# ELEC-A7151 Object oriented programming with C++
# Software project - Traffic Simulator
# Project plan

Hatanpää Väinö, Herold Adam, Kelekar Aditya, Selvarasa Gowshigan, (Köylijärvi Otto)

October 30, 2020

## 1 Introduction and Motivation

For our software project we have chosen to create a traffic simulator in C++. A traffic simulator is a useful tool, which can be used in many ways such as to analyze the traffic situation in a given part of a city, evaluate and optimize the impact of future roads and get a deeper understanding of the current traffic problems. Such a simulator can be used by city planners, civil and road engineers or public transport experts.

## 2 Scope of the project

The objective of this project is to simulate a typical traffic scenario with a 2D map and mobile entities such as cars and persons, and fixed ones such as buildings and roads. The traffic simulator will work with static inputs. Therefore, changing the inputs while running the simulator will not be possible. The static components of the simulator such as building, lanes, bikes will remain same during the simulation. These components are created based on the user input in the beginning of the program. The simulator will run upto a fixed time provided by the user (user provides the input in the main program). We use SFML library to visualize maps and other objects.

In the simulation, the number of cars that can appear on the roads has an upper limit. The passengers take actions according to their schedules. An intersection will connect three or more roads. The map can be saved and loaded for future use. The simulator also has an analytic tool to analyze a specific road and plot the histogram of cars on that road for a specific time. These results can be exported.

# 3 High-level structure

## 3.1 Main modules

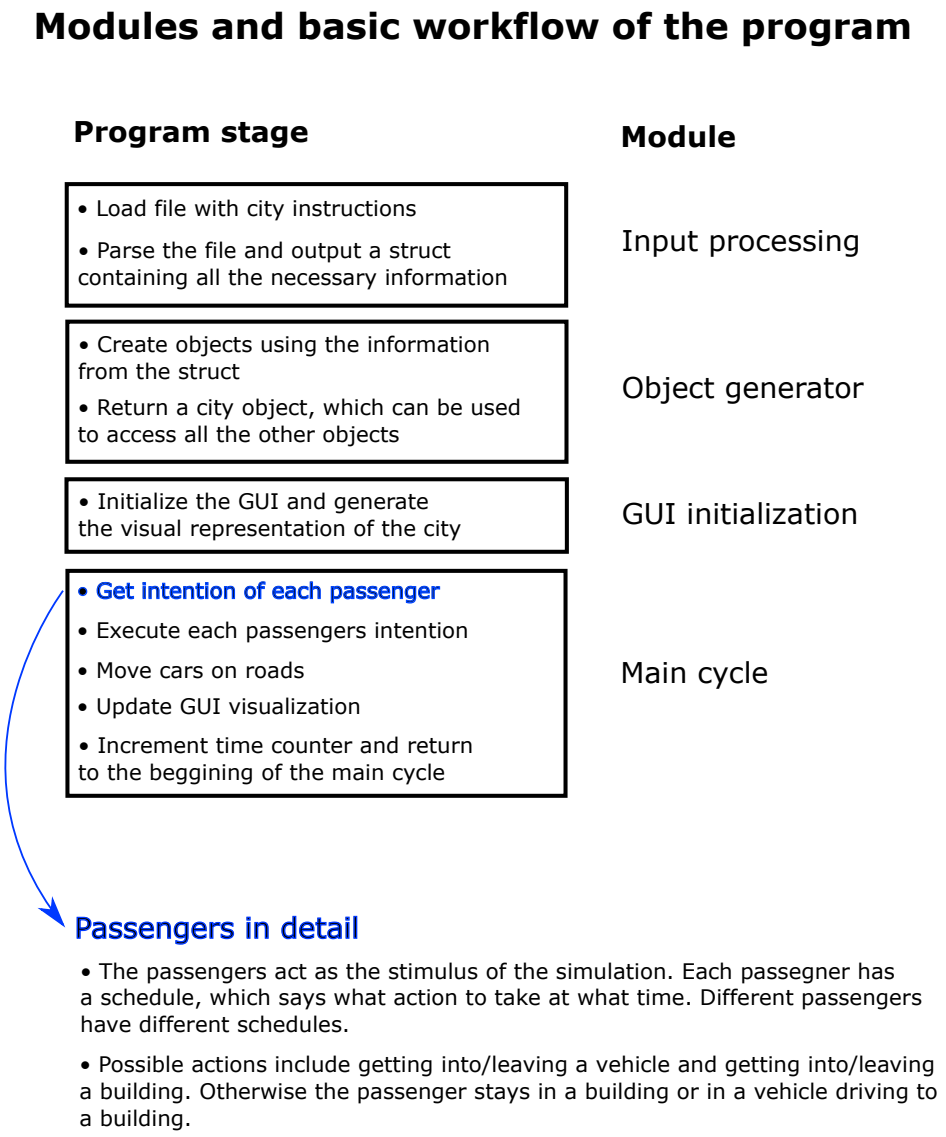


Figure 1: A rough draft of the program's workflow separated into different modules.

### 3.2 Main classes

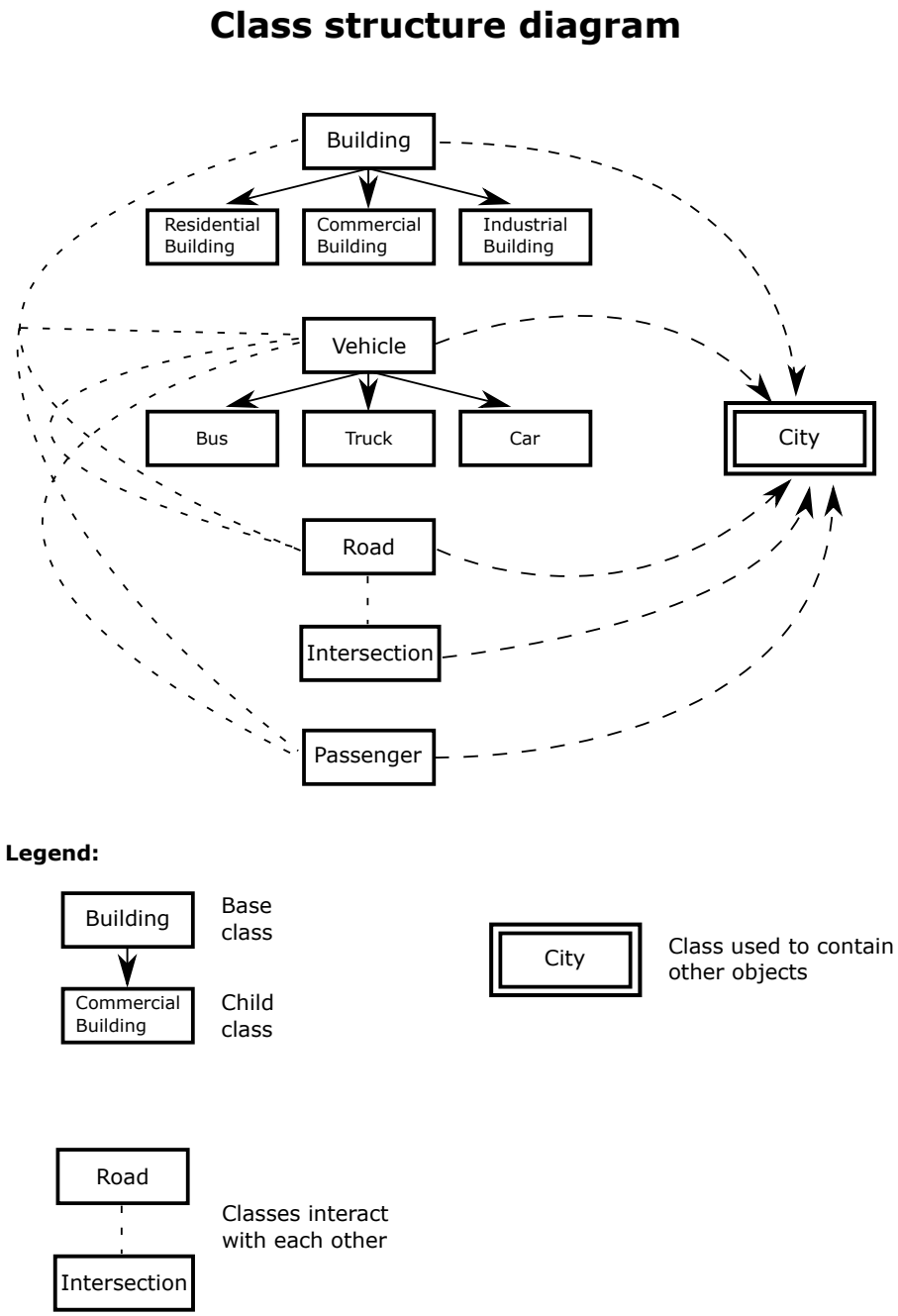


Figure 2: Diagram illustrating the basic structure of the main classes.

- **Building**: An abstract class, from which we derive subsequent classes for different types of buildings (residential, commerical, etc.).
- **Vehicle**: An abstract class, from which we derive classes for vehicles such as a car, a truck, etc.
- **Road**: Roads always connect two points (buildings or intersections). They have a number of lanes and they fit a finite amount of cars, based on their size.
- **Intersection**: Connects two or more roads. May also include traffic lights or traffic signs.
- **Passenger**: Passengers act accordingly to their schedule by taking actions at appropriate times.
- **City**: A container class used to store other objects in one place.

# 4 Use of external libraries

- **Gtest** Gtest is a unit testing library for the for c++ programming language.
- **SFML**: Simple and Fast Multimedia Library, is a portable API written in C++. It is composed of five modules: system, window, graphics, audio and network.
- **matplotlib-cpp**: Visualizations for anlytics
- **nlohmann/json**: Used for parsing the input file.

# 5 Division of work between the group

Here we list the parts of the project each member will primarily focus on. As the difficulty of different parts of the project is not yet clear, it is presumed that the workload will be balanced between all members as needed.

- **Hatanpää Väinö**: Analysis tool.
- **Herold Adam**: Classes and city loading from file.
- **Kelekar Aditya**: GUI to visualize roads and cars using SFML library
- **Selvarasa Gowshigan**: Car class, main, GUI to visualize roads and cars using SFML library
- (**Köylijärvi Otto**: Does not seem to be participating)

# 6 Planned schedule

| **Mile stones** | **Date** |
|---|---|
| Submitting Plan | 30.10.2020 |
| Some components or classes working | 06.11.2020 |
| Complete classes excluding the GUI part | 13.11.2020 |
| Traffic simulator with properly working GUI | 20.11.2020 |
| Complete required basic features and improve on features completed previously | 27.11.2020 |
| Try to add advanced features,<br>work on the source code quality such as optimization of the code to avoid memory leak<br>Prepare the documentation | 11.12.2020 |