Antonio Scalfaro

CMSC 335

Project 3 – Traffic Light GUI

**Lessons Learned:**

This is the Traffic Light GUI project that contains five classes: Main extends Application, Car extends Rectangle, Lane extends Pane implements Runnable, TrafficLight, and the enum class TrafficLightState. The program can be initiated by compiling and running from the Main class. This program produces a GUI that has a timer at the top and buttons (start, stop, pause/continue, and add lane) at the bottom. The middle portion of the GUI will display the lanes (up to 5) as they are added with the add lane button. Cars can be added to each lane (up to 4) by left clicking on the desired lane. When the start button is pressed, each lane (on their own threads) will be started. When started, each lane moves the cars the appropriate distance based on the time for the car to move and the speed the car is going. Cars will not move during the duration of the red light, cars are moved following the change of light for how the car would have moved during the previous light sequence. If a car moves past the intersection line, the car will be removed from the lane. Car position and speed will be labeled at the top of each lane. As cars are removed from the lane, so will the label be updated to reflect the remaining cars in the lane.

This program was a difficult program to create, I have not made many programs that require concurrent programming to this level, and I found managing the threads to be difficult and my program is far from perfect. In fact, this is the first project I have finished that does not entirely satisfy me with its behavior. Though the program mostly acts as it should, there are quite a few unpredictable behaviors that are problematic. First, some of the run throughs produce one lane that ends up stalling GUI updates. The lane itself will continue to act as normal as indicated by the car label at the top (i.e the car still updates position and ultimately is removed when it reaches the end but the gui remains unchanged in both light color and car position). This issue may lie in the moveCars() method in the Lane class. I believe it should be refactored to handle GUI changes better. Another unsatisfactory behavior is if a lane is added, the start button is pressed, and then a car is added, the car will not be moved for the duration of the green light. This behavior does not happen if the lane is started after the addition of the car and the behavior only affects the first green light. This is not as bad as the previous issue but still does not perform as smoothly as I would like. Lastly, I did not fit this program to mimic a true lane of traffic in the sense that cars will not move during a red light even if they have not reached the intersection yet. I would have liked to do this so that the cars moved until they reached the intersection and stopped only if the light was red and they had not reached the intersection.

Overall, I am happy with the work and effort I put forth. The good outweighs the bad for me in this project and I learned a lot about managing threads and updating a GUI in a concurrent program. I was particularly happy with the overall user experience adding lanes and cars, pausing and resuming lanes, and stopping and clearing all lanes. I was happy with the behavior the cars display in the lane, if a car is about to “crash” into the car in ahead of it, it will adjust its speed to match the car ahead and position itself half a car length behind the car ahead. The testing is all manual testing of the GUI and it performs moderately well, with the aforementioned issues already noted.

**Figure 1:**

**A screenshot of a computer

Description automatically generated**

**Table 1:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Description** | **Screenshot** | **Pass/Fail** |
| **1** | Start GUI, add 3 lanes, add 1 car to each lane, do not start, click stop button to clear all lanes, exit. | Figures 2.1, 2.2 | pass |
| **2** | Start GUI, add 3 lanes add 1 car each, start. Add 1 car per lane until 3 cars. Exit after all cars cleared. | Figures 3.1, 3.2, 3.3 | pass |
| **3** | Start GUI, add 1 lane and 1 car, start, add 4 lanes and 1 car per new lane, pause, continue. Exit after all cars cleared. | Figures 4.1, 4.2, 4.3 | pass |
| **4** | Start GUI, add 5 lanes and 1 car per lane, start. Add new car every lane after car previous has moved once until 4 cars per lane. Exit after cars cleared. | Figures 5.1, 5.2, 5.3 | fail (lane 2 does not produce the GUI of all 4 cars and fails to clear car image) |

**Figure 2.1:**

A screenshot of a computer

Description automatically generated

**Figure 2.2:**

A screenshot of a computer

Description automatically generated

**Figure 3.1:**

A screenshot of a computer

Description automatically generated

**Figure 3.2:**

A screenshot of a computer

Description automatically generated

**Figure 3.3:**

A screenshot of a computer

Description automatically generated

**Figure 4.1:**

A screenshot of a computer

Description automatically generated

**Figure 4.2:**

A screenshot of a computer

Description automatically generated

**Figure 4.3:**

A screenshot of a computer

Description automatically generated

**Figure 5.1:**

A screenshot of a computer

Description automatically generated

**Figure 5.2: Cars match speed of cars in front to avoid crash (lanes 3, 4, 5)**

A screenshot of a computer

Description automatically generated

**Figure 5.3: (Lane 2 is the malfunctioning lane)**

A screenshot of a computer

Description automatically generated