Final Project: Part 2
City Optimization
BK7084 Computational Simulations

City optimization

In the second part of the final project, you are going to implement an optimization algorithm to make your city a city that people would love to live in.

In total, your city consists of 64 plots of land (8x8) and should be populated with the following buildings:

- 6 Skyscrapers
- 10 High rises
- 18 Offices
- 26 Houses (Given)
- 4 Parks (Given)

Initializing the grid

As of yet all the plots in your grid are only populated by a few buildings (two of each). Your first task will be to populate the grid so that the distribution above is respected, and you have a total of 64 buildings placed in the city. In the file *city.py* you will find a function called *reset()*. In this function you have access to the grid that you must fill. The grid consists of 8x8 plots of land, each with a specific *PlotType*.

The following plot types exist:

- `EMPTY`
- `PARK`
- `HOUSE`
- `OFFICE`
- `HIGHRISE`
- `SKYSCRAPER`

City has the following functions that you can use to place your buildings:

- **clear()**
  Fills the city with empty plots.
- **row, col**
  Returns the number of plots in each direction of the grid (8 in this case).
- **get_plot_type(i, j)**
  Returns the plot type at position (i,j) on the grid.
- **set_plot_type(i, j, type)**
  Sets the plot type at given position (i,j) to be of **type** instance.
- **grid.cell_position(i, j)**
  Returns the position in 3D space of the plot center.

Optimization

Your city is now populated, but it is probably not very pleasant to live in. As the sun rises and sets throughout the day the skyscrapers and high rises obscure many of the parks and houses from seeing the sunlight. Some houses might be very far from their office, or any recreational park. Skyscrapers might be right next to each other, causing one side to be completely dark.

In *optimization.py* there's a function called *step()*. This function gets called every time you press the 'Step' button. You can also make the function run a number of times using a 'Start' and 'Stop' or in the background (without showing it on the screen) with 'Optimize offline'. In this function you should iteratively improve the initial layout that you generated to comply with the rules you designed.

A naive way to do this is to simply generate another random layout of plots and see if it is better. Running this for a while will get you better and better layouts of your city, but it will take a long time.

There are many better ways to optimize the city in a more direct manner. Try to come up with your own clever solution for improving this naive optimization. In your demo presentation you will show the iterative optimization process in real-time.

Of course, you need some way to define what a better city layout is. In the following **example** we try to raise the price value of all houses as much as possible. The price of a house increases according to the following criteria:

- Houses should be as close as possible to offices
- Houses should be as close as possible to parks
- Houses and parks should receive as much light as possible during a day
- No two skyscrapers or high rises can be next to each other (1 plot in between diagonally, horizontally and vertically).

You should come up with your own set of rules for your city. Implement this in the *score()* function in *optimizer.py*. You have access to functions that compute the contribution of light from a given light position for either the ground of a plot or the building on the plot at cell (i, j). Look at *compute_light_of_plot_day(i, j)*, which sums the light contribution over 12 light positions. In *score()*, we use this function to compute the light score for the full grid.

There's quite a bit of computation happening in the background. If you find your program is too slow, try reducing the number of lights in *compute_light_of_plot_day(i, j)*, compute the light for only one plot, or reduce the size of your city. **You can always comment out the light computation while your testing other rules in your city.**

Presentation and Report

At the end of the project you (or your group) will submit a report explaining how you designed your rules and implemented the optimization. In the report you should describe what you have done in both the building generation and optimization parts of the final project, and how you have done it. What challenges did you face during the optimization and how did you overcome them?

There will also be a small demo session when each group can briefly present their solution and give a live demo of the optimization algorithm. Here, you don't have to go over all details but just highlight the most important points and original ideas. Each presentations including the live demo should take between 5-7 minutes, please do not go over time.

Minimum requirements

What follows are the minimum requirements for a passing grade.

Building generation

At least your building should satisfy or out-do the following minimum requirements:

- Building made out of simple shapes (rectangles, triangles)
- Building is textured decently
- Building includes simple decorations (windows, doors)
- Buildings fit inside a grid cell.

You will get extra points for the following:

- Using randomness to generate a variety of buildings
- Non-trivial shapes included in your building
- Properly textured non-trivial shapes
- Interesting extra decorations (e.g. cornices, quoins, balconies, be creative!)

City optimization

At least your city optimization should satisfy the following minimum requirements:

- Computation of score for set of rules to be optimized
- Optimization implemented in a naive manner (e.g. random layout on every iteration or randomly swapping two cell types)

You will get extra points for the following:

- Optimizing city layout in a more clever way