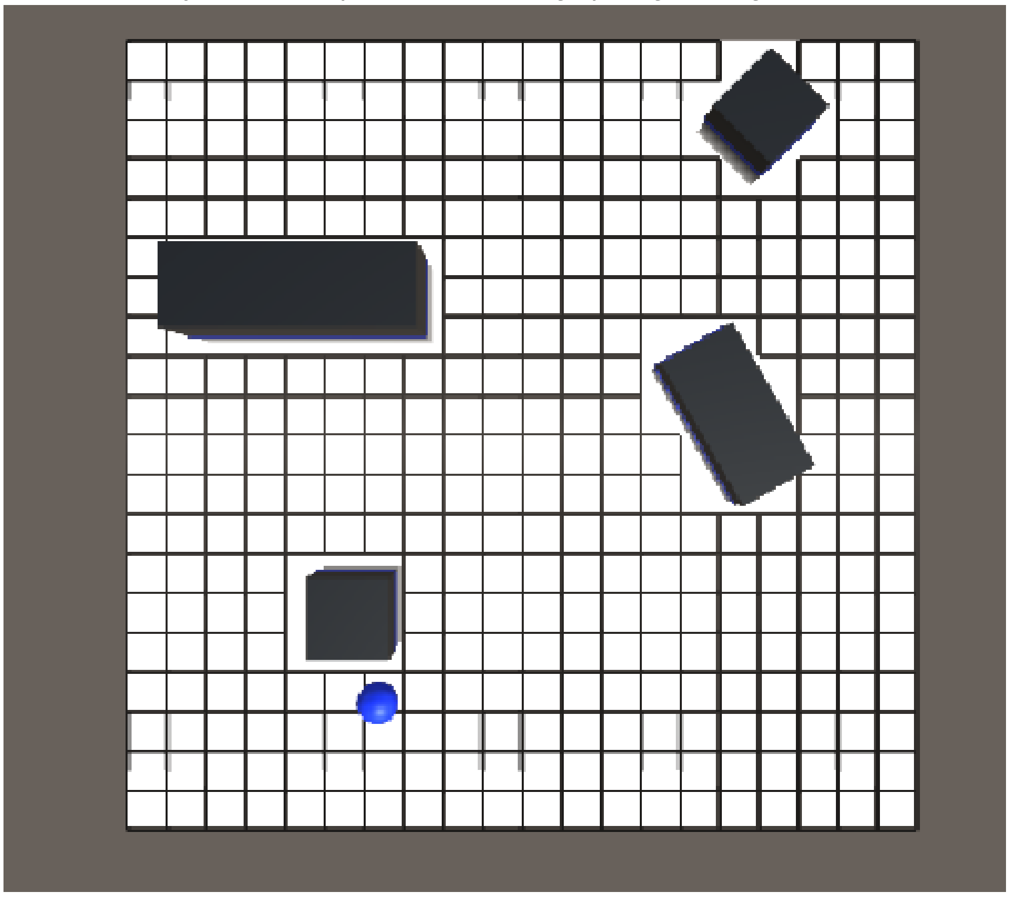
Homework 1

Homework 1: Grid Navigation

The purpose of this exercise is to acquaint you with the Unity game engine we will be using in the class as well as some basics of supporting agent movement in a simulated environment.
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 (e.g. a *discretized space*). 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 sometimes 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 (4-way) 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

In Unity, we work with Game Objects to achieve the behaviour we desire. You can see a list of these Game Objects in the Hierarchy tab. The terrain on which you'll be working is represented using the 'NavigationArea' game object (a plane geometry). Each Game Object can have multiple scriptable components attached to them. By 'Scriptable', we mean that the components behaviour can be programmed. You can see a list of these components in the Inspector view when you click on the 'NavigationArea' Game object.

For this assignment, you will be editing a file that the 'Game Grid' component calls. In the project view, you can expand Assets/Scripts/GameAIStudentWork/GridNavigation/ to find the 'CreateGrid' file. The ../../FrameWork/'GameGrid' file calls 'CreateGrid' in order to create the grid overlay data structures. If you double click the Script, this will open the C# Script in an editor.

There are other objects within the scene that are worth exploring including the agent and the obstacles.

Every iteration of the game loop, called a *tick*, the Update() method is called on all dynamic objects, their components and the scripts associated with them.

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

CreateGrid

Location: Assets/GameAIStudentWork/GridNavigation/CreateGrid.cs
Member functions:

- Create(): This is the method that you need to fill out. It should pass back a grid, which is represented using a two-dimensional Boolean array. For each element in the grid, it is true if the sphere can pass through it and false if it cannot. You also need to create a graph of nodes and edges that corresponds with the grid. Assume 4-way adjacency between cells (Up, Down, Left, Right). No diagonals.

Parameters:

canvasOrigin: bottom left corner of navigable region in world coordinates

canvasWidth: width of navigable region in world dimensions

canvasHeight: height of navigable region in world dimensions

cellWidth: target cell width (of a grid cell) in world dimensions

obstacles: a list of collider obstacles

grid (out): an array of bools. row major. a cell is true if navigable, false otherwise

pathNodes (out): a list of graph nodes, centered on each cell

pathEdges (out): graph adjacency list for each graph node. corresponding index of pathNodes to match node with its edge list. All nodes must have an edge list (no null list) entries in each edge list are indices into pathNodes

Game Grid

Location: Assets/Scripts/Framework/GameGrid.cs

You don't actually modify this file but be aware that GameGrid calls CreateGrid.Create() (your static function) to actually create the grid. It also dictates what arguments are passed to your function.

Presets

Location: Assets/Scripts/Config/CustomPresetConfig.cs

This is another file that you aren't required to modify (regardless, you won't be turning it in). However, you may find it useful to add special presets for debugging your CreateGrid.Create() code. Certainly, you should test with all presets provided via the defined numerical keypresses!

Obstacle

Each obstacle in the plane is a polygon (leverages Polygon class) 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.

- 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)

In addition, you might want to check the array and list methods of C# and Unity Engine's Vector2 class. Also, C# System.Tuple may be useful.

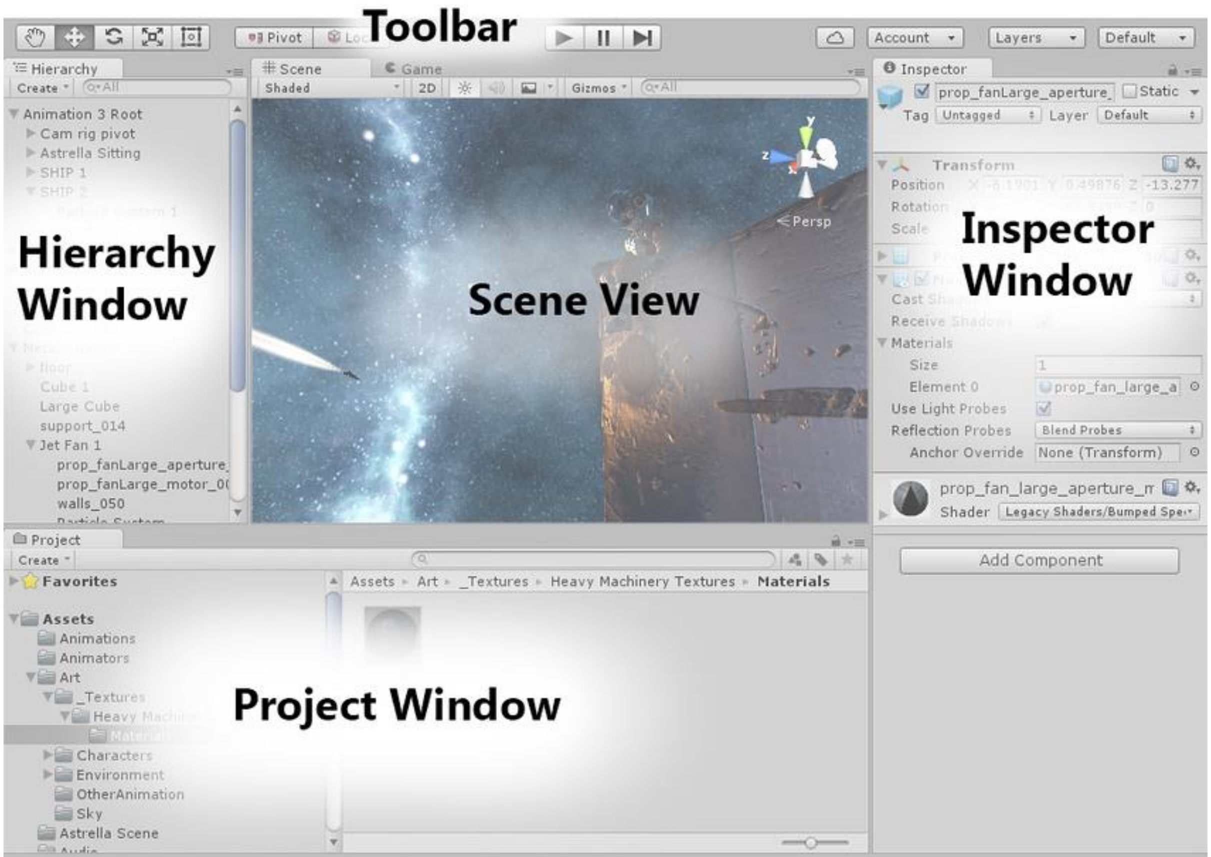
- https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netframework-4.8
- https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/

- https://docs.unity3d.com/ScriptReference/Vector2.html
- https://docs.unity3d.com/Manual/index.html
- https://docs.unity3d.com/ScriptReference/

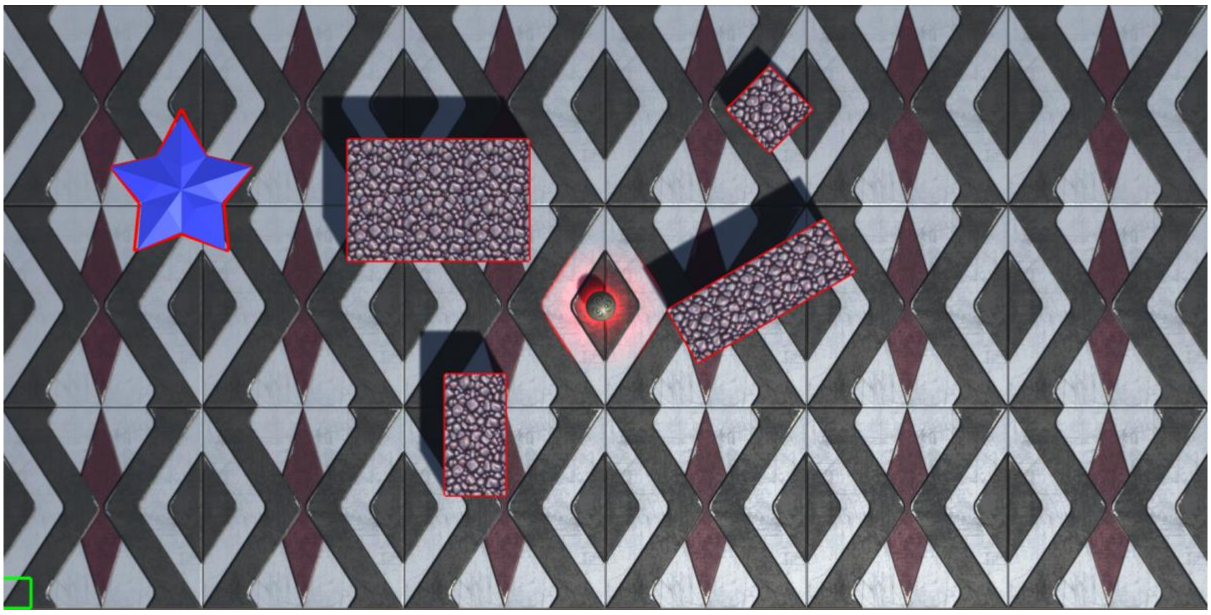
Instructions

You must superimpose a grid over an arbitrary, given game world terrain consisting of obstacles. The grid is a 2D array of Booleans such that a *false* in any particular cell means the Agent cannot walk into the cell and a *true* in any particular cell means the Agent can walk into the cell. The Navigator that we use in this assignment does not guarantee delivery of an Agent to their final, desired destination.
When you click on the screen, you indicate where you want the Agent to traverse.
Step 1: Download and install Unity.
https://store.unity.com/download?ref=personal
Step 2: Download the Unity project for this assignment
https://github.gatech.edu/IMTC/GameAIAssignment1/tree/master

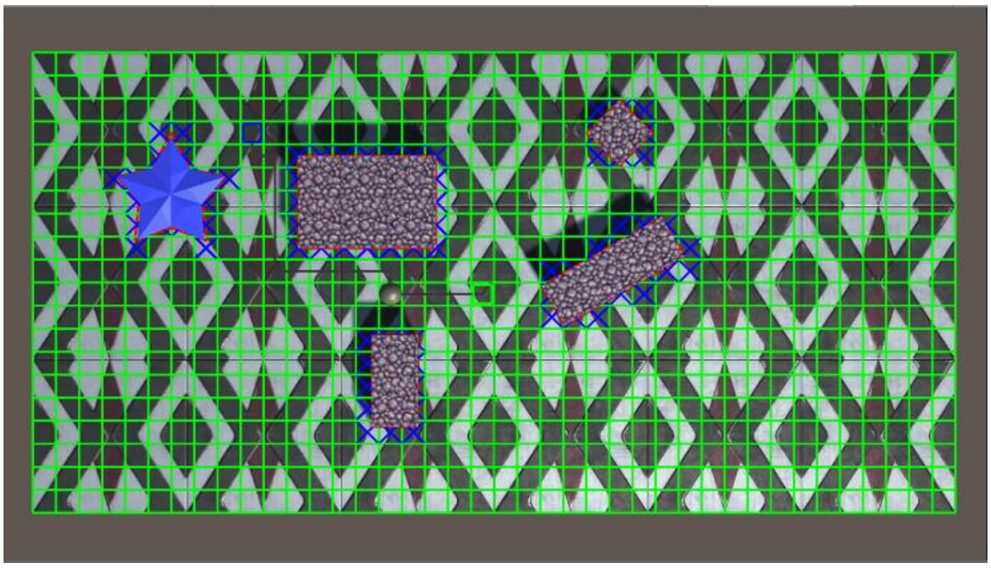
Step 3: Open Unity and load this project. Click on File -> Open Project and select the project folder for this assignment
Step 4: If not already open, open GridNavigation. Click on File -> Open Scene and select Scenes/Gridnavigiation. These are the major components of the UI.



In the toolbar click the ‘play’ button. You should now be in the game view with a screen like the following:



You can click and drag the obstacles using the left mouse button. If you click on the plane, the Agent will attempt to navigate there via the graph. However, the limited placeholder code won't allow for much functionality until you complete implementation. You can press buttons 1,2,3,.. to select some preset configurations of obstacles.
Here is a screenshot of a working implementation:



Note that a properly populated grid will show boxes for navigable cells and X's for blocked cells. This rendering code is already provided and should work automatically for you.

Step 5: Navigate through the Project Window Assets and double-click on the CreateGrid.cs file. This should open up your IDE to edit the file. You have to now fill in the Create() method of this file such that:

1. It must create and return a 2D array of Booleans such that grid[column][row] indicates the traversability of the cell at (column, row).

2. Create the associated graph nodes and edges. (Note that a typical grid lattice implementation would not require explicit creation of the graph. However, to synchronize visualization code across assignments we require it. It’s also good practice to become familiar with these structures now before upcoming assignments.)

Step 5: Test your implementation:
Click the play button at the top of your toolbar. Drag and move the obstacles around and try preset configuration. Click on a point on the grid to see that the sphere starts moving towards it by traversing the grid. Note that the Agent may not always be able to navigate due to the Greedy search utilized.

Grading

Your solution will be graded by autograder. The autograder will look for grid cells that intersect with obstacles. The autograder will test your solution on several maps, some of which are provided as test maps in the presets.

Hints

Carefully consider all possible geometry situations you may encounter. Not just what is shown in the presets!

Debugging within the game engine can be hard. Print statements will be one possible way of figuring out what is going on (print() or Debug.Log()/Error()). You can also use the Visual Studio debugger to debug your code if you’ve installed the Unity plugin for it.

Submission

To submit your solution, upload your modified CreateGrid.cs file **with correctly set student name string**. All work should be done within this file. Don’t forget to change the student name string!

You should not modify any other files in the game engine. You may modify the presets but don’t turn them in.

DO NOT upload the entire game engine.