

KING'S COLLEGE LONDON

TRAFFIC SIMULATOR

GROUP PROJECT

Team Diversity

Members:

Balázs Kiss

Eddy Mukasa

Yukolthep Visessmit

Pongsakorn Riyamongkol

Snorri Hannesson



March 24, 2015

Contents

1	Introduction	3
1.1	Background	3
1.2	Descriptions	3
1.3	Objectives	3
1.4	Scope	4
1.5	Methodology	4
1.6	Schedule	4
1.7	Obstacle	5
2	Related Work	6
3	Requirement and Design	7
3.1	Requirement	7
3.2	Design	7
4	Implementation	8
4.1	Maps	8
4.2	Drivers & Vehicles	8
4.3	Traffic Policies	8
4.4	Running the Simulation	8
4.5	Graphical User Interface	8
4.5.1	Main Stage	9
4.5.2	Canvas	9
4.5.3	Settings panel	9
4.5.4	Result stage	10
5	Team Work	11
5.1	Methodology	11
5.1.1	Roles	11
5.2	Physical meetings	11
5.3	GitHub	11
5.4	Facebook	12
5.5	Trello	12
6	Evaluation	13
6.1	Result	13
6.1.1	Shortest time	13
6.1.2	Longest time	13
6.1.3	Average time	14

7	Peer Assessment	15
7.1	How do we evaluate our member?	15
7.2	What is the criteria that we have used?	15
7.3	What is there result and summary of peer assessment?	17
A	Gitlog	18
B	Source Code	24

1 Introduction

1.1 Background

Over the past few decades, the world's population has been continuously increasing which is becoming issues in some countries. This is resulted in overwhelming traffic in the most major cities around the world such as London, Bangkok, and New York. Those cities are looking for the way to solve traffic congestion problem. There are many theories and methods to handle this issues. For example, in Bangkok, it is used priority lane in the peak time hour to relive traffic congestion problem. Above all, any theories and methods which is applied for solving this problem may need to use together with good traffic management policies. In this paper, it will suggest to use traffic simulator program to test the traffic management policies before these policies can be implemented in the real world. This traffic simulator are supposed to be abstract models of the real world, if a traffic management policy works well on the simulator. This probably may work in the real world. Therefore, this is the reason why the traffic simulator program is generated.

1.2 Descriptions

The traffic simulator is an abstract model of actual real world traffic. The simulator will have both cars and buses. There are three type of drivers' behaviour— cautious, reckless or normal. In this simulator, there are two different traffic management policies witch are supposed to relief traffic congestion issues. Those two policies are fixed time policy and congestion control policy. This policies can be compared by average time each vehicle is in the system. Moreover, an emergency strategy will be implemented to see how the system reacts when an ambulance is injected to the system.

The simulator will be programmed in Java programming language. The simulator will have a graphical user interface (GUI). The GUI is created with the help of JavaFX software platform. The rational for using a GUI: 1. Better visualisation and understanding of code during development, i.e. actually seeing what is happening when programming collision detection. 2. When the final product is ready users can see the road system and the cars and therefore get a better understanding of how the road system is and how the policies work. Opposed to just get the result log and results of which policy is superior and have no visual understanding of what happened.

1.3 Objectives

- To develop a traffic simulator program which has the following structure: two types of vehicles, three types of drivers, functional road system with many roads and lanes, junctions, intersections and traffic lights.
- To compare two different traffic management policies: Fixed Time Policy and Congestion Control Policy.
- To examine how the system reacts in an emergency period by injecting an ambulance to the simulator.

1.4 Scope

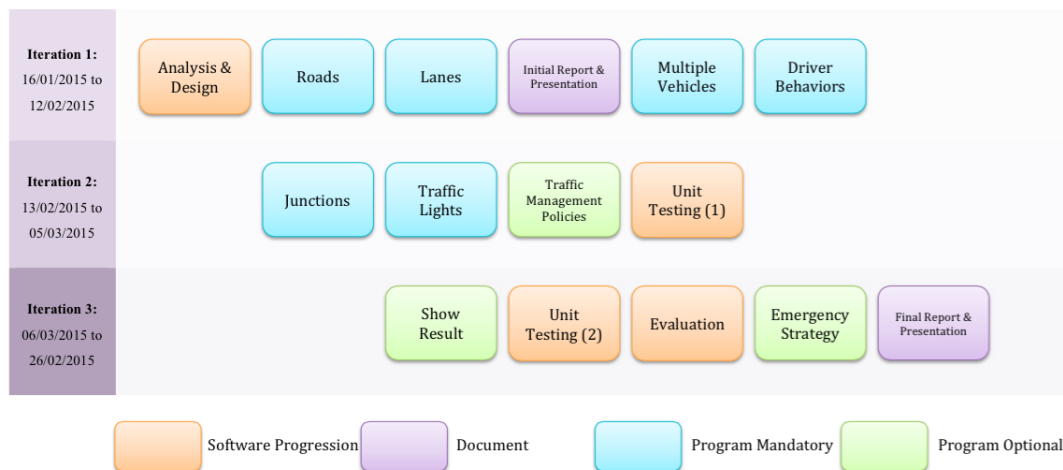
- Each road can have multiple lanes, which can be in the same or opposite direction.
- British's traffic is left-lane oriented.
- The system has only two types of vehicle (cars and buses)
- There are three types of driver behaviour which is cautious, reckless, and normal.

1.5 Methodology

- I. Analysing: requirement
- II. Planing and Organising: schedule and assigned task
- III. Developing:
 - Software: used JAVA
 - Source code: stored at GitHub
- IV. Evaluating:
 - Program
 - Peer Assessment
- V. Reporting:
 - Document: written by LaTeX
 - Presentation

1.6 Schedule

The development of Traffic Simulator program is divided into three iteration in 10 weeks. In the first iteration (week 1 to week 4), our team has focused on the requirement and how to design. After our plan is committed, we have started to develop traffic simulator programme. Then, we has continued to complete the mandatory function in the second iteration (week 5 to week 7). Moreover, in this period, we have finished traffic management policy function, as a program optional and done the unit testing also. In the last iteration (week 8 to week 10). this traffic simulator program has been launched after finished second unit testing and evaluation. The detail of traffic simulator progress is illustrate in the figure, as followed.



1.7 Obstacle

- Language:

Our member come from different country especially in Thailand, Iceland, Hungary and..... Those country has their own language which caused to the communication issue. Communication is the most important aspect for team work to understand any function of tasks and behaviour also.

However, in this case, we try to communicate more by using another channel. For example, we use picture helping our member to understand the same concept and point that we need to achieve. Therefore, we are able to get rid of language issues between each other and we do more communication.

- Skill and Background:

Our member has different skills and background in programming. There are some members who have experience about software development due to their own job. In addition, this project is not required only programming skill but other related skill also (communicating and reporting etc.) Nevertheless, each member is not able to be perfect skill, as the project aims.

At this point, we know that our member has an unique skill and background. We need to help each other to complete the project task. We will train our member to improve other skill which this person does not have or less. Thus, we can overcome this problem and use this problem to gain more benefit.

- Time:

We have got only 10 weeks for developing traffic simulator which is a large number of requirements.

On the other hands, we have set task schedule for each person in deep detail. This help us to follow up our task and complete the project on time.

2 Related Work

In today's world a lot of effort is made to make transportation as good as possible. For most countries and cities the road systems is a vital part of it's transportation system. With the ever growing population and increasing purchasing power of the public, good traffic control has never been as urgent. Changes to road systems are hard to make and drivers wouldn't be pleased if many experiments were made on live traffic. That's why traffic simulators play a big role in increasing the quality of the road system. Any change can be simulated and the result of the change can be analysed. There are many challenges that traffic simulation creators face. Trying to predict the behaviour of drivers and the synergistic effects of different factors can have on a driver behaviour is perhaps the most challenging, the goal is to make it as realistic as possible.

Many traffic simulators exist as well as many papers and books on that subject. In this module we were given two papers on traffic simulation for inspiration for this project and an insight into this field. Sewall, Wilkie, Merrell and Lin [?] presented a novel method for the synthesis and animation of realistic traffic flows on large-scale road networks. Their technique is based on a continuum model of traffic flow they extended to correctly handle lane changes and merges, as well as traffic behaviours due to changes in speed limit. They demonstrated how their method can be applied to the animation of many vehicles in a large-scale traffic network at interactive rates and showed that their method can simulate believable traffic flows on publicly available, real-world road data. They furthermore demonstrated the scalability of this technique on many-core systems.

Namekawa, F. Ueda, Hioki, Y. Ueda and Satoh [?] spent several years developing a general purpose road-traffic simulation system to analyse road traffic jams. The concept of their system was using the running line model as opposed to fixed road-network information database, which is not effective in their opinion. Their simulator uses the a cell automaton model.

3 Requirement and Design

3.1 Requirement

3.2 Design

4 Implementation

4.1 Maps

Flexible model
Map, Roads, Lanes, Junctions
Class Diagram
Usage examples

4.2 Drivers & Vehicles

Separation of driver and vehicle, driver makes decisions
Different driver classes
Different vehicle classes

4.3 Traffic Policies

Traffic Lights
Fixed time & congestion policies

4.4 Running the Simulation

Simulation class and subclasses
ISteppable interface
timing
Generating cars
Results

4.5 Graphical User Interface

In our application, JavaFX is selected as a graphic platform to be used. This platform allows developers to create an application on various platform such as desktop, mobile, or web. Furthermore, this platform introduce the use of cascading style sheets (CSS) to decorate the interface. This way, coders won't have to worry about decorating the interface themselves in their codes and let CSS designer to handle this instead. Although in our application the use of this benefit haven't been implemented along with the java code but we think that this is a very helpful technique worth mentioned from JavaFX. Our application consists of 2 stages, which are both a top-level container of JavaFX, one is the main stage that will show a simulation and another one is the one used to show a simulation's result. The first stage is separated into 2 parts by using Border pane. This pane works like BorderLayout in Javaswing but in JavaFX this layout is implemented into a pane rather than being one of the layout to be set into an empty pane. The followings are put into left and center region of this border pane respectively:

4.5.1 Main Stage

[need fix on sectioning]The first stage is separated into 2 parts by using Border pane. This pane works like BorderLayout in Javaswing but in JavaFX this layout is implemented into a pane rather than being one of the layout to be set into an empty pane. The followings are put into left and center region of this border pane respectively.

4.5.2 Canvas

Canvas is a node that is a blank image. This node can be painted on using GraphicsContext class object. Our canvas has a width of 800 pixels and height of 600 pixels so we don't have to worry about how our application will run in different screen resolution since this dimension is probably comply with all present screen resolutions. To draw a simulation, we can call a GraphicsContext object from canvas itself using `getGraphicsContext2D()` and by doing so, we can manipulate the canvas with various methods from GraphicsContext like `fillPolygon(double[] X, double[] Y)` which will draw a polygon with specified fill colour from a set of coordinate X and Y. To draw roads, we can call an above method and get a set of parameters from a simulation class which contains a list of roads. Each road can provide its own coordinate at each corner so we use this as elements to be added into a set of X and Y coordinate. Because this is a "fill" type method which will also paint an enclosure of a polygon we can provide only 3 points to the method which these are `leftStartPoint`, `leftEndPoint`, and `rightEndPoint`. The method automatically close the gap between a start point and last point with a straight line and fill a polygon afterwards. For vehicles, we have already provided a picture for each type of vehicle in a JPEG format and draw these pictures based on what type of vehicle we are dealing with from a list of vehicle provided from simulation class. There is a useful method in Vehicle class called `getDisplacementVector()`, using this method along with the one in Point class from utils package called `angleVectorDegree()` we obtain an angle in which a vehicle is running compare to a horizontal line measuring In counter-clockwise manner. After we obtained a value of angle, a method called `drawRotatedImage()` will do a job to draw a vehicle picture onto a canvas. This method, again, call another method `rotate()` which receive 4 parameters consist of GraphicsContext object, angle value, x-position, and y-position where these x and y form a coordinate in which will be an anchor point of the rotation. After that, we create a rotate class object by using above parameters. This rotate class object will perform a calculation in order to make a GraphicsContext rotate before we actually draw something using it. After we have rotated the GraphicsContext, we can normally draw a picture onto a canvas. Note that all of these happened after the start button is pressed, which the button will send a signal to start a simulation.

4.5.3 Settings panel

In this panel, we take an advantage from JavaFX features called HBox and VBox, which will place objects horizontally and vertically next to previous one by specified insets.

The idea is to put many Hboxes into one VBox but since radio buttons still require us to put them in vertically, so we have put one VBox into Hbox to be able to put radio buttons in their correct position. Figure x given below will give an overall image of how we put each component together within this pane (HBox indicated by red border, VBox indicated by black border and object are indicated by blue green border).

4.5.4 Result stage

This stage will only appear after the simulation has ended and user press the result button that locates aside of the start button. A simulation will pass various information onto a new stage which will behave as child of the main stage. This means that some behaviour like minimizing and closing will be done to this stage too, if it's happened with its parent. This stage use GridPane which allow us to create a table-like contents to show to users. A constructor required enough data from a simulation to create a report to user of a current simulation that just has ended.

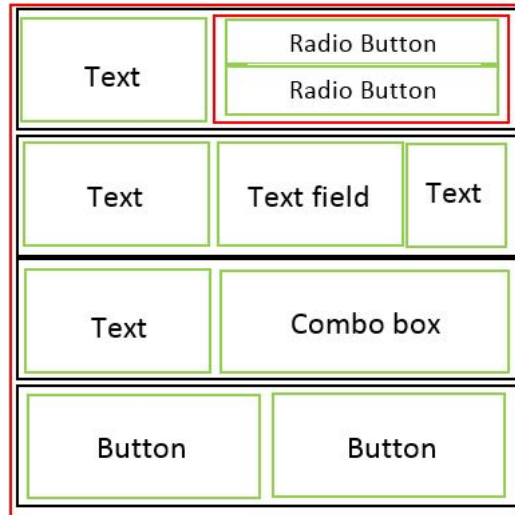


Figure x

5 Team Work

In this chapter we'll describe how we worked together during this project, including the tools and processes we used to facilitate group work. We'll will not reflect on what went wrong and what worked well, that is the focus of chapter 6 - Evaluation.

5.1 Methodology

A slightly adjusted Agile methodology was used for this project. What is meant be that is we did not follow Agile to the bone but we took bits and pieces from Agile that we knew from previous studies and work experience. The bits and pieces we chose to use from Agile are things that we believed would work for us in this project, i.e. iterations, user stories, roles and we used test driven development to some extend.

5.1.1 Roles

Balázs Kiss: Lead programmer

Eddy Mukasa: Architect

Yukolthep Visessmit: Graphical designer

Pongsakorn N. Riyamongkol: Project Manager

Snorri Hannesson: Tester and Coordinator

Each member of the group had a responsibility to oversee one aspect of the project but was not expected to do all the work defined in his role. I.e. each member should/could do some programming but the lead programmer should oversee the code and make sure nothing is missing and everything is done properly. The same goes for the other roles.

5.2 Physical meetings

Every Thursday at 10 o'clock we had a physical meetings where all team members were expected to attend. We kept log of our meetings so those who were unable to attend the meeting could get up to date by reading the log and for those who did attend the meeting to refresh their memory of what was discussed in the meeting. In these meetings we discussed the progress we made from last meeting and the problems we were faced with. At the end of each meeting we allocated work to each member witch is supposed to be done during the week until next meeting. Occasionally extra meetings have been scheduled where a certain aspect of the project been focused on. Not all members are expected to attend these meetings but only the members who are focusing on this certain aspect of the project.

5.3 GitHub

Github was used for version control. Our branching strategy was that every time a member wanted to implement a new feature to the system or write something to the report, a new branch was created for that feature. When that branch was created it

would be up to date with the master. When work were done on that branch a pull request would be made to the master. The leader of that aspect of the project would then decide to merge or to create a new branch with suggestions of improvements or alterations. Those improvements could then be merged with the original branch which could then be merged with master.

5.4 Facebook

Facebook was used as a communication channel. The first thing this team did was to create a facebook group for the team. Most people these days use facebook everyday so this is very convenient communication channel. In this group we would have various discussions about the our progresses or problems. The facebook chat was also used for individual members to discuss matters that were not directly associated to the group as a whole but only the members in the chat.

5.5 Trello

Trello is a versatile tool which was used for project management. The rationale for using Trello is that it's very flexible so you can modify the interface to your will. In this project not all tasks were code related which is not a problem when using Trello.

6 Evaluation

6.1 Result

After a simulation has ended, we obtain a set of data regarding to that simulation in the result window. Some of these data, especially average time, can be used to analyze the effect of each traffic policy or to compare them between two policies. In traffic management study, we will use the following in order to analyze a simulation or even to the real world.

6.1.1 Shortest time

[may need to change to numerical list later]This data tells users how many smallest seconds for a vehicle to leave the system. A really small value of this type of result does not tell much about the effect of each policy as there might be some vehicles that enter a system with good timing so that they arrive at the junction when a traffic signal for that road turns green. The more that this value getting close to the average time the more it tells us that a policy is working well. The reason behind this logical conclusion is that most of the vehicles that went into the system left the system nearly as fast as the fastest vehicle so when we calculate for the average, the value will be as near as the shortest time if most of the input is very close to the shortest time. The table below shows an example of this situation.

Vehicle#	Time(seconds)
1	18
2	17
3	23
4	21
5	30
6	19
7	17
8	20
9	22
10	18

Table x

We can see that the average time is $\frac{\sum Time}{10} = 20.5$ seconds, which is only 3 seconds more than the shortest time.

6.1.2 Longest time

In contrast to the shortest time, this data tells users the opposite which is the longest time a vehicle take to leave the system. A really large value in this case can be divided into 2 different situation. First is that, it warns users that a selected traffic management

policy may has something wrong since there is at least a single vehicle that has to stay in the system for a long time, compared to the average or not. Second, on the other hand, might be happened just because there is a vehicle that appear to stuck twice at the junction where others have already left.

6.1.3 Average time

This is one of the most important factor to be used in analyzing how effective each policy can be. It gives users a broad view of the system as a whole, not specific to a single vehicle. One of the advantage we can take from obtaining this valuable data is that we can make a better decision when there are more than one system to be managed (which always be). Assume the situation that there are three systems in our interest and either two of them can lead to the last one, we can compare their average time simulated with various policies and choose the best result. This result can help drivers to make a decision in which system should they take to reach their destination system as fast as possible.

7 Peer Assessment

Robert T. [?] states that “The term peer assessment refers to the process of having the learners critically reflect upon, and perhaps suggest grades for, the learning of their peers.” In the other words, peer assessment is a process which student are able to assess their friends based on the criteria. This causes student to provide some feedbacks and evaluate their friends, which may help learning together (University of Reading, 2015). Therefore, TeamDiversity is going to use the peer assessment method for grading our member. This will be going to focus on the methodology which is used for assessment following by the criteria. It will be then shown the result and summary of each member in TeamDiversity.

7.1 How do we evaluate our member?

- I. Distribute the assessment form and assessment criteria to our member.
- II. In each member, he/she must score himself/herself as well as other group member. For example, if our team has 4 people, it will grade 1 for yourself and 3 for our friends.
- III. When you have completed a score (your friends and yourself), you need to mark total add up all scores and calculate the average score for yourself.
- IV. We use only the average score to evaluate our friends and present in this report.

7.2 What is the criteria that we have used?

University of Sydney (2015) has published the assessment criteria and form on the website. TeamDiversity has adapted both documents in the appropriate way for supporting our task. This criteria is going to evaluate ten aspects of member behaviour. In each aspect, we have scored in the range from 0—10. Thus, the total marks of each member will vary from 0—100 inclusive. There will be then illustrated the detail in each aspect of peer assessment criteria, as followed

A. Quantity of Work:

- 0 - not taking part in it, having no prospect of progress/value
- 1—2 - doing a particular, not too much but enough
- 3—4 - sometimes above standard, generally needs improvement
- 5—6 - satisfactory, doing more than requirement
- 7—8 - always working hard and consistent
- 9—10 - outstanding, always over productivity standards

B. Quality of Work:

- 0 - not giving sufficient attention, making frequent mistakes
- 1—2 - giving attention, making some mistakes
- 3—4 - doing well, basically correct
- 5—6 - satisfactory, accurate in some aspect
- 7—8 - almost accurate in all involving fields

9—10 - outstanding, perfect work

C. Communication Skills:

- 0 - having bad manners, not showing respect for other people, not listen
- 1—2 - friendly and easy to talk to once know by others
- 3—4 - warm and friendly, sociable
- 5—6 - showing good manners, kindly, listens and understands
- 7—8 - courteous and respectful, best wish
- 9—10 - Inspiring to others, excellent at listening and understanding

D. Initiative:

- 0 - acts without plan/purpose
- 1—2 - need encouragement to do task
- 3—4 - putting in minimal effort to complete task
- 5—6 - desire to achieve task/goal
- 7—8 - strongly desire to achieve task/goal
- 9—10 - beyond duty, high motivation

E. Efficiency:

- 0 - always delayed
- 1—2 - occasionally finished on time
- 3—4 - usually finished on time, having minor errors
- 5—6 - always finished on time
- 7—8 - absolutely completed, consistent in troubleshooting and solving major problems
- 9—10 - invariably completed ahead of schedule, showing creativity, making major contributions

F. Personal Relations:

- 0 - very disruptive influence
- 1—2 - some friction
- 3—4 - no problem, commonly
- 5—6 - satisfactory, tuneful, harmonious
- 7—8 - positive factor
- 9—10 - respect by others

G. Group Meeting Attendance:

- 0 - never attended to meeting, not interest
- 1—2 - sometime attended
- 3—4 - usually attended, hard to get touch with
- 5—6 - attend, normally late
- 7—8 - count on to attend
- 9—10 - never ever missed a meeting, on time

H. Attitude and Enthusiasm:

- 0 - low disposition, having no prospect of value, unconcerned
- 1—2 - feeling/showing few excitement, blasé
- 3—4 - half hearted
- 5—6 - positive outward behaviour/bearing
- 7—8 - positive attitude and spirited
- 9—10 - excitement and eager, inspiring to others, positive thinking and influence

I. Effort:

- 0 - expects others to carry the load
- 1—2 - leave some effort
- 3—4 - displays enough endeavour
- 5—6 - firm and stable contributions
- 7—8 - energetic
- 9—10 - self starter, normally beyond duty

J. Dependability:

- 0 - unreliable
- 1—2 - unsteady, but slightly dependability
- 3—4 - inconsistent, occasionally be
- 5—6 - suitable, need some improvement
- 7—8 - very trustworthy, responsibility
- 9—10 - always responsible, steady influence

7.3 What is there result and summary of peer assessment?

A Gitlog

Author	Date	Message
Balázs Kiss	2015-01-25	Initial architecture
Balázs Kiss	2015-01-26	Ignore build directory
snh11	2015-01-26	Meeting log created, first three meetings documented
Snorrihann	2015-01-26	A template for the initial report
Snorrihann	2015-01-28	More work done on the intital report
Snorrihann	2015-01-29	Modification of roles
Snorrihann	2015-01-29	sdfs dfds ds
Snorrihann	2015-01-29	Change to conflicts
Snorri Hannesson	2015-01-29	Merge pull request #1 from teamDiversity/Snorri_Conflicts
Balázs Kiss	2015-02-01	Speed model for vehicles, basic collison avoidance
Balázs Kiss	2015-02-01	Merge pull request #2 from teamDiversity/develop_speed
Snorrihann	2015-02-03	meeting 29th of january
Snorri Hannesson	2015-02-03	Merge pull request #3 from teamDiversity/meeting_log
Snorrihann	2015-02-04	Implementation of different types of vehicles
Snorrihann	2015-02-04	Changes on the initial report made by Neab
Snorri Hannesson	2015-02-04	Merge pull request #4 from teamDiversity/initialReport_NeabChanges
Snorri Hannesson	2015-02-04	Update .gitignore
Snorri Hannesson	2015-02-04	Delete initialReport.txt
Snorri Hannesson	2015-02-04	Delete initialReport.wc
Balázs Kiss	2015-02-05	Corrected my name
Balázs Kiss	2015-02-05	Merge pull request #5 from teamDiversity/report_typo
Balázs Kiss	2015-02-05	Removed junk
Balázs Kiss	2015-02-05	Merge pull request #6 from teamDiversity/report_typo
Balázs Kiss	2015-02-05	Moved vehicle classes to separate package
Balázs Kiss	2015-02-05	Use protected topSpeed field for vehicles
Snorrihann	2015-02-05	Merge branch 'develop_carsAndBuses_improvements' of https://github.com/teamDiversity/trafficSim
Snorrihann	2015-02-05	Succestions of improvements implemented
yukolthep	2015-02-05	first gui implementation

Balázs Kiss	2015-02-05	Merge pull request #9 from teamDiversity/master
Snorrihann	2015-02-05	meeting 5th of feb
Snorri Hannesson	2015-02-05	Merge pull request #7 from teamDiversity/develop_carsAndBuses_improvements
Snorri Hannesson	2015-02-05	Merge pull request #10 from teamDiversity/develop_carsAndBuses
Snorrihann	2015-02-05	Merge branch 'master' of https://github.com/teamDiversity/trafficSim
Snorrihann	2015-02-05	acceleration re-changed to <code>maxAcceleration</code>
Snorri Hannesson	2015-02-05	Merge pull request #11 from teamDiversity/development_maxAcceleration
Snorrihann	2015-02-05	Meeting 5th feb logged
Snorri Hannesson	2015-02-05	Merge pull request #13 from teamDiversity/meeting_5thFeb
Snorrihann	2015-02-05	The mandatory/optional table and some proofreading
Balázs Kiss	2015-02-07	These files were not the latest
Balázs Kiss	2015-02-07	Merge branch 'master' into <code>gabb_branch_merge</code>
Balázs Kiss	2015-02-07	put back <code>maxAcceleration</code>
Balázs Kiss	2015-02-07	Merged vehicle subclasses
Balázs Kiss	2015-02-07	Changed project type to Java FX
Balázs Kiss	2015-02-07	Cleanup
Balázs Kiss	2015-02-07	Merge pull request #14 from teamDiversity/gabb_branch_merge
Balázs Kiss	2015-02-07	Simulation classes
Balázs Kiss	2015-02-07	Removed unnecessary parts from <code>GUISimulation class</code>
Balázs Kiss	2015-02-07	Merge pull request #15 from teamDiversity/simulation_classes
Balázs Kiss	2015-02-07	Small changes
Balázs Kiss	2015-02-07	Merge pull request #16 from teamDiversity/initial_report_balazs
Pongsakorn N. Riyamongkol	2015-02-08	1. This is the same as <code>InitialReport.tex</code> 2. If i edit, it is in this pool 3. I will put presentation slide too
Snorrihann	2015-02-09	Changes made on the meeting 9 feb
Snorri Hannesson	2015-02-09	Merge pull request #17 from teamDiversity/initialReport_changesFromMeeting9Feb
Snorrihann	2015-02-09	Meeting log 9th of feb
Snorrihann	2015-02-09	Changes made to fit requirements on the 'Nodes on the initial report'

Snorri Hannesson	2015-02-09	Merge pull request #18 from teamDiversity/meeting_9thFeb
Snorri Hannesson	2015-02-09	Merge pull request #19 from teamDiversity/initialReport_changesFromMeeting9Feb
eddy mukasa	2015-02-10	UML Use case and Class diagram
eddy mukasa	2015-02-10	Updates to UML Class multiplicities
eddy mukasa	2015-02-10	Merge pull request #20 from teamDiversity/UMLDesigns
eddy mukasa	2015-02-10	Updates to UML Class multiplicities
eddy mukasa	2015-02-10	merge conflict resolution
eddy mukasa	2015-02-10	new class diagram
yukolthep	2015-02-11	update on car's picture
Balázs Kiss	2015-02-11	Merge branch 'UMLDesigns'
Balázs Kiss	2015-02-11	Merge branch 'master' into develop_gui
Balázs Kiss	2015-02-11	Renderer classes, code cleanup
Balázs Kiss	2015-02-11	Fixed thread synch bug
Balázs Kiss	2015-02-11	Better map for testing
Snorri Hann	2015-02-12	First tests, just for fun
Snorri Hannesson	2015-02-12	Merge pull request #22 from teamDiversity/test_basicInitialTestSuite
yukolthep	2015-02-16	draw horizontal and vertical roads, still rotated road left
Snorri Hann	2015-02-18	More tests on vehicle and road system and a test suite class created
Snorri Hannesson	2015-02-18	Merge pull request #23 from teamDiversity/test_basicFunctions
Snorri Hann	2015-02-18	jUnit library added to project.properties
Snorri Hann	2015-02-18	jUnit library added to project.properties
Snorri Hann	2015-02-18	meeting log updated
Snorri Hannesson	2015-02-18	Merge pull request #25 from teamDiversity/meeting_11feb
Snorri Hann	2015-02-19	First creation of final report
Snorri Hannesson	2015-02-19	Merge pull request #26 from teamDiversity/finalReport_SnorriInitialWork
Snorri Hannesson	2015-02-19	Merge pull request #24 from teamDiversity/test_jUnitLibrary
Snorri Hann	2015-02-19	meeting 19feb
Snorri Hannesson	2015-02-19	Merge pull request #27 from teamDiversity/meeting_19feb

yukolthep	2015-02-19	try some manual drawing. still have problem with wrong drawing of rotated road. The angle calculated using the formula seems to be correct but after draw the rotated rectangular with specified angle, it is not correct as the road is drawn with wrong angle. Still can't figure out what is the cause.
eddymukasa	2015-02-19	Adding driver Classes to project
eddymukasa	2015-02-19	CautiousDriver fix
yukolthep	2015-02-20	- fix drawVehicle to correctly draw car using correct angle. - Simulation2 class is used for testing different rotated roads drawing
Snorrihann	2015-02-20	Roads and lanes now have four points as a paramiter: leftStart, rightStart, leftEnd, rightEnd. Roads are initialised by the leftStart and leftEnd.
Balázs Kiss	2015-02-21	Project properties
Balázs Kiss	2015-02-21	step counter
Balázs Kiss	2015-02-21	Removed vehicle position from constructor
Balázs Kiss	2015-02-21	inherited type method
Balázs Kiss	2015-02-21	Removed lane from constructor
Balázs Kiss	2015-02-21	Removed unneded methods
Balázs Kiss	2015-02-21	Vehicles added at entripoint
Balázs Kiss	2015-02-21	vehicles exit the system
Balázs Kiss	2015-02-21	Simualtion stops when all cars exited the system
Balázs Kiss	2015-02-21	Passing tests
Balázs Kiss	2015-02-21	Merge pull request #28 from teamDiversity/entry_and_exit_points
Balázs Kiss	2015-02-21	Merge development_widthOfLanes
Balázs Kiss	2015-02-21	Merge development_widthOfLanes
Balázs Kiss	2015-02-22	Merge branch 'development_widthOfLanes'
yukolthep	2015-02-23	- added new simple car and bus image. - draw vehicles based on their type. - fixed Normal and Cautious bus to extend from Bus class instead of Car class.
yukolthep	2015-02-23	- fixed using class instead of string to decide what type of vehicle it is
yukolthep	2015-02-23	- changed canvas size back to 800x600

yukolthep	2015-02-27	- clean up some unused test code - bug in drawing a car with wrong position possibly caused by the position of vehicle itself as the position reading from a command line is wrong. need further investigation.
Snorrihann	2015-02-28	roads and lanes tested
Snorrihann	2015-02-28	bugs fixed in Road and Lane classes
Snorrihann	2015-02-28	minor fix on Lane
eddy mukasa	2015-03-01	Implementing driver logic
Pongsakorn N. Riyamongkol	2015-03-01
Balázs Kiss	2015-03-02	Merge branch 'finalReport_IntNeab'
Balázs Kiss	2015-03-02	Merge branch 'gui_draw_vehicles_with_correct_size'
Balázs Kiss	2015-03-02	jfxrt in project description file
Balázs Kiss	2015-03-02	Merge branch 'test_RoadsLanes'
Balázs Kiss	2015-03-02	Fixed syntax errors
Snorrihann	2015-03-02	incorrect pos when road has slope
Snorrihann	2015-03-02	centerPoints for start and end
eddy mukasa	2015-03-04	driverLogic fix
Snorrihann	2015-03-04	time for vehicles in system printed in console
yukolthep	2015-03-04	- add start button - implement the program to automatically stop after closing the window
Pongsakorn N. Riyamongkol	2015-03-05	meeting 26th of feb
Pongsakorn N. Riyamongkol	2015-03-05	meeting 26th of feb logged
Snorri Hannesson	2015-03-05	Merge pull request #38 from teamDiversity/meeting_26thfeb
Pongsakorn N. Riyamongkol	2015-03-05	outline of final report
Snorri Hannesson	2015-03-05	Merge pull request #39 from teamDiversity/finalReport_NeabInitialWork
Snorrihann	2015-03-05	meeting 5th of march logged
Snorri Hannesson	2015-03-05	Merge pull request #40 from teamDiversity/meeting_5thmarch
Snorrihann	2015-03-05	cleanup
Snorri Hannesson	2015-03-05	Merge pull request #41 from teamDiversity/snorri_cleanup

Balázs Kiss	2015-03-06	Merge commit 'c12333d2b4a4e92263268e834ad0e56c0417e108'
Balázs Kiss	2015-03-06	Merge branch 'master' into driver_logic_merge
Balázs Kiss	2015-03-06	updated gitignore file
Balázs Kiss	2015-03-06	Fixing incompatibilities
Balázs Kiss	2015-03-06	Default constructor for vehicles
Balázs Kiss	2015-03-06	format code
Balázs Kiss	2015-03-06	Merge commit '8f5224a3bfea688218a80388091d5406e923f386'
Balázs Kiss	2015-03-06	formatting
Balázs Kiss	2015-03-06	Renamed package
Balázs Kiss	2015-03-06	Removed private netbeans files
Balázs Kiss	2015-03-06	Default speeds
Balázs Kiss	2015-03-06	Removed unused imports
Balázs Kiss	2015-03-06	Removed vehicle properties from driver classes
Balázs Kiss	2015-03-07	Moved decision methods to driver
Balázs Kiss	2015-03-07	driver dependant deceleration
Balázs Kiss	2015-03-07	dont round new positions
Balázs Kiss	2015-03-07	draw lanes
Balázs Kiss	2015-03-07	Added basic classes
Balázs Kiss	2015-03-07	steppable, traffic light states
Balázs Kiss	2015-03-07	add junctions to simulation
Balázs Kiss	2015-03-08	Merge branch 'master' into gui_create_start_button_merge
yukolthep	2015-03-08	- change draw road to fillPolygon() which does not need to calculate anything
yukolthep	2015-03-08	- edit car and bus images (rotate 90 deg) - made getDisplacementVector() public for using in SimulationRenderer class - implement new way to draw vehicles - add angleVectorDegree which returns in degree instead of radian
yukolthep	2015-03-08	- remove testing code
yukolthep	2015-03-08	Merge pull request #43 from teamDiversity/gui_rework_on_drawing
yukolthep	2015-03-08	- add a new class which will show a result of a simulation (left blank for now)
Snorri Hannesson	2015-03-08	Merge pull request #44 from teamDiver- sity/gui_add_more_user_interface

yukolthep	2015-03-08	- add radio buttons group to specify policy to be used - add textbox for user to enter duration of a simulation - add combobox for user to choose from pre-defined maps
yukolthep	2015-03-08	Merge pull request #45 from teamDiversity/gui_add_more_user_interface
yukolthep	2015-03-08	- add show result button
Snorri Hannesson	2015-03-08	Merge pull request #46 from teamDiversity/gui_add_result_window_button
Snorrihann	2015-03-08	results
Snorri Hannesson	2015-03-08	Merge pull request #47 from teamDiversity/development_results
Snorrihann	2015-03-11	Chapter Related work written
Snorri Hannesson	2015-03-11	Merge pull request #49 from teamDiversity/finalReport_ChapterRelatedWork_1
Balázs Kiss	2015-03-12	merged tex file
Snorrihann	2015-03-12	appendix change
Snorri Hannesson	2015-03-12	Merge pull request #52 from teamDiversity/finalReport_appendixtemplatechange

B Source Code

Mar 08, 15 18:33

TrafficSimulator.java

Page 1/3

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import trafficsimulator.gui.SimulationRenderer;
import trafficsimulator.gui.SimulationResults;
import trafficsimulator.simulations.Simulation1;
import trafficsimulator.simulations.Simulation2;

/**
 *
 * @author balazs
 */
public class TrafficSimulator extends Application {

    @Override
    public void start(final Stage primaryStage) {

        //main layout
        BorderPane root = new BorderPane();
        //canvas layout (white bg)
        StackPane canvas_holder = new StackPane();
        canvas_holder.setStyle("-fx-background-color: white");
        //create a control panel
        StackPane control_panel = new StackPane();
        control_panel.setStyle("-fx-backgrond-color: blue");
        //create canvas
        Canvas canvas = new Canvas(800,600);
        //add canvas to its holder
        canvas_holder.getChildren().add(canvas);
        //create a GraphicsContext
        GraphicsContext gc = canvas.getGraphicsContext2D();

```

Mar 08, 15 18:33

TrafficSimulator.java

Page 2/3

```

//add canvas layout into main layout
root.setLeft(canvas_holder);
//create set of option selectors (buttons, textfields, radio buttons...)
final Button startSim = new Button("Start");
startSim.setPrefSize(100, 50);
BorderPane button_pane = new BorderPane();
final Button showResults = new Button("Result");
showResults.setPrefSize(100, 50);
//showResults.setDisable(true);
HBox button_box = new HBox();
button_box.setPadding(new Insets(10,15,10,15));
button_box.setSpacing(25);
button_box.getChildren().addAll(startSim, showResults);
button_pane.setCenter(button_box);

ToggleGroup policies_selector = new ToggleGroup();
RadioButton fixed_time = new RadioButton("Fixed time policy");
RadioButton congestion_control = new RadioButton("Congestion control policy");
fixed_time.setToggleGroup(policies_selector);
congestion_control.setToggleGroup(policies_selector);
fixed_time.setSelected(true);
VBox policy_radio_box = new VBox();
policy_radio_box.setSpacing(15);
policy_radio_box.getChildren().addAll(fixed_time, congestion_control);
HBox policy_box = new HBox();
policy_box.setPadding(new Insets(10,15,10,15));
policy_box.setSpacing(10);
policy_box.getChildren().add(new Text("Policy: "));
policy_box.getChildren().add(policy_radio_box);

HBox duration_box = new HBox();
duration_box.setPadding(new Insets(10,15,10,15));
duration_box.setSpacing(10);
duration_box.getChildren().add(new Text("Duration: "));
TextField duration_field = new TextField();
duration_box.getChildren().add(duration_field);
duration_box.getChildren().add(new Text("seconds"));

HBox map_box = new HBox();
map_box.setPadding(new Insets(10,15,10,15));
map_box.setSpacing(10);
map_box.getChildren().add(new Text("Map: "));
ObservableList<String> options = FXCollections.observableArrayList("Map 1", "
Map 2", "Map 3");
ComboBox map_list = new ComboBox(options);
map_list.setValue("Map 1");
map_box.getChildren().add(map_list);

VBox container = new VBox();
container.setPadding(new Insets(10,15,10,15));
container.setSpacing(15);
container.getChildren().addAll(policy_box, duration_box, map_box, button_pane);
//add control panel into main layout
root.setCenter(container);
//create simulation
final Simulation1 simulation = new Simulation1();

```

Mar 08, 15 18:33

TrafficSimulator.java

Page 3/3

```

SimulationRenderer renderer = new SimulationRenderer(gc, simulation);
simulation.setRenderer(renderer);
//simulation.start();

//add function to option selectors
startSim.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
        simulation.start();
        startSim.setDisable(true);
    }
});
showResults.setOnAction(new EventHandler<ActionEvent>(){

    @Override
    public void handle(ActionEvent event){
        new SimulationResults(primaryStage, simulation);
        showResults.setDisable(true);
    }
});
//set stage config
primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {

    @Override
    public void handle(WindowEvent event) {
        System.exit(0);
    }
});
primaryStage.setTitle("TrafficSimulator");
primaryStage.setScene(new Scene(root,1200,700,Color.LIGHTGRAY));
//show stage
primaryStage.show();

}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    launch(args);
}
}

```

Mar 08, 15 13:41

Driver.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author Eddy
 */
public abstract class Driver {

    protected String name;
    protected Vehicle vehicle;

    public Driver(String name){
        this.name = name;
    }

    public void setVehicle(Vehicle vehicle){
        this.vehicle = vehicle;
    }

    abstract public double getOptimalDeceleration();

    public double getOptimalSpeedForDistance(double distance) {
        double speed = getOptimalDeceleration() * distance;

        // Capping for max speed
        if (speed > vehicle.getTopSpeed()) {
            speed = vehicle.getTopSpeed();
        }

        return speed;
    }

    public double getOptimalFollowingDistance() {
        double stoppingDistance = vehicle.getCurrentSpeed() / getOptimalDeceleration
();
        return 30.0 + stoppingDistance;
    }

    public boolean AccelerationStatus(double currentSpeed, double optimalFollowing
Dist, double distanceFromNextVechicle, double distanceFromEOLane) {
        boolean choice;
        //no car ahead
        if (distanceFromEOLane == Double.MAX_VALUE) {
            choice = true;
        }
        if (distanceFromNextVechicle <= optimalFollowingDist) {
            choice = false;
        } else {
            choice = true;
        }

        return choice;
    }
}

```

Mar 08, 15 13:41

Driver.java

Page 2/2

```
public boolean DecelerationStatus(double currentSpeed, double optimalFollowing
Dist, double distanceFromNextVechicle, double distanceFromEOLane) {
    boolean choice;
    if (distanceFromEOLane == Double.MAX_VALUE) {
        //This will depend on the state of the traffic light
    }
    if (distanceFromNextVechicle <= optimalFollowingDist) {
        choice = true;
    } else {
        choice = false;
    }

    return choice;
}
}
```

Mar 08, 15 13:41

EntryPoint.java

Page 1/1

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 *
 * @author balazs
 */
public class EntryPoint implements ISteppable{

    private Lane lane;
    private Map<Long, List<Vehicle>> steps = new HashMap<>();
    private Map<Vehicle, Long> vehicles = new HashMap<>();

    public EntryPoint(Lane lane) {
        this.lane = lane;
    }

    public Lane getLane() {
        return lane;
    }

    public void addVehicle(Vehicle vehicle, long step) {
        vehicles.put(vehicle, step);

        List stepList = steps.get(step);
        if (stepList == null) {
            stepList = new ArrayList<Vehicle>();
            steps.put(step, stepList);
        }
        stepList.add(vehicle);
        vehicle.startTime = System.currentTimeMillis();
    }

    public int numberOfVehicles() {
        return vehicles.size();
    }

    public void step(long step) {
        List<Vehicle> vehiclesForStep = steps.get(step);
        if (vehiclesForStep == null) {
            return;
        }
        for (Vehicle vehicle : vehiclesForStep) {
            //Add vehicle to system
            System.out.println(vehicle + " entered the system");
            vehicle.setLane(lane);
        }
    }
}

```

Mar 08, 15 13:41

ExitPoint.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author balazs
 */
public class ExitPoint {

    private Lane lane;
    private List<Vehicle> vehicles = new ArrayList<>();

    ExitPoint(Lane lane) {
        this.lane = lane;
    }

    public int numberOfVehicles() {
        return vehicles.size();
    }

    void addVehicle(Vehicle vehicle) {
        System.out.println(vehicle + " exited the system");
        vehicles.add(vehicle);
        vehicle.endTime = System.currentTimeMillis();
    }
}
```


Mar 08, 15 13:41

ISteppable.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author balazs
 */
public interface ISteppable {
    public void step(long step);
}
```

Mar 08, 15 13:41

Junction.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 *
 * @author balazs
 */
public abstract class Junction implements ISteppable{

    private HashMap<Lane, List<Lane>> connections;

    public Junction() {
        connections = new HashMap<>();
    }

    public void connect(Lane source, Lane destination) {
        if (!connections.containsKey(source)) {
            connections.put(source, new ArrayList<Lane>());
        }
        List<Lane> lanes = connections.get(source);
        lanes.add(destination);
        source.setJunction(this);
    }

    public List<Lane> getConnectedLanes(Lane lane) {
        return connections.get(lane);
    }

    public boolean shouldVehicleEnterJunction(Vehicle vehicle){
        return true;
    }
}
```

Mar 08, 15 13:41

Lane.java

Page 1/3

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;
import trafficsimulator.utils.Point;

/**
 *
 * @author balazs
 */
public class Lane {

    public static double laneWidth = 22;

    public enum Direction {

        IDENTICAL, OPPOSITE
    }

    private Road road;
    private List<Vehicle> vehicles = new ArrayList<>();
    private Junction junction;
    private Direction direction;
    private ExitPoint exitPoint;

    public Lane(Direction direction) {
        this.direction = direction;
        exitPoint = new ExitPoint(this);
    }

    public void enter(Vehicle vehicle) {
        vehicles.add(vehicle);
    }

    public void exit(Vehicle vehicle) {
        vehicles.remove(vehicle);
    }

    public Junction getJunction() {
        return junction;
    }

    public void setJunction(Junction junction) {
        this.exitPoint = null;
        this.junction = junction;
    }

    public ExitPoint getExitPoint() {
        return exitPoint;
    }

    public Road getRoad() {
        return road;
    }
}
```

Mar 08, 15 13:41

Lane.java

Page 2/3

```

public void setRoad(Road road) {
    this.road = road;
}

public Direction getDirection() {
    return direction;
}

public void setDirection(Direction direction) {
    this.direction = direction;
}

public Point getLeftStartPoint() {
    Road road = getRoad();
    int pos = road.getLaneIndexPosition(this);
    if (getDirection() == Direction.IDENTICAL) {
        return road.getLeftStartPoint().plus(acrossLaneVector().mult(pos));
    } else {
        return road.getLeftEndPoint().minus(acrossLaneVector().mult(pos + 1));
    }
}

public Point getLeftEndPoint() {
    Road road = getRoad();
    int pos = road.getLaneIndexPosition(this);
    if (getDirection() == Direction.IDENTICAL) {
        return road.getLeftEndPoint().plus(acrossLaneVector().mult(pos));
    } else {
        return road.getLeftStartPoint().minus(acrossLaneVector().mult((pos + 1)));
    }
}

private Point calculateRightPoints(Point p) {
    return p.plus(acrossLaneVector());
}

public Point getRightStartPoint() {
    return calculateRightPoints(getLeftStartPoint());
}

public Point getRightEndPoint() {
    return calculateRightPoints(getLeftEndPoint());
}

public Point getCenterStartPoint() {
    return (getLeftStartPoint().plus(getRightStartPoint())).div(2);
}

public Point getCenterEndPoint() {
    return (getLeftEndPoint().plus(getRightEndPoint())).div(2);
}

public Point getDirectionVector() {
    Road road = getRoad();
    if (getDirection() == Direction.IDENTICAL) {
        return road.getLeftEndPoint().minus(road.getLeftStartPoint());
    } else {
        return road.getLeftStartPoint().minus(road.getLeftEndPoint());
    }
}

```

Mar 08, 15 13:41

Lane.java

Page 3/3

```

    }
}

private Point acrossLaneUnitVector() {
    Point dir = getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    Point rotateUnitDir = unitDir.rotateVector(Math.PI / 2);
    return rotateUnitDir;
}

private Point acrossLaneVector() {
    double x = Math.floor(laneWidth * Math.cos(acrossLaneUnitVector().angleVector()));
    double y = Math.floor(laneWidth * Math.sin(acrossLaneUnitVector().angleVector()));
    return new Point(x, y);
}

public double getDistanceFromNextVehicle(Vehicle vehicle) {
    double minDistance = Double.MAX_VALUE;

    for (Vehicle v : vehicles) {
        if (vehicle == v) {
            continue;
        }

        double distance = vehicle.getPosition().distance(v.getPosition());
        if (distance < minDistance) {
            Point dir = v.getPosition().minus(vehicle.getPosition());
            if (dir.inSameQuadrant(getDirectionVector())) {
                minDistance = distance;
            }
        }
    }

    return minDistance;
}
}

```

Mar 08, 15 13:41

Map.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author balazs
 */
public class Map {

    private List<Road> roads;
    private List<Junction> junctions;

    public Map() {
        roads = new ArrayList<>();
        junctions = new ArrayList<>();
    }

    public List<Road> getRoads() {
        return roads;
    }

    public void addRoad(Road road) {
        roads.add(road);
    }

    public List<Junction> getJunctions() {
        return junctions;
    }

    public void addJunction(Junction junction) {
        junctions.add(junction);
    }

}
```

Mar 08, 15 13:41

Road.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;
import trafficsimulator.utils.Point;

/**
 *
 * @author balazs
 */
public class Road {

    private List<Lane> lanes;
    //The road is initialised by specifying the left paramiters of the road.
    //Each lane will be added to the right these paramiters and the right
    //paramiters of the road will be calculated by the numbers of lanes on the roa
d.
    private Point leftStartPoint;
    private Point leftEndPoint;

    public Road(Point leftStartPoint, Point leftEndPoint) {
        lanes = new ArrayList<>();
        this.leftStartPoint = leftStartPoint;
        this.leftEndPoint = leftEndPoint;
    }

    public void addLane(Lane lane) {
        lanes.add(lane);
        lane.setRoad(this);
    }

    public List<Lane> getLanes() {
        return lanes;
    }

    public void setLanes(List<Lane> lanes) {
        this.lanes = lanes;
    }

    public Point getLeftStartPoint() {
        return leftStartPoint;
    }

    public void setLeftStartPoint(Point leftStartPoint) {
        this.leftStartPoint = leftStartPoint;
    }

    public Point getLeftEndPoint() {
        return leftEndPoint;
    }

    public void setLeftEndPoint(Point leftEndPoint) {
        this.leftEndPoint = leftEndPoint;
    }
}

```

Mar 08, 15 13:41

Road.java

Page 2/2

```

public Point getRandomPosition() {
    Point dir = leftEndPoint.minus(leftStartPoint);
    return leftStartPoint.plus(dir.mult(Math.random()));
}

public Point getDirectionVector() {
    return leftEndPoint.minus(leftStartPoint);
}

public int getLaneIndexPosition(Lane l) {
    return lanes.indexOf(l);
}

public double calculateWidth() {
    double width = 0;
    for (Lane l : lanes) {
        width += Lane.laneWidth;
    }
    return width;
}

private Point acrossRoadUnitVector() {
    Point dir = getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    Point rotateUnitDir = unitDir.rotateVector(Math.PI / 2);
    return rotateUnitDir;
}

private Point acrossRoadVector() {
    double x = Math.round(calculateWidth() * Math.cos(acrossRoadUnitVector().angleVector()));
    double y = Math.round(calculateWidth() * Math.sin(acrossRoadUnitVector().angleVector()));
    return new Point(x, y);
}

public Point getRightStartPoint() {
    Point rightStartPoint = leftStartPoint.plus(acrossRoadVector());
    return rightStartPoint;
}

public Point getRightEndPoint() {
    Point rightEndPoint = leftEndPoint.plus(acrossRoadVector());
    return rightEndPoint;
}
}

```


Mar 08, 15 18:33

Simulation.java

Page 1/4

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import javafx.scene.text.Text;
import trafficsimulator.gui.IRenderer;

/**
 *
 * @author balazs
 */
public abstract class Simulation extends TimerTask {

    private long stepCounter = 0;
    protected Timer timer = new Timer();
    protected Map map = new Map();
    protected List<Vehicle> vehicles = new ArrayList<>();
    protected List<EntryPoint> entryPoints = new ArrayList<>();
    protected List<ExitPoint> exitPoints = new ArrayList<>();
    protected IRenderer renderer;

    public Simulation() {

    }

    public Simulation(IRenderer renderer) {
        this.renderer = renderer;
    }

    protected abstract void init();

    @Override
    public void run() {

        stepCounter++;
        System.out.println("Step " + stepCounter);

        if (numberOfVehiclesAtExitPoints() == vehicles.size()) {
            printStats();
            System.out.println("Simulation end");
            timer.cancel();
            return;
        }

        for (ISteppable ep : entryPoints) {
            ep.step(stepCounter);
        }

        for (ISteppable junction : map.getJunctions()) {
            junction.step(stepCounter);
        }
    }

```

Mar 08, 15 18:33

Simulation.java

Page 2/4

```

    for (ISteppable vehicle : getVehicles()) {
        vehicle.step(stepCounter);
    }

    if (renderer != null) {
        renderer.render();
    }
}

private EntryPoint getEntryPointForLane(Lane lane) {
    for (EntryPoint ep : entryPoints) {
        if (ep.getLane() == lane) {
            return ep;
        }
    }
    EntryPoint ep = new EntryPoint(lane);
    entryPoints.add(ep);
    return ep;
}

protected void addVehicle(Vehicle vehicle, Lane lane, long step) {
    EntryPoint ep = getEntryPointForLane(lane);
    ep.addVehicle(vehicle, step);
    vehicles.add(vehicle);
}

private List<ExitPoint> getExitPoints() {
    List<ExitPoint> exitPoints = new ArrayList<>();
    for (Road road : getMap().getRoads()) {
        for (Lane lane : road.getLanes()) {
            ExitPoint ep = lane.getExitPoint();
            if (ep == null) {
                continue;
            }
            exitPoints.add(ep);
        }
    }
    return exitPoints;
}

private int numberOfVehiclesAtExitPoints() {
    int n = 0;
    for (ExitPoint ep : exitPoints) {
        n += ep.numberOfVehicles();
    }
    return n;
}

public void start() {
    init();
    this.exitPoints = getExitPoints();
    timer.scheduleAtFixedRate(this, 0, 100);
}

public IRenderer getRenderer() {
    return renderer;
}

```

Mar 08, 15 18:33

Simulation.java

Page 3/4

```

public void setRenderer(IRenderer renderer) {
    this.renderer = renderer;
}

public Map getMap() {
    return map;
}

public List<Vehicle> getVehicles() {
    List<Vehicle> vehiclesInSystem = new ArrayList<>();
    for (Vehicle vehicle : vehicles) {
        if (!vehicle.isInSystem()) {
            continue;
        }
        vehiclesInSystem.add(vehicle);
    }
    return vehiclesInSystem;
}

public List<Vehicle> getExitedVehicles() {
    List<Vehicle> vehiclesInSystem = new ArrayList<>();
    for (Vehicle vehicle : vehicles) {
        if (vehicle.isInSystem()) {
            continue;
        }
        vehiclesInSystem.add(vehicle);
    }
    return vehiclesInSystem;
}

public void printStats() {
    for (Vehicle vehicle : getExitedVehicles()) {
        System.out.println(vehicle.getType() + " was in the system for " + vehicle.timeSpentInSystem() + " seconds");
    }
}

public Text averageTime() {
    double total = 0;
    double average = 0;
    for (Vehicle vehicle : getExitedVehicles()) {
        total += vehicle.timeSpentInSystem();
    }
    average = total/getExitedVehicles().size();

    if ( getExitedVehicles().isEmpty() ) return new Text( "Average time: 0" );
    else return new Text(String.valueOf( "Average time: " + average));
}

public Text longestTime() {
    double longest = 0;
    for (Vehicle vehicle : getExitedVehicles()) {
        if (longest < vehicle.timeSpentInSystem()) {
            longest = vehicle.timeSpentInSystem();
        }
    }
    if ( getExitedVehicles().isEmpty() ) return new Text( "Longest time: 0" );
    else return new Text(String.valueOf( "Longest time: " + longest));
}

```

Mar 08, 15 18:33

Simulation.java

Page 4/4

```
public Text shortestTime() {  
    double shortest = Integer.MAX_VALUE;  
    for (Vehicle vehicle : getExitedVehicles()) {  
        if (shortest > vehicle.timeSpentInSystem()) {  
            shortest = vehicle.timeSpentInSystem();  
        }  
    }  
    if ( getExitedVehicles().isEmpty() ) return new Text("Shortest time: 0");  
    else return new Text(String.valueOf("Shortest time: " + shortest));  
}
```

Mar 08, 15 13:41

TrafficSimulator.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author balazs
 */
public class TrafficSimulator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

    }

}
```

Mar 08, 15 18:16

Vehicle.java

Page 1/5

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.List;
import java.util.Random;
import trafficsimulator.utils.Point;
import trafficsimulator.utils.Size;

/**
 *
 * @author balazs
 */
public abstract class Vehicle implements ISteppable{

    private Lane lane;
    private Point position;
    private double currentSpeed = 0;
    protected double topSpeed;
    protected double maxAcceleration;
    protected double maxDeceleration;
    protected double optimalDeceleration;
    protected Size size;
    protected Driver driver;
    protected boolean accelerate;
    protected boolean decelerate;

    protected String type = "Vehicle Base Object";
    public long startTime = 0;
    public long endTime = 0;

    public Vehicle(Driver driver) {
        this.currentSpeed = 0;
        if (driver == null) {
            this.driver = NormalDriver("Default Driver");
        } else {
            this.driver = driver;
        }
        this.driver.setVehicle(this);
    }

    public Size getSize() {
        return size;
    }

    public double getTopSpeed() {
        return topSpeed;
    }

    public double getMaxAcceleration() {
        return maxAcceleration;
    }

    public double getMaxDeceleration() {
        return maxDeceleration;
    }
}

```

Mar 08, 15 18:16

Vehicle.java

Page 2/5

```

public String getType() {
    return type;
}

public Point getPosition() {
    return position;
}

public Lane getLane() {
    return lane;
}

public boolean isInSystem() {
    return lane != null;
}

public void setLane(Lane lane) {
    if (lane == null) {
        this.lane = null;
        return;
    }
    if (!isInSystem()) {
        this.position = lane.getCenterStartPoint();
    }
    this.lane = lane;
    this.lane.enter(this);
}

public double getCurrentSpeed() {
    return currentSpeed;
}

private void setCurrentSpeed(double speed) {
    if (speed > getTopSpeed()) {
        currentSpeed = getTopSpeed();
    } else if (speed < 0) {
        currentSpeed = 0;
    } else {
        currentSpeed = speed;
    }
}

private double getDistanceFromEOLane() {
    double distance = getLane().getLeftEndPoint().distance(this.getPosition());
    return distance;
}

private void changeSpeed() {
    accelerate = driver.AccelerationStatus(this.currentSpeed, driver.getOptimalFollowingDistance(), getLane().getDistanceFromNextVehicle(this), getDistanceFromEOLane());
    decelerate = driver.DecelerationStatus(this.currentSpeed, driver.getOptimalFollowingDistance(), getLane().getDistanceFromNextVehicle(this), getDistanceFromEOLane());

    if (accelerate) {
        accelerate();
    }
}

```

Mar 08, 15 18:16

Vehicle.java

Page 3/5

```

    } else if (decelerate) {
        decelerate();
    } else {
        currentSpeed = currentSpeed;
    }
}

private boolean leftRoad(Point oldPosition, Point newPosition) {
    Point endPoint = lane.getCenterEndPoint();
    if (oldPosition.getX() <= endPoint.getX() && newPosition.getX() > endPoint.g
etX()) {
        return true;
    }
    if (oldPosition.getX() >= endPoint.getX() && newPosition.getX() < endPoint.g
etX()) {
        return true;
    }
    if (oldPosition.getY() <= endPoint.getY() && newPosition.getY() > endPoint.g
etY()) {
        return true;
    }
    if (oldPosition.getY() >= endPoint.getY() && newPosition.getY() < endPoint.g
etY()) {
        return true;
    }
    return false;
}

private Lane chooseRandomNewLane() {
    Junction junction = lane.getJunction();
    if (junction == null) {
        return null;
    }
    List<Lane> lanes = junction.getConnectedLanes(lane);
    if (lanes.isEmpty()) {
        return null;
    }
    Random randomGenerator = new Random();
    int index = randomGenerator.nextInt(lanes.size());
    return lanes.get(index);
}

public Point getDisplacementVector() {
    Point dir = getLane().getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    double x = getCurrentSpeed() * Math.cos(unitDir.angleVector());
    double y = getCurrentSpeed() * Math.sin(unitDir.angleVector());
    return new Point(x, y);
}

public double timeSpentInSystem() {
    return (endTime - startTime) / 1000;
}

public void step(long stepCounter) {
    System.out.print(getType() + " #" + hashCode());

    // Change speed of vehicle
    changeSpeed();
}

```


Mar 08, 15 18:16

Vehicle.java

Page 4/5

```

// Calculate new position
Point newPosition = position.plus(getDisplacementVector());

// Check if vehicle has to change lane
if (leftRoad(this.position, newPosition)) {
    // Move vehicle to random next lane
    Lane newLane = chooseRandomNewLane();
    if (newLane != null) {
        this.lane.exit(this);
        this.position = newLane.getCenterStartPoint();
        this.setLane(newLane);
    } else {
        this.lane.exit(this);
        this.lane.getExitPoint().addVehicle(this);
        this.setLane(null);
    }
} else {
    // Move vehicle
    position = newPosition;
}

System.out.println(" position: " + Math.round(position.getX()) + ", " + Math.round(position.getY()) + " speed: " + Math.round(currentSpeed));
}

protected void accelerate() {
    double dist = getLane().getDistanceFromNextVehicle(this) - driver.getOptimalFollowingDistance();

    double optimalSpeed = driver.getOptimalSpeedForDistance(dist);

    if (optimalSpeed > getCurrentSpeed()) {
        double speedDifference = optimalSpeed - getCurrentSpeed();
        if (speedDifference < getMaxAcceleration()) {
            setCurrentSpeed(getCurrentSpeed() + speedDifference);
        } else {
            setCurrentSpeed(getCurrentSpeed() + getMaxAcceleration());
        }
    }
}

protected void decelerate() {
    double dist = getLane().getDistanceFromNextVehicle(this) - driver.getOptimalFollowingDistance();

    double optimalSpeed = driver.getOptimalSpeedForDistance(dist);

    if (optimalSpeed < getCurrentSpeed()) {
        double speedDifference = getCurrentSpeed() - optimalSpeed;
        if (speedDifference < getMaxDeceleration()) {
            setCurrentSpeed(getCurrentSpeed() - speedDifference);
        } else {
            setCurrentSpeed(getCurrentSpeed() - getMaxDeceleration());
        }
    }
}

```

Mar 08, 15 18:16

Vehicle.java

Page 5/5

```
private Driver NormalDriver(String default_Driver) {  
    throw new UnsupportedOperationException("Not supported yet."); //To change body o  
f generated methods, choose Tools / Templates.  
}  
  
}
```

Mar 08, 15 13:41

CautiousDriver.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class CautiousDriver extends Driver {

    public CautiousDriver(String name) {
        super(name);
    }

    @Override
    public double getOptimalDeceleration() {
        if (Car.class.isInstance(vehicle)) {
            return 3;
        } else if (Bus.class.isInstance(vehicle)) {
            return 2;
        } else {
            return 1;
        }
    }
}
```

Mar 08, 15 13:41

NormalDriver.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class NormalDriver extends Driver {

    public NormalDriver(String name) {

        super(name);
    }

    @Override
    public double getOptimalDeceleration() {
        if (Car.class.isInstance(vehicle)) {
            return 3;
        } else if (Bus.class.isInstance(vehicle)) {
            return 2;
        } else {
            return 1;
        }
    }
}
```

Mar 08, 15 13:41

RecklessDriver.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class RecklessDriver extends Driver {

    public RecklessDriver(String name) {

        super(name);
    }

    @Override
    public double getOptimalDeceleration() {
        if (Car.class.isInstance(vehicle)) {
            return 3;
        } else if (Bus.class.isInstance(vehicle)) {
            return 2;
        } else {
            return 1;
        }
    }
}
```

Mar 08, 15 13:41

IRenderer.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;

/**
 *
 * @author balazs
 */
public interface IRenderer {

    public void render();

}
```

Mar 08, 15 18:16

SimulationRenderer.java

Page 1/3

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;

import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import javafx.animation.FillTransition;
import javafx.animation.ParallelTransition;
import javafx.animation.RotateTransition;
import javafx.animation.ScaleTransition;
import javafx.animation.Timeline;
import javafx.animation.TranslateTransition;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.*;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.image.Image;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.transform.Rotate;
import javafx.stage.Stage;
import javafx.util.Duration;
import trafficsimulator.core.Lane;

import trafficsimulator.core.Lane.Direction;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author yukolthep
 */
public class SimulationRenderer implements IRenderer {

    private Stage stage;
    private Simulation simulation;
    private GraphicsContext gc;

    Image car_image = new Image("pic/car_tran.gif", 20, 0, true, false);
    Image car = new Image("pic/car.jpg");
    Image bus = new Image("pic/bus.jpg");

    public SimulationRenderer(GraphicsContext gc, Simulation simulation) {
        this.stage = stage;

```

Mar 08, 15 18:16

SimulationRenderer.java

Page 2/3

```

    this.simulation = simulation;
    this.gc = gc;
}

public void render() {
    Platform.runLater(new Runnable() {

        @Override
        public void run() {
            clear();
            drawRoads();
            drawLanes();
            drawVehicles();
        }
    });
}

/*Clear canvas before painting updated components*/
private void clear() {
    gc.clearRect(0, 0, 700, 700);
}

private void drawRoads() {
    List<Road> roads = this.simulation.getMap().getRoads();
    for (Road road : roads) {
        Point leftStartPoint = road.getLeftStartPoint();
        Point rightStartPoint = road.getRightStartPoint();
        Point leftEndPoint = road.getLeftEndPoint();
        Point rightEndPoint = road.getRightEndPoint();
        gc.setFill(Color.GRAY);
        gc.fillPolygon(new double[] {leftStartPoint.getX(), leftEndPoint.getX(), rightEndPoint.getX(), rightStartPoint.getX()}, new double[] {leftStartPoint.getY(), leftEndPoint.getY(), rightEndPoint.getY(), rightStartPoint.getY()}, 4);
    }
}

private void drawLanes(){
    List<Road> roads = this.simulation.getMap().getRoads();
    for (Road road : roads) {
        List<Lane> lanes = road.getLanes();
        for (int index = 0 ; index < lanes.size()-1 ; index++) {
            Lane lane = lanes.get(index);
            Point leftStartPoint = lane.getLeftStartPoint();
            Point leftEndPoint = lane.getLeftEndPoint();
            Point rightStartPoint = lane.getRightStartPoint();
            Point rightEndPoint = lane.getRightEndPoint();
            gc.setLineWidth(1);
            gc.setStroke(Color.BLACK);
            if(index == lanes.size()-1){
                break;
            }
            if(lane.getDirection() == Direction.IDENTICAL){
                gc.strokeLine(rightStartPoint.getX(), rightStartPoint.getY(), rightEndPoint.getX(), rightEndPoint.getY());
            }else{
                gc.strokeLine(leftStartPoint.getX(), leftStartPoint.getY(), leftEndPoint.getX(), leftEndPoint.getY());
            }
        }
    }
}

```


Mar 08, 15 18:16

SimulationRenderer.java

Page 3/3

```

    }
}

private void drawVehicles() {
    List<Vehicle> vehicles = this.simulation.getVehicles();
    for (Vehicle vehicle : vehicles) {
        if (Car.class.isInstance(vehicle)) {
            Double angle = vehicle.getDisplacementVector().angleVectorDegree();
            drawRotatedImage(gc, car, angle, (vehicle.getPosition().getX() - car.getWidth() / 2), (vehicle.getPosition().getY() - car.getHeight() / 2));
        } else if (Bus.class.isInstance(vehicle)) {
            Double angle = vehicle.getDisplacementVector().angleVectorDegree();
            drawRotatedImage(gc, bus, angle, (vehicle.getPosition().getX() - bus.getWidth() / 2), (vehicle.getPosition().getY() - bus.getHeight() / 2));
        }
    }
}

private void rotate(GraphicsContext gc, double angle, double px, double py) {
    Rotate r = new Rotate(angle, px, py);
    gc.setTransform(r.getMxx(), r.getMyx(), r.getMxy(), r.getMyy(), r.getTx(), r.getTy());
}

private void drawRotatedImage(GraphicsContext gc, Image image, double angle, double tlpX, double tlpY) {
    gc.save(); // saves the current state on stack, including the current transform
    rotate(gc, angle, tlpX + image.getWidth() / 2, tlpY + image.getHeight() / 2);
    gc.drawImage(image, tlpX, tlpY);
    gc.restore(); // back to original state (before rotation)
}
}

```

Mar 08, 15 18:33

SimulationResults.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;

import javafx.geometry.Rectangle2D;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Screen;
import javafx.stage.Stage;
import trafficsimulator.core.Simulation;

/**
 *
 * @author yukolthep
 */
public class SimulationResults extends Stage{
    public SimulationResults(Stage primaryStage, Simulation simulation){
        initModality(Modality.NONE);
        initOwner(primaryStage);
        VBox dialogVbox = new VBox(20);
        dialogVbox.getChildren().add(simulation.averageTime());
        dialogVbox.getChildren().add(simulation.longestTime());
        dialogVbox.getChildren().add(simulation.shortestTime());
        Scene dialogScene = new Scene(dialogVbox, 300, 200);
        setScene(dialogScene);
        Rectangle2D primScreenBounds = Screen.getPrimary().getVisualBounds();
        setX((primScreenBounds.getWidth() - getWidth()) / 2);
        setY((primScreenBounds.getHeight() - getHeight()) / 4);
        show();
    }
}
```

Mar 08, 15 13:41

TrafficLight.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.junctions;

import trafficsimulator.core.Lane;

/**
 *
 * @author balazs
 */
public class TrafficLight {
    public static final int GREEN_DURATION = 10;
    public static final int YELLOW_DURATION = 2;
    public static final int RED_DURATION = 10;
    public static final int REDYELLOW_DURATION = 3;

    public enum State {

        GREEN, YELLOW, RED, REDYELLOW
    }

    private State state = State.RED;
    private Lane lane;

    public TrafficLight(Lane lane){
        this.lane = lane;
    }

    public State getState() {
        return state;
    }

    public void setState(State state) {
        this.state = state;
    }

    public Lane getLane() {
        return lane;
    }

    public void nextState(){
        switch(state){
            case GREEN:
                setState(State.YELLOW);
                break;
            case YELLOW:
                setState(State.RED);
                break;
            case RED:
                setState(State.REDYELLOW);
                break;
            case REDYELLOW:
                setState(State.GREEN);
                break;
        }
    }
}

```

Mar 08, 15 13:41

TrafficLight.java

Page 2/2

```
}
```

Mar 08, 15 13:41

TrafficLightJunction.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.junctions;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Vehicle;

/**
 *
 * @author balazs
 */
public class TrafficLightJunction extends Junction{

    private List<TrafficLight> trafficLights = new ArrayList();
    private TrafficLight activeTrafficLight;
    private int stepCounter = 0;

    private TrafficLight getTrafficLightForLane(Lane lane){
        for(TrafficLight trafficLight : trafficLights){
            if(trafficLight.getLane()==lane){
                return trafficLight;
            }
        }
        return null;
    }

    public void connect(Lane source, Lane destination) {
        super.connect(source, destination);

        if (getTrafficLightForLane(source) == null) {
            TrafficLight trafficLight = new TrafficLight(source);
            trafficLights.add(trafficLight);
        }

        public boolean shouldVehicleEnterJunction(Vehicle vehicle){
            TrafficLight trafficLight = getTrafficLightForLane(vehicle.getLane());
            if(trafficLight.getState() == TrafficLight.State.GREEN){
                return true;
            }else{
                return false;
            }
        }

        private void activateTrafficLight(TrafficLight activeTrafficLight){
            // Making sure all traffic lights are red
            for(TrafficLight trafficLight : trafficLights){
                trafficLight.setState(TrafficLight.State.RED);
            }

            // Activating light

```

Mar 08, 15 13:41

TrafficLightJunction.java

Page 2/2

```

    this.activeTrafficLight = activeTrafficLight;
    activeTrafficLight.nextState();
    stepCounter = 0;
}

private void activateNextTrafficLight(){
    int index = trafficLights.indexOf(activeTrafficLight);
    if(index == trafficLights.size()-1){
        activateTrafficLight(trafficLights.get(0));
    }else{
        activateTrafficLight(trafficLights.get(index+1));
    }
}

public void step(long step) {
    if(activeTrafficLight == null){
        activateTrafficLight(trafficLights.get(0));
        return;
    }

    stepCounter++;

    if(activeTrafficLight.getState() == TrafficLight.State.GREEN && stepCounter
== TrafficLight.GREEN_DURATION){
        activeTrafficLight.nextState();
        stepCounter = 0;
    }else if(activeTrafficLight.getState() == TrafficLight.State.YELLOW && stepC
ounter == TrafficLight.YELLOW_DURATION){
        activateNextTrafficLight();
    }else if(activeTrafficLight.getState() == TrafficLight.State.REDYELLOW && st
epCounter == TrafficLight.REDYELLOW_DURATION){
        activeTrafficLight.nextState();
        stepCounter = 0;
    }else if(activeTrafficLight.getState() == TrafficLight.State.RED && stepCoun
ter == TrafficLight.RED_DURATION){
        activeTrafficLight.nextState();
        stepCounter = 0;
    }
}
}

```

Mar 08, 15 13:41

Simulation1.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.simulations;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author balazs
 */
public class Simulation1 extends Simulation{

    @Override
    protected void init() {
        Road r1 = new Road(new Point(70, 20), new Point(500, 20));
        Lane l11 = new Lane(Lane.Direction.IDENTICAL);
        Lane l12 = new Lane(Lane.Direction.OPPOSITE);
        r1.addLane(l11);
        r1.addLane(l12);
        Road r2 = new Road(new Point(500, 20), new Point(500, 450));
        Lane l21 = new Lane(Lane.Direction.IDENTICAL);
        Lane l22 = new Lane(Lane.Direction.OPPOSITE);
        r2.addLane(l21);
        r2.addLane(l22);
        Road r3 = new Road(new Point(500, 450), new Point(20, 100));
        Lane l31 = new Lane(Lane.Direction.IDENTICAL);
        Lane l32 = new Lane(Lane.Direction.OPPOSITE);
        r3.addLane(l31);
        r3.addLane(l32);
        Road r4 = new Road(new Point(500, 20), new Point(600, 20));
        Lane l41 = new Lane(Lane.Direction.IDENTICAL);
        Lane l42 = new Lane(Lane.Direction.OPPOSITE);
        r4.addLane(l41);
        r4.addLane(l42);
        Road r5 = new Road(new Point(600, 20), new Point(600, 450));
        Lane l51 = new Lane(Lane.Direction.IDENTICAL);
        Lane l52 = new Lane(Lane.Direction.OPPOSITE);
        r5.addLane(l51);
        r5.addLane(l52);
        Road r6 = new Road(new Point(600, 450), new Point(500, 450));
        Lane l61 = new Lane(Lane.Direction.IDENTICAL);
        Lane l62 = new Lane(Lane.Direction.OPPOSITE);
        r6.addLane(l61);
    }
}

```

Mar 08, 15 13:41

Simulation1.java

Page 2/2

```
r6.addLane(l62);
Road r7 = new Road(new Point(600, 450), new Point(650, 450));
Lane l71 = new Lane(Lane.Direction.IDENTICAL);
r7.addLane(l71);

Junction j1 = new TrafficLightJunction();
j1.connect(l11, l21);
j1.connect(l11, l41);
j1.connect(l22, l12);
j1.connect(l22, l41);
j1.connect(l42, l12);
j1.connect(l42, l21);
Junction j2 = new TrafficLightJunction();
j2.connect(l21, l31);
j2.connect(l21, l62);
j2.connect(l32, l22);
j2.connect(l32, l62);
j2.connect(l61, l22);
j2.connect(l61, l31);
Junction j3 = new TrafficLightJunction();
j3.connect(l31, l11);
j3.connect(l12, l32);
Junction j4 = new TrafficLightJunction();
j4.connect(l41, l51);
j4.connect(l52, l42);
Junction j5 = new TrafficLightJunction();
j5.connect(l51, l61);
j5.connect(l51, l71);
j5.connect(l62, l52);
j5.connect(l62, l71);

map.addRoad(r1);
map.addRoad(r2);
map.addRoad(r3);
map.addRoad(r4);
map.addRoad(r5);
map.addRoad(r6);
map.addRoad(r7);
map.addJunction(j1);
map.addJunction(j2);
map.addJunction(j3);
map.addJunction(j4);
map.addJunction(j5);

Driver tom = new CautiousDriver("Tom");
Driver mary = new NormalDriver("Mary");
Driver jerry = new RecklessDriver("Jerry");

addVehicle(new Car(tom), l11, 1);
addVehicle(new Bus(jerry), l11, 20);

}

}
```


Mar 08, 15 18:16

Simulation2.java

Page 1/2

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.simulations;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author yukolthep
 */
public class Simulation2 extends Simulation {

    @Override
    protected void init() {
        Road r1 = new Road(new Point(70, 300), new Point(270, 100));
        Lane l11 = new Lane(Lane.Direction.IDENTICAL);
        Lane l12 = new Lane(Lane.Direction.OPPOSITE);
        r1.addLane(l11);
        r1.addLane(l12);
        Road r2 = new Road(new Point(270, 100), new Point(470, 300));
        Lane l21 = new Lane(Lane.Direction.IDENTICAL);
        Lane l22 = new Lane(Lane.Direction.OPPOSITE);
        r2.addLane(l21);
        r2.addLane(l22);
        Road r3 = new Road(new Point(470, 300), new Point(270, 500));
        Lane l31 = new Lane(Lane.Direction.IDENTICAL);
        Lane l32 = new Lane(Lane.Direction.OPPOSITE);
        r3.addLane(l31);
        r3.addLane(l32);
        Road r4 = new Road(new Point(270, 500), new Point(70, 300));
        Lane l41 = new Lane(Lane.Direction.IDENTICAL);
        Lane l42 = new Lane(Lane.Direction.OPPOSITE);
        r4.addLane(l41);
        r4.addLane(l42);

        Junction j1 = new TrafficLightJunction();
        j1.connect(l11, l21);
        j1.connect(l22, l12);
        Junction j2 = new TrafficLightJunction();
        j2.connect(l21, l31);
        j2.connect(l32, l22);
        Junction j3 = new TrafficLightJunction();
        j3.connect(l31, l41);
        j3.connect(l42, l32);
    }

```

Mar 08, 15 18:16

Simulation2.java

Page 2/2

```
Junction j4 = new TrafficLightJunction();
j4.connect(141, 111);
j4.connect(112, 142);

map.addRoad(r1);
map.addRoad(r2);
map.addRoad(r3);
map.addRoad(r4);
map.addJunction(j1);
map.addJunction(j2);
map.addJunction(j3);
map.addJunction(j4);

Driver tom = new CautiousDriver("Tom");
Driver jerry = new RecklessDriver("Jerry");

Vehicle olo = new Car(tom);
Vehicle olo_v2 = new Car(jerry);

addVehicle(olo, 111, 1);
addVehicle(olo_v2, 122, 1);

}

}
```

Mar 08, 15 18:16

Point.java

Page 1/3

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.utils;

/**
 *
 * @author balazs
 */
public class Point {

    public double x, y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }

    public Point plus(Point p) {
        return new Point(this.x + p.x, this.y + p.y);
    }

    public Point minus(Point p) {
        return new Point(this.x - p.x, this.y - p.y);
    }

    public Point mult(double k) {
        return new Point(this.x * k, this.y * k);
    }

    public Point div(double k) {
        return new Point(this.x / k, this.y / k);
    }

    public double distanceFromOrigin() {
        Point origin = new Point();
```

Mar 08, 15 18:16

Point.java

Page 2/3

```

    return distance(origin);
}

public double distance(Point p) {
    double dx = x - p.x;
    double dy = y - p.y;
    double distance = Math.sqrt(dx * dx + dy * dy);
    return distance;
}

public boolean inSameQuadrant(Point p) {
    if (getX() > 0 && p.getX() < 0) {
        return false;
    }
    if (getX() < 0 && p.getX() > 0) {
        return false;
    }
    if (getY() > 0 && p.getY() < 0) {
        return false;
    }
    if (getY() < 0 && p.getY() > 0) {
        return false;
    }
    return true;
}

public Point rotateVector(double degrees) {
    double X = Math.round(this.x * Math.cos(degrees) - this.y * Math.sin(degrees));
    double Y = Math.round(this.x * Math.sin(degrees) + this.y * Math.cos(degrees));
    return new Point(X, Y);
}

public double angleVector() {
    if (y == 0) {
        if (x < 0) {
            return Math.PI;
        } else {
            return 0;
        }
    } else if (x < 0) {
        if (y > 0) {
            return Math.atan(this.y / this.x) + Math.PI;
        } else {
            return Math.atan(this.y / this.x) - Math.PI;
        }
    } else {
        return Math.atan(this.y / this.x);
    }
}

public double angleVectorDegree() {
    if (y == 0) {
        if (x < 0) {
            return Math.PI*(180/Math.PI);
        } else {
            return 0;
        }
    }
}

```

Mar 08, 15 18:16

Point.java

Page 3/3

```
    } else if (x < 0) {  
        if (y > 0) {  
            return (Math.atan(this.y / this.x) + Math.PI)*(180/Math.PI);  
        } else {  
            return (Math.atan(this.y / this.x) - Math.PI)*(180/Math.PI);  
        }  
    } else {  
        return Math.atan(this.y / this.x)*(180/Math.PI);  
    }  
}  
}
```

Mar 08, 15 13:41

Size.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.utils;

/**
 *
 * @author balazs
 */
public class Size {

    public double width;
    public double height;

    public Size(double width, double height) {
        this.width = width;
        this.height = height;
    }
}
```

Mar 08, 15 13:41

Bus.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.vehicles;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Size;

/**
 *
 * @author snorri
 */
public class Bus extends Vehicle {

    public Bus() {
        this(null);
    }

    public Bus(Driver driver) {
        super(driver);
        topSpeed = 6;
        maxAcceleration = 1;
        maxDeceleration = 3;
        optimalDeceleration = 2;
        size = new Size(20, 10);
    }

    @Override
    public String getType() {
        return "Bus";
    }
}
```

Mar 08, 15 13:41

Car.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.vehicles;

import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Size;
import trafficsimulator.core.Driver;

/**
 *
 * @author snorri
 */
public class Car extends Vehicle {

    public Car() {
        this(null);
    }

    public Car(Driver driver) {
        super(driver);
        topSpeed = 10;
        maxAcceleration = 2;
        maxDeceleration = 4;
        optimalDeceleration = 3;
        size = new Size(14, 8);
    }

    @Override
    public String getType() {
        return "Car";
    }
}
```


Mar 08, 15 13:41

JunctionTest.java

Page 1/1

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.List;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
import trafficsimulator.utils.Point;

/**
 *
 * @author snorri
 */
public class JunctionTest {

    private Lane lane1;
    private Lane lane2;

    @Before
    public void setUp() {

        Road r1 = new Road(new Point(20, 20), new Point(500, 20));
        lane1 = new Lane(Lane.Direction.IDENTICAL);
        lane2 = new Lane(Lane.Direction.IDENTICAL);
        r1.addLane(lane1);
        r1.addLane(lane2);
    }

    /**
     * Test whether a lane can connect to itself at a junction. This test is not
     * complete
     */
    @Test
    public void testLanesJunction1() {
        System.out.println("Opposite lanes at a junction");

        Junction junction = new Junction();
        junction.connect(lane1, lane1);

        fail(); // We shouldn't be able to get to this point
    }
}

```

Mar 08, 15 13:41

LaneTest.java

Page 1/5

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.Test;
import static org.junit.Assert.*;
import trafficsimulator.utils.Point;

/**
 *
 * @author snorri
 */
public class LaneTest {

    private Lane lane1;
    private Lane lane2;
    private Road road;

    public void setUp(Point start, Point end, Lane.Direction dir1, Lane.Direction
dir2) {

        road = new Road(start, end);
        lane1 = new Lane(dir1);
        lane2 = new Lane(dir2);
        road.addLane(lane1);
        road.addLane(lane2);

    }

    /**
     * This test checks if right and left parameters are calculated correctly if a
     * horizontal road is created and its direction is to the right like this: ->
     * with two lanes that are both IDENTICAL.
     */
    @Test
    public void testLanesHorizontalRight() {
        System.out.println("Lanes horizontal right");

        Point startLeft = new Point(100, 100);
        Point endLeft = new Point(400, 100);

        setUp(startLeft, endLeft, Lane.Direction.IDENTICAL, Lane.Direction.IDENTICAL
);
        /*
        System.out.println("lane1LeftStartPointX: " + lane1.getLeftStartPoint().get
X());
        System.out.println("lane1LeftStartPointY: " + lane1.getLeftStartPoint().get
Y());
        System.out.println("lane1LeftEndPointX: " + lane1.getLeftEndPoint().getX())
;
        System.out.println("lane1LeftEndPointY: " + lane1.getLeftEndPoint().getY())
;
        System.out.println("lane1RightStartPointX: " + lane1.getRightStartPoint().g
etX());
        System.out.println("lane1RightStartPointY: " + lane1.getRightStartPoint().g
etY());

```

Mar 08, 15 13:41

LaneTest.java

Page 2/5

```

        System.out.println("lane1RightEndPointX: " + lane1.getRightEndPoint().getX(
));
        System.out.println("lane1RightEndPointY: " + lane1.getRightEndPoint().getY(
));
        System.out.println("lane2LeftStartPointX: " + lane2.getLeftStartPoint().get
X());
        System.out.println("lane2LeftStartPointY: " + lane2.getLeftStartPoint().get
Y());
        System.out.println("lane2LeftEndPointX: " + lane2.getLeftEndPoint().getX())
;
        System.out.println("lane2LeftEndPointY: " + lane2.getLeftEndPoint().getY())
;
        System.out.println("lane2RightStartPointX: " + lane2.getRightStartPoint().g
etX());
        System.out.println("lane2RightStartPointY: " + lane2.getRightStartPoint().g
etY());
        System.out.println("lane2RightEndPointX: " + lane2.getRightEndPoint().getX(
));
        System.out.println("lane2RightEndPointY: " + lane2.getRightEndPoint().getY(
));
        */
        double expXLeftStartLane1 = startLeft.getX();
        double expYLeftStartLane1 = startLeft.getY();
        double expXRightStartLane1 = startLeft.getX();
        double expYRightStartLane1 = startLeft.getY() + lane1.laneWidth;
        double expXLeftEndLane1 = endLeft.getX();
        double expYLeftEndLane1 = endLeft.getY();
        double expXRightEndLane1 = endLeft.getX();
        double expYRightEndLane1 = endLeft.getY() + lane1.laneWidth;
        double expXLeftStartLane2 = startLeft.getX();
        double expYLeftStartLane2 = startLeft.getY() + lane1.laneWidth;
        double expXRightStartLane2 = startLeft.getX();
        double expYRightStartLane2 = startLeft.getY() + lane1.laneWidth + lane2.lane
Width;
        double expXLeftEndLane2 = endLeft.getX();
        double expYLeftEndLane2 = endLeft.getY() + lane1.laneWidth;
        double expXRightEndLane2 = endLeft.getX();
        double expYRightEndLane2 = endLeft.getY() + lane1.laneWidth + lane2.laneWidt
h;

        double resultXLeftStartLane1 = lane1.getLeftStartPoint().getX();
        double resultYLeftStartLane1 = lane1.getLeftStartPoint().getY();
        double resultXRightStartLane1 = lane1.getRightStartPoint().getX();
        double resultYRightStartLane1 = lane1.getRightStartPoint().getY();
        double resultXLeftEndLane1 = lane1.getLeftEndPoint().getX();
        double resultYLeftEndLane1 = lane1.getLeftEndPoint().getY();
        double resultXRightEndLane1 = lane1.getRightEndPoint().getX();
        double resultYRightEndLane1 = lane1.getRightEndPoint().getY();
        double resultXLeftStartLane2 = lane2.getLeftStartPoint().getX();
        double resultYLeftStartLane2 = lane2.getLeftStartPoint().getY();
        double resultXRightStartLane2 = lane2.getRightStartPoint().getX();
        double resultYRightStartLane2 = lane2.getRightStartPoint().getY();
        double resultXLeftEndLane2 = lane2.getLeftEndPoint().getX();
        double resultYLeftEndLane2 = lane2.getLeftEndPoint().getY();
        double resultXRightEndLane2 = lane2.getRightEndPoint().getX();
        double resultYRightEndLane2 = lane2.getRightEndPoint().getY();

        assertEquals(expXLeftStartLane1, resultXLeftStartLane1, 2.1);
        assertEquals(expYLeftStartLane1, resultYLeftStartLane1, 2.1);

```

Mar 08, 15 13:41

LaneTest.java

Page 3/5

```

    assertEquals(expXRightStartLane1, resultXRightStartLane1, 2.1);
    assertEquals(expYRightStartLane1, resultYRightStartLane1, 2.1);
    assertEquals(expXLeftEndLane1, resultXLeftEndLane1, 2.1);
    assertEquals(expYLeftEndLane1, resultYLeftEndLane1, 2.1);
    assertEquals(expXRightEndLane1, resultXRightEndLane1, 2.1);
    assertEquals(expYRightEndLane1, resultYRightEndLane1, 2.1);

    assertEquals(expXLeftStartLane2, resultXLeftStartLane2, 2.1);
    assertEquals(expYLeftStartLane2, resultYLeftStartLane2, 2.1);
    assertEquals(expXRightStartLane2, resultXRightStartLane2, 2.1);
    assertEquals(expYRightStartLane2, resultYRightStartLane2, 2.1);
    assertEquals(expXLeftEndLane2, resultXLeftEndLane2, 2.1);
    assertEquals(expYLeftEndLane2, resultYLeftEndLane2, 2.1);
    assertEquals(expXRightEndLane2, resultXRightEndLane2, 2.1);
    assertEquals(expYRightEndLane2, resultYRightEndLane2, 2.1);

}

/**
 * This test checks if right and left parameters are calculated correctly if a
 * horizontal road is created and its direction is to the right like this: ->
 * with two lanes where one is IDENTICAL and the other OPPOSITE.
 */
@Test
public void testLanesHorizontalRight2() {
    System.out.println("Lanes horizontal right");

    Point startLeft = new Point(100, 100);
    Point endLeft = new Point(400, 100);

    setUp(startLeft, endLeft, Lane.Direction.IDENTICAL, Lane.Direction.OPPOSITE);
;
    /*
    System.out.println("lane1LeftStartPointX: " + lane1.getLeftStartPoint().get
X());
    System.out.println("lane1LeftStartPointY: " + lane1.getLeftStartPoint().get
Y());
    System.out.println("lane1LeftEndPointX: " + lane1.getLeftEndPoint().getX())
;
    System.out.println("lane1LeftEndPointY: " + lane1.getLeftEndPoint().getY())
;
    System.out.println("lane1RightStartPointX: " + lane1.getRightStartPoint().g
etX());
    System.out.println("lane1RightStartPointY: " + lane1.getRightStartPoint().g
etY());
    System.out.println("lane1RightEndPointX: " + lane1.getRightEndPoint().getX(
));
    System.out.println("lane1RightEndPointY: " + lane1.getRightEndPoint().getY(
));
    System.out.println("lane2LeftStartPointX: " + lane2.getLeftStartPoint().get
X());
    System.out.println("lane2LeftStartPointY: " + lane2.getLeftStartPoint().get
Y());
    System.out.println("lane2LeftEndPointX: " + lane2.getLeftEndPoint().getX())
;
    System.out.println("lane2LeftEndPointY: " + lane2.getLeftEndPoint().getY())
;
    System.out.println("lane2RightStartPointX: " + lane2.getRightStartPoint().g
etX());

```

Mar 08, 15 13:41

LaneTest.java

Page 4/5

```

        System.out.println("lane2RightStartPointY: " + lane2.getRightStartPoint().getY());
        System.out.println("lane2RightEndPointX: " + lane2.getRightEndPoint().getX());
        System.out.println("lane2RightEndPointY: " + lane2.getRightEndPoint().getY());
    */
    double expXLeftStartLane1 = startLeft.getX();
    double expYLeftStartLane1 = startLeft.getY();
    double expXRightStartLane1 = startLeft.getX();
    double expYRightStartLane1 = startLeft.getY() + lane1.laneWidth;
    double expXLeftEndLane1 = endLeft.getX();
    double expYLeftEndLane1 = endLeft.getY();
    double expXRightEndLane1 = endLeft.getX();
    double expYRightEndLane1 = endLeft.getY() + lane1.laneWidth;
    double expXLeftStartLane2 = road.getRightEndPoint().getX();
    double expYLeftStartLane2 = road.getRightEndPoint().getY();
    double expXRightStartLane2 = road.getRightEndPoint().getX();
    double expYRightStartLane2 = road.getRightEndPoint().getY() - 22;
    double expXLeftEndLane2 = road.getRightStartPoint().getX();
    double expYLeftEndLane2 = road.getRightStartPoint().getY();
    double expXRightEndLane2 = road.getRightStartPoint().getX();
    double expYRightEndLane2 = road.getRightStartPoint().getY() - 22;

    double resultXLeftStartLane1 = lane1.getLeftStartPoint().getX();
    double resultYLeftStartLane1 = lane1.getLeftStartPoint().getY();
    double resultXRightStartLane1 = lane1.getRightStartPoint().getX();
    double resultYRightStartLane1 = lane1.getRightStartPoint().getY();
    double resultXLeftEndLane1 = lane1.getLeftEndPoint().getX();
    double resultYLeftEndLane1 = lane1.getLeftEndPoint().getY();
    double resultXRightEndLane1 = lane1.getRightEndPoint().getX();
    double resultYRightEndLane1 = lane1.getRightEndPoint().getY();
    double resultXLeftStartLane2 = lane2.getLeftStartPoint().getX();
    double resultYLeftStartLane2 = lane2.getLeftStartPoint().getY();
    double resultXRightStartLane2 = lane2.getRightStartPoint().getX();
    double resultYRightStartLane2 = lane2.getRightStartPoint().getY();
    double resultXLeftEndLane2 = lane2.getLeftEndPoint().getX();
    double resultYLeftEndLane2 = lane2.getLeftEndPoint().getY();
    double resultXRightEndLane2 = lane2.getRightEndPoint().getX();
    double resultYRightEndLane2 = lane2.getRightEndPoint().getY();

    assertEquals(expXLeftStartLane1, resultXLeftStartLane1, 2.1);
    assertEquals(expYLeftStartLane1, resultYLeftStartLane1, 2.1);
    assertEquals(expXRightStartLane1, resultXRightStartLane1, 2.1);
    assertEquals(expYRightStartLane1, resultYRightStartLane1, 2.1);
    assertEquals(expXLeftEndLane1, resultXLeftEndLane1, 2.1);
    assertEquals(expYLeftEndLane1, resultYLeftEndLane1, 2.1);
    assertEquals(expXRightEndLane1, resultXRightEndLane1, 2.1);
    assertEquals(expYRightEndLane1, resultYRightEndLane1, 2.1);

    assertEquals(expXLeftStartLane2, resultXLeftStartLane2, 2.1);
    assertEquals(expYLeftStartLane2, resultYLeftStartLane2, 2.1);
    assertEquals(expXRightStartLane2, resultXRightStartLane2, 2.1);
    assertEquals(expYRightStartLane2, resultYRightStartLane2, 2.1);
    assertEquals(expXLeftEndLane2, resultXLeftEndLane2, 2.1);
    assertEquals(expYLeftEndLane2, resultYLeftEndLane2, 2.1);
    assertEquals(expXRightEndLane2, resultXRightEndLane2, 2.1);
    assertEquals(expYRightEndLane2, resultYRightEndLane2, 2.1);

```

Mar 08, 15 13:41

LaneTest.java

Page 5/5

```
}  
  
}
```

Mar 08, 15 13:41

RoadTest.java

Page 1/5

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.Test;
import static org.junit.Assert.*;
import trafficsimulator.utils.Point;

/**
 *
 * @author snorri
 */
public class RoadTest {

    private Road road;

    public void setUp(Point start, Point end) {
        road = new Road(start, end);
        Lane lane1 = new Lane(Lane.Direction.IDENTICAL);
        Lane lane2 = new Lane(Lane.Direction.OPPOSITE);
        road.addLane(lane1);
        road.addLane(lane2);
    }

    /**
     * This test checks if right parameters are calculated correctly if a
     * horizontal road is created and its direction is to the right, like this: ->
     * .
     */
    @Test
    public void testRoadHorizontalRight() {
        System.out.println("Road horizontal right");

        setUp(new Point(100, 100), new Point(400, 100));

        double expYStart = road.getLeftStartPoint().getY() + road.calculateWidth();
        double expXStart = road.getLeftStartPoint().getX();
        double expYEnd = road.getLeftEndPoint().getY() + road.calculateWidth();
        double expXEnd = road.getLeftEndPoint().getX();

        double resultYStart = road.getRightStartPoint().getY();
        double resultXStart = road.getRightStartPoint().getX();
        double resultYEnd = road.getRightEndPoint().getY();
        double resultXEnd = road.getRightEndPoint().getX();

        assertEquals(expYStart, resultYStart, 0.001);
        assertEquals(expXStart, resultXStart, 0.001);
        assertEquals(expYEnd, resultYEnd, 0.001);
        assertEquals(expXEnd, resultXEnd, 0.001);
    }

    /**
     * This test checks if right parameters are calculated correctly if a
     * horizontal road is created and its direction is to the left, like this: <-
     * .
     */

```

Mar 08, 15 13:41

RoadTest.java

Page 2/5

```

    */
    @Test
    public void testRoadHorizontalLeft() {
        System.out.println("Road horizontal left");

        setUp(new Point(400, 100), new Point(100, 100));

        double expYStart = road.getLeftStartPoint().getY() - road.calculateWidth();
        double expXStart = road.getLeftStartPoint().getX();
        double expYEnd = road.getLeftEndPoint().getY() - road.calculateWidth();
        double expXEnd = road.getLeftEndPoint().getX();

        double resultYStart = road.getRightStartPoint().getY();
        double resultXStart = road.getRightStartPoint().getX();
        double resultYEnd = road.getRightEndPoint().getY();
        double resultXEnd = road.getRightEndPoint().getX();

        assertEquals(expYStart, resultYStart, 0.001);
        assertEquals(expXStart, resultXStart, 0.001);
        assertEquals(expYEnd, resultYEnd, 0.001);
        assertEquals(expXEnd, resultXEnd, 0.001);
    }

    /**
     * This test checks if right parameters are calculated correctly if a vertical
     * road is created and its direction is to the up, like this: ^ | .
     */
    @Test
    public void testRoadVerticalUp() {
        System.out.println("Road vertical up");

        setUp(new Point(100, 400), new Point(100, 100));

        double expYStart = road.getLeftStartPoint().getY();
        double expXStart = road.getLeftStartPoint().getX() + road.calculateWidth();
        double expYEnd = road.getLeftEndPoint().getY();
        double expXEnd = road.getLeftEndPoint().getX() + road.calculateWidth();

        double resultYStart = road.getRightStartPoint().getY();
        double resultXStart = road.getRightStartPoint().getX();
        double resultYEnd = road.getRightEndPoint().getY();
        double resultXEnd = road.getRightEndPoint().getX();

        assertEquals(expYStart, resultYStart, 0.001);
        assertEquals(expXStart, resultXStart, 0.001);
        assertEquals(expYEnd, resultYEnd, 0.001);
        assertEquals(expXEnd, resultXEnd, 0.001);
    }

    /**
     * This test checks if right parameters are calculated correctly if a vertical
     * road is created and its direction is to the right, like this: | v .
     */
    @Test
    public void testRoadVerticalDown() {
        System.out.println("Road vertical down");

```


Mar 08, 15 13:41

RoadTest.java

Page 3/5

```

    setUp(new Point(100, 100), new Point(100, 400));

    double expYStart = road.getLeftStartPoint().getY();
    double expXStart = road.getLeftStartPoint().getX() - road.calculateWidth();
    double expYEnd = road.getLeftEndPoint().getY();
    double expXEnd = road.getLeftEndPoint().getX() - road.calculateWidth();

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);
}

/**
 * This test checks if right parameters are calculated correctly if a road
 * that has a downward slope is created and its direction is to the right,
 * like this: \ v .
 */
@Test
public void testRoadDownwardRight() {
    System.out.println("Road downward right");

    setUp(new Point(100, 100), new Point(400, 400));

    double expYStart = road.getLeftStartPoint().getY() + 31;
    double expXStart = road.getLeftStartPoint().getX() - 31;
    double expYEnd = road.getLeftEndPoint().getY() + 31;
    double expXEnd = road.getLeftEndPoint().getX() - 31;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);
}

/**
 * This test checks if right parameters are calculated correctly if a road
 * that has a downward slope is created and its direction is to the left, like
 * this: / v .
 */
@Test
public void testRoadDownwardLeft() {
    System.out.println("Road downward left");

    setUp(new Point(400, 100), new Point(100, 400));

    double expYStart = road.getLeftStartPoint().getY() - 31;

```

Mar 08, 15 13:41

RoadTest.java

Page 4/5

```

    double expXStart = road.getLeftStartPoint().getX() - 31;
    double expYEnd = road.getLeftEndPoint().getY() - 31;
    double expXEnd = road.getLeftEndPoint().getX() - 31;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);
}

/**
 * This test checks if right parameters are calculated correctly if a road
 * that has an upward slope is created and its direction is to the right, like
 * this: ^ / .
 */
@Test
public void testRoadUpwardRight() {
    System.out.println("Road upward right");

    setUp(new Point(100, 400), new Point(400, 100));

    double expYStart = road.getLeftStartPoint().getY() + 31;
    double expXStart = road.getLeftStartPoint().getX() + 31;
    double expYEnd = road.getLeftEndPoint().getY() + 31;
    double expXEnd = road.getLeftEndPoint().getX() + 31;

    double resultYStart = road.getRightStartPoint().getY();
    double resultXStart = road.getRightStartPoint().getX();
    double resultYEnd = road.getRightEndPoint().getY();
    double resultXEnd = road.getRightEndPoint().getX();

    assertEquals(expYStart, resultYStart, 0.001);
    assertEquals(expXStart, resultXStart, 0.001);
    assertEquals(expYEnd, resultYEnd, 0.001);
    assertEquals(expXEnd, resultXEnd, 0.001);
}

/**
 * This test checks if right parameters are calculated correctly if a road
 * that has an upward slope is created and its direction is to the left, like
 * this: ^ \ .
 */
@Test
public void testRoadUpwardLeft() {
    System.out.println("Road upward left");

    setUp(new Point(400, 400), new Point(100, 100));

    double expYStart = road.getLeftStartPoint().getY() - 31;
    double expXStart = road.getLeftStartPoint().getX() + 31;
    double expYEnd = road.getLeftEndPoint().getY() - 31;
    double expXEnd = road.getLeftEndPoint().getX() + 31;

```

Mar 08, 15 13:41

RoadTest.java

Page 5/5

```
double resultYStart = road.getRightStartPoint().getY();
double resultXStart = road.getRightStartPoint().getX();
double resultYEnd = road.getRightEndPoint().getY();
double resultXEnd = road.getRightEndPoint().getX();

assertEquals(expYStart, resultYStart, 0.001);
assertEquals(expXStart, resultXStart, 0.001);
assertEquals(expYEnd, resultYEnd, 0.001);
assertEquals(expXEnd, resultXEnd, 0.001);

}

}
```

Mar 08, 15 13:41

TrafficSimulatorTestSuite.java

Page 1/1

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

//JUnit Suite Test
@RunWith(Suite.class)
@Suite.SuiteClasses({
    RoadTest.class,
    VehicleTest.class,
    JunctionTest.class,
    LaneTest.class
})

/**
 *
 * @author snorri
 */
public class TrafficSimulatorTestSuite {
}
```

Mar 08, 15 13:41

VehicleTest.java

Page 1/3

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import org.junit.Test;
import static org.junit.Assert.*;

import trafficsimulator.utils.Point;
import trafficsimulator.utils.Size;
import trafficsimulator.vehicles.*;

/**
 *
 * @author snorri This class will be changing a lot. After the changes more
 * thorough testing will be done.
 */
public class VehicleTest {

    /**
     * Test the height of a reckless car.
     */
    @Test
    public void testHeightRecklessCar() {
        System.out.println("Height of a reckless car");

        Lane lane = new Lane(Lane.Direction.IDENTICAL);
        Vehicle recklessCar = new Car();

        Size expResult = new Size(14, 8);
        Size result = recklessCar.getSize();

        assertEquals(expResult.height, result.height, 0.001);
    }

    /**
     * Test the height of a reckless bus.
     */
    @Test
    public void testHeightRecklessBus() {
        System.out.println("Height of a reckless bus");

        Lane lane = new Lane(Lane.Direction.IDENTICAL);
        Vehicle recklessBus = new Bus();

        Size expResult = new Size(20, 10);
        Size result = recklessBus.getSize();

        assertEquals(expResult.height, result.height, 0.001);
    }

    /**
     * Test if a reckless bus moves
     */
    @Test
    public void testRecklessBusMovement() {
        System.out.println("Movement of a reckless bus");
    }

```

Mar 08, 15 13:41

VehicleTest.java

Page 2/3

```

Road road = new Road(new Point(20, 20), new Point(500, 20));
Lane lane = new Lane(Lane.Direction.IDENTICAL);
road.addLane(lane);
Vehicle recklessBus = new Bus();

double initialPos = recklessBus.getPosition().getX();
recklessBus.step();
double finalPos = recklessBus.getPosition().getX();

assertTrue(finalPos > initialPos);
}

/**
 * Test vehicle outside of road boundaries
 */
@Test
public void testRecklessBusOutsideRoad1() {
    System.out.println("Creation of vehicle outside of Road parameter");

    Road road = new Road(new Point(20, 20), new Point(500, 20));
    Lane lane = new Lane(Lane.Direction.IDENTICAL);
    road.addLane(lane);
    Vehicle recklessBus = new Bus();

    double roadStartX = road.getLeftStartPoint().getX();
    double roadStartY = road.getLeftStartPoint().getY();
    double roadEndX = road.getLeftEndPoint().getX();
    double roadEndY = road.getLeftEndPoint().getY();
    double recklessBusX = recklessBus.getPosition().getX();
    double recklessBusY = recklessBus.getPosition().getY();

    assertTrue((recklessBusX >= roadStartX && recklessBusX <= roadEndX)
        || (recklessBusY >= roadStartY && recklessBusY <= roadEndY));
}

@Test
public void testRecklessBusOutsideRoad2() {
    System.out.println("Movement of a vehicle outside of road");

    final Road road = new Road(new Point(20, 20), new Point(500, 20));
    final Lane lane = new Lane(Lane.Direction.IDENTICAL);
    road.addLane(lane);
    final Vehicle recklessBus = new Bus();

    Simulation s = new Simulation() {

        @Override
        protected void init() {
            map.addRoad(road);
            addVehicle(recklessBus, lane, 1);
        }
    };

    double roadStartX = road.getLeftStartPoint().getX();
    double roadStartY = road.getLeftStartPoint().getY();
    double roadEndX = road.getLeftEndPoint().getX();
    double roadEndY = road.getLeftEndPoint().getY();
    double recklessBusX = recklessBus.getPosition().getX();

```

Mar 08, 15 13:41

VehicleTest.java

Page 3/3

```
double recklessBusY = recklessBus.getPosition().getY();

assertTrue((recklessBusX >= roadStartX && recklessBusX <= roadEndX)
           || (recklessBusY >= roadStartY && recklessBusY <= roadEndY));
}
}
```

Project description

Background Over the past few decades, the world's population has been continuously increasing. This increase in population has resulted in overwhelming traffic which is becoming a serious problem in most major cities around the world. Cities like London, Bangkok, and New York are faced with serious traffic congestion challenges. To be able to handle these challenges and make the traffic run smoother many different traffic management policies have been tried. Before these policies can be implemented in the real world they are tested on traffic simulators. These simulator are supposed to be abstract models of the real world, so if a traffic management policy works on the simulator it probably works in the real world. In this module we will develop a traffic simulator.

Objectives

- To develop a traffic simulator program which has the following structure: two types of vehicles, three types of drivers, functional road system with many roads and lanes, junctions, intersections and traffic lights.
- To compare two different traffic management policies: Fixed Time Policy and Congestion Control Policy.
- To examine how the system reacts in a time of emergency by injecting an ambulance to the simulator.

Scale

- Each road can have multiple lanes, which can be in the same or opposite direction.
- As in Britain the traffic is left-lane oriented.
- The system will have cars and buses.
- Driver behaviour can be cautious, reckless, or normal.

Outline

Description : The traffic simulator is an abstract model of actual real world traffic.

The simulator will have both cars and buses. Drivers' behaviour can be cautious, reckless or normal. Two different traffic management policies will be implemented which are supposed to relief traffic congestion issues. The two policies are fixed time policy and congestion control policy. These policies can be compared by average time each vehicle is in the system. Moreover, an emergency strategy will be implemented to see how the system reacts when an ambulance is injected to the system.

The simulator will be programmed in Java programming language. The simulator will have a graphical user interface (GUI). The GUI is created with the help of JavaFX software platform. The rationale for using a GUI: 1. Better visualisation and understanding of code during development, i.e. actually seeing what is happening when programming collision detection. 2. When the final product is ready

users can see the road system and the cars and therefore get a better understanding of how the road system is and how the policies work. Opposed to just get the result log and results of which policy is superior and have no visual understanding of what happened.

General Concept :

1. Vehicle Types: There are two types of vehicles: cars and buses. Cars go faster and have higher acceleration than buses. The shape and size of the bus is bigger than car, so it takes more space on the road.
2. Driver Behaviour: The drivers can behave in three different ways: cautious, reckless, and normal. Reckless drivers drive faster than normal but the cautious drivers drive slower than normal. Reckless drivers are more prone to change lanes and overtake other vehicles, if a fast vehicle is stuck behind a slow vehicle the fast vehicle will try to overtake the slow vehicle.
3. Emergency Strategy: When an ambulance is injected to the system it gets highest priority. That means that cars on the right lane will change to the left lane if possible to free the right lane for the ambulance. Also, the ambulance gets to pass at the intersection first.
4. Traffic Management Policy: There are two different traffic management policies; *Fixed Time policy* and *Congestion Control policy*:

The **Fixed Time Policy** will have peak hours (when the traffic density is the highest) and off-peak hours (when the traffic density is normal). On peak hours will the duration of the green light be longer than on off-peak hours.

The **Congestion Control Policy** will always have the same duration of green light in each direction unless the system detects a congestion. When a congestion is detected in some direction the duration of the green light will be increased in that direction for one time to avoid traffic jam.

The two different policies will be compared by average time each vehicle is in the system. Each vehicle will have a timer that starts when it enters the system and gets written to a log when the vehicle exits the system. The average time that a vehicle is in the system is calculated and the the policy that generates lower average time is considered better.

Schedule

The time period is divided into three iterations and then the tasks allocated to the iterations. Each task is either mandatory or optional for the simulator:

Expectation of Project Outcome

- The traffic simulator is developed and implemented.
- Results showing which traffic management policy is superior or if they are equal.

	Mandatory				Optional	
Iteration 1: 16th of January - 12th of February	Roads	Lanes	Multiple vehicles	Different driver behaviour		
Iteration 2: 13th of February - 5th of March	Junctions and intersections	Traffic lights			Different traffic management policies	
Iteration 3: 6th of March - 26th of March					Time logging	Emergency Strategy

- The simulator can be used in case of emergency.

Current progress We are currently finished with iteration 1 and we are on schedule. We have implemented the roads, lanes that can go in the same or opposite direction, different vehicles and different drivers' behaviour. However, we are still trying to find the best architecture for the different drivers' behaviour and vehicles.

As the name of the team implies; we are a very diversified group. From four different countries in three different continents, each with different programming background and experience. Although some of the members of the team were familiar with git and GitHub, LaTeX and some of the other technical tools we are using, some team members weren't. For everyone in the team to be able to make an contribution we have spent a quite a lot of time discussing those technical tools.

The biggest challenge so far is to get everyone on the same page about the project and the tools used for developing and collaboration.

Project organisation **Project management** A slightly adjusted Agile Methodology is used for this project. We have divided the time period for this course into iterations and at the end of each iteration we want to be inching forward towards the final product.

Roles

Balázs Kiss: Lead programmer

Eddy Mukasa: Architect

Yukolthep Visessmit: Graphical designer

Pongsakorn N. Riyamongkol: Project Manager

Snorri Hannesson: Tester and Coordinator

Each member of the group has a responsibility to oversee one aspect of the project but is not expected to do all the work defined in his role. I.e. each member should/could do some programming but the lead programmer should oversee the code and make sure nothing is missing and everything is done properly. The same goes for the other roles.

Collaboration We have physical meeting every Thursday at 10 o'clock. We have a log of all meetings. Other means of collaborations are:

Facebook is used as a communication channel.

GitHub is used for version control.

Trello is used for project management.

Peer Assessment and Self Assessment Team members should evaluate their own and other members' performance regarding work done in the group. This can be done by using an assessment form where every member secretly grades themselves and other members of the team. Then can contributions to **GitHub** and **Trello** be examined for evaluation of members performance.

Conflicts If we have a conflict during this project the will have democracy; the majority decides. However, if a major conflict arises we will have to contact the instructor to resolve the conflict. We have defined a simple method to resolve conflicts:

- I. Realise conflict.
- II. Handle conflict sooner rather than later.
- III. Find the solution together - democracy.
- IV. Apologise.
- V. Appreciate.