**CP2406\_Programming 3\_Assignment 1**

**By: Zar Ni Tun**

**Java Based Traffic Simulator**

**Program Working Document**

**Specification**

The client needs a way to be able to simulate traffic in city areas; needing to simulate different vehicle types behaviour with traffic lights, roads and intersections. The program will be console based for this iteration and will be used in the console. Taking inputs from the user and displaying the status of objects within the simulation to the user. This version of the program will be able to simulate a single car moving along a single lane road, interact with a traffic light, move to a second road connected to the first, representing a very basic intersection, and move to the end of that road.

**Decomposition**

The problem can be broken into operate objects that interact with each other to achieve the desired behaviour. These objects include:

**Model.Vehicle (abstract)**

The vehicle class will be abstract, having the Model.Car Model.Bus and Model.Motorbike as it subclasses. Having the following attributes;

id – a unique identifier that will differentiate each vehicle.

Length – the physical space the vehicle occupies longways.

Breadth – the physical space the vehicle occupies widthways, half the vehicles length.

Speed – how far the vehicle moves for each simulation turn.

Position – where the vehicle is located on a road.

Current Model.Road – the road the vehicle is currently traveling on.

Colour - a colour assigned to each vehicle for when it is represented graphically.

A vehicle will be able to move, using the move() method, within the simulation along a road depending on its speed, the speed will be defined by the speed limit of the road the car is traveling on. When the car is in the same position as a traffic light it will check its state before moving, if the light is red it will stop, if the light is green it will move to the next road. When the vehicles position is equal to the end of a road and there is no connected road it will stop and be removed ending the simulation. The vehicle will also be in control of how it is represented graphically with its draw() method.

**Model.Car**

The car class will be a subclass of vehicle, an object that describes an average size road vehicle.

**Model.Bus**

The bus class will be a subclass of vehicle, describing a large road vehicle. It will inherit its attributes and behaviour from Model.Car except its length will be defined as being three times that of the car’s length.

**Model.Motorbike**

The motorbike class will be a subclass of vehicle, describing a small road vehicle. It will inherit its attributes from and behaviour Model.Car except its length will be defined as being half that of the car’s length.

**Model.Road**

The road class will be an object that describes a single lane road. Holding the following attributes;

id - a unique identifier that will differentiate each road

Speed limit – the maximum speed that vehicles on that road may travel at.

Length – the number of segments the road is comprised of and the physical space it occupies.

Start location – the (x, y) coordinate that represents where the road begins.

End location – the (x, y) coordinate that represents where the road ends.

Connected roads – all of the roads that this road is physically connected to.

Lights on the road – all the traffic lights that are on the ends this road.

Cars on the road – all of the vehicles that are currently traveling on this road.

Orientation - The direction the road is facing graphically.

The for the first version of the program the speed will be constant and set to 1. Meaning the car will only be able to move a single position each turn making it easier deal with traffic lights and the ends of roads. The length of the road will be variable depending of user input. Roads will interact with other roads by being connected to them, have the ability for multiple connections, creating a very basic intersection with multiple options for the car to take. Cars will move along the road from the starting position (1) to the end position, that depends on the road’s length. Traffic lights can be placed at only the end position of the road. The road will also be in control of how it is draw graphically with the draw() method depending on it orientation.

**Traffic Light**

The traffic light class will represent a simple red or green traffic light. Holding the following attributes;

id - a unique identifier that will differentiate each traffic light.

State - the colour the light is displaying.

Position - where the traffic light is located on the road.

Model.Road attached to - the road that the light is attached to.

The traffic light will operate, using the operator() method. Randomly changing from green to red. This operation will be based on pseudo-random numbers generated by the program. The light will be placed on a road only at its final position, the end of the road, and will interact with vehicles that are also at that position. If the light is red the vehicles will stop and not move to the next road. If the light is green the vehicles will continue past and move to the next road. it will also be able to control how it is drawn graphically with the draw () method.

**View.SimulationPanel**

vehiclesToSpawn - total numbers of vehicles to be spawned in the sim,

vehiclesSpawned - how many vehicles have been spawned so far.

vehiclesRemoved - vehicles that have reached the end of the map and been removed.

numberOfCycles - the number of simulation cycles between spawns.

scale - the scale of objects drawn graphically.

This class will generate the graphical display of the simulation within the main frame. Loading the location of road and traffic light objects from the editor. Vehicles will be created for the simulation, types of vehicle created being random, here and the update rate is also set here depending on user input form dialog boxes in main. The simulation will be run within a timer.

**View.EditorPanel**

scale - the scale of objects drawn graphically. This class will generate the graphical display for the city editor within the main frame. Allowing users to click to place roads, letting them define orientation and connection with dialog boxes within the New menu option.

**Model.SaveFile**

This class will handle the saving of simulation files.

**Main**

This class will have the main() method that will Call the simulation and editor panel. It will also Have the menus used for user navigation. User input for simulation parameters will also be handled here; number of vehicles spawned

**User Stories**

**User 1**

**Name:** Noelle

**Age:** 18

**Purpose:** The user wanted to save the progress of the simulated map and reuse the map later.

**Priority:** High

**Test:** This is done by enabling the users to pick and make their map and roads using the editor panel and when the drawing of the simulated map is done the information is then saved so that it could be used at a later time

**User 2**

**Name:**  Jason

**Age:** 20

**Purpose:** He wanted to make sure the simulation would allow complex roads as the road area he is situated in has multiple roads with several twists and turns

**Priority:** Medium

**Test:** Roads will interact with other roads by being connected and the user are able to place the road on the map by clicking on it thus enabling the road to overlap and have multiple connections.

**User 3**

**Name:**  Michael

**Age:** 30

**Purpose:** The users want to make sure that he can input cars other than himself so that simulation is more like a traffic and is more realistic allowing him to a more probable simulation.

**Priority:** High

**Test:** To this I have added features such as vehiclesToSpawn, vehiclesSpawned and vehiclesRemoved so that the user can spawn the number of cars he desires and is able to limit or remove the number of cars that are in the simulation or that are have reached the destination.

**User 4**

**Name:**  Jonathan

**Age:** 24

**Purpose:** The person wants to be able to start and stop the simulation when he wants to when he wants to record the information or etc.

**Priority:** High

**Test:**  To this we have added the stop feature than enables the user to stop the simulation whenever he wants to record the values or to see in which road or position his car is situated In the simulation and to start whenever he wants to.

**Git Hub repository**

A screenshot of a computer

Description automatically generated<https://github.com/BlazeZNT/CP2406AssignmentTask2Zarni.git>

**UML Diagrams**

Diagram

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated