NATURE SIMULATION SANDBOX GAME

A REPORT SUBMITTED TO MANCHESTER METROPOLITAN UNIVERSITY FOR

THE DEGREE OF BACHELOR OF SCIENCE

 IN THE FACULTY OF SCIENCE AND ENGINEERING

07/05/2021

By

Nesta Chukwuma Rubio – 18061418

Project (6G6Z1101\_2021\_9Z6)

School of Computing, Mathematics and Digital Technology

# Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Ethics Reference Number: 26058

Signed:



Date: 07/05/2021

# Acknowledgements

I would like to express my deepest appreciation to my project supervisor, Huw Lloyd for always giving valuable advice and guidance. Also trusting in my work and abilities compelling me to stay on track with the project and not only meet his expectations but set high ones for myself.

I am also grateful to all my friends and most of my family for their unwavering support. Even though at the time their words might have not seemed like much to me, looking back on them they really helped a lot.

# Abstract

This report analysis all the steps for creating a prototype of a nature simulation sandbox game. Before the development of the prototype extensive research was required in the form of tutorials about how to make a game using Unity and the behaviour of the wildlife being implemented. For the development of the prototype an agile methodology with weekly sprints was used and participants were selected for black box testing for more practical feedback. The analysis shows a prototype exhibiting basic mechanics of a nature simulation with extensible code and a research driven approach. I conclude that a lot of research is required for the development of a game prototype, particularly a game related to nature that has very intricate behaviour patterns. The project was limited to a 7-week time period so future research could ensue with the implementation of different types of wildlife and a focus on the relaxing aspect that was missing from the first iteration of development.

Table of Contents

[Declaration 1](#_Toc71312760)

[Acknowledgements 2](#_Toc71312761)

[Abstract 3](#_Toc71312762)

[Introduction 5](#_Toc71312763)

[Chapter 1 6](#_Toc71312764)

[1.1 Literature Review 6](#_Toc71312765)

[1.2 Design 8](#_Toc71312766)

[1.3 Implementation 11](#_Toc71312767)

[1.4 Evaluation 19](#_Toc71312768)

[1.5 Conclusion 20](#_Toc71312769)

[1.6 Bibliography 21](#_Toc71312770)

[Appendix A 22](#_Toc71312771)

[Feasibility Study 22](#_Toc71312772)

[Course-Specific Learning Outcomes 22](#_Toc71312773)

[Project Background 23](#_Toc71312774)

[Aim 23](#_Toc71312775)

[Objectives 23](#_Toc71312776)

[Problems 24](#_Toc71312777)

[Required Resources 24](#_Toc71312778)

[Schedule 25](#_Toc71312779)

[References 25](#_Toc71312780)

[Appendix B 26](#_Toc71312781)

[Showcase Presentation 26](#_Toc71312782)

# Introduction

The purpose of this end-of-project report is to showcase how a prototype for a video game is made and all the steps and research required to achieve that goal. The project is required to be a prototype of a simulation game either something nature related like Equilinox or genetics and survival related like Spore. Since nature simulation interests me more, that aspect of life simulation is the focus of the project.

The aim of this project is to learn how to create a game, more specifically, a prototype for a relaxing game featuring entities that can grow and spread on their own, an ecosystem that consistently sustains itself relevant to what the user chooses to add to the environment. With the consistent increase in interest in video games over the years there is more of a demand than there ever was. Especially during recent times where the jump in a vast variety of video game players is so prevalent due to the current circumstances. The prototype must showcase the games main functionalities and mechanics in action and be very data driven. This report highlights all the research and development steps required to create something so influential to present society. Unity is be used to create environments with prefabs of plants and animals and using C# scripts to program them to act like their real-life counterpart would. Following the factory design pattern is vital to make the prototype extensible enough to be picked up by another developer and easily allow the inclusion of new capabilities and functionality. Various tutorials are available on Unity’s own website that will aid with the development of a 3d game and research on the behaviour of certain plants and animals needs to be undertaken to make the simulation as accurate as possible.

Research on Equilinox is also be needed as that game is the main inspiration for this prototype and various playthroughs are available to watch online. The term ‘prototype’ means that this project is only be a preliminary version of the full game so it is not as detailed as a final product, instead the main objective is on getting the functionality of the game to work properly so it can later be developed into something more, helping put into perspective more about how video games are made. The main functionality of the game being the process of having plants that grow and spread over time and having animals that move around, eat and reproduce and putting them together in a suitable environment so they can interact with each other and create a simulation of nature. The game needs to have an easily comprehensible graphical user interface to aid any user through all the features and be well optimised for the majority of systems.

With the short time frame and weekly meetings with the project supervisor, doing sprints in an agile methodology is the best process for this situation. Meeting with the supervisor to plan an idea, do research on the topic and then developing and testing it before the next meeting and then repeating that process every one to two weeks and documenting each sprint on an agile sprint board.

# Chapter 1

## 1.1 Literature Review

The project description mentioned being able to base the project off of either a nature simulation game like Equilinox where you can create and nurture your own ecosystems in a relaxing atmosphere or a life simulation game like Spore where you control the development of a species from its start as a microorganism and follow its evolution. Given the current state of the world and a global increase in stress levels, a relaxing game like Equilinox is the perfect opportunity to create something helpful.

Since the project is only a prototype that simulates nature, Unity is a more suitable engine than its main competitor, Unreal Engine. This is due to the fact that Unreal Engine focuses more on the visual aspects of games which usually requires a larger team of members that are dedicated to different areas and a relatively powerful PC setup whereas Unity is more centred around simplicity and effectivity making it easy to pick up for beginners and for independent projects. The bigger asset store also makes it easier to find assets for nature. In terms of deciding what dimension to develop the prototype in, a 3D approach is more suitable seeing as how the inspiration is Equilinox, a 3D game. Additionally, creating a 3D game will add a layer of complexity to the relatively simple outside perspective of a simulation of nature by having to deal with more challenging physics, collisions and camera management.

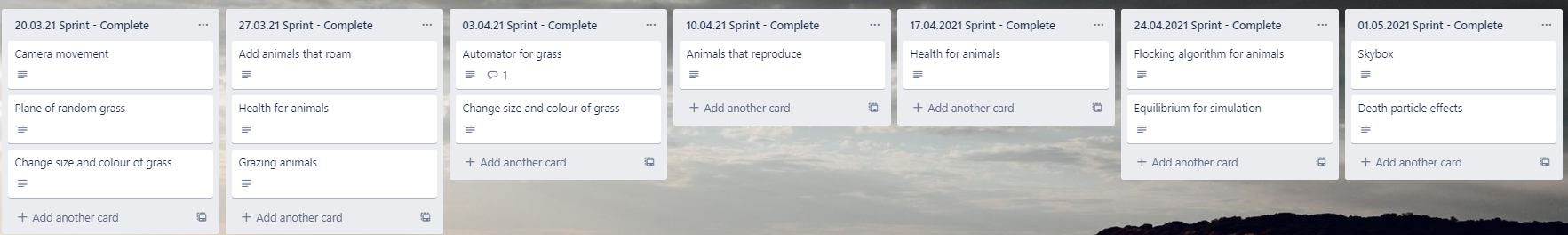
In order to accurately represent the process of creating a prototype it has been shown that the agile methodology is the most suitable approach because of the use of sprints. Sprints are short periods, usually 1-3 weeks, where tasks are set out to complete within that time frame. This makes developing independent projects a lot easier to manage and given the 7-week block dedicated to the project and having meetings with the project tutor each week, sprints offer a lot more flexibility which is perfect for prototypes because changes can be made easily if necessary and without conflict after any of the weekly meetings. Whereas with other methodologies like waterfall where it is discouraged to go back and revisit previous phases changes are a lot harder to be made or with spiral where even though steps are iterative it is more appropriate for long-term projects. Although, Rapid Application Development shares various similarities with the agile methodology and focuses on creating prototypes within tight deadlines, the advantage that agile has over Rapid Application Development is the focus on incremental features allowing issues to be addressed quicker and features to be adjusted earlier through feedback from meetings.

Seeing as how Unity are a very beginner friendly engine a variety of tutorials are available such as the LEGO® Microgame that help teach how behaviour scripts are made and implemented and several other beginner tutorials provided by Unity themselves that help with things like UI and game commands, all being relevant to the development of a simulation. And because Unity have such an extensive catalogue of free assets available it is very useful in aiding with the visual representation aspect by finding low-poly prefabs that do not require much animation work.

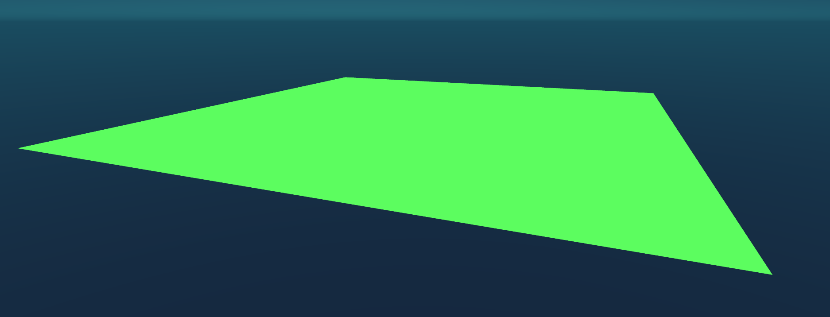
When first starting a world in Equilinox you are presented with a bare plane and a steadily increasing point system that is used to unlock certain plants and animals. The first plant being grass and the first animal being sheep. Considering the project being a prototype, those two entities are the only ones needed to accurately represent an ecosystem and focus more on the functionality rather than the features.

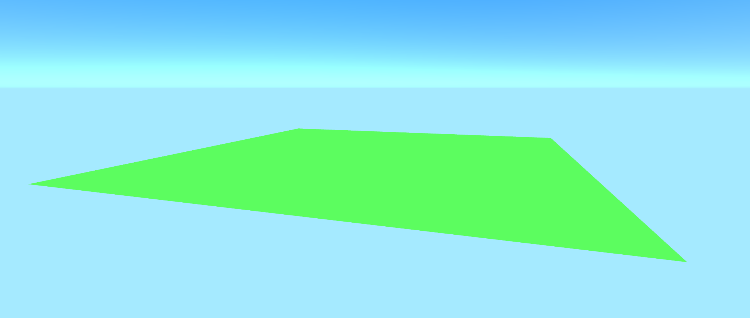
Sheep being an important part of the game their behaviour is very important to research and emulate. “*Sheep have a strong flocking instinct. They feel safer when gathered together. Sheep need other sheep to feel safe. Separating one sheep out from the rest of the flock is disturbing and frightening to them.” – (Goodling, 2018).* In order to accurately represent this kind of behaviour, boids will need to be implemented. This way the sheep will move together all the time. This can be further developed in the future by adding emotion so if sheep are separated, they start to feel anxious.

Boids is an artificial life simulation developed by Craig Reynolds with the aim of replicating the behaviour of a flock of birds. Instead of controlling the movement of entire flock, boid simulation defines the actions of each individual boid. Providing rules to produce a result realistic enough to accurately simulate a flock of birds. A theory proposed by Richard Dawkins talked about mentality of herd behaviour. *“We can picture each prey individual as being surrounded by a 'domain of danger'. This is defined as that area of ground in which any point is nearer to that individual than it is to any other individual.” – (Dawkins, 2006, p. 151)* Although there are no current hostile animals it is important to the functionality of future development and accuracy of simulating nature. An individual sheep on its own has a wide domain of danger since to a predator they are the closest prey. In response to that, the strategy to reduce the domain of danger lies in forming a pack. Since each sheep is trying to reduce its domain of danger as much as possible my getting as close to the middle of the herd as it can this behaviour complements the functionality of boids very well as the nature of boids is to get as close to the average position of other boids as it can. Even though there is a limit to how close boids can be to each other due to the separation rule.

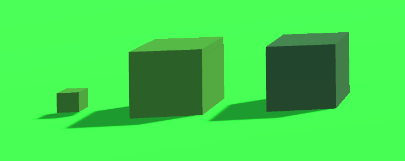
An agile sprint board is shown below documenting each of the weekly sprints taken place and a general description of each process that took place.

## 1.2 Design

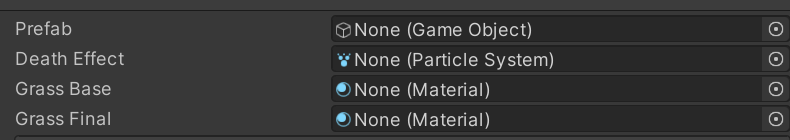
To start the project a simple green plane is added to represent a field and a skybox from the asset store to easily represent day and night.



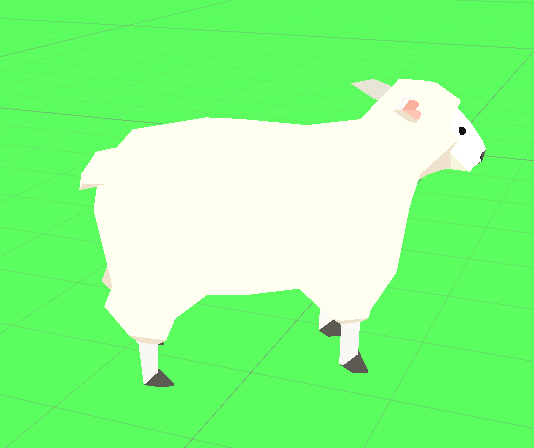
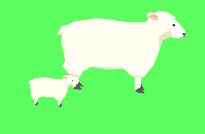
Starting with the grass prefab, cubes were first used to get a general feel for how the grass should act. Over time the cube will grow and slowly start to change colour.

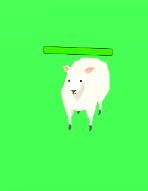


After getting the cube to grow and change colour they were replaced with actual grass models downloaded from Unity’s asset store. Factory method pattern is prevalent here, the script causing the object to grow and change colour is not dependent on the object the script is attached to, but the prefab and material stored in the interface. This way the models can be replaced with different models of grass or different materials related to different environments.

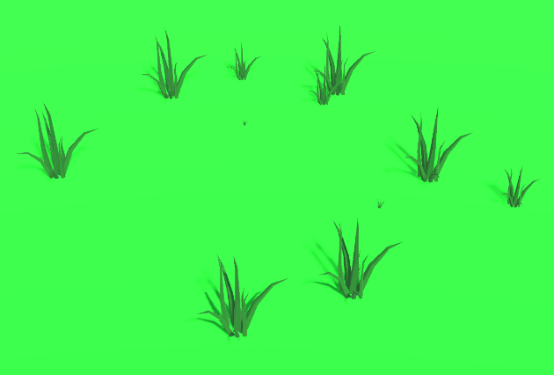
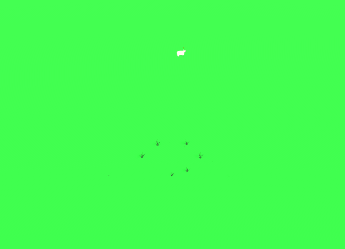


The sheep prefab also comes from the asset store and like the grass prefab, their size scales with their age. Even though it is easier to just use the default game objects provided by Unity, like a cube or a cylinder, it helps more to understand the situation by being able to visualise those entities whilst developing the game and helps with testing.

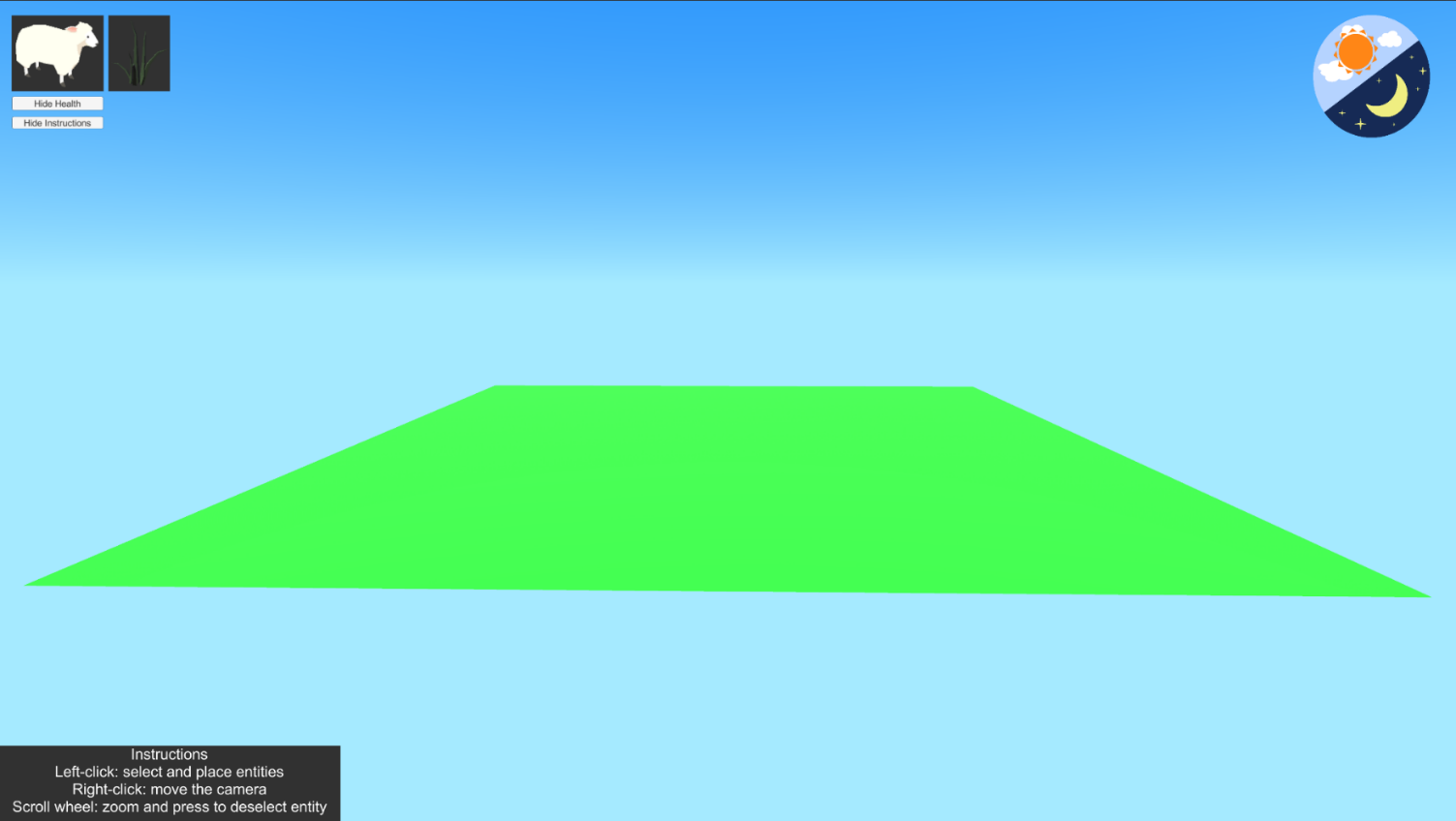


To display a sheep’s health a slider is attached to the game object with a green foreground and a red background so it appears that the health bar gets lower whenever the sheep would lose health.

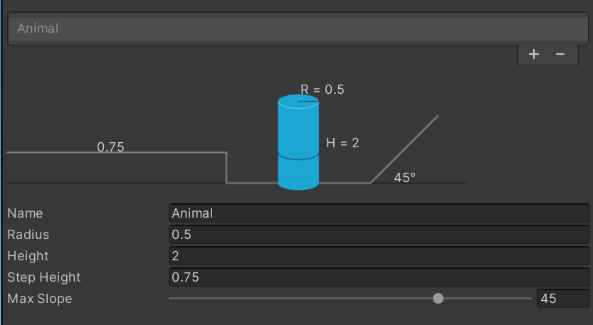
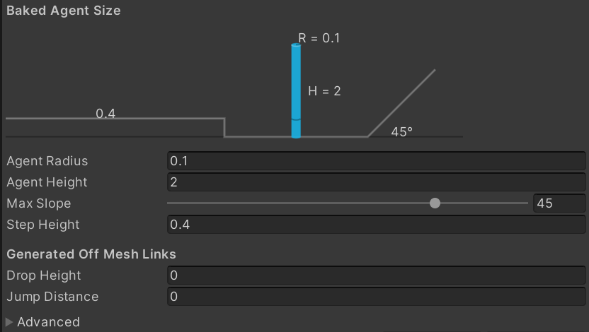
Camera movement in this prototype is similar to Equilinox’s camera movement where you can pan around the area to get different angles and perspectives and also zoom in and out a reasonable amount so you can get a closer look at all your entities or see the entire world all at once.



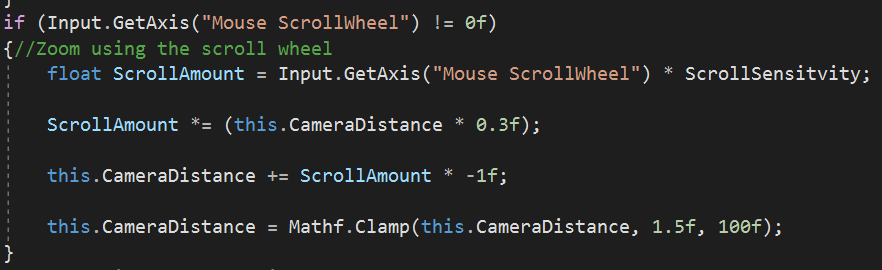
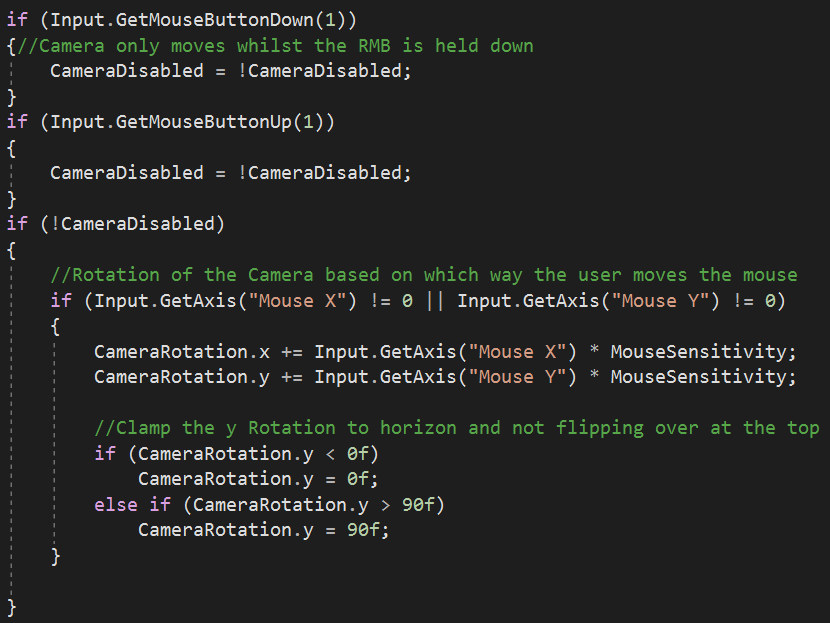
Whenever a sheep or a piece of grass would die, a particle effect will play at their position to make it easier to understand what happened instead of having the objects just dissapear into thin air. Sheep will dissipate into a cloud of white and grass will shoot out smaller and older versions of the grass at different angles.

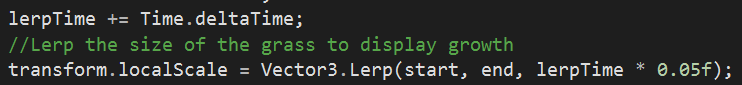
UI is placed on a canvas as an overlay of the screen space with buttons that allow you to select what you want to place on the plane or show and hide certain UI like the sheep’s health bar, instructions for the game or changing the skybox to make it appear that it is night time.

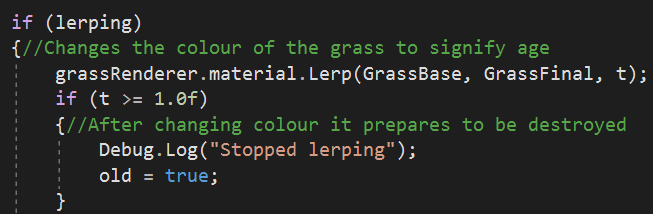
When hovering over a button a tooltip is displayed to assist the user with figuring out what each button does.

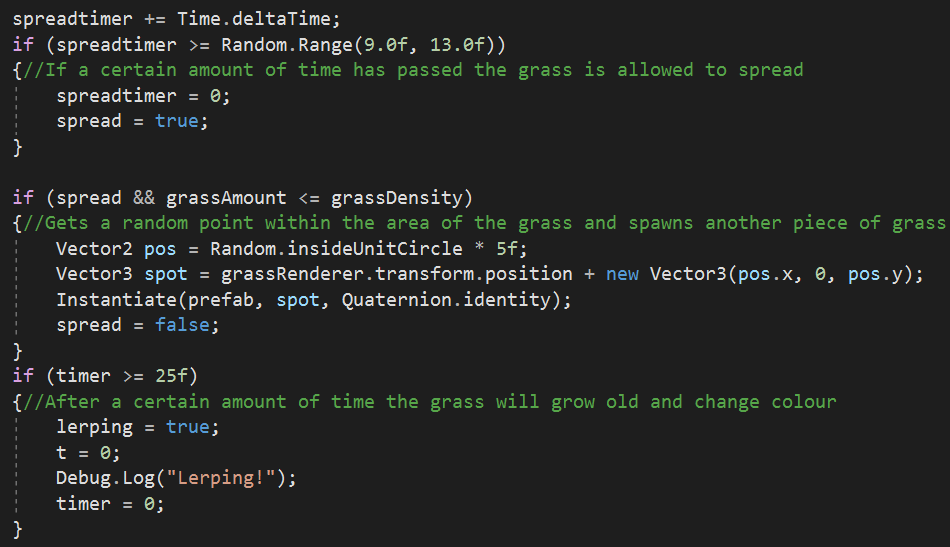
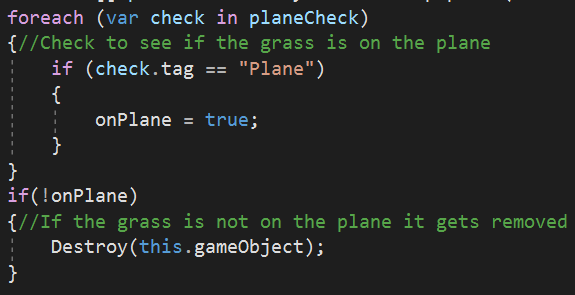
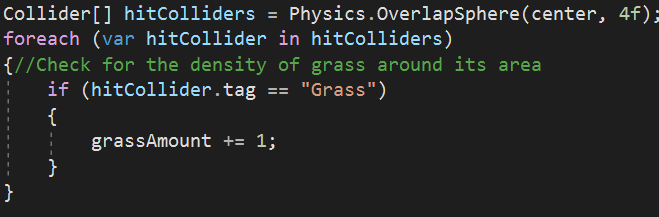
A navmesh is built onto the plane to prevent the sheep from moving outside of the plane and can be further developed to walk around bodies of water or up and down altered terrain.

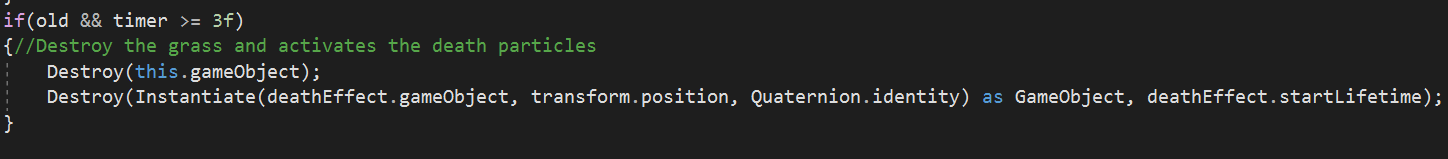
## 1.3 Implementation

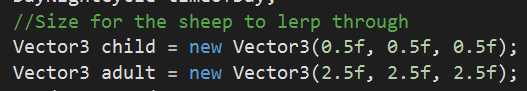
Camera movement is centred around swivelling around the plane by right-clicking and dragging in the opposite direction that you would like the camera to go around. The y-axis also needs to be restricted so the camera does not flip over or go too far under the plane. To bring the camera closer or further away, the distance it needs to travel is calculated by getting the amount of scrolling the user inputted and the amount of sensitivity being applied.

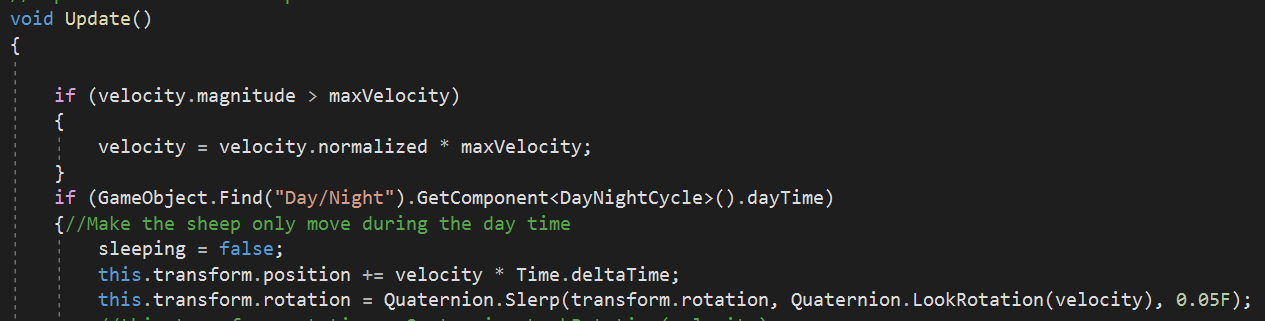
To show the growth of grass, the grass prefab is scaled down at the beginning and once it starts to update it runs a lerp from its current size to the desired size. Lerp is used instead of incrementing the scale every update because update is only called once per frame so the size will only be updated every frame whereas with lerp the size will gradually increase over time.

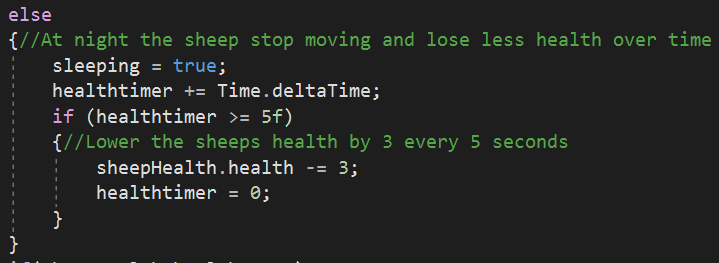
The material of the grass is also done in the same way. Once the grass is old enough it will start to lerp from its base material to a darker material. When first coding the change in colour it would only be able to lerp between two colours and the range of colours available were very limited so instead duplicating the material that was already on the grass and changing it to a darker colour provided a better visual representation.

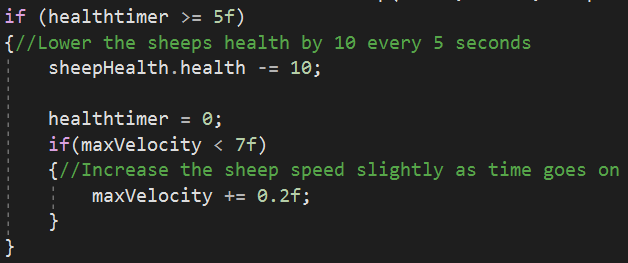
Getting grass to spread first requires a check to make sure that the grass object is physically on the plane to make sure that grass does not grow outside of the map. Afterwards it runs a test every update to check how much grass is already positioned within its radius so that too much grass is not spread in a certain area and is instead more spread out. Since the placements of spreading is random, moments where a bunch of grass will only spread very close to each other happened frequently. Once all the requirements are met it can search for a random position within its radius and instantiate a grass prefab at that location. At the beginning of the project it was made so that the grass will spawn on a random position on the plane but this method gives the user more control over where the general area of the grass should grow and spread.

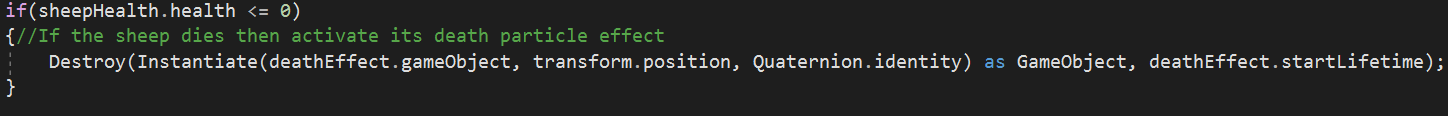
When a grass object would die it will instantiate a particle effect at its position and destroy it right after.

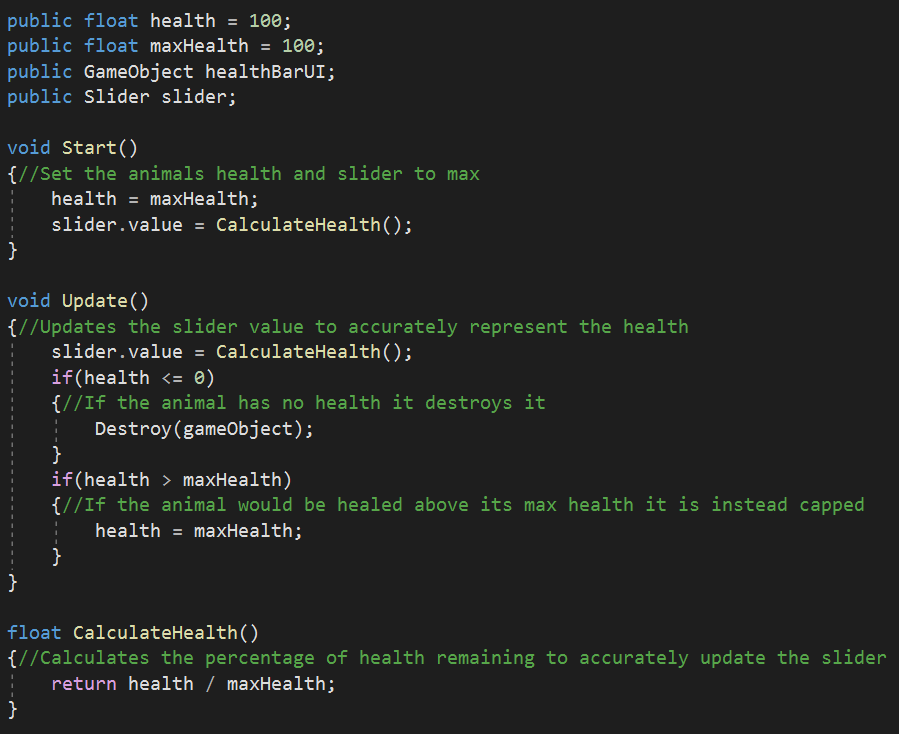
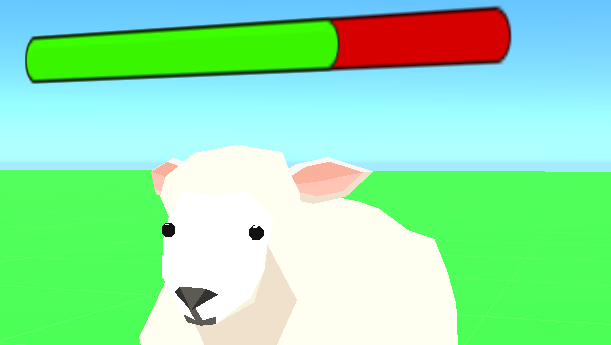
Like the grass prefab, the sheep prefab also starts at a smaller scale that then grows over time. However, the sheep need to move around. At first this was done in the same way the grass would spread, it finds a random position within its radius and it moves to that location but since implementing boids it was changed so that a forward and max velocity is set so the sheep will always move forward at a certain speed as it would be awkward to have sheep move together but go in random directions. Sheep also always follow a leader, so it makes sense to have them all move forward. *“Sheep will follow a leader. If you can get one sheep moving, then the rest will most likely follow.” – (Goodling, 2018).* The rotation of the prefab needs to always match where the sheep is facing, only looking forward when moving forward. Lerp gives it a smooth rotation instead of quickly snapping to one direction.



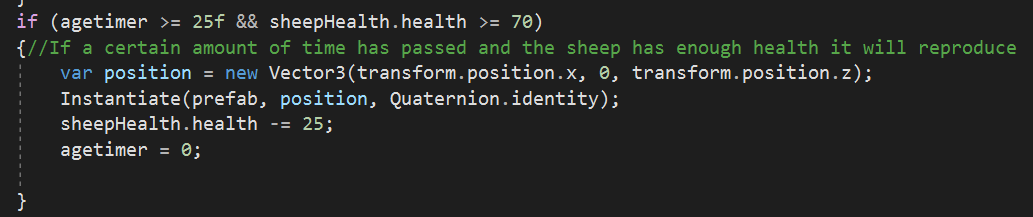
During the sheep’s lifespan it will lose 10 health every 5 seconds by calling the health variable from the health script and slightly increase its max velocity as it gets more eager to find food. Unless it is night-time when the sheep stop moving and only lose 3 health.

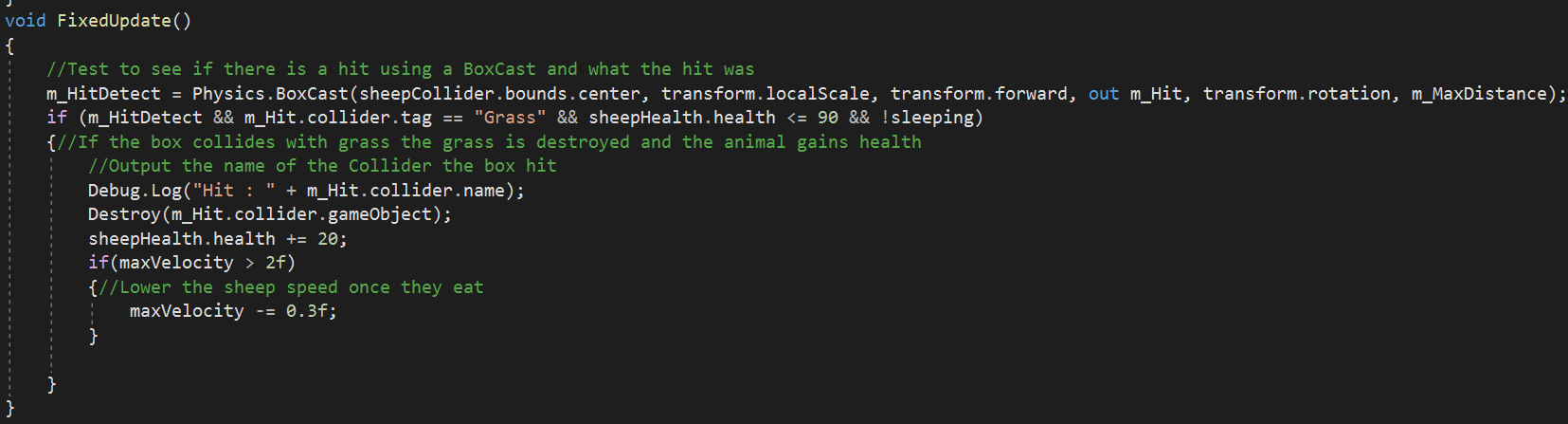


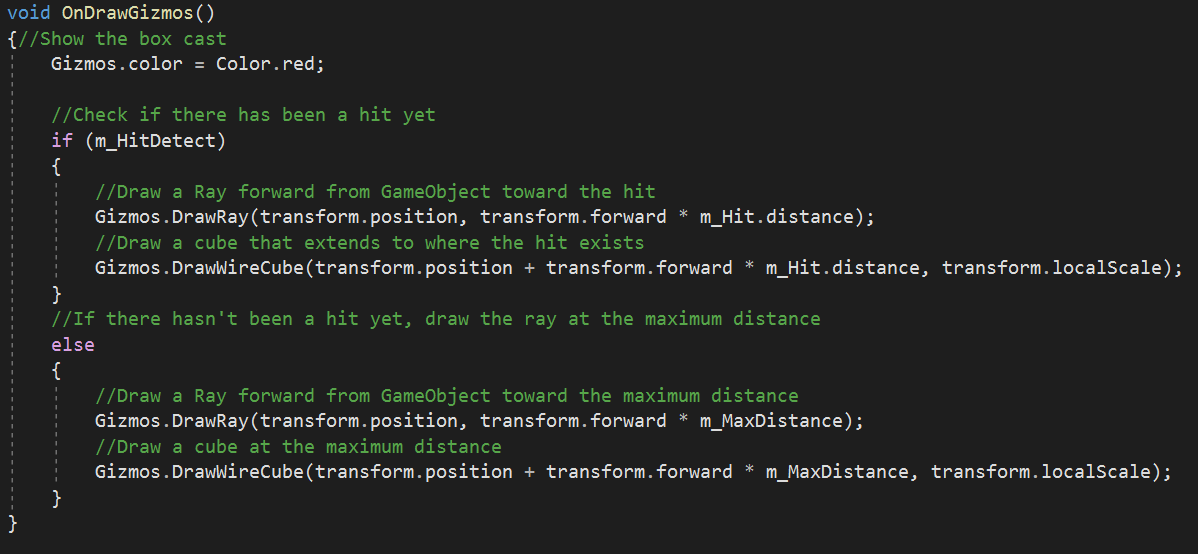
Sheep will too instantiate and destory a particle effect once its health reaches 0.

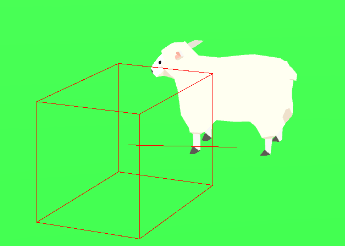
Representing the sheep’s health as a slider is done by calculating how much health the sheep currently has and dividing it by its max health. If a sheep has 70 health, it will return 0.7 and that is the value being passed to the slider. When the sheep’s health reaches 0 it will destroy the sheep object and if the sheep’s health ever goes over the 100-health limit it gets corrected as to not surpass the cap.

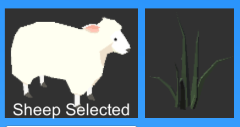
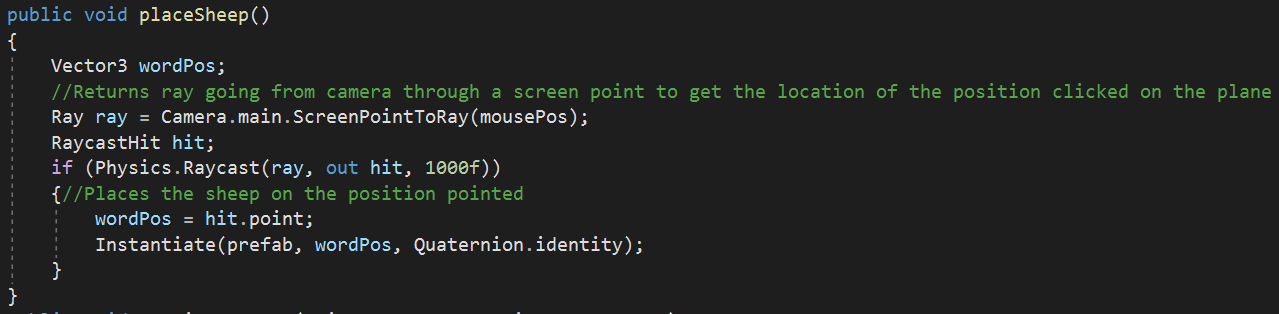


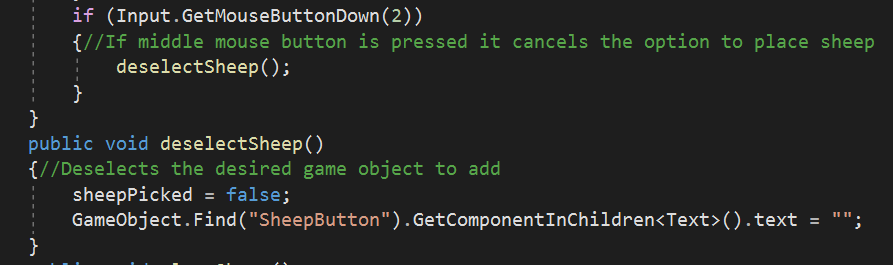
Reproduction takes place once the sheep has a certain amount of health and a specific amount of time has passed. A sheep prefab is instantiated at the current sheep’s location and 25 health is lost. This helps control the population of sheep, so the game is not overloaded with large amounts of sheep being instantiated all the time. Utilising some of the functionality that the grass uses to check how much grass is surrounding it, the sheep could be further developed to check if there is another sheep is nearby to reproduce and even assign genders .

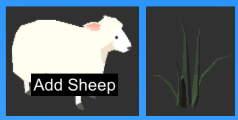
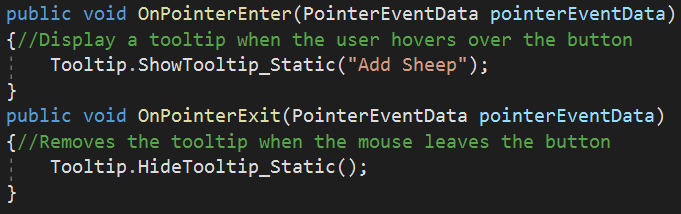
Gaining health utilises a box cast located directly in front of the sheep that checks the tag of the object it is colliding with and if that tag is a grass tag then the grass object is destroyed and the sheep gains health. This has proven to be more effective than using colliders since some grass objects could be accidentally destroyed if a sheep somehow happened to come into contact with a grass object using its body and giving the object a grass tag makes sure that only grass is eaten if there is a collision.

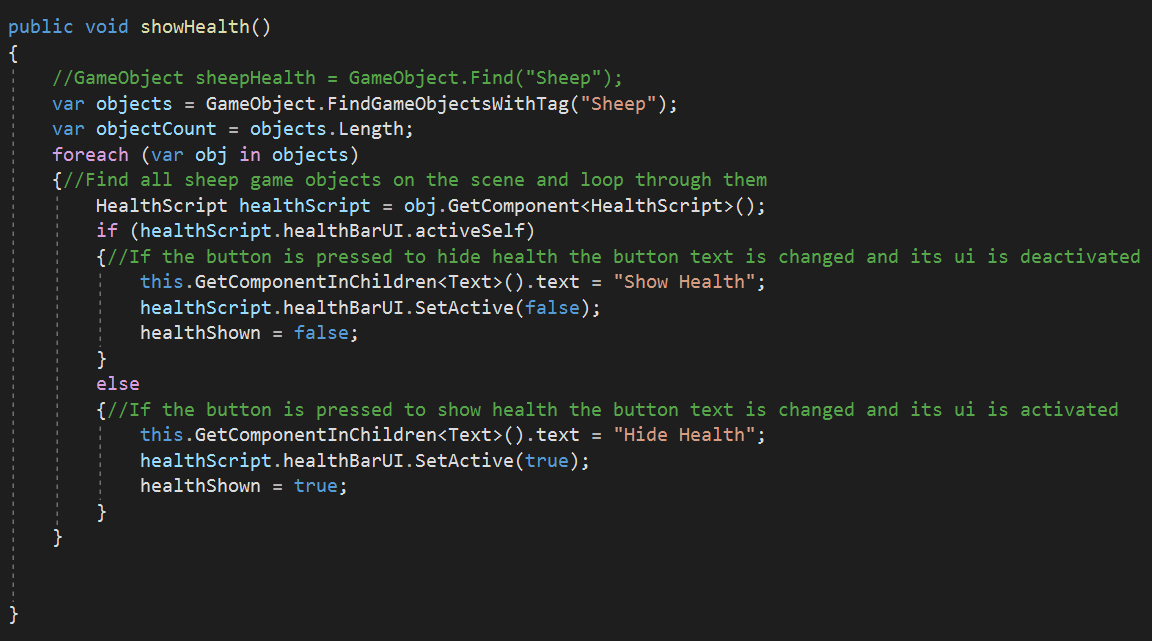
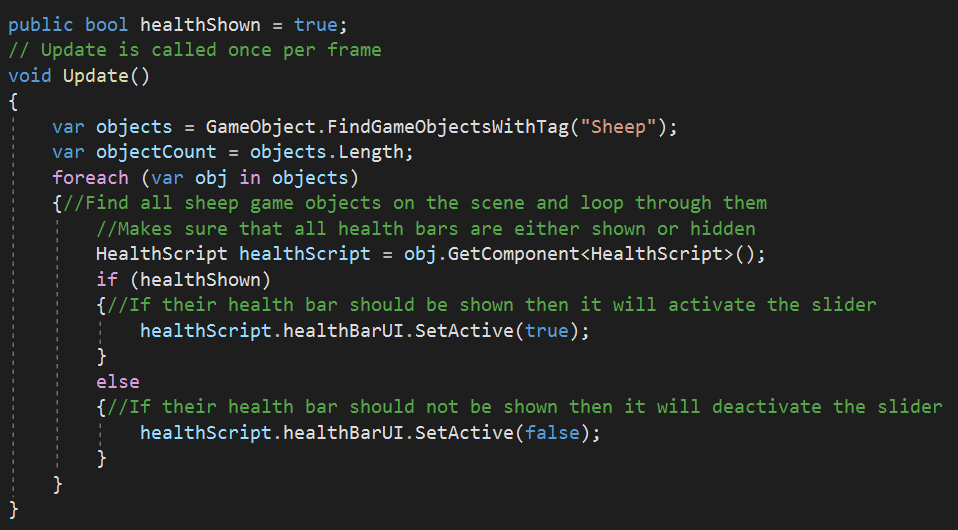
To illustrate this the box cast takes the objects position and draws the gizmo directly in front at maximum distance. Encountering a grass object draws a new ray and cube stopping at the object being hit.

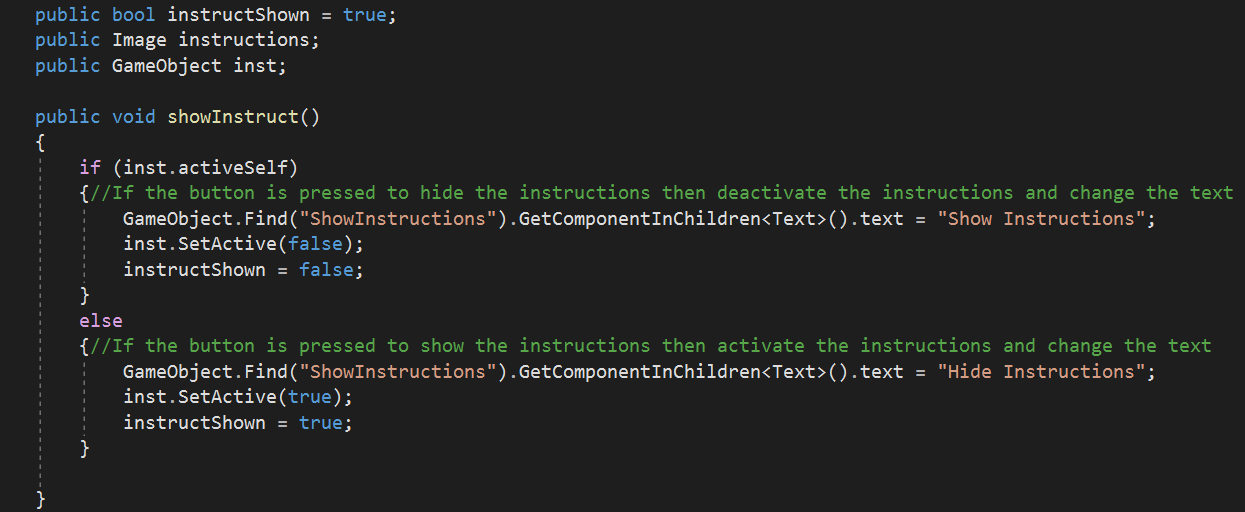


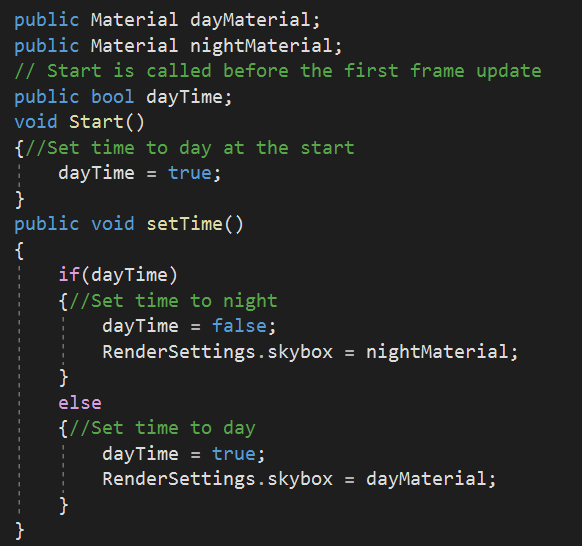
Placing sheep or grass on the plane is achieved by pressing one of the two buttons on the top left of the screen, once pressed a small piece of text will appear on the button describing what occurred and the user is able to left-click anywhere on the plane and place the object chosen. A ray is drawn from the camera to the location of where the mouse-click occurred and if it hit the plane then an object is instantiated at that location.

Pressing the middle mouse button (scroll wheel) or selecting another object will cancel any selection currently in place, preventing the ability to instantiate multiple different objects at one location.

Hovering over a button will display a tooltip of what that button does. An image and text object are located on the canvas and follows the user’s mouse but stays hidden until an OnPointerEnter function is called and the tooltip text is replaced corresponding to the button being hovered. If the mouse enters the sheep button’s area it calls a function that passes that information to the script that manages the tooltip and changes the text. In addition to hiding the game object when the mouse leaves the button.

Underneath those two buttons are another set of buttons that give the user the option to show or hide specific UI. Hiding the health requires the script to loop through every game object in the scene with a “sheep” tag, deactivates the slider component and changes the text. During the update the script also needs to loop through every sheep object to make sure that if the user has selected to hide the health bars then if a new sheep was to be placed or spawned then their health bars will need to be deactivated upon instantiation. Looping through all the game objects every update and every time the health is shown or hidden may not be the most efficient method but in terms of functionality it works and will need to be changed if implementing multiple different animals and on a larger plane to reduce the workload required for the game.

Hiding instructions is a lot simpler with just deactivating the image and object with the text.



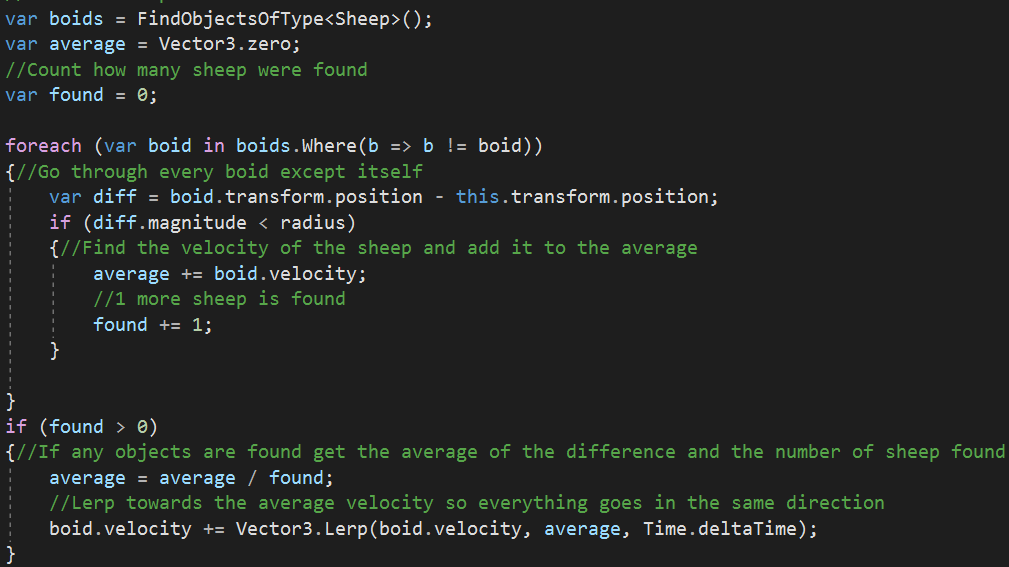
Another part of the scene that can be changed is the skybox. Starting the game defaults the time to day so the main functionality of the game can be explored nevertheless the option to set the time to night is included, forcing the sheep to sleep so they stop moving, losing less health, and give the grass time to grow and spread providing a substantial amount of food for the sheep when they wake up in the morning. Materials for the day and night skybox is downloaded from Unity’s asset store with the potential to include mornings and sunsets. Pressing the button changes the skybox to the alternate material provided.

Further development can ensue changing the day/night cycles to be automatic and perhaps increase the growth and spread of grass during the early morning hours of the day. “*A less-known fact is that grass actually grows during the early hours of each day, at dawn. Pretty nocturnal, the grass takes the energy it receives during the day and processes it during the night. The full process is usually completed at dawn, so to answer your question, yes. Grass does, indeed, grow at night! Or at least it doesn’t stop the process.” – (Holmes, 2021).*

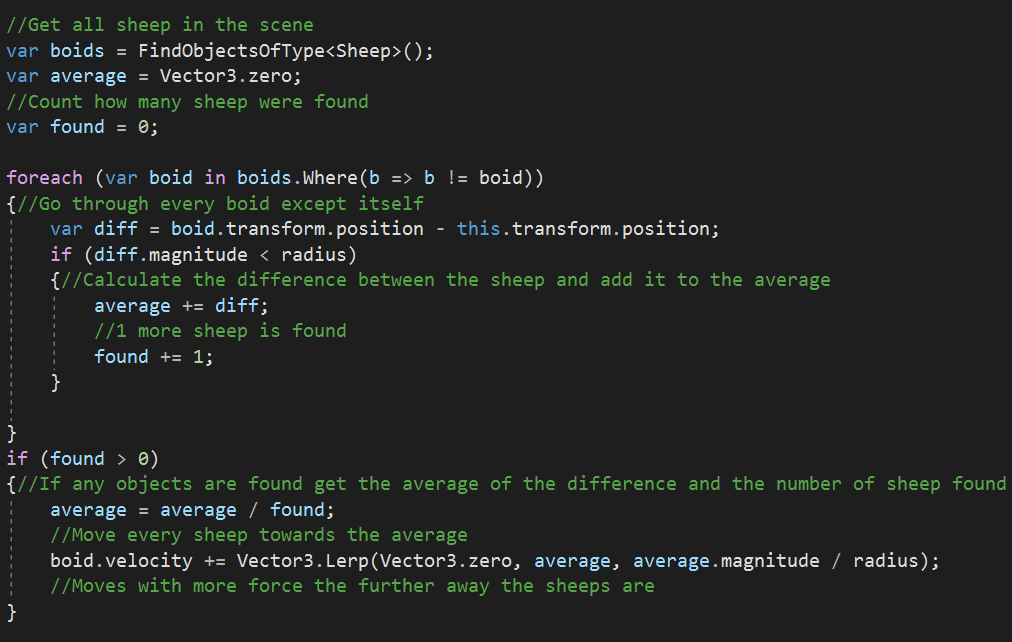
Equilibrium is a fundamental part to preventing the game being too one-sided. Sheep velocity is increased when their health is lowered to help bring them to the front of the herd and maintain an average health pool throughout the herd. In addition, when a sheep would eat a piece of grass then their velocity is lowered. Increasing and decreasing the velocity of the sheep will keep them healthy enough to reproduce which helps clear more grass from the area allowing more grass to grow and spread. To counteract this sheep also lose a percentage of health when reproducing thus avoiding overloading the game with an excessive amount of sheep. Grass follows this construct by restricting how many grass is able to grow within a certain radius and randomising intervals between spread cycles so there is always a guarantee that a piece of grass will instantiate another piece of grass with the possibility to instantiate an extra one during its lifespan. Dead grass is required to be removed as excessive amounts will cause the game to lag. Further development could result in giving the sheep different cycles of grazing and wandering similar to the day/night cycle.

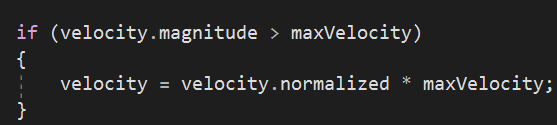
Sheep have a strong flocking instinct so it is important that during the game all the sheep move together as a herd. This is possible with the implementation of boids. Boids are an integral part of the functionality and are seperated into four functions: Alignment, Cohesion, Seperation and Containment.

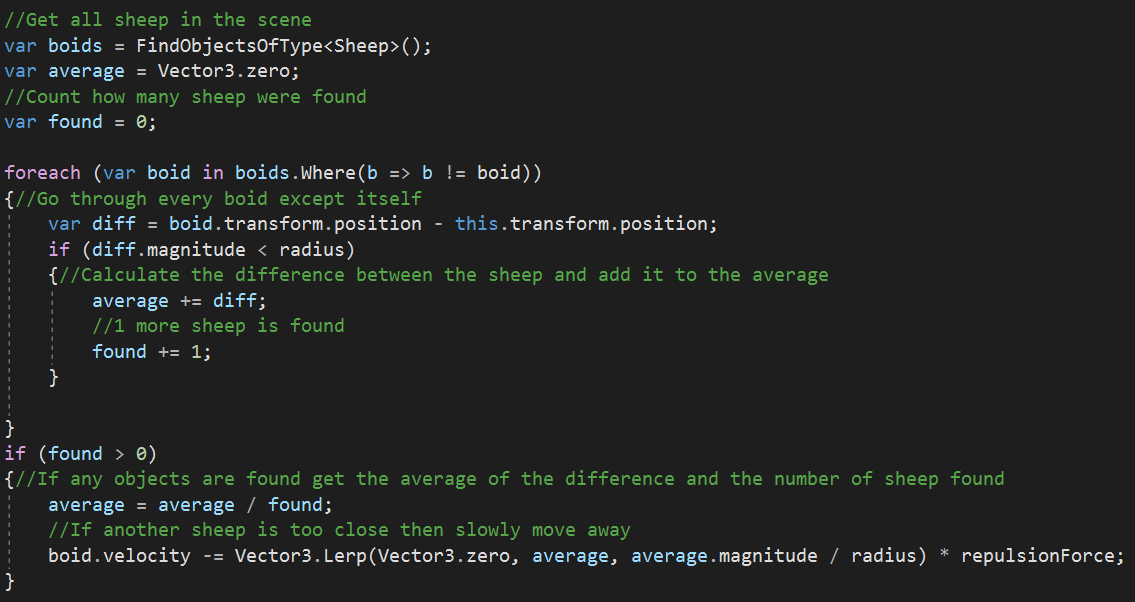
Firstly, Alignment has the sheep try to change their position so it matches with the average alignmenmt of nearby sheep. Looping through each sheep object in the scene except for itself lets the script calculate the difference in position between each sheep found within a radius and itself. If there is a sheep within its radius, that sheeps velocity is accumulated to the average variable and a count of the number of sheep is recorded. If there is more than one sheep nearby the average is divided by the total number of sheep nearby and that average is used as the end location the sheep needs to travel to. The sheeps velocity is increased until it matches the average velocity of all the nearby sheep which makes all the nearby sheep match each others velocity, resulting in all of them moving in the same direction.

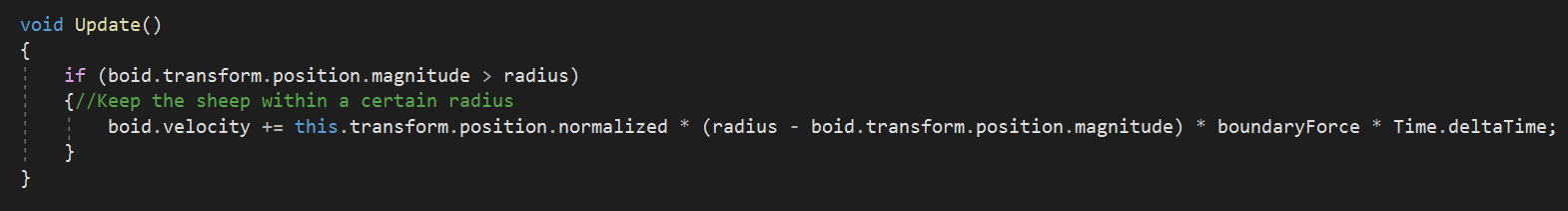


Secondly, Cohesion causes all the sheep to move towards the general area of other nearby sheep. Like with Alignment the script needs to loop through all the nearby sheep in the scene to retrieve their location. Once the average of every nearby sheep is found, their position will slowly match the average position. Using a vector of zero as the starting value and dividing the average magnitude by the radius means there is little change if close to the average since the number is so small and more change further away as the radius is the max distance the velocity is higher and move faster to that position.



As this would cause the sheep to accelerate into infinity, the velocity is limited in the sheep’s script with a clamp that lowers the velocity once it reaches its max velocity.

Thirdly, Separation prevents overcrowding by maintaining a reasonable distance between the other sheep. In comparison, the values used in the lerp is the same as Cohesion except instead of incrementing the velocity, it needs to decrement since we want to move the sheep away from each other. The repulsion force variable is the amount of force that each sheep will move away from each other when getting too close. Nevertheless, sheep do not like being too far away from other sheep, so the value is relatively low to maintain the realism but see each sheep clearly.

Lastly, Containment is not a rule of boids but will also prove useful during further development. If a sheep was to move outside of the specified radius a negative value is calculated by subtracting the radius from the sheep’s position and that negative value is multiplied by the sheep’s direction to move the sheep back to the centre. This is useful for further expanding the plane to keep the sheep, or other animals, localised within a certain area and not travel infinitely in a straight line.

## 1.4 Evaluation

Testing centred around using the black box methodology as the description of the project requires a prototype showing basic mechanics in action and a relaxing sandbox game with emergent gameplay, black box testing hides the internal structure of the game and is a functional test of the software which is the most important part of the project. Using the agile methodology decreased the amount of time available for testing and the already short timeframe resulted in black box being the only option as it is less time consuming than white-box testing and does not require any knowledge of programming so testers can be found quicker.

Testers were asked three questions: How they feel about the overall impressions of the game, what they would change and what they would like to see implemented. These questions provide important feedback with not only the current functionality of the prototype but how the prototype can be developed further, aiding with the extensible requirement of the project. Participants were selected based on their relation to video games and their individual disparity between the knowledge of coding and development offered different perspectives.

Participant #1

1. “Simplistic with some neat flair, like when the sheep vanished: it felt like their souls went up into the virtual abyss.”
2. “Maybe some more wildlife like trees, bushes etc. Would like to see some clouds and possibly even some other farm animals.”
3. “A bit more of an open world with control over more than just grass and sheep.”

When asked to further elaborate on the last question they mentioned giving more personality to the animals. Like how in Equilinox all the animals have names that can be changed and different sizes and genders. Since all the base functionality for adding grass is already implemented adding other wildlife is a lot more manageable. Wildlife and other animals can easily be implemented thanks to the factory method pattern.

Participant #2

1. “The overall feel is smooth. Missing a little bit of animation for walking sheep’s (probably will be implemented in the future). The scroll and camera angle control is very well implemented.”
2. “Nothing to change, the simulator gives an overall good feel. There will probably be more content to come in the future.”
3. “Timer / slower growing processes / limit on number of sheep / predators / water source / terrain / shadows.”

This feedback is one I heavily agree with bearing in mind that things they would like to see are already ideas for future development.

Participant #3

1. “I think overall it feels good. I think it could do with some explaining about what the bars above the sheep's heads are but the sheep themselves function well. I like that they move as a herd.”
2. “I would change the map to have different terrain and bodies of water.”
3. “I'd just like to see more animals with separate behaviours.”

Overall, there was not a heavy enough influence on the relaxing aspect of the game which is something that needs to be considered in future development. Nevertheless, participants were satisfied with the main functionality of the game and are excited to witness its progress.

## 1.5 Conclusion

This report has showcased the creation of a prototype for a nature simulation sandbox game. This project has brought to light the different steps required to successfully create a prototype that features relaxing and emergent gameplay, is extensible enough to be easily developed further and uses data driven decision making. Using an agile methodology proved to be successful in conjunction with the time constraints set as the sprints allowed for frequent meetings and feedback on which direction to take the prototype in. Utilising the factory design pattern throughout the development of the prototype allowed the creation of scripts that in a way that did not need to be defined by a class so most of the scripts can be used for similar entities. For example, the script that grows and spreads grass can be used for a flower or a tree. Emergent gameplay was achieved with the implementation of boids. Sheep will follow a simple set of rules that work together to deliver a situation as complex as herd behaviour. As a result of feedback from the black box testing a relaxing environment was not prominent enough to be considered a factor which is something to be looked into for future development.

A comprehensible graphical user interface helped users understand each of the core mechanics of the prototype and certain aspects of gameplay were managed to maintain the equilibrium between plants and animals by limiting how much grass can spread during its lifespan and only allow a certain density of grass within a radius and figuring out a balance between how much health a sheep would get when eating and how much health it should lose over time. This equilibrium helped keep the game optimised enough so it is not overloaded with grass or sheep but not scarcely populated to provide acceptable gameplay.

The entire project was very data driven as reasonable amounts of research occurred determining the characteristics of wildlife and how to script certain behaviour to apply them to an object. Since Equilinox was the inspiration for the prototype, watching gameplay helped understand what was required from the project and how particular mechanics should word. More research is required to include more types of wildlife and more advanced behaviour and interactions.

This project was very important in helping understand more about video game design and development and gave a lot of practical experience that I would not have been able to undergo otherwise. Helping solidify my goal of becoming a video game developer.

## 1.6 Bibliography

Dawkins, R., 2006. *The selfish gene*. 3rd ed.

Goodling, K., 2018. *Five Things You Should Know About Sheep Behavior*. [online] Living with Gotlands. Available at: <https://www.livingwithgotlands.com/2018/07/five-things-you-should-know-about-sheep-behavior/?cn-reloaded=1> [Accessed 3 May 2021].

Holmes, J., 2021. *Does Grass Grow At Night? - Planted Shack*. [online] Planted Shack. Available at: <https://www.plantedshack.com/does-grass-grow-at-night/> [Accessed 4 May 2021].

# Appendix A

## Feasibility Study

**Project Title**: A Nature Simulation Sandbox Game

**Student**: Nesta Chukwuma Rubio 18061418

**Supervisor**: Huw Lloyd

**Ethics Reference Number:** 26058

### Course-Specific Learning Outcomes

* Use knowledge, abilities and skills for further study and for a range of employment in areas related to scientific and technical computing.
* Interpret legislation appropriate to computer professionals and be aware of relevant ethical issues and the role of professional bodies.
* Analyse, design, and implement algorithms using a range of appropriate languages and/or methodologies.
* Demonstrate effective communication, decision making and creative problem-solving skills, and identify appropriate practices within a professional, legal and ethical framework.
* Critically appraise and apply suitable artificial intelligence techniques for a variety of software systems.

### Project Background

This project is about creating a prototype for a nature simulation sandbox game. These types of games have already been made before, some examples being Equilinox which is based on growing an ecosystem and Spore which is about genetics and survival. Since the main purpose of this project is to create a relaxing sandbox game with emergent gameplay, a fair amount of research is dedicated to learning how to develop a game and an extensive amount of research will need to be dedicated to life and nature.

Last year there was a presentation at our university by two video game developers, Andrew and Philip Oliver, since I was interested in game development, I attended the presentation. There they talked about all the aspects of game development that they went through all their years and as a person interested in game development it helped understand more clearly what I wanted to do so when I had the opportunity to make a game as a final year project, I was very excited. My goal before the deadline is to create a relaxing and entertaining game using a 3D real-time creating program like unity or unreal engine. The game should be well optimised so users on different systems can experience it too.

### Aim

The aim of the project is to create a prototype for a nature simulation game which shows that it is functional and can provide basic mechanics in action. A significant amount of research is needed in both game development and nature in order to complete this project. Hopefully I should have a working prototype a few months before the deadline, so I have enough time to work on some additional features as I am heavily invested in the project and I want to see it more developed. However, the project would realistically just be a prototype with the focus being creating a working model.

### Objectives

To be able to fully comprehend the requirements of the project there will need to be certain research objectives set.

* Adequate research for the important areas in the project needs to be performed. For game development make it extensible so it can be further developer, for specifics it would be to research what aspects of nature I want to implement and how much of a focus it should be to the game
* Learn how to create a video game using books and online tutorials and use suitable game development programs. For Example, Unity or Unreal Engine
* Research into how current existing games like Equilinox were made to help further understand my goal and possibly give inspirations to new ideas I could implement in my own game
* Use design patterns that can be correctly implemented into the project like using factory method and flyweight to create npc’s and store shared features. Using the book provided by my supervisor (Game Programming Patterns) is very helpful for this objective
* Create a modern and easily comprehensible Graphical User Interface. A good interface is key to a smooth experience
* Ensure that the game is well optimised for suitable use on various machines
* Have a working prototype that can be used by people to provide relevant feedback which will be useful for future development and to help improve as a game developer for any future games I make

### Problems

Some issues that could arise from doing a project like this are as follows.

* Data loss – A loss of data could occur randomly due to any type of hardware failure, but this can be prevented by regularly uploading to a cloud service or using Git Hub
* Time management – balancing the project with other units might be tough and with the new block system implemented by the university it puts a lot of pressure on needing to complete the project within a specific time frame which could also affect the testing portion
* Covid-19 – The pandemic that is currently affecting the world could cause several issues such as forcing me to self-isolate or it could force a local lockdown which would mean that the testing would need to be done online
* Third-party testing – sending the game will most likely be difficult due to a large file size so compression and good optimisation will be necessary. Which will also help for testers who do not have strong hardware

### Required Resources

* Unity or Unreal Engine
* Internet
* Computer
* Microsoft Word
* GitHub/Git
* Game Programming Patterns book

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tasks | Weeks | | | | | | | | | | | | | | | | | | | | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Research |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Unity/Unreal Engine tutorials |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Creating game |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Brainstorming ideas and concepts |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Interim report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design and development map |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Development of game |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Creation of assets |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Animations and sound effects |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing and evaluation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Final report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

### Schedule

### References

* <https://unity.com/>
* <https://www.unrealengine.com/en-US/?sessionInvalidated=true>
* <https://www.equilinox.com/>
* <http://www.gameprogrammingpatterns.com/>

# Appendix B

## Showcase Presentation

Background pattern

Description automatically generatedGraphical user interface, application, website

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated