­

Traffic Simulation

Software Design Document

Jack Lee

12 December 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 will be equal to the distance from the leading car to the trailing car.

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.

UML Diagram

Description automatically generated**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 the getUserInput method.

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 core methods:

getUserInput- asks the user to select between inputting a file or running test cases. If the former is chosen, the user is prompted for a file name and calls readFromFile, then calls runSimulation. If the second option is chosen, runTestCases is called.

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. The first line of the file is designated as the “Header” and contains information about the weather and speed limit. The number of lanes is discerned by the method countLanes.

Example header: Sunny, 75

Example body: FastCar,30(distance) ,0(lane),FastDriver

runSimulation- runs the simulation while not paused by first calling deployPauseListener. When the simulation is running, enforceWeatherConditions is called, followed by assignMinimumValuesToCars. Then, Cars are ordered to speed up to the maxSpeed or speedLimit (whichever is lower), and to checkMirros. Then, the statistics for Total Cars, Average Speed, and Flow Rate are displayed. If the simulation is paused, then the runSimulation method calls initPauseMenu.

deployPauseListener- Because Windows Commnd Prompt buffers its input, a GUI was needed to utilize a KeyListener. The GUI window appears only when deployPauseListener is called, and disappears when the “P” key is pressed and the simulation is subsequently paused.

checkMirrors- Akin to checking mirrors in real life, this method determines whether or not a Car can change lanes or if it should simply slow down for traffic ahead. If there is no Car in the same position in an adjacent lane and there are Cars ahead, the Car will change lanes to one of the open adjacent lanes. If there are cars ahead, but no adjacent lane is open, then the car will slow down. Lastly, the car will do nothing if there is no traffic ahead.

initPauseMenu -The pause menu appears when the simulation has been paused and has options to change the speed limit, change the weather, add or remove lanes, slow down a lane, resume the simulation, and exit the simulation. The options involving changing variables such as the speed limit or weather have their own methods.

enforceWeatherConditions- Assigns a modifier to the different weather types and applies it to all Drivers’ visibility. Sunny weather has no modifier, Rainy weather has a -4 modifier, Overcase has a -2 modifier, and Storming has a -6 modifier.

assignMinimumValuesToCars- for every Car-Driver pairing, assigns the minimum acceleration, deceleration, and maximum speed variables to the Car. This allows the Driver type to influence the car’s behavior.

speedUp/slowDown/slamBrakes- Modify the car’s speed by either its acceleration, its deceleration, or -10 in the case of slamBrakes. Then, the speed is added to the position.

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.