

Blender-basierte Erstellung von 3D-Modellen von Orten mit Hilfe von OSM-Daten

Vladimir Elistratov

github.com/vvoovv

vladimir.elistratov@gmail.com

Motivation: Karten von Ruben Atojan



Karte von Altstadt Lembergs. Herausgeber: M. Tschumak

<http://www.reykjavikcentermap.com>

Reykjavik Center Map



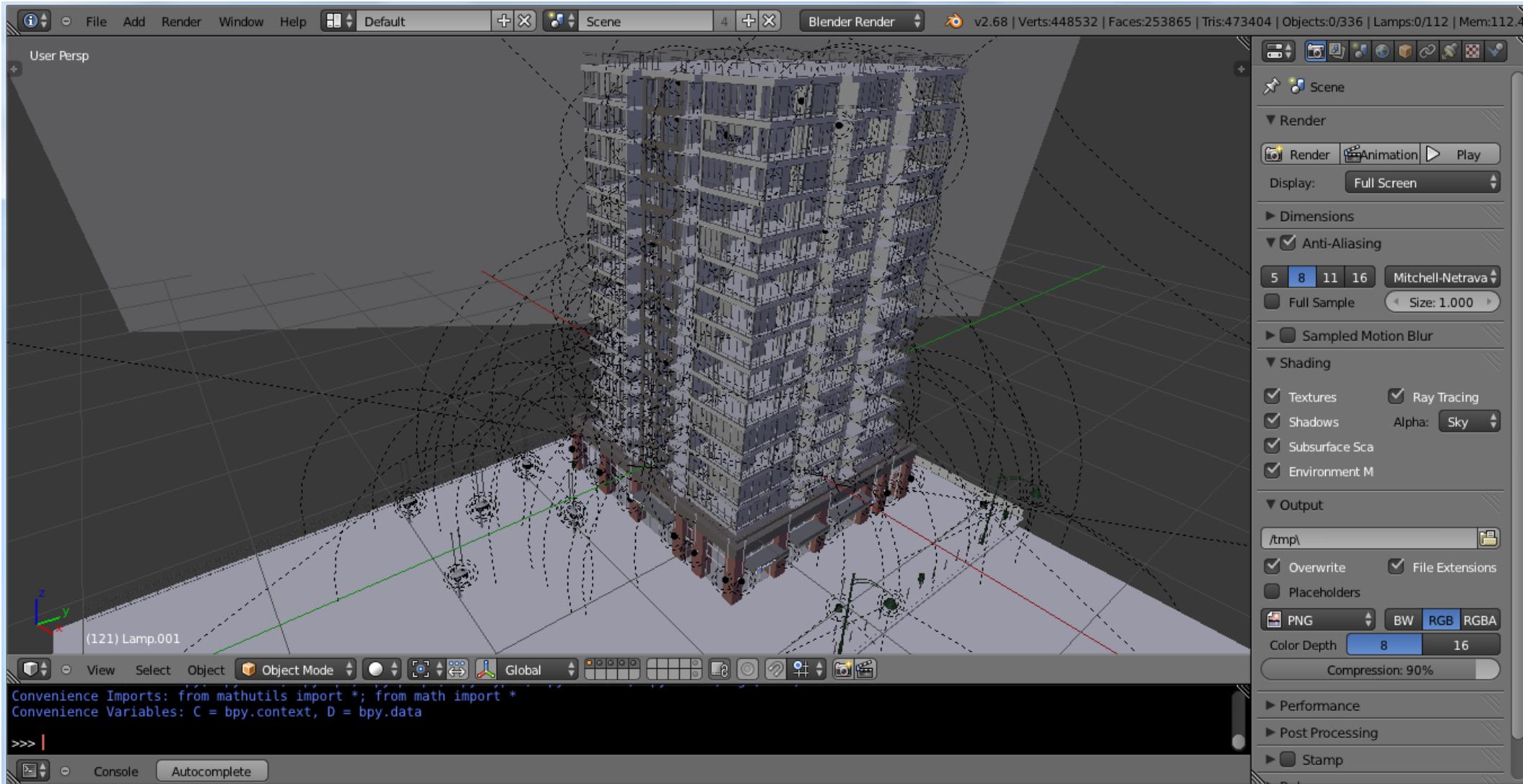
Beispiel von 2.5D Karten: <http://2gis.ru>





<http://www.blender.org>

3D-Modellierung



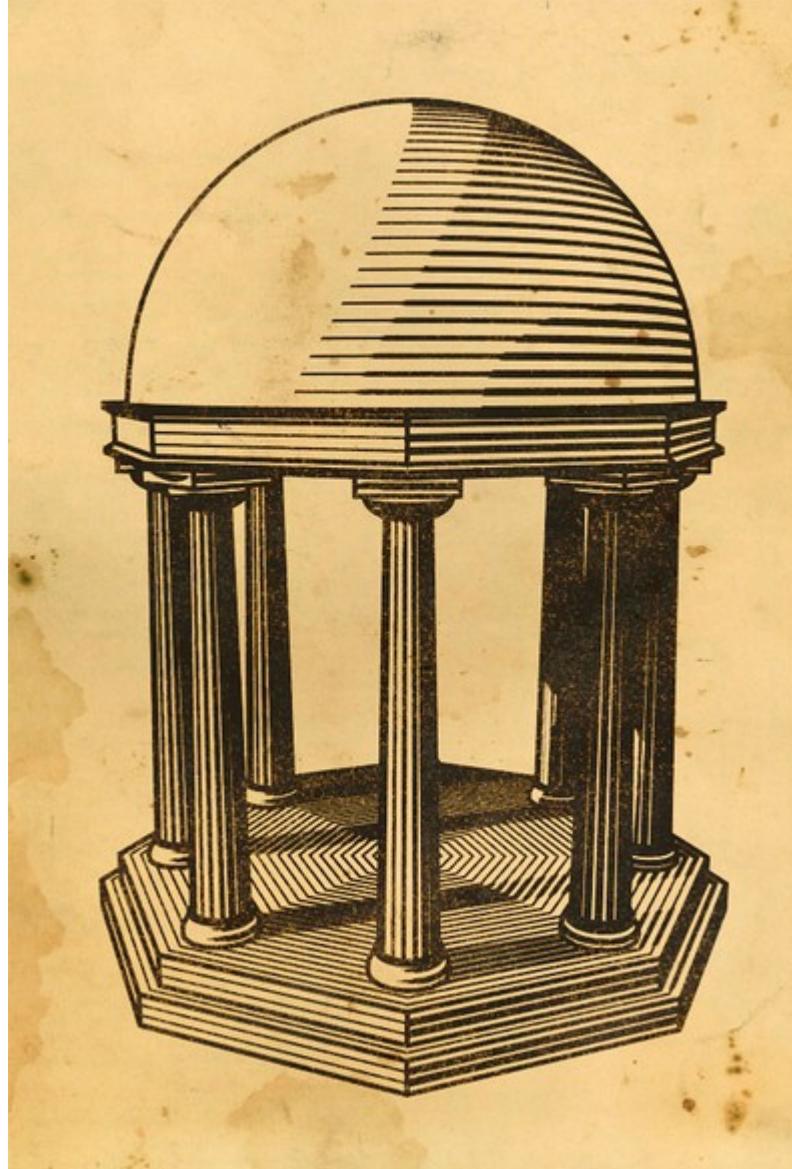
Photorealistisches Rendering



<http://www.blenderguru.com>



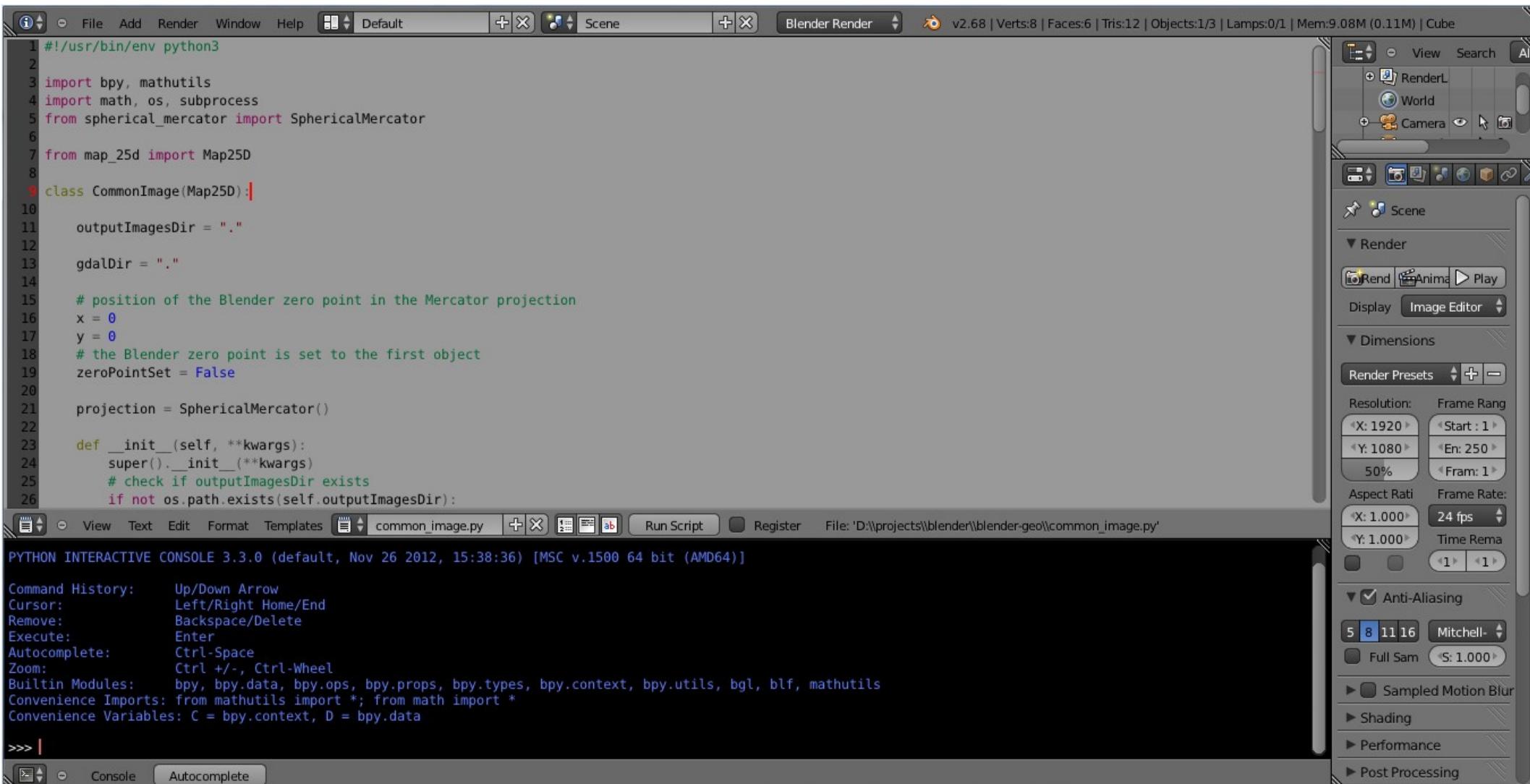
Nicht-Photorealistisches Rendering



Charblaze CC-BY



Alles is mit Python abrufbar



The screenshot shows the Blender 2.68 interface with the Python Script Editor open. The code in the editor is a Python script named 'common_image.py' which defines a class 'CommonImage' that inherits from 'Map25D'. The script sets up a projection and checks if an output directory exists. The Python Interactive Console at the bottom shows the command history and various key mappings.

```
1 #!/usr/bin/env python
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

>>> |



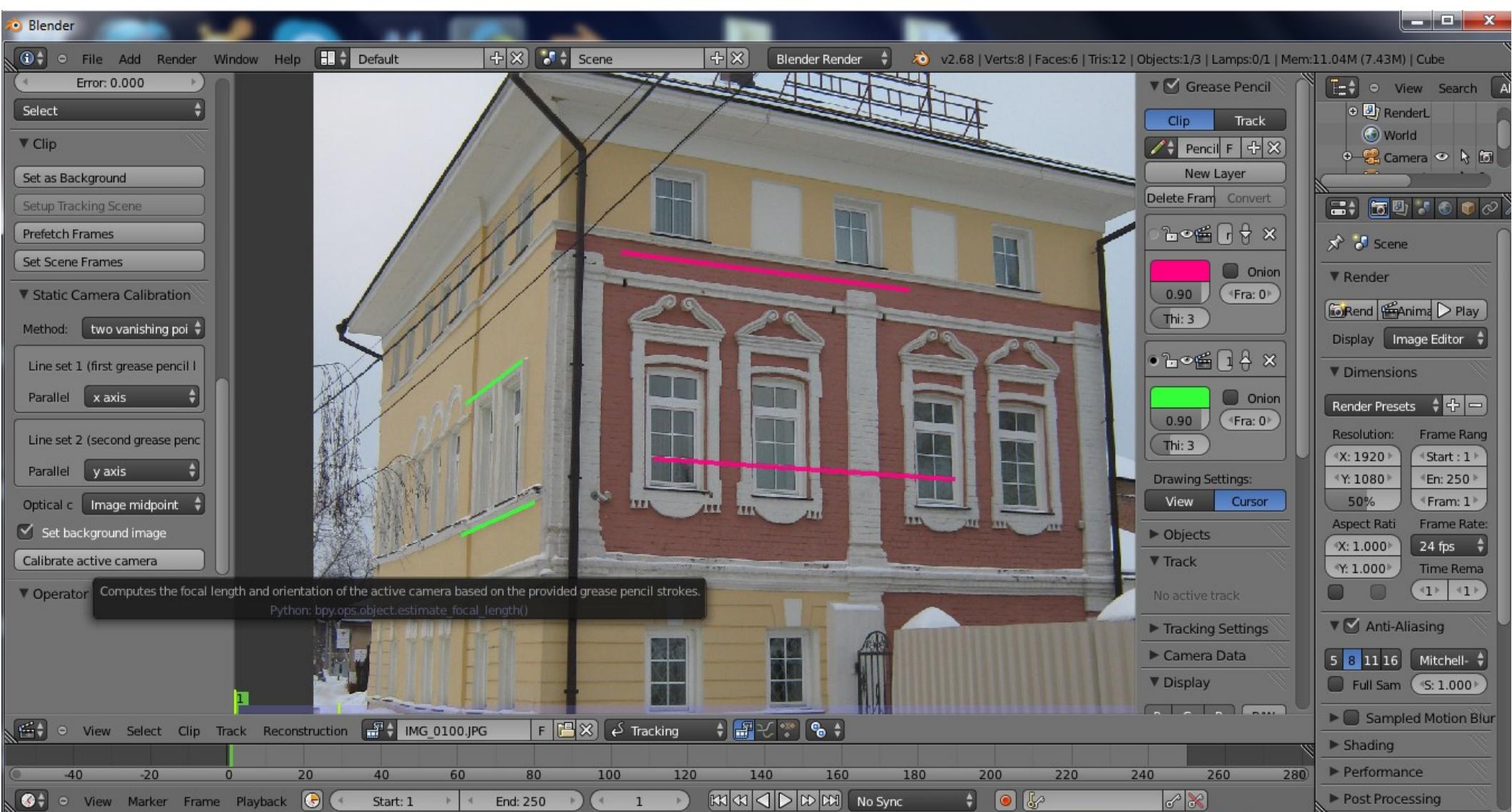
Vorgehen

- Aufnahme von Gebäuden
- 3D-Modellierung mit Hilfe von Photos
- Georeferenzierung von jedem 3D-Modell
- Zusammenstellung von einzelnen georeferenzierten 3D-Modellen je nach Anwendung (Beispiel: 2.5D Karten)

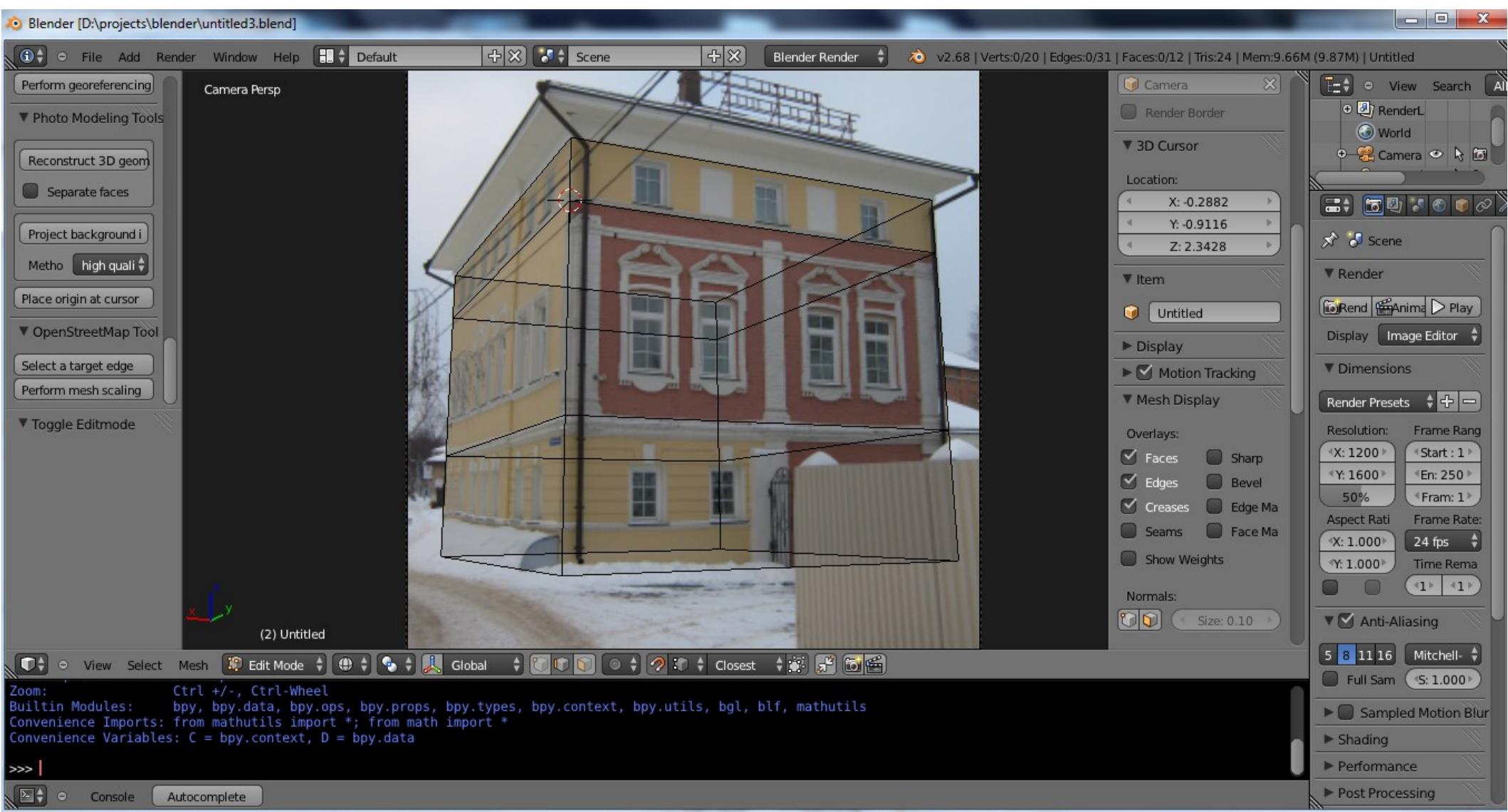
Aufnahme von Gebäuden



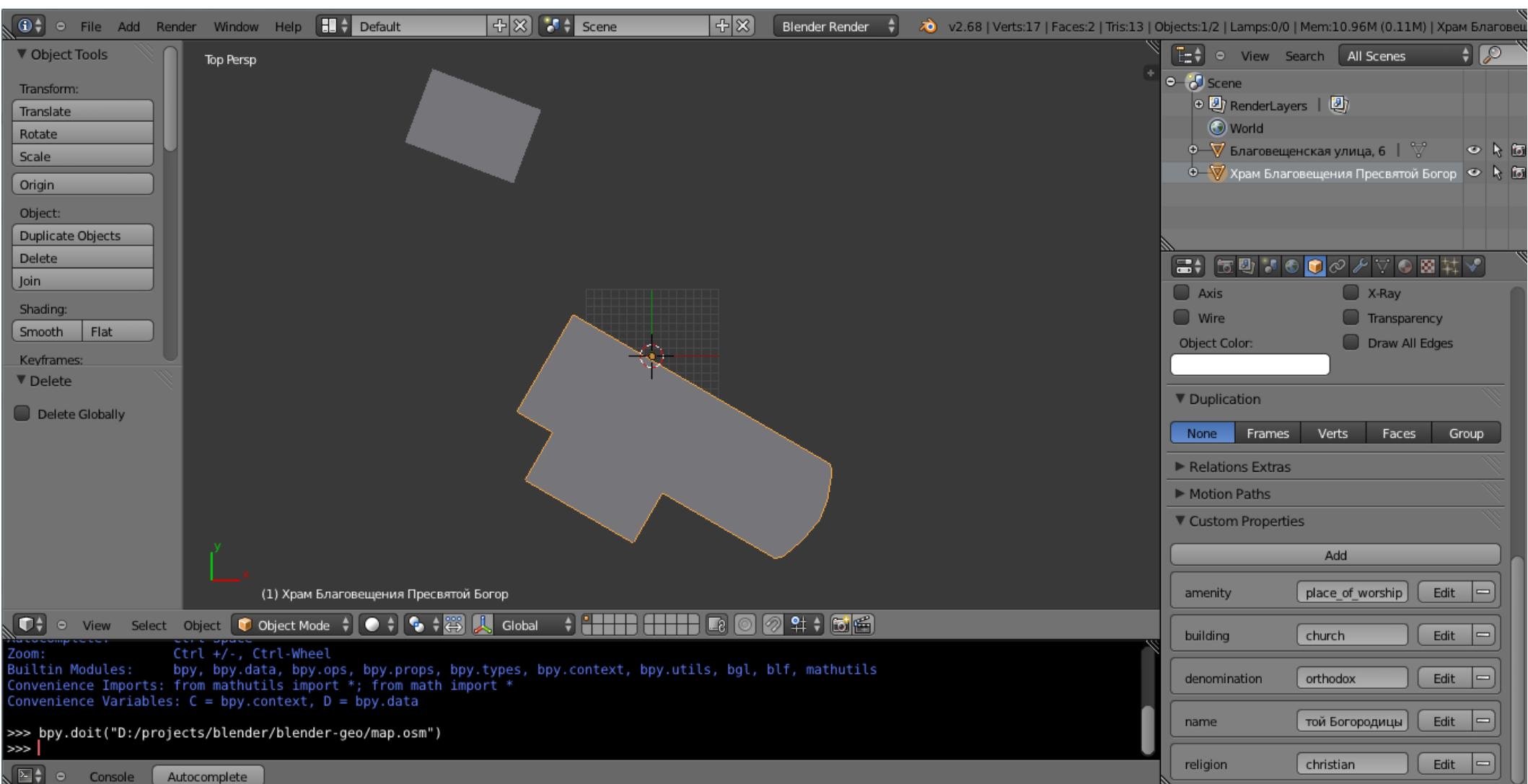
3D-Modellierung mit Hilfe von Photos



3D-Modellierung mit Hilfe von Photos

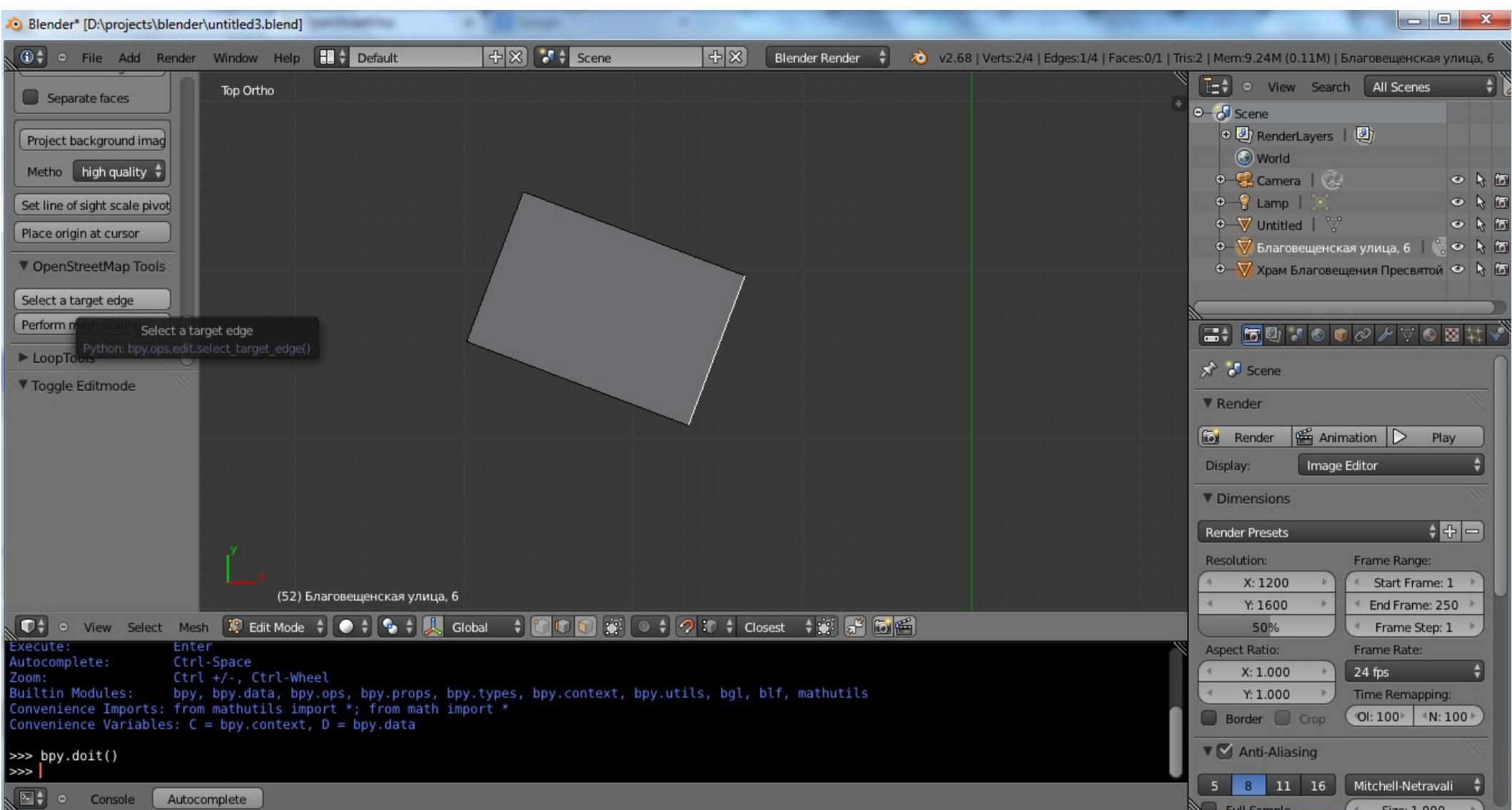


3D-Modellierung: OSM Import

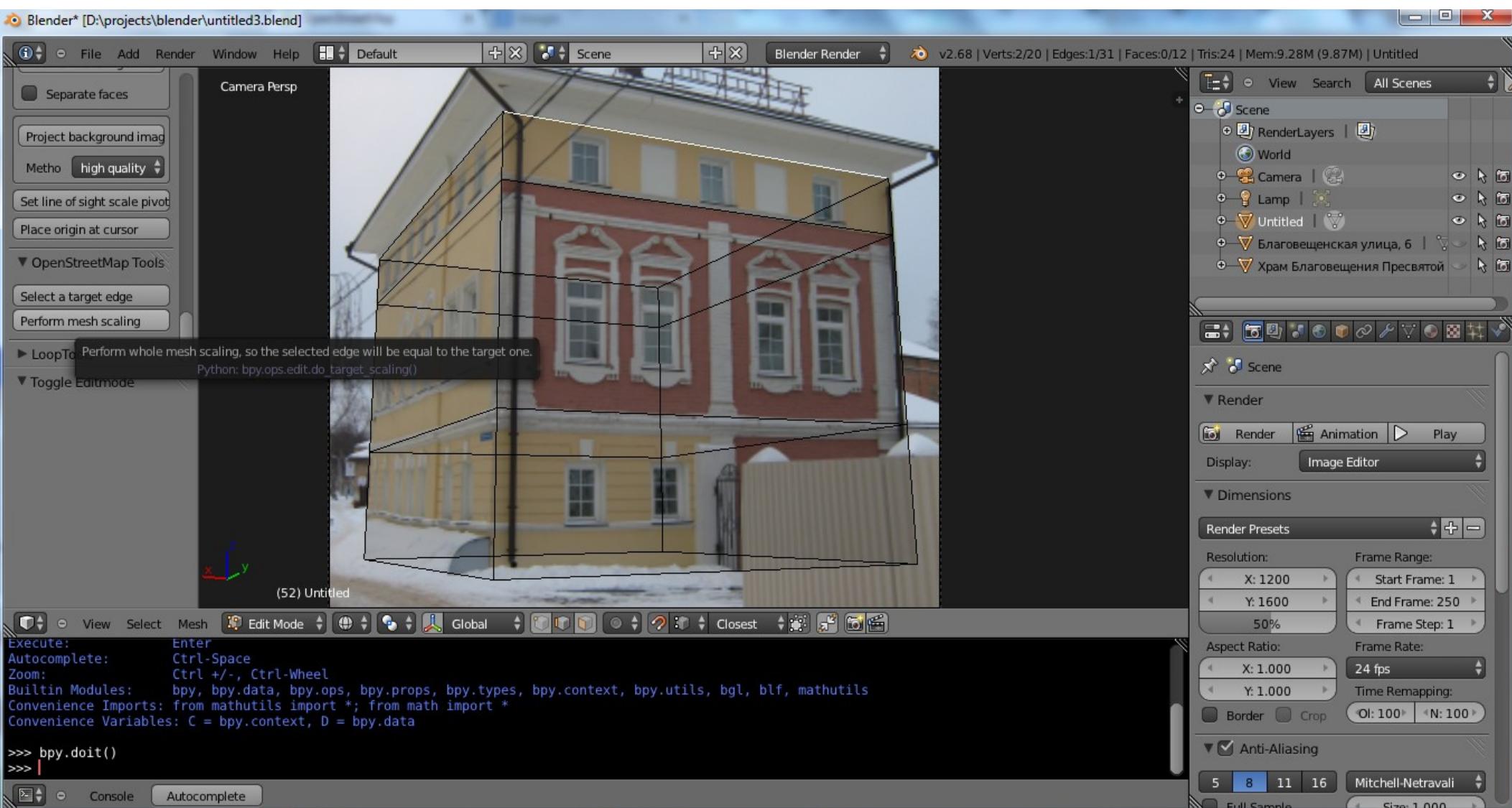


Transversale Mercator-Projektion für importierte OSM-Daten

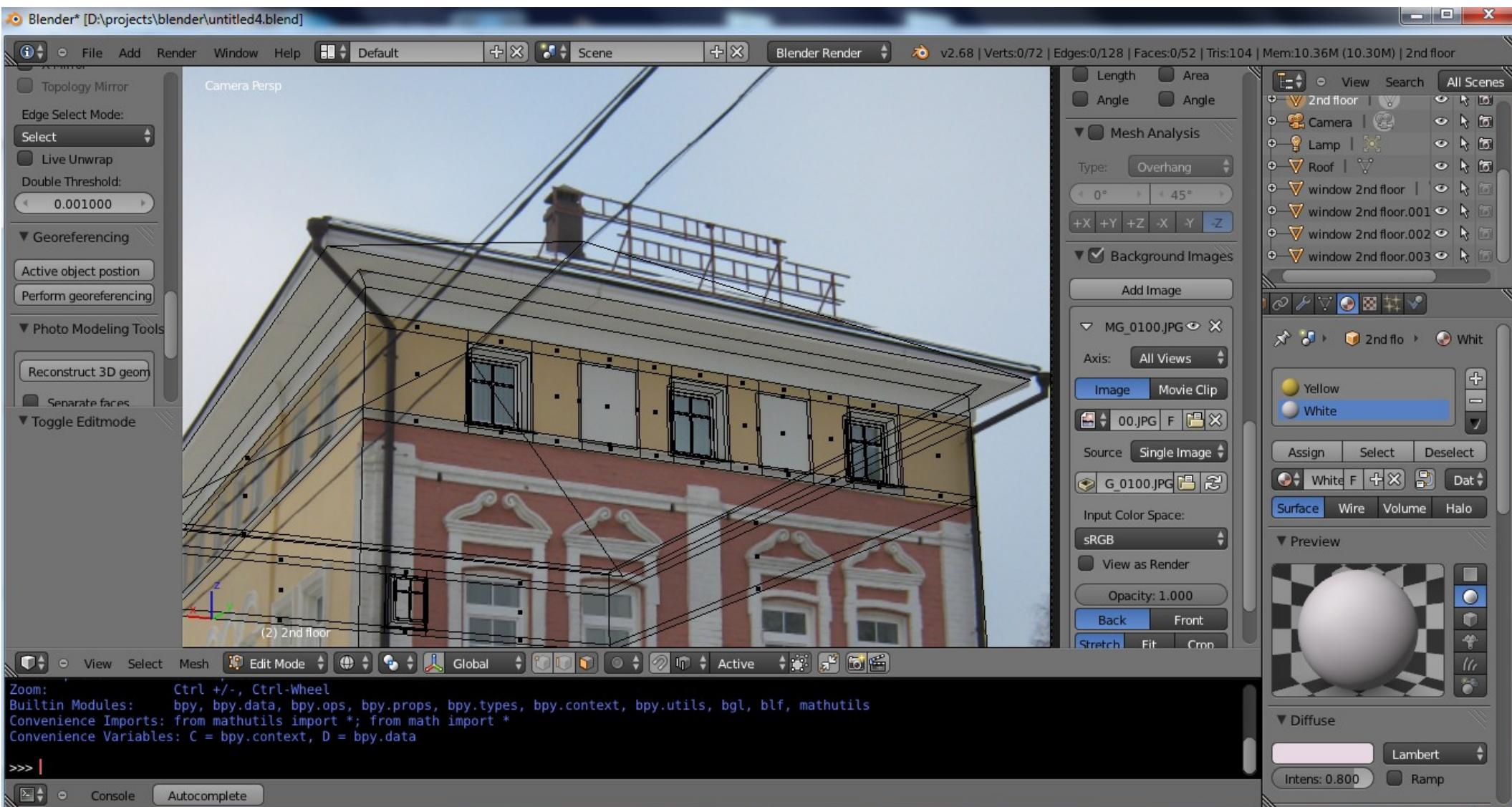
3D-Modellierung: richtige Größen mit OSM



3D-Modellierung: richtige Größen mit OSM



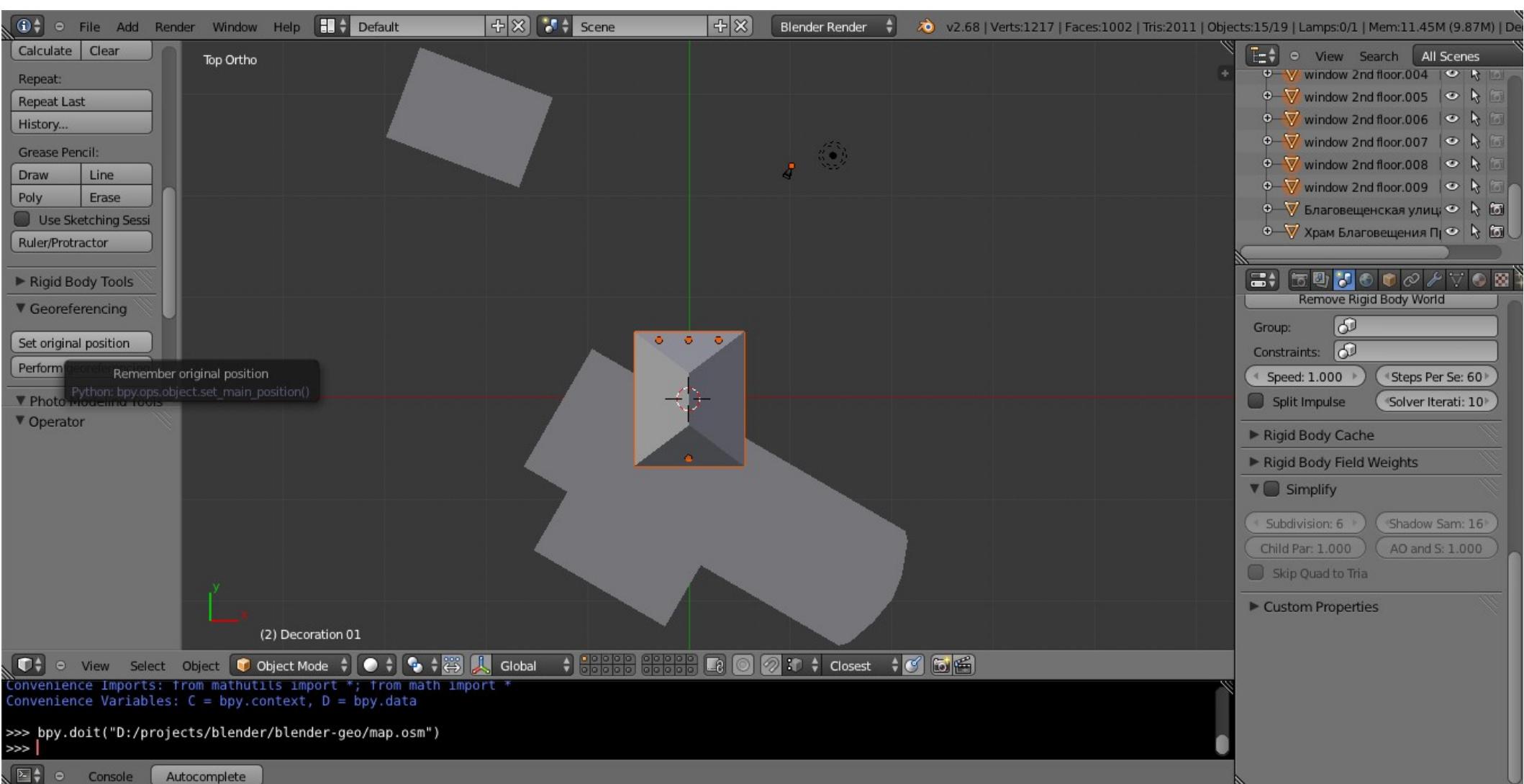
3D-Modellierung: das Gebäude ist fertig!



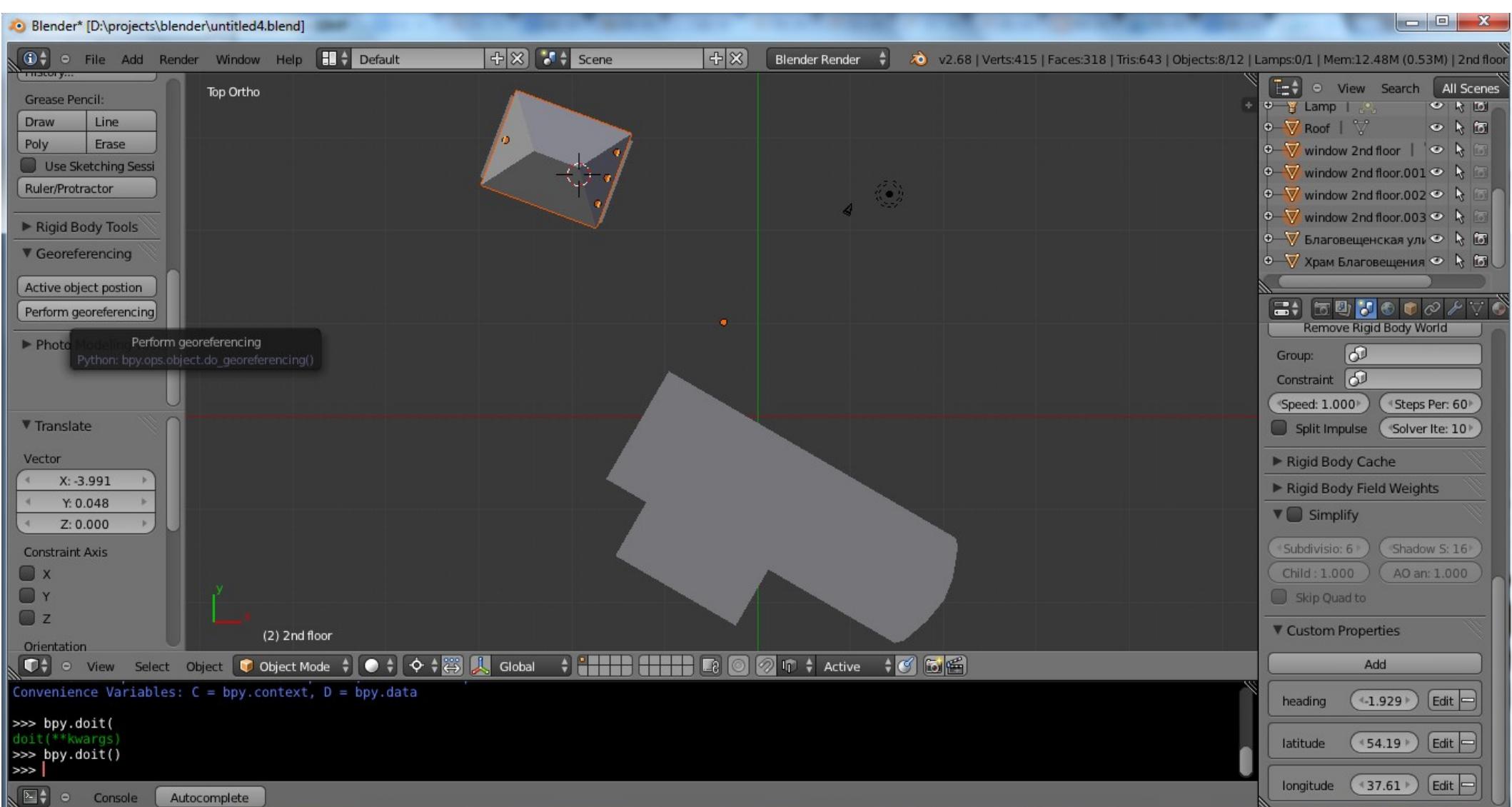
3D-Modellierung: das Gebäude ist fertig!



Georeferenzierung mit Hilfe von OSM



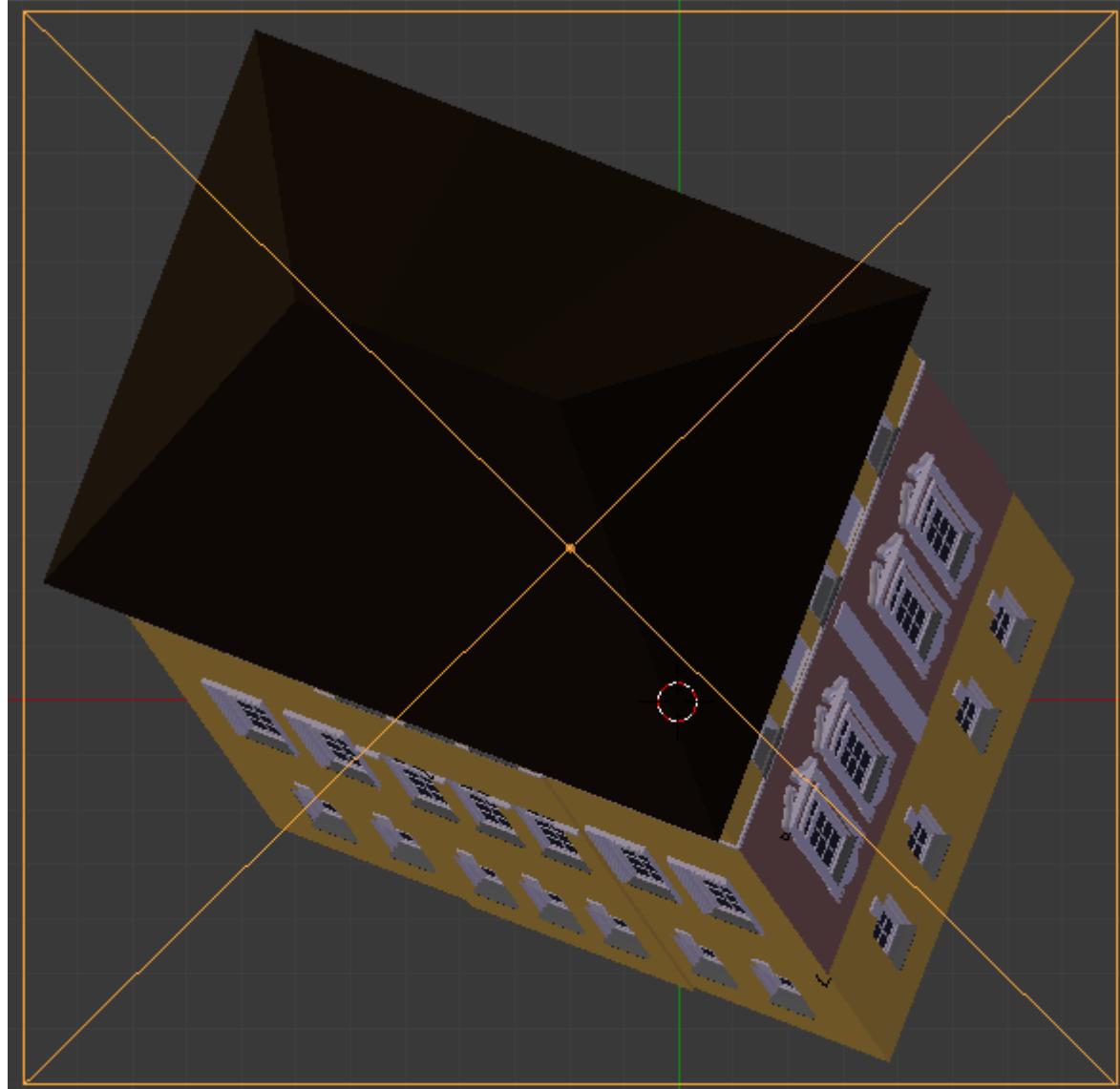
Georeferenzierung mit Hilfe von OSM



2.5D Karten: 2 Methoden

- Jedes Gebäude getrennt rendern und mit Mapnik PointSymbolizer einzelne Bilder zusammenstellen
- Alle Gebäude auf demselben Bild rendern und als Raster Ebene im Mapnik anwenden

2.5D Karten: einzelne Bilder



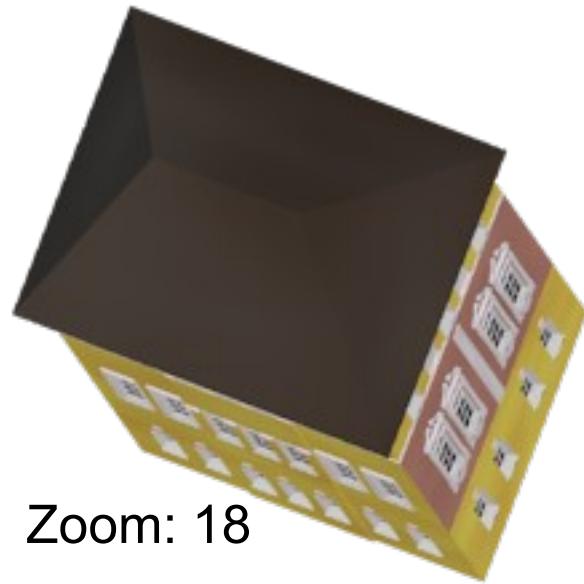
- Skalierung $1/\cos(\text{Breite})$ anwenden
- Schrägprojektion nach den gewählten Winkeln simulieren
- Die Viewport-Größen und Lage der orthographischen Kamera anpassen
- Die Ergebnisbildgrößen anpassen: die hängen von Zoom ab

2.5D Karten: einzelne Bilder

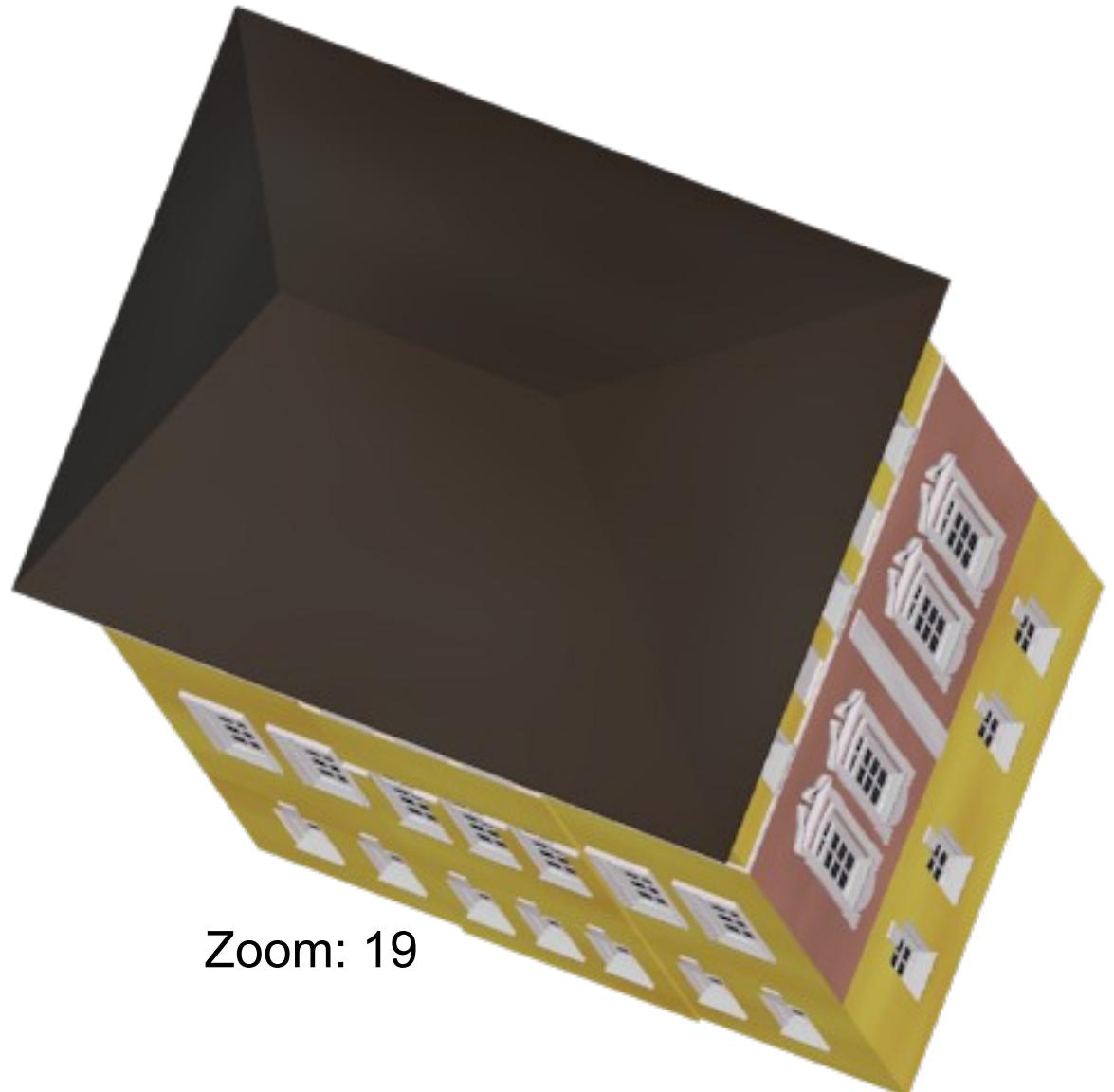
Das Output: Bilder



Zoom: 17



Zoom: 18



Zoom: 19

2.5D Karten: einzelne Bilder

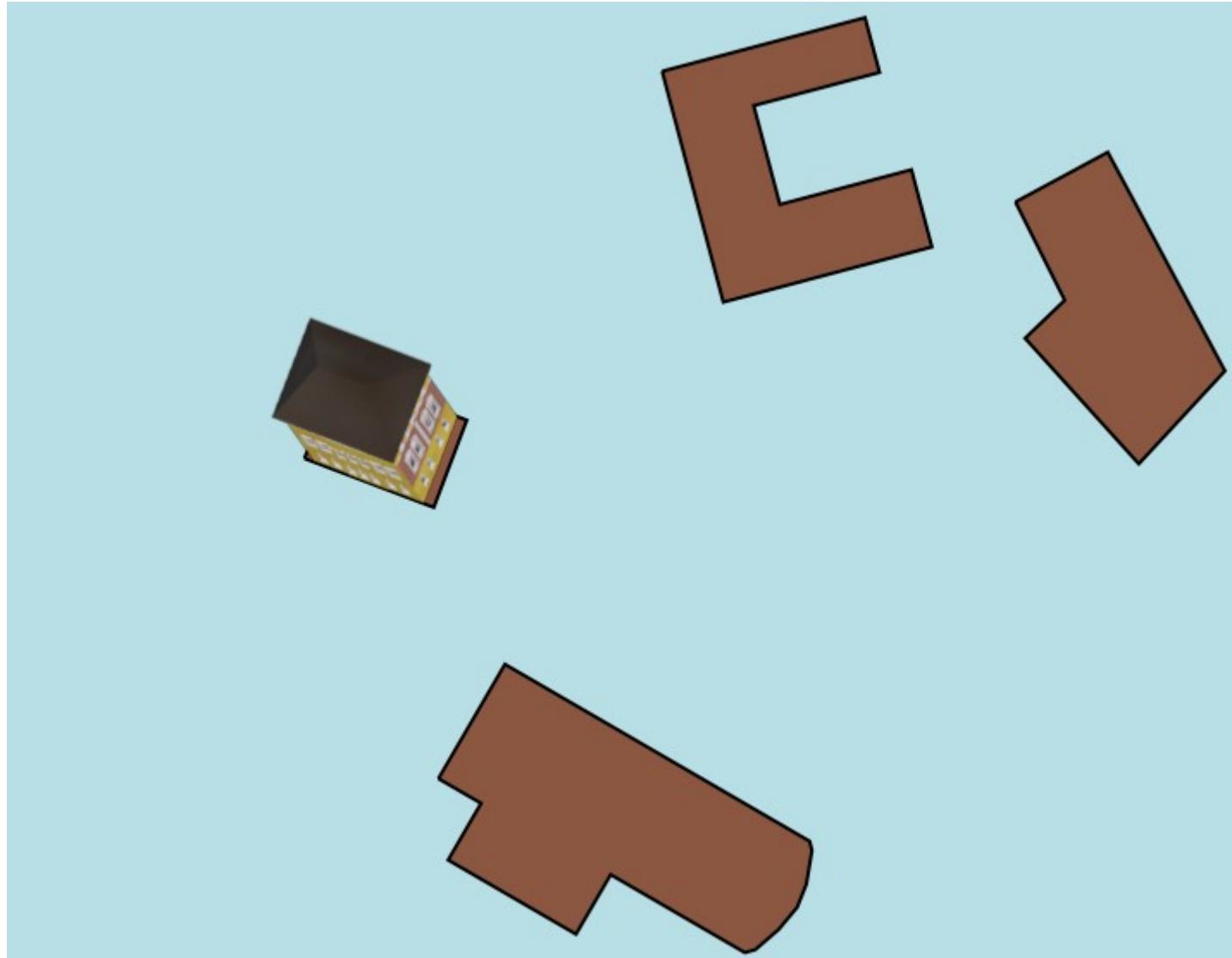
Das Output: eine .csv Datei

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

2.5D Karten: einzelne Bilder CartoCSS .mss Datei

```
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 }
```

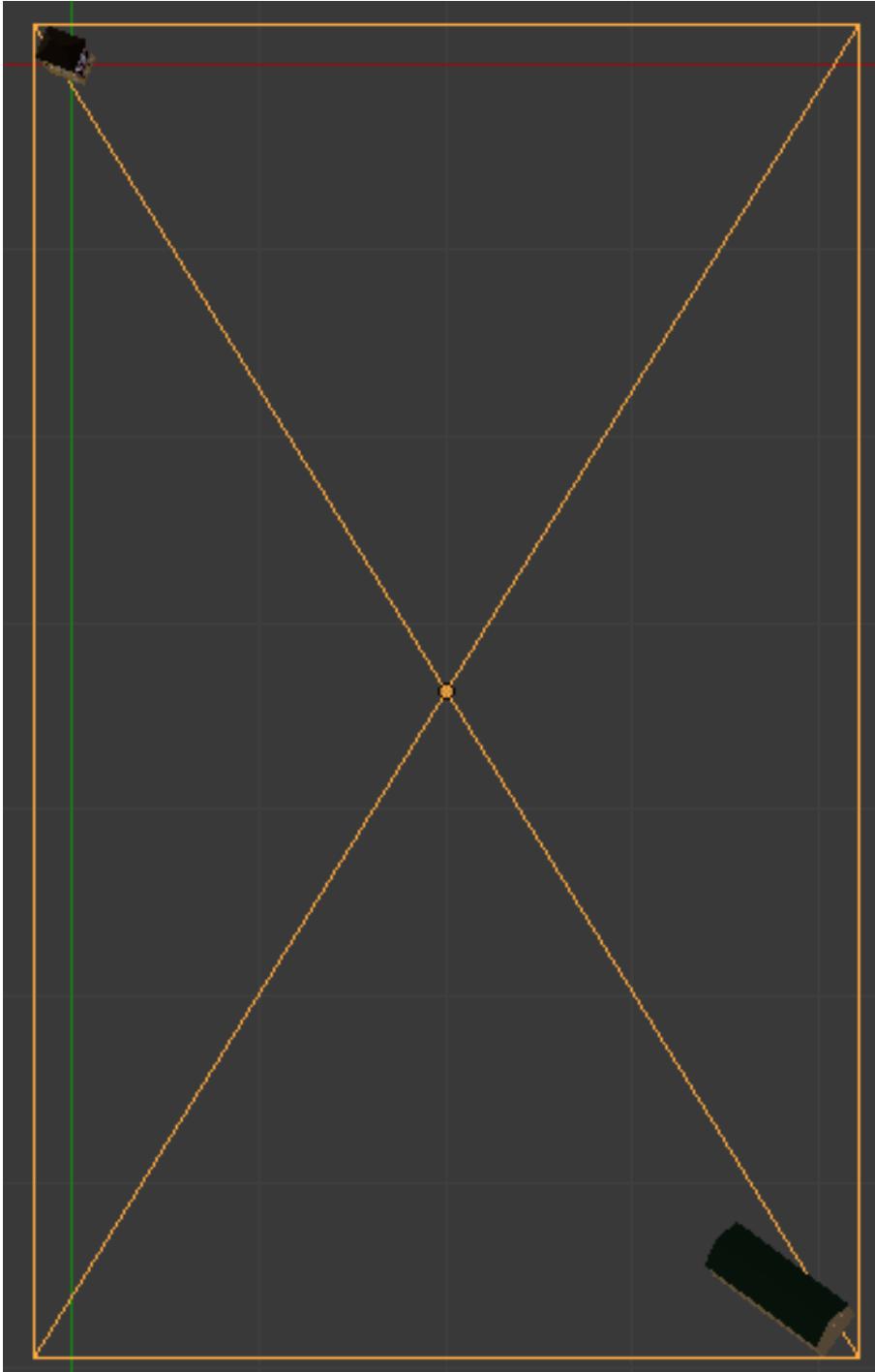
2.5D Karten: einzelne Bilder



2.5D Karten: einzelne Bilder Nachteil

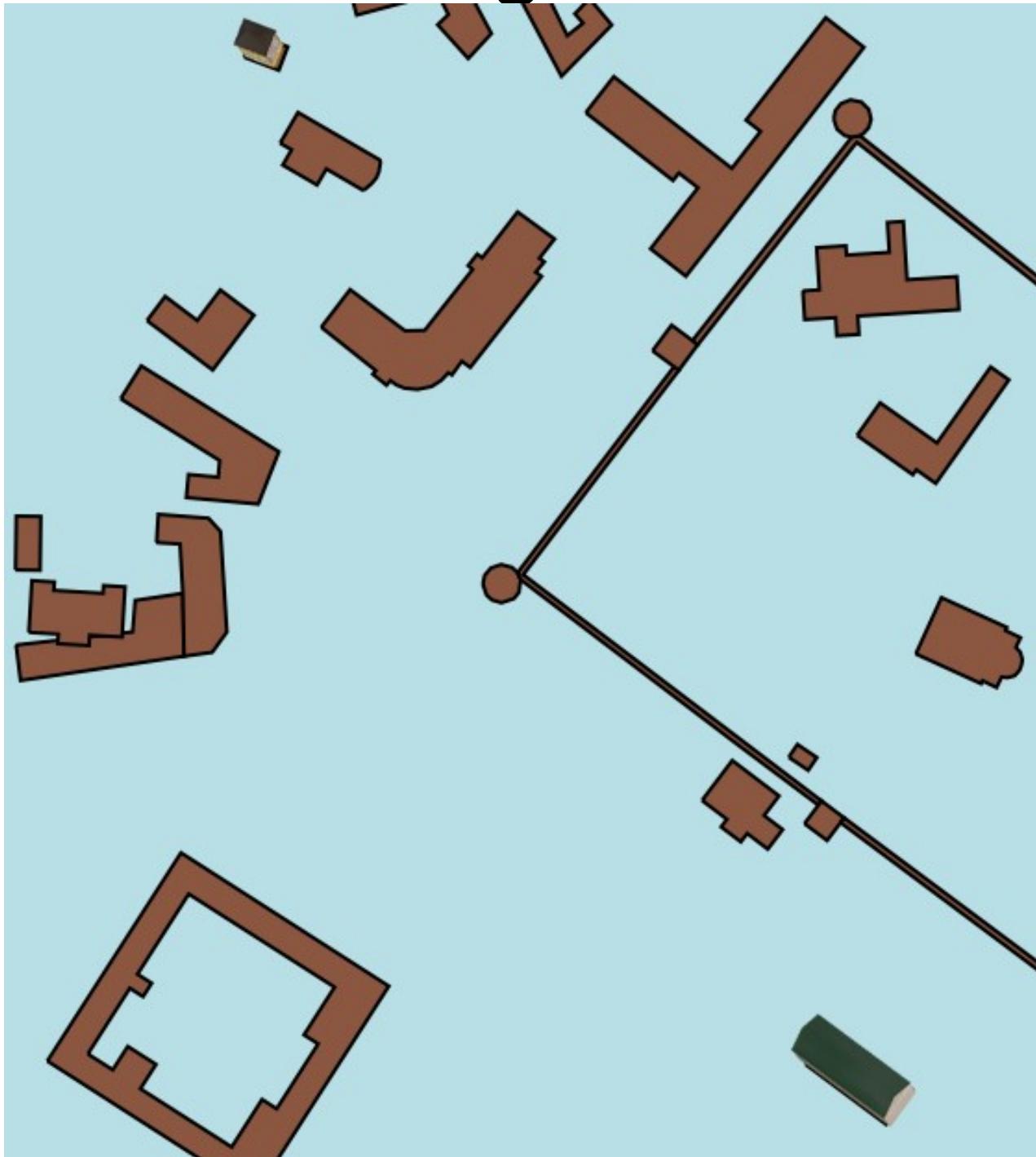
Gebäude müssen nicht nebeneinander stehen!

2.5D Karten: das gemeinsame Bild



- Skalierung $1/\cos(\text{Breite})$ für jedes 3D-Objekt anwenden
- Schrägprojektion nach den gewählten Winkeln simulieren
- Die Viewport Größen und Lage der orthographischen Kamera anpassen
- Die Ergebnisbildgrößen anpassen: die hängen von Zoom ab
- Eine GeoTiff Datei mit `gdal_translate` generieren
- Die GeoTiff Datei als Raster Ebene mit Mapnik verwenden

2.5D Karten: das gemeinsame Bild



Zukünftige Arbeit

- Vogelperpektive Karten
- Plugins für automatisierte Erstellung von Gebäudeteilen (Fenster, Türen, Fasadendekorationen)

github.com/vvoovv/blender-geo

vladimir.elistratov@gmail.com