CSC 584/484 Spring 22 Homework 3: Pathfinding and Path Following

Due: 03/22/22 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. Using `SFML`, you will implement the algorithms described below and analyze your results in a 2–3 page writeup. *Note that for some of these tasks you will be integrating your new solutions with your code from Assignment 2.*

This is an ***individual assignment***, you are to work alone. As always, you are expected to abide by the University's Academic Integrity Policy (`http://policies.ncsu.edu/policy/pol-11-35-01`), which includes providing appropriate attribution for *all* external sources of information consulted while working on this assignment.

This assignment is worth 30% of your homework grade.

First Steps (4pts)

Create two graphs that you will use for your experiments. Both graphs should be a weighted digraph. 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 100 is likely too many; the sweet spot is probably somewhere between 20 and 50 vertices, and an appropriate number of edges (*i.e.*, three or four times the number of vertices).

Your second graph should be designed to test the limits of the algorithms. It should be big. Very big. You can generate it randomly or using an algorithm/library you find (make sure you cite where you found it!), 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 targeting graphs with thousands or tens-of-thousands of vertices and four or five times that number of edges.

In your writeup, include a figure of your small graph and a description of your large graph. The figure can be a photo of a drawing in your notebook or rendered using a software package of your choosing. What does the first graph represent and why did you choose it (technical reasons please, not personal significance)? How did you create/obtain the second one?

Dijkstra's Algorithm and A* (8pts)

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 this part of the assignment, pick a simple heuristic for A* to get something working. If your large graph doesn't have any meaningful distance to base your heuristic on, come up with something reasonable to use (perhaps the cluster heuristic). What you choose isn't all that important for this section of the assignment provided it is reasonable (*i.e.*, a random heuristic or constant guess aren't sufficient).

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 (4pts)

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 hand-authored, Manhattan distance, euclidean distance, or the cluster heuristic. These are not the only possibilities, be creative. These heuristics don't need to work for your second, larger graph. *One of your heuristics should be admissible and consistent, the other should be inadmissible.*

Describe each of your heuristics in detail. For the inadmissible heuristic, how frequent are the overestimates? By how much does it over-estimate? *Etc.* 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 (7pts)

Combine your **pathfinding** algorithms with the appropriate algorithms from Homework 2 to demonstrate **pathfollowing**. Design an "indoor environment" that contains three or more rooms with a total of three or more obstacles. Using a division scheme of your choosing, create a graph representation of that environment and use it to perform pathfinding. You should be able to click anywhere on the screen, 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. Screenshots (with breadcrumbs) of your character in action are a must.

Writeup (7pts)

Now that you have implemented and evaluated a number of algorithms, write a 2–3 page single-spaced paper summarizing your findings. That is at least **two 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 the first thing that worked.

As an appendix to your writeup, please include all relevant screenshots to support your analysis. The appendix does not count toward your 2–3 page requirement.

What to submit

By the start of class on 03/22/22, please upload a .zip archive to Moodle. This archive should contain all of your appropriately labeled files from each part of the assignment, your Makefil, a README file, plus your writeup in .pdf format. The README must contain explicit instructions for compiling and running your code to demonstrate each part of the assignment. In the README, you should list which files correspond to which parts of your homework assignment.