CSC 584/484 Spring 17 Homework 2: Pathfinding and Path Following

Due: 3/22/17 by the start of class

Overview

Your task for this assignment is to explore some of the **pathfinding** and **path following** algorithms we discussed in class. Working alone and using *Processing* (`http://www.processing.org/`), you will implement the algorithms described below and analyze your results in a 3–5 page writeup. Note that for some of these tasks you will be integrating your new solutions with your code from Assignment 1.

First Steps (10pts)

Create two graphs that you will use for your experiments. Both graphs should be a weighted, singly-connected digraph. This means for any ordered pair of vertices (v_i, v_j) there is at most one weighted $w_{i,j}$ edge. Remember to only use positive weights.

Your first graph should represent something meaningful in the world. Perhaps a road map of Raleigh or your hometown, the offices in EB2, *etc*. Your choice of what you represent is up to you. Hint: read ahead to figure out how to maximize your work effort. This graph should be large enough so that you can learn some interesting things about the algorithms, but small enough to enable efficient computation in your experiments. As a rule of thumb, 10 vertices is probably too few, but 50 is likely too many.

Your second graph should be designed to test the limits of the algorithms. It should be big. Very big. You can generate it randomly, find a graph somewhere on the internet (make sure you cite where!), or try to generate it using some data set of your choosing. One thing is for sure, you should not author this graph by hand. As a rule of thumb, you should be targetting graphs with thousands or tens-of-thousands of vertices and edges.

In your writeup, include a figure of your small graph and a description of your large graph. What does the first one represent? How did you create the second one?

Dijkstra's Algorithm and A* (15pts)

Your next task is to implement both **Dijkstra's Algorithm** and the **A* Algorithm**. Hint: get them both working on your first graph before you begin testing on your second graph. For A*, pick a simple heuristic like a constant guess or some form of Manhattan Distance. If your large graph doesn't have any meaningful distance or way to compute a heuristic, use a random one, constant guess, or the cluster heuristic discussed in class. What you choose isn't all that important for this section of the assignment.

Compare and contrast the performance of the two algorithms on both of your graphs in terms of runtime, number of nodes visited, and memory used. What else can you say about these algorithms? What effects does the graph structure have on performance? You are expected to present data in your writeup to support your analysis.

Heuristics (10pts)

Looking solely at the A* algorithm, design and implement at least **two heuristics** for your first graph. The heuristic you used in the previous portion of the assignment can count as one of these two. A few options are random, hand-authored, Manhattan distance, a constant guess, or euclidean distance. These are not the only possibilities, be creative. These heuristics don't need to work for your second, larger graph.

Describe each of your heuristics in detail. Is is admissible? Consistent? Overestimating? Underestimating? What happens to the performance of A* when you use the different heuristics? Again, present data in your writeup to support your analysis.

Putting it All Together (25pts)

Combine your **pathfinding** algorithms with the appropriate algorithms from Homework 1. Design an "indoor environment" that contains a number of obstacles. Using a principled technique of your choosing (*e.g.*, a **division scheme** we discuss in class), create a graph representation of that environment and use it to perform pathfinding. You should be able to click anywhere in the processing sketch, have that click location be quantized into the graph, and have your A* algorithm compute an efficient path to that location. Have your character **follow the computed path using the pathfollowing algorithm** we discussed in class. Hint: Although more computationally intensive, having a dense graph can avoid the need for obstacle avoidance in your movement system. Videos or screenshots (with breadcrumbs) of your character in action are a must.

Writeup (40pts)

Now that you have implemented and evaluated a number of algorithms, write a 3–5 page single-spaced paper summarizing your findings. That is at least **three full pages**. It is **strongly** suggested that you do not limit yourself to only answering those questions posed in this assignment. Think creatively about what you have done. What other parameters can you tweak and what effect do they have on the results? The most successful writeups will contain evidence that you have thought deeply about these algorithms and what they can produce and have gone beyond what is written in the assignment.

As an appendix to your writeup, please include all relevant screenshots to support your analysis. The appendix does not count toward your 3–5 page requirement.

What to submit

By the start of class on 3/22/17, please upload a .zip archive to moodle. This archive should contain all of your appropriately labeled files from each part of the assignment, a README file, plus your writeup in .pdf format. For this homework, rather than require a specific naming convention for each of your files, please include a README with instructions for compiling and running your code. In the README, you should list which files correspond to which parts of your homework assignment.