Project

ETH Zurich

Hand-out: 12 March 2013
Team registration: 19 March 2013
Due: 7 May 2013

1 Synopsis

The Federal Office of Transportation wants to simulate changes to their traffic system. The Swiss government pays well, so this is a great opportunity for your small team of developers. You have to hand in a prototype that can successfully simulate a subset of the current traffic system.

2 Requirements

You must implement a traffic simulation that handles cars driving on roads, including crossroads (T and X). A car cannot suddenly change its velocity, but only accelerate and decelerate (break). The acceleration rate, deceleration rate and top speed are specific for each car.

Most of the roads have a speed limit. The prototype can assume that all the cars obey the speed limit but still drive as fast as possible given the situation.

Cars have to be able to spot other cars and drive accordingly in order to not crash into each other. A car takes up 6 meters of the road it travels along. The center of the car is considered the coordinate for the log. A car follows a route to its destination and disappears from the simulation on arrival.

If the edges of more than two roads are connected, then this point is considered a crossroad. Only one car can be located on a crossroad at any any point in time. Some crossroads may (optionally) be controlled by traffic lights. A traffic light permits a car to travel from one road to another during its green phase.

The input to the program is a file containing a traffic situation. Every line in the file describes one element. Those elements are:

car $n\ x_1\ y_1\ x_2\ y_2\ a\ d\ v$ A car named n starting at point (x_1, y_1), going to (x_2, y_2) with a maximal acceleration rate of a, deceleration rate of d (d is always negative) and top speed of v. Coordinates are given in meters and speed in meters per second.

road $n\ x_1\ y_1\ x_2\ y_2\ v$ A straight road segment named n from (x_1, y_1) to (x_2, y_2) with a speed limit of v. If a road segment's end point matches another road segment's start point, those two roads are connected. Roads are always one-way. A two-way road is represented by two roads next to each other. When more than two roads are to be connected, then the connecting point is a crossroad.

phase $t_1\ t_2\ t_3\ n_1\ n_2$ Optional. Specifies a phase for traffic light controlled crossroads. t_1 is the time (in seconds) before the phase is activated. t_2 is the time the phase is activated. t_3 is the time after the phase is activated. After $t_1 + t_2 + t_3$ the process repeats. n_1 and

n_2 specifies the incoming and outgoing road segments the phase controls. If during the times where the phase is active a car wants to go from the end of n_1 to the start of n_2 it is allowed to do so.

Note that numerical values may be non-integral, so using a floating point type as a representation may be useful.

All movements of cars are events. The time granularity of the simulation is one second. A moving car generates therefore an event every virtual second. All events are saved in the event log:

s n **position** x y At second s after the start of simulation the car named n is at position (x, y)

s n **arrived** x y At second s after the start of simulation the car named n arrived at its destination (x, y)

This simulation is concurrent. It is therefore important that the components are in sync concerning the events. To speed up the simulation, a time tolerance of up to one (virtual) second is permitted, as it is below the reaction delay of a driver. All cars in the simulation have to account for this delay when calculating safety margins. For synchronizing events, take a look at the Wikipedia articles about Lamport timestamps [2] and the Vector clock [3].

To develop and debug your system, you will need support for visualizing the traffic simulation. A starting point for a visualization is provided on the course website [1]. This application is written in C++ and uses the Qt library (this should make it relatively easy to use across different platforms). You may need to add extra functionality to make it suitable for your project.

3 Deliverables

3.1 Report (25 points)

The project report must include the following elements:

- An overview of the system design and its implementation. This should include an explanation of how SCOOP was used to implement the requirements. (12 Points)
- A description of your test cases and an explanation how they cover the requirements. (5 Points)
- A discussion of SCOOP's usability for the task at hand. Analyze the aspects that SCOOP handled elegantly and analyze the aspects that were difficult to express with SCOOP. (4 Points)

The report will also be expected to have a clear structure and proper grammar and spelling. (4 Points)

3.2 Design and implementation (25 points)

Design and implementation of the simulation should adhere to the requirements (20 Points). The code should be well-documented (3 Points) and well-structured (2 Points).

4 Teams

You can work in teams of up to three persons. Please send an e-mail to Mischael Schill (mischael.schill@inf.ethz.ch) to register your team. The due date for the registration is indicated at the top of this assignment.

5 Support

You have the possibility to ask questions during the exercise sessions. If you need immediate help, you can also contact the assistants directly. For resources on SCOOP and EiffelStudio, you can also consult the course website [1].

6 Updates

Please check the course website [1] for potential updates of the project description and supporting material.

7 Submission

Please submit a zip file that contains the source code and the report in PDF format by e-mail to Mischael Schill (mischael.schill@inf.ethz.ch) by the due date. The source code should be *without* the EIFGENs directory (which contains compiler-generated files).

Your zip file should contain two folders: a *source* folder for the source code and a *documentation* folder for the report.

References

[1] ETH Zurich. CCC. http://se.inf.ethz.ch/courses/2013a_spring/ccc/, 2013.

[2] Wikipedia. Lamport timestamps. http://en.wikipedia.org/wiki/Lamport_timestamps, 2013.

[3] Wikipedia. Vector clock. http://en.wikipedia.org/wiki/Vector_clock, 2013.