

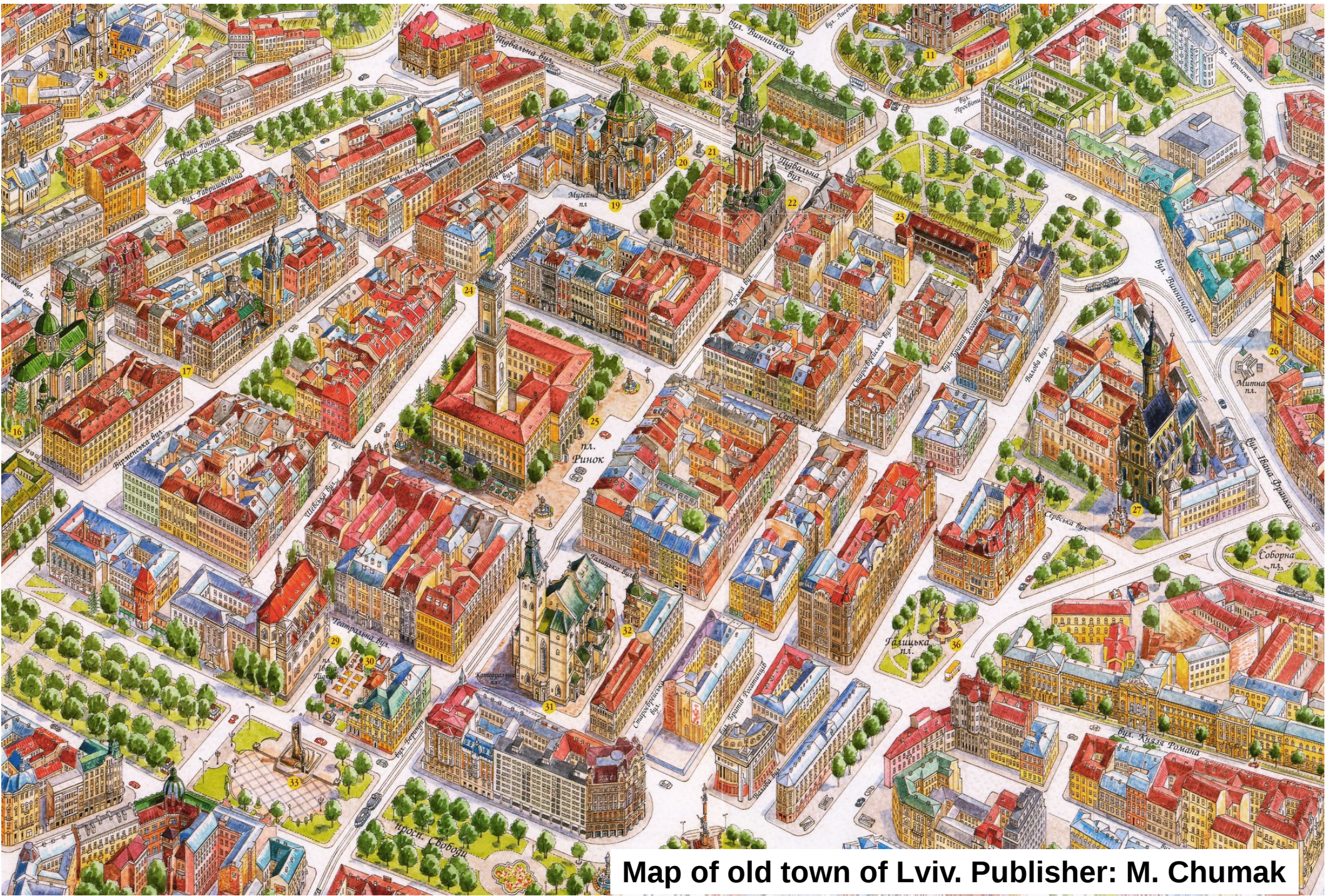
2.5D maps and bird views with OpenStreetMap and Blender

Vladimir Elistratov

github.com/vvoovv

vladimir.elistratov@gmail.com

Motivation: maps of Ruben Atoyan



Map of old town of Lviv. Publisher: M. Chumak

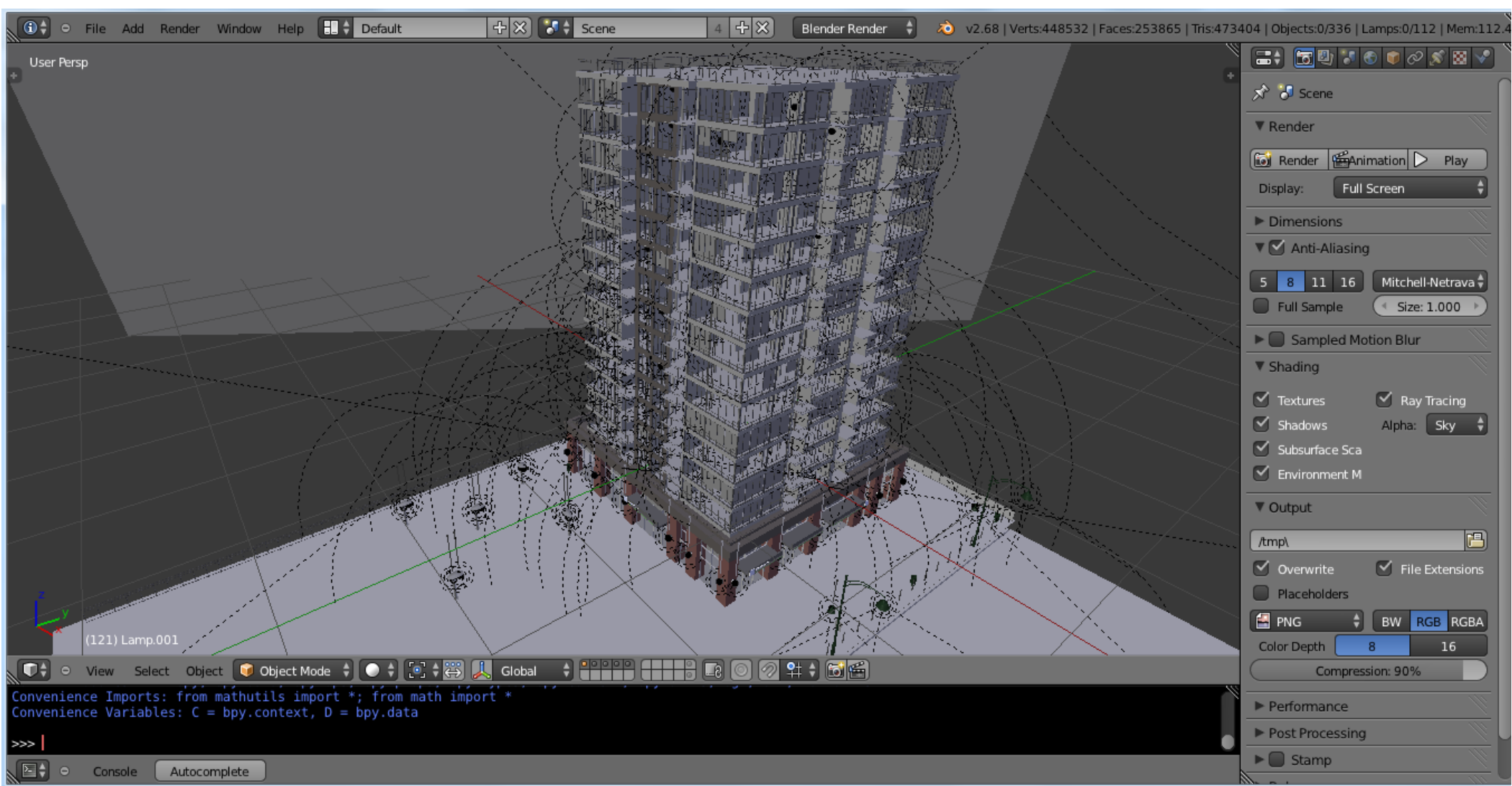
Example of a 2.5D map: <http://2gis.ru>





<http://www.blender.org>

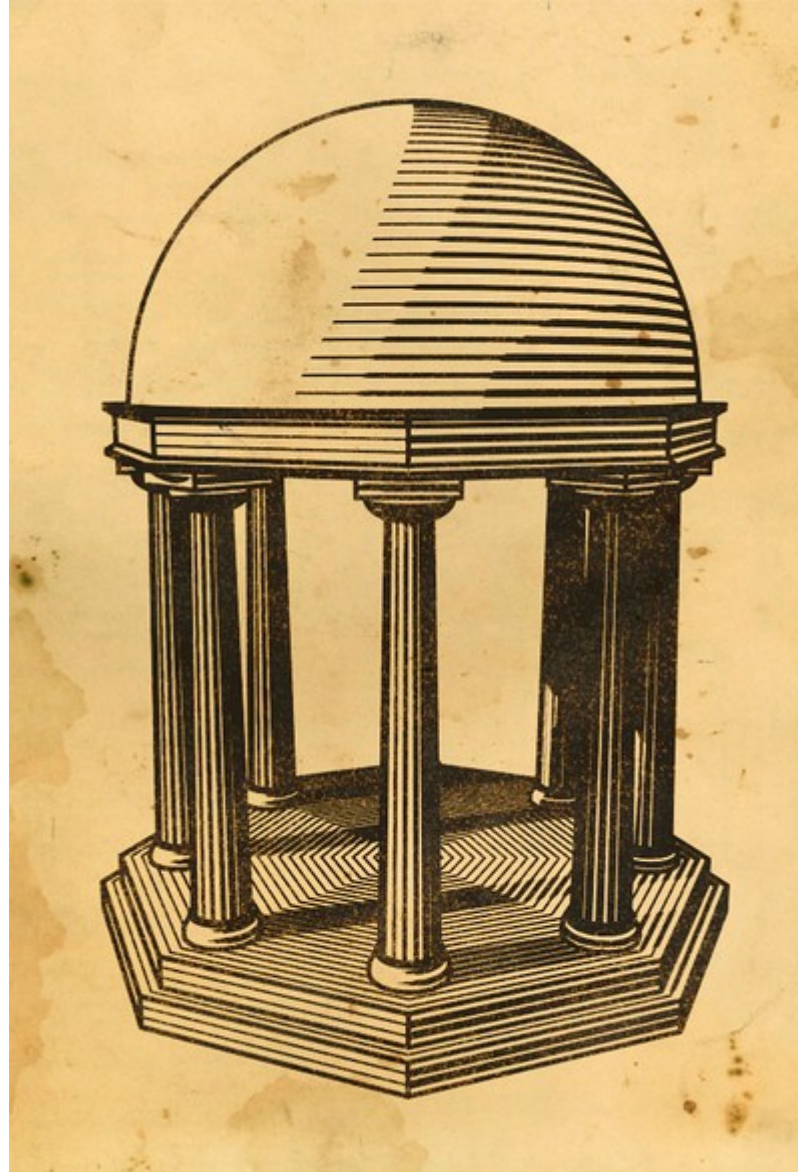
3D modeling



Photorealistic rendering



Non-photorealistic rendering



Charblaze CC-BY



Everything is accessible via Python

The screenshot displays the Blender 2.68 interface. The top status bar shows 'v2.68 | Verts:8 | Faces:6 | Tris:12 | Objects:1/3 | Lamps:0/1 | Mem:9.08M (0.11M) | Cube'. The main area is a Python script editor for a file named 'common_image.py'. The script defines a 'CommonImage' class that inherits from 'Map25D'. It sets 'outputImagesDir' to '.' and 'gdalDir' to '.'. It also defines a 'projection' attribute as 'SphericalMercator()' and an '__init__' method that checks if 'outputImagesDir' exists. The bottom panel is the 'PYTHON INTERACTIVE CONSOLE 3.3.0', which shows the command history and built-in modules for the current session.

```
1 #!/usr/bin/env python3
2
3 import bpy, mathutils
4 import math, os, subprocess
5 from spherical_mercator import SphericalMercator
6
7 from map_25d import Map25D
8
9 class CommonImage(Map25D):
10
11     outputImagesDir = "."
12
13     gdalDir = "."
14
15     # position of the Blender zero point in the Mercator projection
16     x = 0
17     y = 0
18     # the Blender zero point is set to the first object
19     zeroPointSet = False
20
21     projection = SphericalMercator()
22
23     def __init__(self, **kwargs):
24         super().__init__(**kwargs)
25         # check if outputImagesDir exists
26         if not os.path.exists(self.outputImagesDir):
```

PYTHON INTERACTIVE CONSOLE 3.3.0 (default, Nov 26 2012, 15:38:36) [MSC v.1500 64 bit (AMD64)]

Command History: Up/Down Arrow
Cursor: Left/Right Home/End
Remove: Backspace/Delete
Execute: Enter
Autocomplete: Ctrl-Space
Zoom: Ctrl +/-, Ctrl-Wheel
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> |



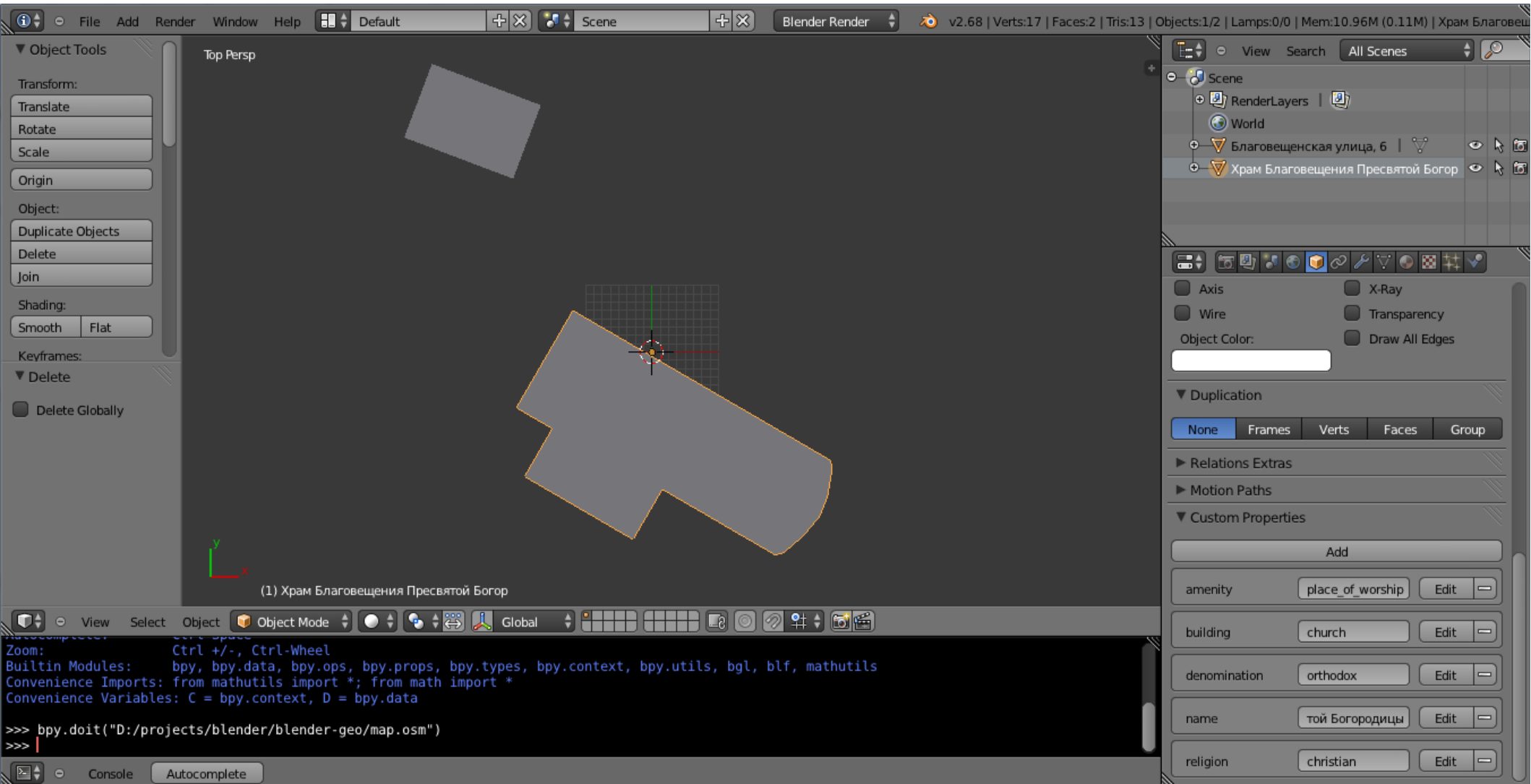
Procedure

- Prepare 3D models of buildings
- Georeference each 3D model in Blender using OpenStreetMap data
- Arrange georeferenced 3D models according to the application (example: 2.5D map)

3D model

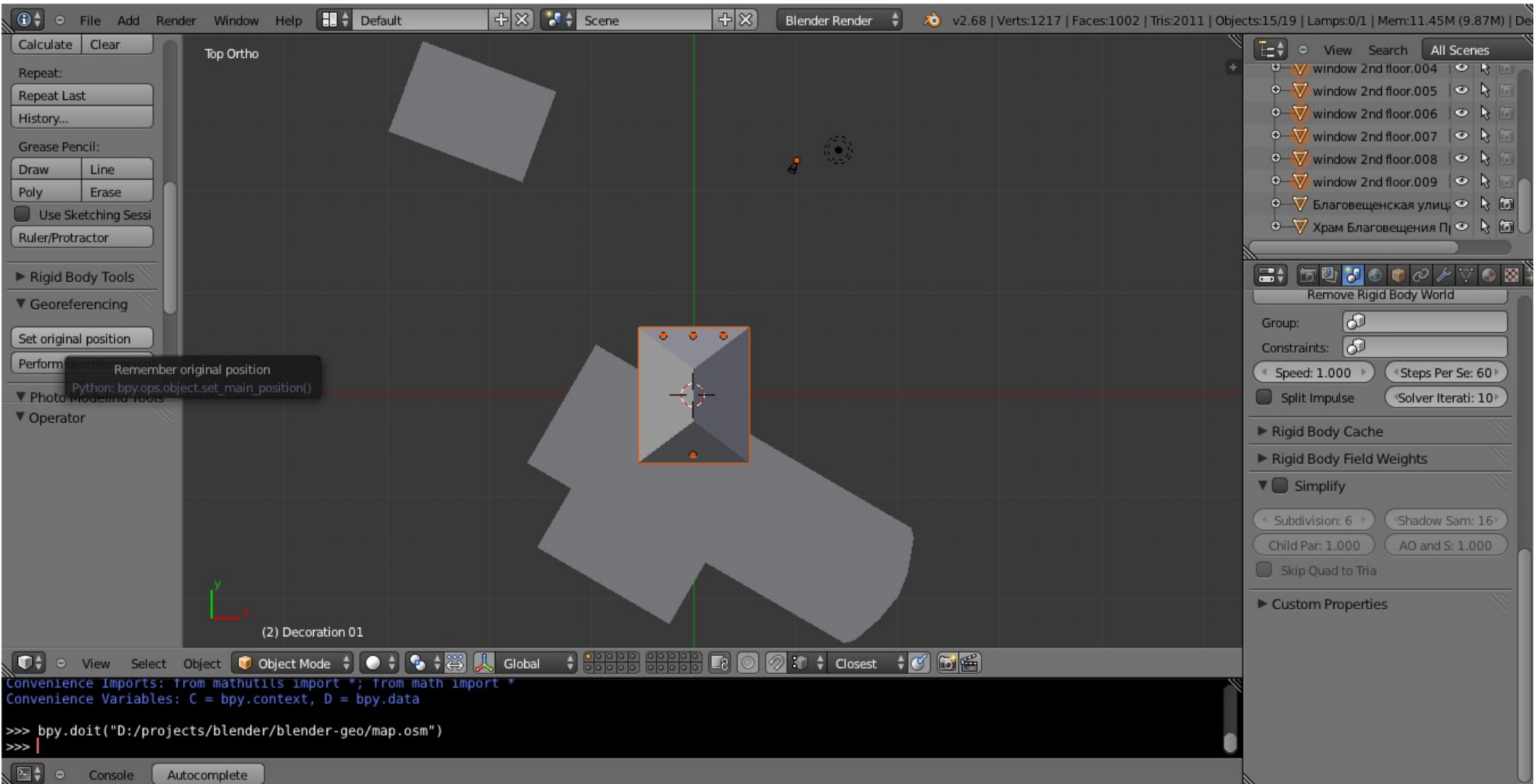


3D modeling: OSM import

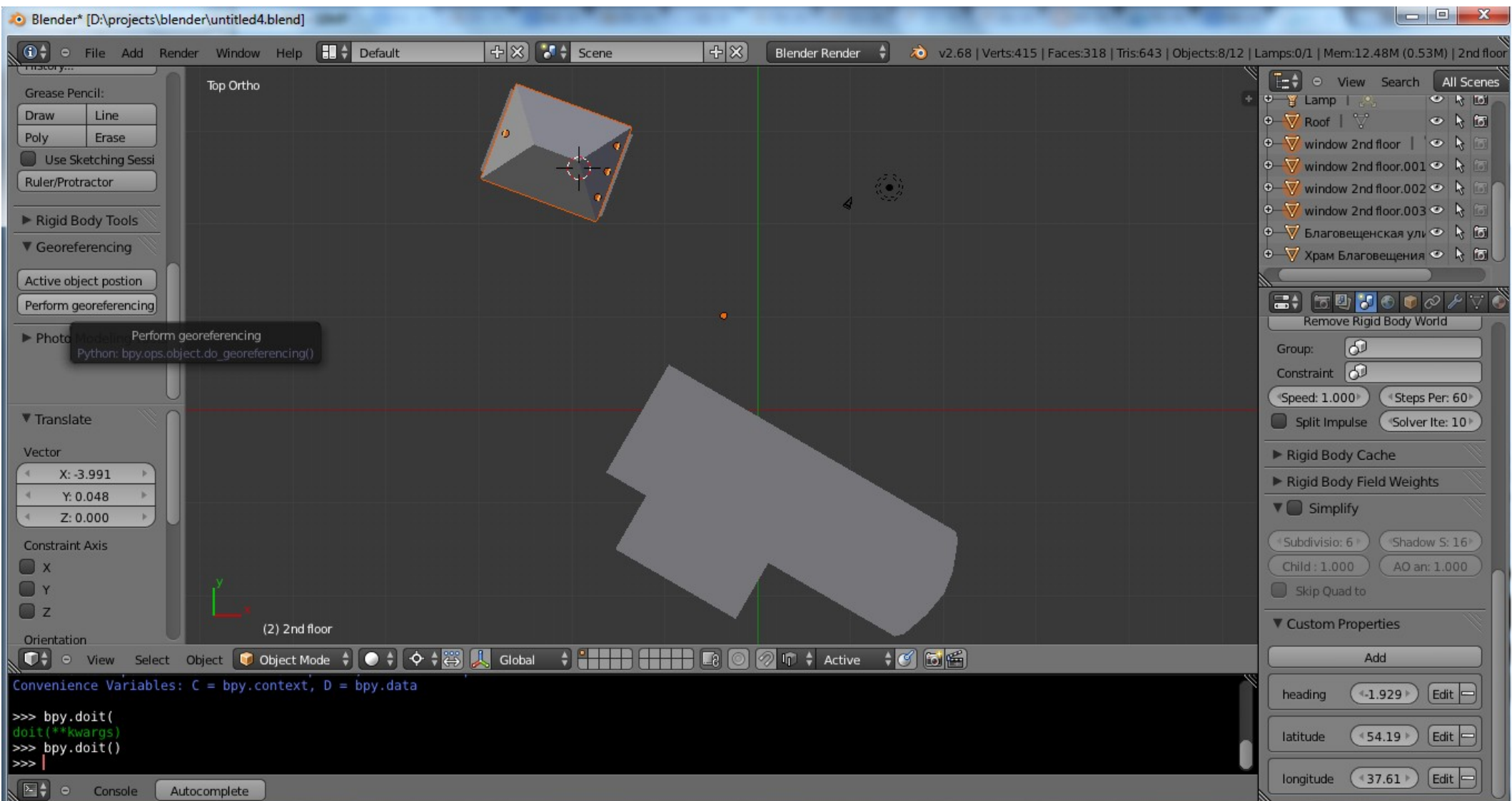


Transverse Mercator projection for the imported OSM Data

Georeferencing with OSM Data



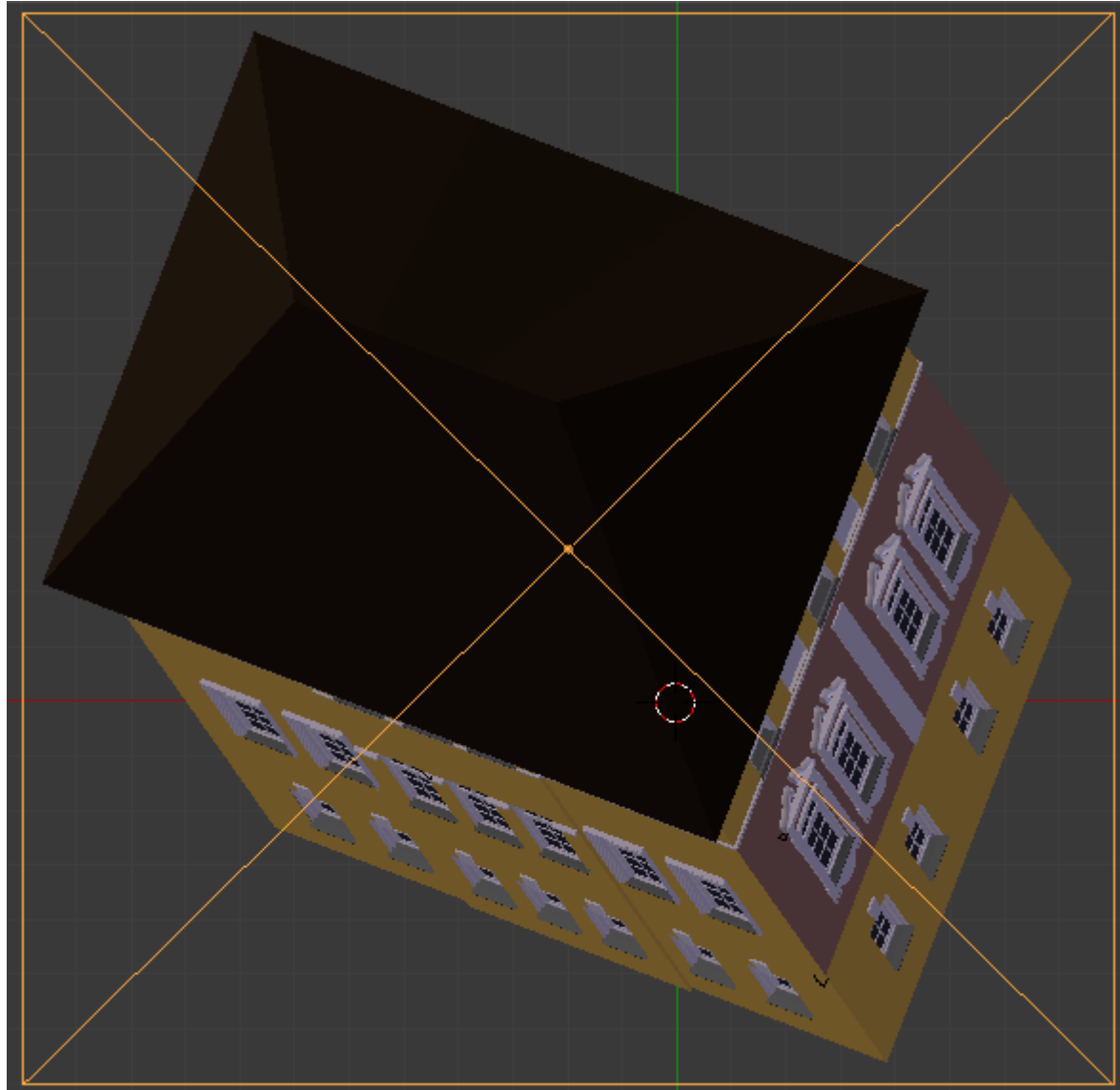
Georeferencing with OSM Data



2.5D maps: 2 methods

- Every 3D object is rendered in the oblique projection as a separate graphic symbol. Mapnik's PointSymbolizer is employed to render those images at the center of the corresponding object.
- All 3D objects are rendered in the oblique projection in one image. The image is georeferenced and rendered on the map as a raster layer.

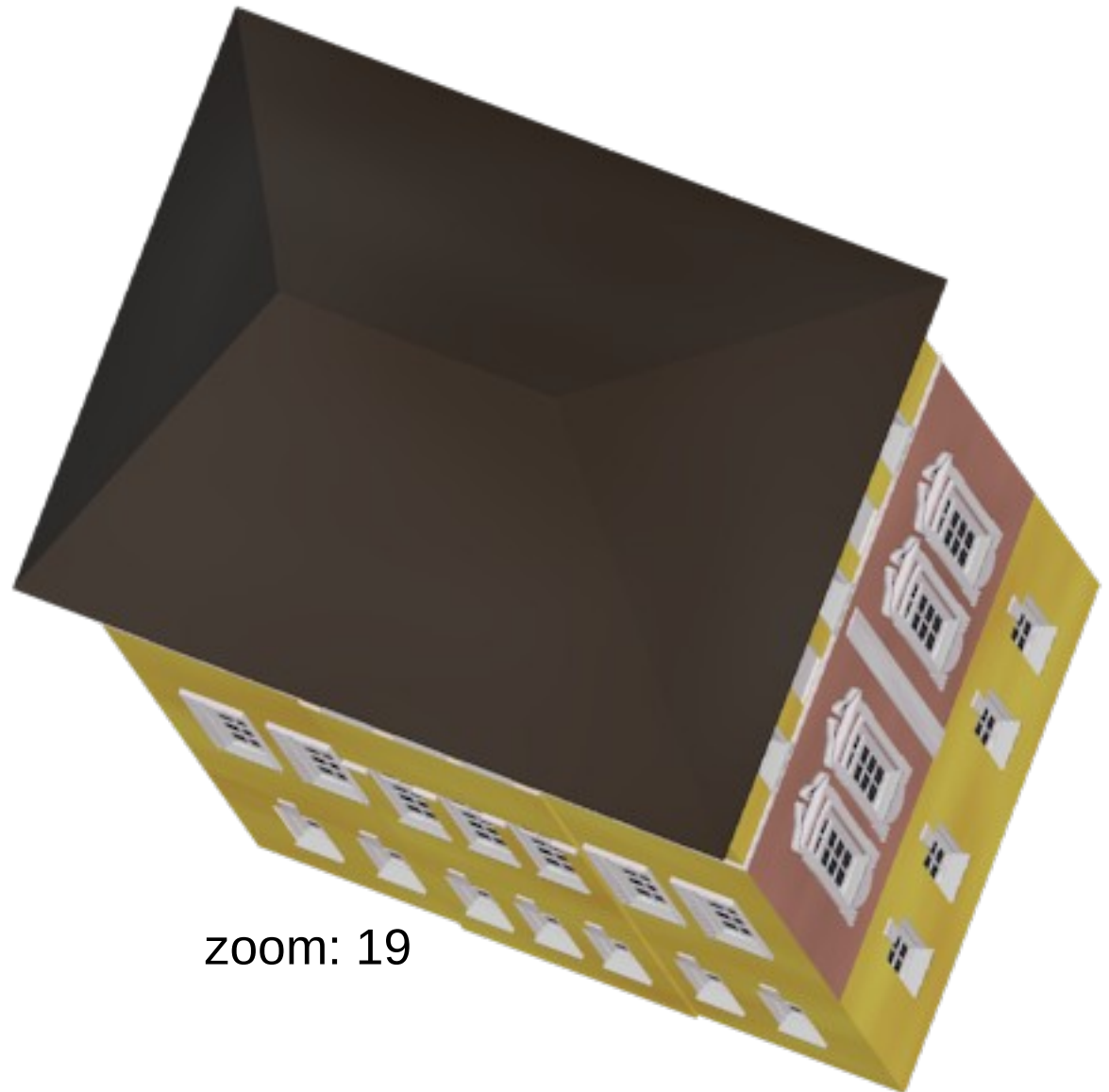
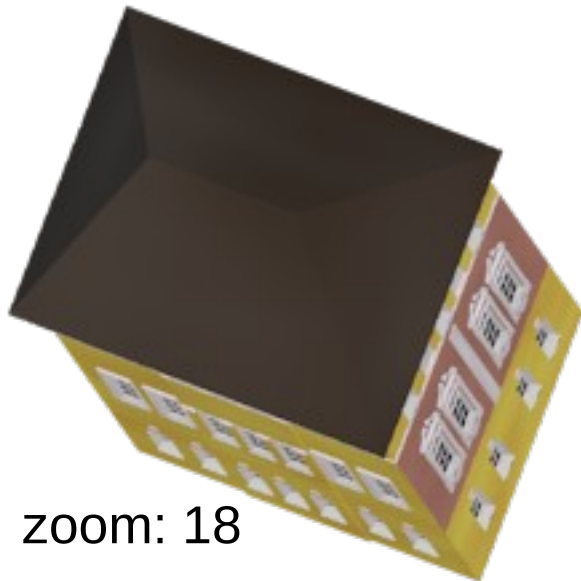
2.5D maps: separate images



- Apply scaling $1/\cos(\text{lat})$ to take distortion of the spherical Mercator projection into account
- Simulate oblique projection in accordance with the given angles of projection
- Adjust the location of the orthographic camera and its viewport size
- Adjust the size of the resulting image. It depends on zoom

2.5D maps: separate images

Output: images



2.5D maps: separate images

Output: .csv file

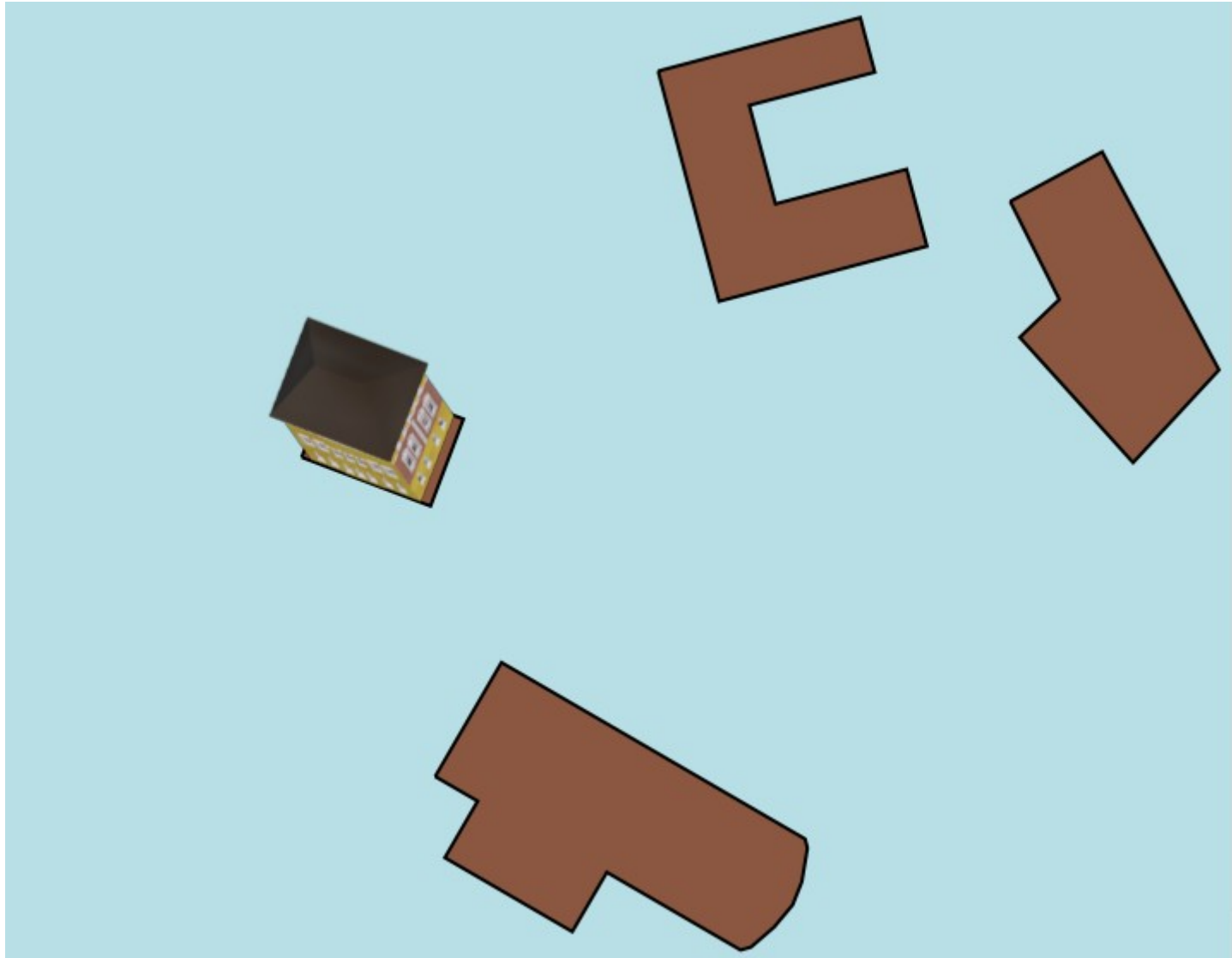
- modelId
- lat
- lon
- image_z17
- dx_z17
- dy_z17
- image_z18
- dx_z18
- dy_z18
- image_z19
- dx_z19
- dy_z19

2.5D maps: separate images

CartoCSS .mss file

```
1 .models {
2   [zoom=17] {
3     point-file: url("models/[image_z17].png");
4     point-transform: translate([dx_z17],[dy_z17]);
5   }
6   [zoom=18] {
7     point-file: url("models/[image_z18].png");
8     point-transform: translate([dx_z18],[dy_z18]);
9   }
10  [zoom=19] {
11    point-file: url("models/[image_z19].png");
12    point-transform: translate([dx_z19],[dy_z19]);
13  }
14 }
```

2.5D maps: separate images

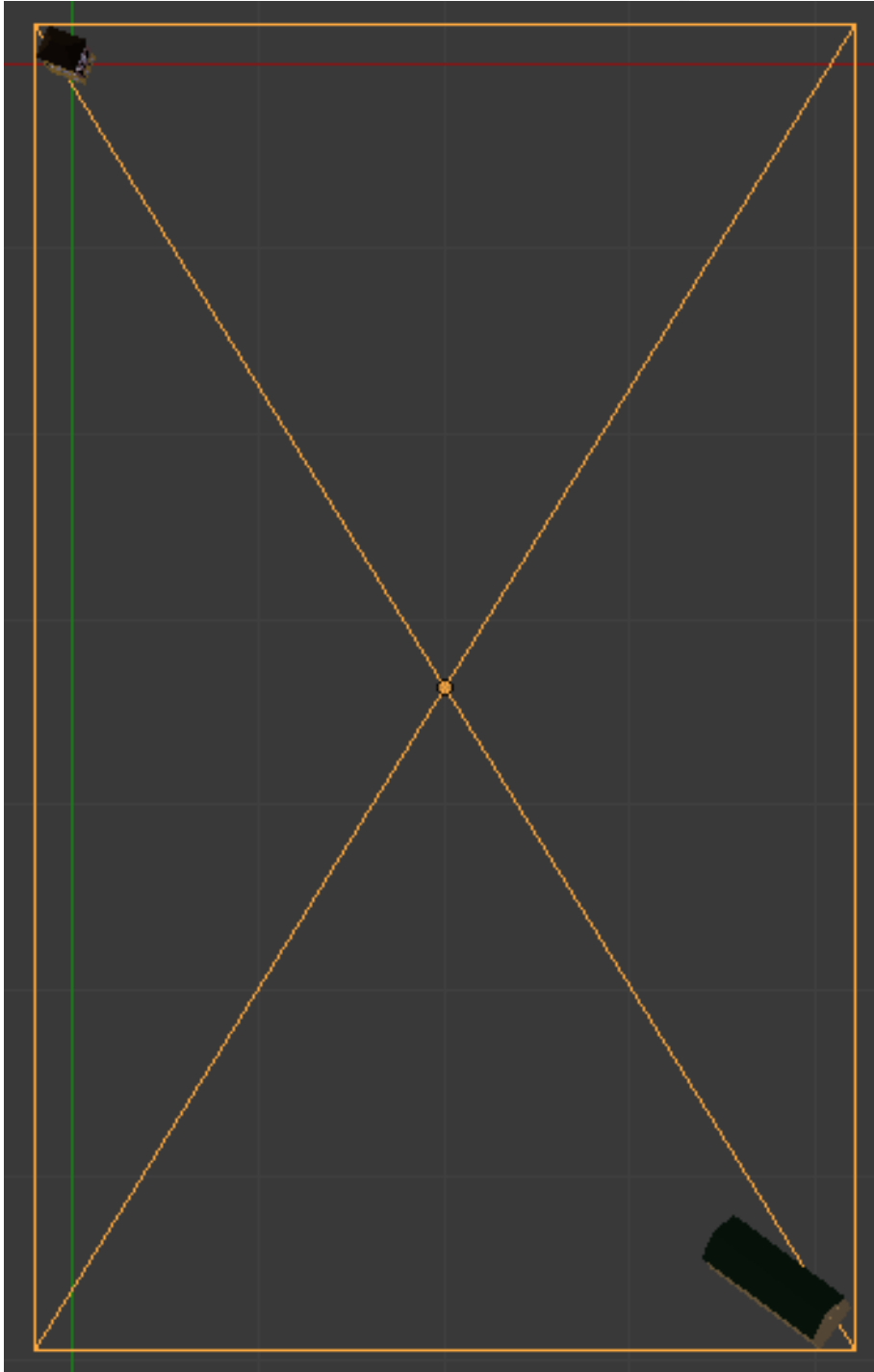


2.5D maps: separate images

Disadvantage

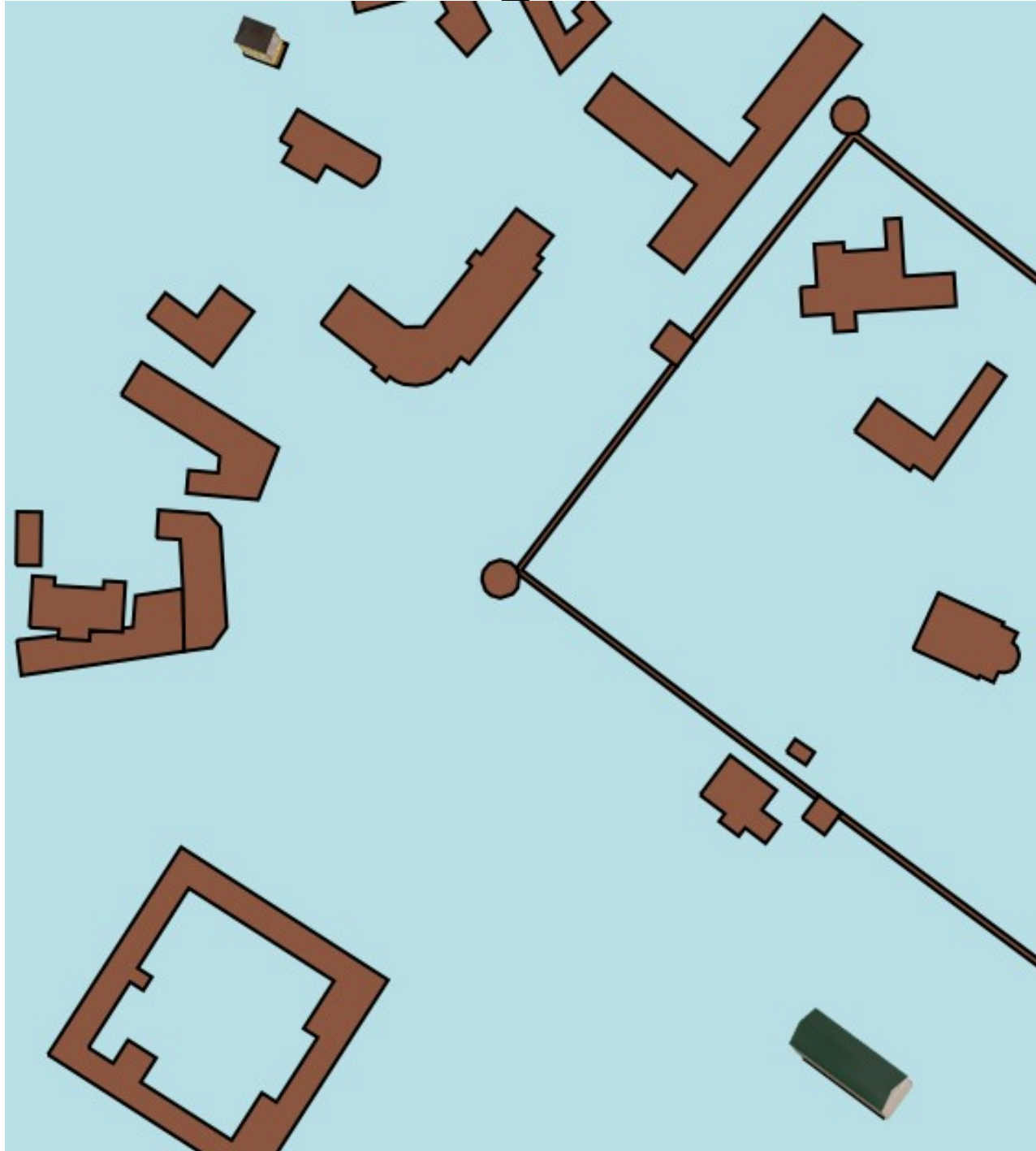
Buildings should not stay too close to each other!

2.5D maps: common images



- Apply scaling $1/\cos(\text{lat})$ to every 3D object
- Simulate oblique projection in accordance with the given angles of projection
- Fit the location of the orthographic camera and its viewport size
- Fit the size of the resulting image. It depends on zoom
- Generate a GeoTiff file with `gdal_translate`
- Use the GeoTiff file as raster layer in Mapnik

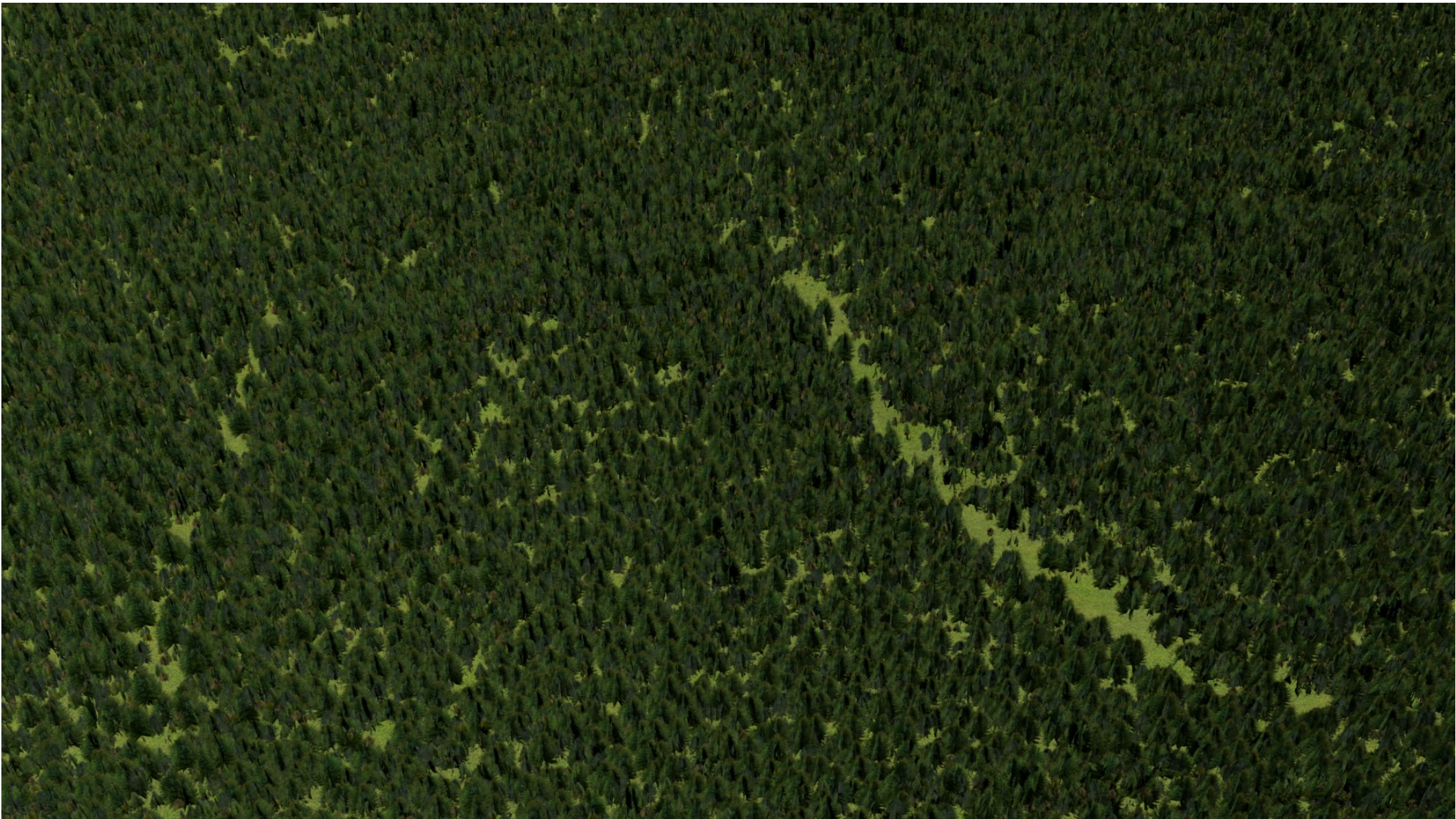
2.5D Karten: das gemeinsame Bild



2.5D maps

<https://github.com/vvoovv/blender-2.5dmaps>

Forests for bird views: next time :(



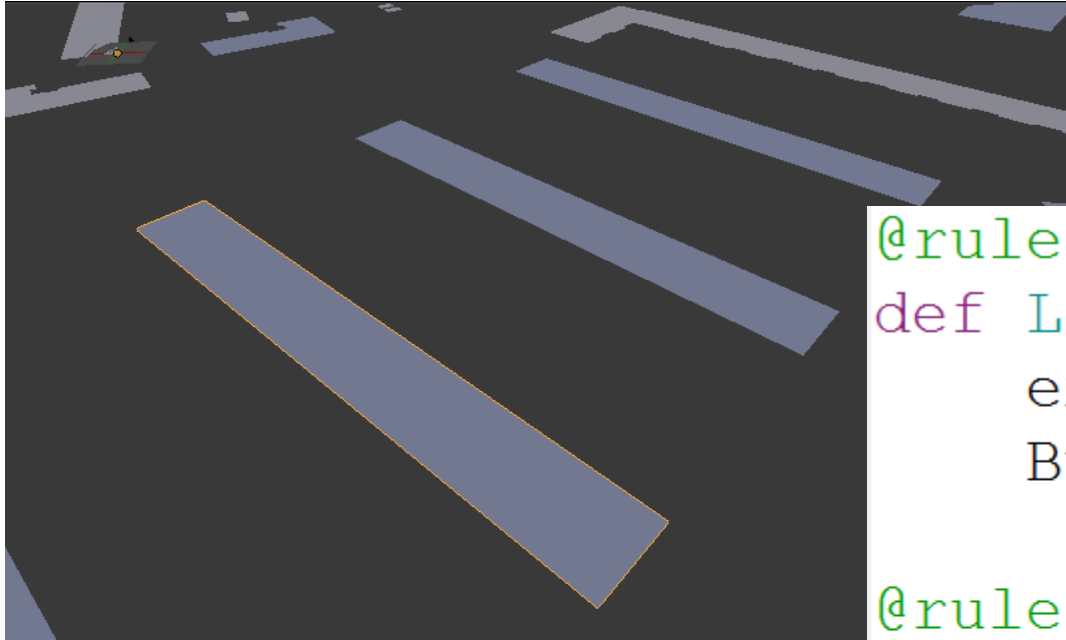
Simplification of 3D modeling: CGA Shape Grammar

- 3D modeling is tedious
- Promising approach to simplify 3D modeling:
CGA Shape Grammar
- CGA stands for Computer Generated
Architecture
- CGA Shape Grammar was developed in the
Swiss Federal Institute of Technology in Zurich
(ETH Zurich)
- CGA Shape Grammar makes up the foundation
of ESRI CityEngine software

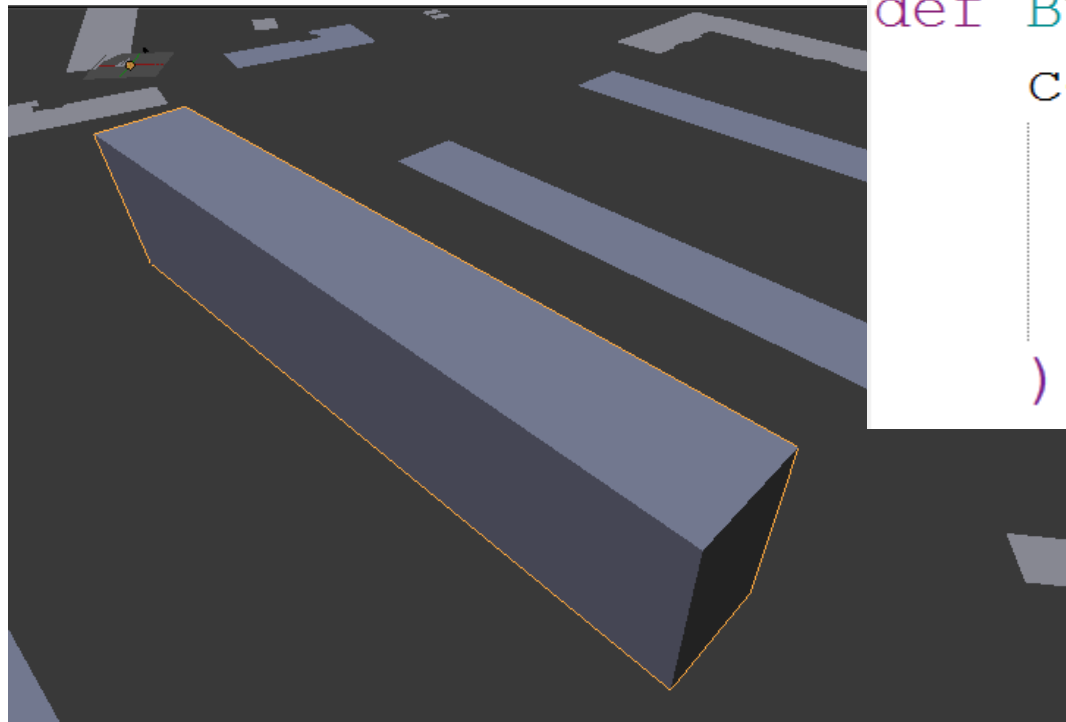
CGA Shape Grammar

- CGA Shape Grammar consists of shape rules
- Each rule iteratively refines a design by creating more details
- The first rule is normally applied to a building outline
- CGA Shape Grammar is expressed in Python in my implementation
- Evaluated in Blender

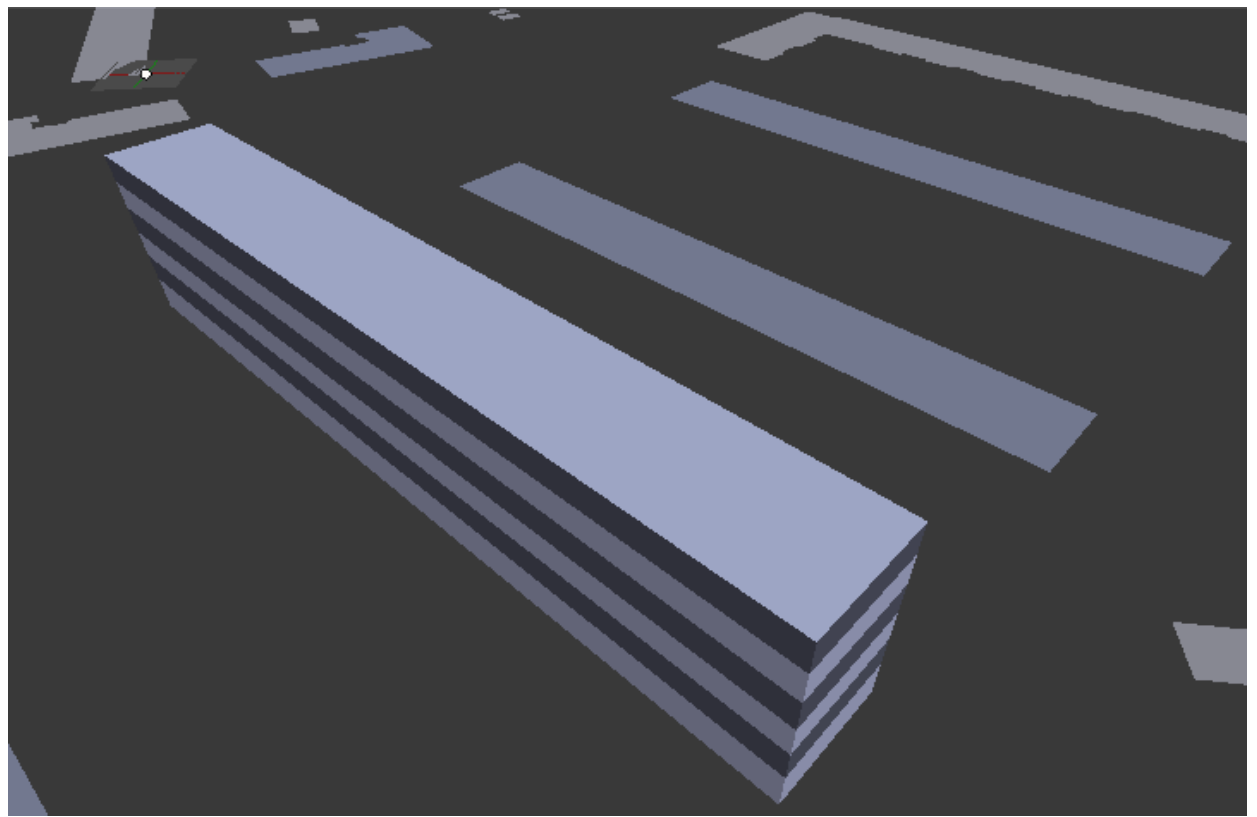
CGA Shape Grammar: Example



```
@rule
def Lot():
    extrude(height)
    Building()
```

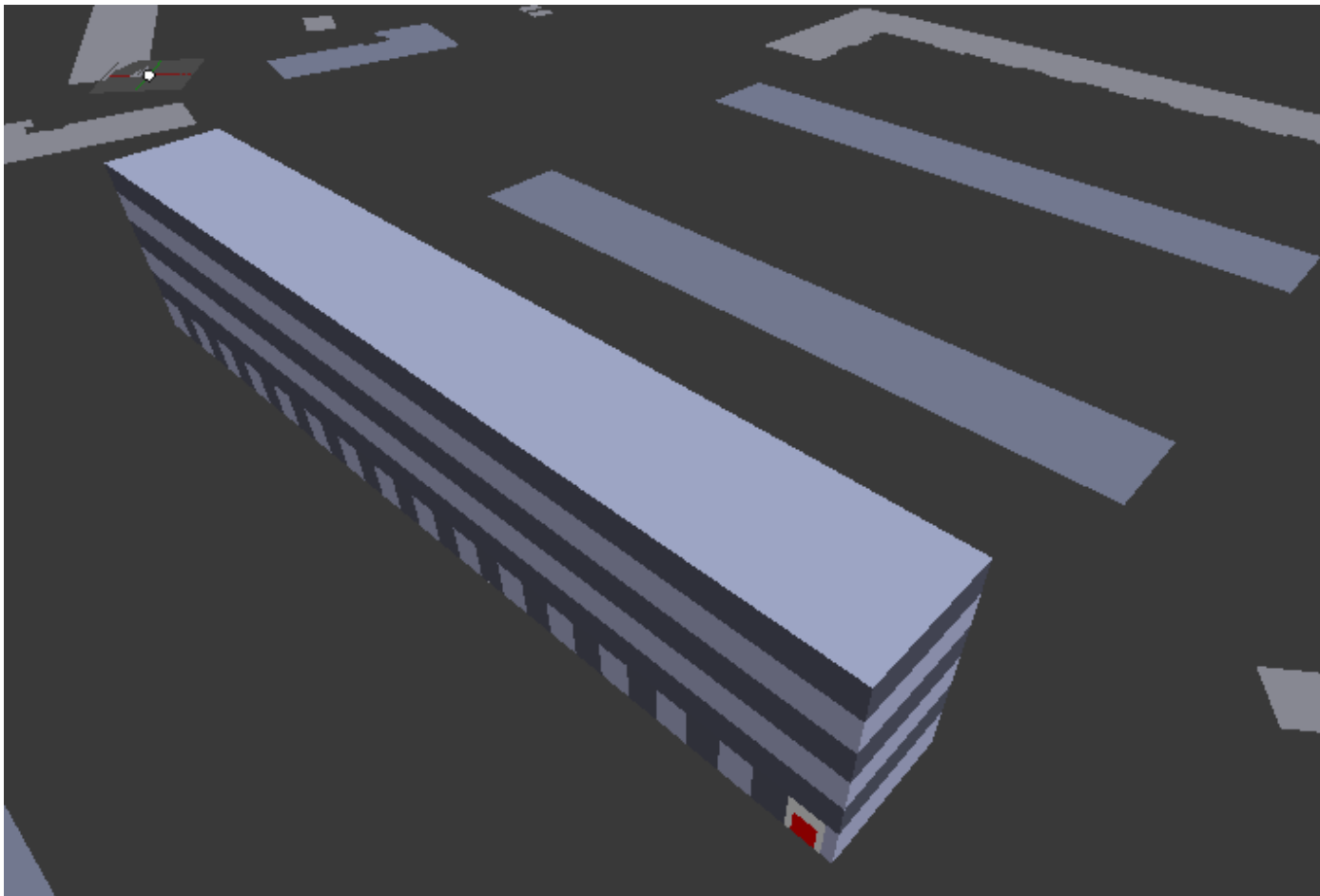


```
@rule
def Building():
    comp(f).into(
        front>>FrontFacade() |
        side>>SideFacade() |
        top>>Roof()
    )
```



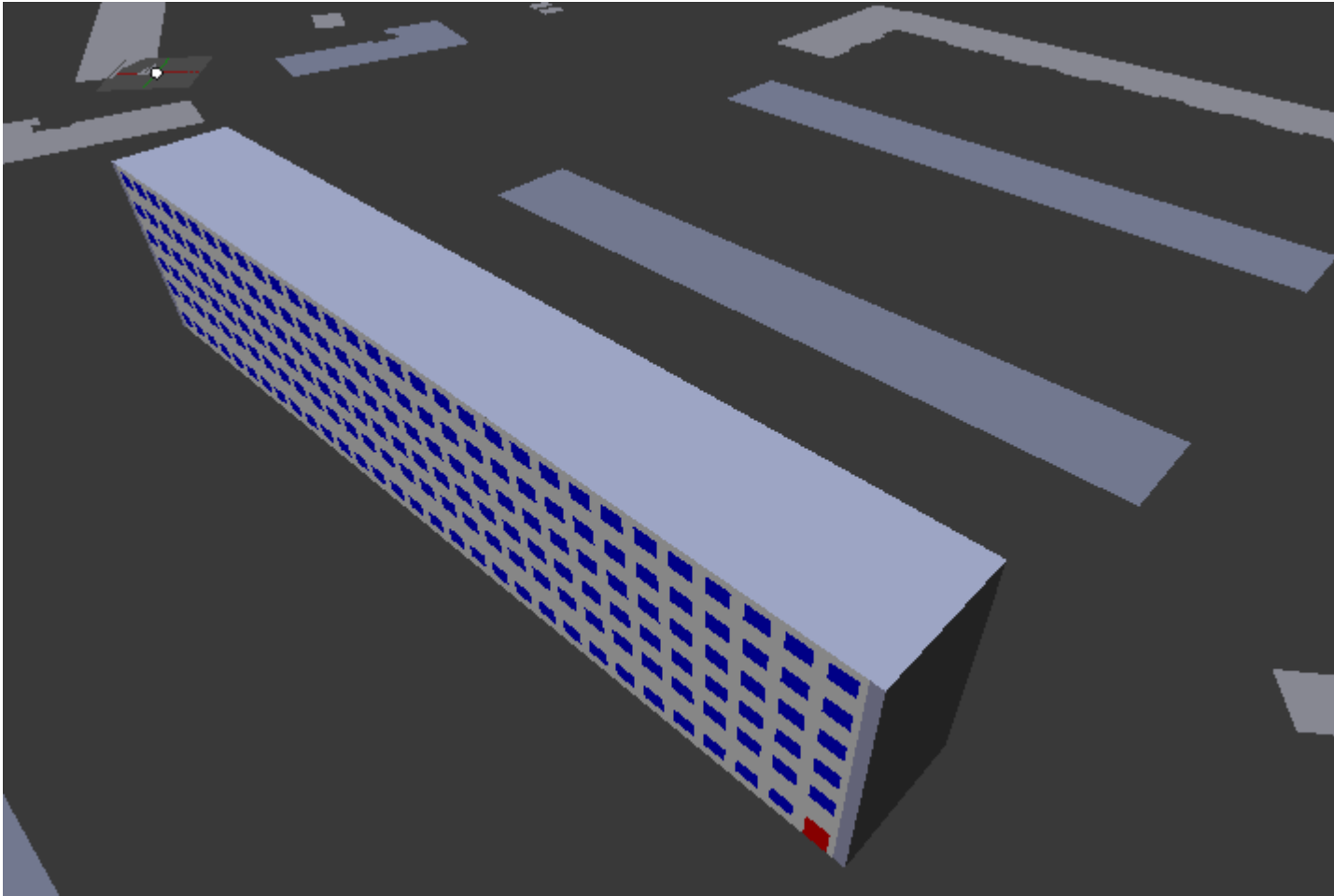
```
@rule
def FrontFacade():
    split(y).into(
        groundFloorHeight>>GroundFloor() |
        repeat(flt(floorHeight)>>Floor())
    )
```

```
@rule
def SideFacade():
    split(y).into(
        groundFloorHeight>>Floor() |
        repeat(flt(floorHeight)>>Floor())
    )
```

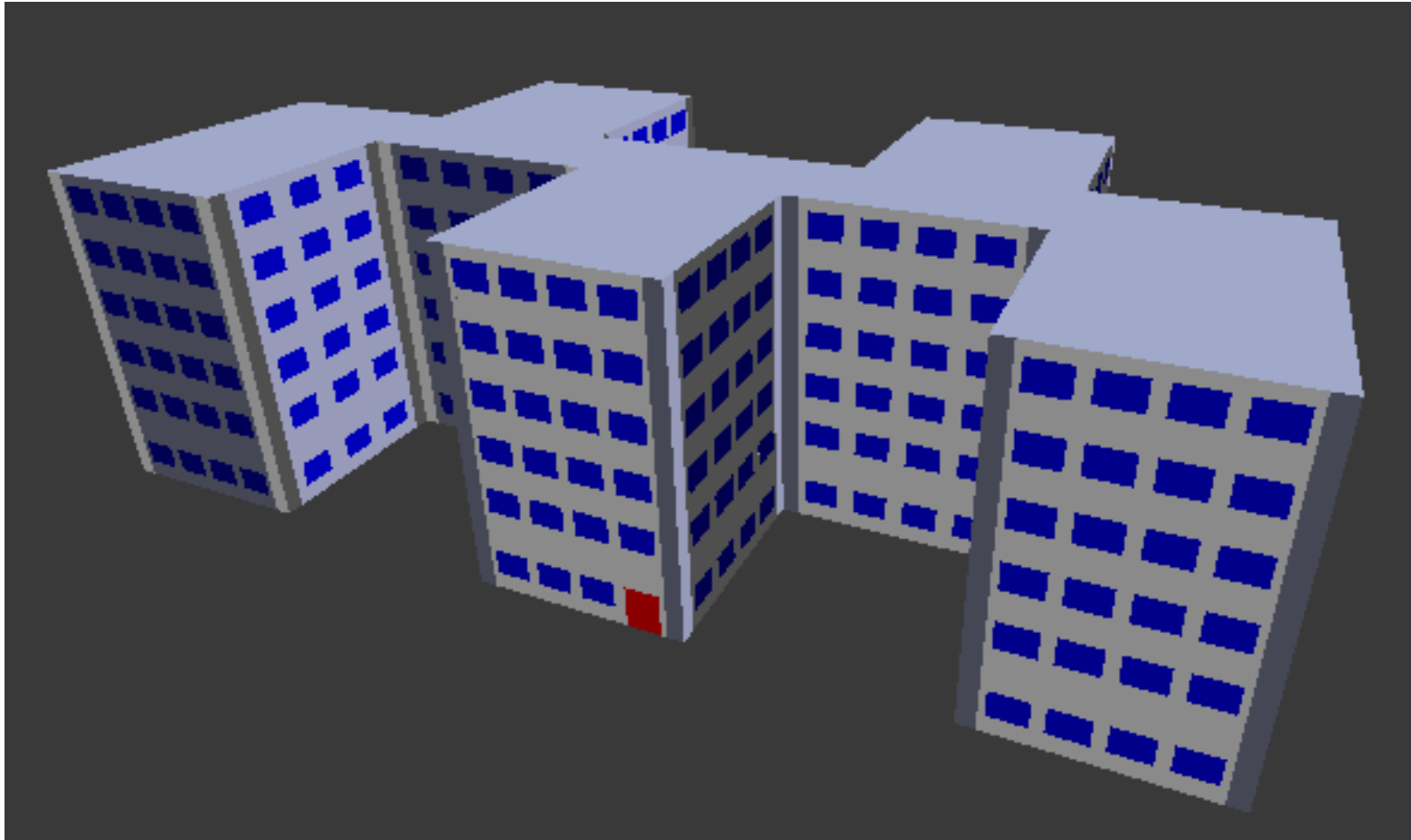


```
@rule
def GroundFloor():
    split(x).into(
        1>>Wall() |
        repeat(flt(tileWidth)>>Tile()) |
        flt(tileWidth)>>EntranceTile() |
        1>>Wall()
    )
```

CGA Shape Grammar: Example



CGA Shape Grammar: Example



CGA Shape Grammar Development

- Roofs
- Textures
- Placement of 3D assets (e.g. windows, facade decorations)

CGA Shape Grammar in Blender

<https://github.com/vvoovv/blender-cga>

github.com/vvoovv

vladimir.elistratov@gmail.com