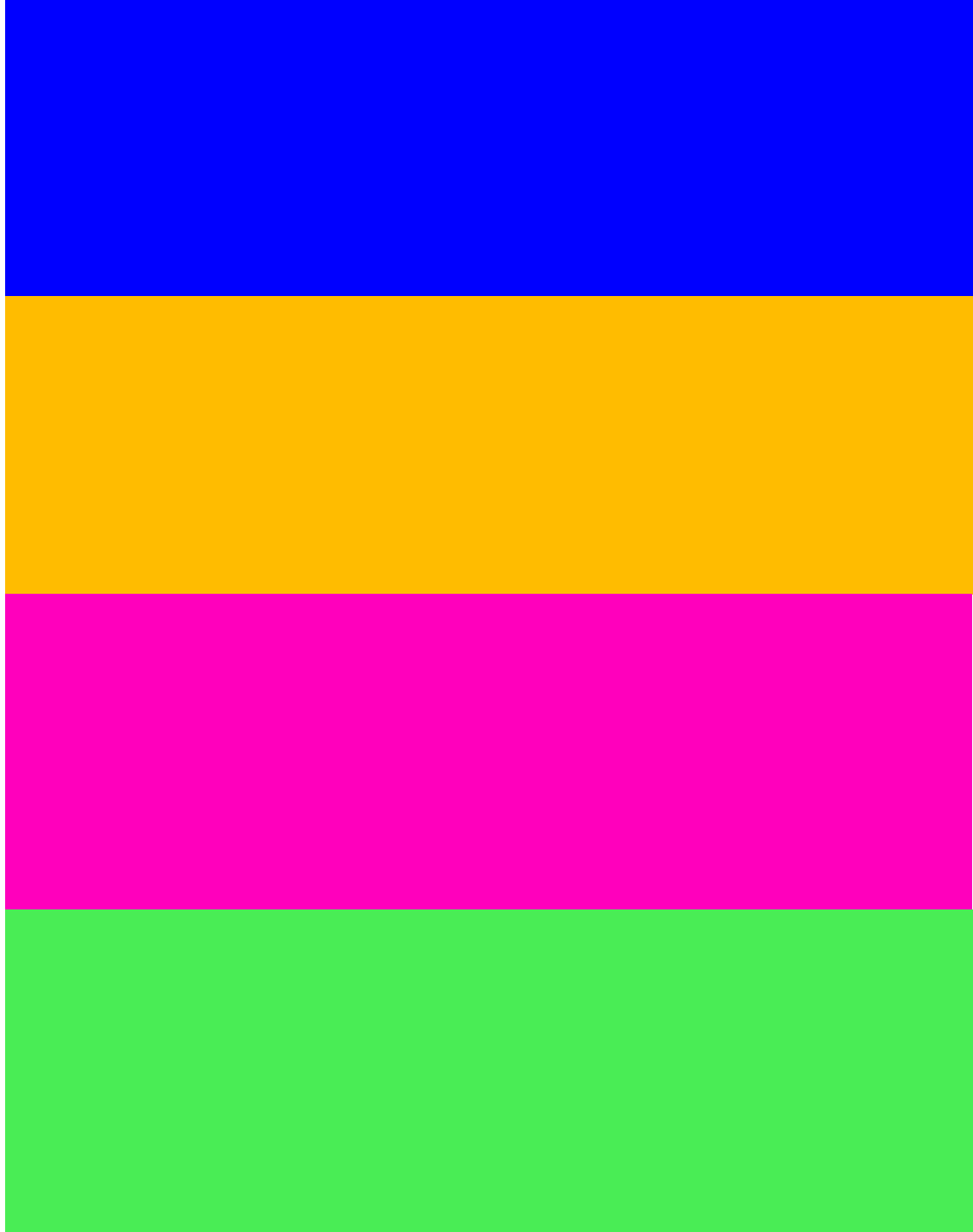
BK7084
Final Project

Lucas de Vaan

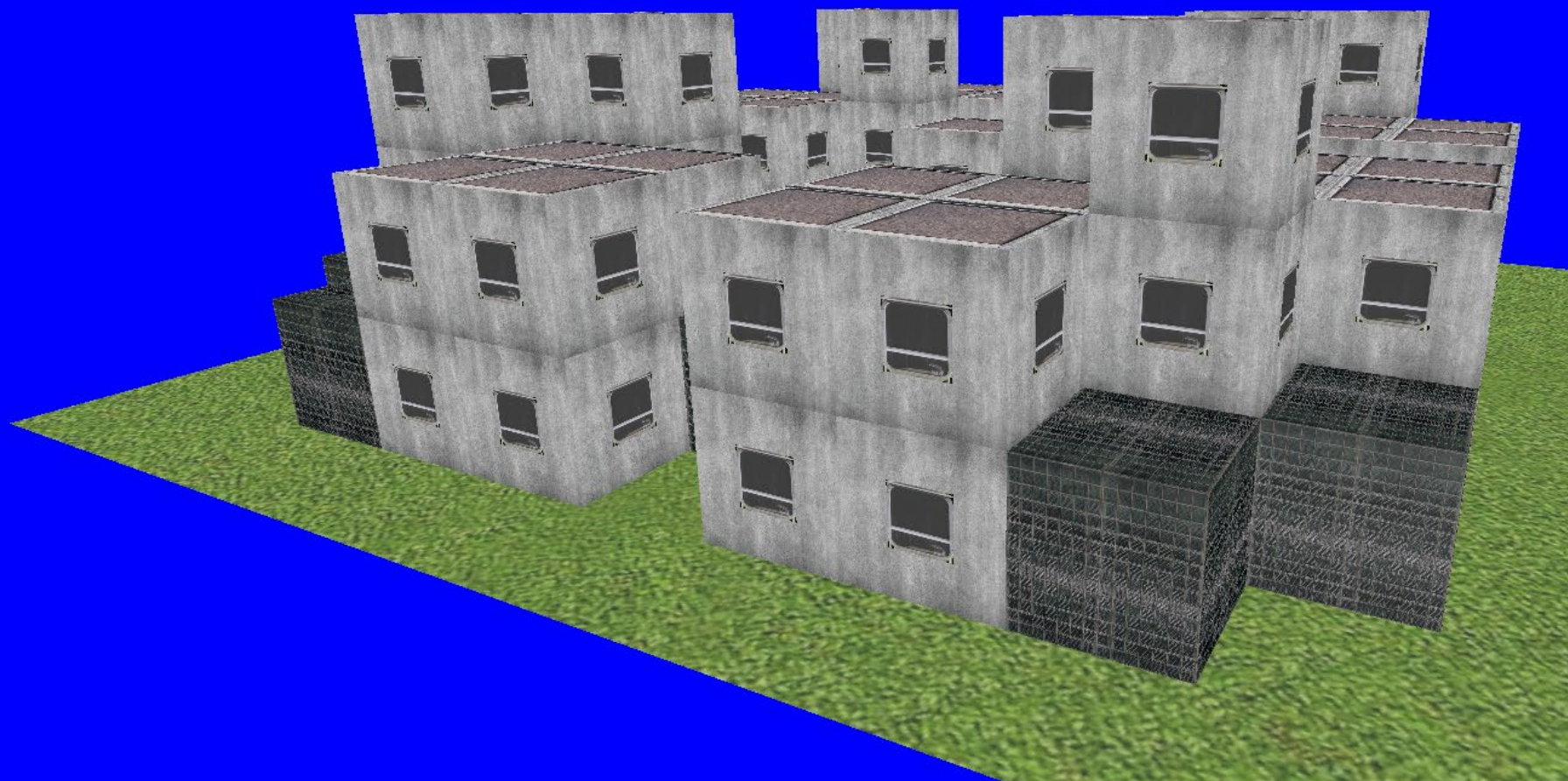
Imke den Boogert



Introduction

- Buildings
  - Office Building
  - High Rise Building
  - Skyscraper
- Approach to designing
- Inspiration
- Optimization of the city

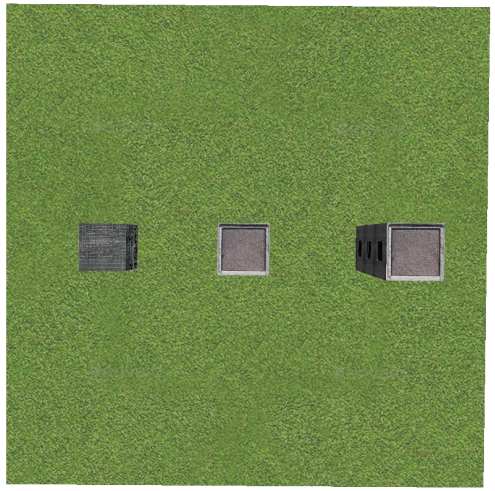
Office Building



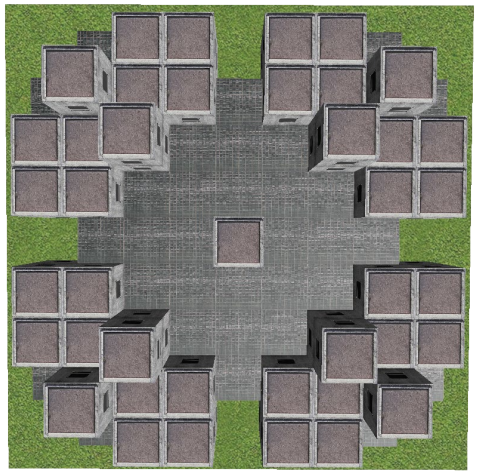
Office Building

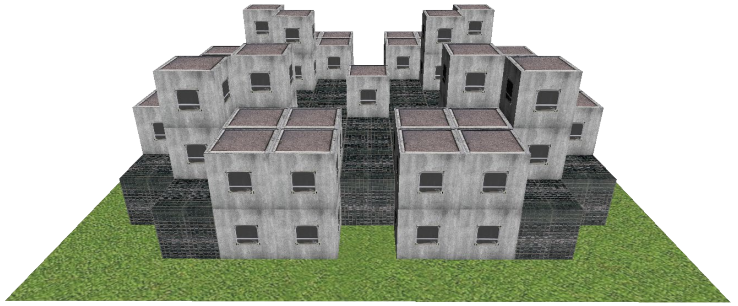
Approach

```
grid_config = [
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '1', '0', '0', '2', '0', '0', '3', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
]
office = Office(app, 3, grid_config=grid_config)
office.building.set_transform(Mat4.from_translation(Vec3(0, 0, 0)))
```



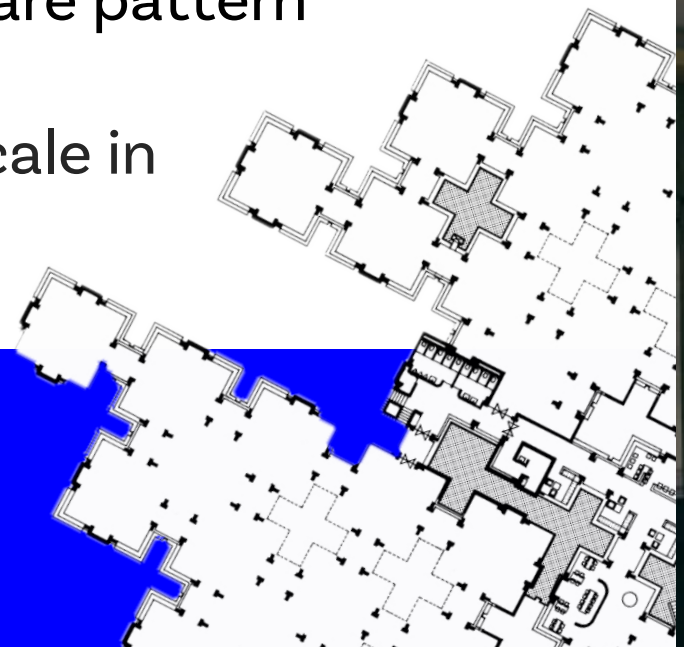
```
grid_config = [
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
    ['0', '0', '1', '2', '2', '0', '2', '2', '1', '0', '0'],
    ['0', '1', '3', '2', '2', '1', '2', '2', '3', '1', '0'],
    ['0', '2', '2', '3', '1', '1', '1', '3', '2', '2', '0'],
    ['0', '2', '2', '1', '1', '1', '1', '1', '2', '2', '0'],
    ['0', '0', '1', '1', '1', '2', '1', '1', '1', '0', '0'],
    ['0', '2', '2', '1', '1', '1', '1', '1', '2', '2', '0'],
    ['0', '2', '2', '3', '1', '1', '1', '3', '2', '2', '0'],
    ['0', '1', '3', '2', '2', '1', '2', '2', '3', '1', '0'],
    ['0', '0', '1', '2', '2', '0', '2', '2', '1', '0', '0'],
    ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
]
office = Office(app, 3, grid_config=grid_config)
office.building.set_transform(Mat4.from_translation(Vec3(0, 0, 0)))
```





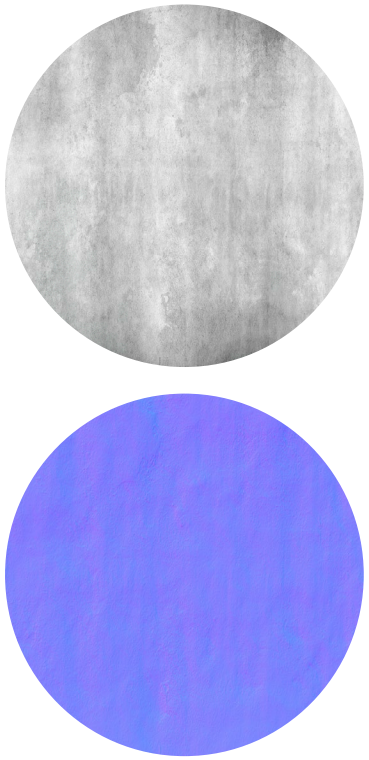
Office Building
Reference

- Inspired by Centraal Beheergebouw - Herman Hertzberger (1970)
- Dutch structralism
- Repeating square pattern
- Uses human scale in design

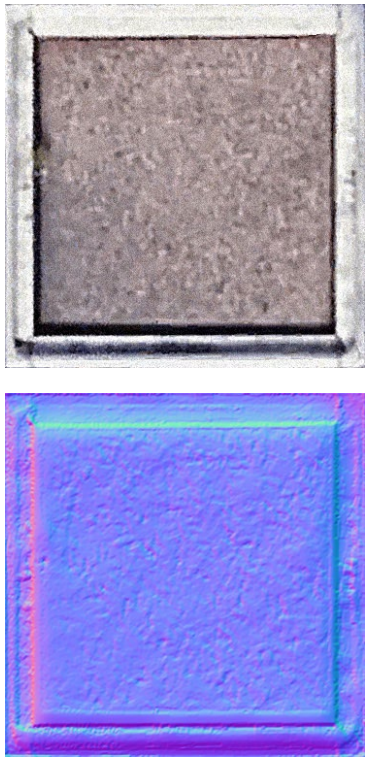


Office Building

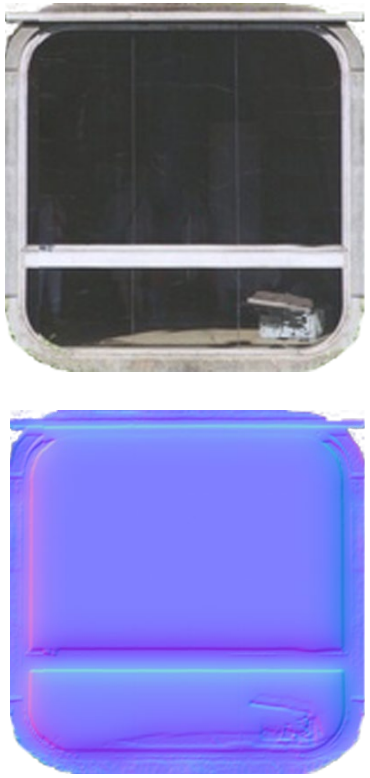
Textures



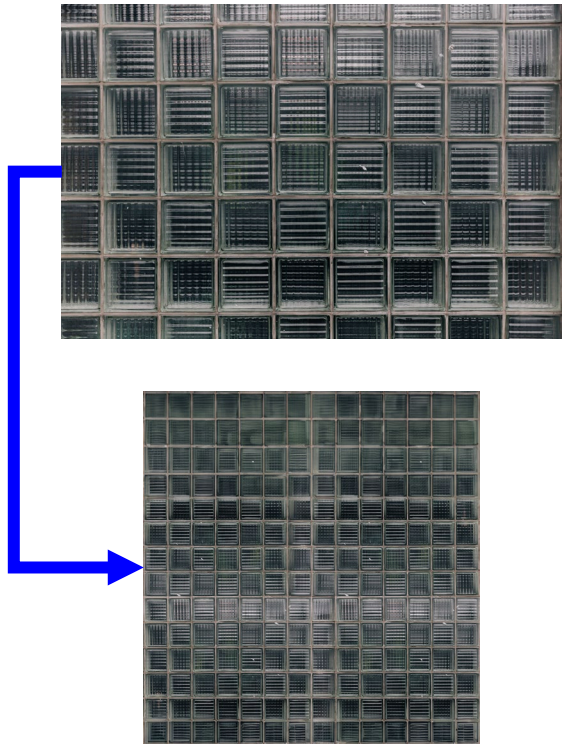
Concrete Wall



Concrete Roof



Window



Glass Wall

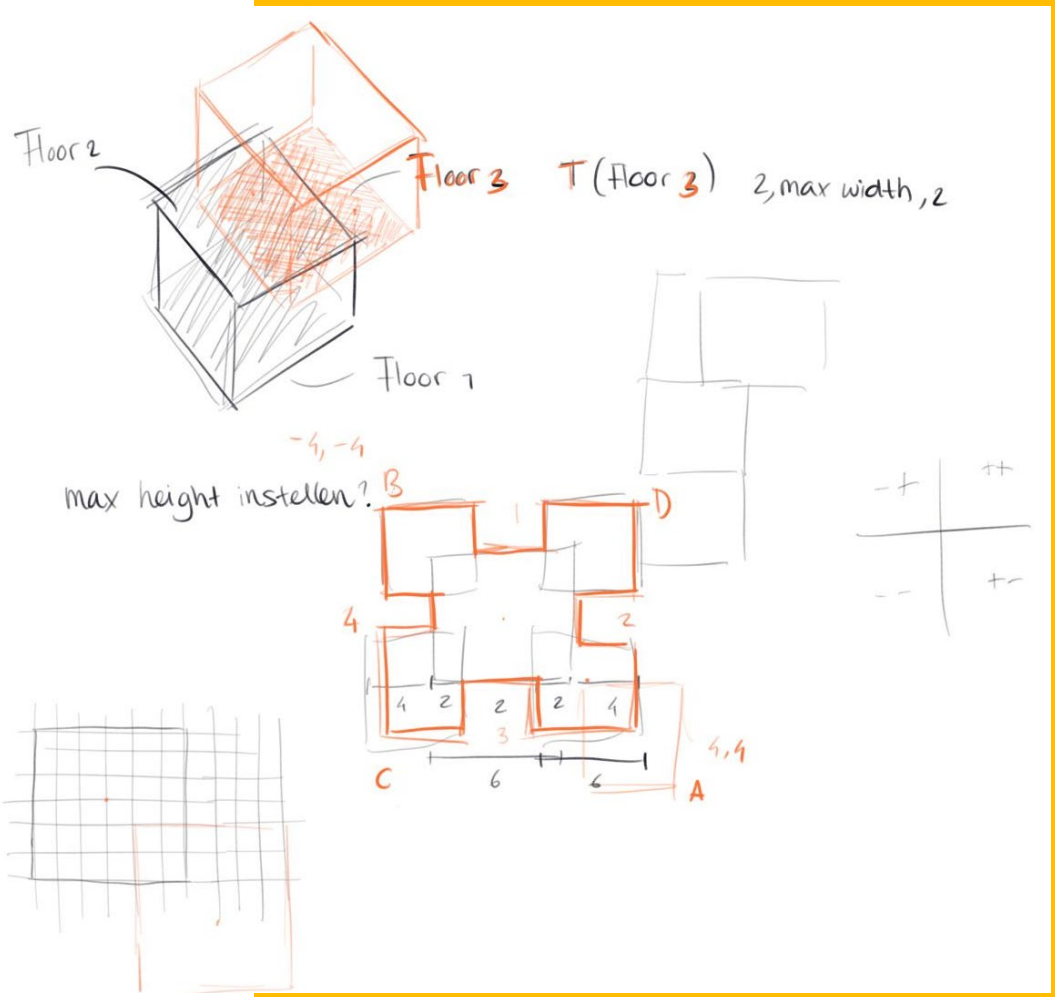
Highrise

High Rise
Approach

- Tower 0, A1, A2, B1, B2, C1, C2, D2, D3
- Add walls to floors

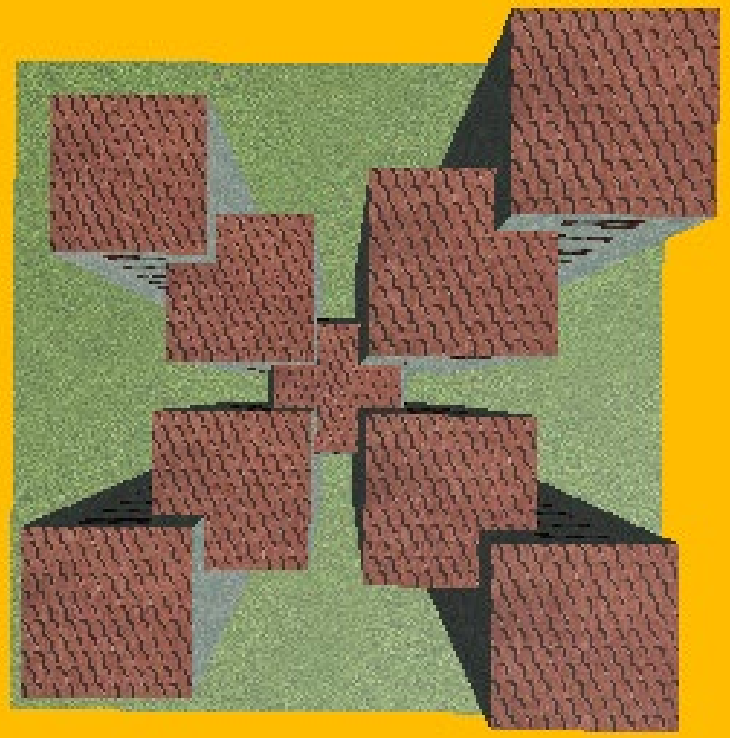
- Assign WindowWall to wall 1 and wall 3

```
# toren A
    #toren A1

        floor2 = app.add_mesh(BasicFloor(max_width, max_width), parent=floor1)
        floor2.set_transform(Mat4.from_translation(Vec3(2, max_width * 3, 2)))
        floor2.set_visible(True)


    #toren A2
        floor3 = app.add_mesh(BasicFloor(max_width, max_width), parent=floor2)
        floor3.set_transform(Mat4.from_translation(Vec3(2, max_width, 2)))
        floor3.set_visible(True)


# toren C
    #toren C1
        floor2c = app.add_mesh(BasicFloor(max_width, max_width), parent=floor1)
        floor2c.set_transform(Mat4.from_translation(Vec3(-2, max_width * 2, 2)))
        floor2c.set_visible(True)


    #toren C2
        floor3c = app.add_mesh(BasicFloor(max_width, max_width), parent=floor2c)
        floor3c.set_transform(Mat4.from_translation(Vec3(-2, max_width, 2)))
        floor3c.set_visible(True)
```

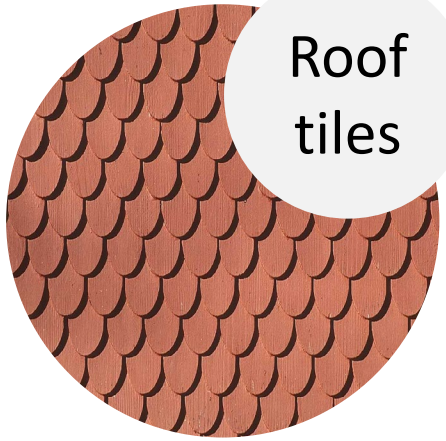


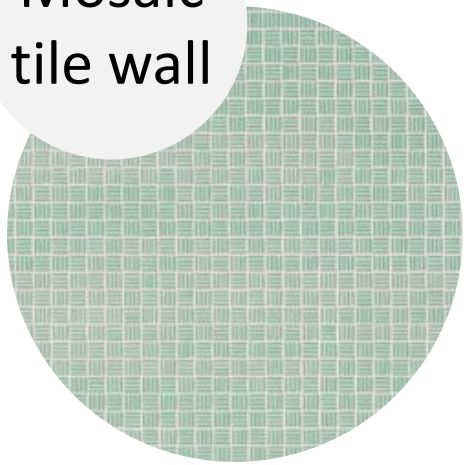
High Rise
Structure

- Symmetry
- Dynamic structure

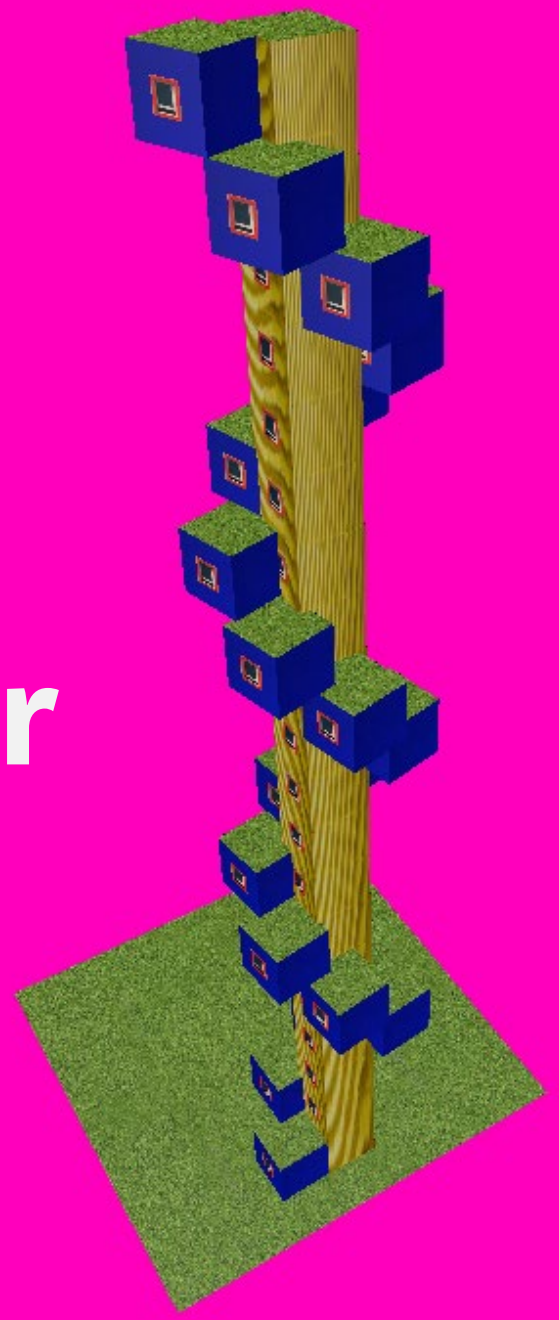


Textures



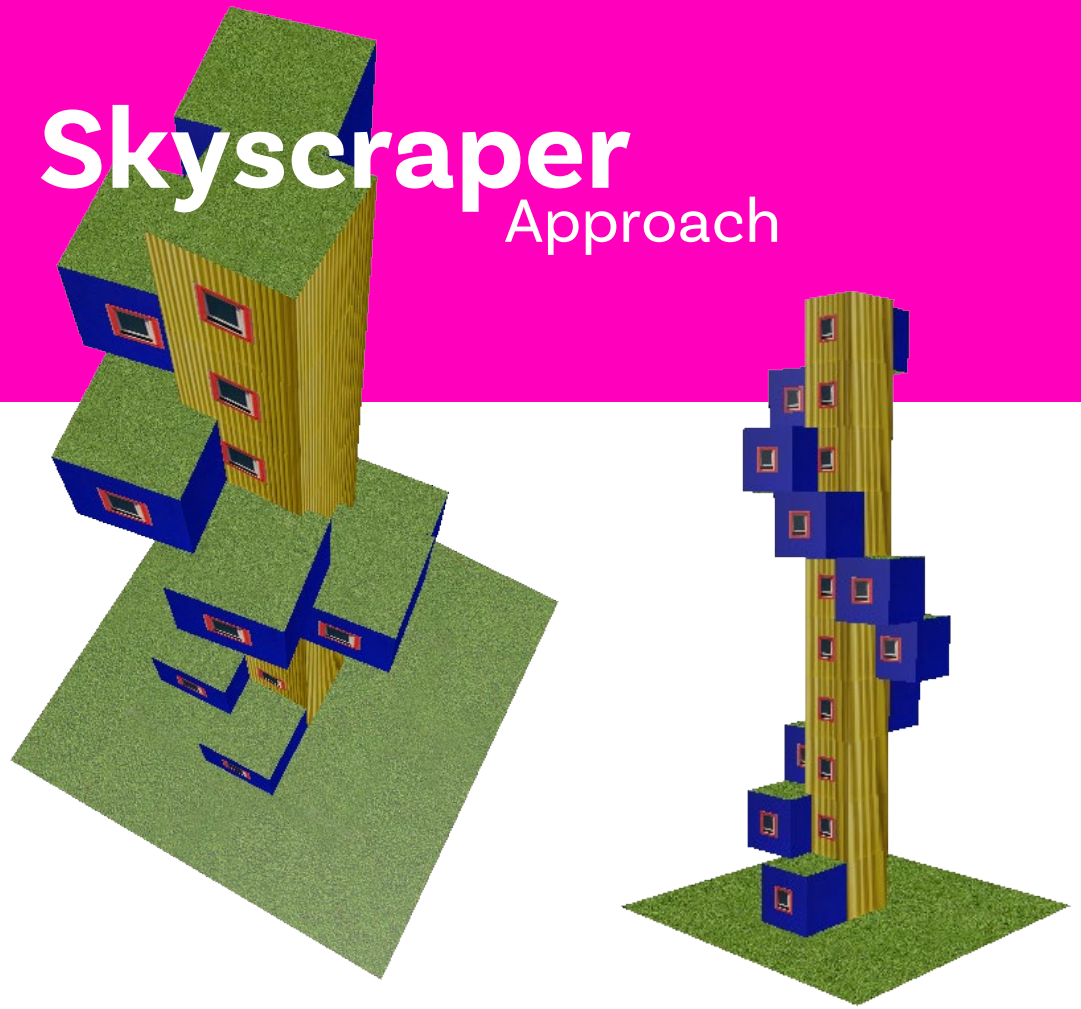


Skyscraper



Skyscraper

Approach



Generating the building

- Spiralling tower by rotating the floor each iteration
- Assigning yellow slats to the walls of the centre tower
- Assigning blue walls to the spiralling tower
- Assigning green roofs to the floors

```
for i in range(self.num_floors):
    angle = i + 2 * math.pi / 4
    height = max_width * i
    floor1 = app.add_mesh(GreenRoof(max_width, max_width), parent=self.building)

    floor1.set_transform(Mat4.from_translation(Vec3(max_width * math.cos(angle), height, max_width * math.sin(angle))) * Mat4.from_rotation_y(angle, True))
    floor1.set_visible(True)
```

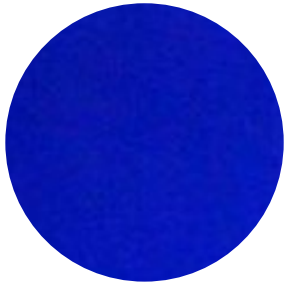
Skyscraper

- Inspired by spiralling elements
- Primary colours
- Green Roof

Textures

Grass

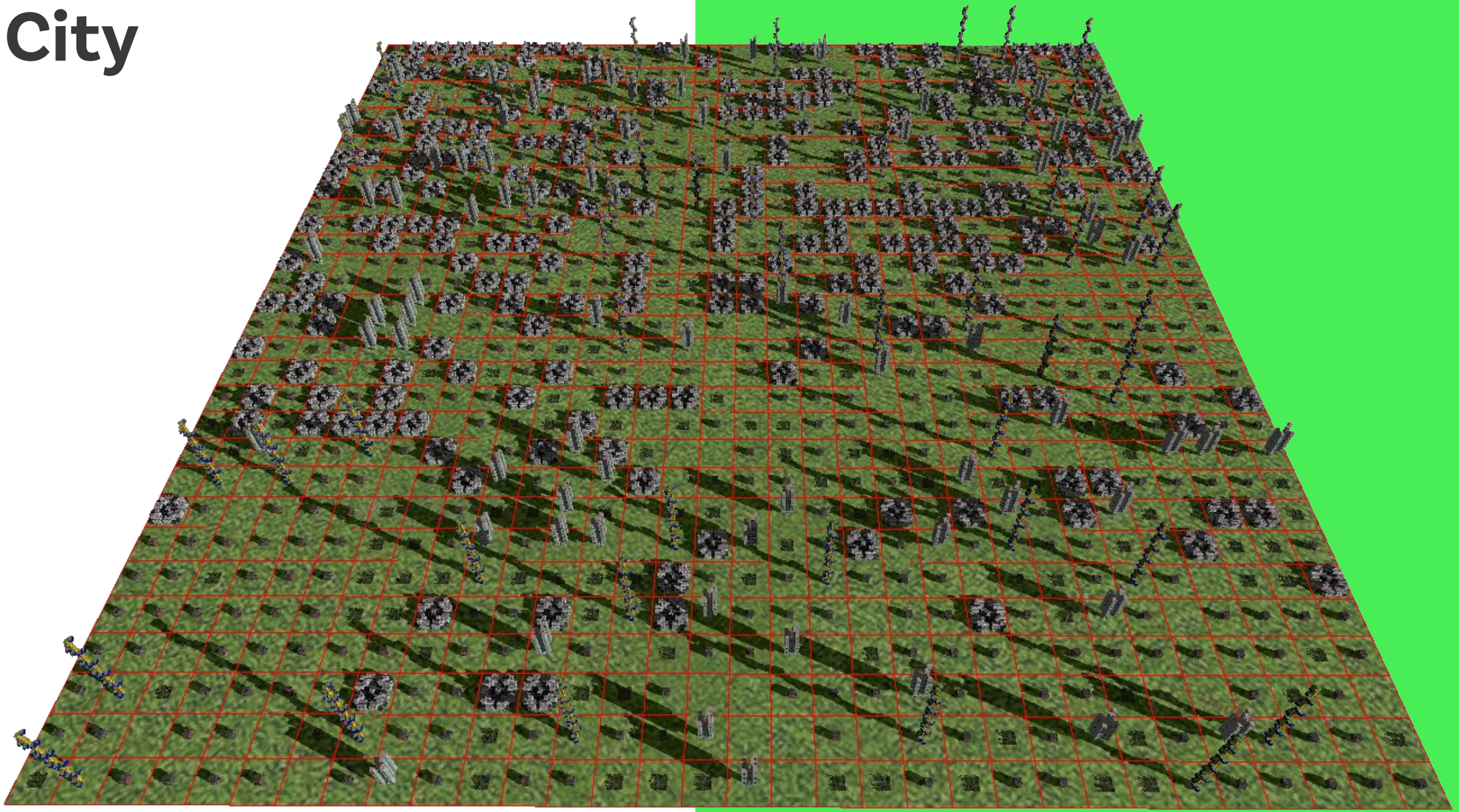
Yellow Slats



Blue Wall

Red Window

City



City

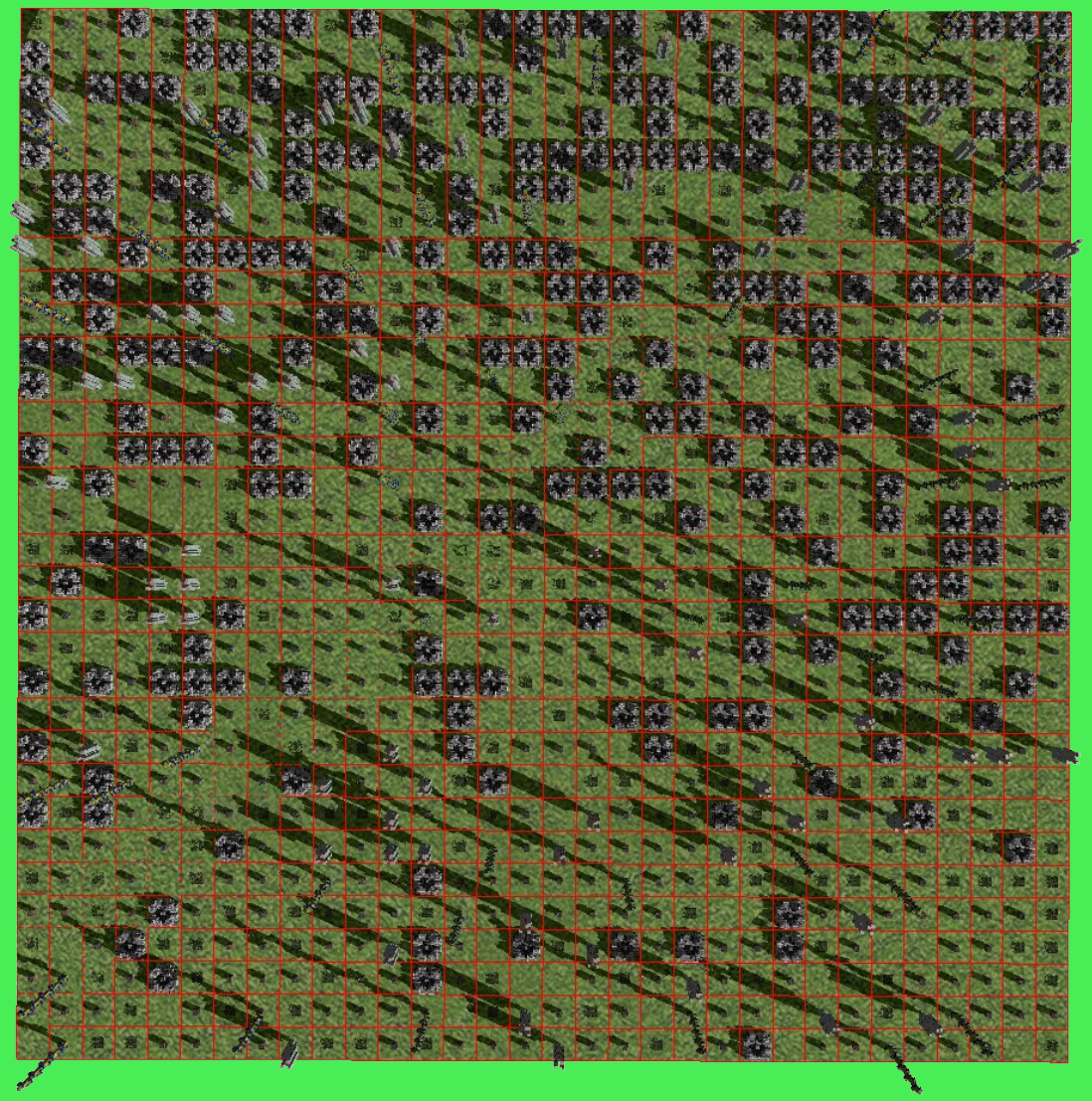
Approach

```
total_plots = self._plots_per_row * self._plots_per_col
skyscraper_percentage = 5
highrise_percentage = 8
office_percentage = 25
house_percentage = 37
park_percentage = 15

skyscraper_count = int(total_plots * skyscraper_percentage / 100)
highrise_count = int(total_plots * highrise_percentage / 100)
office_count = int(total_plots * office_percentage / 100)
house_count = int(total_plots * house_percentage / 100)
park_count = int(total_plots * park_percentage / 100)

building_types = (
    [BuildingType.SKYSCRAPER] * skyscraper_count +
    [BuildingType.HIGHRISE] * highrise_count +
    [BuildingType.OFFICE] * office_count +
    [BuildingType.HOUSE] * house_count +
    [BuildingType.PARK] * park_count +
    [BuildingType.EMPTY] * (total_plots - skyscraper_count - highrise_count - office_count - house_count - park_count)
)
```

```
def construct_building(self, row: int, col: int, building_type: BuildingType):
    building = None

    if building_type is BuildingType.HOUSE:
        building = House(self._app)
    elif building_type is BuildingType.OFFICE:
        grid_config = [
        ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
        ['0', '0', '1', '2', '2', '0', '2', '2', '1', '0', '0'],
        ['0', '1', '3', '2', '2', '1', '2', '2', '3', '1', '0'],
        ['0', '2', '2', '3', '1', '1', '1', '3', '2', '2', '0'],
        ['0', '2', '2', '1', '1', '1', '1', '1', '2', '2', '0'],
        ['0', '0', '1', '1', '1', '2', '1', '1', '1', '0', '0'],
        ['0', '2', '2', '1', '1', '1', '1', '1', '2', '2', '0'],
        ['0', '2', '2', '3', '1', '1', '1', '3', '2', '2', '0'],
        ['0', '1', '3', '2', '2', '1', '2', '2', '3', '1', '0'],
        ['0', '0', '1', '2', '2', '0', '2', '2', '1', '0', '0'],
        ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'],
        ]
        building = Office(self._app, 3, grid_config=grid_config)
        building.building.set_transform(Mat4.from_translation(Vec3(0, 0, 0)))
    elif building_type is BuildingType.HIGHRISE:
        building = Highrise(self._app, 10, 3)
    elif building_type is BuildingType.SKYSCRAPER:
        building = Skyscraper(self._app, 30, 3)
    elif building_type is BuildingType.PARK:
        building = Park(self._app)

    self._plots[row * self._plots_per_col + col] = building
```

City

Optimalization



```python
def count_adjacent_skyscrapers(self, row, col):
    """Counts the number of skyscrapers adjacent to a given position (including diagonals)."""
```

```python
def calculate_score(self):
    """Calculates the total score based on the number of adjacent skyscrapers and bad office placements."""
```

```python
def skyscraper_optimization_step(self):
    """Optimizes skyscrapers until the score is 0."""
```

```python
def swap_offices(self):
    """Swaps offices with random non-office, non-skyscraper, non-high-rise tiles."""
```

```python
def step(self, print_info=False):
    """Performs a single optimization step."""
```

```python
def optimize(self, print_info=True):
    """
    Runs the optimizer until the score reaches 0 or no further improvement, or after 50 iterations.
    Args:
        print_info (bool):
            Whether to print information about the optimization step.
    """
```