underthesea.png

Fine a Car!

November 3, 2019

Contents

1 Introduction

Submit a **zipped folder** that is **only** containing your Java source files (*.java) in course's Black Board.

Please use the following naming convention for the submitted folders:

YourPSLetter_CourseCode_Surname_Name_HWNumber_Semester
Example folder names:

- **PSA_COMP130_Surname_Name_HW3_F19**

- **PSB_COMP131_Surname_Name_HW3_F19**

Additional notes:

- Using the naming convention properly is important, **failing** to do so will be **penalized**.

- **Do not** use Turkish characters when naming files or folders.

- Submissions with unidentifiable names will be **disregarded** completely. (ex. "homework1", "project" etc.)

- Please **write your name** into the Java source file where it is asked for. **Failing** to do so will be **penalized**.

- If you are resubmitting to update your solution, simply append **v#** where # denotes the resubmission version. (i.e. **v2**)

1.1 Academic Honesty

Koç University's *Statement on Academic Honesty* holds for all the homework given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult with your instructor.

1.2 Aim of The Homework

The aim of this project is to allow you to practice coding, testing and debugging a graphics program.

In this project, you are asked to simulate the traffic at a crossroads at a basic level using two cars on two perpendicular lanes with two traffic lights. The background containing the roads, intersection and traffic lights are given to you. You are expected to create and move the cars, change the traffic lights and fine any car that violates the laws.

1.3 Given Code

You are provided a code which has the necessary setup. It contains helper methods; variables and constants for you to start with.

Do not change anything in the code if it is indicated to you with a comment. The code given to you has something called **JavaDoc** comments above all the

methods. These comments allow you to view various information about the method when you mouse over the name of the method. Below are the methods given to you in the code with their explanation.

1.3.1 Given Methods

There are methods provided for each part.

- **void init()**

  This method is implemented in the Graphics Program itself and is guaranteed to be called before the void run(). This is the reason that you are seeing objects drawn on the screen even before starting the project. This method simply calls other methods to construct the starting state of the simulation's background.

- **void placeLights ()**

  This method places the light objects GOval hLight and GOval vLight to the correct positions on the screen. These objects are given to you as well and you are expected to use these for the lights on the screen.

- **void constructRoads ()**

  This method constructs the roads and the crossing and adds them to the screen. See below for how the roads are created individually.

- **GRect createRoad(int center , int width, String dir)**

  This method is used by void constructRoads() to generate the GRect objects that corresponds to the roads. int center is the coordinates of the center of the road to be placed. (note that String dir affects how this variable is used) int width is the width of the road to be created. String dir is used to switch the direction of the road, creating either a horizontal or a vertical road. (note that this is not implemented using a switch)

1.3.2 Given Variables and Constants

There are some variables and constants provided to you. Feel free to use the variables declared in the given code, meaning you can either use them or create new ones if you need to. HOWEVER, you need to **use the constant variables** provided to you. You can create additional constants if you need to.

All constants provide **JavaDoc** comments above them. Please read these to understand what constant is used for what. **Do not** use another variable or a static value for something if there is a constant variable defined for that purpose.

1.4 Further Questions

For further questions **about the project** you may send an email to **course SLs** at **(comp198-fall19-sls-group@ku.edu.tr]** and **Ayca Tuzmen** at **[atuzmen@ku.edu.tr]**. Note that it may take up to 24 hours before you receive a response so please ask your questions **before** it is too late. No questions will be answered when there is **less than two days** left for the submission.

2   Task

This project aims to simulate the traffic at a crossroads at a basic level using two cars on two perpendicular lanes with two traffic lights. The background containing the roads, intersection and traffic lights are given to you. You are expected to create and move the cars, change the traffic lights and fine any car that violates the laws.

Before starting any of the tasks, run the project to verify that it is running without errors. Make sure that your setup produces the exact same result as the **Figure 1**.
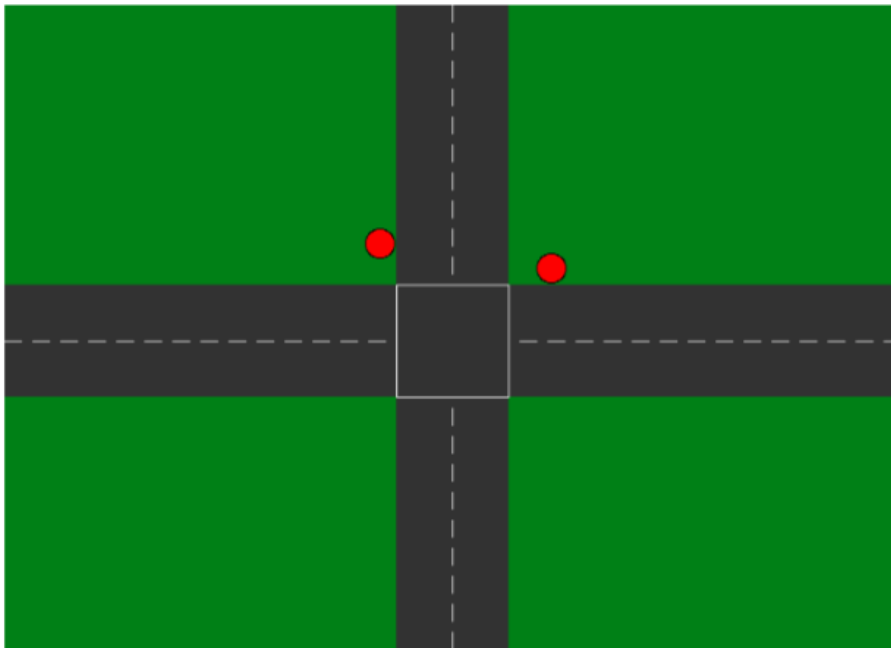

Figure 1: Initial state of the world. Note that this is achieved by the given code and must not require any tinkering with the code to work.

2.1   Creation, Animation and Collision of Cars

The project features two cars of equal sizes with different colors. You can find the required constant variables for the creation, animation and collision of the cars among all the constant variables. All subtasks of this task are dependent to the previous subtasks apart from 2.1.4 which does not require 2.1.3 to be completed.

2.1.1   Creating the Cars

Create two cars that will be traveling **perpendicular** to each other. The car traveling on the horizontal road is expected to be colored and the car traveling on
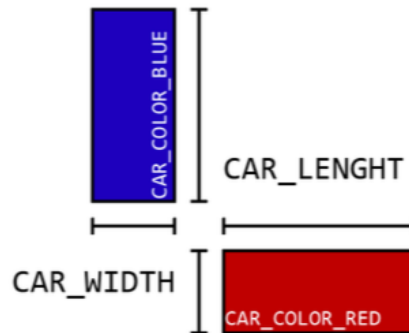
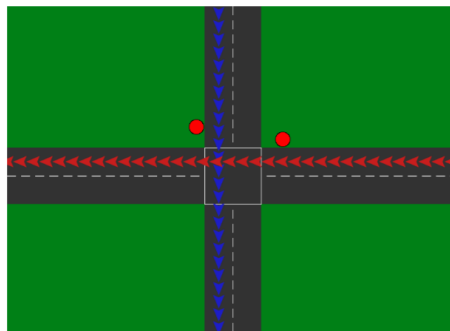Figure 2: Specifications of the cars related to given constant variables.

Figure 3: Movement directions of the cars. Route of each car is denoted with it's corresponding color.

the vertical road is expected to be colored blue. Refer to Figure 2 for detailed specifications on how to create the cars. The world has right-hand traffic so please use the right-hand lanes for the cars. The red car is expected to enter the world from the **East** and the blue car is expected to enter the world from the **North**. Use the constants given in the code, in creating and placing the car.

2.1.2   Moving the Cars

The cars are expected to move with **random** speeds that are between the given limits which can be found at the constant variables section. Once you generate random speeds for both cars you will need an animation loop to move the cars. The pause time is defined in the constant variables section as well. Note that each car is required to move with the same random speed until it leaves the screen. Make sure that cars are not changing their speed **while they are visible**. Refer to **Figure 3** for a visual representation of the route of the cars.

2.1.3   Collision of the Cars

The animation must stop when two cars collide with each other. This state will be used later in the project in another task. Note that for this part you **do not**
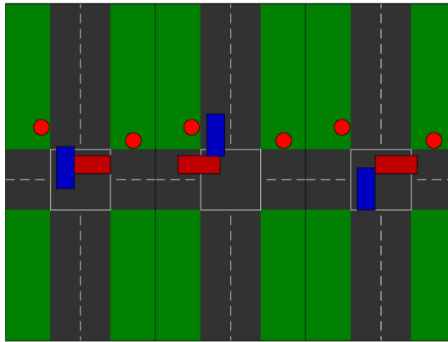
Figure 4: Example accident scenarios showing how collisions may happen.

need to make sure that cars are not inside each other, just stopping the cars when this happens is sufficient. See **Figure 4** for examples.

    Hint: You **will not need** to animate anything else after this point.

2.1.4   Continuity of the Traveling Cars

Once a car leaves the screen, it is expected to return from the same entry position with a **different random speed**. The red car leaves the screen from the **West** and enters again from the **East**, while the blue car leaves the screen from the **South** and enters again from the **North**. Note that you **must not wait** for both cars to exit the screen before a car re-enters the screen.

2.1.5   Traffic Lights

The project features two traffic lights that are placed to the sides of the lanes that you are expected to use. The lights are given to you as GOval objects. GOval hLight is the traffic light on the horizontal road where as GOval vLight is the traffic light on the vertical road.

2.1.6   Changing Lights Randomly

The traffic lights are expected to change randomly, however a light must stay at one state for a **random amount of time** that is between the minimum and maximum allowed by the constant variables. Note that two lights are **independent** of each other, in other words both lights may be at the same or opposing states at a given time. Do not alternate the states of the lights forcing one to be red while the other is green.

2.2   Fining the Cars

As implied by the name of the project, the project focuses on fining the cars as they violate the laws. You are expected to track the total amount of fines received by each car. For the purpose of this project, the world presented to you has the following laws:

- **Speed Limit Violation:**

Cars are fined a **dynamic** amount of money each time they violate the predefined speed limit. The fining happens as they enter the intersection, assume that a traffic camera is watching the intersection. The amount is calculated by fining each unit of speed the speeding car is over the speed limit by the predefined fine amount for the fine for speeding. (Simply put the fine is multiplied with the difference between the speed of the car and the speed limit.)

- **Red Light Violation:**

  Cars are fined a **static** amount of money each time they enter the intersection while the traffic light for the lane is red. The amount is defined by a constant variable.

Hint: Cars are always fined as **they enter the intersection**. It may be beneficial to detect when this happens.

2.2.1   Fining for Speed Limit Violation

You are expected to fine cars as they **enter the intersection**. Despite that at this point it looks irrelevant at what point you fine a speed limit violation as speed never changes during one pass of a car, it will become important in the following tasks. Make sure you use the **given constant variable** for fining the cars.

2.2.2   Fining for Red Light Violations

Again, you are expected to fine the cars as they **enter the intersection**. As the fine amount is **static** for this violation, no calculations are necessary for the fining, you can directly fine the cars. Make sure you only fine cars that pass during a red light and use the **given constant variable** for fining.

2.2.3   Fining for Multiple Violations

Cars may violate both laws at the same time. You are expected to fine a car that violates both laws at once for both the violations.

     Hint: Both violations occur as the car enters the intersection.

2.3   Displaying Fines

So far the screen displays no information about fining which is not acceptable as you implemented a fining system already. You are expected to display an indication on the screen each time a car is fined.

2.3.1   Creating Labels for Display

You are expected to have two different labels to display the fine received by each car. The label for the red car must be on the **top right corner** of the screen and **aligned to the right** whereas the label for the blue car must be on the **top left corner** of the screen and **aligned to the left**. Both labels are expected to have the given font, be properly margined and colored with the same color as the corresponding car.
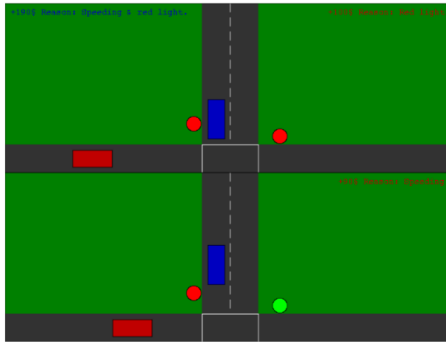
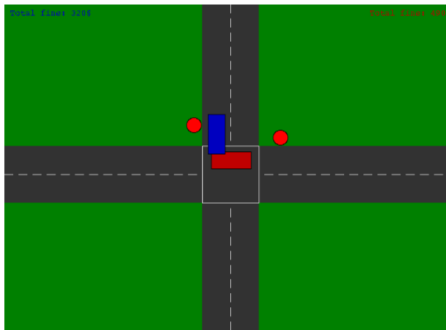Figure 5: Example displaying all the possible fining situations for the cars.

Figure 6: Displaying total fines after a collision occurs between two cars.

2.3.2 Displaying Fines

Every time a car gets fined, you are expected to indicate the fine amount and the reason the car is fined for. **Figure 5** shows the texts for the labels in all possible scenarios. If a car is not fined when it enters the intersection, the label for that car must be **empty**, **do not** leave it unchanged.

2.3.3 Displaying Total Fines

When the cars collide, the animation ends, however the total fines of both cars are still unused. You are expected to display the total fines for each car when a collision occurs. See the **Figure 6** for the display format of the total fines.

2.4 Bonus Part

This part is optional and if you implement this section, you will get a bonus point. The amount of the bonus points will be between 5-15 determined depending on your design. In this section, you can be creative in designing your own car, using sound effects. Feel free to use images, graphics objects to represent your car. You can also add any special effects to simulate collision, fining, etc. However keep in mind that the bonus part should not change the major requirements of the project. It shouldn't change the rules of the fining, the fine display and the overall layout of the street.

2.5   End of Project

Your project ends here. You may continue to tinker with the code to implement any desired features and discuss them with your section leader. Below in the **Section 3** are two further tasks for you to implement if you are willing to continue practicing the topics. **Do not** include any additional features that you implement after this point in to your submission.

    Final Warning: Do not include anything beyond this point to your submission. Points may be deducted from your grade as Section 3 alters the normal behavior of the simulation.

3   Further Tasks

Tasks described in this section are not supposed to be included to your project, but are provided for studying the topics further. **Do not** submit your project with any of these tasks completed. You will only be graded for the tasks in **Section 2**. Also note that tasks below are meant to be implemented on their own but may function together as well.

3.1   Normalizing Traffic Light

A normal traffic light behaves differently than what is implemented in this project. The light switches to yellow before switching to red to indicate the drivers that they should stop.

3.2   Yellow Lights

Implement the yellow light state to the traffic lights. The light must only change to yellow before switching to red. There is no green light. The yellow state has constant time and lasts for **1000 ms**.

3.3   Sensible Drivers

A sensible driver would try to stop at a red light as not doing so would violate the laws and may result in an accident. Define an maximum acceleration value that you will use in your project. Adjust the value so that it is possible for a driver driving with the speed limit to stop in **1000 ms**. Drivers will begin slowing down when the light turns to red. Remember that the maximum acceleration (in this case deceleration) is defined and a driver may not exceed this value when decelerating. Note that not all drivers will successfully slow down at the light to a stop and you need to accelerate when you pass the light. If a car successfully stops at a light it will accelerate back to its speed with the acceleration value **when the light switches** to green.

3.4   Alternating Lights

Normal traffic lights would not allow intersecting lanes to pass a crossroad at the same time. The project asked you to not implement this feature specifically before. Now implement it as it is required for this task.

3.5   Realistic Drivers

Not all drivers will attempt to stop when they see the light switch to yellow. More aggressive drivers will attempt to speed up to pass at the yellow light. Implement it so that any driver that will not be able to stop at the light will speed up to pass in the yellow. Note that maximum acceleration is still a constraint.