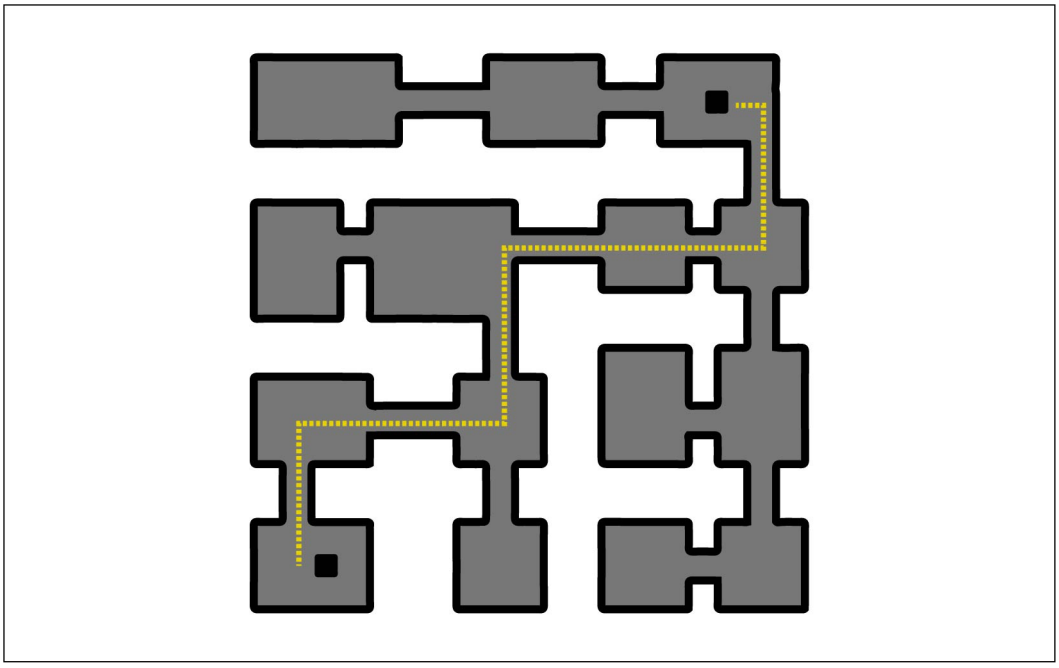
Escola Politècnica Superior

Universitat de Girona

Project ADD: Adaptative Difficulty Dungeons

Trabajo final de grado

Grado en Diseño y Desarrollo de Videojuegos



Document: Report

Author: Elio Matías Valenzuela Muñoz

Director: Gustavo Ariel Patow and Nicolas Barriga Richards

Department: Informàtica, Matemàtica Aplicada i Estadística

Area: Llenguatges i Sistemes Informàtics (LSI)

Call: September/2020

Chapter 1

Introduction

For a lot of people, playing video games is something that allows them to have many hours of entertainment. In general, games test the skills, instincts and strategies of the players to make them fall into a state of pure concentration or complete absorption called "Flow".

When players want to start a game, they choose among the different difficulty levels the one that better fits their skills to start playing. However, those difficulties are created by the game designers, indicating that, in order to play in that difficulty, the player must develop a set of skills that fit that difficulty. That minimum skill level has been already set by the designer.

These two conceptions of difficulty and skill are different for each player and designer. Players may think that they possess the skills to overcome all the challenges and enjoy the game in the difficulty that they have already chosen. But, what happen when a player chooses the max difficulty and is not as hard as he thought it will be, or if the player want to start the game for the first time and the easy difficulty is not that easy? Or if the player develop the skills faster or slower and get bored of winning or losing too much? Players who are struggling with difficulty will no longer want to play that game anymore, they get bored and that is the last thing, that as a game developer we want our players to feel.

When the presence of challenge in the game is low, players get bored and when repeatedly confronted with defeat, they get frustrated and possibly lose interest in the game. In this work, we will focus our efforts on analyzing the player "skills" and quantify the difficulty of the level structure. We will find a way of generating content with a level of difficulty that fits the player's skill level.

The purpose and main objective of this project is to make a first approach to the creation of a tool that uses DDA techniques to generate personalized content. Using pathfinding algorithms such as A* and Breadth-First Search (BFS), we plan to find relations that help us interpret the player's skills and the procedurally generated level difficulty.

Chapter 2

Analysis and Design

For research purposes of this project, we establish two sections, the creation of a simple game in Unity3D to gather information from players about their performance and skills, and the analysis of the data gathered from playing the game and the correlation with the player steps in the maze and the pathfinding algorithms (A*, BFS). The game is divided into 4 main modules. These are the main processes in which we are going to explain the functionalities of the game:

- Dungeon Generation Module
- Path Finding Module
- Data gathering and mailing Module
- Game Process Module

2.1 Dungeon Generation

The dungeon generation process has 3 stages to create a maze: The initialization, generation and render. In that specific order, the maze needs to be initialized making the whole area full of walls without any room or connection. Then, we proceed to generate the maze completely separated from the render module. We create the dungeon using the BTP process and we select the start and the exit points of the maze level. Finally we render the maze with the connections and the rooms created, along with the entrance and the exit.

2.2 Pathfinding: A*

The algorithm was split into two parts, the calling and the procedure. The calling was made to return the validation of the maze and the information about the optimal path that were used in-game to complete the map creation. By default, we use a cardinal representation (4 directions) of the map navigation and the heuristic calculations. We start by resetting any previous calculated path and creating a node map representation of the tiles matrix. Then we call the A* implementation to proceed with the algorithm.

2.3 Pathfinding: BFS

With the A* implementation finished, the BFS implementation was a simple task. We are going now to show which parts of the algorithm are crucial to implement it with just a few adjustments to the A* code. If we do not use the heuristic, all the nodes that will be expanded

will have the same value, so we are not going to choose the node with the smallest F value because they will be all at the same in the open list.

2.4 Game Process

Once we have our dungeon creator system and pathfinding algorithms working, we proceed to create the game logic. The game characteristics that we need are the following:

- Player Movement
- Player Camera
- Game loop
- Menu & Settings

The "GameManager" class is the controller of all the things that occur in the game and is in charge of beginning and finishing the game processes, such as to start the game, reset the levels and finish the game process. With this in mind, the game manager is the one that controls the dungeon creator system and stays alert about the player behavior in all of the game process.

2.5 Menu & Settings

There are three UI objects in the game, the main menu, the settings menu and the Head-up Display. The first one refers to the first menu and most interactable of them all. It gives three options: "Play" (to start immediately the game), "Settings" (to open a settings menu) and the "Exit option" (to close the game).

2.6 Data gathering and mailing

The final part of our game is to be able to save the information into a convenient format and send it to us. We are going to gather the number of mazes completed by the player, along with the following information of every maze completed. The player steps are the nodes visited by the player through the level. Every time the player steps into a new tile, it is added to the counter. The completion time is the total time that the player took to complete the maze. The maze structure is the complete maze broken down into a single string data. Finally the size of the maze is also sent.

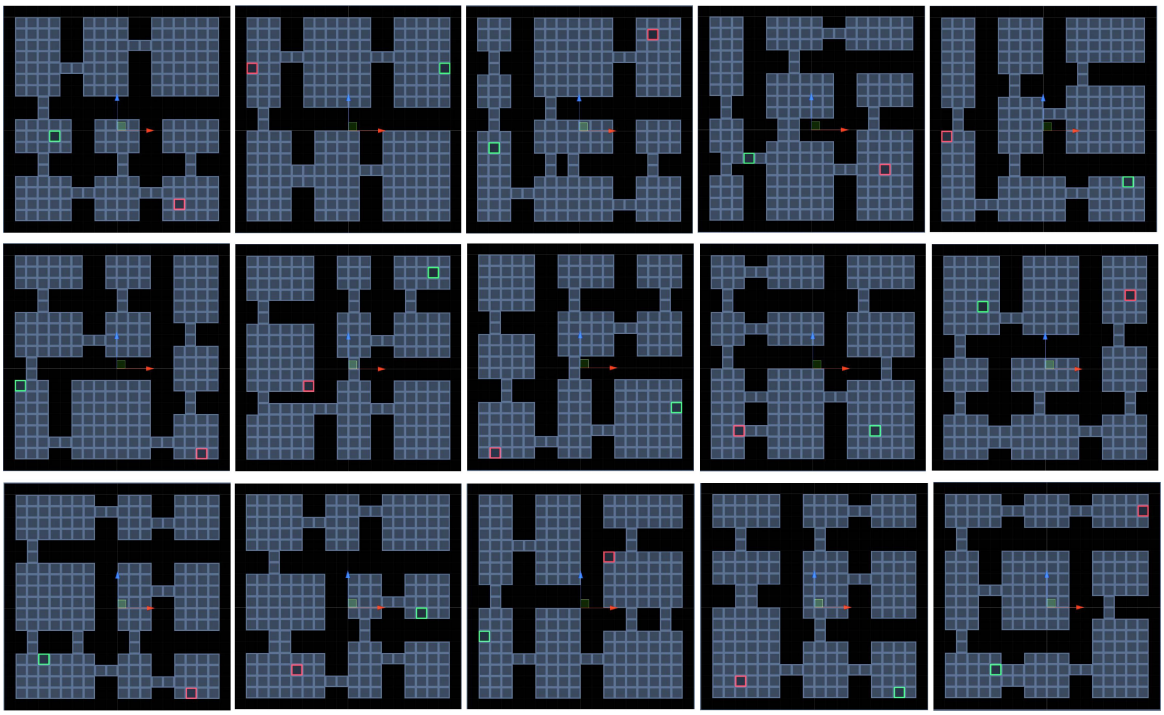


Figure 2.1: Diferent Maze Creation Process Results

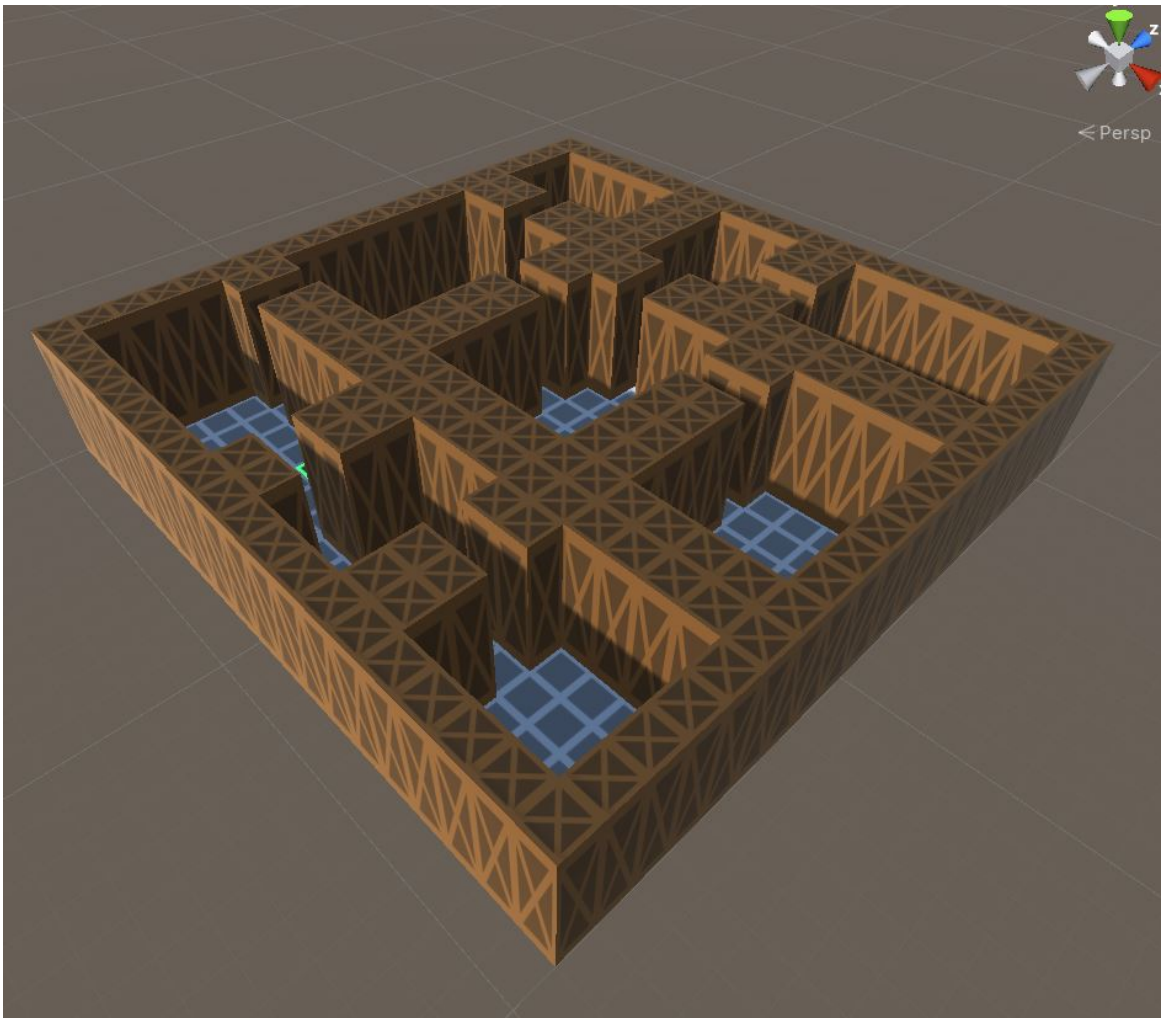


Figure 2.2: 3D Maze

Chapter 3

Results & Conclusion

Once we got our system working on the Windows platform, we proceed to gather test subjects to play the game. To do so, we first tested the system our self, gathering our own data.

We succeeded in our application of PCG with DDA techniques to create a first approach of a tool to measure the player performance and the difficulty of a level. In this case we managed to make a formula that makes a numeric representation of how good or how bad the player is reaching the exit of the level, which is the objective that we want them to pursue.

Equation 3.1 Formula to measure the players performance

$$x = \frac{player\_steps - optimal\_path}{opened\_nodes\_bfs - optimal\_path} \tag{3.1}$$

The formula is a normalization of the player performance, were we use the optimal path as minimum and the opened nodes of BFS as the maximum. The formula is used to measure the performance in previous mazes completed by the player. Once the player passes the level, the formula calculates how good or how bad he/she has performed in the maze, placing the player "skills" between the best
and the worst result.

With BTP we create a complete maze in which we can establish a size and the algorithm will create a complete dungeon maze with it. We use BFS and A* to measure the difficultly and the performance of the player. With that information we create a formula that represent the player performance and other formula that represent the maze difficulty, with tat information we can increase or decrease the size of the maze.

Equation 3.2 Formula to measure the level of difficulty of a maze

$$x = \frac{opened\_nodes\_bfs - optimal\_path - (maze\_size/2)}{opened\_nodes\_bfs} \tag{3.2}$$

We accomplish the purpose of the investigation, we found a formula that represent that specific layer of difficultly to the player. The BFS algorithm shows a strong relation to the player's behavior and the data analyzed (player steps) are a good start point to into the the creation of the DDA tool that we want to create in further work.