Reactive Procedural Level Generation

COMP3180 RESEARCH REPORT
Name: Edgar Murga Garcia De Leon
Student ID: 47094133
Topic Category: Procedural Generation

Introduction

Procedural generation is a technique used in game development to create content algorithmically rather than manually. This is achieved by mixing rules, algorithms and randomization to generate content during runtime. This topic is of great relevance for advance game development as in today's industry procedural generation is used in a variety of fields to reduce the amount of manual labour needed for some of these fields. These include (not limited to):

- Level/Terrain Generation
- Character Dialogue & Animations
- Object Instantiation & Loot Systems

While there are a lot of fields in which procedural generation is beneficial for game development, I will only be focusing on level generation (and population but more on that later) with an emphasis on level generation beneficial to a roguelike video game.

For context, a roguelike is a subgenre of video games which comes from the video game *Rogue*, and ever since 2008 through the Berlin Interpretation **Random Environment Generation** is one of the most important factors of roguelikes [1]. Therefore, I plan to link my learnings with the need for procedural generation in roguelikes to create a procedural level generation algorithm which would make a roguelike game feel reactive to real time player conditions.

Literature Review

A Hybrid Approach to Procedural Generation of Roguelike Video Game Levels [2]:
To begin my research, I looked at how procedural generation had been used in video games before. This is where I found an article by *A. Gellel & P. Sweetser (2020)* where multiple procedural level generation techniques are analysed in order to create a hybrid approach to procedural level generation for roguelikes.

Through the research of procedural generation techniques explored in this article, I've been able to identify which known techniques would be beneficial for my project. While the article discusses various techniques, I acknowledge that some of them are not relevant to my project. Therefore, I have chosen to concentrate solely on two techniques: Context Free Grammars, which I am already familiar with, and Cellular Automaton, a new technique that appears highly promising for achieving my project goals. Furthermore, I've recognised the importance of combining different approaches to develop a hybrid method, as suggested by *Gellel & Sweetser (2020)*. Consequently, I've chosen to dedicate time researching various approaches that can synergize effectively, allowing me to create a unique method for generating and population procedurally generated levels tailored to a roguelike video game.

Overall, *Gellel & Sweetser's* work has exposed me to a variety of material, some of which doesn't align with my specific requirements. Nevertheless, a portion of their exploration and analysis has proven exceptionally valuable for my project.

Procedural Map Generation Techniques – Roguelike Celebration [3] & Cellular Automata | Game Development Tutorial [4]

After coming across the Cellular Automaton technique in the previous source, I decided to delve deeper into its workings and application. To achieve this, I turned to two sources: firstly, a talk by *H. Wolverson (2020),* which explores a variety of procedural map generation techniques, and secondly a YouTube tutorial by *White Box Dev (2020)*. In his tutorial, he demonstrates the practical use of cellular automaton in game development.

Through the combination of these sources, I've been able to establish Cellular Automaton as a valid technique which could help in the project development. From the talk by *H. Wolverson*, I got gained a better understanding as to how this technique is used in game development. However, it wasn't unit I watched *White Box Dev's* work that I understood how the technique can be implemented in game development and manipulate cell states within a grid to generate levels. Additionally, these sources helped start a train of thought regarding how I can utilise the principles from cellular automaton to not only generate levels for the project but also populate them based on surrounding rooms (aka cells).

In general, I found both these sources to be beneficial. While *H. Wolverson's* work contained some material that wasn't directly beneficial to my purpose and covered some topics I was already familiar with, his introduction to Cellular Automata led me to discover valuable material. On the other hand, *White Box Dev's* work provided valuable information, enabling me to understand the fundamental principles of cellular automaton and its potential utility for my project.

WaveFunctionCollapse [5]

While learning about cellular automaton, I noticed similarities with wave function collapse. To explore this further in the context of video games, I discovered a project by *Maxim Gumin (2023),* sponsored by *Embark Studios (Stockholm Games Studio)*, which generates bitmaps from input bitmaps utilizing wave function collapse.

Gumin's research, provided me with a comprehensive understanding of wave function collapse and its application in content generation. While exploring his work, it became evident that this technique could be particularly helpful for level generation. Furthermore, I noticed various similarities between how this algorithm and cellular automaton generate content. This realization allowed me to see how combining these two techniques could create levels fit for a roguelike and thus aligning with the project's needs.

Overall, this proved helpful in sparking ideas for achieving my goals. However, it primarily focused on areas unrelated to video game level generation. Therefore, it served as more of an inspirational source rather than foundational research for the final project.

Binding Of Isaac: Room Generation Explained! [6]

During my research, I became curious about procedural level generation methods used in games that inspired me. To explore the approach behind '*The Binding of Isaac'*, I explored a video by *Florian Himsl (2020)*, the original flash game's programmer, in which he provides a detailed explanation of his algorithm.

Watching *Himsl's* work gave me a visual understanding of how to achieve procedural level generation. As *Himsl* explained his algorithm, my research started to come together in my mind, forming a clearer visual representation of how I could combine all my findings to create a procedural level generation algorithm tailored to my project's requirements. Furthermore, observing a developer explain an algorithm without rigidly adhering to a predefined set of instructions based of wave function collapse or cellular automaton, made me realise that these techniques are not strict instructions but rather guiding principles. Ultimately, this boosted my confidence in applying the principles I've explored to create a custom algorithm tailored to my specific requirements.

Overall, this source was incredibly helpful. The content was informative and will act as foundational research for my project. Additionally, this source helped boost my confidence and inspired me to begin development.

Project Proposal

As for the project I will focus on developing an algorithm which procedurally generates levels fit for a roguelike. This means generating a level using simple shapes such as rectangles to represent rooms (similarly to 'The Binding of Isaac'). However, I want to create a feel of reaction, meaning that I want the levels to feel reactive to certain player conditions. Therefore, I plan to introduce an element system similar to *Rock Paper Scissors* where the levels get populated with stronger or weaker hazards based on the character's chosen element, and a difficulty statistic which would change as the character progresses through the levels.

Learning Goals

As a result of this project, I plan to learn the following outcomes:

- **Demonstrate knowledge of procedural level generation.**
    - **Industry Relevance:** procedural generation is commonly used for a variety of development areas, making it a useful tool for game advanced game development.
    - **Personal Relevance:** my favourite games are roguelikes and procedural generation is the backbone of the genre, thus I want to learn about it.
- **Demonstrate understanding of various known procedural generation techniques.**
    - **Industry Relevance:** specific mastery within multiple subareas of procedural generations will show personal interest.
    - **Personal Relevance:** plans for own game development will require knowledge in multiple techniques commonly used for level generation.
- **Apply procedural generation techniques to develop a custom level generation approach.**
    - **Industry Relevance:** the result of this will be a great portfolio piece which demonstrates my dedicated research, and applied theory.

    - **Personal Relevance:** the result of this, would be beneficial for personal game development, as it could be used for games of my own.
- **Apply research findings with previous experience to develop complex algorithms without strict strictions/guidelines.**
    - **Industry Relevance:** this will demonstrate my ability to independently learn new concepts to an extent where I can apply my learnings and create working prototypes without extensive help – independent learner.
    - **Personal Relevance:** this will help me gain confidence in doing more personal projects, as well as provide me with experience on what to keep doing and what to change for future endeavours.

Deliverables

The specific proposed deliverables (e.g. prototypes, documentation, experiments, analyses) you intend to create for your final project that build on the ideas or address the gaps from your reading. Be sure that each deliverable includes some detail on how its success will be measured (e.g. number of words for a report, supported features for software, number of testers for prototype, quality goal for evaluation, etc.). Evaluation of outputs (e.g. their utility, usability or UX, as appropriate) is a required component of the final project, as is creating a portfolio piece with industry relevance (including a short demonstration video showcasing relevant prototypes), so be sure to identify the individual deliverables that address these requirements.

In order to demonstrate success on the learning goals I plan to accomplish the following deliverables:

- **Final Project Report (FPR) including evaluation write-up. (7 hrs)**
    - **Success Measurement:** 1500 – 2500 words.
- **Unity prototype demonstrating 1st technique for procedural level generation. (6 hrs)**
    - Simple prototype used to learn one technique.
    - **Success Measurement:** Different iterations constantly produce similar result to template.
- **Unity prototype demonstrating 2nd technique for procedural level generation. (6 hrs)**
    - Simple prototype used to learn one technique.
    - **Success Measurement:** Different iterations constantly produce similar result to template.
- **Context Free grammar for my intended level structure. (2 hrs)**
    - Guideline for I plan to generate/populate my levels.
    - **Success Measurement:** Grammar typed in document with no ambiguity and understood by fellow developers.
- **Unity project demonstrating my hybrid approach to level generation. (20 hrs)**
    - Level generation algorithm influenced by both prototypes. (12 hrs)
    - Level population algorithm influenced by character statistic. (8 hrs)
    - **Success Measurement:** Different iterations constantly produce similar result to template & population reacts to player conditions such as element type and current difficulty.
- **Short video demonstration of hybrid PCG approach. (4 hrs)**
    - **Success Measurement:** 2-3 minutes of significant video footage.

- **Technical document outlining the influence character statistics has on the population algorithm. (4 hrs)**
    - **Success Measurement:** ~500 words outlining the element system and difficulty stat which influences the level population.
- **Evaluation artefacts in GitHub Repo. (15 hrs)**
    - **Success Measurement:** Content includes data gathering, analysis & evaluation.

References

[2] Gellel, A., & Sweetser, P. (2020). A hybrid approach to procedural generation of roguelike video game levels. *International Conference on the Foundations of Digital Games*. doi:10.1145/3402942.3402945

[5] Gumin, M. (2023). Retrieved from https://github.com/mxgmn/WaveFunctionCollapse *Bitmap & tilemap generation using Wave Function Collapse*

[6] Himsl, F. (2020). Retrieved from https://www.youtube.com/watch?v=1-HIA6-LBJc *Binding of isaac: Room generation explained!*

[1] Stegner, B. (2021). Retrieved from https://www.makeuseof.com/what-are-roguelike-and-roguelite-video-games/

[4] White Box Dev. (2020). Retrieved from https://www.youtube.com/watch?v=slTEz6555Ts *Cellular automata | procedural generation | game development tutorial*

[3] Wolverson, H. (2020). Retrieved from https://www.youtube.com/watch?v=TlLIOgWYVpI *Herbert Wolverson - Procedural map generation techniques*