Kyle Zalewski

CS202

Programming Project 1

The main coding challenge in this assignment was figuring out how to best access data between the various classes when needed. There were several times where a particular function seemed to be needed in two different contexts, but inheritance was not an option. Namely, the node class and associated functions gave me quite the challenge at several times during the coding process. I personally found that had I allocated more time for this project, I may have been able to come up with a more elegant way to pass the appropriate data around to various classes and create a better, more efficient hierarchy. As the due date approached, I realized that my best option was to make my existing structure as efficient and object-oriented as possible.

In terms of the data structures used, for the purpose of this assignment, the array of linear linked lists worked fine, and recursive implementation of traversal functions kept those sections of code concise. In a real-world application for an automatically-piloted car, I think that a 2-dimensional array may have some benefits. For example, if all the cars on the road were able to send information to each other, I can see a potential need for rapid addressing of various parts of the road well away from the user car, which may be slower if linked lists need to be traversed several times.

The main hierarchy for my program had, as planned, a base class of location. This allowed me to derive several classes for different types of vehicles (or open spaces) to occupy a given “slot” in traffic. Classes for car and “space” were derived from location, and represented the very start of what’s necessary in the car asking “can I be there or not?” From space, I derived a class for bike lane and a class for bus lane, which each have their own requirements. Some bike lanes are on the edge of a street, some are between lanes, and some share space with cars. These are distinct from the main characteristic of a bus lane, which really only needs an identifier to say “can a car be here if it’s open or not?”

I decided to get rid of the followingDistance class from the initial design, to keep the code simpler and easier to work with. The functions that followingDistance would have been responsible for are easily integrated into other parts of the program.

If I had more time on this project I would certainly look to implement a graphical interface for the user. I would allow for the sometimes erratic movement of other cars on the road. My implementation of the location class would have more provisions for communications to child classes. I spent a lot of time working on getting the data structure to behave the way I wanted it to and less time than I would have liked on features of the program. I will adjust my timeline for the next project now that I have a better understanding of the scope of this project, which I’m assuming resembles the rest of the projects in the course.