Tommi Gröhn, Kaisa Kärkkäinen, Akseli Oinaanoja, Stas Tatun

Object oriented programming with C++

Traffic simulator - project plan

Scope

Our project topic is Traffic Simulator. The project will include basic features listed in the topic assignment as well as some additional ones. The first goal is to get the traffic simulator to work implementing only the basic features and then later on, keep adding new features to the project.

The essential idea is to create different objects: roads, destinations, city and drivers. Drivers have planned their day before they leave their home and they have also decided their time spent in their destinations. There is going to be some kind of a center where there are going to be a lot of workplaces and also areas full of homes. Every single person is going to move with a car in this simulation and there are no pedestrians because they would not add a lot of value to the simulation.

One of our biggest goals is to create a coherent simulation implementing randomness of traffic with well chosen functions and well-structured code. We are going to have many separate Poisson processes that represent people leaving their homes. There is going to be implemented randomness related to the destinations of drivers and the time spent in the destinations. Most likely, we have to generate the profile of the buildings ourselves based on the road structure of the data. Because the simulation for every single day is different, data analysis is going to use in some manner Monte-Carlo simulation for a longer period than just one day.

We are going to create a city file based on real world map with simple attributes. We are going to create nodes (crossroads, homes, etc.) which are linked with roads which form a simple grid. We are going to form areas with different profiles: there is going to be some kind of city center, as well there are going to be residential areas as well. By creating the data ourselves, we are capable of concentrating on the simulation itself without using too much time on cleaning the data.

The graphical user interface will be a simple visualization of the simulation, i.e. a map showing traffic. Informativeness will be prioritized over aesthetics. As the team has little experience in graphics libraries, we're fairly cautious about GUI features; menus, real time manipulation, etc. will be decided later on.

Structure of the software

We are going to separate our project on multiple different files: *main.cpp, city.hpp, city.cpp, node.hpp, node.cpp, road.hpp, road.cpp, driver.hpp, driver.cpp*.

The file *main.cpp* first reads the city data and creates a graph of nodes and edges. Edges represent the roads and nodes some kind of destinations for example someone's home, grocery store but also crossroads.

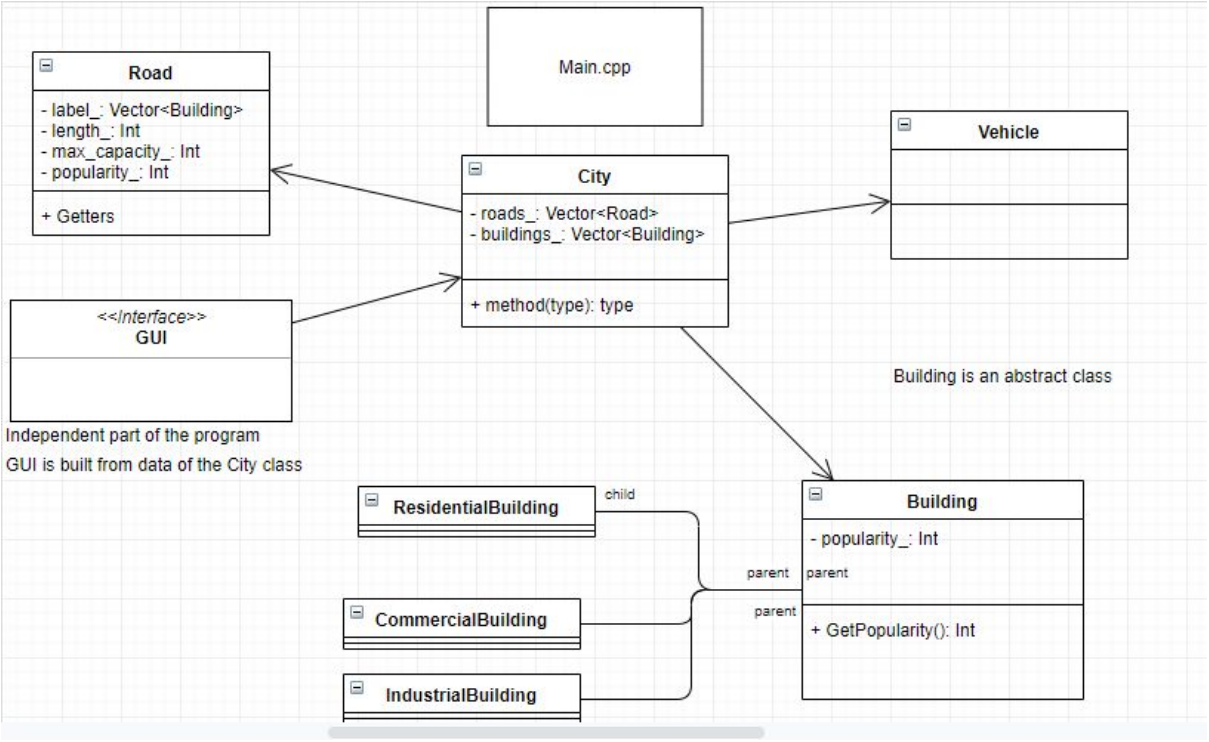
In other words, a lot of objects are created at the beginning of this program.
The objects are defined on files *node.hpp, node.cpp, road.hpp, road.cpp.* Nodes and roads are going to have some kind of labels that describe their nature. For example edges are going to have a label what is the total capacity of the road and what kind of buildings are near the road. Nodes are going to have at least a label that describes what kind of node it is: a crossroad, house etc. This property could be implemented well by using inheritance.

Some of the data of nodes and roads changes during the simulation and some does not. The file *city.cpp* mostly holds the information of the changing data and the previously described unchangeable data is going to be stored in files *node.hpp, node.cpp, road.hpp, road.cpp*. The file *city.cpp* holds for example the information on how many cars are on different roads at the current times. It also collects the data for the data analysis.

Essential way of creating randomness in this work is that there is a big amount of independent Poisson processes which describes people who are leaving some building. For example, in the morning time people will wake up and leave their home quite likely. This can be implemented using Poisson distribution using quite big value of lambda which is the parameter of the Poisson distribution. Because there are many Poisson processes happening at the same time, the file *city.cpp* has to know the summed value of all independent Poisson processes. Based on this summed value we can easily estimate the next time point when someone is going to leave some building. Then based on values of different lambdas we are going to randomly choose, where to is a new person with a car going to be generated.

If there is a lot of traffic on a road, the amount of time a vehicle spends on the road increases. The file *city.cpp* will modify the parameter of a specific road related to this property.

Every time a new person is generated, in the files *driver.hpp, driver.cpp* there is going to be defined the profile of a driver. A driver has a starting point which is also the driver's endpoint. A driver must have at least one destination, for example work or a shopping mall. The time spent in the destination is defined beforehand but the time spent in traffic naturally depends on the traffic. After the driver comes back home, the driver object is going to be released from the memory.



External libraries

The GUI will be done with Qt. https://doc.qt.io/. Standard library is sufficient for everything else.

Responsibilities in the group and schedule

Tommi is the project manager. According to our preliminary schedule, we have had two group meetings before submitting the project plan, and weekly meetings will continue throughout the project.

By mid-November we are going to have the rudimentary classes defined and implemented in addition to the parsing of the data and, of course we will have decided the format of the data to be read to create the city. For the late-November we will have time to implement the missing functionality to all of the classes and start working with the graphical user interface (GUI). Before project deadline, we will have time to finish the GUI and implement additional functionality to the classes.

We have divided our group in two pairs. Stas and Tommi belong to the first group, and Kaisa and Akseli to the second one. Basically, the Stas and Tommi will do the first version of the simulation. The GUI will be implemented by Akseli and Kaisa. Responsibilities are fluid.

.