UNIVERSITY OF MALAYA

WID3009 AI GAME PROGRAMMING

MAZE GENERATION

Student Name : Yeoh Kai Wen

Matric Number : WID170056 / 17149195/1

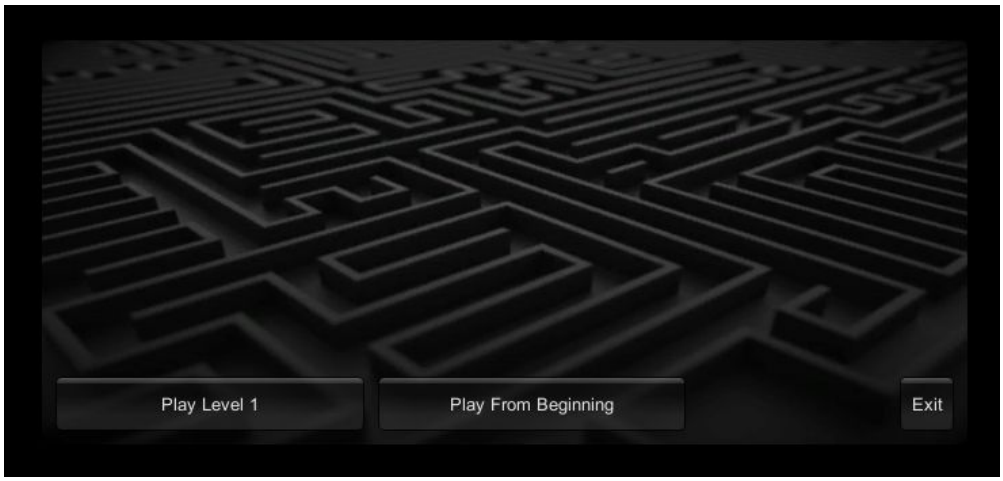
Lecturer : Prof. Dr. Loo Chu Kiong

Introduction

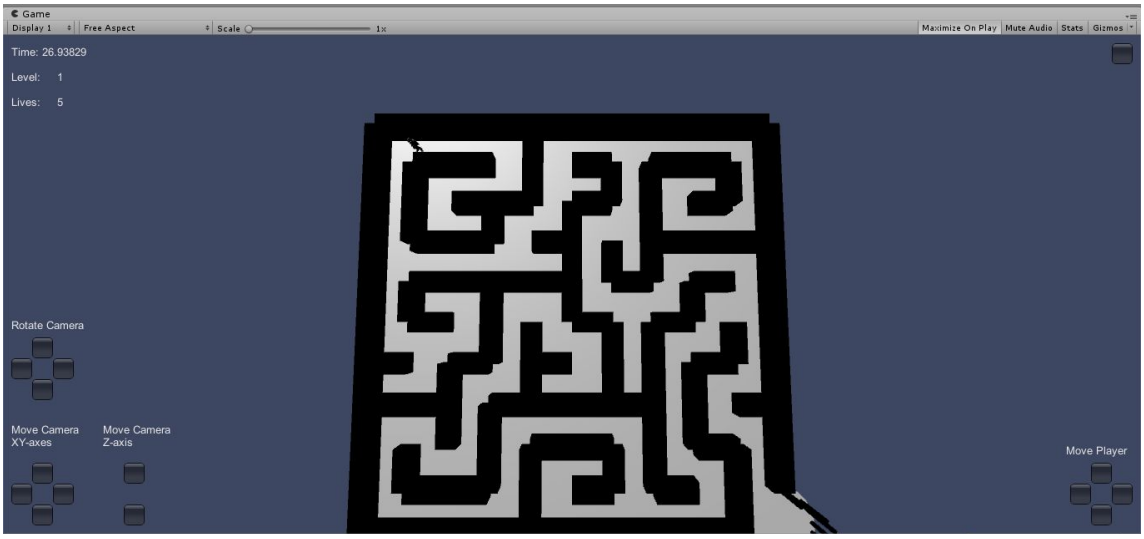
This project is developed using Unity and the programming language used is C#. Along with the report, this project folder contains assets such as C# scripts, textures, scenes, materials, prefabs and standard assets. This project is designed only for desktop PCs as part of the requirement to generate 3D mazes in Unity by applying a constructive PCG method. The starting and exit point of the maze is fixed while the paths used are designed using randomized algorithms to create a more fun game.

Assuming that cell is a wall, we fixed the starting position then followed with a depth-first search pattern by considering the cells around us. If there are any cells that are unvisited, that cell is chosen as the next cell. To break down the walls between the cells and to put up a new cell onto the top of the search stack by marking it as visited. Once the cells around it have been visited, pop the current location from the stack. As long as there is one cell on the stack, choose it as the current location and continue searching from the beginning.

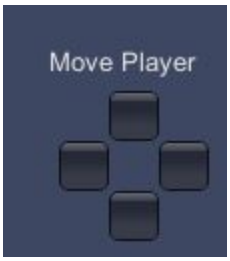
In this project, the player is trapped inside a maze and has to find a way out from the maze before the time runs out/ends. Once the game is Run /Play, it is as shown as below.



Whenever the game starts, the maze is generated randomly and if the user manages to escape from Level 1, Level 2 awaits him/her before they can finish the game.



To move the player, there are button controls such as Up, Down, Left, Right for the movements or using key controls on the user's keyboard for the inputs.



As it is in 3D, users will be able to adjust the angle of the camera on the screen using the Up, Down, Left, Right button for better view.

Conclusion

Given a particular location, the goal is to find the potential neighbors to satisfy two conditions. One is that it should not be in the visited cell and the other is that the boundary to the location should be a wall.. Therefore, each candidate will contain four items which are the next step, the bounds to use for the next step, learning from the previous location and the new bounds for the previous location. The random generator will then push the new location onto the search stack and it will update the new locations that are bound in the maze and lastly, it is able to add a new location to the visited cells. In conclusion, there are many different maze generation algorithms but it sums up that a maze is perfect if it has one and only solution.