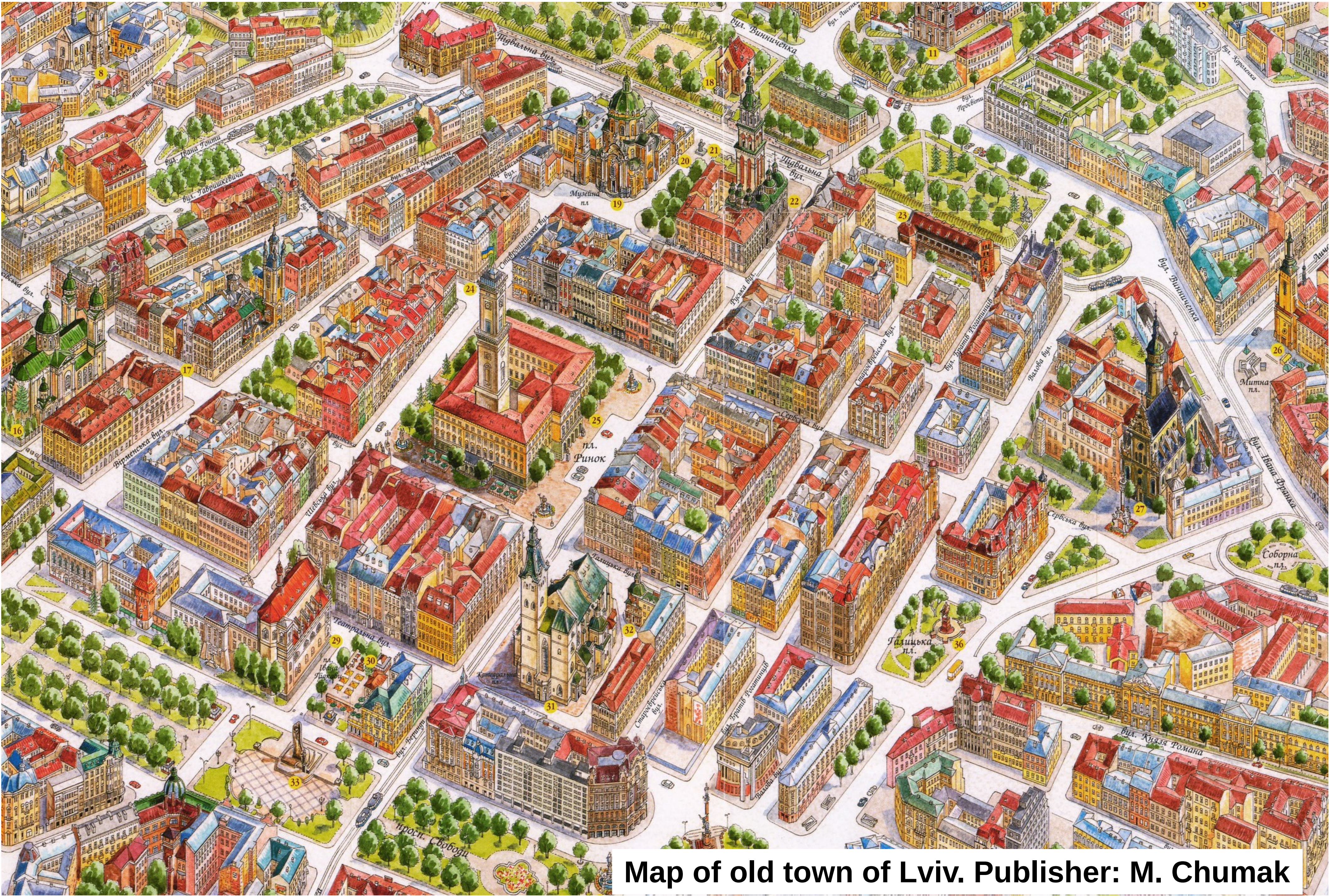
2.5D maps and bird views with OpenStreetMap and Blender

Vladimir Elistratov

github.com/vvoovv

vladimir.elistratov@gmail.com

Motivation: maps of Ruben Atoyan

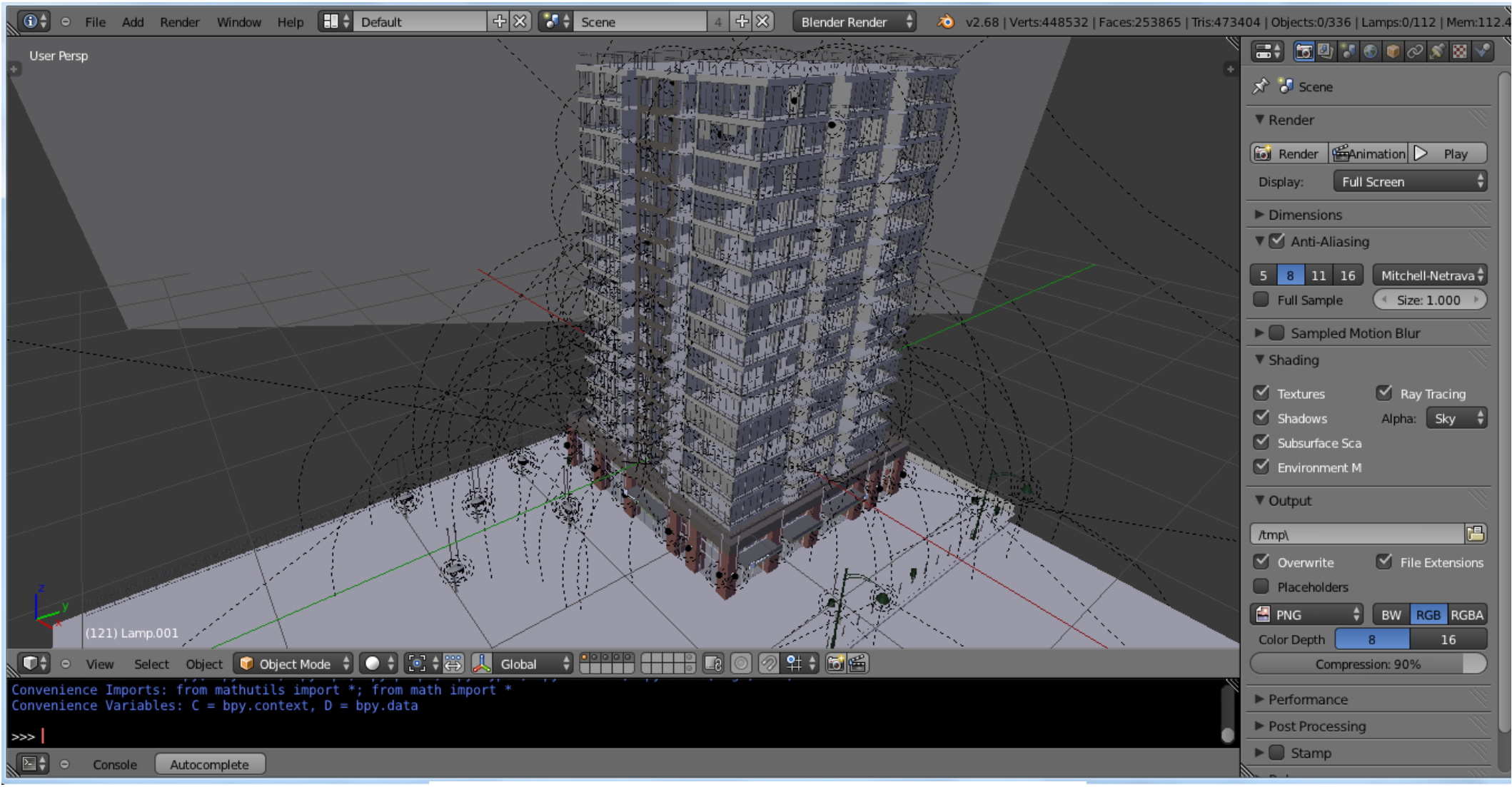


Map of old town of Lviv. Publisher: M. Chumak

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

http://www.blender.org

3D modeling



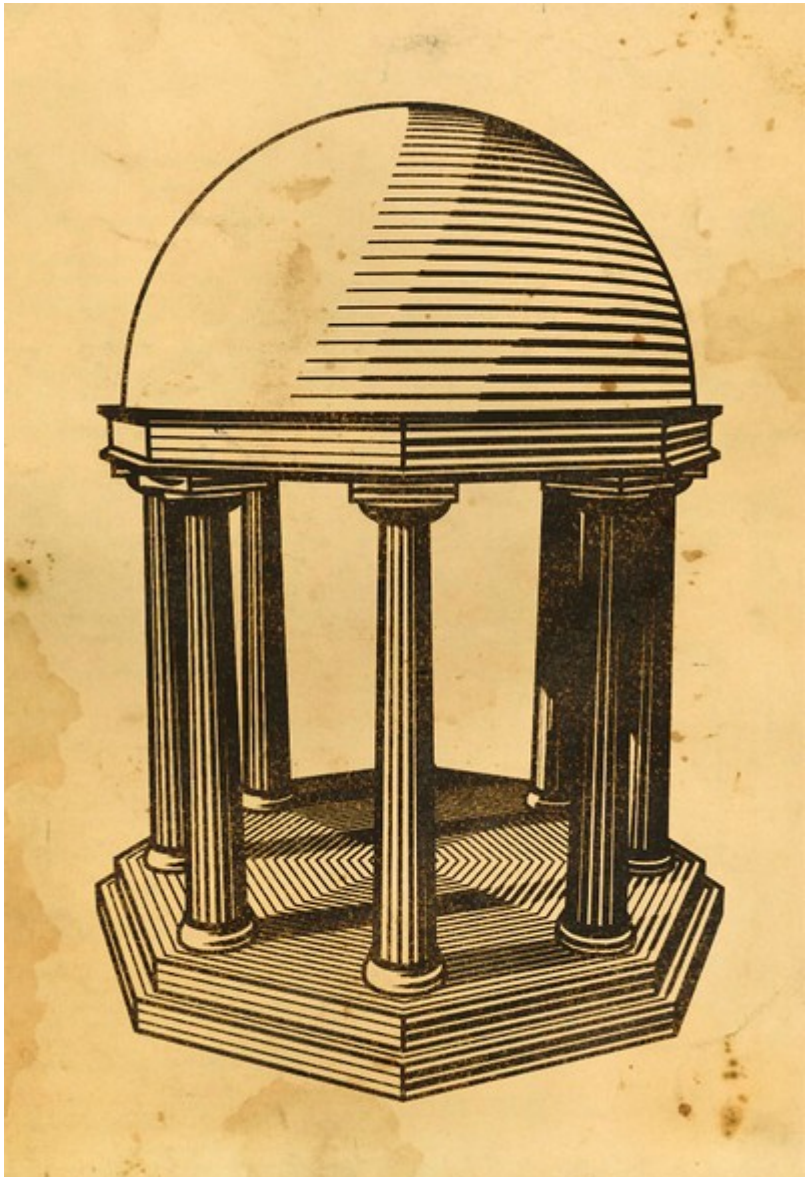
blender™

Photorealistic rendering

http://www.blenderguru.com

blender™

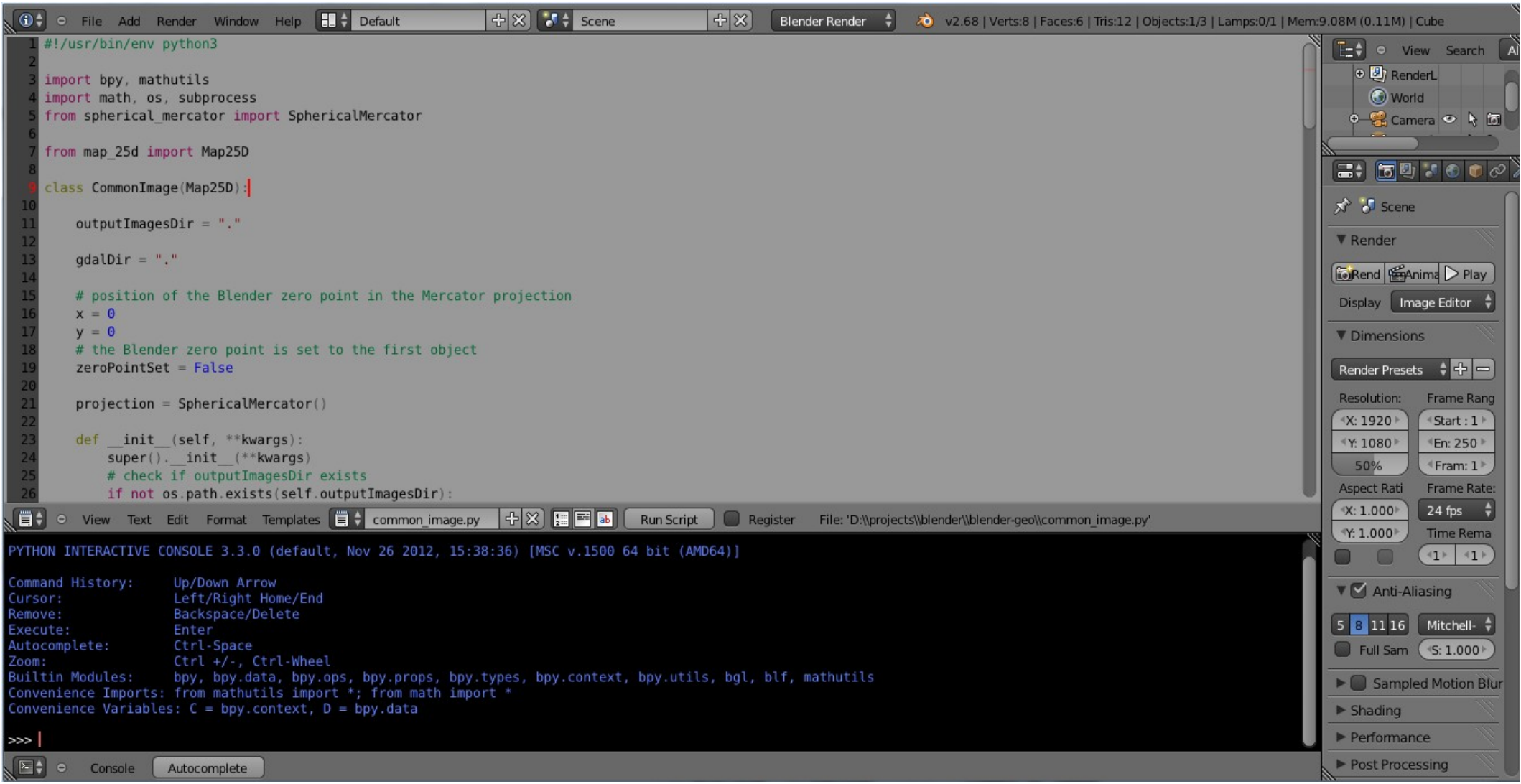
Non-photorealistic rendering



Charblaze CC-BY

C

Everything is accessable via Python



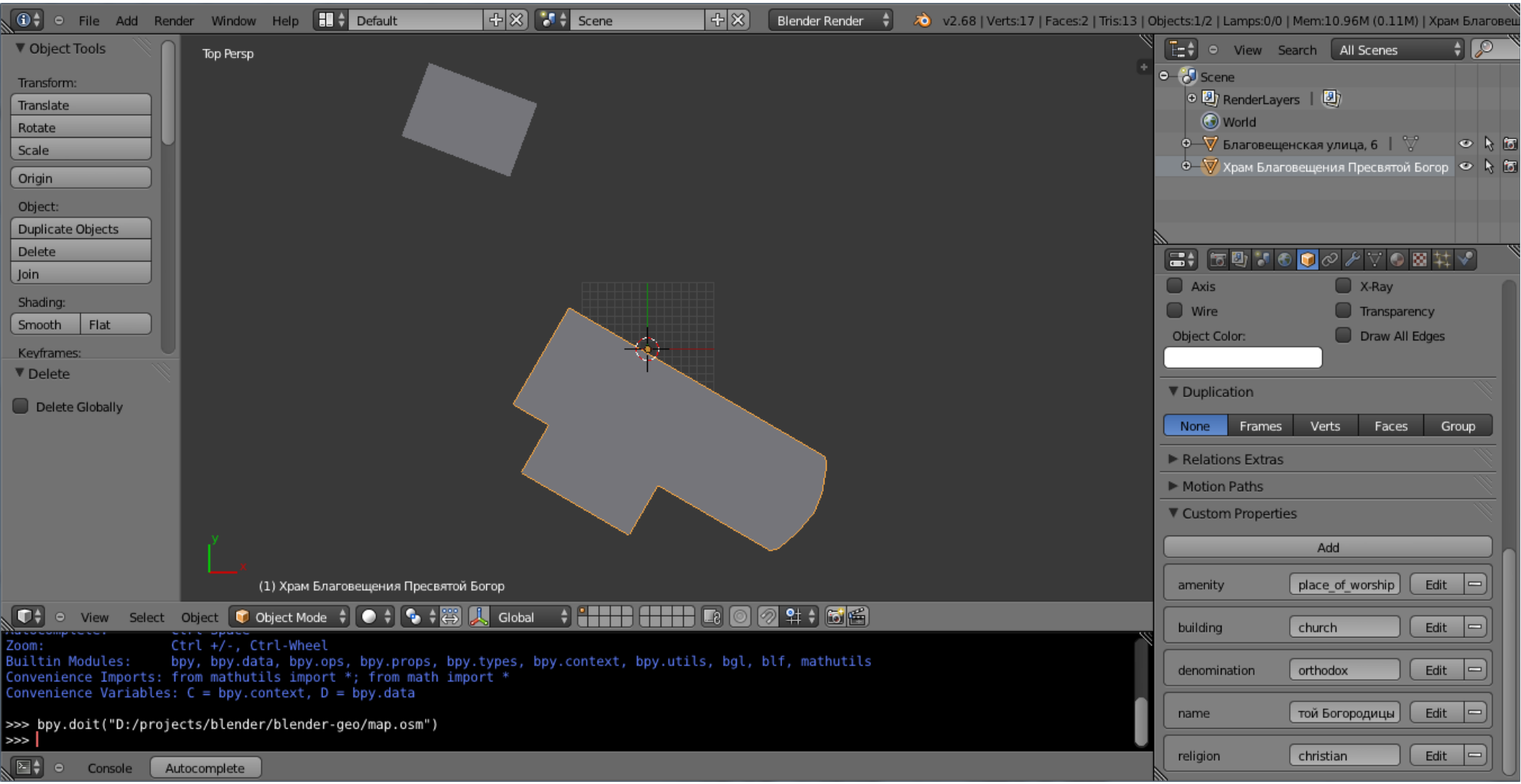
blender™

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)

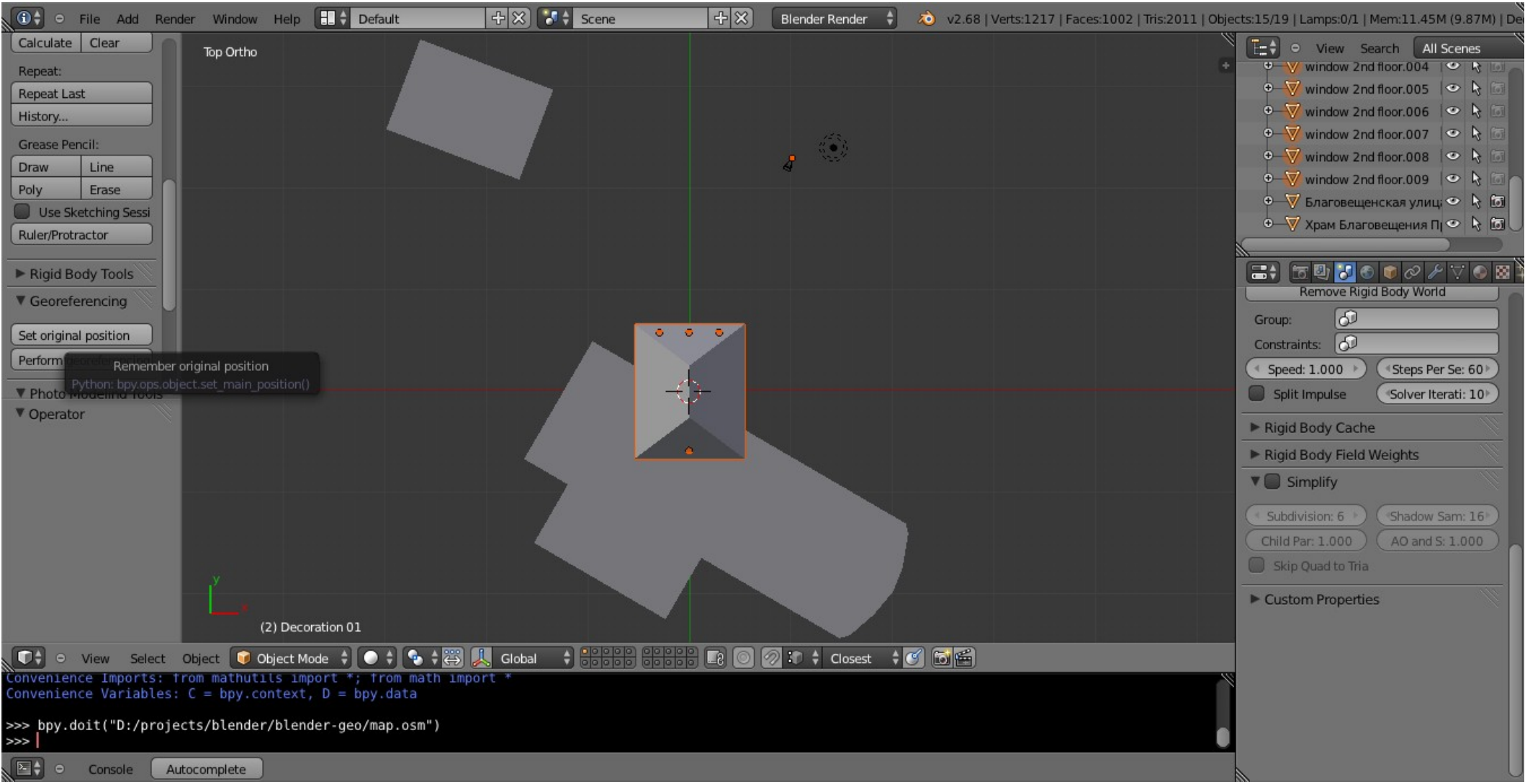
3D model

3D modeling: OSM import

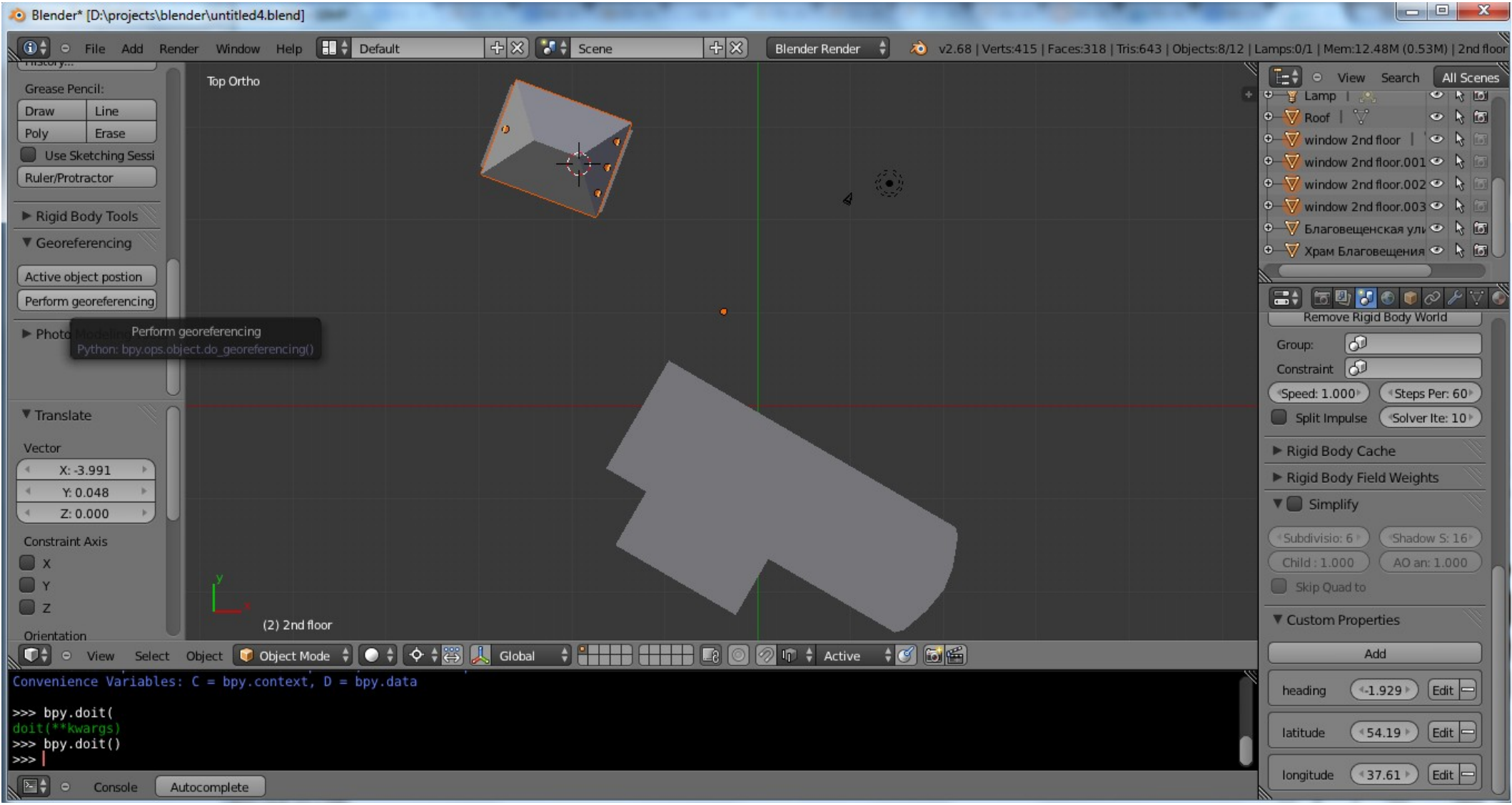


Transverse Mercator projection for the imported OSM Data

Georeferencing with OSM Data



Georeferencing with OSM Data

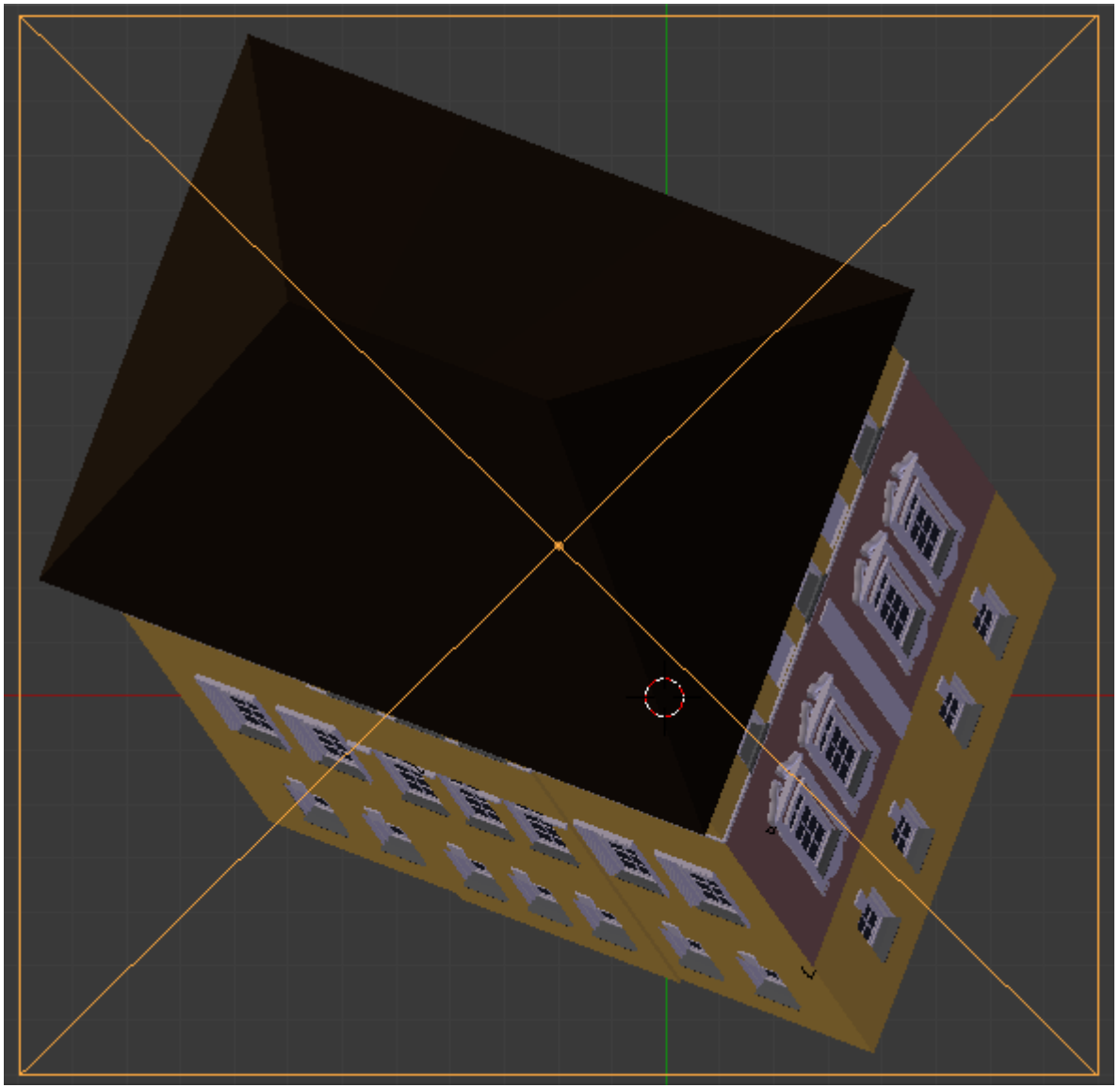


2.5D maps: 2 methods

- 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.

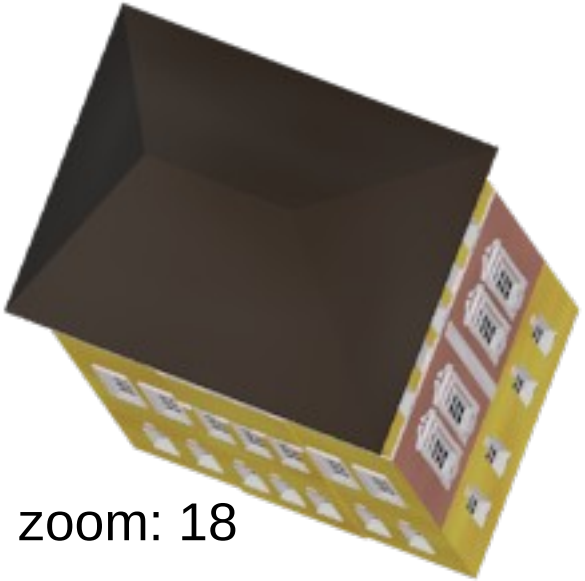
2.5D maps: separate images



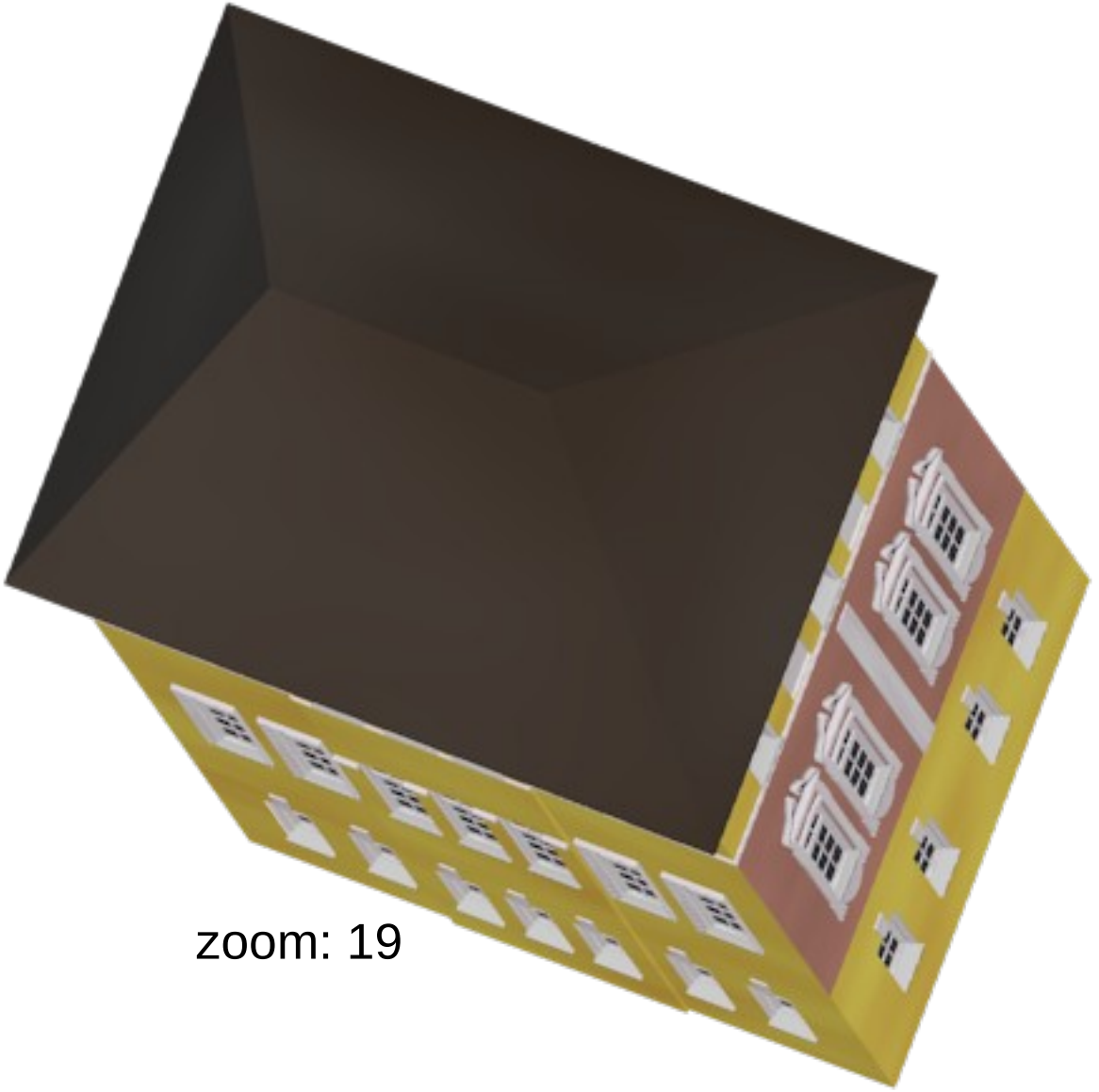
- Apply scaling 1/cos(lat) to take distortion of the spherical Mercator projection into account
- Simulate oblique projection in accordance with the given angles of projection
- Adjust the location of the orthographic camera and its viewport size
- Adjust the size of the resulting image. It depends on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

2.5D maps: separate images
Output: images

zoom: 17



zoom: 18



zoom: 19

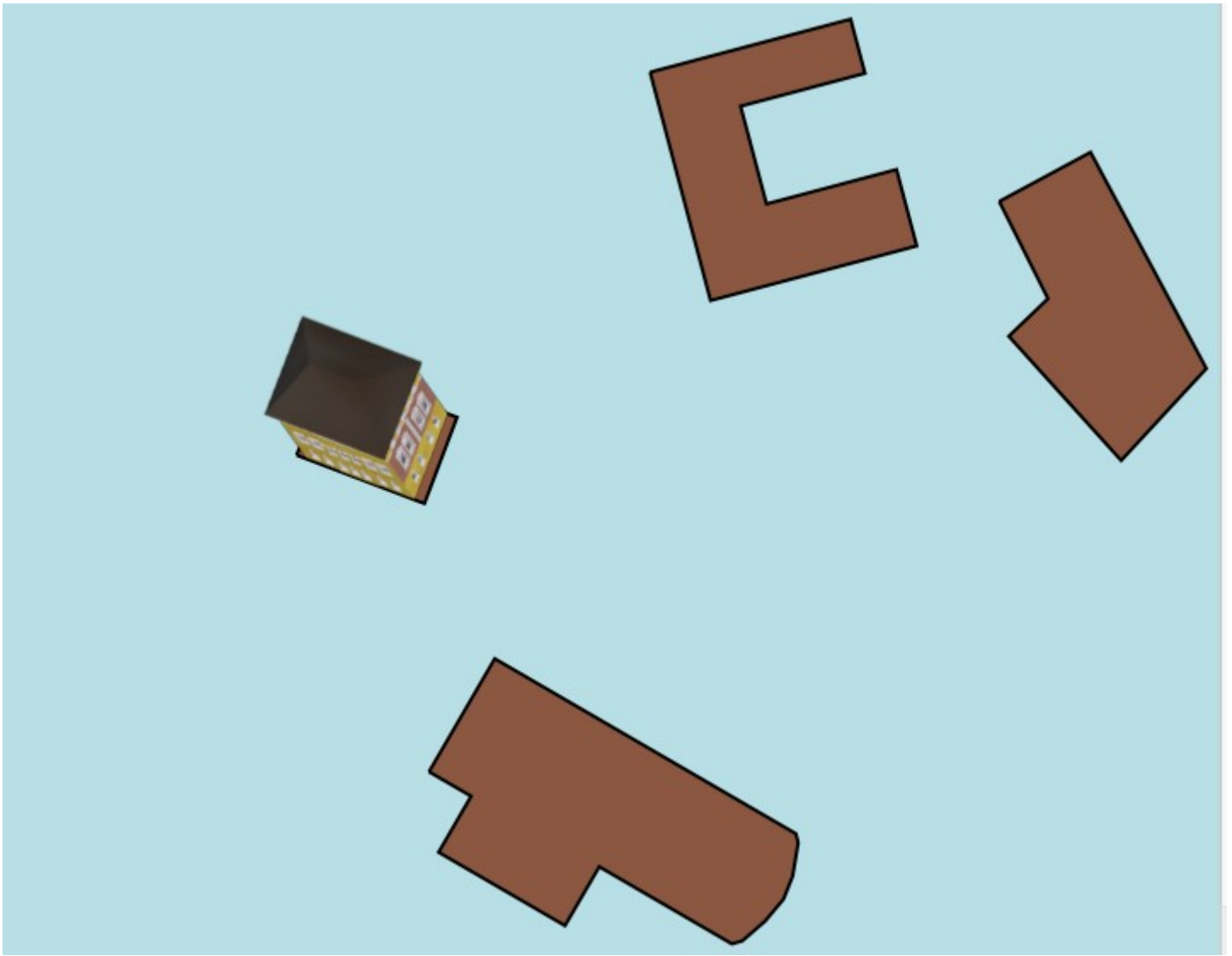
2.5D maps: separate images
Output: .csv file

- 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

```
.models {
  [zoom=17] {
    point-file: url("models/[image_z17].png");
    point-transform: translate([dx_z17],[dy_z17]);
  }
  [zoom=18] {
    point-file: url("models/[image_z18].png");
    point-transform: translate([dx_z18],[dy_z18]);
  }
  [zoom=19] {
    point-file: url("models/[image_z19].png");
    point-transform: translate([dx_z19],[dy_z19]);
  }
}
```

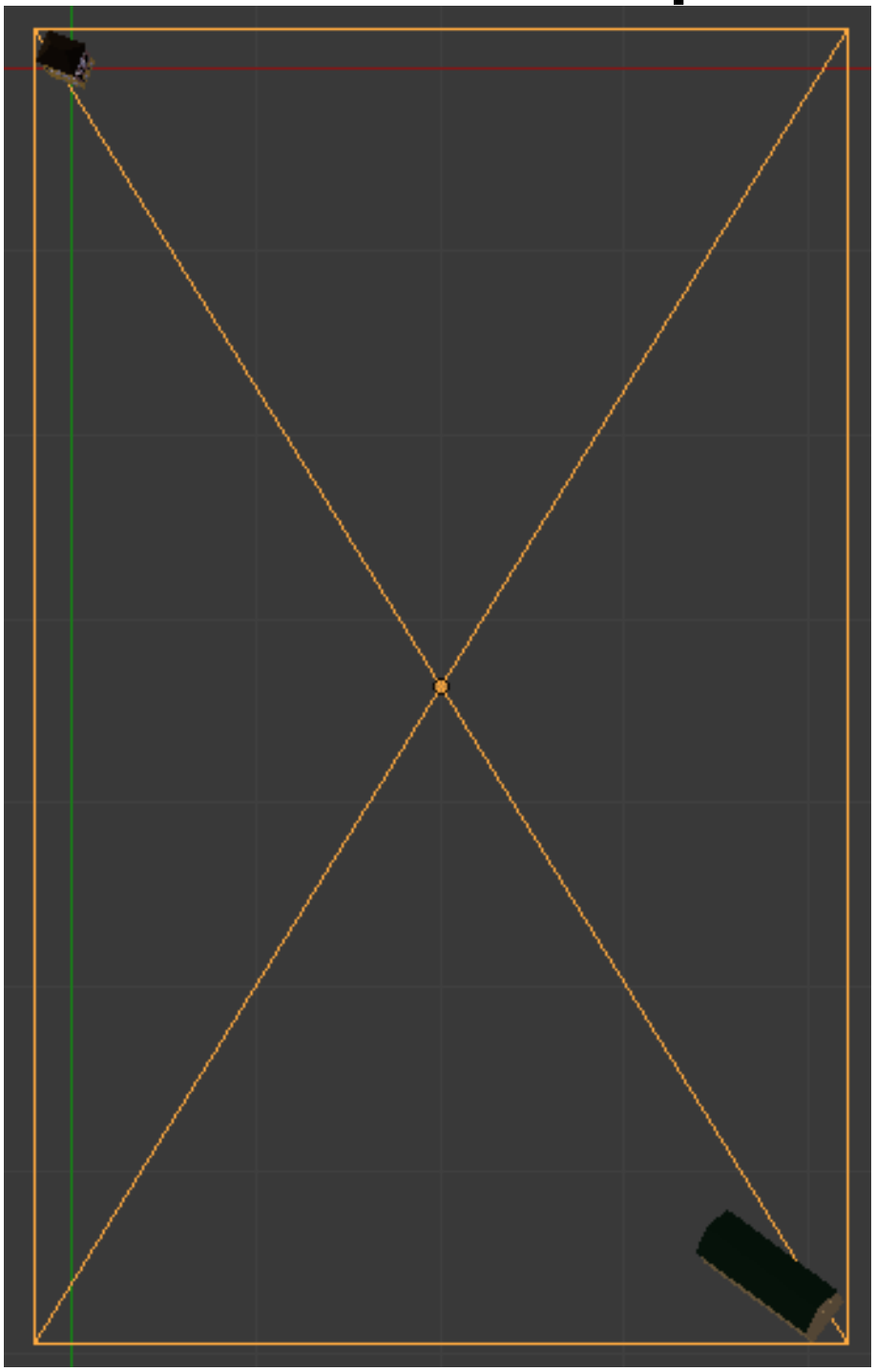
2.5D maps: separate images



2.5D maps: separate images Disadvantage

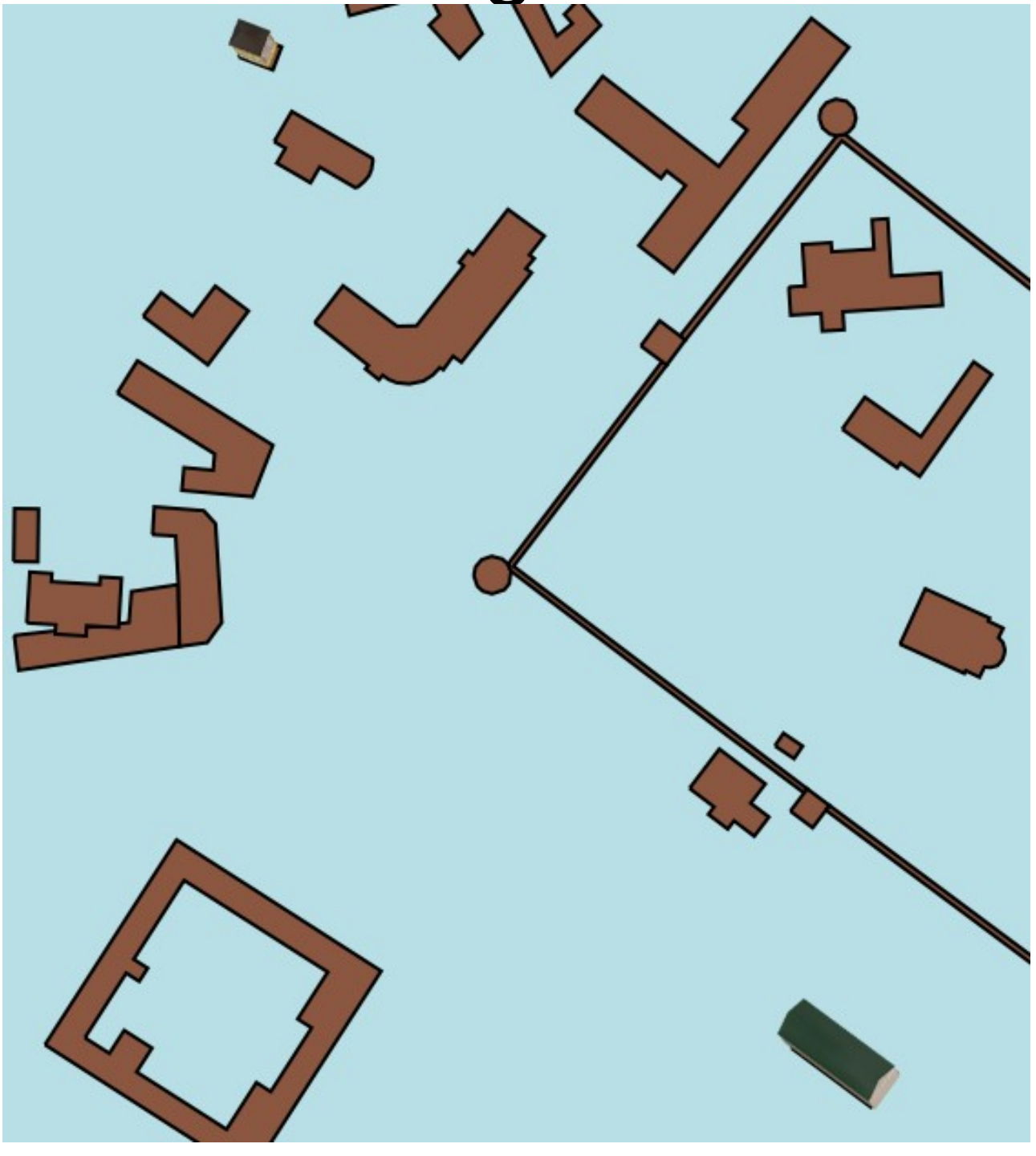
Buildings should not stay too close to each other!

2.5D maps: common images



- Apply scaling 1/cos(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

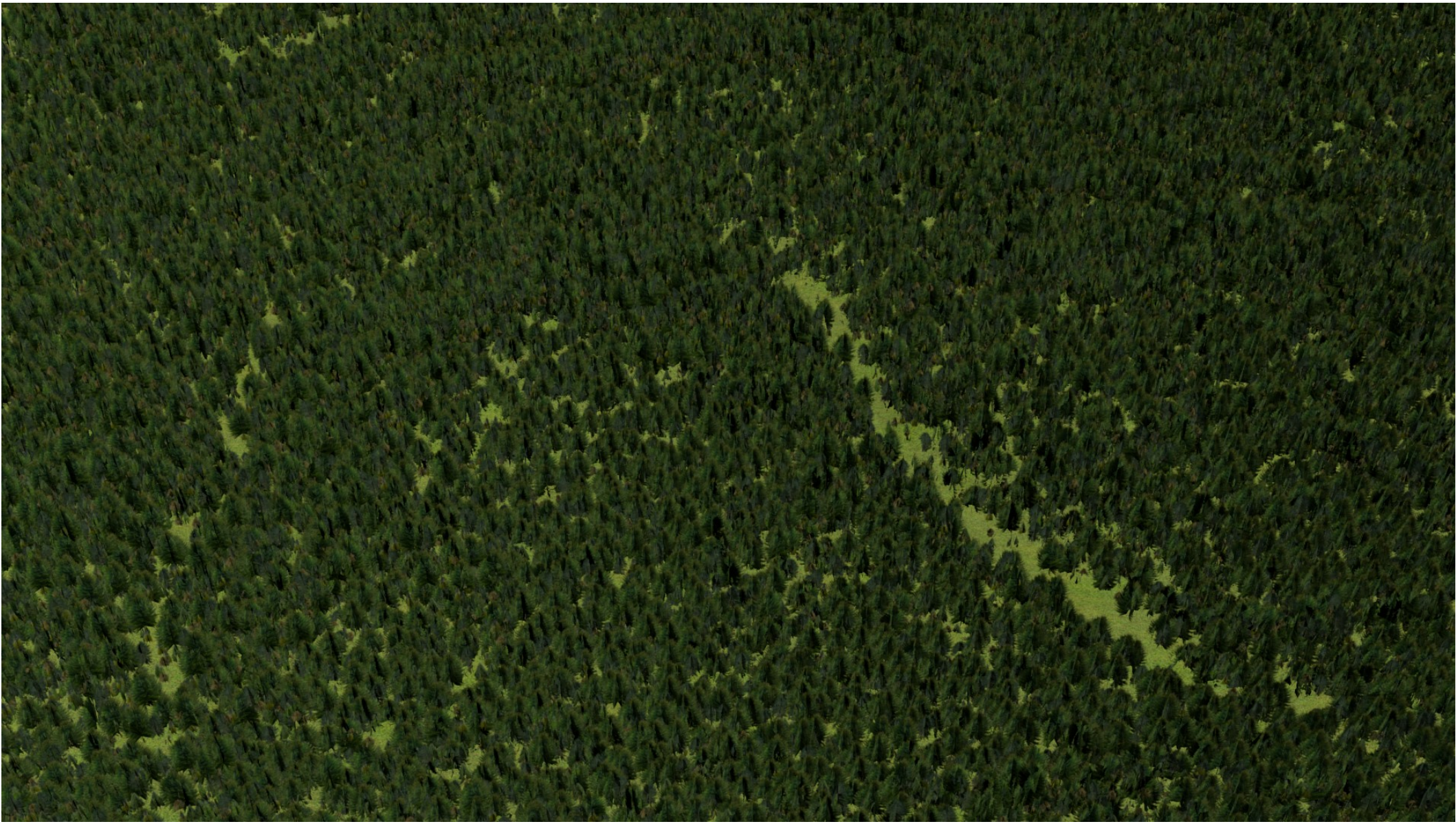
2.5D Karten: das gemeinsame Bild



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

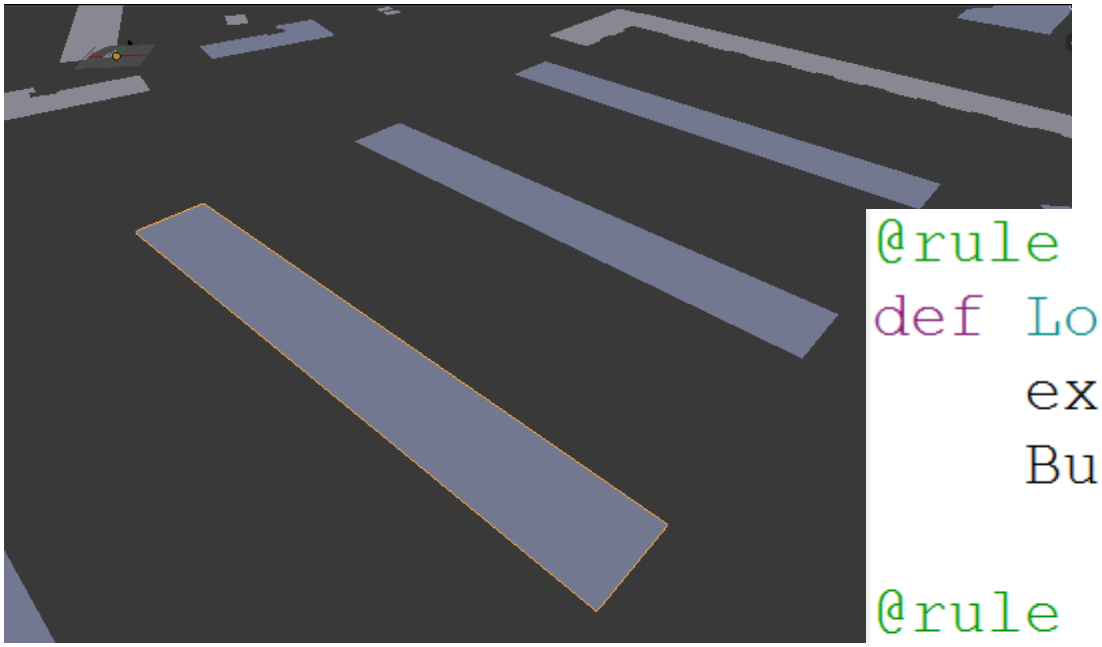
■ CGA Shape Grammar was developed in the Swiss Federal Institute of Technology in Zurich (ETH Zurich)

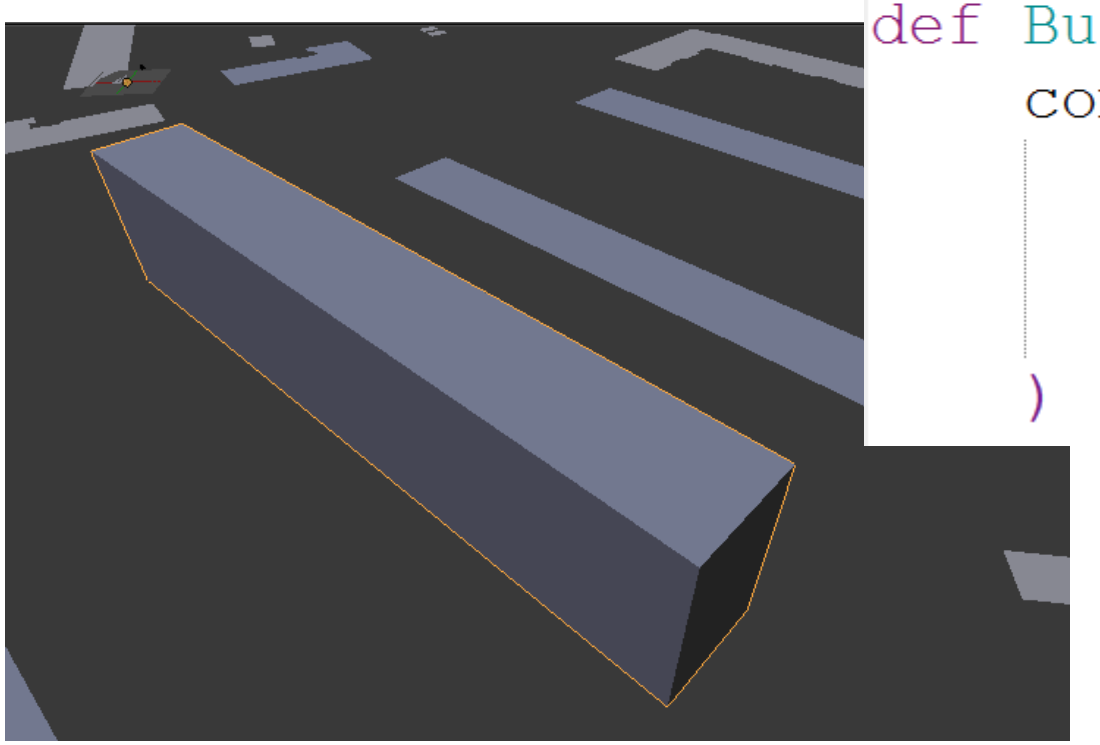
• CGA Shape Grammar makes up the foundation of ESRI CityEngine software

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

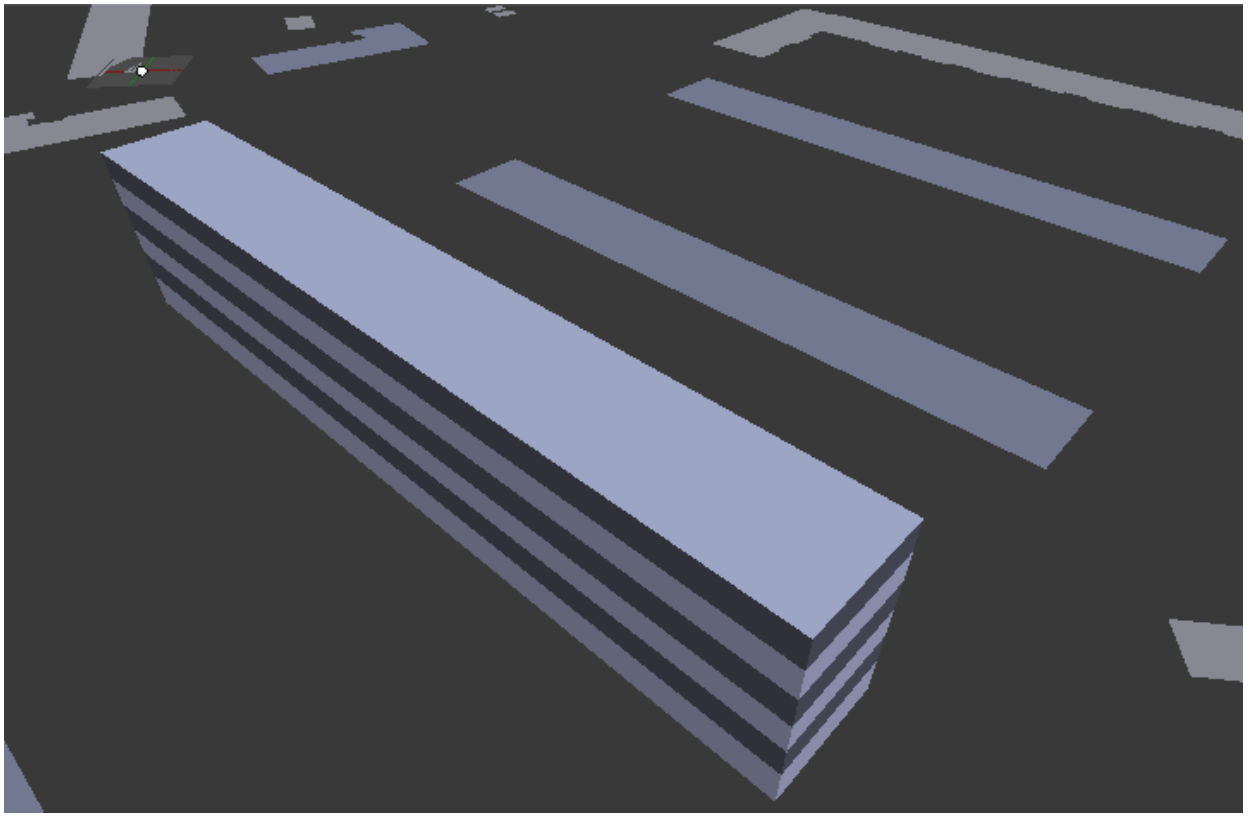
CGA Shape Grammar: Example





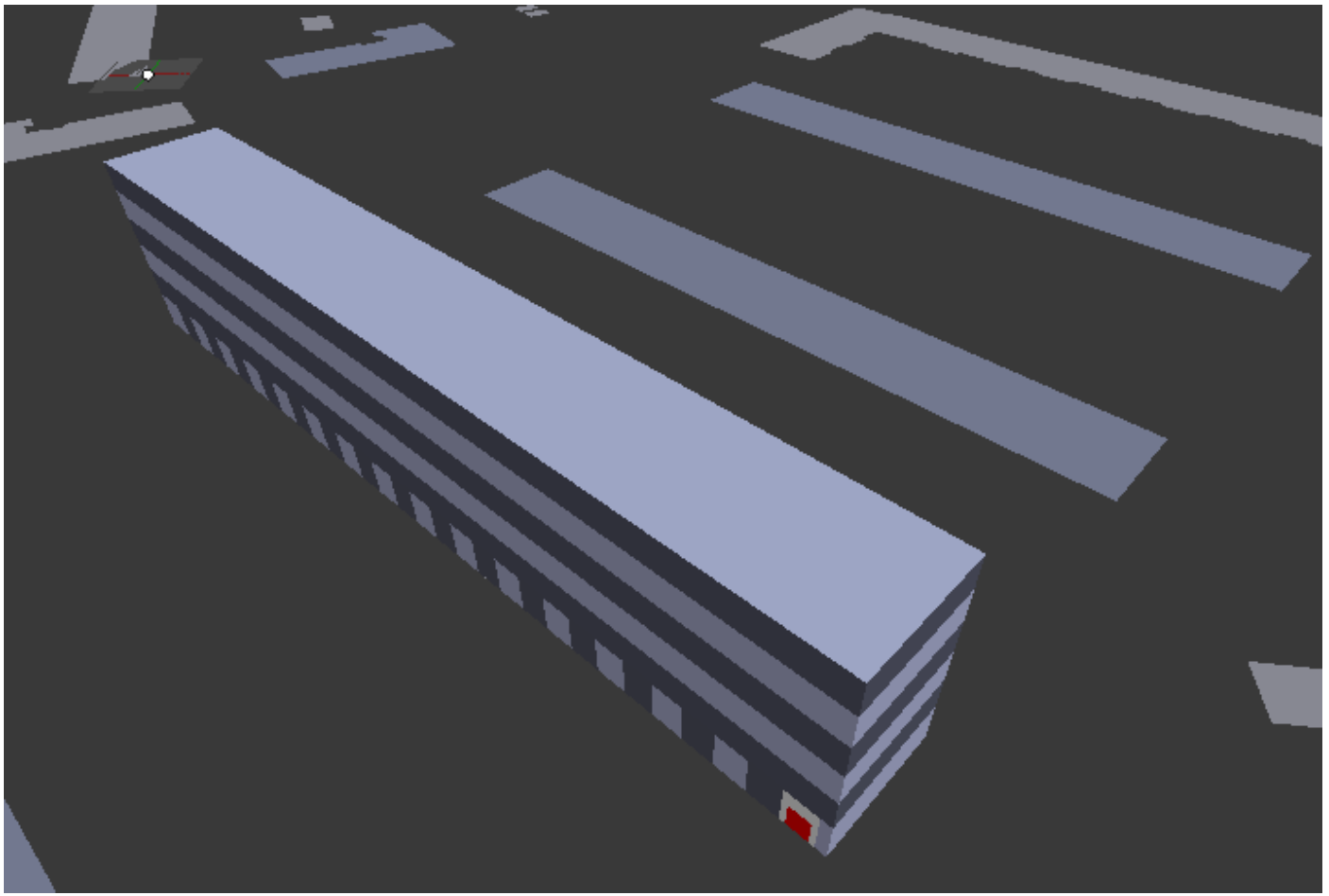
```
@rule
def Lot():
    extrude(height)
    Building()

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



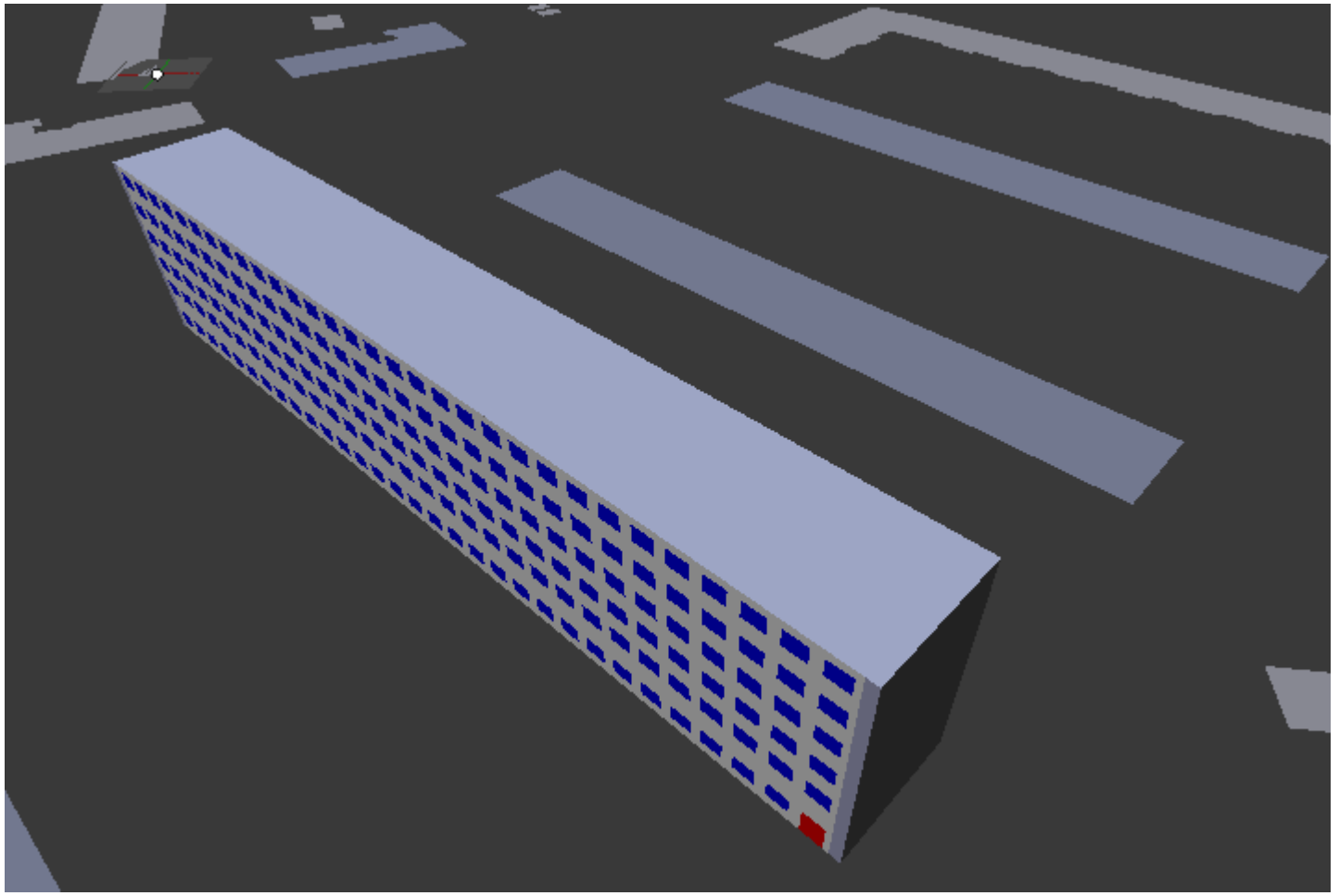
```
@rule
def FrontFacade():
    split(y).into(
        groundFloorHeight>>GroundFloor() |
        repeat(flt(floorHeight)>>Floor())
    )

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

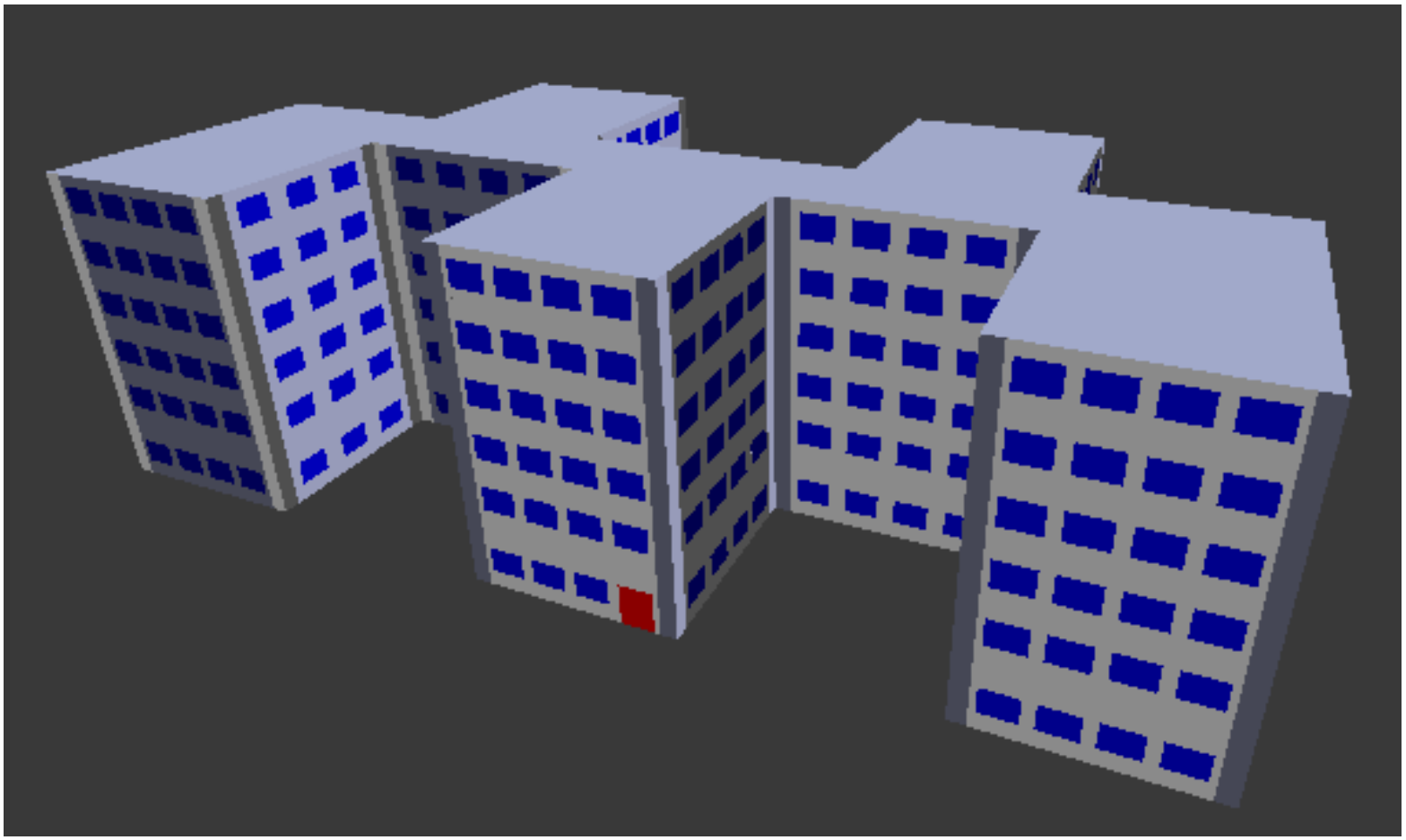


```
@rule
def GroundFloor():
    split(x).into(
        1>>Wall() |
        repeat(flt(tileWidth)>>Tile()) |
        flt(tileWidth)>>EntranceTile() |
        1>>Wall()
    )
```

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