Programming Assignment 2
Detailed Instructions

Overview

In this programming assignment, you're building a lazy professor graph traversal game.

To give you some help in approaching your work for this assignment, I've provided the steps I implemented when building my assignment solution.

Step 1: Build the Graph

For this step, you're building the graph.

I provided a stub for a `GraphBuilder` script, but I didn't attach the script to a game object. Attach the script to a game object in the scene so it will run when the game starts.

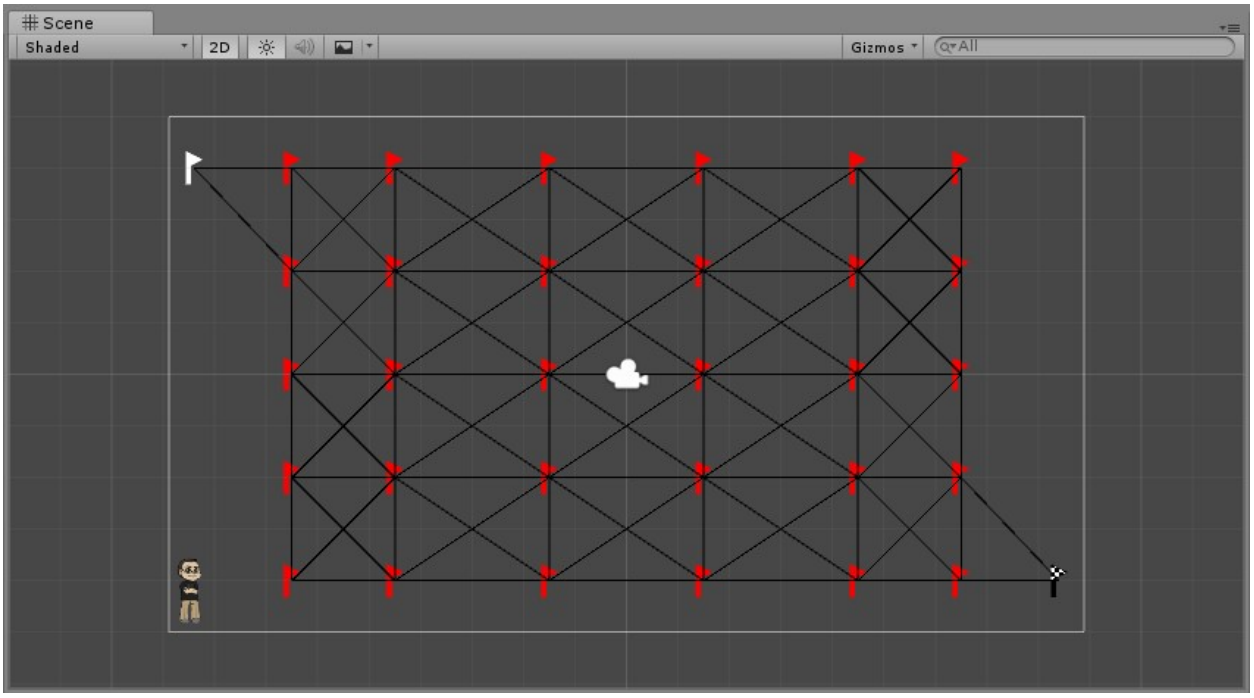
Add the code to the script as indicated by the comments in the `Awake` method. Using the `Awake` method ensure the graph is built before anyone (like the traveler) accesses it in their `Start` method.

Make sure you add the Start object, the End object, and all the Waypoint objects to the graph. You'll probably find the tags for those objects useful as you do that.

Add neighbors for each node in the graph, making sure you only make one node a neighbor of another node if they're within 3.5 units horizontally and 3 units vertically of each other. Make sure the weight is set to the distance between the two nodes when you call the `GraphNode AddNeighbor` method.

Attach the EdgeRenderer script I provided to the main camera. When you run the game, you should see all the edges for the graph drawn in the scene.

Your graph should look like the picture below.



Step 2: Find Shortest Path from Start to End

For this step, you're finding the shortest path (in other words, the path with the least cost) from the start to the end in the graph.

Before you start thinking about that, you need to implement the method bodies in the `SortedLinkedList` class I provided to you. You'll use that class as you finish implementing this step, and the automated grader has numerous test cases that test your `SortedLinkedList` class.

Although we know that breadth-first search guarantees that the first path we find will have the least number of edges, it doesn't guarantee that the first path we find will have the minimum cost in terms of the edge weights.

One pathfinding algorithm that's guaranteed to find an optimal solution in terms of total cost is called Dijkstra's algorithm (named after Edsgar Dijkstra, a giant in the Computer Science domain).

Dijkstra's algorithm is like breadth-first search, but we'll approach it by keeping references to graph nodes (waypoints) with some extra information in a sorted linked list, then build the shortest path by exploring each node exactly once. When we get to the target, we're done.

The discussion of Dijkstra's algorithm on Wikipedia is really good; in fact, I've modified the algorithm from Wikipedia and provided that detailed algorithm in as comments in the `Search` method in the `Traveler` class I provided, since the Traveler is the game object that will follow the path.

Hint 1: Build a smaller scene with a Start waypoint, an End waypoint, a few "normal" waypoints, and a Traveler. Do your testing/debugging on this smaller scene so you don't have to deal with so many waypoints as you make sure your code works. Be sure to attach your GraphBuilder script to the main camera in this scene.

Hint 2: Setting the ID (in the Inspector) for each waypoint in the scene makes it much easier to tell what waypoint you're looking at as you test/debug. I did this for you in the scene I provided, but you'll have to do it yourself in the smaller scene you build.

Hint 3: Write a method that converts the search list to a string so you can easily look at the order of the nodes in the search list. I didn't put every piece of information about each search node into the string when I built one, I just included the waypoint ID and the distance for each search node.

Step 3: Have the traveler follow the path

For this step, you're having the traveler follow the path you found from the start node to the end node. Move the traveler to the start node, then have them follow the path. Each time the traveler collides with a waypoint along the path, turn that waypoint green.

The Ted the Collector game from the Arrays and Lists lesson in the first module of the More C# Programming and Unity course (the second course in the specialization) uses similar ideas for moving a collector to a series of pickups. If you haven't taken that course yet, that game is also covered in Chapter 9 of my Beginning C# Programming with Unity book, which I've provided to you in a reading in Week 1. You can download the Ted the Collector code from http://www.burningteddy.com/Books/Default.aspx and look through it on your own.

Step 4: Stop drawing edges and blow up intermediate waypoints

For this step, you stop drawing the graph edges in the scene. You're also blowing up all the waypoints the traveler visited EXCEPT the start and end waypoints. This all happens when the traveler reaches the end waypoint in the graph.

For the graph edges, you should consider using the event system I've provided. I've provided an Explosion prefab that you should use as you see fit for blowing up the intermediate waypoints.

That's the end of this assignment.