9.11.2018
Juho Sillanpää, 589903
Samuli Räikkönen, 427146
Elias Hinkkanen, 605609
Niko Saurio, 603452

Traffic simulator - programming project plan

Scope of the project

- A basic user configurable traffic simulator that fulfills the assignment requirements

Class structure

Building

- Residential (where people live, no children will probably be implemented)
- Industrial (where people work)
    - Commercial (Where people spend their free time. Inherits from Industrial, as people also work there)
        - Shop
        - Gym
        - etc.
- People park their cars inside buildings
- Size of one Square, but allocates more than one Square

Human

- Different types of people, which have different properties. The user can choose the proportions (employed, retired, unemployed, yms)
- Driving behaviour
- Where they work, and which time of the day
- What they do after work
- Which Routes use for commuting
- Lives in a residential building (maybe not alone)

Car

- Acceleration
- Capacity
- Knows the speed limit

- Knows its own location on the map by precision of a decimal number
- Communicates with intersections and other cars
- Has a size

<u>Road</u>

- List of squares that belong to it
- Speed limit
- Knowledge of its endpoints
- Direction
- Knows what cars it contains

<u>Intersections</u>

- Traffic lights which can be optimized for the differing amounts of traffic
- A car sends a request to pass and the intersection answers
- Rules of traffic
- Different types of intersections (with and without traffic lights/crosswalks/etc.)

<u>Route</u>

- Contains the information of a route a car takes
- Choosing the best route will require some work

<u>Square</u>

- Knows its coordinates in the grid
- Knows if it’s part of a road
- Knows if it’s part of a building

<u>Engine</u>

- Runs the simulation
- Contains the tools for analysis
- Uses clock-functionality for real-time analysis and rendering (for the GUI)

<u>GUI</u>

- Runs the possibly to be implemented graphical interface

<u>FileReader</u>

- Reads the city information from a file
- Possibly saves the specifics of a city, if a city is customizable
- Creates a file for the analytic data
- Information is saved in a table

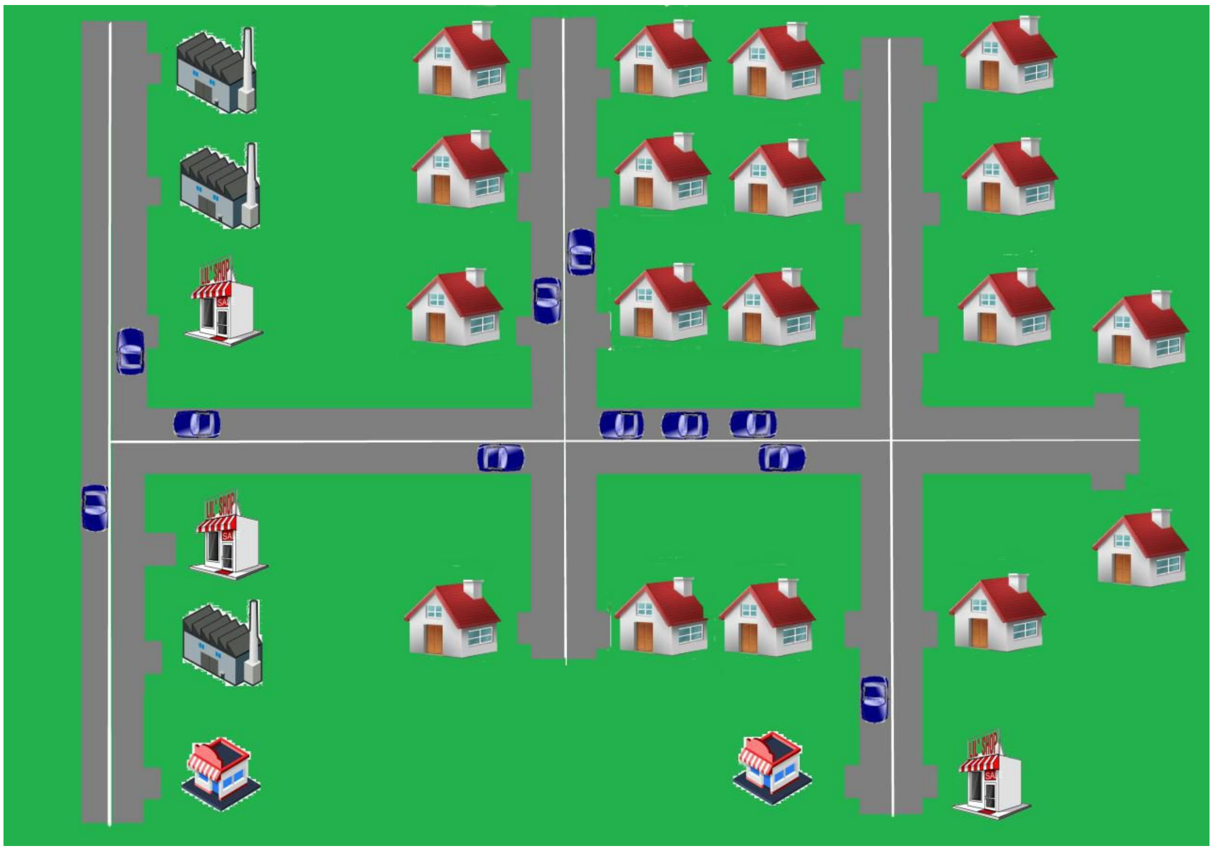
Architectural decisions

Data of the city will be stored in a file, which contains the information in a table. Engine class creates same size grid (vector matrix). For each slot there will be an object Square, which then contains something else, like a Road or a Building (or nothing). Filereader-class will have functions that can read the file and create all needed structures.

Movement of cars will be simulated like it is linear. Cars will also have a variable of the travelled distance inside Square x. Placement of cars can be visualized as a dot in the back end of a car. Cars will know their own size, so they know how much room to leave between itself and the next car in front of it. This makes it easier to implement different lengths for cars (busses for example).

All of the basic functionality is to be implemented with the standard library.

Sketch of graphical interface



This is the harsh sketch of the graphical interface we plan to implement.

Additional elements we consider adding (if there is enough time):

- Parking slots
- Ability to zoom
- Ability to click intersection and improve its capacity
- Ability to click a road and change the speed limit
- Ability to select buildings and control the residents of those buildings (send all of the to shop/ change their working hours etc.)

Different intersections would have of course different visualizations.

Project schedule

- wk 45: project planning
- wk 46:
  - implement basic forms of the classes and get them to interact with each other
- wk 47: mid-term meeting
  - continue with basic operation
  - implement basic analyzation to ease with development
  - load sample city from file
- wk 48:
  - check if we have the basic functionality and test if it works or not
  - make people act more randomly
  - how to choose the best route?
- wk 49:
  - start documentation
  - perform testing for documentation and presentation
  - fix issues found while testing
- wk 50: final demonstration
  - last tweaks and fixes
  - finish documentation

Distribution of roles in the group

We were unable to divide the project into detached parts, as there are no clearly independent parts, and there is too much uncertainty about how we will implement the project. Therefore we need to actively communicate what work is needed when and who should work on what.