Zurich University
of Applied Sciences

School of
Engineering

INFPROG2 P05 – Individual Application Project

This project runs over 4 net weeks (nominally SW9-SW14 minus two) and thus yields 4 points. It can be solved in teams of 2-3 students or individually if deemed feasible by the lecturer. In this final programming project, you have the ability to bring in your individual preferences: Out of 5 possible subtasks, choose the one (per student) that you like most! Add topping according to your own liking. In agreement with the lecturer, another (effort-wise equivalent) topic can also be arranged.

1.1 Basic Application

Remember the «game of life» by mathematician J.H.Conway (†2020)? With your current programming skills, you are able to build such complex simulations. Instead of simple cells and reproduction rules, your application will simulate a city with richer content and more sophisticated rules. (Perhaps you already know the related «sim city» game concept.)

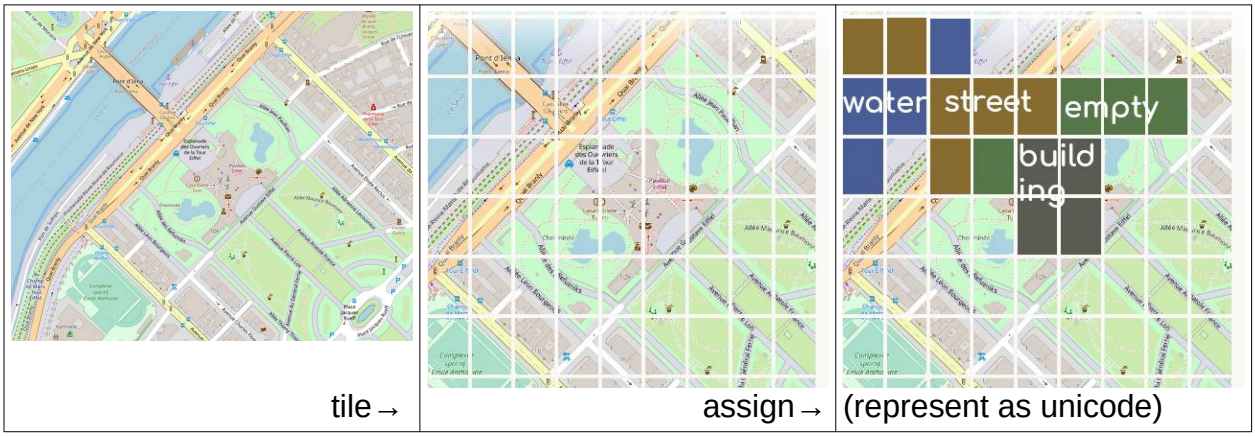
Use the prepared basic application skeleton '`P05_pycity_basic.py`' and adjust it to your liking. The basic application contains a surface of MxN (e.g. 50x30) fields. Each field contains a typical city structure: empty land, water, residential house, business house, empty street, street with car, land with person. When the application runs, the field is updated once per update interval T (e.g. 1s) and displayed (with appropriate Unicode symbols) on the terminal. Be creative in using other symbols - use a unicode symbold finder (e.g. `http://shapecatcher.com/`) to find your symbols quickly.

Example screen of the prepared basic application skeleton:

The basic requirements for both individual students and teams to advance the basic application and make it usable for individual simulation are:

- give the application user the choice of realistic pseudo-random generation of surfaces vs. loading them from pre-defined scenarios stored in files
  - addition of a command-line parameter through the `argparse` module
  - improve the pseudo-random generation: add some logic and useful constraints (e.g. considering long vertical, horizontal & diagonal streets or adjacent street segments) so that not everything is completely random

  - loading: ability to load a surface given as scenario textfile with the exact Unicode symbol representation
- create at least one 30x30 example scenario file with simulation of your town or village; you can use a tiling map tool or a generic image rasteriser to get a good representation of MxN fields based on which you can assign the city structures (e.g. `http://posterizer.online/rasterbator/`)
  - start with a 10x10 scenario file and leave the extension to the end of the project to make progress quickly



tile → assign → (represent as unicode)

Further requirements for teams are:

- sensible colouring of the map to make it easier to understand it
- one scenario file of at least 30x30 fields per student
- one subtask per student

The solutions to the following subtasks need to be of high quality, and they need to be obvious from the demonstration of the solution. Do not underestimate the time needed to fine-tune graphical representation, timing/movement visualisation, logic and so forth. Please check back with your lecturer if your choice of subtask(s) is valid to ensure balance and variety of solutions.

1.2 Subtask A: Moving cars

A car has a preferred direction where to go. In each round, the car is shifted by one field (i.e. street with car becomes empty street, adjacent empty street becomes street with car). The empty street closest to the preferred direction is chosen. In case the car is stuck, the preferred direction changes. This way, a relatively smooth movement of the car along a street can be seen.

1.3 Subtask B: Growing population

Each residential house has a population and a capacity. In each round, the population grows by a factor (e.g. 1.1). When the population spills over the capacity, a person “moves out”, i.e. the population is reduced and another residential house is created nearby on empty land. Occasionally, with 10% probability, a person dies. When the population of a house converges towards zero, the house disappears. You could furthermore use small house and big house symbols to visualise the growth better.

1.4 Subtask C: Dynamic economy

Each business house generates income according to a specific productivity and the ability to sell to other nearby businesses. The more a business gets revenue, the more it will invest into other businesses, with a radius depending on the revenue (e.g. 3x3 or 5x5 neighbours). As the business grows, this is indicated in the visualisation, e.g. by colour level. Occasionally, a business goes bankrupt. In this case, the surrounding businesses will take a 50% hit in their productivity.

1.5 Subtask D: Public safety

Two new categories, police and firefighter houses, are introduced. Each such house protects residents in a 5x5 neighbourhood. The absence of one them however leads to a 10% likelihood of one death per resident house; the absence of both even to 20%. Notice how residents far away from public safety facilities will soon disappear.

1.6 Subtask E: Virus contamination

Persons move so that “land with person” tiles and “empty land” tiles are updated according to a logical movement pattern. Persons have a remaining time to live which by default is a high value, and an infection state that can flip randomly with very low probability. Any infected person infects adjacent other persons, leading to lower time to live. Once the time is up, the person disappears and the land becomes empty. Use colours to highlight the different health states of a person.