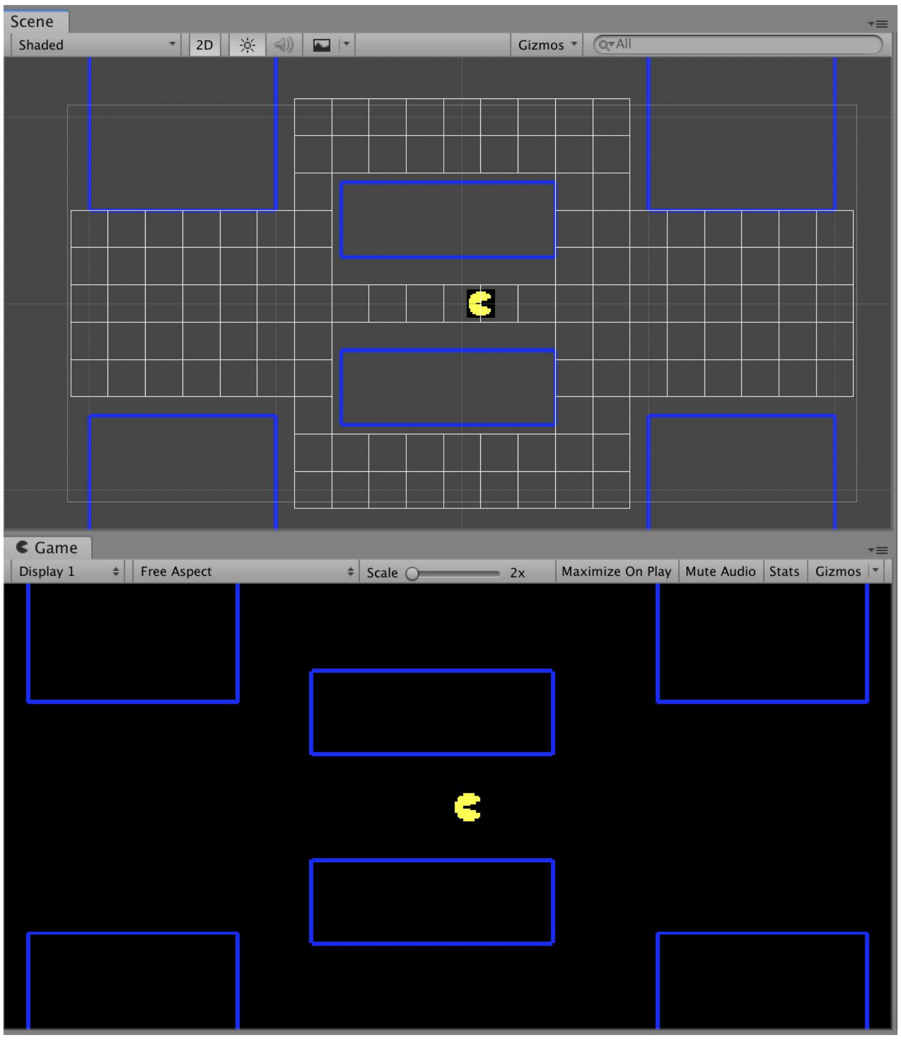
Assignment 1: Grid navigation

The purpose of this exercise is to acquaint you with the Unity game engine we will be using in the class.

One of the main uses of artificial intelligence in games is to perform path planning, the search for a sequence of movements through the virtual environment that gets an agent from one location to another without running into any obstacles. For now we will assume static obstacles. In order for an agent to engage in path planning, there must be a topography for the agent to traverse that is represented in a form that can be efficiently reasoned about. The simplest topography is a grid. Think of an imaginary lattice of cells superimposed over an environment such that an agent can be in one cell at a time. Moving in a grid is relatively straightforward: from any cell, an agent can traverse to any of its four (or eight) neighboring cells.



In this assignment, you will write the code to superimpose a grid over any given terrain so that an agent can navigate by moving left, right, up, or down from cell to cell. The code to generate the grid should work on any terrain such that an agent can never collide with an obstacle.

But first, you need to become familiar with the Unity game engine in which you will be working.

What you need to know

The game engine is script-based. The primary script you will interact with is **ObstacleHandler.cs**, which is a container for all the obstacles. Most importantly, the ObstacleHandler object contains the terrain of the virtual environment. The terrain is represented as a list of **Polygon** objects, which themselves are polygons---lists of points such that there is a line between every adjacent point (and the first and last points in the list.

Below are the important bits of information about scripts that you will be working with or need to know about for this assignment.

ObstacleHandler.cs

You can interact with the ObstacleHandler via calling ObstacleHandler.Instance.X where X is a name of a function or public variable.

Member public variables:

- Obstacles: a public array of Polygons that represents all the obstacles

Member functions:

- AnyIntersect(Vector2 pt1, Vector2 pt2): checks to see whether any of the obstacles intersect the line made by these two points.
- AnyIntersect(Vector3 pt1, Vector3 pt2): same as above, but only checks the x and y values of these vectors
- PointInObstacles(Vector2 pnt): returns true if pnt is in any obstacle
- GetMapCorners(): returns the corners of the map as an array of Vector2s
- GetObstaclePoints(): returns an array of Vector2s representing all of the obstacle points

Polygon.cs

Member public variables:

- Points: a public array of Vector2s that represents all the points of this polygon

Member functions:

- SetPoints(): sets an array of Vector2 points to this polygon’s points, the points must be listed such that they do not form a concave shape.
- GetLines(): returns a 2D array of Vector2 points, where each index stores an array representing a pair of Vector2 points that compose a line
- AnyIntersect(Vector2 pt1, Vector2 pt2): returns true if the polygon intersects with the line defined by these two points
- ContainsPoint(Vector2 pnt): returns true if this point (pnt) is in the polygon
- RenderObstacle(): renders the obstacle to the screen

Instructions

You must superimpose a grid over an arbitrary, given game world terrain consisting of obstacles. The grid is a dictionary stored in GridHandler.cs that contains all of locations that the agent is allowed to exist.

When you click on the screen, you indicate where you want the Agent to traverse.

Step 1: Download and install Unity https://unity3d.com/get-unity/download

Step 2: Download the Assignment1.zip file from eclass, unzip it, and open it via Unity. You can open it by either selecting the unzipped folder from Unity or double clicking Scene1-5.

Step 3: Open Scene1-5 located inside Assets/Hw1. Hit the play button. Click inside the game window to see the pacman agent traverse the game world.

Step 4: Modify GridHandler.cs to complete the CreateNodes() function (see the commented lines of where to edit).

Inside CreateNodes() you can reference the ObstacleHandler object via ObstacleHandler.Instance.X where X is a function or public variable.

Step 5: Test your implementation across the five given scenes (Scene1-5)

Additional testing can be done by changing the maps by modifying the ObstacleTestPoints scripts.

Grading

This homework assignments is worth 10 points. Your solution will be graded by an autograder. The autograder will look for grid cells that intersect with obstacles or go beyond the boundaries of the map. For every map that your solution is tested against, 1 points will be deducted if at least one cell intersects with an obstacle or goes outside the boundaries of the map. The autograder will test your solution on 10 maps, the five provided maps and five withheld test maps.

Hints

Debugging within the game engine can be hard. Print statements (Debug.Log(“”) will be one possible way of figuring out what is going on.

In general, you should be able to get all the data you need from the ObstacleHandler obstacle. If you find yourself directly accessing member variables of other objects, you may want to rethink your approach.

It is good to test your techniques on new maps. If you want to make new maps, modify one of the provided ObstacleTestPoints (e.g. ObstacleTest1Points.cs). This will alter the obstacles in the same numbered scene.

Submission

To submit your solution, upload your modified GridHandler.cs. All work should be done within this file.

You should not modify any other files in the game engine.

DO NOT upload the entire game engine.