Game Design-friendly Procedural Generation of Game Levels

# Abstract

Procedural Generation (PG) is often used in games for smaller aspects of the game’s scope and as such is scarcely used to produce entire levels for games, specifically in the Action genre, however, the increasing popularity of Roguelike games has led to further exploration with the concept of fully PG levels and experiences. A Roguelike is a game designed to have the player perform multiple runs of the same game with the overarching goal of reaching its end. While the player is completing multiple runs of the same game, each run differs in various magnitudes from enemy placement, types, strength and sometimes the room itself. This idea of generatively creating slightly different experiences of the same game each run makes a Roguelike game a suitable showcase for the usefulness of PG with the added challenge of making the resulting level designable while the designer is not handcrafting the levels themselves.

# Introduction

For Procedurally Generated content to be Game Design-friendly, the content must use elements of handcrafted and designed content in tandem with the content produced by the system when run. This means that for this project, a system will be built to showcase the power of Procedural Generation while still using the work and content created by a designer without the designer creating entire levels themselves. To that end, a Roguelike game will give the structure needed to create a game in which the player aims to clear a dungeon of enemies without knowing the layout of the level each time the player enters the game.

Roguelike games such as *Hades* (SuperGiant Games, 2020) feature a string of rooms filled with enemies, these rooms form the level or dungeon that the player must clear to either finish the game or progress the story. *Hades’* dungeon however does not have the rooms occupying the same space, instead, the rooms are treated as if in a vacuum. The proposed system for a procedurally generated dungeon is then to have these rooms explicitly connected through hallways and have them coexist with one another.

To achieve this and to achieve a desirable level of design, a similar workflow to Vazgriz’s *Procedurally Generated Dungeon* (Vazgriz, 2019) is adapted to use prebuilt rooms that contain at least one designed encounter inside. The proposed workflow has been chosen due to its flexibility and to lighten the work needed to create the hallways by hand.

The workflow includes three key concepts to succeed, Delaunay Triangulation (DT) (Rebay, 1993), Minimum Spanning Trees (MST) (D.Kalpanadevi, 2013), and the A\* (A Star) pathfinding algorithm (Xiao & Hao, 2011). DT allows the rooms to be connected to each of the closest immediate neighbours, MST allows for the path through the dungeon to be the shortest route through every room, and A\* will handle the positioning and groundwork for the generation of hallways connecting each room as dictated by the MST.

Once completed, the workflow should produce a well-connected dungeon that includes all the placed rooms with the shortest path. The workflow requires a low level of input from the developer at runtime and runs fully autonomously once started.

# References

D.Kalpanadevi. (2013). Effective Searching Shortest Path in Graph Using Prim's Algorithm. *International Journal of Computer & Organization Trends*, 310-313.

Rebay, S. (1993). Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics*, 125-138.

Vazgriz. (2019, November 18). *Procedurally Generated Dungeons*. Retrieved from VAZGRIZ: https://vazgriz.com/119/procedurally-generated-dungeons/

Xiao, C., & Hao, S. (2011). A\*-based Pathfinding in Modern Computer Games. *International Journal of Computer Science and Network Security*, 125-130.