**Honours Project Feasibility Demo**

**Callum Myers**

**2102549**

**Research Question**

How can procedural level generation be used to lengthen players’ playtime and improve player engagement in 2D platformer games?

**Aim**To investigate and analyse the benefits of procedural generation algorithms when creating levels for 2D platformers, then consider how this could be used to lengthen the time a player can stay engaged with the game.

**Objectives**To research how procedural level generation could be used in future 2D platformers to give the player a unique experience each time they start a new game.

To evaluate the level and length of player engagement in a classic linear platformer when compared to one featuring procedural level generation techniques.

To explore the links between new, unique levels and player engagement levels.

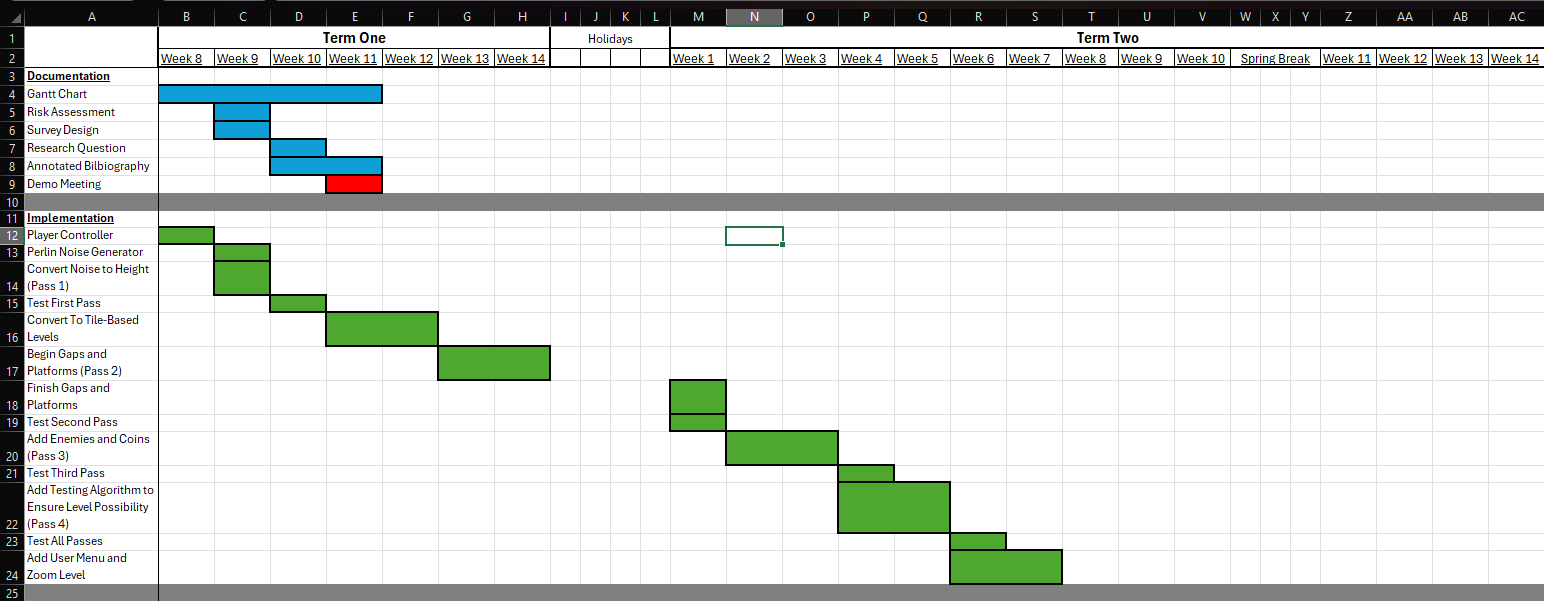
To utilise programming techniques such as Perlin Noise and Markov Chains to implement a procedural level generator using the Unity game engine.

**Statement of Changes**

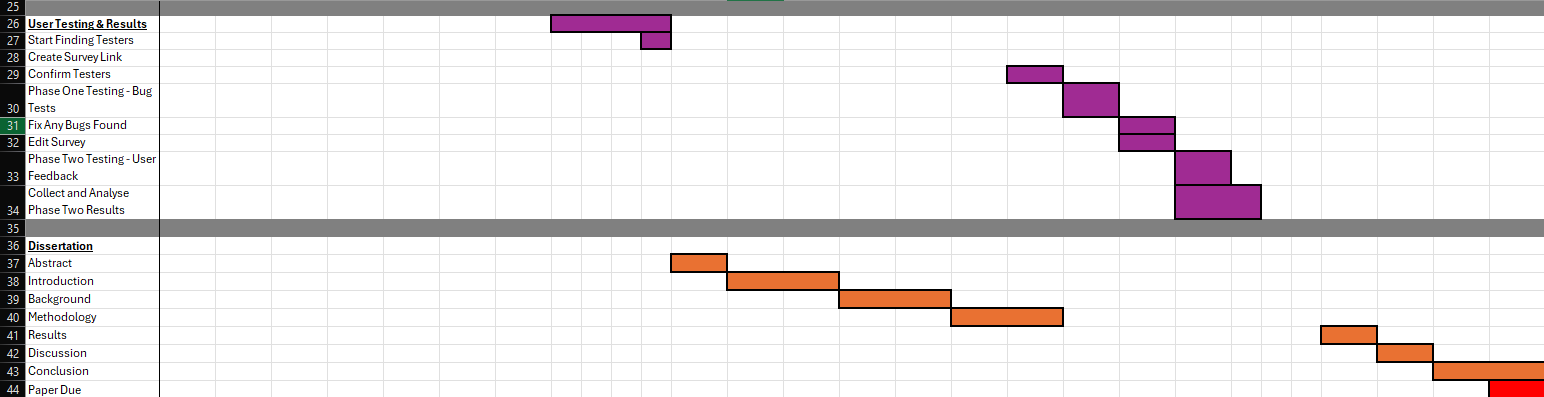
I no longer plan to use Wave Function Collapse in my implementation since it is made mostly obsolete by my use of Markov Chains.

**Gantt Chart**

I have included a file for the Gantt Chart in the .zip folder.







**Risk Analysis**

**Time**

The biggest risk to my project is time. Since the implementation is a new project for me, there are some aspects which I need to learn whilst implementing the main level generator. For example, I have not used tilesets and tilemaps in Unity before however I feel these would be vital to prevent performance issues. To mitigate the risk of running out of time, I will be using my Gantt Chart to ensure I am meeting expected deadlines, as well as reviewing how the progress is going at the end of each week and making changes to the timeline and deadlines if needed. I have also set a priority for each task within the implementation which needs completed and have set some goals to be stretch goals which I will only attempt if I find myself ahead of schedule. The order of priority for primary goals is:

* The generator uses Perlin Noise to generate a new level layout every time the level is restarted.
* Every level layout generated can be complete by the player.
* The generator always completes the first pass (using the Perlin Noise to set the height of the ground at every point in the level) and the fourth pass (the level is tested to ensure it can be complete by the player, and if it is not then it will decide whether to rework certain sections of the level or discard the entire level and create a new one).
* The generator completes the second pass (which will be spawning a variety of enemies at random points through the level) and the third pass (which will be spawning a variety of collectibles throughout the level).

The following are stretch goals which I hope to complete however these could be skipped without any major impact to the base project if the primary goals cannot be completed in time for any reason. The priority for these is:

* The sprites for each section will be varied and show biomes or background variation.
* The player can use the menu to set parameters for the next level which will be generated, as well as being able to generate this new level at any point.
* The player will be able to zoom in and out to see more or less of the level at the same time.

**Testing**

Since my testing will take the form of a user survey, it could cause an issue if I cannot get a suitable number of candidates who are willing to give me feedback. If this issue does arise, I may have to cut some time from my stretch goals to do extensive testing myself. I also plan to mitigate this risk by finding people who are willing to participate earlier on in my project, with the intention that if it does become an issue, and I need to spend more time testing by myself, I will more easily be able to plan for this by working it into my deadline schedule.

**Machine Damage**

Since I will be working primarily from home on my own laptop for this project, one risk I may encounter would be my laptop breaking or corrupting. I have mitigated this risk by ensuring my files are constantly backed up to a GitHub repository. This includes both documentation and implementation work, and if for any reason I do not have access to my computer for a period, I can still access Unity using the computers on campus, and any documentation I need to edit could be done at my local library’s computers.