# Review form for project traffic-simulator-2019-2

**Name of project to be reviewed:
- Traffic Simulator 2019 - Group 2

**Names of reviewers:
- Jonathan Leinola - 477329
- Niko Uski - 51961T
- Artur Gynter - 569499
- Hien Cao - 716718

Provide short comments (2-4 sentences) for each item below.

## 1. Overall design and functionality (0-6p)

*** 1.1: The implementation corresponds to the selected topic and scope. The extent of project is large enough to accommodate work for everyone (2p)**

The project corresponds well to the selected topic. They took an analytical approach for this project and their plan was to simulate traffic on real roads. They succeeded well in their main goal: to simulate traffic and they also added couples additional features as well as GUI to visualize the simulation itself. (2p)

*** 1.2: The software structure is appropriate, clear and well documented. e.g. class structure is justified, inheritance used where appropriate, information hiding is implemented as appropriate. (2p)**

Information hiding is implemented as appropriate. No classes inherit in the project but this might be due to the fact that there is no need to inherit anything because all classes are different types of data structures. Also, there is not much duplicate code in the project and this makes the program more scalable. (2p)

*** 1.3: Use of external libraries is justified and well documented. (2p)**

They used QT for GUI and drawing and rapidxml to parse xml data from Open Street Map files. Both are mentioned in the documentation. The version of QT should have been mentioned as the compilation process depends on QT5. (1p)

## 2. Working practices (0-6p)

*** 2.1: Git is used appropriately (e.g., commits are logical and frequent enough, commit logs are descriptive). (2 p)**

The project has 106 commits which are the mix of Finnish and English. Most of the commits are descriptive, but some of them are very short and do not describe well. Commit logs are not so frequent enough as the three-fourths of them have come for the last two weeks. (1p)

*** 2.2: Work is distributed and organised well. Everyone contributes to the project and has a relevant role that matches his/her skills. The distribution of roles is described well enough. (2p)**
The works were divided equally. Everyone has contributed to the project. The tasks of each member are described in the project documentation. However, the details of the work logs are missing as only the total number of hours is mentioned. (1p)

*** 2.3: Quality assurance is appropriate. Implementation is tested comprehensively and those testing principles are well documented. (2p)**
In the project documentation, it is indicated that the testing was handled manually. There is no mention of the test was documented. Some of the commits describe bugs and bugs fixed but no description about what happens error and what was fixed. (1p)

## 3. Implementation aspects (0-8p)

*** 3.1: Building the software is easy and well documented. CMake or such tool is highly recommended. (2p)**

Building software is documented but we were not able to build the project successfully. Also, the documentation does not mention the version of additional libraries so we couldn’t figure out how to build it. (0p)

*** 3.2: Memory management is robust, well-organised and coherent. E.g., smart pointers are used where appropriate or RO3/5 is followed. The memory management practices should be documented. (2p)**

No smart pointers used. There are memory leaks because objects are allocated with “new” but the memory was not freed, one example: creating Car object with new but it never gets freed. Also no documentation on ownership of manually allocated objects. Data containers are passed by copy and const not used for functions :(. (1p)

*** 3.3: C++ standard library is used where appropriate. For example, containers are used instead of own solutions where it makes sense. (2 p)**
Vector, map, and list are used to store different types of data structures. (2p)

*** 3.4: Implementation works robustly also in exceptional situations. E.g., functions can survive invalid inputs and exception handling is used where appropriate. (2p)**

As we cannot compile the software, there is no test implemented. (0p)

## 4. Project extensiveness (0-10p)

*** Project contains features beyond the minimal requirements: Most of the projects list additional features which can be implemented for more points. Teams can also suggest their own custom features, though they have to be in the scope of the project and approved by the course assistant who is overseeing the project. (0-10p)**

Basic features:
All of the basic features are implemented.

Additional features:
Gui and passenger profiles.

Advanced/custom features:
Able to create a simulation from OpenStreetMap (.osm) :)
Djikstra algorithm

Additional comments:

+ Very interesting simulations can be achieved with this project and we liked that it allowed to simulate real roads and buildings. Also visualization of the traffic is done well and their analytic approach was on point!
- Code is inconsistent and this makes it hard to read. This is because indentation and brackets are inconsistent, there are a lot of variables and naming conventions are not followed consistently.