­

Traffic Simulation

Software Design Document

Jack Lee

2 November 2022

CS 225, Fall 2022

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

**Introduction:**

This project is a traffic simulation involving different cars, drivers, and weather conditions along a straight variable-lane road with a variable speed limit. The goal of the project is to allow the user to observe how cars react to changes in driving conditions, such as a changing speed limit, weather, or a stopped vehicle in one lane. For example, the simulation could be run as a three-lane highway with a slowdown in one lane. The user could observe how the simulated cars react to the slowdown and determine the optimal course of action if the situation happens in real life. This document provides a description of the traffic simulation project, including detailed descriptions of the goals the project hopes to achieve and the ways those goals are to be accomplished.

**Problem Description:**

Cars are an extremely common transportation method for much of the world. This fact can be described with the Private Car Index, or the number of personal cars per 1,000 people, as illustrated by V. A. Profillidis, G. N. Botzoris in *Modeling of Transport Demand: Analyzing, Calculating, and Forecasting Transport Demand*:

*Chart

Description automatically generated*

This project aims to simulate traffic, defined as “the movement (as of vehicles or pedestrians) through an area or along a route” by Merriam-Webster. In this case, the simulated traffic is cars along a straight road. Each car is driven by a driver, and the driver-car pairs are referred to as “cars” in this document unless one specific component is being discussed. It is assumed that every car has infinite gas and will not break down, and that the roadway is perfectly maintained. These simplifications exist to allow the simulation to be completed within the time allotted. The “road” is a linear plane the cars move along. The simulation is designed to be used and interpreted by a human operator, referred to as the “user.”

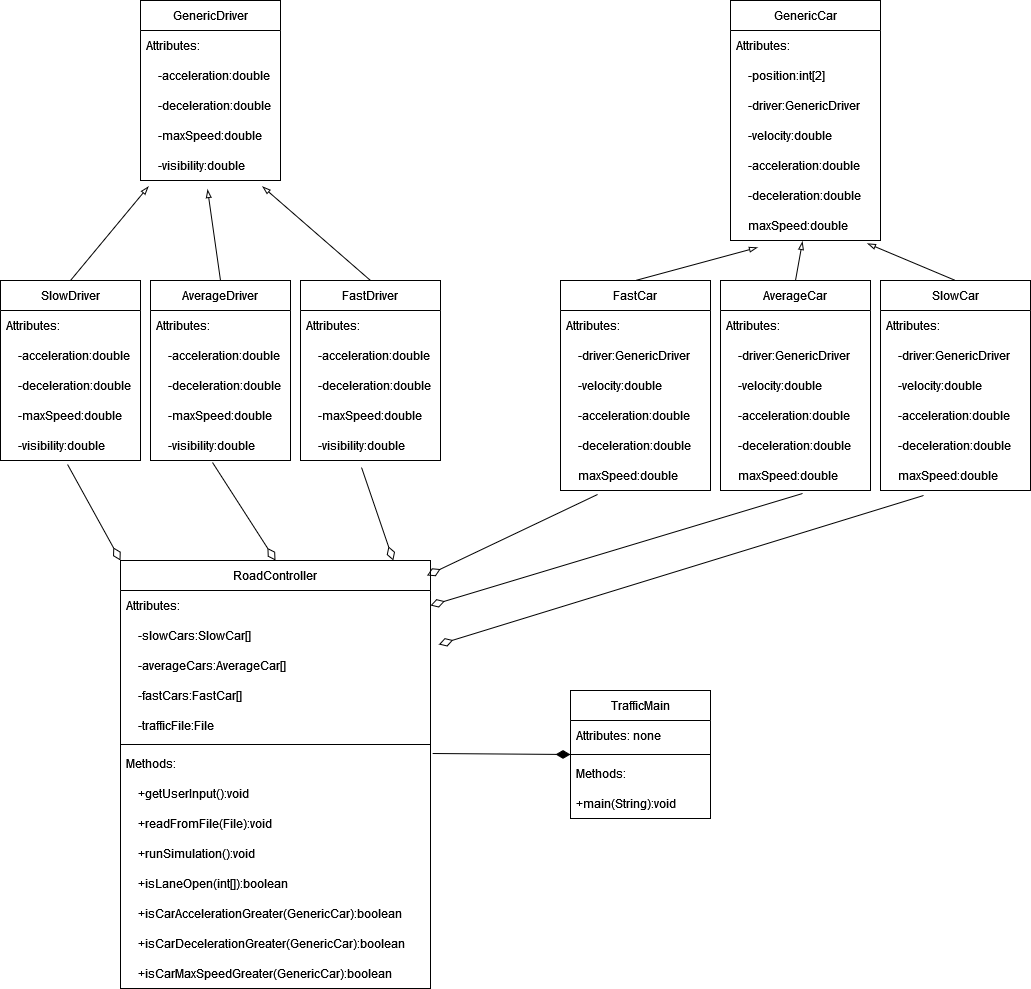
Each car has a position, velocity, acceleration, deceleration, and maximum speed. The position is measured in miles from the start, which has a position of 0. The maximum position is 100 miles and upon reaching this position, a car’s position resets to 0 and keeps increasing. The velocity is a measure of how many miles a car can travel in an hour (mph) at the current instant in time. Acceleration and deceleration are used to determine how fast a car’s velocity can increase and decrease respectively and are measured in miles per hour squared.

Along with cars, the simulation includes drivers with different acceleration and deceleration (reaction speeds), visibilities, and maximum speeds. The driver’s speed properties take precedence over the car’s speed properties as long as the car’s properties exceed those of the driver. A driver’s visibility is measured in yards and is affected by the weather. For example, rain and nighttime decrease visibility while daytime and clear skies increase it. Visibility is the maximum distance another car can be from the driver for the driver to detect it and respond to actions the car ahead makes(braking, accelerating).

The simulation reports the number of cars on the road, the average speed, and the traffic flow rate. The average speed is calculated by dividing the sum of each car’s current speed by the total number of cars on the road, and is measured in mph. The flow rate is the product of the traffic density and the average velocity, measured in cars per hour (University of Idaho, 2003). The traffic density is the number of cars within a one-mile reference point, measured in cars per mile (University of Idaho, 2003). The traffic flow rate depends on the density reference point; this point can be changed by the user if it is within the bounds of the road (0-100).

The user can pause and the simulation at any time and choose to set the speed limit, weather, or lane number. These changes affect the entire road all at once when the user un-pauses the simulation. The user can also select lanes of cars and cause the leading car to slow to a stop as quickly as the driver’s deceleration speed allows. Cars can only drive forwards in this simulation, so the lowest possible velocity is 0. The following cars will respond to the slowed car by slowing down or switching lanes if there is no other car beside them. The simulated cars should react as they would in reality, not accounting for the simplifications necessary for this project. All of the cars on the road and the road conditions are specified from files that users can create.

**Problem Solution:** This section provides a description of the software design, according to the following Unified Modeling Language (UML) diagram.



Details of each class and their functions are provided below:

TrafficMain: This class contains the Main method and is the class the simulation is run from. It has no attributes and creates one GameController object to run methods from it. The readFromFile() method is run, followed by the runSimulation method. The GameController may be turned into an attribute in a future alteration.

RoadController: This class contains methods and attributes to create various Cars, position them along the road, and allow them to interact with each other. The attributes are arrays for each Car type and a File used for input. Below are descriptions of the methods:

getUserInput- prompts the user for a file name and initializes trafficFile to the file name. It then calls readFromFile on the file.

readFromFile- reads a File to create Cars for the array attributes. Traffic Simulation files are written in CSV (Comma Separated Values). For example, each line of the file would represent a different Car on the road with its attributes.

Exampe: FastCar,distance,lane,FastDriver

runSimulation- runs the simulation with the Car attributes by incrementing the position by the velocity. Once a Car’s position is greater than 100, it resets to 0 and continues to increment. The velocity increments by the acceleration, which is determined using the lesser acceleration value between the Car and the Driver. The lesser acceleration is determined by calling isCarAccelerationGreater. Drivers and Cars each have a maxSpeed attribute, and the lesser of these (determined by isCarMaxSpeedGreater) will be used as the threshold beyond which the velocity cannot increase. If there is another Car with slower velocity and greater position than the current car within the current Driver’s visibility range, and if isLaneOpen is true, the current Car will change lanes. If isLaneOpen is false, then the Car will slow down by decrementing the velocity by the lesser deceleration value between the Car and Driver, determined with isCarDecelerationGreater. All of this happens for each car until the simulation is paused.

isLaneOpen- Given an integer array containing the lane changing car’s position and lane number respectively, checks if there are any cars in the same position in the desired lane. If there is a car, false is returned and if there are no cars then true is returned.

isCarAccelerationGreater, isCarDecelerationGreater, isCarMaxSpeedGreater- determines if the Car’s attribute is greater than the Driver’s attribute. If so, returns true, if else, returns false.

GenericDriver: This class is the blueprint for Driver subclasses. It contains all the attributes needed for a Driver to be created. The subclasses differ in the following ways:

SlowDriver- Acceleration and deceleration are random double values between 1 and 3. Visibility is 10, stored as a double. MaxSpeed is a random double value between 50 and 70.

AverageDriver- Acceleration and deceleration are random double values between 4 and 5. Visibility is 7, stored as a double. MaxSpeed is a random double value between 60 and 80.

FastDriver- Acceleration and deceleration are random double values between 5 and 8. Visibility is 5, stored as a double. MaxSpeed is a random double value between 70 and 110

GenericCar: This class acts as the blueprint for all Car subclasses. It contains the attributes necessary for Cars to be created and utilized by the GameController. Each car object has a Driver attribute. The subclasses are defined below:

SlowCar- Acceleration and deceleration are random double values between 2 and 4. MaxSpeed is a random numerical value between 60 and 70. MaxSpeed is an integer, but it is stored as a double to assist with calculations.

AverageCar- Acceleration and deceleration are random double values between 3 and 5. MaxSpeed is a random numerical value between 75 and 90. MaxSpeed is an integer, but it is stored as a double to assist with calculations.

FastCar- Acceleration and deceleration are random double values between 6 and 8. MaxSpeed is a random numerical value between 90 and 110. MaxSpeed is an integer, but it is stored as a double to assist with calculations.

**References:**

Merriam-Webster. (n.d.). Traffic. In Merriam-Webster.com dictionary. Retrieved October 16, 2022, from https://www.merriam-webster.com/dictionary/traffic

Profillidis, V. A., Botzoris, G. N., Profillidis, V. A., & Botzoris, G. N. (2018). *Modeling of transport demand : Analyzing, calculating, and forecasting transport demand*. Elsevier.

University of Idaho. (2003). *Speed-Flow-Density relationship*. Transportation Engineering Online Lab Manual; University of Idaho. <https://www.webpages.uidaho.edu/niatt_labmanual/Chapters/trafficflowtheory/theoryandconcepts/SpeedFlowDensityRelationship.htm>

**Appendices:**

No appendices are needed for this document.