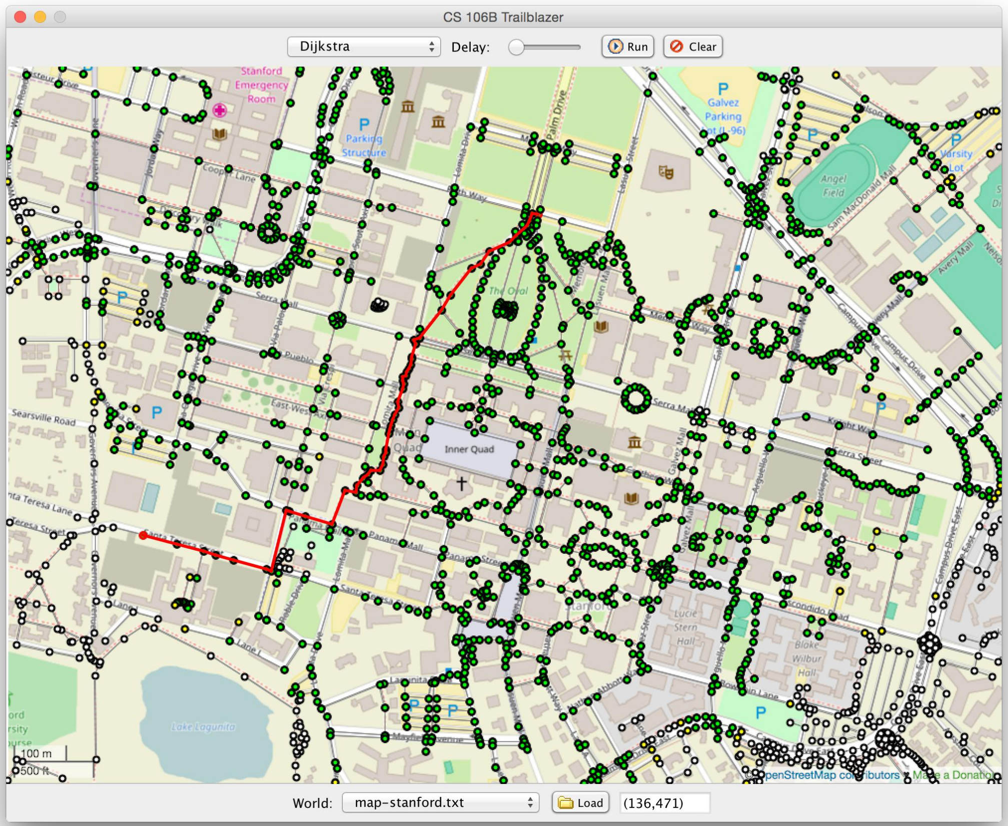
Trailblazer

Overview

This program displays various road maps and allows users to find the shortest path between any two nodes. For example, if the user loaded the Stanford map and selected a node at the top of the Oval and another node at FroSoCo, your program should display the best route:



If you click on any two nodes in the world, the program will find a path from the starting position to the ending position. As it does so, it will color the vertexes green and yellow based on the colors assigned to them by the algorithm. Once the path is found, the program will highlight it and display information about the path cost in the console. The user can select one of four path-searching algorithms in the top menu:

1. Breadth-first search (BFS)
2. Dijkstra's algorithm

The window contains several controls. You can load world maps by selecting them from the bottom drop-down menu and then clicking the "Load" button.

To Do

In your *trailblazer.cpp* file, you must write the following 2 functions for finding paths and creating mazes in a graph:

```
Path breadthFirstSearch(RoadGraph& graph, RoadNode* start, RoadNode* end)
```

```
Path dijkstrasAlgorithm(RoadGraph& graph, RoadNode* start, RoadNode* end)
```

You should search the given graph for a path from the given start vertex to the given end vertex. If you find such a path, the path you return should be a list of all vertexes along that path, with the starting vertex first (index 0 of the vector) and the ending vertex last.

If no path is found, return an empty path. If the start and end vertexes are the same, return a one-element vector containing only that vertex. Though some graphs may be undirected (all edges go both ways), your code should not assume this. You may assume that the graph passed in is not corrupt.

Provided Code

We provide you with a lot of starter code for this assignment. Here is a quick breakdown of what each file contains, though you do not need to examine or know about each file or its contents in order to complete the assignment.

Trailblazer.h/.cpp: We provide a skeleton version of these files where you will write your path-searching code for the assignment.

RoadGraph.h/.cpp Defines and implements the RoadGraph, RoadNode and RoadEdge classes

Color.h/.cpp Constants representing colors of verticies

TrailblazerGUI.h/.cpp The app's graphical user interface.

Main.cpp The main function that initializes the GUI and launches the application.

WorldDisplay.h/.cpp Hierarchy of types of world graphs.

Graph Algorithm Details

Coloring: In addition to searching for a path in each algorithm, we also want you to add some code to give colors to various vertexes at various times. This coloring information is used by the GUI to show the progress of your algorithm and to provide the appearance of animation. To give a color to a vertex, call the color member function on that vertex's RoadNode object, passing it a global color constant such as Color::GRAY, Color::YELLOW, or Color::GREEN. For example:

```
// set the start vertex's color to green
start->setColor(Color::GREEN);
```

Here is a listing of colors available and when you should use them:

- enqueued = yellow: Whenever you enqueue a node to be visited for the first time, such as in BFS and Dijkstra's algorithm when you add a node to a data structure for later processing, color it yellow (Color::YELLOW).
- visited = green: Whenever your algorithm directly visits and examines a particular vertex, such as when it is dequeued from the processing queue in BFS, color it green (Color::GREEN).

The provided GUI has an animation slider that you can drag to set a delay between coloring calls. If the slider is not all the way to its left edge, each call to setColor on a vertex will pause the GUI briefly, causing the appearance of animation so that you can watch your algorithms run.

Hand In

Please hand in *Trailblazer.cpp* in provided code, and rename it to your student id such as 516030910XXX.cpp.

Evaluation

Correctly complete BFS (20')

Correctly complete Dijkstra (20')

Code style (10')

Tools

In order to program your homework, you must download and install an editor named Qt Creator.

Please follow the instructions on(http://web.stanford.edu/class/cs106b//handouts/qt-creator.html).