Requirements and Analysis Document for Group 5

Gustaf Ringius, Andreas Löfman, Robert Wennergren, Felix Bärring

May 8, 2014

Contents

1 Introduction

Chaotic Traffic Simulator simulates the chaos that one could find in traffic by spawning cars with personalities that can change based on environmental variables that the user controls and give snickers to the driver if he/she is getting too hungry and stressed and turn into a driving diva.

This project uses and displays the concurrency of the world in a fun and chaotic way where chaos is actually seen as the objective to achieve by the user of the program. It also displays active objects traveling over a graph continuously calculating the shortest path to the destination taking into account traffic jams and queues.

1.1 Purpose of application

To simulate traffic on a fun childish level by changing variables that will affect the behavior of the drivers of the cars. It is mostly fun and somewhat educational where the user can see how the thinking and decisions are affected as the user gets more stressed from some environmental variables.

1.2 General characteristics of application

The application will be a desktop, stand alone with a graphical user interface for the Windows/Mac/Linux platforms.

The application will be started with displaying a configuration window where the user can see the current random variables where the user also has the possibly to change any environmental variables or just press a big start button to use the current setup. When a simulation is started it will run continuously until the user has had enough or until the entire map is in total chaos and everything is just a big traffic jam. It should be possible to create multiple different situations on every run of the simulation with different variables even if the end result might be very similar.

1.3 Scope of application

In the minimum case the user only have to press start to watch a beautiful simulation with a potential to create chaos on the screen. The user can also input different variables to the chaos before it starts and the user could while the application is running change the variables. For example turning a traffic light on and off for specific lanes on the road.

1.4 Objectives and success criteria of the project

1. It should be possible to watch the chaos of the application until a complete jam occurs.
2. It should be possible to change from a deadlocked/jammed state by changing the variables in the world and it should also be possible to alter the personalities of the drivers.
3. The system can be controlled but also unleashed into total chaos by making the drivers more stressed and removing any traffic controls like traffic lights.

4. The cars actively calculate the shortest path to their destination taking into account any possible traffic jams that will increase the time count.
5. The drivers of the cars have to make an active decision if the way around potential traffic jams is worth the extra traveling time.

1.5 Definitions, acronyms and abbreviations

- CTS, Chaos Traffic Simulator
- GUI, graphical user interface
- JRE, Java Runtime Environment. Additional software needed to run a Java application
- Deadlock, a situation where two or more parties want to use the same exclusive resource at the exact same time leading to the parties waiting indefinitely for each other.
- Concurrency, is the possibility to run something sequentially or in parallel.

2 Requirements

2.1 Functional requirements

The user should be able to, note that these requirements do not have to be linked to an explicit GUI button or panel but a console should be supoplied in the first release to where it is possible to change the variables.

1. Change global variables
    - Change the Weather
    - Change current time of the day
2. Start the Simulation
3. Solve a deadlock/jam
4. Exit the application
5. End a simulation
6. Read a help dialog
7. Read an about dialog
8. Add a tile
9. Generate a worldMap
10. Add traffic lights
11. Remove traffic lights

12. Control the traffic light

Optional

- Select a single car
- Track a single car
- Change the properties of a car
- Change the color
- Change the stress level of the drivers
- Change the destination

2.2 Non-functional requirements

The cars will calculate the shortest path to its destination and may account for traffic jams in the application. The cars could show the user which way they are thinking of going and the user should see the path change if a better path is found by the car. This should only be displyed in the GUI if there's enough time and priority for such functionality. The shortest route to the destination may be going around the traffic jam but if the way around the jam is too heavy in terms of driving distance compared to the waiting time in the jam the driver has to make a choice.

Cars may have to know about the priority to the right rule. That means that if there's no traffic light in a crossing the cars have to try to obey this rule to avoid collisions but it is also possible for a driver to ignore the rule if the stress level is high enough.

2.2.1 Usability

The application should have some nice sound and graphics and indicate if a button was pressed and should also explain somewhat what the user has or could do to setup a custom simulation.

2.2.2 Reliability

The application should never hang the computer but it may slow it down quite noticeably.

2.2.3 Performance

The program aim to utilize as many available threads on the computer as possible and link these to a thread in the application.

2.2.4 Supportability

We give no support at all. You run this at your own risk and evaluate the strength of your computer by yourself. The application is not built only for computers with parallel support and can be run on single core machines to execute sequentially but it is highly recommended that the user runs this program on a multi threaded computer system.

2.2.5 Implementation

To achieve platform independence the application will use the Java environment. All hosts must have the JRE installed and configured. The application needs to be installed on all hosts where it will run (possibly downloaded).

Since the project will be made in an agile manner we reserve ourselves the right to make potential changes in the models as problems, better solutions and alternatives are developed or brought to our attention.

2.2.6 Packaging and installation

The application will be delivered as a zip-archive containing;

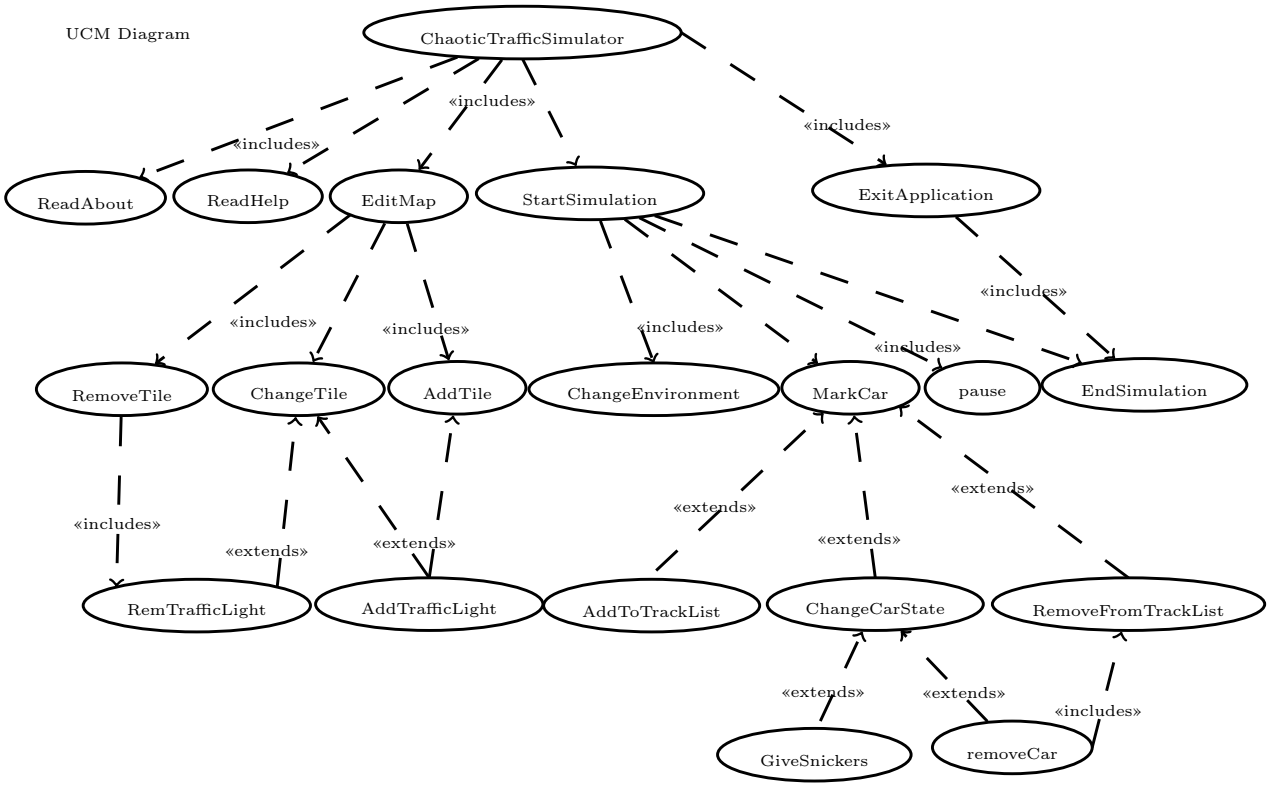
1. A file for the application code (a standard Java jar-file).
2. All needed resources
3. Start program script to start the game on different platforms.
4. A copy of the MIT license.
5. A README-file documenting installation and start of application.

2.2.7 Legal

There are no legal issues with this application if the MIT license is followed.

2.3 Application models

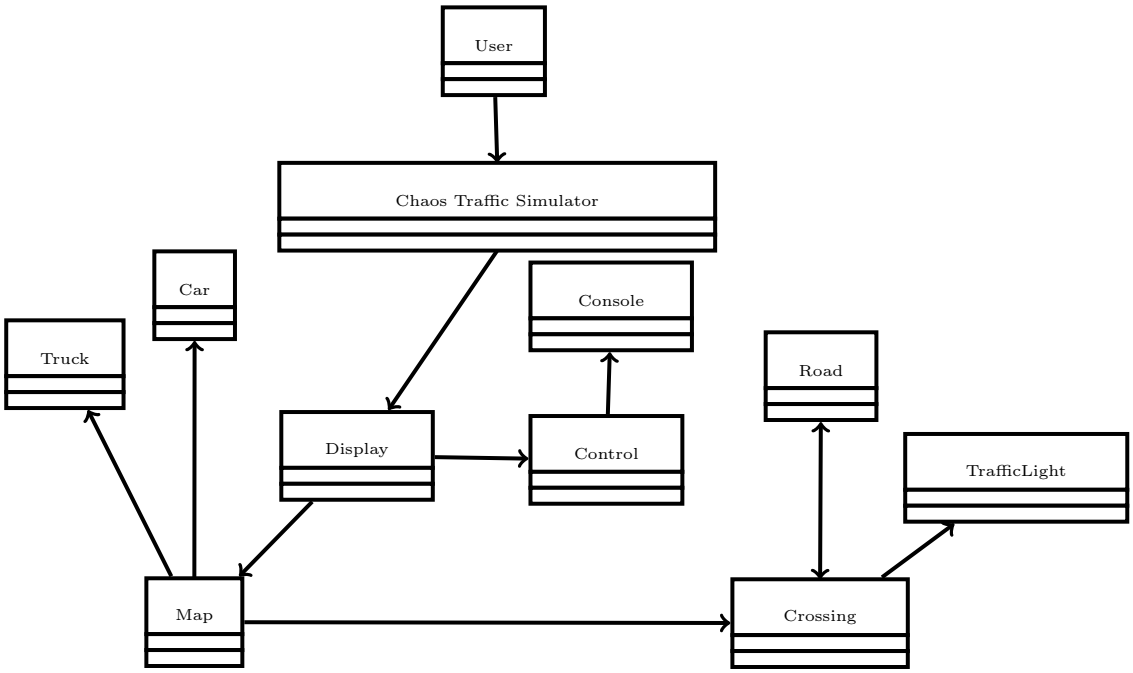
2.3.1 Use case model



2.3.2 Use cases priority

We aim to actually continue to work on this project as a hobby even after the course is finished so we want to have a solid model to expand on and a presentable GUI for the presentation but we do not aim to get all functionality in at the same time. We will create the base for all our thoughts for future expansion. This means that we will prioratize the development of the shortest path and generation algorithm for example and not making everything completely linked and finished all the way to the GUI.

2.3.3 Domain model



2.3.4 User interface

Application will use a fixed (non skin-able, non theme-able) GUI following standard conventions. The GUI must take into account different screen sizes, minimum 800x600.

2.4 References