and important calendar dates; which tasks to prioritise; and ideas for material to include in the project report. These notes were both taken in a Notepad app and on paper. It was felt that handwriting notes was overall more efficient, as it helped commit to memory areas of particular importance.

Make player spawn at correct location
Match 3/physics puzzle/slide puzzle?

- Introduce a timed element to completion of game?
- Revisit Oculus Quest stats - tris/verts/framerate, etc.

Mention in report how app build crashing was overcome - problem and how it was solved, very useful!

Figure 85: Snippet from notes.

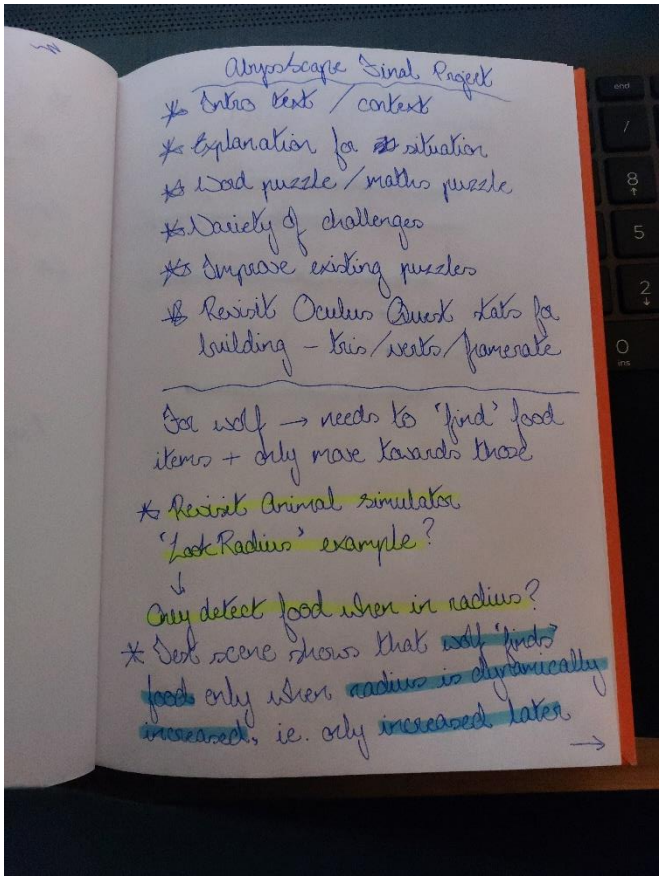


Figure 86: Section from handwritten notes.

7.5 Reflection

7.5.1 Your views on the project

The project ran quite smoothly, all things considered. Though it was frustrating to work on at times, it offered me the opportunity to learn new techniques and overcome new challenges. Working with VR technology was certainly new to me, though this element also kept the project fresh and exciting in comparison to games made previously. It was also not too difficult to implement this functionality, as several features utilised in the application had been practiced before in a prior VR application.

I feel that my experience and knowledge of both Unity and the C# language have improved significantly over the course of the project. This will undoubtedly serve me well should I continue to develop games as a hobbyist.

Overall, though it had its challenges, the project was a positive experience.

7.5.2 Completing a large software development project

As with similar projects, I found the process of completing such a large-scale project quite insightful. It taught me a lot about the importance of planning tasks, sticking to deadlines, and which tasks to prioritise. It also taught me important organisational skills in relation to the visualisation of what was to be done, what was in progress, and what was completed.

Again, as before, it was surprisingly not a difficult project once all the processes involved were broken down into sets of smaller tasks. This helped keep the workload manageable, and as a result, the project never felt overwhelming, even when there was much to be completed.

7.5.3 Working with a supervisor

Working with a supervisor was incredibly helpful throughout this project. Weekly meetings ensured that the project was kept on track, and important deadlines were reached. These weekly meetings helped keep me focused, and increased my motivation to complete tasks before the next. Email communication was frequent; again, this was an immense help in ensuring that deadlines were met. Overall, my supervisor was incredibly supportive of the project, and was always eager to hear about each new development as it arose.

7.5.4 Technical skills

Many useful skills were either learned or improved upon during the course of this project.

For example, as mentioned before, the area of virtual reality games was new to me at the start of the year, and it seemed quite daunting at first. Though I had experience with Unity and C# scripting, I had never before made a game for VR technologies. It was certainly a new experience; however, after completing two assignments throughout the VR module during the term, I felt much more confident in my ability to create games for VR. Added to this was the ease of use of Unity's inbuilt XR Interaction Toolkit; it eased the programming of certain processes.

Several tutorials and guides were also followed during the implementation of certain features; the game was not coded completely from scratch. However, I feel that my overall knowledge of C# scripting has become stronger as a result of developing this game. Not only was I able to iterate on scripts written previously, and adapt them for *AbyssScape,* but I was also able to write code for some processes completely from my own knowledge. A few scripts were not used in the final product; but, even so, they were a useful exercise.

Overall, this project only heightened my level of expertise in Unity and C# scripting, with the added bonus of having experience developing for VR now part of my toolkit.

7.5.5 Further competencies and skills

In terms of further competencies and skills, I would definitely like to continue improving on my Unity skills. To me, Unity, and the C# language, are beginner-friendly, easy to use, and enjoyable once the basics are learned. Having gained even more knowledge of these technologies, I would enjoy putting them into practice in personal projects, and, consequently, improving my skills.

I would also consider making more virtual reality games, if I felt so inclined. Developing for VR was a new and challenging experience for me, but one that I felt paid off. I look forward to making use of the technology in future.

7.6 Conclusion

In conclusion, I feel that this project was managed quite well. There were definitely aspects which could have been handled better - such as fixing problems as they arose, rather than putting them off until later – but, ultimately, these issues did not hamper the development of *AbyssScape* in a meaningful way. This project helped me gain an insight into the value of project management tools when working on large-scale software projects, particularly as to how they help manage the workload. The parts of the project with which I struggled, but overcame in the end, will also serve as a valuable learning experience, particularly should I go on to develop more games in future.

8 Business Opportunities

8.1 Paid Download

An opportunity to monetize the application could come in the form of uploading it to various online platforms, such as Steam or the Oculus Store, for others to purchase and download. Alternatively, platforms such as Itch.io or SideQuest could be used for this purpose.

Itch.io is a website where indie developers can upload games and other interactive applications, such as visual novels, for either free or paid download. Even if a download is free, the site presents the option to support the creator financially, through donations. Uploading *AbyssScape* to a platform like Itch.io – even as a free download – would undoubtedly help it gain popularity.

Another platform frequently used to download and/or purchase software for virtual reality devices is SideQuest. Through this platform, games can be downloaded and loaded directly onto the user's headset without going through the Oculus Store. Consequently, it is frequently used by VR developers. Like Itch.io, SideQuest makes use of both free and paid downloads – so, like the above, it would be ideal for monetization, for gaining popularity, or both.

8.2 VRChat/Public VR Spaces

A frequent use case of virtual reality technology is that of virtual "hang out" spaces. This concept has very much re-entered the public eye since the announcement of Meta's "metaverse" – an online shared space which, utilizing VR and/or AR technology, contains persistent virtual worlds that continue to exist even when you're not playing (Ravenscraft, 2021). These spaces are not games as such; rather, they are places, much like in real life, where players, represented by their avatars, can meet, chat, and interact with one another.

Though there are undoubtedly conflicting ideas at present over what "the metaverse" entails, there are existing applications that function in much the same way. One of the most popular of these is VRChat, an application where users can take on different personas, explore virtual worlds, and hang out with friends over voice chat.

VRChat consists completely of user-generated worlds, developed in Unity. Consequently, with some modifications, *AbyssScape* could become its own VRChat world, either as an interactive escape room, or simply an interactive space for players to explore.

8.3 Interactive Simulations

The realm of virtual reality is not exclusive to gaming; indeed, it has many other uses. VR applications have been used in such areas as healthcare, real estate, and tourism (Thompson, 2017). The immersion provided by VR technology makes it especially ideal for simulations – interactive applications used to help overcome phobias. Computer simulations have been proven to reduce the debilitating effects of phobias such as acrophobia (fear of heights) and arachnophobia (fear of spiders) (Strickland, Hodges, North, & Weghorst, 1997).

With minor modifications, such a game as *AbyssScape* could be used as a tool to help overcome phobias such as claustrophobia (fear of tight spaces) or nyctophobia (fear of the dark).

9 Conclusion

The aim of *AbyssScape* was to create an immersive, puzzling virtual reality dungeon escape, demonstrating not only the immersion offered by games, but the enhancements virtual reality brings to the experience. It would utilise a simplistic, low-poly art style, not only for optimisation purposes, but as a unique stylistic choice.

Overall, the intended goal was to draw on the principles of immersive gaming to provide a virtual reality game that would engage and beguile the player through its environment and interactions.

The game was developed using a robust combination of the Unity game engine and C# coding. Unity is a fun, beginner-friendly development engine, and is well suited to a variety of applications, including 2D and 3D games, mobile applications, VR games, and more. Though it was already a familiar technology, using it to create a VR experience was a new and fascinating experience. It provided many an opportunity for learning and growth.

Throughout the project, many indispensable skills were learned and improved upon. This included C# programming skills, level design and wireframing, lighting, optimisation, and debugging. It also strengthened the user testing principles that are so fundamental to modern application design.

The project remained on course throughout the entire process. There were little, if any, aspects scrapped; most of the planned functionality was implemented in some form.

Undoubtedly, there are many areas in which *AbyssScape* could be improved, and developed. Were the game to become a large-scale project, additions might include more rooms to puzzle through, a wider variety of challenges, an inventory system, and even a boss room, with a large creature, such as a dragon, for the player to defeat. These would all contribute positively to the sense of peril, yet heroism, the game sought to evoke.

References

Allen, P. T. (2018, May 16). A brief history of immersion, centuries before VR. Retrieved October 13, 2021, from The Conversation website: https://theconversation.com/a-brief-history-of-immersion-centuries-before-vr-94835

Armor Games. (2021). *A Rogue Escape*. Retrieved from https://store.steampowered.com/app/1476100/A_Rogue_Escape/

Bannerflow. (2016, July 26). How Pokémon GO and Augmented Reality are changing marketing | Bannerflow. Retrieved October 21, 2021, from Bannerflow website: https://www.bannerflow.com/blog/pokemon-go-augmented-reality-changing-marketing/

Berve, C. (2019, November 10). Ignited Ink Writing, LLC | Book Editor | Website/Blog Content Editor/Writer. Retrieved November 2, 2021, from Ignited Ink Writing, LLC | Book Editor | Website/Blog Content Editor/Writer website: https://www.ignitedinkwriting.com/ignite-your-ink-blog-for-writers/how-to-build-atmosphere-in-creative-writing/2019

Bianchi, F. (n.d.). Coolors. Retrieved February 16, 2022, from coolors.co website: https://coolors.co/

Bura, S. (2008, July 29). Emotion Engineering: A Scientific Approach for Understanding Game Appeal. Retrieved October 7, 2021, from Game Developer website: https://www.gamedeveloper.com/design/emotion-engineering-a-scientific-approach-for-understanding-game-appeal

Changing the Image of a Button When It Is Clicked in Unity. (2020). Available from https://www.youtube.com/watch?v=__M37MbOa8Q&ab_channel=UnityMechanics

Cherry, B. (2020). Why Are Social Media Sites Blue? How Color Psychology Drives Engagement. Retrieved October 20, 2021, from Bluleadz.com website: https://www.bluleadz.com/blog/why-are-social-media-sites-blue

Coffee Duck. (2018). Simple Iron UI Set Pack | 2D Icons | Unity Asset Store. Retrieved February 16, 2022, from Unity Asset Store website: https://assetstore.unity.com/packages/2d/gui/icons/simple-iron-ui-set-pack-124295#content

Creating a Mute Button in Unity. (2021). Available from https://www.youtube.com/watch?v=ZmQAHhZ7784&ab_channel=UnityMechanics

Cyan. (2016). *Obduction*. Retrieved from https://store.steampowered.com/app/306760/Obduction/

Design Wizard. (2019, July 8). Color Theory: The Science and Art of Using Color - Design Wizard. Retrieved October 20, 2021, from Design Wizard website: https://www.designwizard.com/blog/design-tips/color-theory

DesignCloud. (2019). Why is colour theory important? A guide to powerful graphic design. | Manchester Digital. Retrieved October 19, 2021, from Manchester Digital website: https://www.manchesterdigital.com/post/design-cloud/why-is-colour-theory-important-a-guide-to-powerful-graphic-design

donjon; 5e Random Dungeon Generator. (2022). Donjon.bin.sh. Retrieved February 7, 2022, from Donjon website: https://donjon.bin.sh/5e/dungeon/

DVNC Interactive. (2018, June 4). Color Theory in Games - An Overview - Bridging Realities. Retrieved October 7, 2021, from Bridging Realities website: https://dvnc.tech/2018/06/04/color-theory-in-games-an-overview/

Edensor, T. (2015, August). Light design and atmosphere. Retrieved October 7, 2021, from ResearchGate website: https://www.researchgate.net/publication/281336144_Light_design_and_atmosphere

FireproofGames. (2019). *The Room VR: A Dark Matter*. Retrieved from https://www.fireproofgames.com/games/the-room-vr-a-dark-matter

Fussell, G. (2020, November 24). The Psychological Meanings Behind Familiar Shapes (And How to Use Them). Retrieved October 13, 2021, from The Shutterstock Blog website: https://www.shutterstock.com/blog/psychological-meaning-shapes-use

Häger, E. (2017). *Enhanced Immersion in Augmented Reality Applications*. Retrieved from https://liu.diva-portal.org/smash/get/diva2:1183609/FULLTEXT01.pdf

HansCo. (2020). Traditian Font | dafont.com. Retrieved February 10, 2022, from dafont.com website: https://www.dafont.com/traditian.font

How to make a Quiz Game with Multiple Choices in Unity. (2020). Available from https://www.youtube.com/watch?v=G9QDFB2RQGA&ab_channel=TheGameGuy

Iyer, A. (2018, June 10). Bastion: How to Design an Atmospheric Video Game - SUPERJUMP. Retrieved October 7, 2021, from Medium website: https://superjumpmagazine.com/bastion-how-to-design-an-atmospheric-video-game-cfc4a435d5a9

James, L. (2019, May). How to Create Atmosphere in Video Games – GameSpew. Retrieved October 7, 2021, from GameSpew website: https://www.gamespew.com/2019/05/create-atmosphere-in-games/

Jarvis, P. (2014, February 25). The importance of emotion in design. Retrieved November 3, 2021, from TNW | Dd website: https://thenextweb.com/news/importance-emotion-design#.tnw_foJRpGmA

Komninos, A. (2020, March 22). Emotion and Design. Retrieved October 13, 2021, from The Interaction Design Foundation website: https://www.interaction-design.org/literature/article/emotion-and-design

Kumar, J. M., Herger, M., & Dam, R. F. (2018, November 20). A Brief History of Games. Retrieved October 13, 2021, from The Interaction Design Foundation website: https://www.interaction-design.org/literature/article/a-brief-history-of-games

Mairi. (2022, January 13). The Best VR Escape Rooms on Oculus Quest 2 - The Escape Roomer. Retrieved May 3, 2022, from The Escape Roomer website: https://theescaperoomer.com/the-best-vr-escape-rooms-on-oculus/

McRae, E. (2017, August 9). Designing game environments that are rich with story. Retrieved October 7, 2021, from EDWIN MCRAE website: https://www.edmcrae.com/article/designing-game-environments-that-are-rich-with-story

Mehrafrooz, B. (2020, June 11). 10 Most Common Challenges of Designing Great Game Environments - Pixune. Retrieved October 7, 2021, from Pixune website: https://pixune.com/9-most-common-challenges-in-game-environment-design/

Meyer, B. D. (2016, August 3). Evoking emotion in pure sound design. Retrieved October 13, 2021, from Designingsound.org website: https://designingsound.org/2016/08/03/evoking-emotion-in-pure-sound-design/

MK Illumination. (2020). Creating atmosphere at Christmas markets yields measurable results. Retrieved November 1, 2021, from MK Illumination website: https://www.mk-illumination.com/article/creating-atmosphere-at-christmas-markets-yields-measurable-results/

Mütterlein, J. (n.d.). *The Three Pillars of Virtual Reality? Investigating the Roles of Immersion, Presence, and Interactivity*. Retrieved from https://scholarspace.manoa.hawaii.edu/bitstream/10125/50061/paper0174.pdf

Nadri, T. (2020). Standrag Font | dafont.com. Retrieved February 10, 2022, from dafont.com website: https://www.dafont.com/standrag.font

Overflow. (2017). *Conductor*. Retrieved from https://store.steampowered.com/app/584930/Conductor/

Ravenscraft, E. (2021, November 25). What Is the Metaverse, Exactly? Wired; WIRED. https://www.wired.com/story/what-is-the-metaverse/

Ribeiro, G., Rogers, K., Altmeyer, M., Terkildsen, T., & Nacke, L. E. (2020). Game Atmosphere. *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. https://doi.org/10.1145/3410404.3414245

Roach, B. (2012). Elementary Gothic Bookhand Font | dafont.com. Retrieved February 10, 2022, from dafont.com website: https://www.dafont.com/elementary-gothic-bookhand.font

Rossi, C. (2020). Psychology of light: creating the right atmosphere for every environment. Retrieved October 13, 2021, from Karmanitalia.it website: https://storybox.karmanitalia.it/en/psychology-light-creating-atmosphere

Ryu, H. (2020, August 3). How to bring your game world to life with impactful sound design. Retrieved October 19, 2021, from GamesIndustry.biz website: https://www.gamesindustry.biz/articles/2020-08-03-how-to-bring-your-game-world-to-life-with-impactful-sound-design

Schafer, S. B. (2011). Handbook of Research on ICTs for Human-Centered Healthcare and Social Care Services. Retrieved November 2, 2021, from Google Books website: https://books.google.ie/books?id=5dGeBQAAQBAJ&pg=PA47&lpg=PA47&dq=immersion+is+a+process+of+temporarily+expanding+consciousness+into+areas+of+the+unconscious%E2%80%94something+like+hypnosis,+but+retaining+consciousness+as+one+does+in+lucid+dreaming+states&source=bl&ots=9o3TCpyW7p&sig=ACfU3U1CC8PjlJofFlj9wqGwozUDUG7iyw&hl=en&sa=X&ved=2ahUKEwjcg7qq2frzAhULHcAKHT40DQAQ6AF6BAgQEAM#v=onepage&q=immersion%20is%20a%20process%20of%20temporarily%20expanding%20consciousness%20into%20areas%20of%20the%20unconscious%E2%80%94something%20like%20hypnosis%2C%20but%20retaining%20consciousness%20as%20one%20does%20in%20lucid%20dreaming%20states&f=false

Schell Games. (2021). *I Expect You To Die*. Retrieved from https://iexpectyoutodie.schellgames.com/

Scruton, R. (2021). aesthetics | Definition, Approaches, Development, Meaning, Examples, & Facts | Britannica. In *Encyclopædia Britannica*. Retrieved from https://www.britannica.com/topic/aesthetics

Shelley, J. (2017). The Concept of the Aesthetic (Stanford Encyclopedia of Philosophy). Retrieved November 2, 2021, from Stanford.edu website: https://plato.stanford.edu/entries/aesthetic-concept/#ConAes

Slater, M. (2018). Immersion and the illusion of presence in virtual reality. British Journal of Psychology, 109(3), 431–433. https://doi.org/10.1111/bjop.12305

Stankovic, S. (2021, June 6). The shape, the color, and the emotion: Angry Birds' character design. Retrieved October 7, 2021, from Medium website: https://uxdesign.cc/character-design-of-angry-birds-4a74feb90e8c

Strickland, D., Hodges, L., North, M., & Weghorst, S. (1997). Overcoming phobias by virtual exposure. Communications of the ACM, 40(8), 34–39. https://doi.org/10.1145/257874.257881

Thomas, F. (2018). Why good sound design matters. Retrieved October 21, 2021, from Productionattic.com website: https://www.productionattic.com/blog/why-good-sound-design-matters

Thompson, S. (2017, April 6). VR Applications: 23 Industries using Virtual Reality. Retrieved May 3, 2022, from Virtualspeech.com website: https://virtualspeech.com/blog/vr-applications

Tyler, D. (2017, March 11). Video Game Sound Design | Beginner's Guide. Retrieved October 19, 2021, from Gamedesigning.org website: https://www.gamedesigning.org/learn/video-game-sound/

Vunira Design. (2020). Dragon Font | dafont.com. Retrieved February 10, 2022, from dafont.com website: https://www.dafont.com/dragon-3.font

W, N. (2018, November 15). VR & diegetic Interfaces: don't break the experience! Medium; UX Collective. https://uxdesign.cc/vr-diegetic-interfaces-dont-break-the-experience-554f210b6e46

Wizards of the Coast. (1993). *Dungeons & Dragons.* Retrieved from https://dnd.wizards.com/

Appendix

There was a lot of content contained in this report which was too long to include in the respective chapters. That information can be found here.

Item 1: Interview Results

A wide variety of answers were received from different types of gamers. These answers can be seen below.

(10/02/2022) Interview conducted with a 25-year-old gamer from Dublin, Ireland

Do you play games? Yes

Apart from playing games, what other hobbies do you have? Guitar, Cycling

How old are you? 25

What is your occupation? Consultant Engineer

What is your highest level of education? Honours Degree

What games do you typically buy and why? Racing, shooters, puzzle games, I have a wide variety of games I enjoy

Do you read games reviews and why/why not? No, I watch trailers and use these as a guide

Do you write reviews for games? No

What are your main reasons for playing games? To relax and spend time with my friends who live abroad

Have you ever played a virtual reality game? If so, which? Yes, Beat Sabre, resident evil

If you haven't played any VR games, would you consider it?

What kind of VR game or experience do you/would you most enjoy? An immersive simulation

How many hours do you play games each day on average? 2-3

How many hours do you/would you play VR games? 1-2

Do you find that you lose track of time easily when playing games? yes

Why do you think this happens? (Engrossing story, gameplay, etc.) Story line or fun with friends and chatting

In your opinion, would a game feel lacking/incomplete if missing key elements, such as sound, UI, or characters? yes

Would you prefer VR games or non-VR games, do you think? Non-VR

What player type describes you best, and why? Relaxed but immersed, get engrossed in an enjoyable game

(10/02/2022) Interview conducted with a 21-year-old gamer from Dublin, Ireland

Do you play games? yes

Apart from playing games, what other hobbies do you have? Filmmaking, writing, cigars, music, exercising

How old are you? 21

What is your occupation? Student

What is your highest level of education? PLC L5

What games do you typically buy and why? Nintendo (Mario, Animal Crossing, Dr. Kawashima etc.) PC (SimCity, Sims, edutainment games from childhood), GTA

Do you read games reviews and why/why not? Only afterwards so I don't taint my own experience

Do you write reviews for games? Nah

What are your main reasons for playing games? I like being able to sink my teeth into the games and immerse in them

Have you ever played a virtual reality game? If so, which? Nah

If you haven't played any VR games, would you consider it? All day

What kind of VR game or experience do you/would you most enjoy? One that's immersive

How many hours do you play games each day on average? 2-3

How many hours do you/would you play VR games? 2-3?

Do you find that you lose track of time easily when playing games? YES

Why do you think this happens? (Engrossing story, gameplay, etc.) Get too lost in it

In your opinion, would a game feel lacking/incomplete if missing key elements, such as sound, UI, or characters? Yea but it depends on the game, I'm typically picky before buying anyway

Would you prefer VR games or non-VR games, do you think? Non

What player type describes you best, and why? ~~angry~~ Passionate

(11/02/2022) Interview conducted with a 27-year-old gamer from Meath, Ireland

Do you play games? Yes

Apart from playing games, what other hobbies do you have? Cosplay, illustration, fursuit making

How old are you? 27

What kind of VR game or experience do you/would you most enjoy? I have only played vrchat, but I really enjoyed getting to share a space with my friends in a virtual world even when we were miles apart

What is your occupation? Student

What is your highest level of education? Level 8

What games do you typically buy and why? Story heavy, single player rpg's. Story is one of the most important things for me.
I also buy a lot of pixel art games as I love that style, and you usually find more projects by indie developers in that genre, which usually means more unique and surprising experiences

Do you read games reviews and why/why not? Yes, I love to get a sense of what other people think about games, and I don't have a lot of expendable income, so I want to make sure each purchase is well researched

Do you write reviews for games? Extremely rarely, I want to make more though

What are your main reasons for playing games? It's a great de-stresser and a fantastic source of inspiration

Have you ever played a virtual reality game? If so, which? Only vrchat

If you haven't played any VR games, would you consider it? Yes I would

How many hours do you play games each day on average? Currently 0.5 as I'm in my final year of college, outside of college hours it could be 1 or 2 hours a day, sometimes 8 if I'm going on a binge

How many hours do you/would you play VR games? Currently 0, and I've only played about 12 hours of vrchat in the last 2 years, but I want to make a personal model and play more after college

Do you find that you lose track of time easily when playing games? Yes, very much so!

Why do you think this happens? (Engrossing story, gameplay, etc.) Engrossing story for sure, but also immersive and challenging gameplay. If I don't feel challenged in a game, I can tend to lose interest a lot faster. Also, relatable and interesting character arcs is a huge reason for losing track of time playing.

In your opinion, would a game feel lacking/incomplete if missing key elements, such as sound, UI, or characters? For most games yes, I think it would take a lot of skill to make a successful game missing these key features, but it has been done in the past (eg, stanley parable with no character development)

Would you prefer VR games or non-VR games, do you think? Non vr personally, but that might be heavily due to the fact that up until recently, my laptop couldn't handle vr games, and as I'm in final year, I haven't really been able to try playing vr games yet

What player type describes you best, and why?
I think I'm a retro gamer, I tend to prefer older titles, or remasters of old games (eg. Spyro Reignited). I'm a girl gamer, with a focus on rpg's, fantasy, and games with romance systems. I also like to seek out rare and obscure games, and I have a lot of in depth knowledge on a wide variety of games, most of which I

never even played, but I researched from getting interested in watching Let's Plays on YouTube, and falling in love with the characters and story.

(11/02/2022) Interview conducted with a 22-year-old gamer from Dublin, Ireland

Do you play games? Yes, of the tabletop, card and video varieties

Apart from playing games, what other hobbies do you have? Reading non-fiction books (history especially), reading academic papers, cosplay, music production, video production, drawing, electric guitar, finding new musical genres, learning about computers, game design, linguistics, conlanging (long list, I could go on)

How old are you? 22

What is your occupation? N/A, previously student but currently unemployed

What is your highest level of education? Level 8 Bachelor's Degree

What games do you typically buy and why? Shin Megami Tensei, Resident Evil, Monster Hunter, Doom, various indie games

Do you read games reviews and why/why not? Not usually. I find looking at gameplay and reading developer provided descriptions of a game to be more useful than user reviews. The majority of people could hate a game but I could love it (e.g Shin Megami Tensei: Devil Summoner: Raidou Kuzunoha versus the Soulless Army), likewise there can be massive acclaim for a game (e.g Elden Ring, Cyberpunk 2077, Read Dead Redemption 2, Doom 4) but I am completely uninterested and as such the user reviews mean little to me. Developers describing their game allows me to see if it contains elements I am seeking but gameplay allows me to understand more efficiently, what a game is offering. Seeing gameplay of Dusk made me want to buy it immediately.

What kind of VR game or experience do you/would you most enjoy? VRChat is a casual chatroom more than a game but I find that quite enjoyable for conversing with friends. As for games by a stricter definition, combat and other mechanics requiring tactility seem quite weak unless there is some way to simulate weight and feedback, otherwise less interactive games may work better such as visual novels. That being said, Beat Saber has succeeded without feedback or weight simulation as that is not its focus. For where virtual reality as a technology is currently, I do not think reaction based games fare well due to the limitations. Card games and tabletop games (e.g Tabletop Simulator) could be enjoyable as a substitute if players cannot join in person for geographic or other reasons and if that was the case, I might enjoy those games as I prefer them in person infinitely more than online.

How many hours do you play games each day on average? Much less than I used to. At most perhaps an hour or two per day now but it used to be three to five.

How many hours do you/would you play VR games? Zero currently. If I had a VR setup likely not very much save exceptional circumstances as they require more setup than simply opening a Steam game and changing some keyboard bindings.

Do you find that you lose track of time easily when playing games? It can happen depending on the game, it happened once in Skyrim when I was twelve. It happened nearly constantly with Morrowind. I found myself losing time with Shin Megami Tensei III and Soulless Army,

Do you write reviews for games? No because I do not think anybody cares what I have to say and I do not find user reviews helpful, as such I do not participate in writing them even though I am aware other people value them.

What are your main reasons for playing games? To be challenged and overcome difficulty, to understand how the mechanics work in tandem and if I adore a game enough, to understand the backend of how everything operates to then exploit and challenge myself further. One excellent example is Doom where I will incessantly search for more information on its inner workings or YuGiOh where I want to understand and play as many Archetypes as possible, specifically those that appeal to me but I want to understand how the game works more generally by doing that.

Have you ever played a virtual reality game? If so, which? Yes, I have played VRChat but never in VR as my PSVR has never worked properly even with TrinusVR.

If you haven't played any VR games, would you consider it? See answer 10.

along with Soul Hackers. I can also lose track of time playing Doom 1 and 2 along with map packs such as Sunlust and Sigil. I also lose time playing Resident Evil Outbreak with a friend. I also lose a lot of time playing Factorio...a lot of time.

Why do you think this happens? (Engrossing story, gameplay, etc.) In the case of Skyrim, it was immersion in the world, which I have never experienced since. For Morrowind it was fascination by the world but also the mechanics encouraging me to be constantly vigilant and keeping my attention as a result, whilst simultaneously having calm dirt path walks between settlements to ease the tension. I lose time in Doom, SMTIII and Soulless Army from wanting to defeat a boss and focusing all of my attention on defeating it. I lose time from Resident Evil Outbreak as my friend and I strategise on optimising our runs and figuring out the map to complete it with minimal issue. I lose time playing Factorio because of its ability to make me the player want to improve the efficiency of my factory endlessly and iterate without end, along with it allowing me to do it alongside encouraging it.

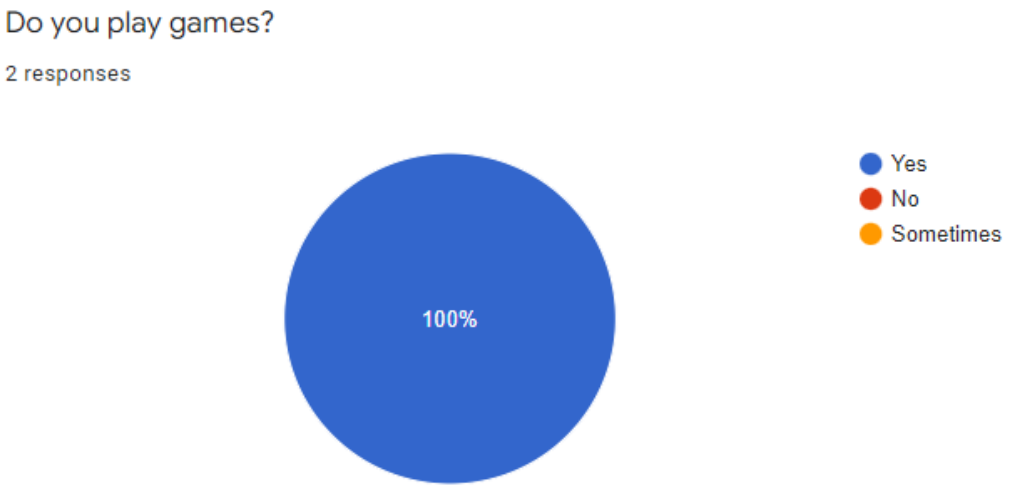
In your opinion, would a game feel lacking/incomplete if missing key elements, such as sound, UI, or characters? Absolutely, despite being so mechanically focused I do still notice sub optimal options menus, missing textures, sounds, broken sounds, low bitrate audio, low quality textures and other such missing elements. These missing elements and faults do factor into my overall impression of a game negatively as it lacks basics. If a game lacks a perfect polish, I am accepting of that, especially for indie games but missing basic features is a red flag.

Would you prefer VR games or non-VR games, do you think? Non-VR for now as the setup process for VR is more involved and there are fewer VR games available than non-VR. I also lack space for proper VR gameplay in my room.

What player type describes you best, and why?
I adore mastery and understanding the mechanics of a game, how it operates and how to play within the rules the game has set before me. I very much appreciate games that are explicitly games and do not try to hide that away by being 'cinematic' or story driven. I can enjoy some visual novels but only in small doses and I had to exert considerable effort to file them as anime/books rather than games to set my expectations accordingly. I do also love an intriguing atmosphere a la Morrowind, Shin Megami Tensei III, Resident Evil or even Doom 1. If a game has highly engaging gameplay with great mechanical depth and an ominous dark atmosphere it is likely an instant buy from me.

Item 2: User Testing Results

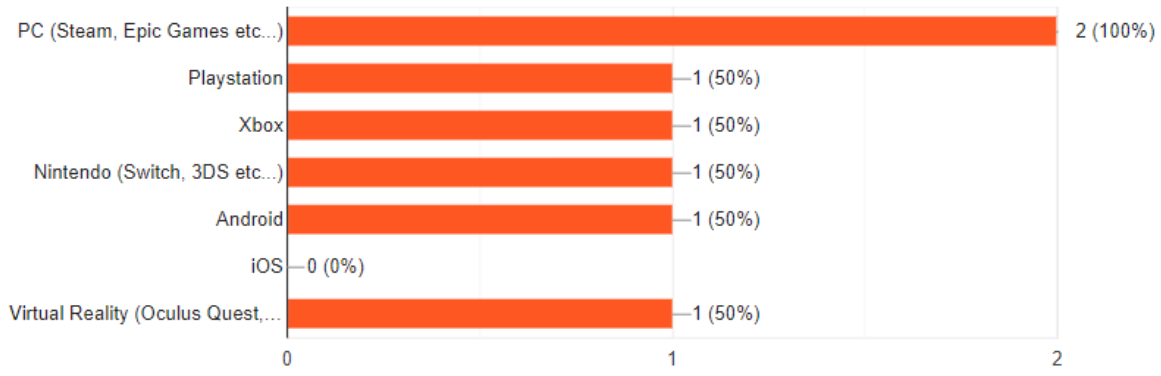
Pre-test Questionnaire Results



What are your preferred platforms?

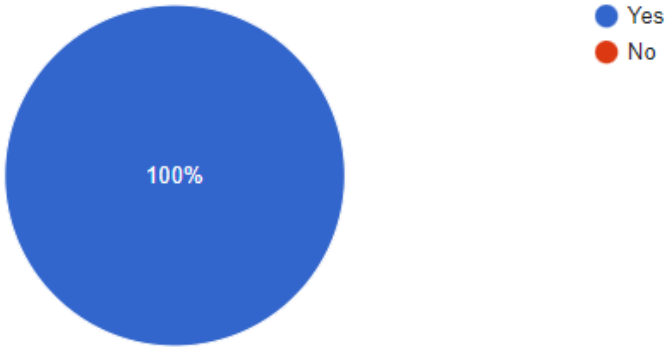
Copy

2 responses



Have you ever tried a Virtual Reality game?

2 responses

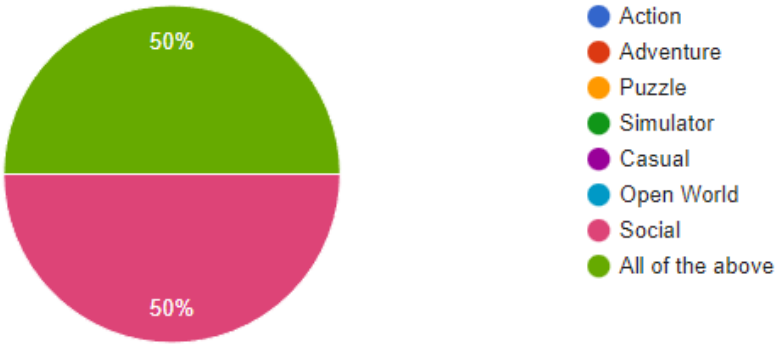


Does the idea of virtual reality games excite you?

2 responses

What type of game, do you think, would you most like to see on VR platforms?

2 responses



Broadly speaking, what is your favourite game genre?

2 responses

Racing

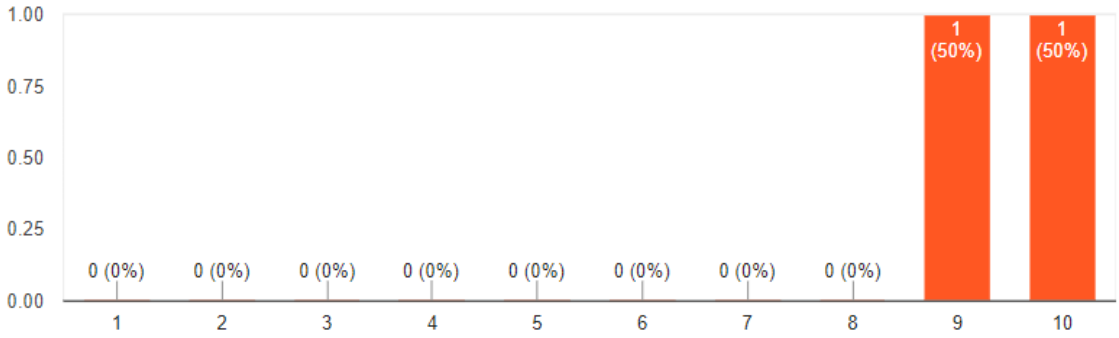
Platformer

Post-test Questionnaire Results

On a scale of 1 - 10, how easy was the main menu to navigate?

Copy

2 responses



If you could improve anything about the Main Menu, what would it be?

2 responses

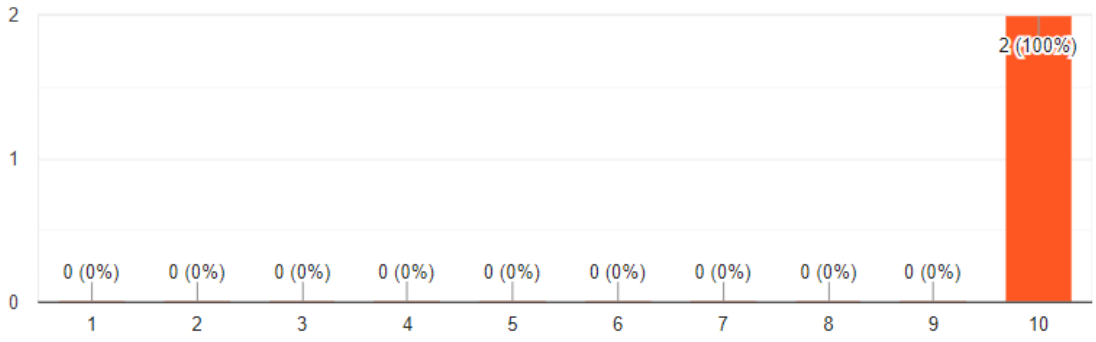
N/A

the options menu, could be more obvious which is on and off

On a scale of 1 - 10, how easy to use was the locomotion (teleport) system?

Copy

2 responses



If you could change anything about the locomotion system, what would it be?

2 responses

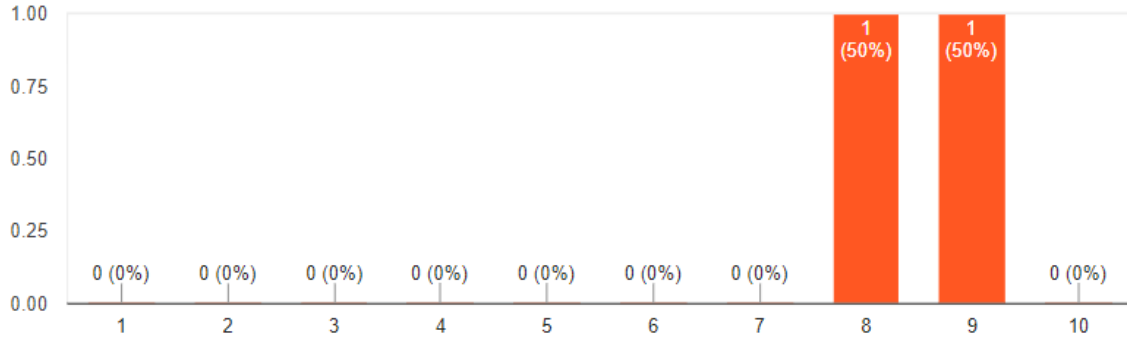
N/A

the distance you can teleport could be limited to experience walking more in depth

On a scale of 1 - 10, how easy was it to interact with items?

Copy

2 responses



If you could change anything about the interactions with items, what would it be?

2 responses

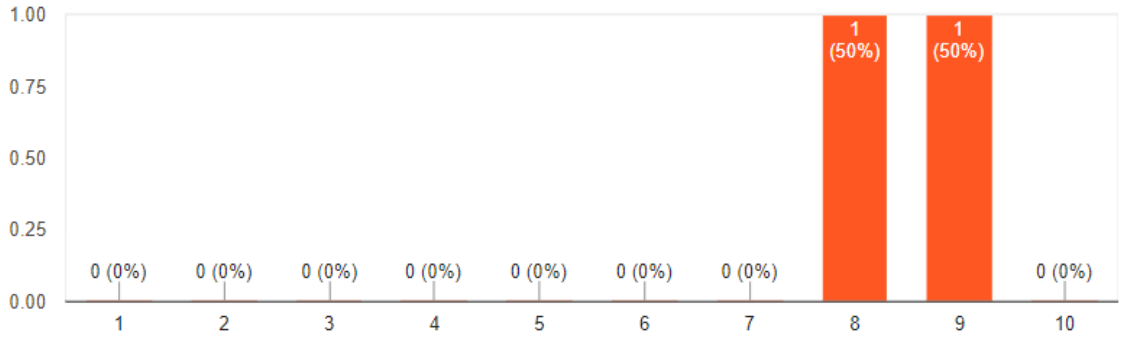
N/A

Small items had fallen through certain small sections of the floor

On a scale of 1 - 10, how easy was it to locate necessary items?

Copy

2 responses



How would you make items easier to locate?

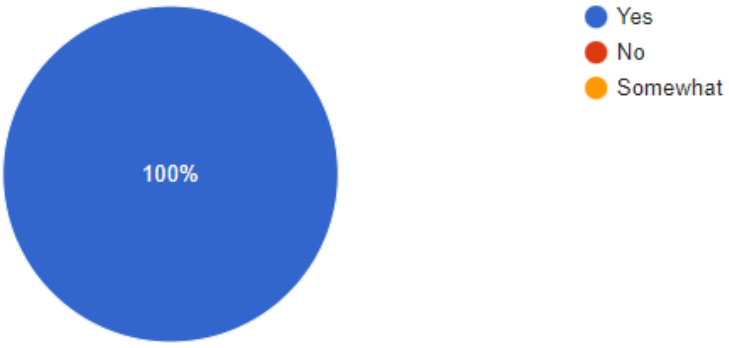
2 responses

N/A

highlight the item when selecting with the ray

Did you like the graphics style of the game?

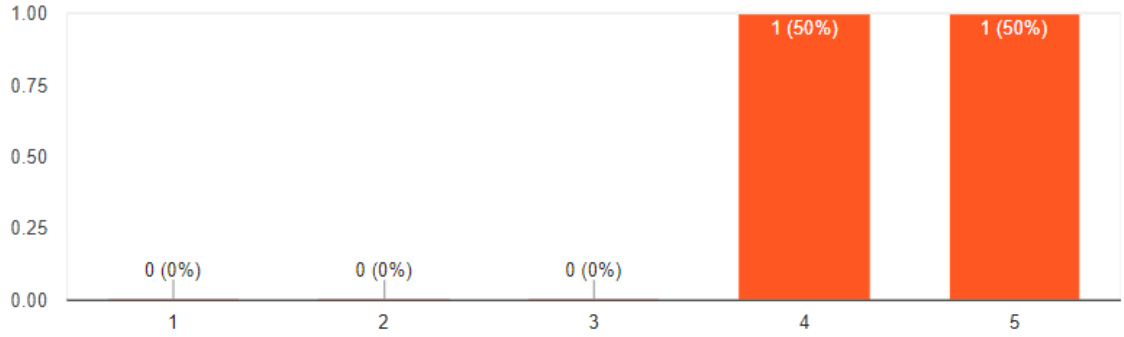
2 responses



How comfortable was the VR experience provided by this game?

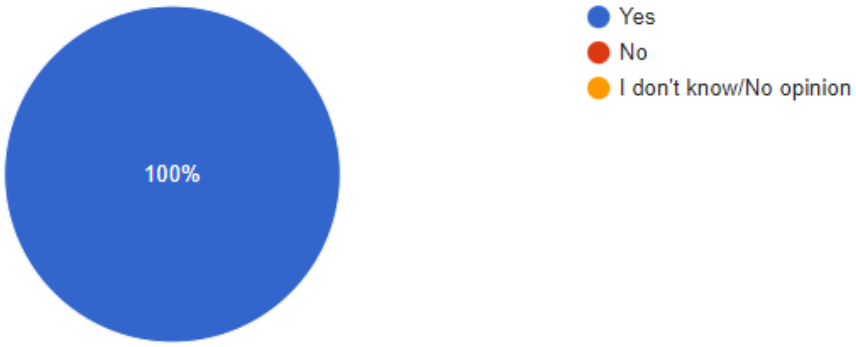
Copy

2 responses



Did the game immerse you in its world?

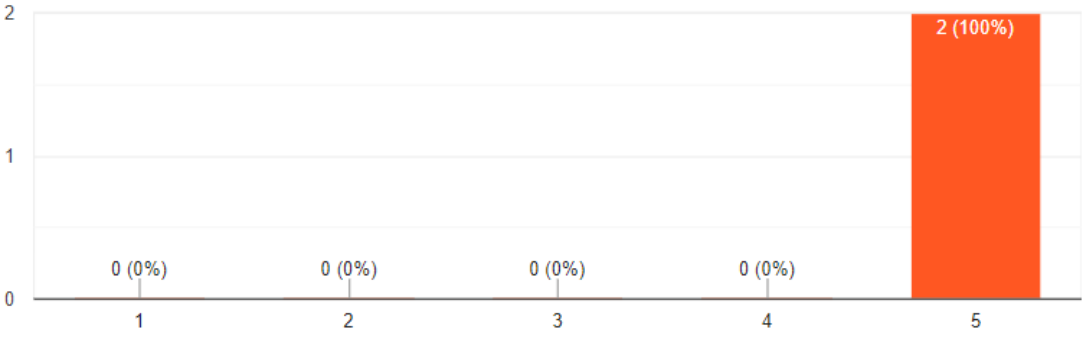
2 responses



Overall, how much did you enjoy this game?

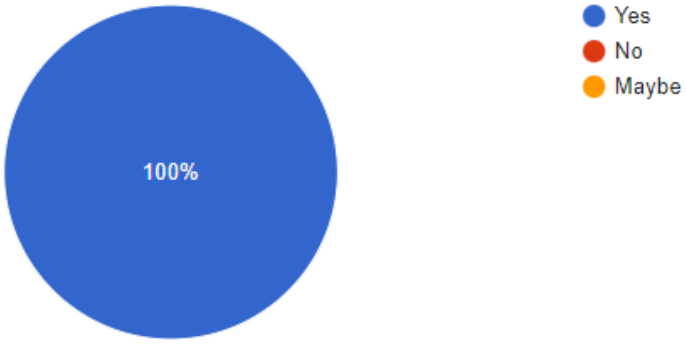
Copy

2 responses



Would you recommend this game to others?

2 responses



Overall, how would you rate the usability of this game?

Copy

2 responses