**Dissertation Literature Review**

**2.1 – Current Use of Procedural Generation In Games**

**2.1.1 – Overview**

The first aim of this project was to research how procedurally generated content is currently being used successfully in games.

Using the Procedural Content Generation Wiki [[Link](http://pcg.wikidot.com/)], three games were highlighted which could provide some valuable insight into the current state of PCG in games:

* Minecraft
* Spelunky
* Infinite Mario

This section will cover each of these games in more detail, highlighting their success, as well as how procedural generation affects the game, and which techniques were used.

**2.1.2 - Minecraft**

Minecraft [Mojang Studio, 2009] is an open-world sandbox game which has become the Guiness World Record holder for the best-selling video game of all time [[Guiness World Records,](https://www.guinnessworldrecords.com/world-records/best-selling-video-game?os=0&ref=app#:~:text=The%20best-selling%20videogame%20of%20all%20time%20is%20Minecraft%2C,sales%20by%20publisher%20Microsoft%20on%2015%20October%202023) 2023]. In 2023, it officially reached over 300 million copies sold, and as of January 2025 still has over 50 million players logging in every day [[Saisuman Revanker](https://www.sci-tech-today.com/stats/minecraft-statistics/), 2025].

With such a huge number of live players almost 16 years after launch, Minecraft must have something to keep players constantly coming back, and a huge part of this can be attributed to its virtually infinite combination of worlds. Each world in Minecraft is generated entirely using procedural generation, through a combination of Perlin Noise, Fractal Noise and 3D noise [[Dawnosaur](https://dawnosaur.substack.com/p/how-minecraft-generates-worlds-you), 2023]. On top of this, similar to how this project’s levels are generated, Minecraft does its world generation across three different passes [[Alan Zucconi](https://www.alanzucconi.com/2022/06/05/minecraft-world-generation/), 2023]. In the first pass, a “biome map” is generated, which dictates how each section of the world is generated. For example, mountain biomes will allow the world to be generated up to a higher altitude, whilst a desert or forest biome may keep the world at a lower altitude. The next pass then covers the biomes in an appropriate material, such as stone for the mountains and sand for the deserts, then grass for the forests. The third pass then goes back over the world, and removes ground from specific areas to create caves or ravines, whilst the fourth and final pass decorates the world with structures.

Through these techniques, the game can generate approximately 18 quintillion unique world combinations [[Minecraft Wiki](https://minecraft.wiki/w/World_generation)]. Overall, Minecraft portrays a great example of how a game can use procedural generation to its advantage and allow players to replay the game multiple times over, causing it to retain popularity many years after release.

**2.1.3 – Spelunky**

**2.1.4 – Infinite Mario**

**2.2. – Procedural Generation Techniques**

**2.2.1 – Overview**

After researching the many procedural generation techniques and algorithms available, Perlin Noise and Markov chains were deemed to be the best options in the context of the project. Before starting the implementation of the level generator, some criteria were laid out to decide what would outline a successful level:

* The terrain of each level should be unique enough that a player could replay the game multiple times without encountering two levels which felt too similar. This can be achieved with effective use of Perlin Noise.
* Each level must have enemies and collectables placed randomly throughout to encourage the player to explore or find alternative routes. This can be achieved using Markov Chains.
* Each level must have at least one path which guarantees the player can reach the end goal and win the level. This can be achieved by settings rules and constraints for the level generator scripts.
* Players should have the ability to edit parameters which will customise how future levels are generated, allowing them to see how the procedural generation can lead to a variety of level types.

**2.2.2 – Perlin Noise**

Perlin Noise is a procedural generation technique which was first introduced and used by Ken Perlin in 1985 [[Ken Perlin](https://dl.acm.org/doi/pdf/10.1145/325165.325247), 1985]. The development for the noise function began because Perlin wanted to find a more efficient way of creating realistic textures to be applied to models in CGI. As Perlin mentioned in his research, many of the procedural generation techniques at the time required combining multiple individual functions, which required rewriting, recompiling and rerunning the program until the desired effect was generated. Perlin found this to be “cumbersome” [Ken Perlin, 1985, Introduction], therefore he created his own program, which would generate a black and white noise map then manipulating the values at each point to correspond to different values on the final texture. Perlin outlines in his paper that he developed functions to convert these values to colours, normal and heights.

Whilst the original intended use of Perlin Noise centred around graphical textures, game developers have since adapted many of these functions to create terrain in their video games. A prominent example of a game which brilliantly utilizes Perlin Noise is Minecraft [Mojang Studios, 2009], and it is fair to assume that much of the game’s success can be attributed to the infinite world generation created partly using this technique. Minecraft’s ground layer is largely generated from a Perlin Noise map, however due to the gradual gradients being one of the characteristics of a Perlin Noise map, the developers did not deem this to create unique enough terrain for an infinite world, therefore they layered multiple maps on top of each other which were combined to make one more significantly unique map [Dawnosaur, 2023]. The height of each block of the world is then scaled according to the grayscale value of the corresponding pixel on the map, like how Perlin created the “Stucco Donut” on his paper [Ken Perlin, 1985, A Pixel Stream Editing Language]. For example, a darker pixel would place blocks up to nearer the upper limit of the world, whereas a white pixel would move towards the lower constraint.

Perlin Noise is a good technique to be used in this project as Perlin designed it to be as computationally efficient as possible. By mapping the height value of points throughout the level to the grayscale value of the noise map, the level generator can effectively adapt Minecraft’s generation technique to be used in the context of a 2D platformer.

A summary of the pros and cons of using Perlin Noise are as follows:  
Pros:

* Perlin Noise was designed with efficiency and simplicity in mind, which will help to lower loading times when levels are being created, as well as allowing small changes to be made which will help achieve desired effects on each level’s terrain.
* Perlin Noise generates softer, more gradual terrain which can be used to create smaller changes in height throughout the level to help prevent impassable walls or slowing down the pace of the level.’
* There has been a significant volume of research surrounding Perlin Noise, and therefore strong documentation, as well as the Unity Editor (being used for the project) including some functions dedicated to getting a Perlin Noise generator started quickly.

Cons:

* Perlin Noise maps may sometimes “cluster”, creating areas of the same colour, which can result in extended areas of flat ground, however this can be negated with secondary passes to add other objects to these areas.
* Perlin Noise maps may start to tile on larger maps, however since the levels in the project will be relatively small, this is not expected to present any issues.

**2.2.3 – Finite Markov Chains**

Markov Chains were originally created by Andrey Markov during his research on the theories of probability in 1906 [Eugene Seneta, 2006].