Orientation to the Starter Code, Data Files and Front-end

This guide will give you an overview of how to use the front-end application, both to visualize the routes you plan in weeks 2 and 3 and also to collect custom street data for all weeks in this course.

You will not submit anything for this assignment, but there is a short quiz that follows this assignment that will ask you about the structure of the code and the data files. The last part of this reading is optional, but encouraged.

The structure of the starter code

Most of the starter code is dedicated to implementing the front-end, and you will not have to use it at all. The following are the packages that you will work with in this course:

- **basicgraph**: The starter files for week 2. You will modify these files.
- **geography**: Contains two classes, but you will only work with the class GeographicPoint. You should not modify this class, but you will need to use it in your assignments in weeks 3 and 4 and 6.
- **roadgraph**: This is the package you will be working with in weeks 3, 4 and 6. You will modify the provided code as well as add classes to this package.
- **util**: Contains a utility class, GraphLoader, for loading data from files. You will need to use this class for testing, but you should not modify it.
- **week3example**: Contains the example code from week 3. Feel free to run this code as well as modify it.

You should not need to touch any of the other packages, but if you are curious, you are more than welcome to poke around.

Overview of the data files

In addition to the starter code, we also provide you with some data to begin working with, as well as a mechanism for collecting your own road data (explained in the last section of this reading).

All provided data can be found in the data directory (at the same level as the src directory). In this directory you will find several sub-folders. Here's an explanation of what is in each:

- **airports**: Data about airports and routes served by United Airlines. These files use the extension .dat. You can load this data file into an object of type basicgraph.Graph using the GraphLoader.loadRoutes method.
- **graders**: Copies of the graph files used for grading each week. You can look at these files, but do not change these files or you will not be able to run our graders on your code to see what might have gone wrong.
- **intersections**: Data suitable for upload into the map visualization tool in week 3. These files use the extension .intersections. You can generate files of this type from raw map files (see below) using the method GraphLoader.createIntersectionsFile. Be sure to save your newly created files into this same intersections folder and give them the .intersections extension so you keep track of their format. More on the process of generating these files can be found in the next section.
- **maps**: Raw map data files that are saved from the front-end data fetcher. These files use the extension .map. You can load these files into an empty roadgraph.MapGraph object or into an empty basicgraph.Graph object using the GraphLoader.loadRoadMap method (it's overloaded for both types). We provide several raw maps from various locations in the USA, but we encourage you to collect your own data using our application. More on this process can be found in the next section.

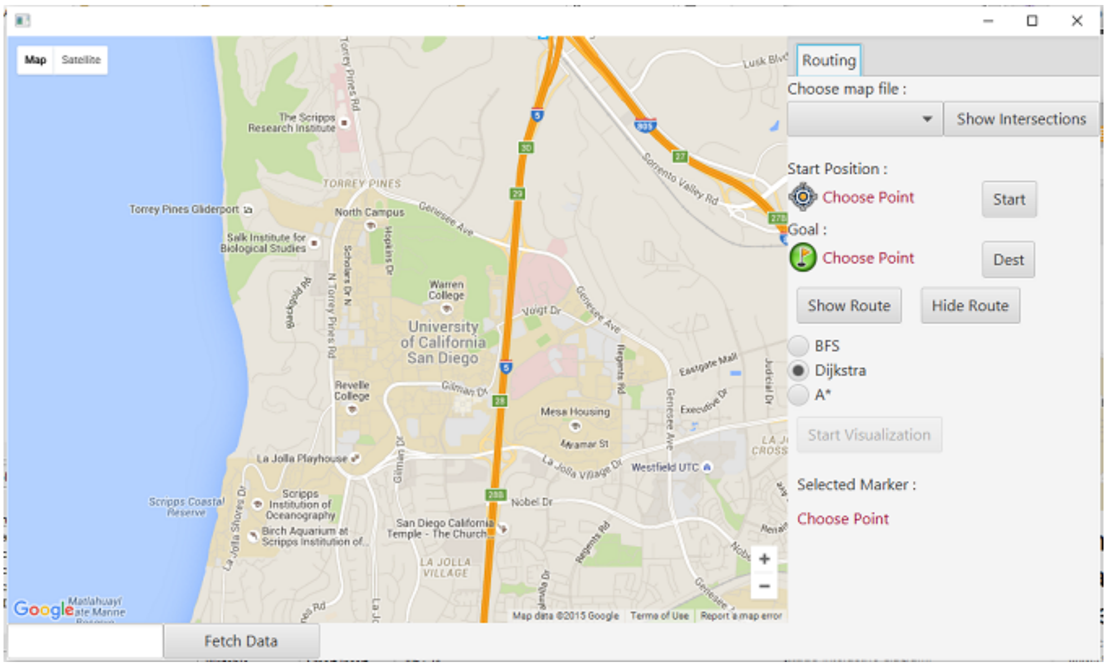
- **mazes**: Maze files used in the example in week 3. These files use the extension .maze. We encourage you to create your own. You can load these files using the MazeLoader utility in the week2example package.
- **testdata**: An artificial map for use in testing in weeks 2, 3, 4 and 6. It is in the same .map format as the files in the maps folder. You can load this file into an empty roadgraph.MapGraph object or into an empty basicgraph.Graph object using the GraphLoader.loadRoadMap method (it's overloaded for both types).

Working with the front-end to create your own data files (optional)

Before you can run the front-end, you need to follow the steps to obtain and set up your API Developer Key in Step 2 in our setup guide. If you have not done this, please do so now.

The front-end application allows you to display the routes that you create in weeks 3 and 4, but it also allows you to generate road data files for custom regions of the world. The display route functionality won't become useful until next week, so we will discuss that part next week, but for now you might want to use the application to create your own custom data.

Run the front-end application by running the class MapApp. (In Eclipse you should be able to open the file MapApp.java and run it as a Java Application). Note: **Do not use the front-end application if your are working offline**. When you run the application, you will see the following interface:



You can use the front end to generate raw map data files. Then you can use the GraphLoader utility class to convert these files into an intersections file, which is suitable for use in the graph visualization tool we introduce in the assignment in week 1.

1. Navigate the map to the section you would like to collect data for. The application will fetch all of the road data in the visible part of the map. Note: **Make sure this region is not too big or the file will be gigantic and it will take forever to download.** The data for a single small to medium city is about as large as we recommend.

2. Enter a name for your map data file in the text box in the bottom left corner of the window. This name must end with the extension .map and it will be automatically saved into the data/maps folder.

3. Click the "Fetch data" button. You will see a dialog box informing you that the fetching is occurring in the background. The "Fetch data" button is disabled as long as the data is still being fetched. Note: this process can take several minutes.

4. When the fetch completes, another dialog box will appear. Your data file is now in the data/maps folder. You probably need to right-click on it and select "Refresh" in Eclipse to see it.

4a. If you want your new map file to appear in the list of files available in the app when you restart it, you need to add it to the file mapfiles.list. You can find this file in the data/maps folder. Just open that file and type the name of the map file you just created then save that file. Your new map will appear in the list when you open the Map app in the future.

Congratulations! You now have a custom map data file that you can use in all of the assignments. However, if you want to use it in the graph visualization tool next week, keep reading.

5. In the main method of the util.GraphLoader class (in the file util/GraphLoader.java just above the class header for the RoadLineInfo class, which is declared in the same file), you will see the following code:

```
public static void main(String[] args) {
    //...
    // To use this method to convert your custom map files to custom
      intersections files    // just change YOURFILE in the Strings below to be
      the name of the file you saved.
    GraphLoader.createIntesectionsFile("data/maps/YOURFILE.map",
                                       "data/intersections/YOURFILE
                                        .intersections");
}
```

Change YOURFILE to be the name of the file you just saved from the front end and then run this class. You will see your .intersections file appear in the data/intersections directory. Again, from Eclipse you will need to right-click on the data directory and select Refresh.

You're

Mark as completed