# Homework 4: NavMesh Generation

In this homework you will be implementing an algorithm for creating NavMeshes.

Make sure to watch the NavMesh lecture before attempting this assignment.

A **navigation mesh** (or **navmesh**) is a method of creating a discretized space representation of a navigable area. A navigation mesh is a set of convex polygons overlaid on an environment such that the area within each polygon is guaranteed to be obstacle-free. The variable size and dimensions of navmesh polygons provide efficient representation of space as compared to the uniform structure of a grid lattice. It also captures more usable information about the environment than a path network alone. The convexity of the polygons is important because an agent within the area of a polygon can move to any other point within without crossing a boundary (e.g. colliding).
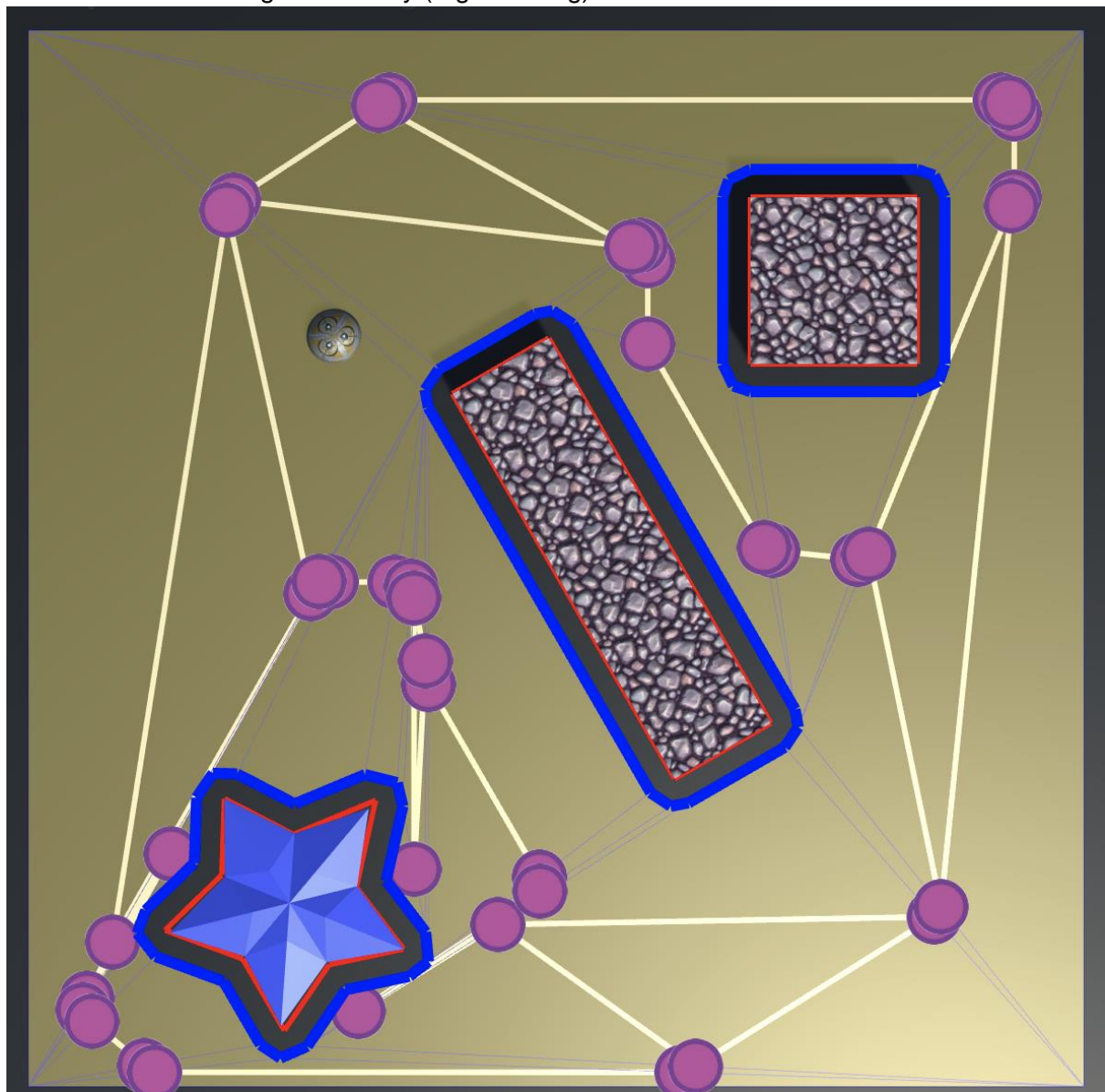
When two navmesh convex polygons are adjacent to each other (i.e., they share an edge), that edge can be thought of as a "portal"—an invisible door—from one safe navigation region to another. Connecting adjacent convex polygons into a network of safe paths results in a path graph through which an agent can travel between connected navigable locations.

There are various ways to convert a navmesh into a graph with path locations (e.g. a path network). A robust approach is to the use the midpoint of each portal edge to create each pathNode. These pathNodes are connected to all other portal edge midpoints within the same navmesh polygon.

In this assignment, you will write the code to generate a navmesh for an arbitrary environment as well as an accompanying path network from the navmesh.

There are three main challenges of the assignment:

1. Form adjacent triangles that cover all the navigable space
2. Merge triangles into larger, more efficient polygons that are also convex
3. Create a path graph from the navmesh

What you need to know

Please consult **homework 1** for background on the Game Engine. In addition to the information about the game engine provided there, the following elements will be used.

Also, make sure to watch the NavMesh lecture before attempting this assignment.

CreateNavMesh

This file contains a Create() method that you must implement. Additionally, be sure to change the student name string. Considerable scaffolding and comments are provided, which will aid in development of your solution.

Create(): Creates a navmesh and pathNetwork (associated with navmesh)
    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
    obstacles: a list of CCW Polygons that are obstacles in the scene. These are already
          expanded and clipped to the canvas. No holes are present in the polygons,
          but are possibly concave.*agentRadius*: the radius of the agent
    origTriangles: out param of the triangles that are used for navmesh generation. These
          triangles are passed out for validation and visualization
    navmeshPolygons: out param of the convex polygons of the navmesh (list).
      These polys are passed out for validation and visualization and should be merged
    adjPolys: out param of type AdjacentPolygons. This is a dictionary that maps
          CommonPolygonEdges to CommonPolygons. Used for validation and
          should be merged

pathNodes: out param: a list of graph nodes, centered on each portal edge of the navmesh

pathEdges: out param: 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. All edges should have a matching edge going opposite direction and no duplicate edges.

The scaffolding and inline comments of CreateNavMesh.Create() gives specific guidance and recommendations on other classes and methods to use (including methods at the top of the file).

Instructions

Download the project from Github and open in Unity. Open the Navmesh scene and the CreateNavMesh.cs file. Follow the comments in CreateNavMesh.Create() to build a working navmesh generator. **Don't forget to set the student name to your name.**

Note: There is an assignment demonstration video on Canvas, but be aware that it demonstrates different versions of the assignment. Be sure and pay attention to the requirements for this semester, particularly related to node placement in navmesh polygons.

Grading

We will grade your solution based generally on three criteria:

- **Coverage:** The navmesh should cover the entire navigable area of the map, no more and no less.

- **Reachability:** The path network should include pathNodes at the midpoint of each portal edge of the navmesh polygons and pathEdges should connect with the pathNodes of other portal edges within the same navmesh polygon. Valid edges must have a corresponding edge in opposite direction and there must not be duplicates of the same edge. Every node must have a non-null edge list. Nodes should not link to self. You should not need test for obstacle distance from edges like in HW2. Instead only use the navmesh polygons to determine node connectivity.

- **Mesh optimization:** Your solution should effectively merge navmesh triangles (or more sided polygons) together resulting in a reduction of pathNetwork complexity

Please remove all print statements before submitting. The autograder will only provide a few hundred lines of feedback and you might overflow the buffer so that the informative part doesn't show up. Also, print statements can cause significant slowdown such that you might fail a test due to timeout (see below). When you remove print statements, please test your

code again! Quite often we receive assignments that don't compile due to a hanging *if-statement* where a print() was the consequent.

Your code will be allowed at least 20 seconds to complete each test.

Submission

To submit your solution, upload your modified CreateNavMesh.cs (did you remember to change the student name string to your name?). All work should be done within this file. Helper methods within the same file are fine.

You should not modify any other files in the game engine.
DO NOT upload the entire game engine.

Refer to the Canvas assignment description for submission details (most likely GradeScope).