Documentation Guide for

UNITY MAZE GENRATION PROJECT

Adam McWilliams

adammcw01@gmail.com

Table of Contents

Description of the solution

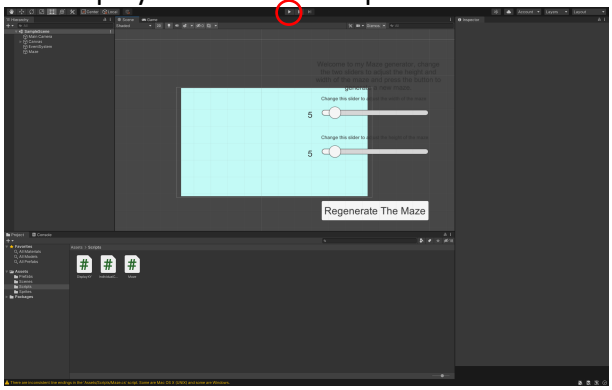
This solution is a Unity project which will create a perfect maze of a specified height and width and display it to the user. The maze features automatic scaling so that it is always visible regardless of the dimensions of the maze.
A perfect maze is a maze with no loops and every cell of the maze is reachable by every other cell in the maze.
The project features a user interface with sliders and a button to randomly generate a maze of the desired size.

Prerequisites and Installation

To install and run the project you must have Unity version 2020.3.6f1 or later installed from https://unity3d.com/get-unity/download.
Once Unity is installed open the Unity hub and click the add button and navigate to this folder, select the folder called 'DTT Maze Generator'.

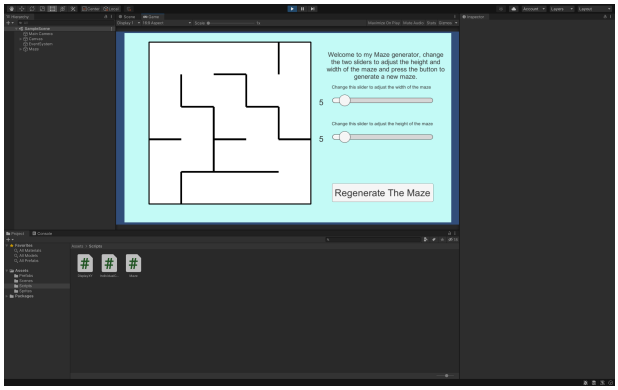
Usage Guide

Once the project is imported into Unity you will see the scene view, to run the project click on the play button at the top of the screen.



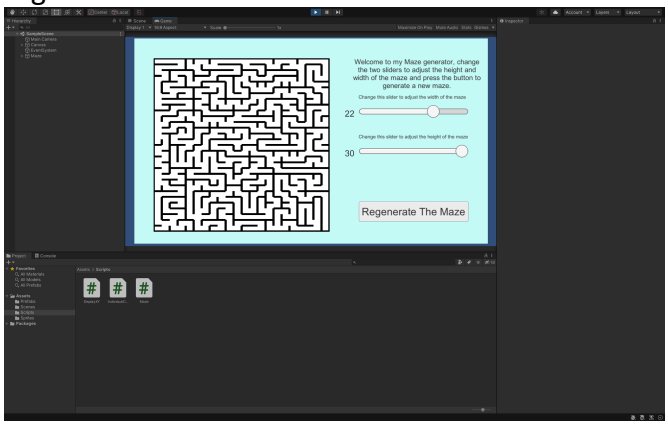
The screen in unity when the project is first loaded in.
Highlighted is the position of the play button.

This will take you into the game view, where a new maze has been generated. The default size is 5 by 5.



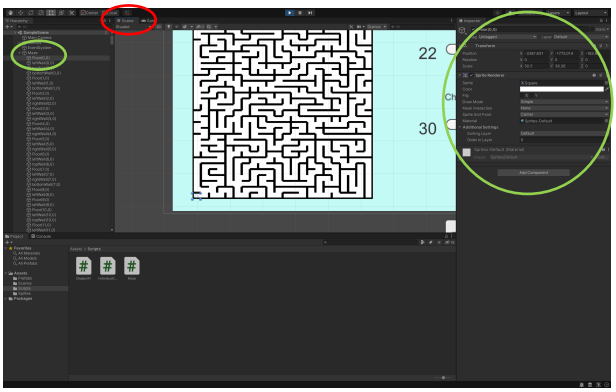
A newly generate maze of default size when the program is executed

Now you can adjust the parameters of the maze and see these changes by pressing the regenerate button.



A new maze has been generated after the user has adjusted the sliders

Aside from this you can also change into the scene view to see how the maze is being made by viewing the properties of each individual object.



The scene view of the project allowing the user to inspect the objects that make up the maze. Highlighted in red is the button to change into this view, highlighted in green is an object being selected and its properties

Finally, to end the program simply click on the play button again to exit.

Bug Report

The following is a report of encountered issues as well as the solutions that have been implemented.

U-M001:

The maze's visibility is inconsistent when the canvas is rotated relative to the camera.
The solution was to decrease the z-coordinate of the maze floors by 1 and the walls by 2.

U-M002:

A runtime error was occurring where the maze generation would not always terminate. The problem was the walk method would repeat infinitely. This was because checks were not in place for edge cases. This was found by use of using debug.log to output the values of certain variables during runtime.
The solution was to count the number of neighbouring cells that had been visited and then compare this value with the number of neighbours that exist, rather than using a hardcoded value.

U-M003:

The size of the maze object as well as the maze floor and their positions are inconsistent when using non-square mazes.
The solution was to translate the objects relative to the height and width values within the initaliseGrid method.

U-M004:

When using different resolutions different elements of the canvas are scaled differently meaning that not all components may be visible.
The solution was found by researching the unity documentation, https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html and ensuring the rendering modes were consistent and all based around the same point.

U-M005:

The program did not work for non-square mazes as the visuals were not rendering correctly as well as the algorithm did not execute as expected with large non-square mazes. The rendering issue was fixed in bug U-M003.
To solve the issue within the algorithm I set up debug points to output values of the iteration during the walk method and the initaliseGrid method, I discovered that IndividualCell objects were being created and pointed to a position in the grid array that didn't exist. To fix this overflow error I modified the way the maze objects were created so that everything was based around a cartesian system, where everything is (x,y) this is also the same as (width, height) to ensure consistency when using these variables and avoid confusion.

Future Developments

Unfortunately, due to the time constraints I was unable to implement additional ideas that I would have liked to implement.
The next stage of the project is to add additional algorithms to generate the maze and be able to compare their efficiencies by time complexity.
An improvement to user-friendliness would be to add additional themes for the project. This would be a good opportunity for me to familiarise myself more with menu systems in unity as well as changing the attributes of Unity objects in C#.
I would also wish to improve the efficiency of the generation algorithms to allow for mazes of extremely large heights and widths to be computable on most hardware.
A final idea I would like to develop for this project would be the ability to pick any two cells and determine the most efficient path between them and display this to the user.

Contact Information

If you encounter any issues or have any enquires about the project, please do not hesitate to contact me at:
adammcw01@gmail.com