Implementation of an Immersive Environment powered by Virtual Reality and an Exploration of Distortions of Perception

Final Project Report

TU856
BSc in Computer Science

Eurico Carajote
C17442362
Supervisor

DR. Michael Collins

School of Computer Science

Technological University, Dublin

6/4/2021

Abstract

Immersive environments are digitally mediated learning environments designed to engage users in an artificially created make-believe "world." Immersive environments may take on a broad range of forms, with affordances for varying degrees of sensory immersion and awareness of the user's physical self or the presence of others(1).

Virtual Reality (VR) brings the concept of an Immersive environment a step further by indulging the user in a sense of presence by submerging him in visual and auditory cues that are not representative of their true environment, bring a sense of escapism from one's true surroundings into a world that is not really there.

The aim of this project is to create and implement an immersive world that explores and distorts the perception of reality to create an intriguing narrative for the user.

This project will be carried on using existing technologies that are widely used in the creation of VR content such as Unity, C#, and Blender.

This project is influenced by the natural world and its complex ecosystem as the foundation of the world, and the intricacies of Irish folklore and mythology as the primary inspiration for the surreal imagery and the twisted perceptions of reality that make this experience unique.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Eurico Carajote

6/4/2021

Acknowledgements

I would like to express my deepest gratitude and appreciation towards my friends, family, and girlfriend for supporting me through the process of this project.

A special thanks to DR. Brian Vaughan for granting me the ability to use the Oculus headset for purposes of development and testing.

And a special thanks to DR. Michael Collins for his input, patience, and guidance for the duration of this project, the input and motivation received were invaluable.

Table of Contents

Table of Figure

1. Introduction

1.1. Introduction

This chapter will serve as a guide for the rest of this document, and will also explore what inspired this project, together with its main focus.

Immersive environments are digitally mediated learning environments designed to engage users in an artificially created make-believe "world." Immersive environments may take on a broad range of forms, with affordances for varying degrees of sensory immersion and awareness of the user's physical self or the presence of others(1).

This project came to fruition as a means of escapism to the world, as the current times seem to be dull and uneventful a need to escape to a different reality arises, no longer is there a possibility to explore our surroundings, to step outside in the comfortable piece of mind once had, VR although presents us with an alternative, an escape from reality in the comfort of our homes, the ability to explore and experience something different, and given this alternative the main idea for this project was conceived, to create a IVE capable of inducing a sense of presence and immersion to the user, a simple yet arduous goal.

This project is mainly focus on delivering an immersive experience that fills the user with a sense of presence into this IVE, together with delivering a beautiful visual and auditory experience that both fascinates and intrigues.

1.2. Project Background

This project was inspired by a hike through the Dublin mountains way, at the time of the hike it was known that the project was going to be based in VR, but a main idea was still lacking, but through the course of the hike inspiration was given by the beautiful surroundings and natural views, as well as the feeling of exploration given by walking through new surroundings (collection of pictures taken during that hike that serve as visual inspiration for this project can be viewed in the appendix), it was in those surroundings that an observation was made, wouldn't everyone be happier if they experienced this feeling in their life.

But the path is arduous and not everyone lives somewhere with access to such beautiful sights. This is where the main idea of VR seemed to fit perfectly, through VR anyone could feel such sense of beauty and immersion with the world. But how could this experience be made even better? That was the main question made throughout the hike.

This was when an idea to couple the beauty of nature with the unknown of the magical, this is when a decision to implement a sense of surrealism into the project, to test the boundaries of human perception of reality so that each step led to the unexpected and that unexpected leads to a sense of delight was made.

Now only a question remains, how does one achieve this, and the answer to the question would be to attempt to develop the most immersive environment possible, an environment that transports the user to a different reality providing a deep sense of presence.

1.3. Research focus

In this project the main research focus is to find out how to produce an immersive and presence inducing experience in the aims to produce a system that is engaging and has the ability to transport the user into a different world altogether. Furthermore, this project will also have a focused research in design and testing methodologies so that a viable development plan can be formed to ensure that this project can benefit the most possible from the time and resources available.

To achieve this immersion and presence will need to be researched and understood to a basic level as to make their implementation felt throughout the project felt.

Furthermore, the project aims to mimic real life natural beauty and for that an understanding of how the human brain reacts to colours and different objects is needed, so visual saliency and colour principles will also be researched.

The design of this project will be taken very seriously, this is why so much of the literature review chapter will be dedicated to finding out more about different Agile methodologies and testing methodologies, as it is a core belief that a good design leads to a good project.

Software and hardware are also important research focuses as the right software or hardware can propel this project into new hights, this is why careful research will also be conducted in these topics, ranging from different HMDs (Head mounted displays) to software needed to produce good, realistic sounds throughout the project, and software that allows for the modelling and animation of 3D entities that will be present throughout the project as aims to best express the creative visual vision aimed for this project.

1.4. Document Structure

This section will present a brief overview of each of the sections in this document.

1.4.1. Literature Review

This chapter will be comprised of all the research done for this project, from Immersion and presence to colour psychology principles, visual saliency, and experiences of body transfer in VR.

This section will also research different Agile methodologies and different testing methodologies to be considered for the project, together with different software research in the topics of audio and 3D modelling.

1.4.2. Prototype Design

This chapter will explore the design of the project, from a brief overview of what the project main systems will be, to detailed use case diagrams and class diagrams, in this section design methodology requirements will be presented together with a final choice on the design methodology to be used with respect to the research done in the literature review section.

Given all the research and requirements provided, a final conclusion on the development design will be given to affirm and explain the methodology choice together with all software and hardware choices that were made.

1.4.3. Experiment Development

This chapter will present a view of the development process, it will start with a brief introduction of Unity followed by an overview of the features presented in the project and how they interact whit each other and how they are presented inside the Unity editor.

This will be followed by a code review of some of the most interesting and difficult pieces of code presented throughout the development of this project.

This chapter will be heavily supplemented by code snippets and screenshots of the Unity editor.

1.4.4. Testing and Evaluation

This chapter will be comprised of the different testing methods used in the course of this project, a look at Unit testing in unity and how Unit tests are conducted will be present together with an explanation of play testing and example tests performed in this manner.

A user evaluation will also be present this this chapter, this evaluation will present the users opinions on the project together with opinions and feature recommendations.

1.4.5. Conclusion

This chapter will be a conclusion to the document and project, it will be comprised of a view of issues expected, these issues were drafted before any development started, and a view of encountered issues, these are the actual issues encountered though out the development of the project.

This chapter will also contain a future work section, this section will indicate all work that was hoped to be part of the project and simply could not be completed in time, it will also contain features that were ideals for the project, these features will mostly consist of extra visual elements and or added diversity as well as features requested by test users.

2. Literature Review

2.1. Introduction

In this chapter a review of relevant research and other software is presented as it relates to the development of an immersive environment and development in VR. First existing research that looks into immersion in VR experiences will be presented, and following that, the technologies that might be used in this experience (including but not limited to game engines, modelling tools, programming language, audio editors, head mounted display (HMD)). Other research including academic papers, books, web articles, and technical standards, are presented following this, a research into different Agile methodologies and finally a research on different testing methods.

2.2. Background research

In this section all the components needed to create a immersive and compelling VR experience will be explored, such components involve 1)Immersion and Presence, the feelings of immersion and presence go hand in hand in transporting the user into the virtual environment and they will both be researched extensively to produce the best results in this virtual experience, 2)Colour psychology principles, colour and light will enhance the users immersive experience throughout this visual experience, 3)Experiences of body transfer while in VR, one of the greatest things about VR is that one can achieve a first person experience of body transfer, in this section the methodology behind achieving such an experience will be researched and reviewed with hopes of applying it to the final project, 4)Visual saliency which will provide ways to make different sections of the world stand out more than others enabling the user to easily identify points of interest in the world.

2.2.1. Immersion and Presence

The main aim of this project is to create an immersive virtual environment and for that immersion and presence must be researched extensively.

Immersion means the player is caught up in the world of the game's story (the diegetic level), but it also refers to the player's love of the game and the strategy that goes into it (the nondiegetic level). It seems clear that if we are talking about immersion in video games at the diegetic level and immersion at the nondiegetic level, then we are talking about two different things, with possibly conflicting sets of aesthetic conventions(2). This project is focused on the diegetic level of immersion as the experience is meant to be a visual and auditory experience meant to induce a calm and relaxed sense of self to the user, the appeal comes from a world filled with intriguing and exciting scenery and a sound track that can transport the user into this different world, and avoiding a sense of difficulty or danger that would be associated with an experience built to challenge the user this will eliminate the strategic and tactical immersion that many virtual experiences tend to use to achieve a sense of flow(a state of concentration so focused that it amounts to absolute absorption in an activity(3)) and instead will focus on a sense of presence. Presence and Immersion are very often interchangeable and so definitions for both will be presented before researching specific methods to make this experience both more immersive and methods to increase the sense of presence inside this virtual world.

The following passage is the most accepted definition of immersion:

A stirring narrative in any medium can be experienced as a virtual reality because our brains are programmed to tune into stories with an intensity that can obliterate the world around us. . . . The experience of being transported to an elaborately simulated place is pleasurable in itself, regardless of the fantasy content. We refer to this experience as immersion. Immersion is a metaphorical term derived from the physical experience of being submerged in water. We seek the same feeling f rom a psychologically immersive experience that we do from a plunge in the ocean or swimming pool: the sensation of being surrounded by a completely other reality, as different as water is from air, that takes over all of our attention, our whole perceptual apparatus. . . in a participatory medium, immersion implies learning to swim, to do the things that the new environment makes possible . . . the enjoyment of immersion as a participatory activity(2).

A description of presence:

Presence is closely related to the phenomenon of distal attribution or externalization, which refer to the referencing of our perceptions to an external space beyond the limits of the sensor y organs themselves. In unmediated perception, presence is taken for granted-what could one experience other than one's immediate physical surroundings? However, when perception is mediated by a communication technology, one is forced to perceive two separate environments simultaneously: the physical environment in which one is actually present, and the environment presented via the medium. . . . Telepresence is the extent to which one feels present in the mediated environment, rather than in the immediate physical environment. . . . Telepresence is defined as the experience of presence in an environment by means of a communication medium. . . .In other words, "presence" refers to the natural perception of an environment, and "telepresence" refers to the med iated perception of an environment. This environment can be either a temporally or spatially distant "real" environment (for instance, a distant space viewed through a video camera), or an animated but non-existent virtual world synthesized by a computer(for instance, the animated "world" created in a video game)(2).

From these two citations we can now conclude that immersion refers to one's ability to adapt to a new environment and rid one's mind of that which hinders the ability to move and manipulate the environment at hand, in short immersion is the act of becoming as comfortable in a new environment as one is in its natural environment. Presence then refers to the act of disconnecting from one's immediate surroundings and become present in the virtual environment instead disregarding the physical feeling that surrounds one's self in the world that surrounds him.

2.2.2. Colour Psychology Principles

When creating an immersive virtual experience one of the many factors to consider are light and colour, it has been confirmed that regarding colour psychology principles enhances the sense of immersion in computer games and consequently affects the performance and behaviours of players(4).

This virtual environment will be crafted with the knowledge that different colours effect one's perception of the experience differently by affecting the user's behaviour, this virtual environment will although only take a regular person eyesight in consideration as there are several variations of colour blindness that would affect the results of colour psychology principles.

In this project the following colour table will be used to pick out different colour gradients depending on the mood wished to be imposed on the user.

Index	Completeness	Psychological Association
1		Happiness, Peacefulness, Authority, and Persistence
2		Powerfulness, Warmness
3		energetic, radiant, Anti-Depressant
4		Aggressive with a threat probability
5		Truthfulness and Responsibility
6		Cool and Entertaining
7		Warmness also Logical
8		The sense of safety and quietness, Not tiring and boring
9		Inducing an unusual environment to intensify understanding of special moments
10		Quietness and deepness
11		Inducing an unusual environment to intensify understanding of special moments
12		Friendly and casual
13		Inducing an unusual environment to intensify understanding of special moments
14		Quietness and deepness and not annoying
15		Inducing an unusual environment to intensify understanding of special moments
16		Quietness and deepness and a little exciting

Figure 2.1 Colour Psychology Theory

2.2.3. Experiences of Body Transfer while in VR

To achieve a greater sense of presence one's virtual body must feel like his original body.

Altering the normal association between touch and its visual correlate can result in the illusory perception of a fake limb as part of our own body, the illusion has also been demonstrated using visuomotor correlation between the movements of a hidden real hand and a seen fake hand. This type of paradigm has been used with respect to the whole body generating out-of-the-body and body substitution illusions(5).

This is a main factor of research to achieve the desired outcome of this experience, how to achieve this illusion of transferred bodies is the hard part, by wearing a HMD and having sight and hearing be taken over by what is displayed and head in the virtual environment will greatly help achieve this illusion, but for a whole-body substitution illusion more than just sight and hearing must be altered.

It is an aim of the project scope to produce a secure treadmill that can be used by the user while wearing an HMD this would increase one's sense of presence by allowing movement to be correlated between virtual and real body this would then help achieve a stronger sense of body substitution illusion and induce a stronger sense of presence.

2.2.4. Visual Saliency

The human visual system (HVS) constantly uses a visual attention mechanism, which allows to rapidly analyse complex scenes by directing its attention to the prominent parts of a scene. This mechanism repeatedly shifts focal attention to the salient feature points, and the details transferred from the real world to the HVS change with these shifts. Such a process lessens the cognitive burden while perceiving the surrounding world(6).

To properly implement this virtual environment one must know how to properly direct the users attention to specific areas of the world, different features must stand out to the user so that a spark of interest can formed in the users mind, according to this research BMS'(Boolean Map Saliency) predictions scored the best in most metrics for the outdoor scene in 3D viewing(6). BMS operates on a basis of contrast and difference between the target and the environment, and by applying the these concepts to the virtual environment the users attention can be directed to different sections of the world, using colour psychology theory principles in conjunction with the research performed on visual saliency a level of attraction to the eye can be achieved by using contrasting colour pallets between the world and objects intended to call the user attention, that will be the main method used on

creatures that populate the world, they will be created with a colour pallet that heavily contrasts the world around them, this method will heavily differ from the real world where creatures tend to blend in to their surroundings and be harder to identify.

Although contrast colour pallets will be used to attract user attention, it is also important to note that a balance must be struck so that the user does not get overwhelmed by the number of objects in the scene calling its attention.

2.3. Technologies researched

A number of different technologies had to be researched in order to get the best idea of what was best for this program, all the different technologies can be broken down into different sections of research, these areas are 1)Game Engine to be used, what game engine would be best suited for this program taking in consideration previous experience in said engines, best suitable engine for a VR project, price and resources, together with all of these points another major driving factors that will decide on what game engine will be used is the computer language adopted by the engine in question. 2)HMD, what HMD to use is also a big factor of research, from price, to availability, which HMD is best for development and which one can suit this project better. 3)3D graphics software, another big section of research is dedicated to figuring out which 3D software tool is best suited for this project, taking in consideration price and span of the tool set and ease of use as time is limited and 3D graphical design of different models can be very time consuming. 4) Audio editing tools, which audio editing tool is best for the project can also be broken down into price and resources, as sound is a vital area of immersion a good audio tool is necessary for the optimal completion of this program.

2.3.1. Game Engine

In this section two game engines will be compared to achieve an answer on which one to use to build this project, these two game engines were chosen due to the fact they are free, have plenty of resources and the fact that these two engines are the most prevalent commercial engines in the market at this time. The two engines to be compared are Unity and Unreal engine, and they will be compared in 1)Programming language adopted by the engine, the programming language used by the specific engine will be a very big factor as they both utilize different languages and both languages are better suited for different types of development, not to mention familiarity with languages. 2)Resources, both engines come with a variety of resources and tools attached to them, and all of these must be weighed against each other to reach a beneficial conclusion.

2.3.1.1. Programming language

In terms of programming language, both engines use two different languages, C++ and C# (c sharp) both these languages are great for game building, C++ has better control hardware on the PC or server but C# allows the game to be produced more quickly as it's more beginner friendly and less prone to errors due to its syntax(7).

Overall, the decision of which language to choose comes down to time and experience, C++ allowing for faster and more polished projects but also being higher in complexity and more time consuming and in contrast C# being more beginner friendly and allowing for projects to be built faster and still to a good standard of polishing.

The performance differences between these two languages are also minimal and can be negligible (7).

2.3.1.2. Resources

A big deciding factor between Unity and Unreal engine will be the tools and resources available in the field of VR.

Unity holds an incredible 60% of AR/VR content in the market(8), this would indicate that more online resources are available for Unity in the field of VR as more content is being developed for it, Unity also provides a greater amount of Assets variety than Unreal Engine, currently holding five times more assets than Unreal Engine, although Unreal Engine tends to have more high quality assets previously used in AAA titles(8).

Unity also comes up ahead in the variety and quantity of online courses available for the engine, Udemy education platform (online course platform) currently holds around 2,200 courses for Unreal Engine and 5,500 courses for Unity(8), this shows the difference in resources available between the two engines, this of course is not counting the resources found in the engines forums, where Unity again comes up with more resources by a big margin where on scripting it holds 128,000+ threads compared to Unreal Engine 12,000 threads, and 6,100+ threads in the field of VR development compared to Unreal Engines 4,600 threads(8).

2.3.2. HMD

HMD are a vital part on VR development, and this project will be able to run in any of the most recent HMD in the market since both Engines that are being considered will allow this, although many HMD are available in the market the decision to use the Oculus Quest HDM was an easy one to make as it was supplied by faculty members of TUD for the development of this project, and given that HDM are an expensive investment the ability to use one that was provided free of cost will be taken kind heartedly.

2.3.3. 3D Modelling Software

To create assets for this project a 3D modelling software is a necessity, and to identify the most complete and cheapest 3D modelling tool will be the aim of this section, as 3D models can also be purchased or obtained freely from both Engines asset store, so the search for the perfect modelling tool does not need to be extensive as this will only be a small portion of work to be done for the completion of this project.

The aim of this section was to look at several 3D modelling tools and compare them to find the optimal software to use, although throughout the research something of notice is that most complete modelling software's are offered at a very expensive rate, this eliminates most of the possibilities and software's such as Maya, Houdini and Modo will not be looked into further due to the price of such software's, furthermore not many 3D modelling software's come equipped with modelling tools and animation tools, and this is a big requirement for this project as most models built for the project will require some sort of animation, so many other software's like Daz Studio, Hexagon and Wings 3D which are all free widely used 3D modelling software's will not be considered either.

For the requirements of this project a software that is all inclusive of both modelling and animation would be preferred as it would be a time saver to learn how to use only one software that is all inclusive instead of suit of software's that are used to achieve the same goal, this is why in this section the only software that will be expanded further is Blender, Blender is a free modelling and animation software that is widely used and in turn several online resources for it can be found.

2.3.3.1. Blender

Blender is a free, open source software that comes equipped with plenty of useful modelling, rigging, animation, simulation, rendering, compositing and motion tracking tools (9), Blender is also one of the

most widely used free modelling software on the market which means that there is no shortage of resources for beginners, due to Blenders popularity an array of free models can be found online ready to be used, this comes at a great advantage as it can be an immense time saver and some of the models obtained for free would be of better quality than anything produced by a beginner in a tight schedule. Blender also offers the possibility to animate and rig(add points of movement) to models which will allow for a more immersive world by having models move in the desired ways without the use of any other external software.

2.3.4. Audio Editing Software

For the aims of this project, together with how it looks and feels audio is going to be a big section, to induce a sense of immersion and presence to the user, the sound of the world that surrounds them must match and enhance the visual aspects of this experience, and for that a good beginner audio editing software that allows the project to excel to the next level as an immersive experience will be a must, to properly identify the best suited audio editor, first an understanding of what is needed for such is needed, this project mostly requires a tool that can facilitate the introduction of world sounds, some of these might include birds or water streams, although some simple music tracks might be produced with the intent of being used inside the game world this idea is not essential and will only come to fruition in later stages of the project.

In this section a research of different free audio editing software's was carried out, these software's include 1) Audacity, an audio editor that is free and supports a different array of plugins, 2) Ocenaudio, another free audio editor which is similar to audacity in terms of potential but more user friendly, and 3) Audiotool, a free online audio editor with great mixing facilities.

One of these sound editors will be used in conjunction with 4) Steam Audio, a software created to implement music into projects like this one, with a focus on spatial audio.

For the purpose of this project all audio tools mentioned carry out very similar functions with little change between them, they all allow for multi-track audio edition and provide live audio recording, so for sake of simplicity Audacity will most likely be used, as it is the longest running audio editor with 20 years of deployment and the largest resource base due to its longevity.

2.3.4.1. Steam Audio

Steam Audio provides, fully-featured spatial audio attuned to the physical attributes of the project geometry, it has built in plugins for all major game engines and platforms and provides a range of fully automatic physical sound simulation to unique handcrafted tuning(10).

Steam Audio is the perfect audio software as it was built with an emphasis on VR immersion by mimicking how sound would reflect in the real world by recreating how sound interacts with the virtual environment(10) .

2.4. Design Methodologies

In this section a look at different methodologies will be presented, from the ever ending list of different design methodologies only one methodology type suits the described requirements and that is Agile methodology, as a look at other methodologies would show that they are better suited for bigger teams with a bigger time allocation to complete the project, although the Agile methodology was picked further research must be conducted in the several different Agile methodologies philosophies to use.

First to describe the concepts of Agile methodologies group so that a better understanding of the following Agile methodologies can be derived, and a more suitable decision can be made.

The Agile principles are: to satisfy the customer through early and continuous delivery of valuable software; welcome changing requirements, even late in development; deliver working software frequently; business people and developers must work together daily throughout the project; build projects around motivated individuals; the most efficient and effective method of conveying information to and withing a development team is face-to-face conversation; working software is the primary measure of progress; Agile processes promote sustainable development and developers should be able to maintain a constant pace indefinitely; continuous attention to technical excellence and good design enhances agility; simplicity is essential; the best architectures, requirements, and designs emerge from self-organizing teams; at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly(11).

These twelve points are the core principles of the Agile methodology and despite the fact that not all of these points applying directly to this project as this project consists of individual work, they still lay the foundation for what this project aims to be.

A selection of Agile methodologies that upon research seemed to best fit the projects criteria will be presented promptly, and an insight on how to make these methodologies work for a single developer will be given as most if not all Agile methodologies were created with a team essence to them.

The methodologies to be presented are, 1) Extreme Programming(XP), 2) Feature Driven Development(FDD) and 3) Crystal.

2.4.1. Extreme Programming

XP was developed and adapted by small teams whose project scope is vague and ever changing, XP is implemented in life cycles, each life cycle has twelve stages and after all the stages are done the life cycle restarts, this process is to accommodate for change in between life cycles, allowing for new ideas and systems to be implemented during the development stage, some of the stages of a life cycle include: planning, to determine the scope of the next life cycle; refactoring, to change or improve previously implemented systems without changing its behaviour; and 40-hour weeks, to work as much as possible in the program, although downsides to working long hours for an extended period of time can come with downsides(12) .

XP makes two sets of promises, it promises that programmers will be able to work on things that really matter, every day, they will be able to do everything in their power to make their system successful, they will make decisions that they can make best and will not make decisions that they are not qualified to make. To the customers and managers, it promises that they will get the most possible value out of every programming week, every few weeks they will be able to see concrete progress on goals they care about and they will be able to change the direction of the project in the middle of development without incurring exorbitant costs(13).

With this information XP seems like an optimal methodology and it can be easily adaptable to a single developer project, it focusses on daily work and progression by stages which will be incredibly useful for testing and improving the project with user feedback at hand.

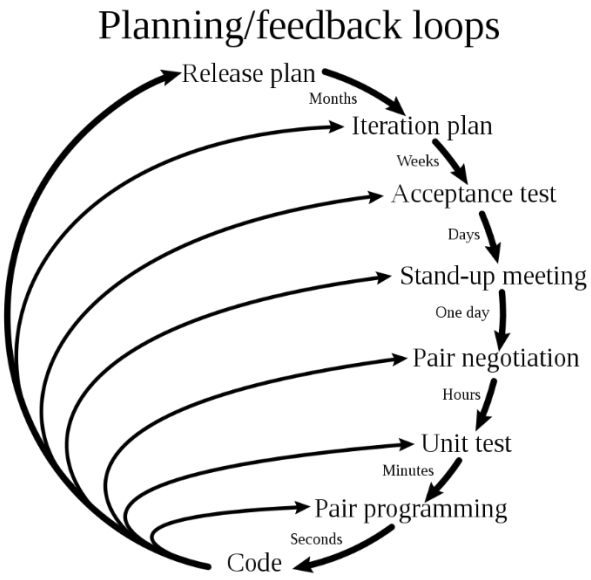

Figure 2.2 XP Life Cycle (14)

2.4.2. Feature Driven Development

FDD as the name entails focuses on the development of features, the process of FDD is to develop an overall model of the program, these can be broken down into domain area models, after the model is complete and an understanding of what must be done is achieved a feature list must be built, this feature list for the model, a plan is then devised by feature, based on feature dependencies and complexity, this is then followed by design by feature which includes grouping different features that may share classes to be developed together and designing each feature group independently, and finally build by feature where classes are built to support the feature design implemented previously, the code is then unit tested and then built(14).

FDD can be easily adapted to a one-person project, and the feature driven approach ties in particularly well with this project as Game Engines work on a script base that can then be implemented into "game objects" this methodology would allow for a clean implementation of features by order of importance allowing for different stages of the project to be deployed for user testing as more features become complete and more can be experienced.

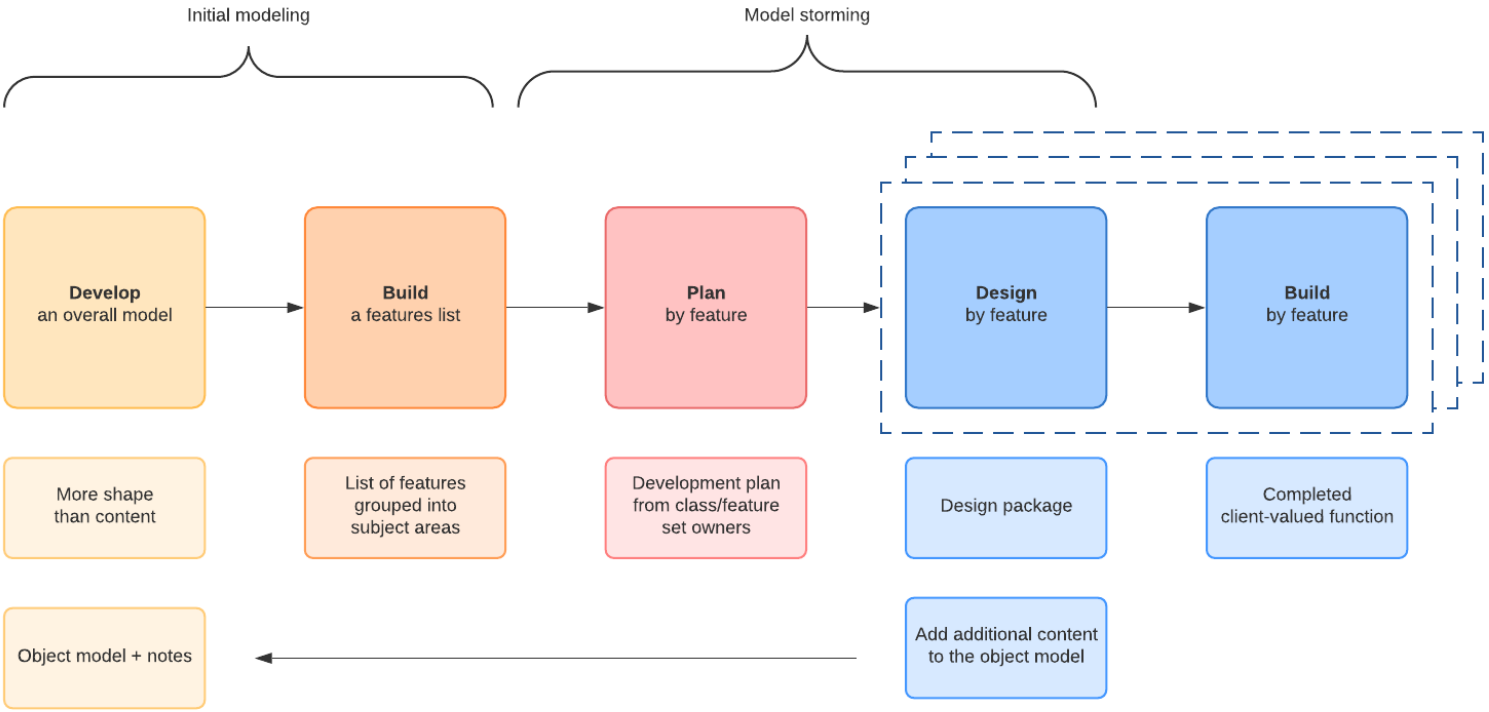

Figure 2.3 FDD Life Cycle (16)

2.4.3. Crystal Methodology

Originated by Alistair Cockburn, Crystal is a family of methodologies (Clear, Yellow, Orange, and Red) and is based on project size (where the number of people involved are about 1-6, 1-20, 1-40, 1-100) and criticality (where defects could cause loss of comfort, discretionary money, essential money, or life). As the team size grows, Crystal implementations change to add more formality to the structure and management of the project(15).

For the scope of this project Crystal Clear could very possibly be the methodology required, but due to the nature of the Crystal methodology family and the fact that every project differs by nature, and the fact that Crystal clear is not fully defined and the first step of this methodology is to uncover the development process strengths and weaknesses and to fit the recommendations of Crystal Clear around them to capitalize on the strong points and cover the weaknesses(16), it leaves the feeling that more experience in the medium of development is needed before attempting to use such a minimal methodology.

Crystal Clear shares some characteristics with XP but it generally less demanding, Crystal focuses on frequently delivery, reflective improvement, osmotic communication, personal safety, focus, easy access to expert users, and technical environment with automated tests, configuration management and frequent integration(16).

In regard to this project all of the Crystal main properties can be achieved, although to implement Crystal the way it was intended a high level of implementation experience is needed as this method aims to be as minimal as possible.

2.5. Testing Methodologies

Testing is a vital part of any development cycle, especially for a project like this one where not only is code testing important to make sure every system is running the way it is supposed to be running but it also allows for different users to test and experiment with the project in order to obtain different points of view and input on how to make the project a better experience.

In this subchapter different testing methodologies will be looked into in order to identify the one best suited for the project needs. Some of the testing methods used will require users to test and give feedback, these testing methods are slightly unrealistic given the project requirements and the current world scenario and test group sizes will be greatly reduced from what originally intended, but the best effort will be made to ensure that this project will receive an appropriate amount of user testing.

The testing methods to be used will likely consists of 1)Unit Testing, 2)Integration testing, as functional testing methods able to be deployed by the developer and 3) Performance testing can also be deployed by the developer, user tests will include 4)System testing, and 5)User Acceptance testing.

Given the current project scope, White Box testing will be used where possible for all testing implementations as the internal structure and code of the project is known and a clear idea of how it should behave and the design of the project is well established, this will only not be the case when the user is the one testing the project.

2.5.1. Unit Testing

A unit test is a piece of code written by a developer that exercises a very small, specific area of functionality of the code being tested. Unit tests are performed to prove that a piece of code does what the developer thinks it should do(17).

In the interest of this project, Unit testing can be used to test individual section of the code, for example to test that one specific animal behaviour is working as intended before applying it to the animal in question.

2.5.2. Integration Testing

Integration testing will be the next group after Unit testing, where software modules are integrated logically and tested as a group(18).

There are several ways of implementing integration testing: Big Bang Testing, where all modules are integrated together at once and then tested as a unit; Incremental Testing, where testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application; Stubs and Drivers, these are dummy programs in integration testing used to facilitate the software testing activity, these programs act as a substitute for missing models in the testing, they don't implement the entire programming logic of these systems but simulate interaction with the module being tested; Bottom-up testing, in this method lower level modules are tested first, these low level models are then used to facilitate the testing of the higher level ones (18).

From all of these methods the only one that does not align itself with the project is Bottom-up as we would want to work on the more important systems first to account for the possibility of not reaching a stage of completion with the project due to time constraints.

2.5.3. Performance Testing

Performance testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under particular workload. The main purpose of performance testing is to identify and eliminate the perfo rmance bottlenecks in the software application(19).

This testing method can be achieved by loading scenes with an absurd amount of game objects and running unrealistic scenarios to test the boundaries of this project, and both game engines previously mentioned also provide great performance profilers.

2.5.4. System Testing

System testing is a level of testing that validates the complete and fully integration software product(20).

The aim of system testing would be for the user to test the working system so that user experience can be recorded, and input can be gathered, this will also allow for the user to express if they feel like the world is behaving in an expected way as this would be required for an immersive feel.

2.5.5. User Acceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the user to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing are done(21).

In this projects case UAT would be used as a final testing ground, a chance for the user to test the program one last time and give their feedback, this would allow for minor changes like colour and small behaviours to be adjusted before the final project is submitted.

2.6. Conclusions

In this chapter, the literature to the immersive virtual environment system was presented. First the background research was presented where immersion and presence were explained, and different methods of achieving the greatest possible feelings of immersion and presence were researched, these methods consisted of Colour Psychology Principles which showed how colour has a serious impact on the users mood and the way the user approached the world around them, experiences of body transfer while in VR were also researched to reach an understanding of how to best induce a feeling of presence in the user and the idea of using a VR treadmill was added to the project scope to allow the user to experience the environment at his own pace and so being closer to reach a body transfer experience while exploring this virtual environment, Visual Saliency where research was conducted on how to shift the users attention to different areas of the world, and how using contrast could be used to bring the user attention towards specific areas of the world.

Promptly after technologies researched were presented, going through the different Game Engines options, taking in consideration the languages the Engines adopted and the available resources for both engines, a discussion on which HDM to use is also present in this chapter together with a discussion on modelling and audio software.

Finally, a research of design methodologies was given together with an overview of testing methodologies so to facilitate a future choice on what methodologies to use.

This chapter paves the way into the next chapter, Design, in this chapter all the information required to make an educated decision was gathered, moving into the next chapter it is now possible to decide between Unity and Unreal Engine, taking in consideration the knowledge gained in both engines, and in the languages, they deploy, although a pretty conclusive decision on modelling and audio software was already derived in conjunction with the HMD to be used.

The testing methods to be used will be discussed in chapter 5 with regards to the research provided in this chapter.

3. Design

3.1 Introduction

This chapter will be one of the most important chapters of this document, as it will explain the design decisions for the project and explain what systems are planned and how the completion of these systems will be achieved.

In this chapter a look into the requirements need from a design methodology for the benefit of this project will be seen, together the final design methodology to be used will be presented taking in consideration the research previously performed in chapter 2.

Promptly after a short overview of the system as it currently stands will be given and different main system families will be talked about and their design simply explained. As well as a predicted project life cycle that shall be created with the design methodology picked in mind so that it aligns with that methodology life cycle.

A variety of Use Case diagrams will be presented to further demonstrate the design of the project, these Use Cases will present the design interactions between the system and the user from a hardware level to a software level, this section will be expanded as necessary given the creative nature of the project and the choice for an Agile methodology.

Finally, an overview that reviews all the previously mentioned subheadings will be given to summarise and further cement how these design choices will be used together, and to expand on the likely future of this section as the project progresses through its life cycle, this methodology also seemed to have a greater team size than one in mind when it was developed, and this seems to be the way with most other methodologies.

3.2. Software Methodology

3.2.1. Methodology requirements

Given the fundamentally creative nature of this project the choice of an Agile methodology felt the most appropriate as previously stated in this document, furthermore, this project would greatly benefit from an organized but lenient methodology.

This project will change and evolve through its development cycle, and given that other than its creative nature, the main focus of the project is to deliver an immersive and visually interesting experience that will induce a feeling of presence and fascination into the user, a methodology that allows for several user testing opportunities would greatly benefit the overall aim of the project.

Time is also a deciding factor when choosing a methodology, to balance the development of this project in the limited time allocated given that its time will be shared with a heap of other educational responsibilities, a finished project is not a guarantee, although best efforts will be placed to make sure the completion of this project is reached and careful planning is being executed, one must be prepared for any eventuality, that is why a methodology that allows for the incremental production of the project must be chosen, a methodology that will allow for main systems to be implemented but not at the expense of any other possibly minor systems, as a program that only has a world but nothing to do inside of it would not feel the most immersive, this is why iterative cycles of development that allow for different stages of the application to be developed and tested could prove very beneficial.

Lastly as the main reason for Agile methodology being picked for the development of this project, the ability to change the scope of the project is essential, again due to the creative nature of the project, no one system is set in stone and all systems are bound to change or new systems are bound to be implemented as required, this being from a new creative direction or by user input, a close look at the projects design must be kept and changes to this will be many and consistent.

3.2.2. Chosen Methodology

Taking in consideration the methodology requirements just explored and the different Agile methodologies previously researched in chapter 2, a careful decision was made in regard to the chosen methodology to be used, this decision will now be elaborated upon.

From the research on the different Agile methodologies an observation was made, and observation that all the mentioned methodologies have their benefits and their drawbacks, so careful deliberation was given to what methodology to pick, and although XP and FDD are more clearly defined, the research of Crystal was the research that provided the most enlightenment, as the main focus of Crystal is that no one project is the same and the methodology used should be adapted given the scope, criticality and team sizes, this is why for the development of this project the decision to implement a slight variation of the methodologies previously researched.

Initially great consideration was given to FDD as it seemed to align itself splendidly with the project, given that the nature of the project lends itself to be built by features, this is the nature of any project built in a Game Engine, although after further deliberation the thought of fully developing a feature before moving on to the next did not seem to be the correct decision, although the construction of a feature list still held great appeal.

XP was then considered, and overall XP seems like a perfect Agile methodology, and the life cycle of this methodology lends itself perfectly for testing and development alike, although 40hour weeks seemed very impossible due to the many other educational obligations present.

In conclusion, given that different aspects of different methodologies were more relevant for the development of this project, a decision was made to use a slightly modified XP methodology that lends itself more to a single person team, borrowing from FDD. This left the project with a methodology that will deploy a feature list and features will be separated in groups of deployment, then development and testing will be performed for each deployment group, allowing for changes to the design to be implemented after user input is given, also allowing for the most important features to be developed first and less critical features to be implemented and changed throughout the development, this will allow for a project that can be tested without feeling overly incomplete, and given that this project is being developed by a single person and not a team of individuals the most important aspects would be to get a user group to test the application and give much needed feedback.

3.3. Overview of System

In this subsection an overview of the entire project will be presented, a project breakdown will be displayed, system architecture will be briefly touched upon and the overall aims of the project will be discussed, by the end of this subsection a complete understanding of what the project entails and the systems that will be required for the project to be deemed complete, a phase breakdown will also be reviewed so that progress can be monitored and documented throughout the development cycle.

In addition to those, use case diagrams will be designed so that a greater understanding of the project can be derived, these diagrams will explain how the user interacts with the system as well as explain how the system interacts with itself, different ways that the user can use the program will also be explored.

But before these are explored, an overview of the project as it stands is required, and as derived from previous sections, the design methodology and hardware to be used will now be given.

3.3.1. System Design

As previously stated in this chapter, the methodology to be used will be a variation of XP and FDD, changed to fit the requirements of this project, as such the development cycle that was derived will consist of three main phases (Alpha, Beta and Version 1.0), these will now be explained.

This project was broken down into different features as per FDD, and these features will be split between the three phases of the project, each phase will mimic the XP life cycle by deploying testing and development of the specified sections and allowing for review and change of the program throughout.

The first stage, Alpha, will be the foundation of the project, the aim is to develop the main bones of the project so that user testing can be started and a feel for the correct approach can be obtained, Alpha stage will be the stage where most systems will undergo creative change as the foundation is still being solidified.

Beta then will involve the implementation of less critical systems, also with a big emphasis on changing the project based on the input provided by the user testing in the previous stage, and more time can be allocated to the initial systems as needed.

The last stage, Version 1.0, is the deployment ready version, the aim is that by this version all systems are implemented and only polishing is required, together with bulk white box and black box testing as to make sure the project is as complete as possible, and it feels as good as possible to the user.

Furthermore, in chapter 2 the design options for software and hardware to be used were explored and researched, and with regards to that research the following decisions were made. It was decided that Unity and C# will be used as the Game Engine and language to be used in this project, given that Unity is a user-friendly Engine with several online resources to aid in the development of this project, equally C# also seemed the best language as it is powerful and user-friendly, and having Unity use C# natively made this choice of Engine and language vastly easier.

In terms of other software, as previously discussed Blender seemed like the optimal 3D rendering tool, paired with the Unity asset store the search and creation of assets should not be an arduous one. Steam Audio and Audacity the perfect pair for audio edition and implementation, and it is hoped that the use of these two software's together will increase the immersion in this IVE.

As for hardware, like previously stated the Oculus Quest will be the HMD used in development and testing as it was kingly supplied by a TUD faculty member for the purpose of this project.

3.3.2. Project Break-Down

This project will be built using a single-tier architecture, this was an easy decision as the project will be a single user experience with no external need to a database and the program will also be ran on the user's machine.

To achieve the aims of this project there is a few distinct systems that will need to be designed, these can be broken down into, 1) World Generation, 2)World interaction and 3)User interaction.

These systems will be further elaborated bellow.

3.3.2.1. World Generation

World generation will be one of the main systems in this project, given that with no world to be transported to none of the other systems can be implemented, world generation will comprise of terrain generation and then the implementation of procedural terrain generation, this will be achieved using Perling Noise for terrain generation and chunks as the basis of the implementation on procedural terrain generation, these two systems will be the main systems to be implemented in the world generation category.

Furthermore, to complete the world a collection of fauna(animal life) and flora(plant life) that will complete the ecosystem needs to be implemented, this ecosystem also needs to be procedurally generated throughout the world in a coherent way.

These sections will be part of the Alpha stage, as they are the foundations that make up this project.

3.3.2.2. World Interactions

In the Beta stage world interactions will be added, as these are still important and necessary systems for the project to feel complete.

These interactions will be comprised of interactions between flora and fauna and their ecosystem so that the user can observe the natural interactions different world life have between each other, flora and fauna and the user, more important than how the world interacts with itself it's the way the world interacts with the user, this would dictate if fauna is intrigued or fearful of the user, and how fauna reacts to the users presence, this stage will give the world meaning and will make exploring and obtaining knowledge of the different life around this immersive virtual world, for the Beta the aim is to have an implementation of two species of flora and two different species of fauna and then during the Gold stage more can be added to the list so to enrich the world surrounding the user.

3.3.2.3. User Interaction

User interaction will reflect how the user interacts with the world, this will be achieved through the HMD and the controllers that come with it to simulate the users hands, the user will also interact with the world through the VR treadmill that will be constructed for the purpose of this project, these segments although not code related will be produced during the Alpha stage as it's a big part of the immersion and presence feelings that this projects aims to project on to the user.

3.3.3. Project Predicted life cycle

Figure 3.1 Project Predicted Life Cycle

Alpha
Terrain Generation
Procedural Terrain
Fauna and Flora Models
Fauna and Flora procedurally Generated
Design and building of VR treadmill
Fauna and Flora interaction with world
Fauna and Flora interactions between self
Fauna and Flora interactions with user
Beta
Testing of Project
Colour coordination in world
World Visual Patterns shifts implemented
Previous systems improvements and redesign as needed
Version 1.0
System overview and testing
Last corrections and small implementation

3.3.4. Project Plan

The project plan will be displayed as a Gantt chart that will show the time planning that goes into this project, this will also be subject to minor changes, but the aim is to have concise deadlines for major systems so that the project reaches a good stage of completion.

The following Gantt chart was produced using Team Gantt(22).

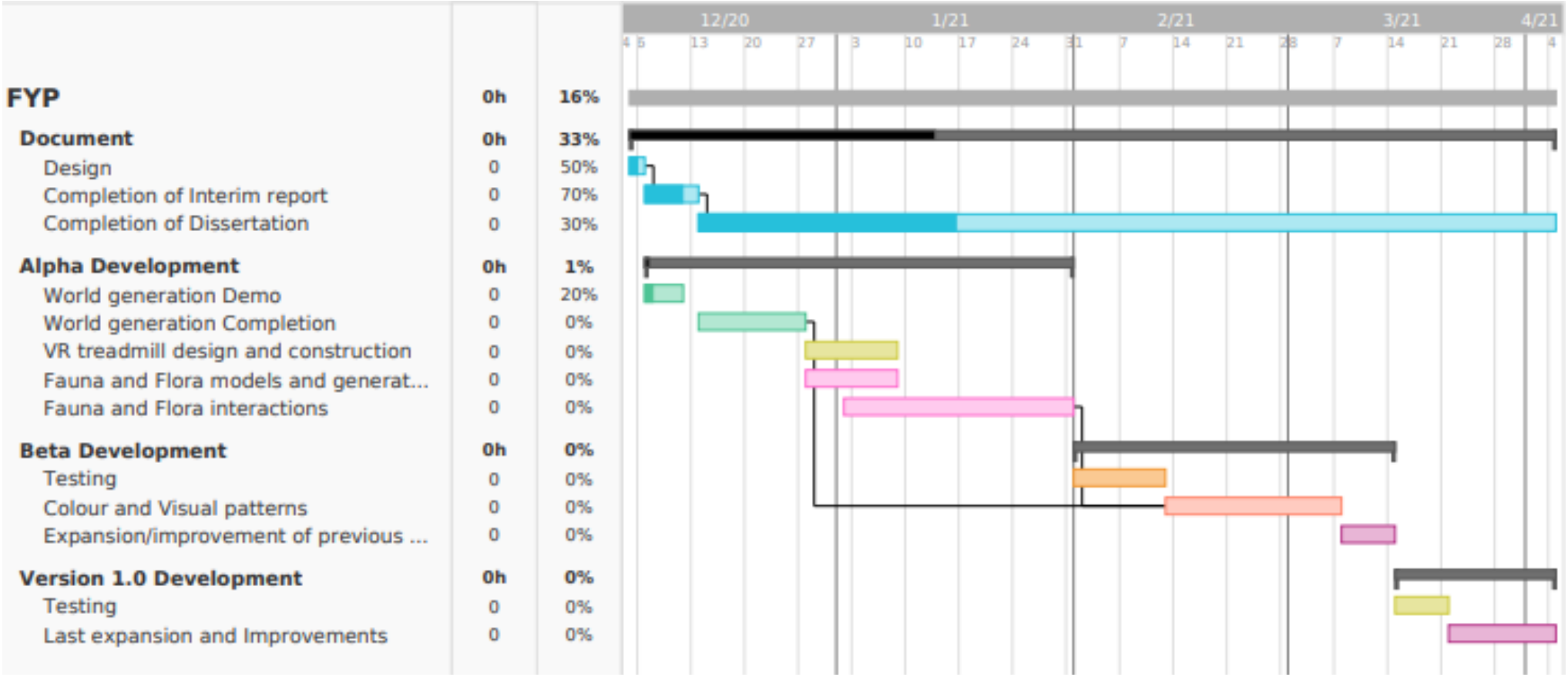

Figure 3.2 Time Allocation Gantt Chart

3.3.5. Use Cases

In this section use case diagrams will be used to exemplify the functionality of this project, 1)firstly a simple use case diagram that depicts the interaction between the user and the system, this will be show how the user can interact with the project at three different levels of immersion and presence, 2)after another use case diagram will show how the fauna and flora can interact with the environment depicting what the user is expected to discover through his exploration of the IVE, 3) finally a depiction of the interaction between the user and the world will be presented in the form a more complex use case diagram, this will show how the user can interact with the world surrounding him, this will include all previous use case diagrams as to show the entire system.

3.3.5.1. User-System Interactions

The user can experience the project in three different immersion and presence levels, the user can experience the world simply through a monitor using a keyboard and mouse, although functionality might be limited and this is not how the project is meant to be experienced, it is completely possible to experience it in this manner, a more immersive way to experience the world would then be via the use of a HMD this would increase immersion and the sense of presence filled by the user, still not the fully intended way to experience this project, but it would still supply a very immersive experience, and finally, the way this project is intended to be experienced is through the use of a VR treadmill, this would exponentially increase immersion and presence by fully immersing the user into the IVE through what is aimed to be a full body transfer experience.

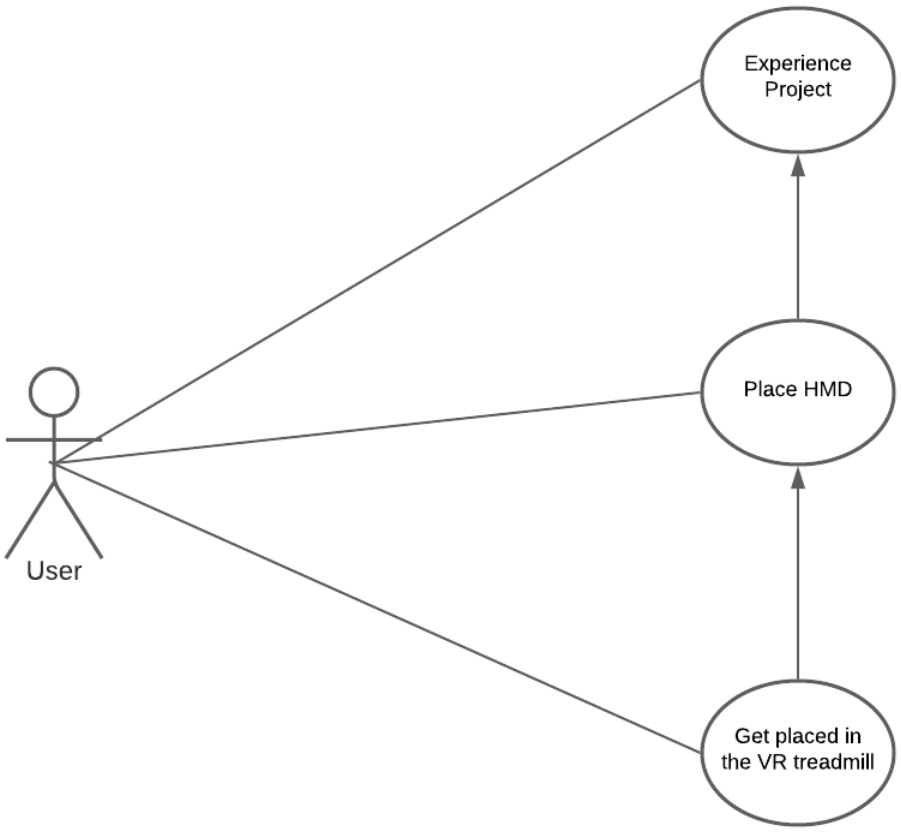

Figure 3.3 User-System Interactions

3.3.5.2. Ecosystem Interactions

The ecosystem must seem alive so that the user can be as immersed as possible into the IVE, to achieve this fauna and flora must have their own patterns and co-exist with each other in a natural way, to achieve this fauna will have depleting status, these status will control hunger, thirst and tiredness, when hungry fauna will look for a source of food, that being other fauna or flora, and by doing this a defensive mechanism must be engaged by the pray were applicable, by having fauna escape predators or intimidate them and having flora engage in any defensive methods available if any, thirst will make fauna flock towards bodies of water, this will make for exploration landmarks for the user as he can deduce that he is more likely to encounter wild life near bodies of water, and finally tiredness will allow the user to approach sleeping fauna with more ease than if they were awake.

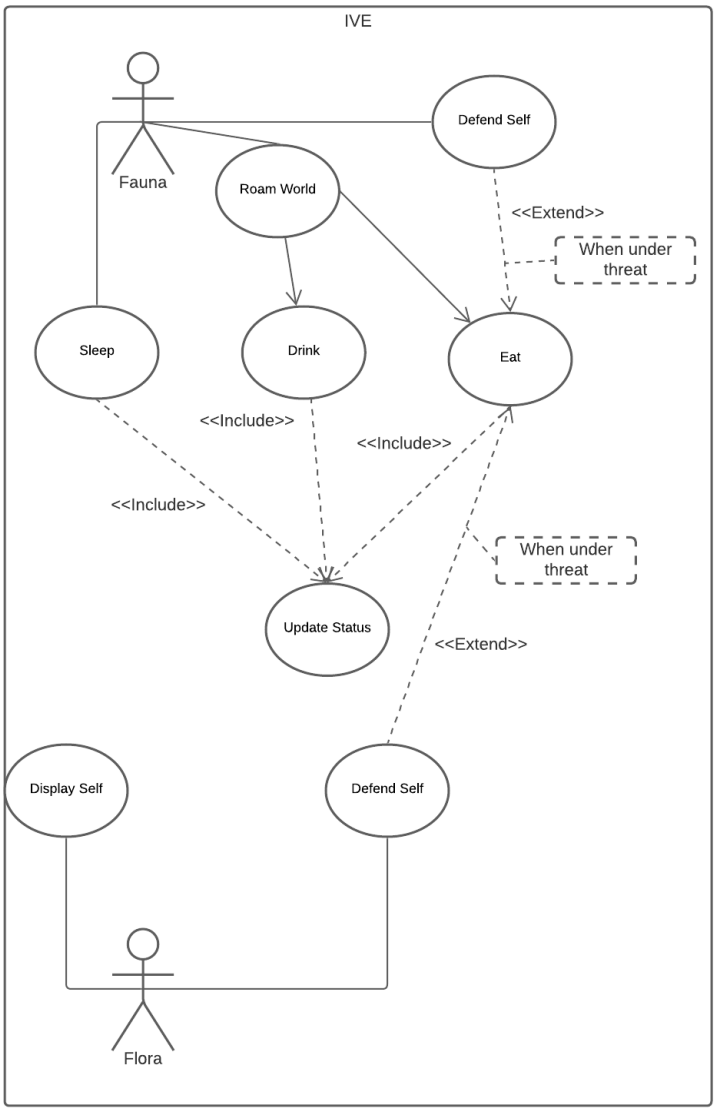

Figure 3.4 IVE-Self Interactions

3.3.5.3. Complete User-IVE Interaction

Once the user is introduced to the IVE the fauna and flora must now also react to the user, although they will not try to harm him as this project revolves around a sense of immersion and presence over a sense of danger and excitement, they will still attempt to defend themselves from the user by intimidation and/or escape, even though some fauna and flora might use defensive methods geared towards distraction and confusion, this is one of the systems that will allow for the implementation of surreal imagery and a sense of faded reality.

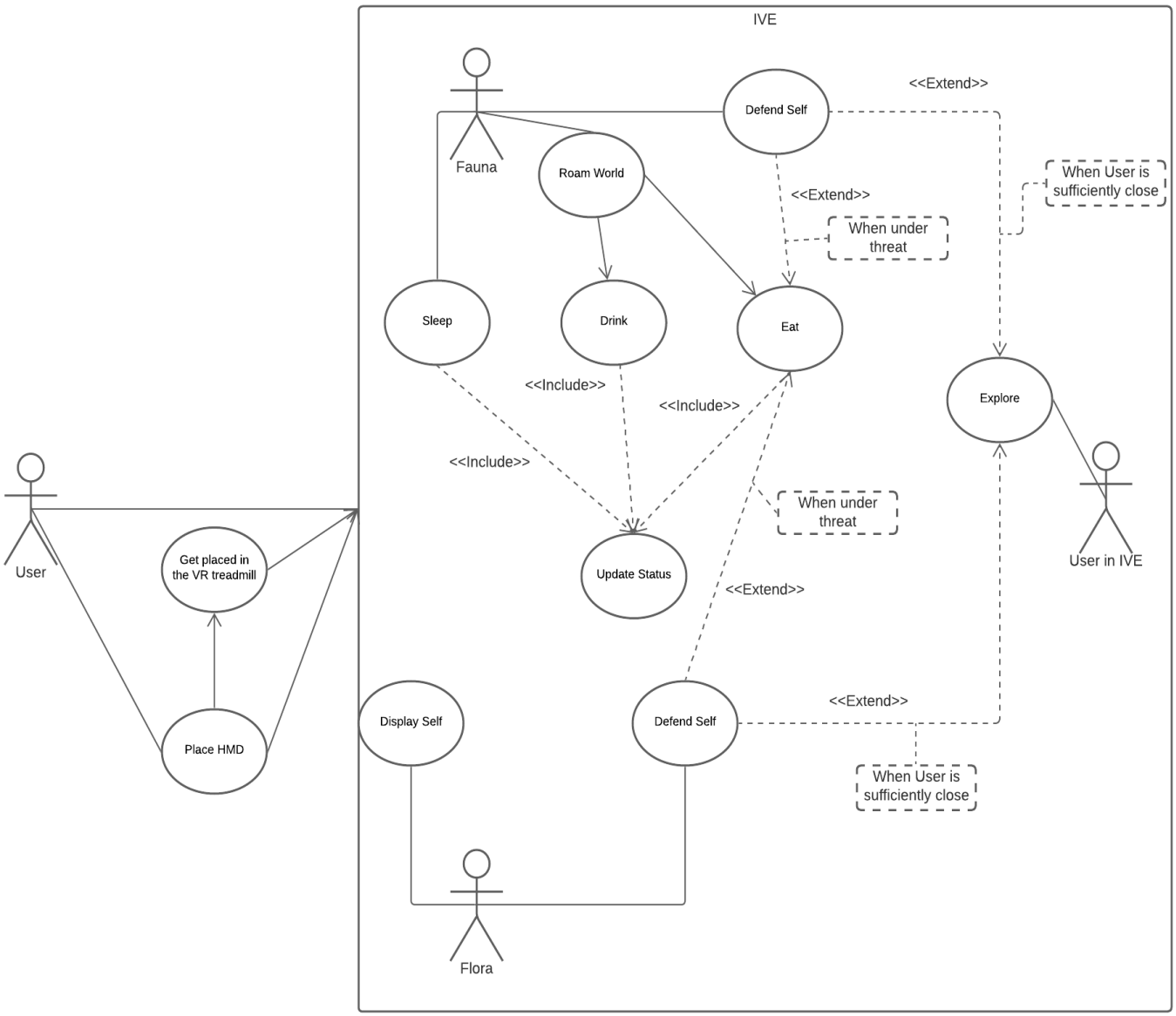

Figure 3.5 User-IVE Interactions

3.4. Conclusions

In this chapter an overview of the design of this project was give, from the methodology that will be used to develop this project, to a project breakdown that explains the main system groups that will compose the project in their simplest form, and a predicted project life cycle given the methodology explanation given prior.

A collection of use case diagrams that further explain the connection between the user and the IVE and the inner connections of the IVE itself were provided.

All of these are bound to change throughout the development of this project as user testing and the creative vision for the project so call for it, all of these changes will be noted and added to this section until the completion of the project.

For now, the project design other than describing software used, hardware, and methodology to be deployed, it also shows what the current plan of development is and what features will be implemented.

As for plan of development it simply states what features will be developed and when, together with projected testing plans, for this a feature list was devised based on what the world is planned to look like, at the moment there is only plans to develop four main fauna and flora entities, these will of course increase as time allows, but a preference to having these four foundation entities developed will be taken. These entities consist of one prey and one predator fauna, and one defensive and one offensive flora, these specific categories of fauna and flora were picked as they once complete will become the foundation for any other fauna and flora to be implemented. More design on these entities will be added to the document as their creative development continues.

4. Development

4.1. Introduction

This chapter will present a basic introduction to Unity and some of its most fundamental components, this will be important for any reader not experienced with Unity as the subsequent sections will use a lot of the terminology explained, this will be backed up by screenshots from within the Unity editor.

A feature overview will also be presented to give a better understanding of how scripts work and how they are used inside the Unity editor in combination with different Unity components, not all scripts will be covered as many follow the same conventions, this will also be supported by screenshots of said components, scripts, and setup present within.

Promptly after a review of some of the main sections of code will be presented and explained, these sections will be presented as code snippets and will be supported by a written explanation.

4.2. Overview

In this subsection an overview of the project completion as of submission date for this document will be given. A fast overview of Unity will be presented such as to aid in the explanation of implementation of features that will be explored in the subsequent subsection. A description of features that are implemented into the project and how they interact with each other will also be present, backed by snippets from the Unity engine to depict how said features are attached to the project via the engine, this subsection serves as an introduction to the subsequent subsection that will explore major features individually to a greater extent, requiring the understanding of project layout and structure given by this subsection.

4.2.1. Unity

This subsection will be a fast explanation of key features in Unity, the purpose of this subsection is to provide a basic understanding of Unity so that subsequent sections can be interpreted efficiently. Given a basic to moderate level of Unity the reader may wish to skip this subsection as it is intended as a ground level introduction into the core principles of Unity.

This subsection will provide explanation of GameObjects, Prefabs, and various Components.

4.2.1.1. GameObjects

GameObjects are the base class for every Unity entity used to create the scene, all scripts must be attached to GameObjects in order to be loaded into the current scene (the scene is the displayable layer of a Unity project to the user, it can be both individual parts of the experience, the full experience or even the main menu).

Figure 4.1 Visual Representation of a GameObject

4.2.1.2. Components

Components give functionality to GameObjects, by themselves GameObjects are empty vessels and have no effect on the scene, by adding components we add functionality to the GameObjects, components can be described as Scripts, these being Unity Engine Scripts like the Transform, Rigidbody and collider components or our own written scripts that are produced to give extra tailored functionality not provided by the Unity Engine. All components must be explored and understood, as often to write our own scripts we must understand how to apply and manipulate different Engine components to achieve the desired results.

4.2.1.2.1. Transform

Every object that is placed inside the scene will have a transform, the transform holds the object position, rotation, and scale.

The transform of an object must be accessed if at any point in the program it is required to move, rescale, or rotate said object.

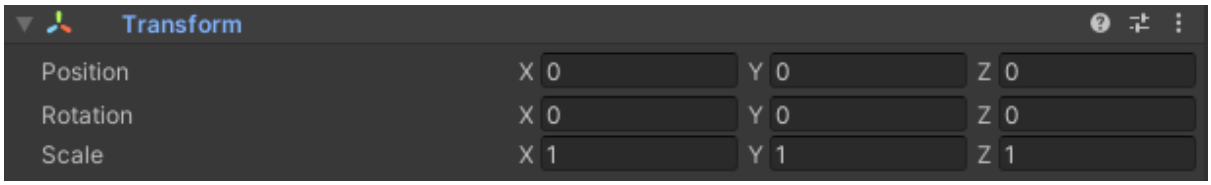

Figure 4.2 Visual Representation of the Transform Component

4.2.1.2.2. Rigidbody

The Rigidbody component actively works as a bridge between the GameObject and the Unity Physics Engine, by applying the Rigidbody component to a GameObject we actively place it under the Physics engine, this will allow the GameObject to be affected by gravity, obtain mass, and drag values, allows for rotation and position constraints to be placed, and it also enables the Engine to calculate collisions with the GameObject. There are also several applications of the Rigidbody component when creating personal scripts and these will be explored more in subsequent sections.

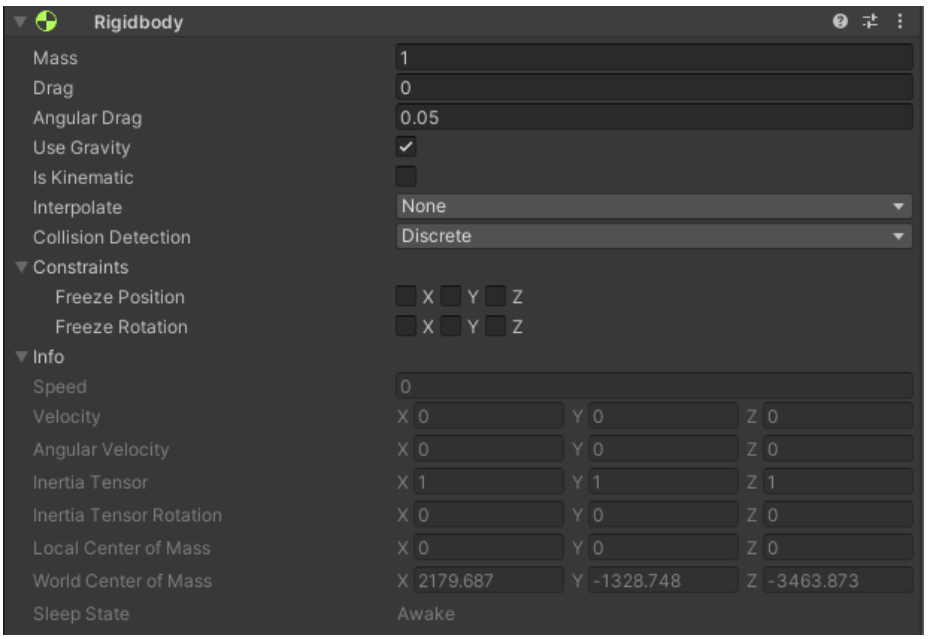

Figure 4.3 Visual Representation of the Rigidbody Component

4.2.1.2.3. Colliders

Colliders are an integral part of Unity coding, by attaching a collider to a GameObject that has a Rigidbody it becomes possible to detect different stages of collision, such as when the collision happens, when the collision is in progress and when the collision ends, this allows for scripts to be made around collisions, examples would include any proximity-based events, as these are most performance friendly when programmed around a collision area. Colliders also have set shapes which are the most performance friendly, or they can be set to contour a mesh although the added triangles in the collider can hinder performance in this case.

Colliders have two states, Trigger and not Trigger, a not Trigger collider will act upon other colliders that come in contact with it, depending on mass and acceleration of both colliders, the GameObjects associated will behave accordingly to expected physical outcomes, meanwhile Trigger colliders don't have a physical impact on the object that it collides against, it will instead call a OnTrigger() function if one is present in a script that is a component to the GameObject, note that this will only be the case when a Trigger collider collides with a not Trigger collider. This notion will be further explored in subsequent sections.

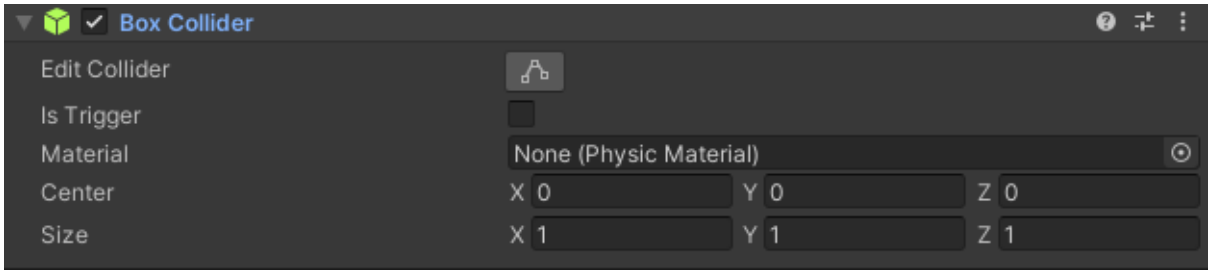

Figure 4.4 Visual Representation of the Collider Component

4.2.1.3 Prefabs

GameObjects must be assembled, once created all the different components must be added to the GameObject for it to serve a purpose.

Prefabs are GameObjects that have been assembled previously and then saved in the project as a Prefab, these can then be called by other scripts and will be instantiated with all the components it was saved with.

Figure 4.5 Visual Representation of a Prefab

4.2.2. Feature Overview

In this section an overview of features and what scripts are used to achieve the desired outcome will be presented, a fast explanation of where and how the different scripts are attached and the reasoning behind it will also be explained, this will be broken down into two separate sections, terrain, and fauna/flora.

Not all scripts will be explored as many of them fallow similar processes, and a deeper view of larger scripts will be explored in the subsequent section.

4.2.2.1. Terrain

The terrain was the most time-consuming part of this project, the initial plan was to use the Unity Terrain module in combination with Perlin noise to create the terrain for the project, although after some time of testing and experimentation a conclusion was reached.

The Unity Terrain module was developed with stationary scenes in mind, and the code for it was protected as it was part of the unity engine and not sourced externally, this meant that accessing data components needed to adjust sizes for different hight points would be extremely time consuming, the lack of information available was also a detouring factor, the performance of the project when using the Unity engine terrain module with all its tools was also subpar.

This now meant that the only solution was to code a terrain generator from the ground up, this was above the expected level of difficulty, luckily an in-depth guide on terrain generation for unity was found(23), there was but one problem, this tutorial was five years old, and given Unity fast growth sections of the tutorial were deprecated, this was a hurdle, so over the span of several weeks the new terrain generation scripts were built using the structural design and notions of this tutorial, and with added research on coordinate maths and the help of other terrain generation tutorials (24) a working prototype was developed.

In the terrain generation chunks were also used to improve performance .

The terrain generation consists of four steps, the first of these steps consists of getting the Perlin noise data, once this data is obtained a mesh is created and hights are set for each vertex using a scale of 0 to 1 that represent a gradient scale of white to black in the Perlin noise, different meshes are created with different vertex densities to create the chunks, and using player position the meshes further away will have less vertices and the meshes closer to the player will have more, this allows for a greater rendering distance as the meshes with low vertex counts will be too far to be noticed, and as the player approaches, the meshes are swapped into meshes with a higher level of detail, the last step then simply consists of giving these meshes a texture shader so that different textures can be applied for different hight levels.

To make the terrain functional there is a collection of scripts and data objects that need to be created and set up, first an empty object is created and the name MapGenerator is given to it, after that we attached our two main scripts to it and attach any component required into the scripts.

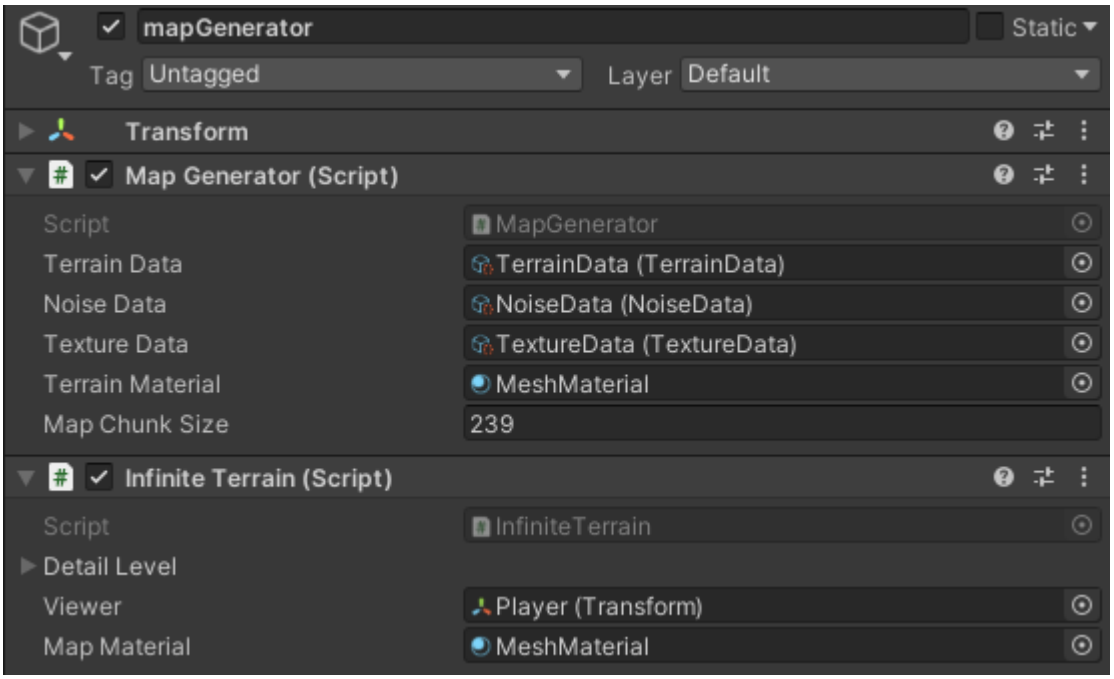

Figure 4.6 Unity Terrain Setup

4.2.2.2. Fauna and Flora

The main goal regarding the Fauna and Flora was to create scripts that were high quality and promoted reusability.

The main sections of inserting animals and plants into the scene are both the placement of the object and the interactions the object presents to the user and other objects, unfortunately due to time constraints the interactions list is very limited, and more work is required, but a solid foundation is presented to add or remove objects from the world, and the main structure that comprises interactions is also developed which promotes fast growth over time.

The positioning of objects in the scene was an interesting challenge, the aim is to be able to have different objects spawn around the player and despawn once the player moves too far away as to keep a good frame rate and reduce the number of resources needed to run this experience, this then brings in the question of spawning objects on top of other objects and in addition to that, the objective is to have different objects spawn at different hights in the world as to introduce variety in the scene.

Originally the main idea was to obtain ground hights from a vertex array produced by the terrain generation script, the size and distance of these points would then be analysed, and objects could be placed at those points based on a set of rules, this proved to be a gross complication of such matter, other than a severe lack in performance, this method was also very time consuming to execute as the

vertices would need to be parsed and there are simply too many of them to make a realistically working script. The alternative to this problem was presented while researching and experimenting with other Unity modules, Unity Physics has a class called Raycast, this will cast a ray from a given position and direction, and this will return true or false depending on if a collider is hit, what makes this so useful is the details obtained from the Raycast hit, after a Raycast hits a collider a number of data is stored in the hit variable and this data can be used for the problem at hand.

A much simpler version of what was before attempted was simply to spawn an object X distance above the ground level, a distance that is greater than the highest point in the ground, cast a ray from the bottom of the object and retrieve the distance of the ray at collision, now if we subtract the ray distance from X we get the hight of the ground directly under said object, this then allows for a decision to be made about if we want the object in that position or not.

With a collection of Trigger scripts and some prefab setup we can then locate if any objects are overlapping and remove these from the scene, making sure each object is carrying the correct tag is essential for this section.

Finally, to place all the objects in scene a ObjectPool was used, the code for which will be explored in the next section, a object pool is simply a group of objects that are created as soon as the scene loads, these objects are not enabled and a greater number than needed is created so that throughout the experience whenever a specific game object is needed it is taken from the pool to save on processing power related to creating a new object, this means that objects are taken and placed into the pool instead of being created and deleted, this object pool was designed to be used from the Unity engine allowing for placement of any prefab and the number of objects to be created avoiding hardcoding as a large number of objects are going to be pooled since all objects on scene will be part of this pool.

The scrip for the pool was placed inside the mapGenerator game object as having all scripts that make up the world inside the same empty game object seemed reasonable.

One thing to note when regarding the gameObjects that make up the object pool is that their mesh renderer must be disabled, making them invisible to the user, unless they are placed on the terrain where we then enable their mesh renderer via code, vegetation is also set to be kinematic in nature since there is no real use of gravity for them.

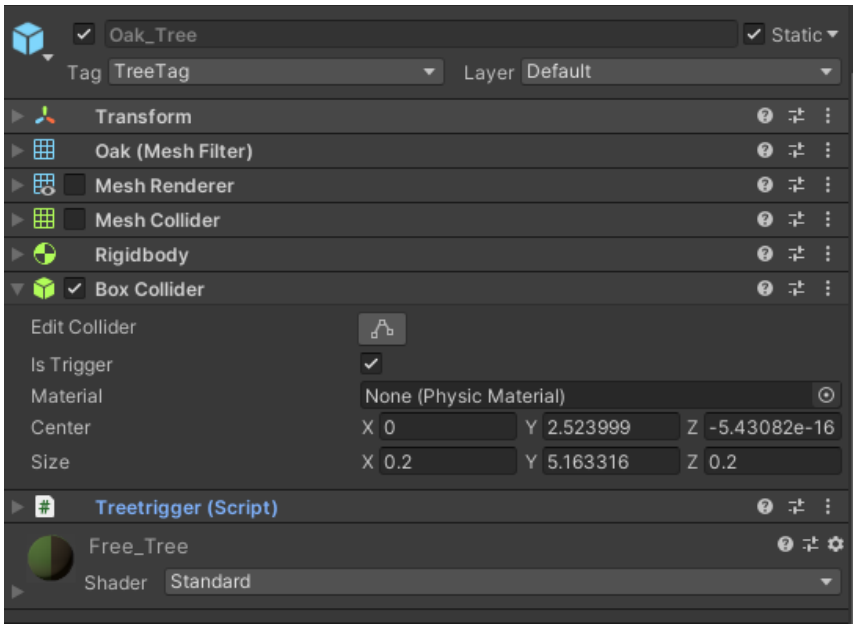

Figure 4.7 Example Object to Pool Setup

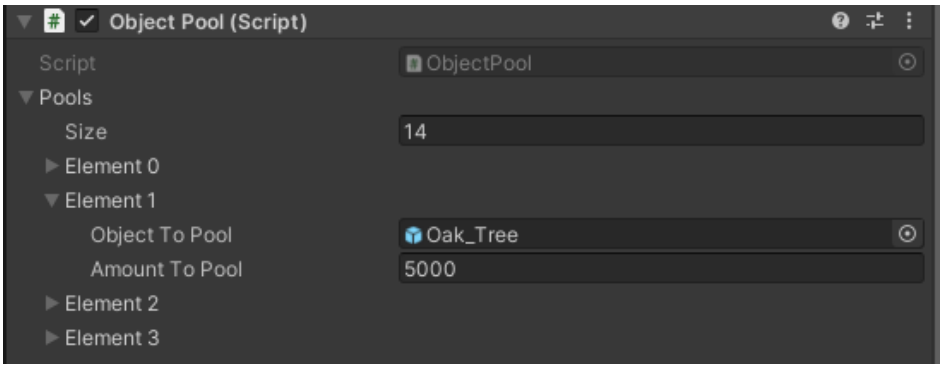

Figure 4.8 Pool Script Setup

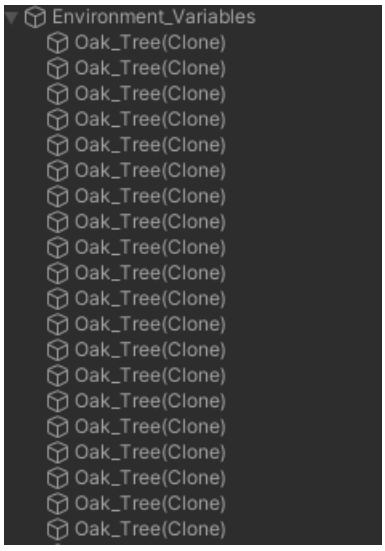

Figure 4.9 Pooled Objects in Hierarchy

4.3. Code Review

In this section the main goal is to review some of the most interesting and harder parts of code produced during the duration of this project, starting with a look at the noise generation, mesh creation, followed by the lod (level of detail) implementation, after this a review of the object pool code will be give, and lastly a quick review of one of the trigger scripts will also be presented.

4.3.1. Perlin Noise

In Perlin noise there are three main variables that must be understood, those are octaves, lacunarity and persistence.

Octaves simply represent the levels of detail in the final sample, lacunarity re presents how much detail is added or removed at each octave, and persistence determines how much each octave contributes to the overall structure of the noise map, the higher the persistence the more successive octaves contribute to the structure.

To code this, we start with a nested for loop that parses every point in the map and produces the Perlin value for each point, this is done using Mathf.PerlinNoise and passing in an x and y point that are selected using a random number generator and a formula that uses both the coordinates of the point being generated to produce a random sample , as Mathf.PerlinNoise can become quite repetitive if only the original point coordinate is used, the value is also multiplied by two and one is subtracted to give us a range of (-1,1) this serves simply for diversity and it will be normalized into the values (0,1) later in the code. All these points are stored into a two-dimensional array that will be used in a different script to produce the mesh for the terrain.

```
for (int i = 0; i < mapHeight; i++) //y
{
    for (int j = 0; j < mapWidth; j++) //x
    {
        amplitude = 1;
        frequency = 1;
        float noiseHeight = 0;

        for (int k = 0; k < octaves; k++)
        {
            float sampleJ = (j-halfWidth + octavesOffsets[k].x)/scale * frequency;
            float sampleI = (i-halfHeight + octavesOffsets[k].y)/scale * frequency;

            float perlinValue = Mathf.PerlinNoise( x: sampleJ,  y: sampleI) * 2 - 1;
            noiseHeight += perlinValue * amplitude;

            amplitude *= persistance;
            frequency *= lacunarity;
        }
```

Figure 4.10 Perlin Noise base function

```
for (int i = 0; i < mapHeight; i++)//y
{
    for (int j = 0; j < mapWidth; j++)//x
    {
        if (normalizeMode == NormalizeMode.Local)
        {
            noiseMap[j, i] = Mathf.InverseLerp( a: minNoiseHeight,  b: maxNoiseHeight,  value: noiseMap[j, i]);
        }
```

Figure 4.11 Perlin Noise Normalization Function

4.3.2. Mesh Generation

To generate the mesh a simple idea is presented, we must create a set of triangles from all the vertices we created using the noise script, once the array is split into a triangle array, we can build our mesh and add hights to the vertices using a hight curve to further adjust how our terrain will look, we must also create a vertex normal array so that light can properly be reflected from our mesh, this array will be a Vector3 array as it is a coordinate in a 3D space.

Firstly, we must create a vertex indices map, note that to create the triangles the lod level of the mesh is taken in consideration, the higher the lod level the less triangles the mesh will contain, also note that "meshTriangleSimplicity" refers to the lod.

```
for (int i = 0; i < borderedSize; i += meshTriangleSimplicity) //y
{
  for (int j = 0; j < borderedSize; j += meshTriangleSimplicity) //x
  {
    bool isBorderVertex = i == 0 || i == borderedSize - 1 || j == 0 || j == borderedSize - 1;

    if (isBorderVertex)
    {
      vertexIndicesMap[j][i] = boarderVertexIndex;
      boarderVertexIndex--;
    }
    else
    {
      vertexIndicesMap[j][i] = meshVertexIndex;
      meshVertexIndex++;
    }
  }
}
```

Figure 4.12 Vertex Indices Array Function

After we have the index for the vertices, we add the vertices to our MeshData class which can then be accessed by other scripts to retrieve the mesh, we then create the triangles and give them a hight based on the hight curve.

After normalization and baking of normal we simply add all the data into our new mesh that will be used to create our terrain.

```
for (int i = 0; i < borderedSize; i+= meshTriangleSimplicity)//y
{
  for (int j = 0; j < borderedSize; j+= meshTriangleSimplicity)//x
  {
    int vertexIndex = vertexIndicesMap[j][i];
    Vector2 percent = new Vector2(x: (j - meshTriangleSimplicity)/ (float) meshSize, y: (i - meshTriangleSimplicity) / (float) meshSize);
    float height = heightCurve.Evaluate(heightMap[j, i]) * heightMultiplier;
    Vector3 vertexPosition = new Vector3(x: topLeftx + percent.x * meshTriangleUnSimplicity, y: height, z: topLeftz - percent.y * meshTriangleUnSimplicity);

    meshData.AddVertex(vertexPosition,percent,vertexIndex);

    if (j < borderedSize - 1 && i < borderedSize - 1)
    {
      int a = vertexIndicesMap[j][i];
      int b = vertexIndicesMap[j + meshTriangleSimplicity][i];
      int c = vertexIndicesMap[j][i + meshTriangleSimplicity];
      int d = vertexIndicesMap[j + meshTriangleSimplicity][i + meshTriangleSimplicity];
      meshData.AddTriangle(a, b: d, c);
      meshData.AddTriangle(a: d, b: a, c: b);
    }

    vertexIndex++;

  }
}
```

Figure 4.13 Triangle Creation Function

4.3.3. Chunks

Now that we have a mesh, we simply need to position it in the world and make sure that the meshes are only visible when near the user, this is also where we implement out lod by calling different mesh densities depending on the distance from the user, here we also give the mesh all the attributes necessary for collision, rendering and any other necessary components that will make sure the mesh will behave and look how it should.

To achieve this, we firstly call our mesh when needed and set it to be visible, and once the user is too far away, we set it to be invisible making sure that only a select number of chunks are visible at any given time.

```
position = coord * size;
bounds = new Bounds( center: (Vector3) position,  size: (Vector3)( Vector2.one * size ) );
Vector3 positionV3 = new Vector3(position.x,  y: 0,  z: position.y);

meshObject = new GameObject( name: "Terrain Chunk");
meshObject.tag = "mesh";
meshRenderer = meshObject.AddComponent<MeshRenderer>();
meshFilter = meshObject.AddComponent<MeshFilter>();
meshCollider = meshObject.AddComponent<MeshCollider>();
meshRigidBody = meshObject.AddComponent<Rigidbody>();

meshRigidBody.isKinematic = true;
meshRigidBody.useGravity = false;

meshRenderer.material = material;

meshObject.transform.position = positionV3 * mapGenerator.terrainData.uniformScale;
meshObject.transform.parent = parent;
meshObject.transform.localScale = Vector3.one*mapGenerator.terrainData.uniformScale;
SetVisible(false);

lodMeshes = new LODMesh[detailLevels.Length];

for (int i = 0; i < detailLevels.Length; i++)
{
    lodMeshes[i] = new LODMesh(detailLevels[i].lod, UpdateTerrainChunk);
    if (detailLevels[i].useForCollider)
    {
        collisionLODMesh = lodMeshes[i];
    }
}
```

Figure 4.14 Chunk Creation Function

```
if (mapDataReceived)
{
    float viewerDstFromNEdge = Mathf.Sqrt( f: bounds.SqrDistance( (Vector3) viewerPosition));
    bool visible = viewerDstFromNEdge <= maxViewDst;

    if (visible)
    {
        int lodIndex = 0;

        for (int i = 0; i < detailLevels.Length - 1; i++)
        {
            if (viewerDstFromNEdge > detailLevels[i].visibleDstTreshHold)
            {
                lodIndex = i + 1;
            }
            else
            {
                break;
            }
        }

        if (lodIndex != previousLODIndex)
        {
            LODMesh lodMesh = lodMeshes[lodIndex];
            if (lodMesh.received)
            {
                previousLODIndex = lodIndex;
                meshFilter.mesh = lodMesh.mesh;

            }
            else if (!lodMesh.requested)
            {
                lodMesh.RequestMesh(mapData);
            }
        }

        if (lodIndex == 0)
        {
            if (collisionLODMesh.received)
            {
                meshCollider.sharedMesh = collisionLODMesh.mesh;
            }
            else if (!collisionLODMesh.requested)
            {
                collisionLODMesh.RequestMesh(mapData);
            }
        }

        terrainChunksVisiblePreviously.Add( item: this);
    }

    SetVisible(visible);
}
```

Figure 4.15 LOD Update Function

```
for (int i = 0; i < terrainChunksVisiblePreviously.Count; i++)
{
    terrainChunksVisiblePreviously[i].SetVisible(false);
}

terrainChunksVisiblePreviously.Clear();

int currentChunkX = Mathf.RoundToInt( f: viewerPosition.x / chunkSize);
int currentChunkY = Mathf.RoundToInt( f: viewerPosition.y / chunkSize);

for (int yOffSet = -chunksVisibleInViewDst; yOffSet <= chunksVisibleInViewDst; yOffSet++)
{
    for (int xOffSet = -chunksVisibleInViewDst; xOffSet <= chunksVisibleInViewDst; xOffSet++)
    {
        Vector2 viewedChunkCoord = new Vector2( x: currentChunkX + xOffSet,  y: currentChunkY + yOffSet);

        if (terrainChunkDic.ContainsKey(viewedChunkCoord))
        {
            terrainChunkDic[viewedChunkCoord].UpdateTerrainChunk();
        }
        else
        {
            terrainChunkDic.Add(viewedChunkCoord, new TerrainChunk(viewedChunkCoord, chunkSize, detailLevel, transform, mapMaterial));
        }
    }
}
```

Figure 4.16 Chunk Update Function

4.3.4. Object Pool

For this object pool the purpose was to have a scalable pool that could hold several different objects as well as being able to retrieve an object of a specific kind, to achieve this a structure of name pools was first created, this structure holds a GameObject and an int, these identify the GameObject to be pooled and the number of instances to be created.

We create an array of Pool structures and make it public as so it can be edited from within the Unity editor. Now we create a nested for loop, the external loop will be identified as (i) and will move from GameObject to be pooled, and the inner array is identified as (j) and will move the amount to pull.

Inside the inner for loop, we instantiate the GameObject j number of times, and add each copy of the object to a new GameObject array that will be placed inside the dictionary with key i, the loops repeat this for the number of Objects defined in the editor.

To retrieve an object, we simply have to pass the key associated with it and search the GameObject array for an object that is not active in the hierarchy.

```
pooledObjects = new Dictionary<int, GameObject[]>();

for (var i = 0; i < pools.Length; i++)
{
    GameObject[] p = new GameObject[pools[i].amountToPool];

    for (int j = 0; j < pools[i].amountToPool; j++)
    {
        var tmp:GameObject = Instantiate(pools[i].objectToPool, environmentVariables.transform, worldPositionStays: true);

        tmp.SetActive(false);

        p[j] = tmp;
    }

    pooledObjects.Add(i,p);
}
```

Figure 4.17 Creation of Pool Function

```
var p:GameObject[] = pooledObjects[i];

for (var j = 0; j < pools[i].amountToPool; j++)
{
    if (!p[j].activeInHierarchy)
    {
        return p[j];
    }
}

return null;
```

Figure 4.18 Retrieving Pooled Object Function

4.3.5. Triggers

The trigger scripts are quite simple, we just need to check for collision between the object and the terrain, and once the object comes in contact with the terrain, we set its render to be active and mesh collider to be active, and now our object is fully functional inside the scene.

The only rules come with hight relative to the terrain and placement on top of other objects, in which case we look for collision with other objects and if it returns true, we simply set the object back to inactive which will effectively return it to the pool, same for height, if an object is too high or low in the terrain, we can simply set it to be inactive.

A trigger exit is also present for recycling objects, the user carries an object which is invisible, this object is tagged as “Environment”, this serves to mark the area in which objects can be visible, after leaving the Environment objects are deactivated and set back into the pool for future use.

```
if (gameObject.transform.localPosition.y > 5f || gameObject.transform.localPosition.y < -4f)
{
    gameObject.SetActive(false);
    //Debug.Log(gameObject.transform.position.y);

    GetComponent<MeshRenderer>().enabled = false;
    GetComponent<MeshCollider>().enabled = false;
}

if (other.gameObject.CompareTag("TreeTag") || other.gameObject.CompareTag("VegetationTag"))
{
    gameObject.SetActive(false);

    GetComponent<MeshRenderer>().enabled = false;
    GetComponent<MeshCollider>().enabled = false;
}

if (other.gameObject.CompareTag("mesh"))
{
    GetComponent<MeshRenderer>().enabled = true;
    GetComponent<MeshCollider>().enabled = true;
}
```

Figure 4.19 Tree OnTriggerEnter Function

```
if (other.gameObject.CompareTag("Environment"))
{
    gameObject.SetActive(false);

    GetComponent<MeshRenderer>().enabled = false;
    GetComponent<MeshCollider>().enabled = false;
}
```

Figure 4.20 Object OnTriggerExit Function

4.5. Conclusion

In this chapter an overview of the Unity basic principles was given, this presentation covered game objects, different components that make up the rendering pipeline and the physics engine, and prefabs. This was the required introduction to Unity in order to easily understand the rest of the sections.

Promptly after, presentation of features and how different scripts, game objects and various components can be used to achieve the desired outcomes was give, in this section the composition of different features was explored and the basics to understand how this project was built and how it functions was now given.

Lastly, a review of some of the projects main functions was given, the project is comprised several scripts, so only a select few functions that were particularly interesting or challenging were presented and an explanation of how they operate was given.

5. Testing and Evaluation

5.1. Introduction

In this chapter an overview of testing methods used, and testing performed on the project will be presented, this will include both unit testing and play testing and how both were performed. An evaluation of the project will also be presented although the scope of this evaluation will be limited to a small group of individuals due to the restrictions forced due to the pandemic, this will be further limited due to the lack of equipment while testing, only a select number of testers had the ability to use a HMD, and due to this the evaluation of this project is less than optimal.

The majority of testing performed on this project was play testing, given the visual nature of the project and the short number of interactions playtesting was very viable, this is not to say that Unit testing was not attempted. Unit testing in Unity is both simple and arduous, given the already low coupling nature of unity code, although unity contains two methods of testing, editor and play mode testing, given the nature of this project most of the testing that needs to be produced is in play mode testing, and this proved to be challenging, this will be further expanded in subsequent subsections.

The evaluation of this project was severely hard, most of the evaluation is based on how the predicted design compares to the achieved level of completion. User testing is severely limited, and evaluation gathered is based on personal opinion of the user, and with such small sample size a true indication of the project is very hard to derive, hence the evaluation of this project will be mostly focused on the technical aspects, and completion aspects, as there simply was not enough tests made to derive a good model of feelings of presence and immersion, as well as emotions and experiences from the user.

5.2. Testing

In this section a quick view of Unit testing in Unity will be presented, this will be coupled with screenshots from the Unity editor.

The different approaches to testing and how testing was conducted will also be presented in subsequent subsections.

This section will explore issues faced while testing and possible solutions to these.

5.2.1. Unit Testing in Unity

Unity comes equipped with its own testing framework, which is based on unit testing.

This framework is very powerful as it allows all tests to be kept inside the folder structure and ran from inside the editor, a list of tests ran and tests to be ran is also presented, and the display of pass or fail is also shown.

The Unity testing framework presents two different methods of testing, editmode and playmode, these two methods are used in their own way.

Editmode testing can be ran from the editor without the need to run an instance of the experience, this makes for faster tests, the one drawback to this is the fact that scripts with the MonoBehaviour base class need to be tested inside playmode unless Interfaces are used to produce a mock class.

Playmode testing is used to test scripts with MonoBehaviour, or scripts that need to be tested while the experience is being ran, playmode testing is much slower than editmode.

In this project most scripts use the MonoBehaviour base class, but best practices would tell us to create Interfaces for our scripts and run tests in editmode.

Unfortunately, Unity is severely strict with its testing, and there is a severe learning curve in producing good quality tests. This would not have been an issue, but due to lack of experience, unit testing was left as the last stage of the project, this proved to be one of the biggest errors in the production of this project, as all scripts now need to be refactored to be able to be tested, and there simply is not enough time. A description of how unit testing in Unity will still be displayed, although, TDD as a testing methodology probably suits Unity much more.

In a basic form, Unity tests are produced following a guideline of, assignment, acting, and asserting, these three steps are the foundation of unit testing in Unity, where the first stage simply sets the variables, the second stage is where we act out our functionality, and finally we compare the results obtained with the results expected.

```
[Test]
public void MakeSureNoiseReturnsMap()
{
    //ACT
    var map :float[,] = Noise.GenerateNoiseMap( seed: 1,  mapWidth: 255 + 2,  mapHeight: 255 +

    //ASSERT
    Assert.IsNotNull(map);
}
```

Figure 5.1 Test Example

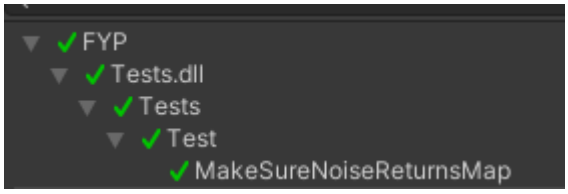

Figure 5.2 Test Runner

5.2.2. Play Testing

Play testing is the most natural way of testing, and it would be the form of testing users would do the most, play testing simply refers to the act of running the experience and looking out for any visual bug or simply to gauge how the scene composition looks.

Through the process of play testing many visual bugs were found, for instances, while play testing, it was noted that sections of the scene were severely dark, through process of elimination it was then discovered that the absence of a second light source in the scene was causing extreme shadows to appear.

This is an example of a visual bug that could not have possibly been captured during any other method of testing but play testing.

Play testing also greatly helped in testing out object density and colouration inside the scene.

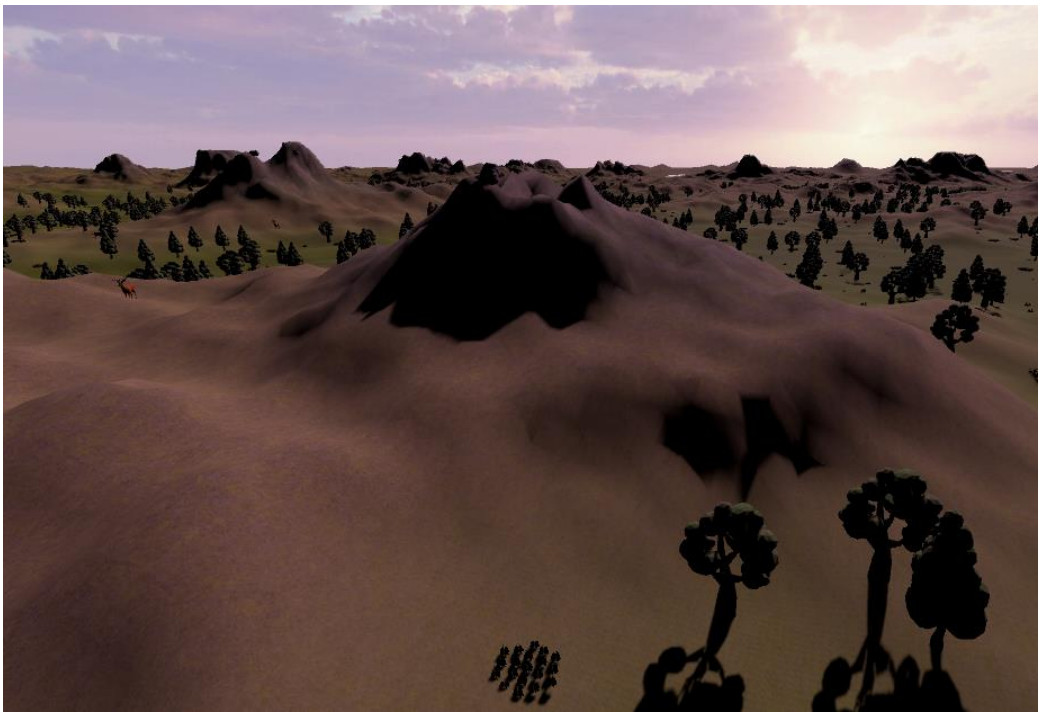

Figure 5.3 Experience Screenshot Before Lighting Fix

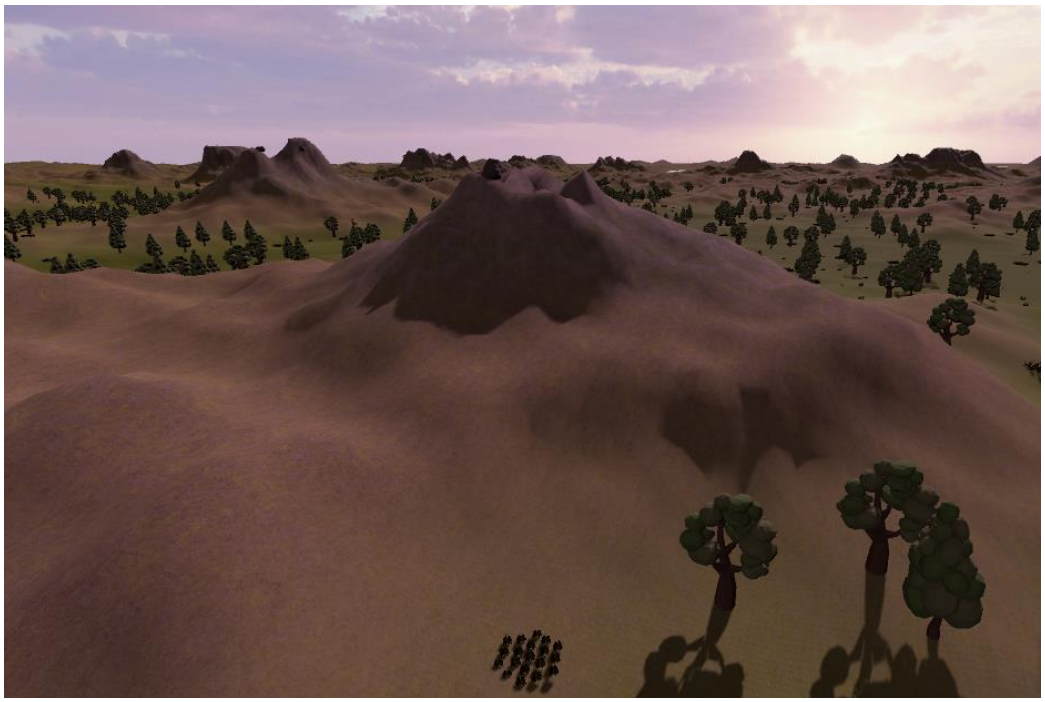

Figure 5.4 Experience Screenshot After Lighting Fix

5.3. User Evaluation

The user evaluation experience was an arduous task, the lack of close-up social interaction makes the evaluation of a project based in VR severely difficult.

Although the best attempt was made to attempt basic user Evaluation, a test run of the project was demonstrated to a selection of users, and relative experience opinion was obtained.

Together with that a level of immersion was also gaged, a visual opinion was also noted, and in the end users were asked to indicate any features that they felt missing.

The results were not severely divergent which can be due to the low sample size, and for that reason conclusions reached are not without a margin of bias.

From 10 users to test out this project only 2 possessed an HMD which might have also injured the results.

Visually, 7/10 users reported the graphics used to bee of satisfactory to low quality, with 3 signalling satisfaction, world interactions were rated limited by a majority, in terms of immersion, on a scale of 1 being not immersive to 10 being very immersive, taking in consideration that these results can be severely bias due to the lack of HMD, the average was of 6.2 in the scale.

For features the largest requests were of a day night cycle and for completion of animation with almost full majority noting these two sections. An increase in diversity was also mentioned by some participants.

The results obtained fall in order with predicted outcomes as the experience contains the foundational features but lacks on the smaller refinements that would push it towards completion.

5.4. Conclusion

In this section a look at testing was presented, testing was not the most advanced section of this project due to the lack of knowledge, it was discussed that unit testing presented a large challenge and it should have been performed in parallel with code production, or TDD should have been used.

A look at play testing and its advantages was also presented, coupled with some in editor screenshots.

User evaluation was also present, and a walkthrough of initial impressions was given, together with results from the queries imposed on the testing users.

6. Conclusion

6.1. Introduction

In this chapter a view of the issues present in this project will be presented, these will include expected issues drafted before the development stage as well as encountered issues through out the development, these will then be compared.

A list of future work that could be performed to add to the project is also present, this is comprised of features not fully developed, features that were initially planned but never developed and features that were requested by test users.

To conclude the document, a final conclusion of the document and project will be presented, this conclusion will include a view of all section of this document and a view of the project from the development side.

6.2. Issues

This section will be comprised of two subsections, predicted issues, a subsection written before the development of the project started that contains a description of issues that were expected, and an issues encountered subsection, in this subsection a description of issues that were encountered during the project will be presented, together with a comparison between previously expected and actual issues.

6.2.1. Predicted Issues

This section was written before any development started on this project, and is written in the present perfect, please note that this is only to preserve authenticity of the original text. This text was written on the 15/12/2020 and refers only to knowledge acquired up to and exclusively to that date.

Issues so far encountered and identified on this project mainly consist of testing problems, as due to the current working conditions and nature of the project the optimal testing conditions are not supplied.

In the projects best interest user testing would have been carried out after each stage so that user input could be taken in consideration for the next stage of project development, although with limited interactions the pool of users able to test the system is greatly reduced as the amount of people that have an HMD available to them is limited and no outside testing with the VR treadmill will be possible.

This although will stop being an issue if the current world climate changes, and even though a slight delay in user testing will be felt, changes can still be made to accommodate user responses to the project and improve the overall user experience.

Other than testing time management is also a concern, the project itself is not an overly complex program but it is a rather lengthy one, this can be a cause for concern taking in consideration the project aims, a project that feels incomplete would break the user's immersion and sense of presence.

One other issue found so far is the difficulty in producing class diagrams, it was an aim of the design chapter, but due to Unity and its component system, this means that a class diagram for this project would just be comprised of loose classes with no real link to other classes. So far, a research of entity/component models is under process so that a comprehensive visual depiction of the system can be produced in the future.

6.2.2. Issues Encountered

Given the project state at the deadline for this dissertation, a description of issues encountered will be promptly displayed.

From the start of this project, it was known that given the project aims an incomplete project would hurt the immersion feeling as anything that does not function in a realistic manner can and will break immersion for the user, this posed to be one of the biggest threats for the project given that time constraints directly affect the amount of work possible.

A large issue present was the project aim itself, a large project with several visual components and a focus on immersion requires an immense amount of time, and even though a good foundation for a project was achieved, the incomplete feeling severely hurts the immersion factor, furthermore given the lack of space and time the building of the VR treadmill was not achieved, this severely injured the feelings of immersion and presence, as the main immersion factor was the HMD and treadmill combined with motion tracking.

The biggest unforeseen issue was the terrain generation, by far the most time-consuming part of the project due to the immense amount of time researching different approaches, initially the Unity Terrain module was going to be used, but after testing and development it was noted that this would be insufficient for the aims of the project, this change costed precious time.

User testing and evaluation was also severely hard to obtain, this project required more user interaction than what it gathered, attempts were made to demo the project to users but without the ability to use an HMD and experience the project the way it was intended to be experienced, data gathered was less than optimal and no real measure of immersion could be perceived, this also impacted the way the project looked as it was very hard to test for visual saliency due to the lack of HMD, colours and density were also hard to balance due to similar issues.

Testing in general proved difficult, it first was thought that unit testing would be made easier by Unity given the nature of code being produced, but Unity is very particular over its testing and scripts must be written with testing in mind, this meant that all scripts would have to be heavily refactored to fit the testing model, and this was simply impossible in the amount of time left, Unity unit testing also presents itself with a large learning curve.

In contrast with expected issues, the issues encountered fall within the expected range, time and length of the project was predicted as an issue due to its immersive qualities, and testing was severely constricted due to the world climate.

6.3. Future Work

This project is a great foundation and demo for a possible experience, but time constraints made it so that final features were left incomplete, some minor visual bugs were left unfixed, and final polishes were left unmade.

A description of all future work that was hoped to be achieved will be presented shortly, all of this work was either originally planned to be developed for the project or was devised throughout the development stage.

Firstly, the main things to work on for the future would be interactions, a set of user animal interactions as well as animal to animal interactions deserves to be properly implemented and given some time this feature can be implemented easily using the already established foundation present in the project.

After, a better implementation of object population should be devised, the current implementation fails in one regard, it allows objects to spawn near the user, this greatly reduces immersion as objects appearing in front of the user does not reflect a natural scenario.

New models combined with rigs and animations should be constructed, given this projects time span, unrigged modules had to be used, since these are the only ones that can be obtained for free, this means that either models are made from scratch and a bone structure is implemented for animations or they are purchased, none of which were a viable alternative during the development of this project.

Taking in consideration user input, a day-night cycle would also be a welcome addition in future work, this will aid in the visual variety presented to the user, helping to maintain user engagement for longer, sound effects also need to be added, as the lack of them can be felt when experiencing this project.

One of the largest sections that was meant to be implemented was a VR treadmill and motion tracking, this would be a pivotal feature to work on in the future, as it would greatly increase both the senses of immersion and presence in the project, although some time and a lot of space for construction would be required.

Also, given that everything else has been implemented, a greater variety of animal and plant models, coupled with more interactions so that the exploration of distortion of reality can be properly implemented, this could be attributed to the implementation of more creative animals and plants, and the implementation of more visually scoring scene alterations and colour switches.

6.4. Conclusion

This document goes through the thought process behind the stages of this project, from initial inspiration, through research of topics that were deemed important in the aid of developing this project. A design overview of the project was also given to aid in its development, where sections from the research chapter were considered and explored when deciding on major design factors, the development of the project was also described and explained, from the fundamentals of Unity to an explanation of all the features present in the project. A testing section is also present in this document, and it explains and demonstrates the different types of testing used during the completion of this project, and finally the conclusion chapter is presented, in this chapter a view of issues expected and encountered can be seen, together with a description of all future work that can possibly be done in this project to assist it into becoming truly immersive, and then the final conclusion, the subsection currently being presented, where the entire document will be reviewed.

In the writing of this document, several main decisions were made and explored. The research and understanding of immersion and feelings of presence were some of the main points explored when it comes to the project aim, the research of software; design methodologies; and testing methodologies were also severely important as a deeper understanding was obtained in each area, and several new resources previously unknown prior to the writing of this document were now available, making the research and design section an important foundation for what this project has achieved and can still achieve if future work is applied to it.

It was decided in this document that, Unity would be used for its development, meaning that the language used for this project was C#.

Blender would have been used as a 3D modelling and animation tool given sufficient time; Audacity and Steam Audio for audio production and implementation, also given enough time.

The Oculus Quest was used as the testing HMD, although any other HMD could have been used to experience the project.

As for design methodologies, inspiration was obtained from XP and FDD to create a methodology that is believed to best suit the project and its development conditions, by having three different stages of project development, these stages would in turn allow for progress to be measured and for different testing methodologies to be deployed in each, this was successful as even if the development stage was not followed directly due to set backs in early sections, it still allowed for the tracking of progress

and to gauge the sections that would later need to be removed in order to achieve a satisfactory level of completion.

The development of the project was both simple and complicated, surely the research and preparations made were a great help and definitely a deciding factor in this projects completions, but it was not without challenges, it was soon apparent that some modules were much more complex than initially expected, this coupled with other educational obligations caused a serious time delay in the project, making it so that the vision for the completed project was not achieved, this although does not come with grieve, the project still holds a satisfactory completion and a strong foundation on to where a great experience can be built, and more was learned through the efforts and complications than it could ever have been hoped for.

For testing, initially it was believed that limited amount of testing methodologies would need to be employed, this was due to gross lack of information in the topic, after the literature review chapter it was abundantly clear that for an optimal development cycle several different testing methodologies will need to be used at different times, these will include a mix between black and white box testing, in small and big modules, leaving white box testing for bug and feature testing, and black box for user testing, as user input is crucial for the project to reach it's aims. It was although later clear that for the majority of the project unit testing would be used for individual modules and play testing would be used to fix visual bugs or tweak any visual aspects of the experience, user testing was also clearly limited, and the amount of time allocated, and number of users initially expected to test the project was simply unreachable, it was also clear that testing should have been a concern from the beginning, as unit testing proves very difficult in Unity if code isn't written in a specific way, and there was not enough time towards the end of the project to refactor code to fit the testing standard, Unity unit testing also presents itself with a large learning curve which only contributed towards the time problem. In insight it is believed that TDD should be looked into when creating code for Unity.

In conclusion, the project document was produced and completed in the desired fashion, the research and design stages were thoroughly crafted and the development stage reached a satisfactory level of completion, although not all features planned were completed to the desired level and due to time management problems, some of the features that would tie the project together and bring the desired level of immersion were not developed and some visual and behaviour bugs are still present, the level at which the project finds itself is still satisfactory, a good foundation is present and due to the nature of the code produced these modules can be reused and or built upon to generate the final vision of this project without much more effort.

Given the opportunity the redo this project, a better look at complexity would be due, this project was due to suffer from time constraints as due to the aims of the project any lack of completion would hurt the immersion feeling required to fully achieve the scope of the project, this is clearly felt throughout the experience.

Given this, it is not believed that the project fully achieved its aims as there is somewhat a lack of immersion and some of the main concepts did not transcribe well given to the lack of completion, this is not to say that the research, design, and testing were not properly conducted but only that the scope and aim of the project was more complex than originally expected, and in hindsight this project would have benefited from a lower scope as the production of a VR experience encompasses much more than just the code produced, and is severely time consuming. Perhaps a scope which would focus more on certain modules and implementations and not on as many features would have been preferred, has this project aimed to produce an immersive experience and that immersion is lost with the slightest lack of detail, majorly the lack of animations and sound.

Bibliography

1. Lui M. Immersive Environments. In: Gunstone R, editor. Encyclopedia of Science Education [Internet]. Dordrecht: Springer Netherlands; 2021 [cited 2020 Nov 22]. p. 1–2. Available from: https://doi.org/10.1007/978-94-007-6165-0_39-1

2. McMAHAN A. Immersion, Engagement, and Presence. 2003;20.

3. Csikszentmihalyi M. FLOW: The Psychology of Optimal Experience. 2000;6.

4. Roohi S, Forouzandeh A. Regarding color psychology principles in adventure games to enhance the sense of immersion. Entertain Comput. 2019 May 1;30:100298.

5. Slater M, Spanlang B, Sanchez-Vives MV, Blanke O. First Person Experience of Body Transfer in Virtual Reality. PLOS ONE. 2010 May 12;5(5):e10564.

6. Celikcan U, Askin MB, Albayrak D, Capin TK. Deep into visual saliency for immersive VR environments rendered in real-time. Comput Graph. 2020 May 1;88:70–82.

7. M L. C# vs C++ Comparison: Find Out the Difference Between C# and C++ [Internet]. BitDegree.org Online Learning Platforms. [cited 2020 Nov 28]. Available from: https://www.bitdegree.org/tutorials/c-sharp-vs-c-plus-plus/

8. Unity vs Unreal Engine - Which is Better for Virtual and Augmented Development? [Internet]. Circuit Stream. 2020 [cited 2020 Nov 28]. Available from: https://circuitstream.com/blog/unity-vs-unreal/

9. Foundation B. blender.org - Home of the Blender project - Free and Open 3D Creation Software [Internet]. blender.org. [cited 2020 Nov 28]. Available from: https://www.blender.org/

10. Steam Audio [Internet]. [cited 2020 Nov 28]. Available from: https://valvesoftware.github.io/steam-audio/#learn-more

11. Manifesto for Agile Software Development. :10.

12. Newkirk J. Introduction to agile processes and extreme programming. In: Proceedings of the 24th International Conference on Software Engineering ICSE 2002. 2002. p. 695–6.

13. Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional; 2000. 228 p.

14. Feature Driven Development. :20.

15. Chang M. Agile and Crystal Clear with Library IT Innovations. :16.

16. Cockburn A. Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams. Pearson Education; 2004. 419 p.

17. Hunt A, Thomas D. Pragmatic unit testing in Java with JUnit. Raleigh, N.C: Pragmatic Bookshelf; 2003. 159 p. (Pragmatic starter kit series).

18. Integration Testing: What is, Types, Top Down & Bottom Up Example [Internet]. [cited 2020 Dec 9]. Available from: https://www.guru99.com/integration-testing.html

19. Performance Testing Tutorial: What is, Types, Metrics & Example [Internet]. [cited 2020 Dec 9]. Available from: https://www.guru99.com/performance-testing.html

20. What is System Testing? Types & Definition with Example [Internet]. [cited 2020 Dec 9]. Available from: https://www.guru99.com/system-testing.html

21. What is User Acceptance Testing (UAT)? with Examples [Internet]. [cited 2020 Dec 9]. Available from: https://www.guru99.com/user-acceptance-testing.html

22. Online Gantt Chart Software & Project Planning Tool | TeamGantt [Internet]. [cited 2020 Dec 4]. Available from: https://www.teamgantt.com

23. Sebastian Lague. Procedural Landmass Generation (E01: Introduction) [Internet]. 2016 [cited 2021 Apr 2] .Available from: https://www.youtube.com/watch?v=wbpMiKiSKm8&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3

24. Scratchapixel. Perlin Noise: Part 2 [Internet]. Scratchapixel. [cited 2021 Apr 2]. Available from: https://www.scratchapixel.com//lessons/procedural-generation-virtual-worlds/perlin-noise-part-2

Appendix

Figure A.1 Reference Picture 1

Figure A.2 Reference Picture 2

Figure A.4 Reference Picture 4

Figure A.3 Reference Picture 3

Figure A.5 Reference Picture 5

Figure A.6 Reference Picture 6

Figure A.7 Reference Picture 7