Hey Im Edgar, and for this project I plan to learn about procedural generation.

# So What is Procedural Generation

Procedural Generation or proc gen for short is a technique used in game development and various other fields to create content algorithmically rather than manually. This is usually achieved by mixing rules, algorithms and randomization to generate content on the fly.

Within game development, proc gen is used in a variety of ways such as:

level and terrain generation,

character dialogue and animations,

and object instantiation and loot systems.

This is why within game development proc gen is very useful and continuously used for advanced game development.

Now its important to note that proc gen has been being used for game development for about 50 years now. It all started with a game called rogue which was released in the 80s. This game utilised proc gen to generate its levels in a way which made the levels different every time the game is played. Ever since then this type of level generation became popular and as a result of its success a new game genre emerged and is now called Rogue-Like.

# Lit Review

As mentioned before, proc gen isn’t something new, which means that ever since then developers have found many more ways to achieve procedural level generation. While all these different ways and algorithms achieve a similar purpose, they all do it in a different manner and have different parameters which one can change. Nowadays, there are some known techniques such as grammars, wave function collapse, Markov chains, and a bunch more which I found through an article which explores these ways and introduces a way of creating a hybrid approach based on these known techniques.

However, there are way too many techniques for me to go learning all of them in one semester. Specially, when some of these are used for completely different aspects of game development. Therefore, for this project I’ve decided to focus on learning how to procedurally generate levels in a way which would be beneficial for a rogue-like game.

When it comes to rogue-likes, procedural generation is the backbone of the genre. Most games in the genre utilise proc gen to generate ever changing levels, and most also use proc gen for their item systems. This is one of the many reasons rogue-likes have high replay value. However, it also means that a lot of games in the genre share similar features.

Nowadays, most games use proc gen to either generate levels randomly with some rules to fit their needs, and others utilise a strict structure to generate more complex levels which feel crafted to fit a criterion. However, there is a lack of middle ground where these two concepts meet. Therefore, I plan to research and come up with a way to fill this gap.

Add one more source going over how rogue-likes do level generation

Showcase prototype.

# Project Outline

So with this in mind lets discuss what my project will look like.

Firstly, what do I want to learn?

I want to not only learn how to create an algorithm that generates a level that fits my needs, but I also want to understand how it is happening and how I can change it to fit other purposes.

Since there are so many different ways to procedurally generate levels, I want to be able to understand the differences between some of these techniques so that I can use them and combine them as I need to.

Now lets look at why I want to learn these things. See most of my favourite games are rogue-likes, so naturally I have some ideas to create one of my own. Now in order to do so, I want to be able to procedurally generate levels which feel reactive to certain players conditions, making the player feel like the levels are crafted to their specific scenarios.

Therefore, for my project I will be focusing on developing an algorithm which uses proc gen to create levels. However, I don’t just want a boring simple dungeon. The whole project will be based on an idea for an elemental driven game where difficulty is an ever changing statistic which is influenced by the player’s actions.

How would this work? For example lets say that at the beginning of the game, a player chooses an element … such as fire. This means that the player will now be weak to water and stronger against grass.

Now as they begin the game their difficulty begins at 0, however as they progress through the levels their difficulty stat goes up by 5 making the game harder. This means that as the algorithm generates the next level, it will take the increase in difficulty into account and generate a level with more water hazards such as enemies and traps, which would then make the game harder.

As for deliverables I plan to have a few prototypes showcasing different algorithms used to generate levels.

Then my hybrid version of these algorithms which I like the best for my intended use.

And finally another algorithm which populates the generated level with enemies or traps based on the player's situation.