1 Project description

1.1 Introduction

The Traffic Simulation System can simulate various kinds of vehicles that are travelling on a complicated road network. The road network contains multi-lane roads, intersections and roundabouts. The user can manage the simulation by changing the quantity of vehicles, setting traffic lights on chosen intersections, changing time step, and so on. The real-time situation of the traffic helps the user to compare and find out which traffic plan works better.

1.2 Features of the system

FE-1: The Traffic Simulation System should simulate different kinds of vehicles, including cars, buses and ambulances.
FE-2: The Traffic Simulation System should build a road network which contains multi-lane roads, intersections and roundabouts.
FE-3: The Traffic Simulation System should have traffic lights and allow the user to manage relevant policy to control the traffic.
FE-4: The Traffic Simulation System should allow the user to define where the vehicles enter and leave the road network.
FE-5: The Traffic Simulation System should allow the user to add arbitrary quantity and different kinds of vehicles into the road network at any time.
FE-6: The Traffic Simulation System should allow the user to decide the behaviors of drivers of cars, which includes reckless behavior, cautious behavior and normal behavior.
FE-7: The Traffic Simulation System should present real time situation of the traffic.
FE-8: The Traffic Simulation System should allow the user to control the simulation.

1.3 Use cases

UC-1: Define quantities of different vehicles: car, bus, ambulance
UC-2: Set traffic lights on chosen intersections
UC-3: Define entry/exit points for vehicles
UC-4: Choose drivers' behaviours
UC-5: See real-time traffic situation and statistics
UC-6: Stop, start and change simulation playback speed and timestep

1.4 Extent of version

We have defined 3 stages of the program's completion:

- First version - version for initial report and presetation
- Second version - the minimum viable product
- Third version - optional features we want to implement if we have time

They are described in detail in the table on page 3.

1.5 Progress

Our target was to complete the first version for the initial presentation and we have achieved it. The code already uses roughly the architecture created for the final version, so it can be easily extended with new features.

2 Project organisation

2.1 Tools

As specified in the requirements, we use:

- Git and GitHub - for version control and repository management
- LaTeX- for creating documentation

For drawing diagrams for the documentation, we use Gliffy.

We have chosen Java 8 as the language to implement our project with. To make building of the project and dependency management as simple as possible, we employ Apache Maven. This in turn enables to use Travis Continuous Integration - a free service that checks if every commit pushed to the repository builds and if the tests run.

For testing, we use the JUnit library.

For distributing tasks and controlling progress, we use GitHub Issues.

2.2 Process

We use an Agile method to manage out project. Every team member is expected to work on the project around 10 hours per week. There are week-long iterations, which start with a meeting every Thursday. During the meeting, we establish the goals to achieve in the iteration and every team

Table 1: Extent of versions

Feature	Version1	Version2	Version3
FE-1	One type of vehicle, car which only stops when the traffic light turns red.	Tow types of vehicle, car and bus. Bus can stop shortly at bus stops.	Three types of vehicles: car, bus and ambulance.
FE-2	One road which has no passing lane, and one intersection.	Multi-lane roads with intersections.	Roundabouts.
FE-3	One traffic light.	Many traffic lights, allow the user to set traffic lights in an intersection.	User can choose the main road.
FE-4	Single entry and exit point.	Random entry and exit points.	Entry and exit points are defined by user.
FE-5	The system has a specific number of cars.	User can add arbitrary quantity of cars and buses.	User can add ambulances into the system.
FE-6	Default behaviour is normal.	User can choose drivers' behaviors from three types: reckless behavior, cautious behavior and normal behavior.	Done.
FE-7	Present the color of traffic lights.	Display the average speed of all vehicles in the simulation, number of different vehicles, different behaviors of the drivers.	Done.
FE-8	No control over the simulation.	User can control the simulation by changing time step and changing time for switching lights.	User can control the system by changing the speed.

member is assigned tasks to complete within the timeframe of the iteration. If any tasks are leftover from the previous iteration, they are also reassigned.

We maintain three levels of branches in the Git repository:

- *master* - branch for working code which integrates and is usable. We make a *master* snapshot by merging changes from *dev* at the end of every iteration. All code has to build and pass its tests on Travis.
- *dev* - the development branch. Here, the team members merge their new features and work on integrating them together. All code has to build and pass its tests on Travis.
- Feature branches - active development branches. Normally, there is one branch per task belonging to one team member. They are merged into dev when the task is finished (or when the task achieved a stable, partially completed state).

When a team member starts working on task, they create a new feature branch with the name specified in the task. The names are all prefixed with *feat-*.

When the work is done, they submit a pull request closing the task. At this point, another team member reviews the code, checking the Travis build and tests status and potentially making suggestions about improvements.

If the reviewer and the submitter are both satisfied with the state of the pull request, the reviewer closes the pull request and the task by merging the changes into *dev*. If the reviewer and the submitter cannot reach an agreement on some issue, they ask for help from a third person, a mediator. In extreme cases, the issue is resulted by team-wide vote.

2.3 Peer assessment

There are two peer assessment rounds, one for the first half of the project (3rd March) and one for the second (31st March).

On each round, each team member anonymously allocates 100 points to team members on their ballot, based how big they think their contributions were. The ballots are then shuffled. The ballots are used to reach an agreement about what the point distribution should be. If an agreement is not reached, an average of the distributions is taken and the result is scaled up so it sums to 100 again.

After the second round, the team members discuss what should be the final distribution based on the two distributions from two halves of the project. If an agreement is not reached, the averaging procedure described above is used.