Car Navigation Program Using GLFW

Final Project of Computer Seminar I

Muhammad Akhdan Fadhilah
ID: B9TB1706
muhammad.akhdan.fadhilah.t3@dc.tohoku.ac.jp

June 2, 2021

1 Introduction

In this final project, I created a car navigation program (mini version of Google Map). The program starts by asking user input whether to choose the points of interest (start and destination) via keyboard or mouse. After it has been decided, the path algorithm, which we use Dijkstra, then determines the shortest path from start to the final destination. Finally, the path is displayed in the GLFW window showing the map, path, and car. The user should push the spacebar for to car to move. After the car reached the destination, we can declare that the program finished. :)

Several code files used in this program are explained below:

- **main.c**, is where the the one and only main function located. It includes the GLFW window initialization.
- **algo.c**, is where all algorithm are stored, e.g. dijkstra algorithm, swap, or some function related to array.
- **display.c**, is where all GLFW display are stored, e.g. how to draw the map or car, read mouse position, or changing map projection.

2 Project Progress

May 20-21	:	Review all codes from lectures and combine it
May 22-23	:	Change the code to the preferred way; Make new **struct.h**
May 24	:	Change basic display (map and path display)
May 25	:	Map scaling using the mouse wheel
May 26	:	Change car display
May 27	:	Car rotates to next crossing
May 28-29	:	Move the map using the mouse button; Crossing name appears on hover
May 30	:	Select start and final points with the mouse; Fixed some bugs
May 31	:	Fixed input bugs; Code simplification and documentation
June 1	:	Final report

3 User Input (Path Selection)

As introduced previously, the program starts by asking user to choose between keyboard and mouse as point selection. In this section, I will explain about the workflow, how it works, and some minor improvements.

3.1 Using keyboard

If the user chooses this input, the program will not directly open the GLFW window. Instead, it will ask the user some questions in the terminal. With this input, it is supposed that the user already knows the name of the crossing as the departure and destination.

The first step is to ask the user to search for the name of the crossing for the starting point. The user is expected to input any text that should be the name of one crossing. The search algorithm in this process uses the search partial algorithm, so if the user input "aoba", then the search result will be everything that includes the text "aoba" in the crossing name. If it can find any result, it will display it in the terminal, and the user needs to choose ONLY among those search results. This concludes the starting point process, while for the destination point, the process is the same.

Once the user finishes all the question, the Dijkstra algorithm takes the process and then display the path result in the terminal. Finally, the GLFW window is opened, and it will display the path result.

3.2 Using mouse

Different from above, when the user chooses this input, the program will directly open the GLFW window and start to draw the plain map to select the points of interest. The user is expected to click on one of the crossings: the first click is for starting point, the second click is for the destination point. While choosing the points, the user can freely move the map, such as zoom in/out using the mouse wheel or move the map using the arrow key or drag it using the mouse right button.

Back to the selection process, all these processes are indicated using the selection_state variable, which means what "scene" it is (0 for keyboard input/finish, 1 for start selection, and 2 for destination selection). Once the user finishes the input, the Dijkstra algorithm takes the process and then display the path in both terminal and the display.

4 Algorithm

4.1 Partial search algorithm

To confirm whether a string is included in another string (a part of it), we can use the function available in **string.h**, which is strstr(string1, string2). It is a function that returns a location (pointer) discovered in string1 in case this string includes string2. It will return NULL in case it does not find any location. Thus, the condition strstr(string1, string2)!=NULL means "in case s1 includes s2".

4.2 Dijkstra path algorithm

Dijkstra algorithm finds the shortest path tree from a single source point (starting point) by building a set of nodes with a minimum distance from the source. This is done by initializing three values:

- an unrealistically large distance value between each points,
- set all crossing to be not visited,
- set distance of starting node as 0.

How the algorithm works:

1. The algorithm starts at the starting node and analyzes the graph to find the shortest path between that node and all the other nodes in the graph.
2. The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path.
3. Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path.
4. To select the next node (the one with the smallest distance and is not visited), it is not necessary to be the neighbour of the current node. The process continues until the destination node is visited.

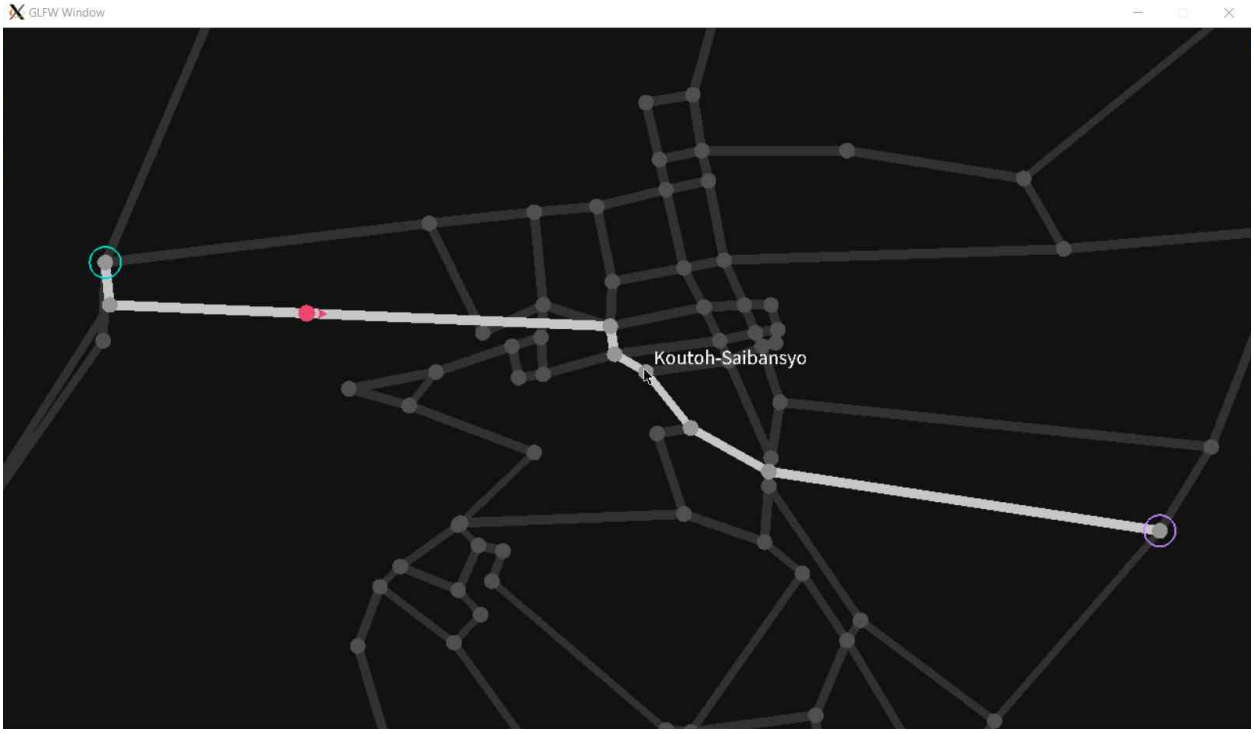
5 Path Display

In this program, the GLFW display to set in a resolution of unresizable 1280x720. The map display can be zoom in/out using the mouse scroll wheel, while the radius of the circle represents crossing and car, also the font size depends on the this zoom value. The map display can also be "translated" or move using the arrow key or drag the mouse right button.

The object shape and colour are listed here:

- Green hollow circle : Starting point
- Purple hollow circle : Destination point
- White road : The shortest path between two points above
- Red "pointer" car : The moving object (car)

Figure 1: The resulting path display with mouse hover in one of the crossing.



When the path is displayed, the car will be at the starting point. It can only be moved if the user presses the spacebar, which then triggers the "pointer" to appear that represents where the next crossing is. The car will move following the white road and once it reaches the destination point, the pointer will disappear and the program finished. In addition to the display, the name of the crossing will appear if the user hovers the mouse on the circle of the crossing.