

**3D Games Programming Assignment**

**Report**

**Soheil Salehinia (K2144160)**

**Maze Runners**

**Tasks, Aims, and Objectives**

Maze Runners is an endless third-person shooter game which the player should compete against an AI to exit through randomly generated mazes. These mazes come with different sizes based on the level of the game, and contain some enemies to slow down or distract the player and AI. There are two keys in the environment which should be collected by player and AI to exit from the exit point.

The main task was to implementing an algorithm to create these random mazes based on the defined size. These mazes should always have a solution, and should have only one solution to complete. The other task was implementing the AI who can find correct way to collect a key and exit through the exit point.

There are some others tasks which I mentioned below:

* Implementing enemies and their actions
* Placing enemies in environment in a fair random way
* Placing keys and exit point
* Implementing player and AI’s actions
* Implementing UI system
* Implementing Scene Manager
* Implementing Save System

**Plot of the Game, and User Instructions**

The plot of the game is to finish the maze faster than AI, and player should first collect a key and then exit through the exit point. In this path there are enemies which player should engage with them. In addition, player should try to slow down the AI by doing specific actions. The AI also do the same to the player to win the game.

The game will start with a 3 by 3 random maze which does not include enemies and by exiting this maze, a new bigger maze will generate with enemies inside. As the player progress the game there will be always bigger mazes to explore and the game will never end. If the AI exit faster than player, the maze will restart, and generated randomly with the same size.

Player can continue with the last size he was playing from the Main Menu at any time. He can also retry each level whenever he thought it is necessary. He can also start a new game from the Main Menu to begin with a 3 by 3 maze.

The attacking actions of player, AI and enemies have cooldowns. Therefore, they can not do an action continuously. There is also a cooldown at the beginning of the level for attacking actions so that player and AI do not engage each other at the beginning.

Player actions are Move, look, Sprint, Aim, Shoot, Punch, Drop Bomb. Player can aim to anywhere and shoot a bullet. If the bullet hit the AI, it will slow him down, and if the bullet hit enemies, it will destroy them. When player punches, if the AI is in the same cell as player, it will slow down the AI. Player can also drop a bomb which will explode after moving out of the cell, and creates fire for a limited amount of time. If the AI or player traverse through the fire, it will slow them down.

The AI actions are Choose Path, Move Fast, Move Slow, Shoot, Punch, Drop Bomb. Attacking actions are as same as player’s attacking actions. He can shoot or punch player to slow him down, or drop a bomb to create fire.

There are three types of enemies in the game which are drones, bombers and runners, I will briefly explain their functionality below.

Drones move randomly with low speed and when the player or AI be in their range and sight they will shoot, and if the bullet hit the player or AI, it will slow them down. Therefore, the player cannot sprint anymore and the AI will move slow for a limited amount of time.

Bombers also move randomly with low speed and the player or AI be in their range they will drop a bomb. As soon as the bomb hit the ground, if the player or AI are not in the same cell, it will explode. This explosion creates a fire in the same cell, and if the player or AI try to traverse through that cell, they will become slow for a limited amount of time. The fire also will vanish after some time.

Runners move randomly with high speed and try to distract the player. There are actually harmless, but they may choose the correct or the wrong way. Therefore, it is better to kill the source of distraction.

In the below table I mentioned user instructions:

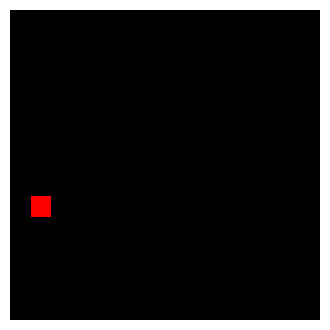
|  |  |  |
| --- | --- | --- |
| **Action** | **Mouse and Keyboard** | **Controller** |
| Move | W/A/S/D | Left Stick |
| Look | Mouse Pointer | Right Stick |
| Sprint | Left Shift | Button North (Triangle) |
| Aim | Right Mouse Button | Left Trigger (L2) |
| Shoot | Left Mouse Button | Right Trigger (R2) |
| Punch | Control | Button East (Circle) |
| Drop Bomb | F | Button West (Square) |
| Pause | Escape | Select (Options) |
| UI Navigate | Up/Down/Left/Right | D-Pad |
| UI Submit | Enter | Button South (Cross) |

**Major Techniques, Algorithms and Tools**

**Random Maze Generator: [[1]](#footnote-1)**

I used a famous algorithm called Backtracking Generator to create the mazes. In this algorithm we create number of cells based on the size of the maze with walls around each cell. We start from a random cell, and randomly choose a wall. Then we delete that wall and go through it to the next cell. We repeat the same job for the new cell, but we should only consider the walls that open a path to an unvisited cell.

We should continue the same procedure until there will be no adjacent unvisited cell remaining. At this point we go back to previous cell and repeat the same functionality. This algorithm will be finished when we go back to the cell which we started from.



**AI Path Finding:**

The AI uses a heuristic approach to find the path first to the key and then to the exit point. AI has a memory of cells which at start it is empty and the path he took. He randomly chooses an available path and go to the next cell. He will continue this until he reaches a dead-end. He will add cells’ information and the path he used to his memory as he traverses the maze. Then he will backtrack to the last cell in his path which has an available new path to explore. He continues this procedure until he finds a key. Again, the AI will continue the same until he finds the exit point.

If the AI finds the exit point before finding a key, he will remember the exit location by adding it to stack of cells. From there he tries to find a key and, in his path, he will add every cell which is not duplicate to the stack. When he finds the key, he knows exactly the shortest way to get back to the exit point by popping cells from the stack and go through them one by one.

**Player Third Person Shooter:**

I used the Starter Assets - Third Person Character Controller[[2]](#footnote-2) from Unity Asset Store for the basic implementation of my player controller. I modified this package to add aim, shoot[[3]](#footnote-3), drop bomb and punch functionality and animations[[4]](#footnote-4).

**AI Choose Action:**

The AI will choose an action when he reaches to a new cell. He can choose between move and attacking options to enemies or the player. He chooses these actions randomly based on availability of the action at that point. I also modified Starter Assets - Third Person Character Controller to use the same system without input system and camera for the AI and the Runner enemy.

**Placing Objects:**

If we consider the maze a matrix of cells, the player and AI always start from the first cell of this matrix. The keys and exit point will place in corners of the maze. Then we divide this maze to 3 by 3 small mazes and in each one except from the one who the player and AI are in, we place an enemy randomly in a random place in these small mazes.

**Specification of the source code**

Below is the class diagram of the game:

1. <https://github.com/john-science/mazelib/blob/master/docs/MAZE_GEN_ALGOS.md#backtracking-generator> [↑](#footnote-ref-1)
2. <https://assetstore.unity.com/packages/essentials/starter-assets-third-person-character-controller-196526> [↑](#footnote-ref-2)
3. <https://unitycodemonkey.com/video.php?v=FbM4CkqtOuA> [↑](#footnote-ref-3)
4. <https://www.mixamo.com/> [↑](#footnote-ref-4)