**System Development**

4.1. Introduction

In this chapter, a detailed analysis of the system development will be given, but first the original prototype will be discussed. There are many aspects of the system development that will be discussed such as the procedural generation, the UI, the inspection system, the quiz feature, the models and animation, the audio, and the AI.

**4.2 System Development**

Overall, this project has around 47 different custom scripts written in C#. Since there’s a large quantity of scripts, not all of them will be described in detail. The important scripts will be discussed that are relied on by the important features of the system and the low importance scripts will have brief descriptions of what role they play in the game.

**4.2. World Generation**

**Procedural generation has a large role in the creation of the environment of this game. The terrain that the player walks on, the mountains in the background, the grass that’s being created and the forest that is created are all generated through code using procedural generation and a technique called Perlin noise as discussed in previous sections. Each of these aspects will be discussed in detail as they make up a large and important part of the system. Overall, There are about 11 different scripts involved in the procedural generation, below are the aspects that are most important.**

**Terrain Generation**

**First, the terrain that is generated in which the player uses to manoeuvre throughout the environment will be discussed.**

**The first step in creating this was to create an object in Unity in which the script for generating the terrain could be attached to, this object was renamed to Terrain. Next, a C# script was created and attached to the object.**

**This script is extremely important because it is connected to both the forest generation and the grass generation. The way in which the terrain is generated is by using a pseudo random pattern called Perlin noise. Variables like the scale, the heights, and the size of the terrain were created and used to generate it. By using this technique, it allows the script to generate a random terrain at different heights. Meaning once the game starts the terrain will be generated randomly through this code at different heights and positions than before.**

**This Scripts is quite long so only some snippets will be shown. Below are all the variables that were set that were used to generate the meshes.**

**Text

Description automatically generated**

**Below are some of the functions used to generate the terrain and pass in the variables to generate the different heights using Perlin Noise.**

**Text

Description automatically generated**

**Generate() Function generating Vertices, triangles, and vectors:**

**Text

Description automatically generated**

Once the Script was done the terrain size, scale and height multiplier were able to be changed in the inspector as shown below. Also, a reference for the forest and grass generator scripts were set.

A screenshot of a computer

Description automatically generated with medium confidence

Below is an image of the terrain being generated:

Chart, surface chart

Description automatically generated

**Forest generation**

**Next, the procedurally generated forest was created. this script uses the Generated terrain script to allow it to be placed on the terrain at the correct heights and be the corrected size. It does this by using the 2D heightmap array set in the terrain scripts.**

**First the forest game object was created and the forest script was attached to it. This script generates the trees by using arrays to pass in tree objects and randomly placing them throughout the terrain once the game starts. Every time the user plays the game the forest is generate and different each time. Multiple vectors were created with random range values so that the forest would look as realistic and less procedural as possible.**

**Below is a snippet of the forest generater class and the Generate function in it:**

**Text

Description automatically generated**

**In the inspector, the tree objects were able to be added so they could be generated and the forest size and spacing was set.**

**Graphical user interface, application

Description automatically generated**

**Below is the forest that was generated on the terrain:**

**Map

Description automatically generated with medium confidence**

**Grass Generation**

Next Up was the Grass Generation. This was also linked to the terrain script. This feature uses its own script just like the forest script. It is quite similar to the forest generation script with a few minor changes so it won’t be discussed in too much detail. As you can see in previous images the grass script is referenced in the terrain object. Below is an image of the grass object inspector where the object is added and the size and spacing are set:

Graphical user interface, application

Description automatically generated

Overall, there are Four scripts that are used to generate the grass in different areas of the game. But they all follow a similar pattern so there is no need to show all of them. Below is an image of the grass generated on the terrain in the forest:

A grassy area with trees in the background

Description automatically generated with medium confidence

**Mountains Generation**

**Finally, the Procedurally generated mountains were created. These were also created using Perlin noise but at a different approach from the terrain generation. This was because I wanted them to look more like mountains and to be a little bit more customizable and random. A game object was created and a script to generate the mountains was created. this was quite a long script so only small parts of it will be shown. One of the big differences to how the terrain was generated is this script uses Octaves, a height curve, lacunarity and a seed to generate a mesh more similar to mountains. The seed is basically a number that is used to generate unique set of mountains each time.**

**Bare just two of the functions in this script, one for the seed function and the other for using the octaves and Perlin noise to generate. There are many more functions in this class but these are quite important.**

**Text

Description automatically generated**

**Below is the an image of the inspector where the attributes of the Mountains could be changed such as the height curve, size, scale, seed and other attributes.**

**A screenshot of a computer

Description automatically generated with medium confidence**

**The images below show an example of the mountains being generated. A few different examples were taken form the game to show that each time the user plays the game a unique set of mountains would be generated.**

**Chart

Description automatically generated**

**As you can see the mountains are different each time.**

**Animal Generation**

**Another area of this project where procedural generation was used was when creating and spawning in the animals throughout the game.**

**Four different scripts were used to specifiy the generation behaviour for each of the animals. This is because some animals have different traits and behaviours so the scripts allowed random amounts and types of animasl to be generated throughout the map. For example some for the foxes, the scripts allowed a random number of them to be generated each tim ethe game is played each with different behaviours, these behaviours will be discussed in more detail in the AI section.**

**Objects were created to act as spawners/genrators for the animals. Each spawner had the script for the animal in wehich it was generating along with the animal itself.**

**Below is an example of one of the animal generator scripots to generate the foxes.**

**Text

Description automatically generated**

**Below is an example of the object and inspector of the fox spawner:**

**Graphical user interface

Description automatically generated**

**As said before each of these spawners were quite similar in the way they were created and written so only one of them have been discussed. Shown below is an image of the Foxes being Generated at random locations and with different behaviours:**

**\*\*Image\*\***

**Procedural generation**

* ***Terrain Generation***
* ***Forest Generation***
* ***Grass Generation***
* ***mountains***

**Menu Creation**

* ***Main Menu***
* ***Pause Menu***

**Inspection System**

* ***Inspection controller***
* ***Inspection ray cast***
* ***Manager NPC UI***
* ***Object controller***

**Quiz**

* ***Question Editor***
* ***Answer drawer***
* ***Answer data***
* ***Audio manager***
* ***Game manager***
* ***UI manager***
* ***Game events***
* ***Question***
* ***Game Utility***

**Models**

* ***Trees***
* ***Rocks***

**Animations**

**Audio**

**Images**

**UI**