Generation of Random mazes / Maze like Levels and Creation of a Game Using them

Project Final Submission

De Montfort University

Computer Games Programming

## Jake Coffey – P15187559

# Abstract

Random generation of various content in games has become commonplace in a subset of games that use some level and style of random generation to design the layout of the game level and what content from a pool is used in each level. The level of random generation in each game varies, with some using pseudo random generation that takes pre-built level portions and content spawn points and combines them in a random way, and others using true random generation with some limiters applied to prevent problems such as content being created within each other. Generation of levels can be a complex matter, with various and varied methods exiting that provide a wide range of different outputs and levels of complexity. The majority of these can and are applied in creating maze or maze-like levels, with random generated dungeons in games typically resembling mazes. The purpose of this project is to take a method a random generation and apply it to creating maze like levels to be used in a game. The game style used in combination with the generated levels is a “rogue-like” game, where the level and content (enemies, items, traps) are randomly generated and placed through out the levels to allow the player to progress. This project shows various generated levels that, while not resembling the typical perfect maze (hedge maze style), resemble a non-typical clear maze-like structure, as well as a rogue like game that utilises these levels appropriately. The process and reasons for the generation and game creation choices will be discussed in this paper.

# Introduction

The program created consists of a fully-fledged game that fully utilises random generation to create maze like levels, not using any pre-set combinations of level designs to achieve “random levels” and instead using an algorithm that randomly iterates through “cells” of the level and chooses which cells should be open walls or open, with some limits set in place, discussed in the generation segment of the report, to ensure a navigable and maze like structure is generated. The algorithm used is similar to a style of generation know as recursive backtracking, that chooses a random direction each iteration and creates a path or wall in that direction if it can, continuing until it can no longer do this and then going back until it reaches a cell where it can continue this process. The differences between this algorithm and the method implemented will be discussed in the generation segment of the report. Creating levels that both resemble a maze-like structure and are suitable for a game level is a challenge that required good programming and debugging skills as well as creativity in the approach used in the implementation. The game implementation also uses random generation to some extent to populate the generated levels with content from the pool of enemies, items and traps created as part of the project in order to give the structure of a “rogue like” game that allows the player to progress and improve in each run through the game.

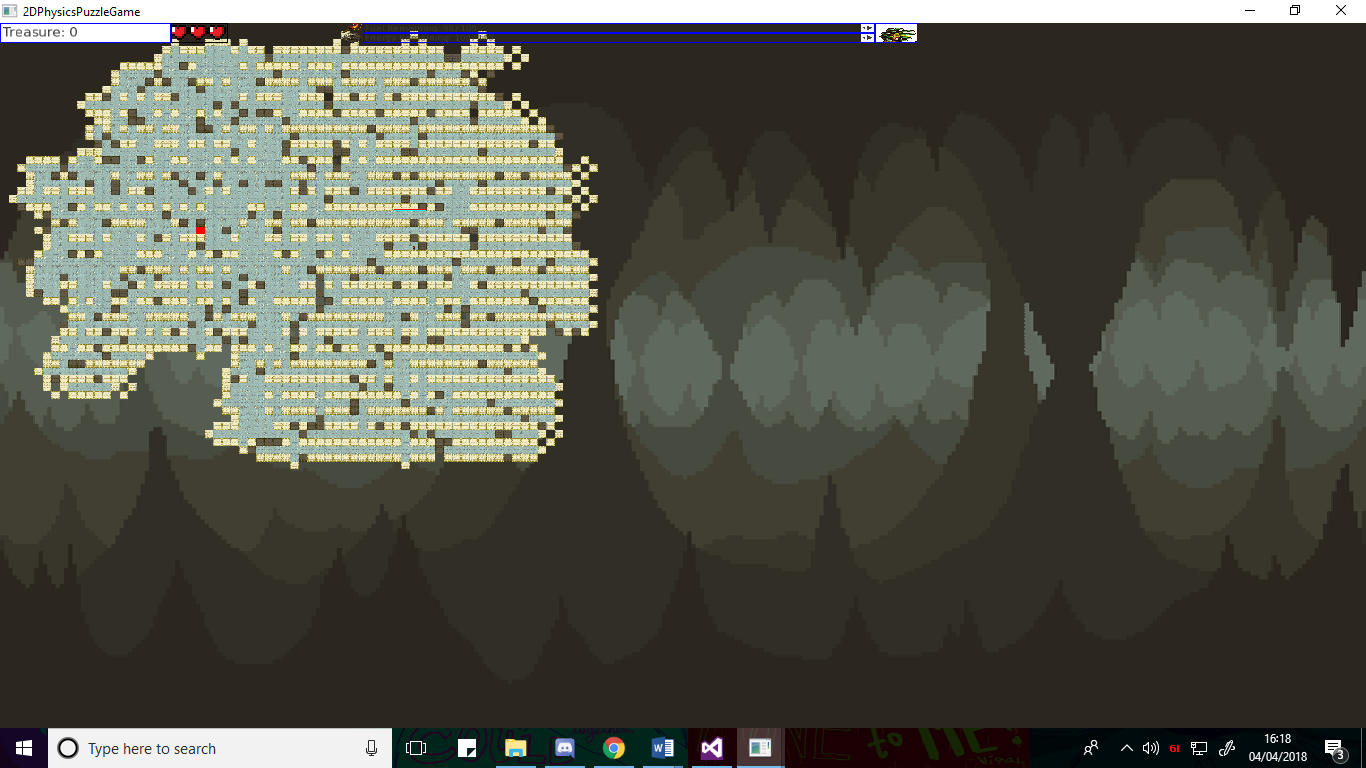


Fig.1 – Left shows the normal view of the game. Right shows a zoomed out shot of the generated levels

Various libraries are used to create the appropriate implementation. The super-fast media library ([4]SFML) is utilised for managing parts of the project that improve the overall quality, including sound, window management, textures and sprites, the game loop, and keyboard input. The game is rendered and designed in 2D to properly utilise the functionality provided by SFML. To properly implement a rogue like game that utilises the generated levels, a GUI is required for a variety of purposes. The GUI will be required for menu systems for a main menu, upgrades and progression, and representing information important to the player during actual gameplay. To implement a professional looking and functioning GUI and menu system, an extension of the SFML library called [6]TGUI was used, that focuses on and provides methods for GUI functionality. [1]Box2D was utilised to provide physics and collision management for the project, including special interactions between different entities in the game and filtering of collisions that were unwanted.

While various rogue like games and level generation algorithms and methods exist, the two in combination are typically not done with true randomness, instead using the randomness algorithms to piece together procreated level portions and content in a random manner. As such, a portion of the reasoning behind the project was to see how effective the level generation would be for a rogue-like game when not using pre-set portions of levels and instead generating the level randomly through a set of iterations, limitations and an algorithm, and then attempting to create a fully functioning game.

Portions of the project are similar or utilise functionality that I have experimented with previously, where as other sections of the game created are completely unlike anything I have undertaken before. As such, this, in combination with the previously mentioned need for creativity in the implementation as well as good programming skills, knowledge, debugging skills and research skills, forms further reasoning for undertaking this project, as it expands upon existing knowledge, as well as providing experience in new areas, allowing for a development of further knowledge and experience in programming various parts of games, and algorithms. Likewise, the amount of content need for a good project as well as for a good “rogue-like” game allows for creativity and experimentation with programming that also leads to further expansion of personal experience and knowledge. For example, I have created games before, however I have never developed items in a game that can be acquired and used to change how the game plays, and this project provided the opportunity to experiment and create a variety of these, developing my skills in game creation.

Further reasoning behind the project is the previously mentioned libraries being used. Undertaking this project allows for further development of skills utilising external libraries, something that is typically required when creating ambitious and complex games, as well as developing skills in the specific areas each of the libraries covered, these being, physics and collision form Box2D, game management, sound and sprite creation from SFML, and GUI creation and management form TGUI.

# Bibliography

[1] Catto, E. (n.d.). *Box2D | A 2D Physics Engine for Games*. [online] Box2d.org. Available at: http://box2d.org/.

[2] Sfml-dev.org. (n.d.). *Controlling the 2D camera with views (SFML / Learn / 2.4 Tutorials)*. [online] Available at: https://www.sfml-dev.org/tutorials/2.4/graphics-view.php.

[3] Sfml-dev.org. (n.d.). *Playing sounds and music (SFML / Learn / 2.4 Tutorials)*. [online] Available at: https://www.sfml-dev.org/tutorials/2.4/audio-sounds.php.

[4] Sfml-dev.org. (n.d.). *SFML*. [online] Available at: https://www.sfml-dev.org/.

[5] Tgui.eu. (n.d.). *TGUI: 0.8 Tutorials*. [online] Available at: https://tgui.eu/tutorials/0.8/.

[6] Tgui.eu. (n.d.). *TGUI: Texus' Graphical User Interface*. [online] Available at: https://tgui.eu/.