**Dissertation Introduction**

With a significant influx of new games hitting the market every single day, there is pressure piling on game developers to make their project stand out from the crowd. This often means having to make vast amounts of content for the game to give players the best value for money possible, and this may cause longer development times as well as huge costs for the studios to keep enough staff on the team to manage the increase in workload.

One way this workload could be reduced is using Procedural Content Generation, or PCG. PCG is defined as “the algorithmic creation of game content with limited or indirect user input” (Shaker, N., Togelius, J. and Nelson, M.J., 2016, pp. 30). Using a variety of algorithms and techniques, developers can create a model which can help them to generate entire levels all at once, without the need for level designers to draw and implement each one by hand, significantly cutting the workload down to be more manageable. Creating these algorithms may require some more time dedicated from the programmers, however this could be seen as a longer-term investment, since an appropriate model could be adapted for use in many different games by the team. By simplifying the level generation process, the team will have more time to improve other aspects of the game, such as mechanics or graphics, to ensure the player has an engaging experience throughout their playthroughs.

Another benefit PCG can offer is creating a new, unique experience for the player each time they play. A strong example of this is seen in the 2D platformer; Spelunky (Derek Yu, 2008). Within Spelunky, levels are semi-procedural, meaning they are created dynamically, using presets which are made by level designers prior to the game’s release. This results in the game containing a massive number of possible level layouts, without requiring the level designers to implement them by hand. Because of this, players can play through the game many times, and the likelihood of experiencing the same levels multiple times is much lower, leading to fun, fresh and enticing gameplay each time.

While PCG is great at keeping content feeling unique and fresh every time, one challenge often encountered when using it in game development is the lack of emotion felt throughout the levels. Hollow Knight (Team Cherry, 2017) displays this, such as early in the game where the player can find remains of knights in The Forgotten Crossroads. This helps to tell a story, since they are placed strategically and expertly by the level design team and would not be as effective if they were randomly scattered using a PCG algorithm. This issue can be combated by using a variety of “online” and “offline” generation techniques. Online generation involves generating the content during gameplay, such as the way Minecraft (Mojang, 2009) handles it, where the world is created in small 16x16 sections, or “chunks”, as the player approaches. This method creates a much more randomised generation. Offline involves generating most or all the content all at once, for example when the player first opens the game or even before the game is released. This method could allow developers to create many levels all at once during the development with the algorithm, then allow level designers to go through and add in their own unique aspects, finding that balance between variety and offered by human touch and the unique feel PCG can offer.

**Research Question:**

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

This project aims to answer this question by designing and developing a procedural level generation model in the Unity game engine which will create a series of levels for a 2D platformer game. This will be done using procedural techniques such as Noise Functions, particularly Perlin Noise, and a series of Markov Chains which will create new and unique levels on demand. By creating terrain generated using Perlin Noise, as well as populating the levels with objects using Markov Chain decisions, this will create unique and diverse levels, keeping players engaged and surprised throughout the levels.

**Aim and Objectives:**   
This project aims 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. This will then be tested by several participants who will use the generation model and fill out a short questionnaire which will provide feedback into how they feel the level generation in the project compares to levels they have played in other, more traditional platformer games. The objectives of the project can be summarized as:

* 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 utilise programming techniques such as Perlin Noise and Markov Chains to implement a procedural level generator using the Unity game engine which effectively generates interesting and varied levels.
* To evaluate the feedback provided by participants to consider how procedural level generation compares to traditional level design in keeping players engaged for longer.
* To research current 2D platformers using procedural generation, for example Spelunky, and consider how this could be improved upon using my own methods and techniques.