CSC3222 Project 2

Welcome to the practical classes for CSC3222. It is essential that we, as programmers, apply the theories we learn in a variety of engineering situations. Understanding the theory is an essential skill, but understanding the limitations of the theory in an implementation sense is just as important as you move forward with your software development career. The purpose of these practicals is to provide some experience of that, and to generate a platform from which you can launch your own exploration of the concepts discussed in the lecture series.

Module Assessment

There are 3 pieces of assessment for this module:

1. Project 1 which is submitted to NESS. (35)
2. **Project 2 which is submitted to NESS. (15)**
3. An exam in June. (50)

Project 2 (Deadline: 11:45pm, Wednesday 4th May, 2016)

Aims

Produce a heuristic path-planner which will find the lowest-cost path from any selected node to any selected node on a spheroid map represented as sixty nodes, inspired by buckminsterfullerene. This path-planner should use the A* algorithm, and account for different edge costs.

Scenario

Consider the polyhedron shown below:

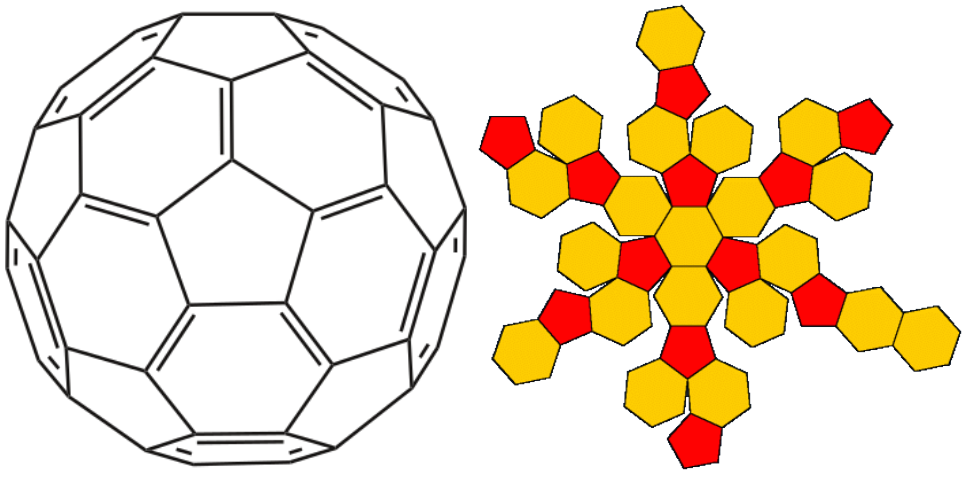

Figure 1: Buckminsterfullerene (Spherical and Exploded)

Buckminsterfullerene, or bucky-ball, is a spherical molecule formed of 60 carbon atoms which make up a shape comprised of 12 pentagons and 20 hexagons, resembling a football (or soccer ball). In this sense, it is a perfect means to represent a spherical environment (such as a planet) in a video game.

Each node has three connecting edges. The coordinates of the nodes are downloadable as a separate document, to save you having to work them out for yourselves; you do have to work out which nodes are connected, however. The arrangement has 90 edges. By default, those edges should have a travel cost of 1. Please note that the exploded diagram duplicates many nodes and edges, and is a visual aid rather than a schematic.

Minimum Deliverable Software Specification (Up to 9 Marks)

Encode the 60 nodes of buckminsterfullerene, and their connecting edges, as a node graph. There must be some software indication (e.g. a command line instruction) which lists which edges are connected to a selected node. Edges should use identifiers E0-E89.

Write a piece of software in C++, built around an A* implementation, which will accept a start node and a goal node (in terms of the indices 0-59, found in the companion text document), and return the lowest cost path between those two nodes, and the cost of that path. This does NOT need to be done graphically; command line input and return are acceptable. The path returned should list the nodes and edges traversed (e.g. 0->E0->2->E14… etc.).

Implement an extension to the above which permits the selection of a given edge, and allows the user to change that edge's traversal cost from the default of 1, to any other integer value. Again, this does NOT need to be done graphically; command line input is acceptable. Until the program restarts, costs should be calculated based on the updated traversal cost. The extension should also allow the user to make an edge 'impassable', meaning that no path will be returned which uses that edge.

The data regarding the cost of an edge, and whether or not that edge can be traversed, should be encoded separately from the node geography and connections, possibly as a second map solely addressing this property for each edge. It is not enough when making an edge impassable to simply delete that edge from the node graph; this should instead be some property of the edge (e.g., a Boolean).

The submitted software must compile and execute; keep in mind that this means you must employ relative pathing when linking libraries, etc., and it is advisable to test the software on another PC before uploading it to NESS.

Software Extension (Up to 6 Marks):

Extend your software to render the map graphically. When a node is selected (be it via command line, or point-click interface), its connected edges should be highlighted. When an edge is selected (again, via command line or point-click interface), the nodes to which it is connected should be highlighted. The colour of an edge should be related to its current cost.

When a path is computed, that path should be displayed on the map's graphical representation. Edges which cannot be traversed should not be displayed. Nodes with no traversable edges should not be displayed.

Deliverables

Zipped Microsoft Visual Studio Project, Source Code and Executable (submitted via NESS)
Demonstration of Working Software (to take place on Thursday 5th May at 11AM in the Rack, or earlier by individual arrangement)

Mark Scheme

Submitted Software:	15
Total:	15