|  |
| --- |
|  |
| Artificial Intelligence |
| Assignment |
|  |
| **Muhammad Salihin Bin Zaol-kefli**  **Kinnear Justin Wong**  **Fong Zhi Zhong** |
|  |

|  |
| --- |
| This document describes the assignment for Artificial Intelligence |

**TABLE OF CONTENTS**

1. **Option**
2. **Game Idea**
3. **Gameplay**
4. **Types Of AI**
   1. **Waypoint AI**
   2. **A\* AI (Random Movement)**
   3. **A\* AI (Chasing)**
5. **Task List & Code Snippet**

**Option**

We have opted to do option 1 of the assignment. This option requires us to implement a demo with some AI feature(s). Code, documentation and presentation are required of this assignment.

**Game Idea**

The game is developed using Unity game engine. It is catered for Windows, Mac and Linux platforms. It is a remake of Pac – Man. It is to be in 3d. The game will be in 3rd person with the camera being top down view. The Artificial Intelligence will be implemented on the ghosts and will be different from that of the original.

**Gameplay**

The game will be player input based. Players will control the Pac – Man and collect as many bits as possible and progress through the levels before the number of lives becomes 0.

**Types Of AI**

**Waypoint AI**

Ghost AI makes use of Waypoints. Waypoints are set in the level’s corresponding waypoint level text file. They are then stored into a list for the AI to traverse through.

**A\* AI (Random Movement)**

Ghost AI makes use of A\* Path Finding. The AI picks a random spot on the map that is not an environmental tile and plots a path towards it.

**A\* AI (Chasing)**

Ghost AI makes use of A\* Path Finding. The A\* Path Finding computation will only take effect when the Pac - Man is within range. As long as the Pac – Man is within range, the Pac – Man’s position will be set as the AI’s destination. A\* Path Finding will plot a path from the AI’s current position to the destination.

**Task List & Code Snippet**

**Kinnear Justin Wong**

**Description Of Scripts**

AstarWaypoint.cs

Finds the path needed for the AI to walk to random areas of the map that are available to the player

DisplayEditorGUI.cs

Takes care of displaying the GUI of Unity for display of the level eitor components of the Level Editor scene

EditLevel.cs

Does editing of the Level Editor in the LevelEditor scene from mouse to world using a Unity raycast

LoadLevels.cs

Loads the level of the respective level from a textfile into memory

NodeContainer.cs

Class container to contain notes that the A\* pathfinding AI uses

NodeDetails.cs

Class container that stores the respective prefab’s level tile’s swappable tile objects while in the Level Editor

ToggleOutline.cs

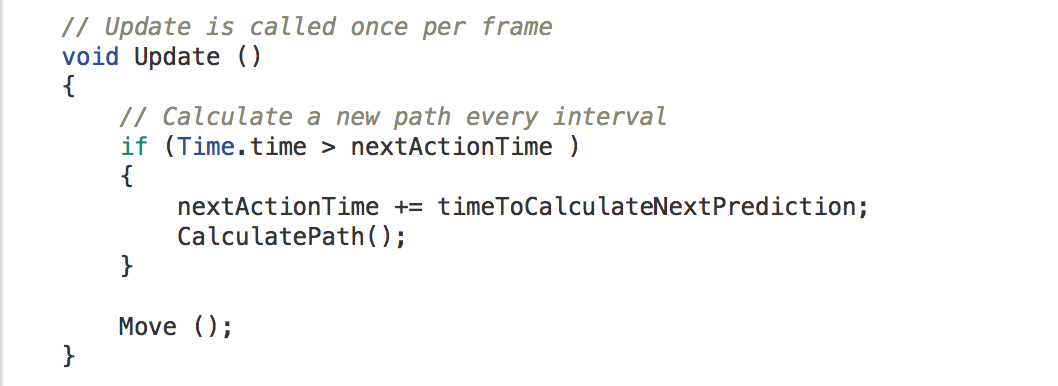
Simply toggles the outline shader to enable or disable the outline shader upon highlighting the tile with your mouse in the Level Editor

WaypointAIScript.cs

Simply obtains the waypoints from the corresponding waypoint level textfile and stores the waypoints into a list for the AI to tranverse through

**A\* AI**

A\* pathfinding is integrated into the AI of the random movement AI. The AI picks a random spot on the map and pathfinds its way there



CalculatePath() runs at an interval set in the Unity Editor. CalculatePath() is to calculate the desired path of the AI towards a target by the A\* algorithm. The method used was obtaining map data from the textfile of a level and then converting it into areas that the AI can walk on.

Move() moves the AI towards targeted positions in the list of nodes stored.

**Waypoint AI**

The Waypoint AI does waypoint movement by obtaining it’s waypoints from a separate textfile from the level text file. The waypoint text file has a exact file name to the Level but with a “PatrolWayPoints” appending to the end of the level’s name  


MoveAI() is to traverse through the list of waypoints that were set in the Level Editor in the particular level. The AI “Ping-Pong’s” back and forth towards the AI’s respective positions in the list.

**Muhammad Salihin Bin Zaol-kefli**

**Description Of Scripts**

AStarPathfinding.cs

Finds and plots the path needed for the AI to chase the Player.

BitCollection.cs

Handles the collision between Player and the Bits and Big Bits. Add a certain point value depending on the Bit collided.

PlayerHUD.cs

Handles the HUD components of the Player and updates itself according to the situation, ie, minus Lives when an AI collides with the Player and gain points when Player collects Bits.

PlayerScript.cs

Handles the Player’s invulnerability mode which allows Player to eat an AI, add points depending on Bit collided, flickering of colour when in invulnerability mode and toggling between invulnerability mode and back within a set duration.

ShadowScript.cs

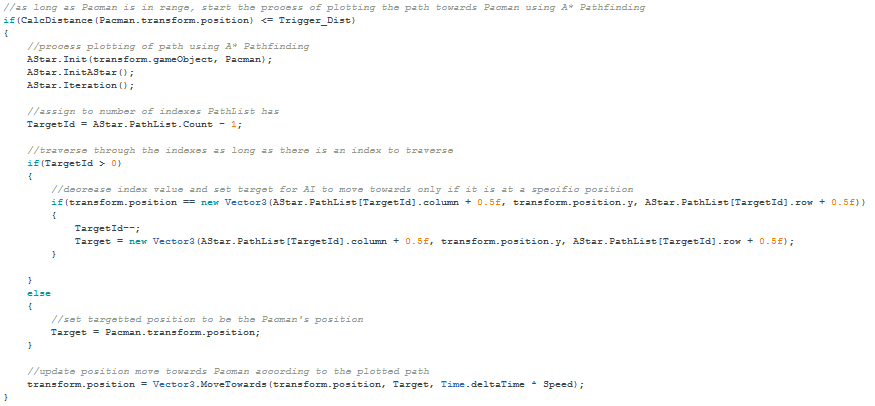
Handles the movement of the AI using A\* Pathfinding. AI will chase the Player based on the plotted path. Does so only if the Player is within a set range.

TileScript.cs

Contains data for each tile in the level and assists in processing A\* Pathfinding.

**A\* AI**

A\* Pathfinding is integrated into the AI to allow the AI to have a path plotted towards the Player in order for it to chase the Player.



CalcDistance() calculates the distance between the AI and the Player.

AStar.Init() takes in the data of the AI and the Player and manipulates them for use in A\* Pathfinding.

AStar.InitAStar() adds the current tile the AI is in to the OpenList.

AStar.Iteration()performs the iteration to add tiles into OpenList and CloseList and to plot path from AI to Pacman.