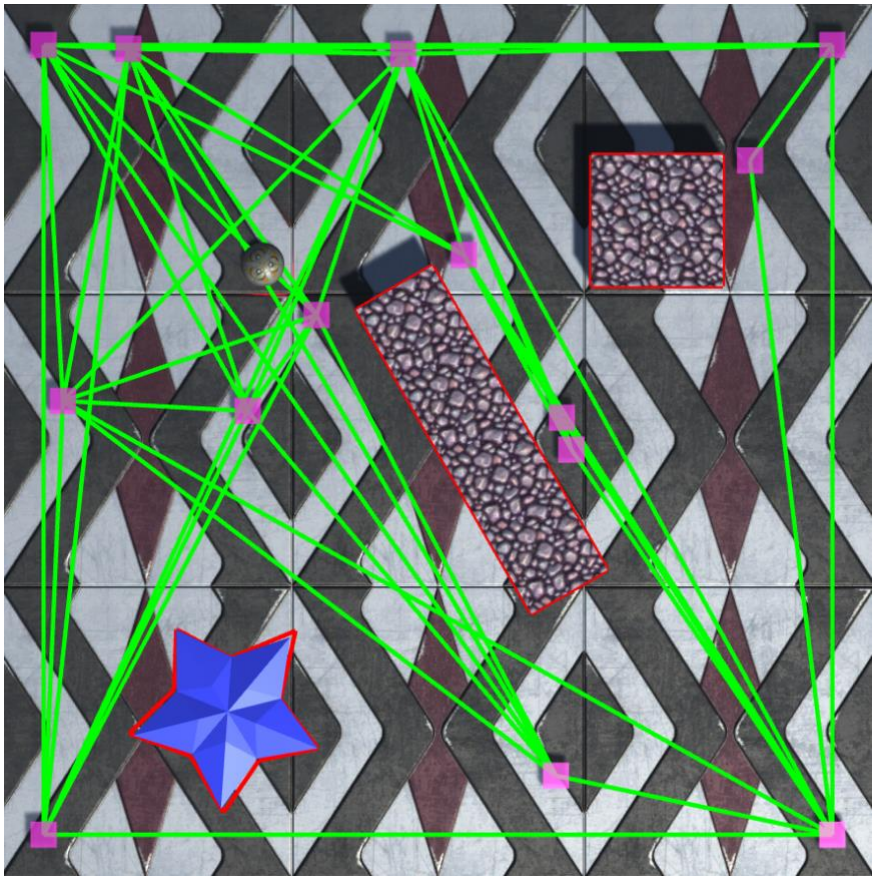
Homework 2: Path Network Navigation

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.

Grid topologies discretize the environment and usually assumes an agent can be in one discrete cell or another. However, for many games such as 1st-person shooters, a more continuous model of space is beneficial. Depending on the granularity of the grid, a lot of space around obstacles becomes inaccessible in grid-based approaches. Finally, grids result in unnecessarily large number of cell transitions.



A **path network** is a set of path nodes and edges that facilitates obstacle avoidance. The path network discretizes a continuous space into a small number of points and edges that allow transitions between points. However, unlike a grid topology, a path network does not require the agent to be at one of the path nodes at all times. The agent can be at any point in the terrain. When the agent needs to move to a different location and an obstacle is in the way, the agent can move to the nearest path node accessible by straight-line movement and then find a path through the edges of the path network to another path node near to the desired destination.

In this assignment, you will be provided with different terrains with obstacles and hard-coded path nodes. You must write the code to generate the path network, as a set of edges between path nodes. An edge between path nodes exists when (a) there is no obstacle between the two path nodes, and (b) there is sufficient space on either side of the edge so that an agent can follow the line without colliding with any obstacles.

What you need to know

Please consult **homework 1** for background on the Unity Project. In addition to the information about the game engine provided there, the following applies.

CreatePathNetwork

This is the only file you will be modifying and submitting for this homework. It provided functionality to create a path network from provided pathNodes.

String StudentAuthorName – Please change to your name

Create()

This is the method you will be finishing. You can create helper methods in the same source code file if you like.

Parameters:

- Vector2 canvasOrigin – Bottom left corner of navigable space
- float canvasWidth – Width of navigable space
- float canvasHeight – Height of navigable space
- List<Obstacle> obstacles – The obstacles that obstruct candidate pathEdges between nodes. Furthermore, corners cannot be within agentRadius distance (less than test) of a candidate pathEdges
- float agentRadius – The radius of the agent (don't expect to always be the same)
- List<Vector2> pathNodes – The nodes of the graph. You do not generate these. They are provided.
- out List<List<int>> pathEdges – A list of valid edges, indexing the pathNodes. This is generated by your code. All valid edges should be generated

PathNetwork

The PathNetwork calls your CreatePathNetwork.Create() to generate the path network and also performs visualization and other related tasks.

Obstacle

Each obstacle in the plane is a polygon through which Agents cannot move.
Member functions:

- GetPoints(): returns a list of all corners in the polygon. A point is a Vector2 of the form (x, y).
- GetLines(): returns a list of all lines in the polygon. A line is an array of the form (point1, point2) where points are Vector2 of the form (x, y).
- IsPointInPolygon(point): returns True if a point (x, y) is inside the obstacle.

Miscellaneous utility functions

Miscellaneous utility functions are found in Utils.cs.

- DistanceToLine(Vector2 point, Vector2 lineStart, Vector2 lineEnd): returns the shortest distance between a point (x, y) and a line (lineStart, lineEnd).
- Intersects(point a1, point a2, point b1, point b2) - Returns true if the line defined by the (a1,a2) intersects with the line defined by (b1,b2)

Instructions

To complete this assignment, you must (1) implement code to generate a path network for a set of given path nodes in a given terrain. The path network should guarantee that an agent will not collide with an obstacle between any starting point and any destination point in the world.

To run the project code, use the following commands:
> **Step 1:** Download the project from github. Open the project in Unity. Open scene PathNetwork
> **Step 2:** If you hit play, you should see a list of path nodes and some obstacles not connected to each other. You can click the obstacles and path nodes to drag them around. You can press buttons 1,2,3,… to test some preconfigured obstacle/pathNode positions. Clicking outside of an obstacle initiates path planning. Spacebar changes search strategies. Note that any search that relies on AStar won't function until you implement it.
> Step 3: Create the edges in the path network.
Open Assets/Scripts/GameAIStudentWork/PathNetwork/CreatePathNetwork.cs. Update the name string first! Implement the functionality to generate a valid list of PathEdges connecting PathNodes that are not obstructed from each other. Also, PathEdges cannot be too close for the agent to fit by while following an edge.

Grading

We will grade your solution based on the following criterion:

- **Reachability:** The path network should be such that an agent can navigate from any path node to any other path node along any edge in the network without colliding with an obstacle.

Hints

Make sure any edge in the path network is traversable by an agent that has physical size. That is, edges in the path network should never come too close to any Obstacle such that an agent blindly following the path edge collides with an Obstacle (or the edges of the map).

Create additional presets by adding more configurations to
Assets/Scripts/Config/CustomPresetConfig.cs
However, you do not submit this file.

Submission

To submit your solution, upload your modified CreatePathNetwork.cs with your name properly set in the name string. All work should be done within this file.

You should not modify any other files in the game engine (other than optionally CustomPresetConfig.cs for your own testing purposes).

DO NOT upload the entire game engine.