PART II: Low Level APIs

- **OpenGL** API bindings
  - Open Graphics Library - based on C
  - http://pyopengl.sourceforge.net/

- DirectPython 11: Direct3D 11 (Windows only)
  - http://directpython11.sourceforge.net/
  - Not Cross Platform

n|w

PyOpenGL

Supports OpenGL 1.1 to 4.4

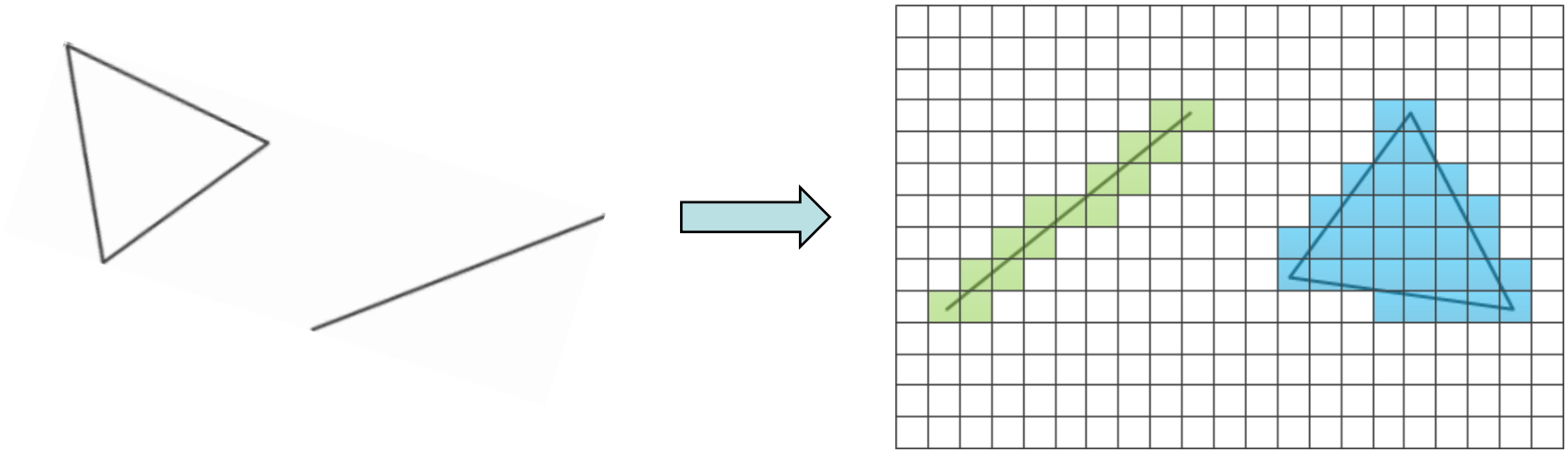
Works with popular GUI libraries, for example:

- wxPython
- PyQT / PySide
- PyGTK
- PyGame
- Tkinter (+ TogI widget)

I‘m not going to make an OpenGL introduction here...

How OpenGL Works

Primitives (=Points, Linies, Triangles) are converted to Fragments (Pixels).



Program **Vertex-Operations** (e.g. projection, transformation).
Program **Fragment-Operations** for shading/lighting/texturing.

Vertex/Fragment Shaders [run on the GPU and] are written in GLSL.

Vispy:
http://vispy.org/ **(https://github.com/vispy/vispy)**

VisPy is a young library under heavy development at this time. It targets two categories of users:

- **Users knowing OpenGL**, or willing to learn OpenGL, who want to create beautiful and fast interactive 2D/3D visualizations in Python as easily as possible.
- **Scientists without any knowledge of OpenGL**, who are seeking a high-level, high-performance plotting toolkit.

n|w

Install vispy

1) Numpy is required
2) OpenGL is required
3) A compatible GUI Toolkit is required

Then:

```
pip install vispy
```

More info: http://vispy.org/installation.html

n|w

Example, lets create a virtual environment named “gl”

```
python3.5 -m venv gl

source gl/bin/activate


pip install vispy        # also installs numpy
pip install pyglet
```

… do your stuff …

```
Deactivate
```

n|w

Development Version (currently 0.5.0.dev0)

```
git clone git://github.com/vispy/vispy.git

cd vispy

python3.5 setup.py develop
```

(Only this version support Jupyter Notebook!)

n|w

Windows

on Windows, download modules* from:
http://www.lfd.uci.edu/~gohlke/pythonlibs

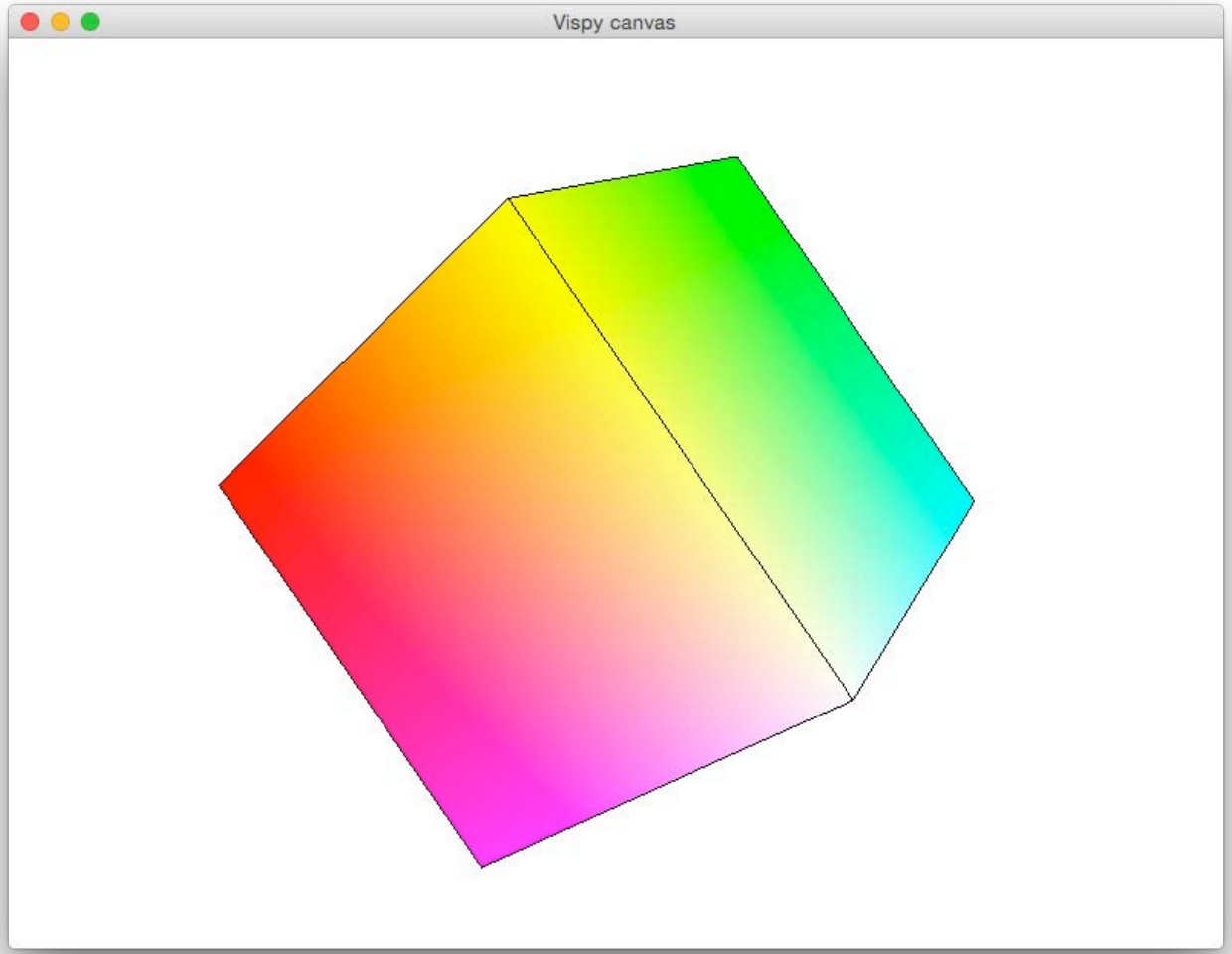
*) vispy, numpy & pyglet

n|w

Our first App

```
import sys


from vispy import app, gloo
canvas = app.Canvas(app='pyglet', keys='interactive', size=(800, 600))
@canvas.connect
def on_draw(event):
    gloo.set_clear_color((1.0, 0.0, 0.0, 1.0))
    gloo.clear()


canvas.show()
if __name__ == '__main__':
    app.run()
```

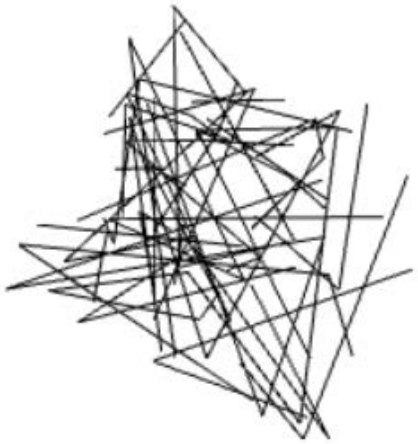
Use Shaders, Geometry



n|w

Jupyter Notebook (WebGL Output)

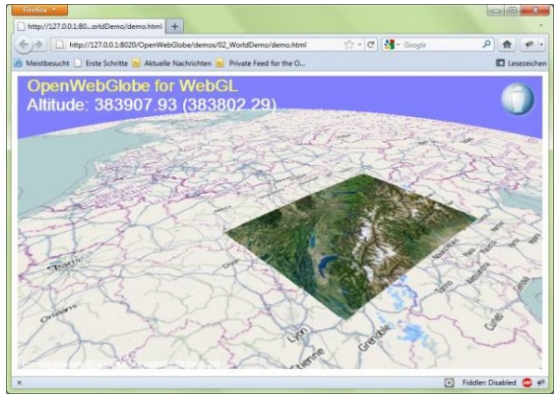
```
In [8]: c = Canvas(size=(300, 300))
```

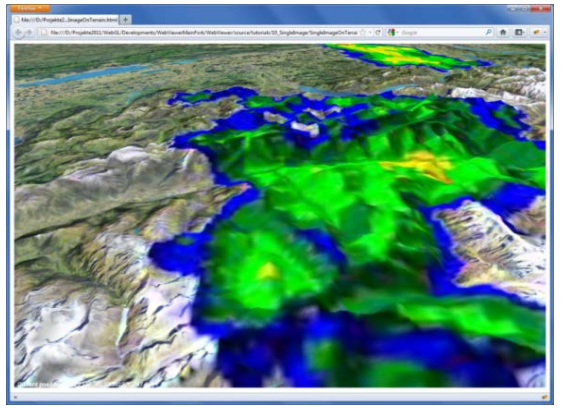


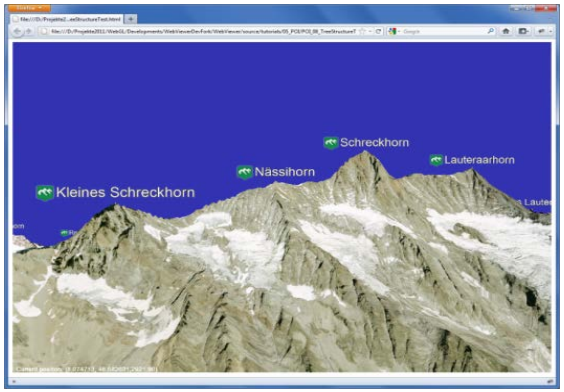
Python for Data Processing

OpenWebGlobe

- Virtual Globe using WebGL
- Open Source Project started in April 2011
- JavaScript Library for rapid development of web-based geospatial 3D applications
- Data Processing in Python

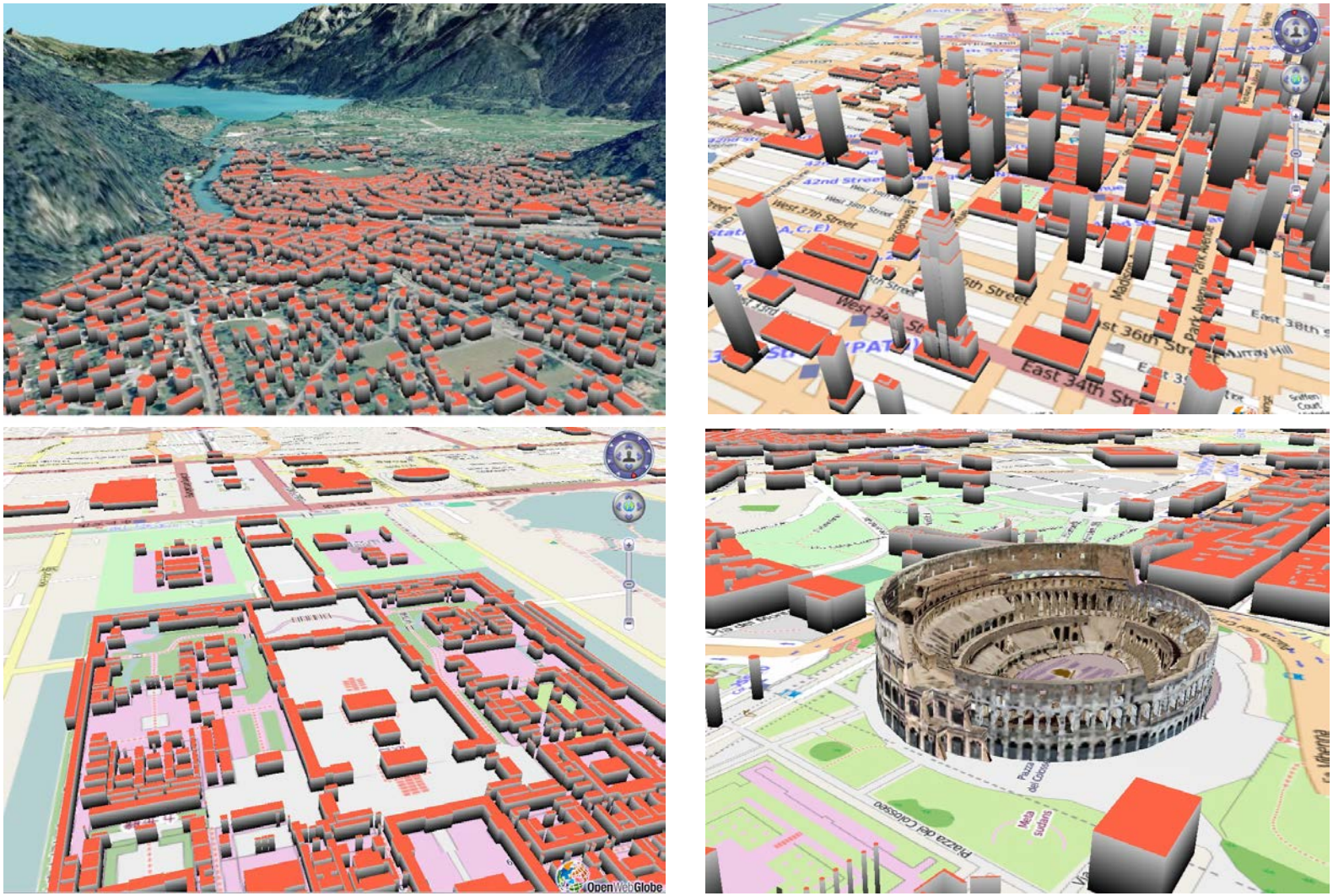







Fork me on GitHub
WebGL
HTML
5
NaturalEarth
OpenWebGlobe
Data Layer
World
Overlays
Elevation
Animate
Tweet
10

Streaming 3D-Geometry Tiles



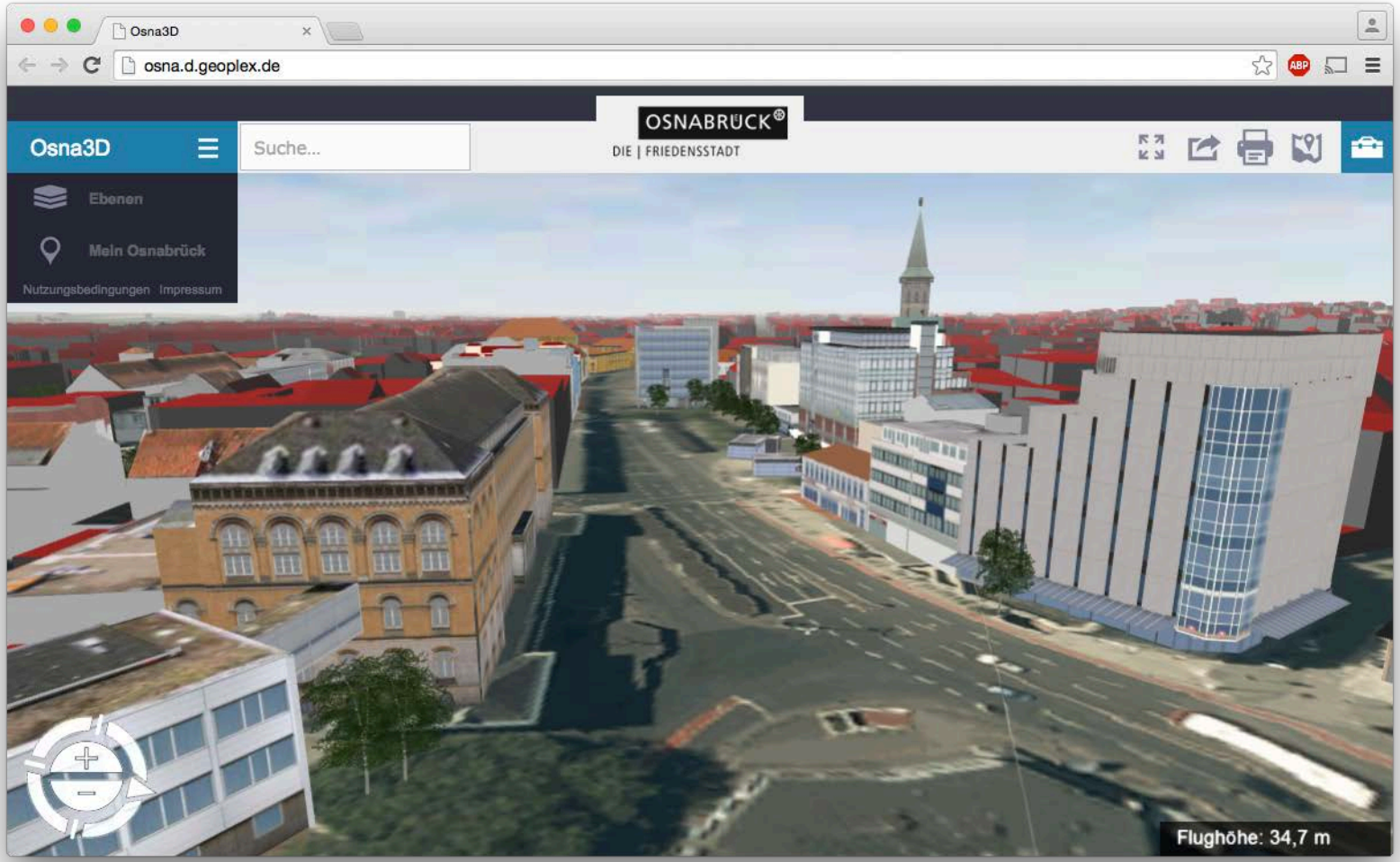
BTh Hürbi/Daetwyler, MTh Lucas Oertli, 2013

MapData © OpenStreetMap contributors

Streaming Example: 3D Geometry using OSM and “BOI” (worldwide streaming)

MTh Lucas Oertli, 2013

Streaming Example: Osnabrück (local streaming)



Created by Geoplex with Plexmap, based on OpenWebGlobe

n|w

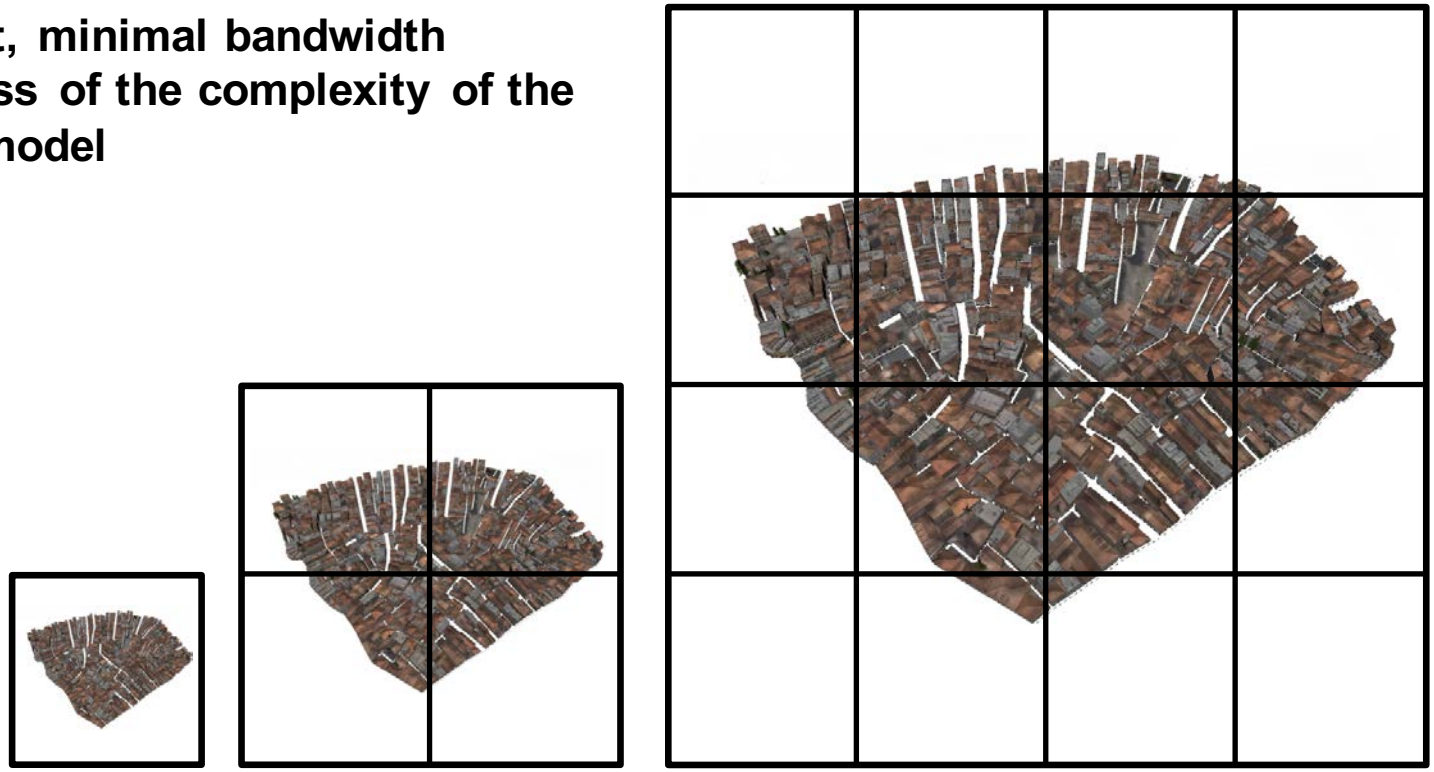
Some Problems I have with (Web-Based) Virtual Globes

- Unfortunately, WebGL compatibility is still an issue… a “fallback” is required
- Most people still prefer 2D Maps
- Navigation in 3D is too complicated for many users…
- In the “Geo-World”, 3D Models are usually not built by 3D game designers:
  - Often there are “too many” & “too big” textures per object
  - Different details per 3D-object
  - Remember “Google 3D Warehouse”
- Level of Detail: Generalization in 2D is accepted, but not in 3D!
- Limited number of people actually do have data **of the whole world**…
- Most virtual globe based applications I know are limited to a certain region/country/…
- Too slow (bandwidth/3D rendering) on mobile devices
- Too power consuming on mobile devices

Bringing together 2D Maps and 3D Globes

Concept: Prerender a 3D Scene using a high quality offline 3D renderer using an orthographic projection and create “2D” image tiles.

Constant, minimal bandwidth regardless of the complexity of the 3D city model



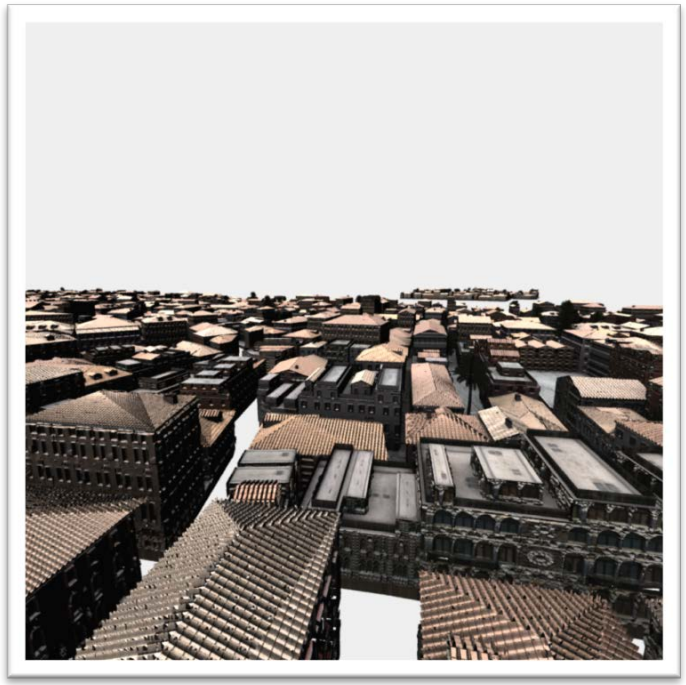
MTh Markus Jung, 2014

(Similar approaches were already done by Döllner et al. and also go back to some concepts by Sutherland)

Display in the Webbrowser as “2D Map”

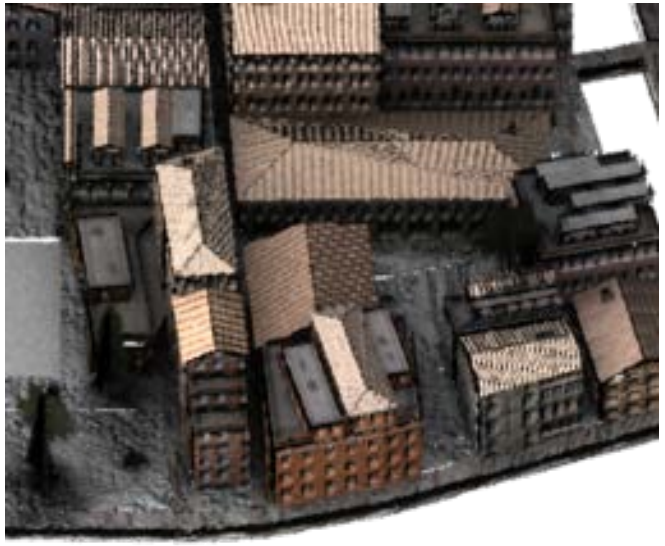
MTh Markus Jung, 2014

Display in the Webbrowser as Panorama



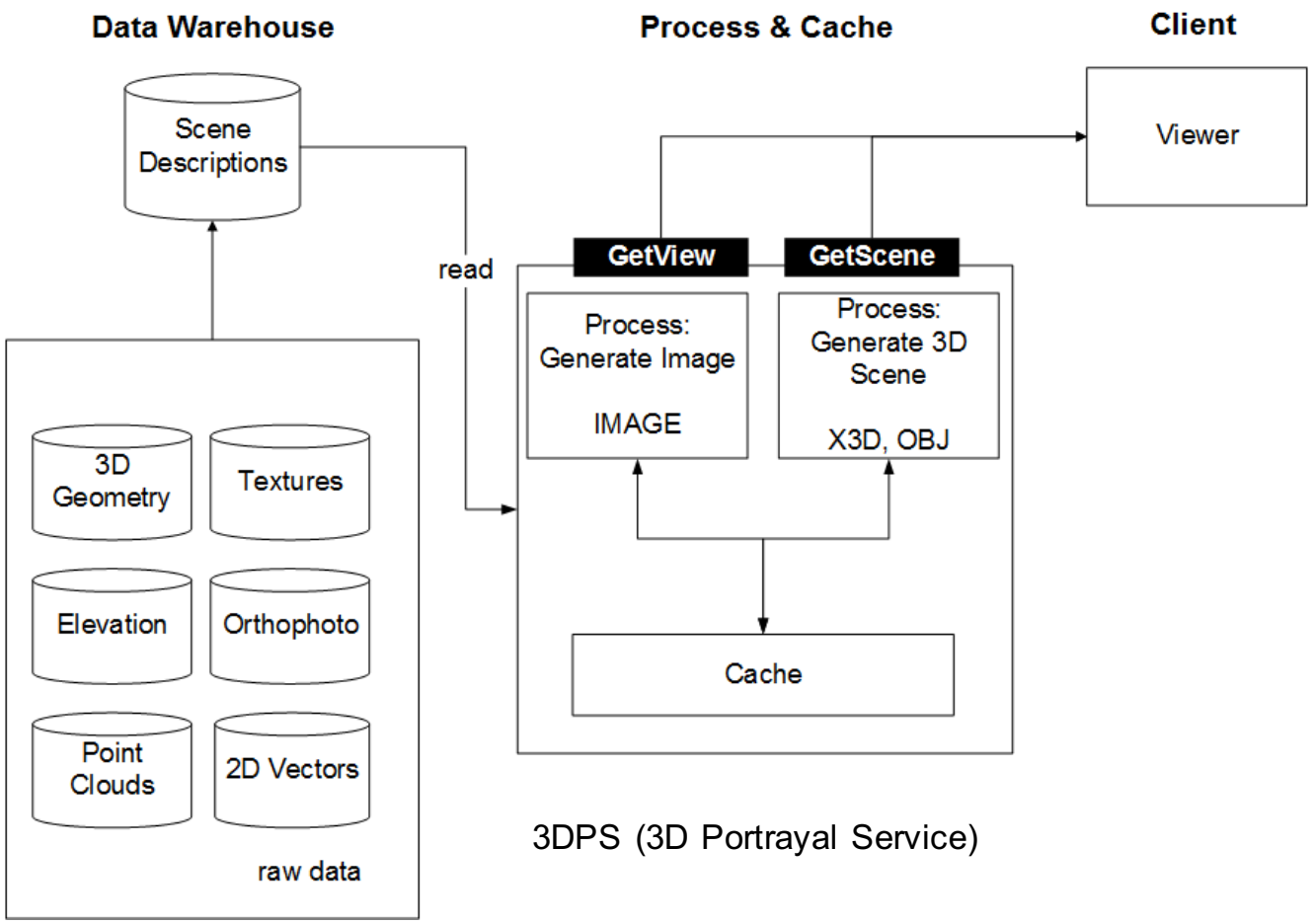
MTh Markus Jung, 2014

High Resolution Geometry doesn't matter: Same download/render speed



MTh Markus Jung, 2014

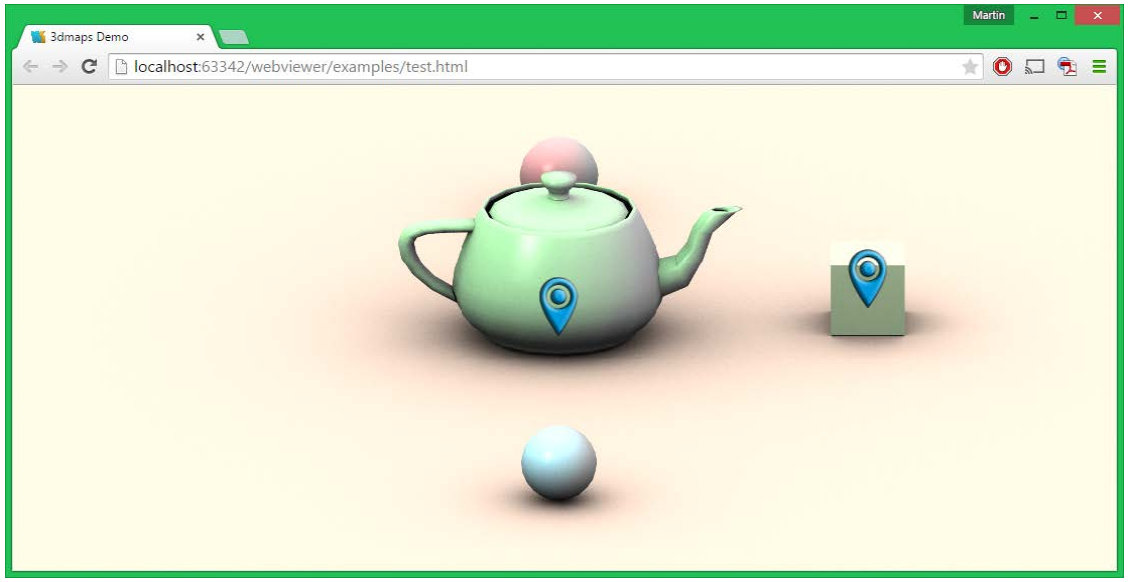
The 3dmaps.ch Project: Bringing it all together!



3DPS (3D Portrayal Service)

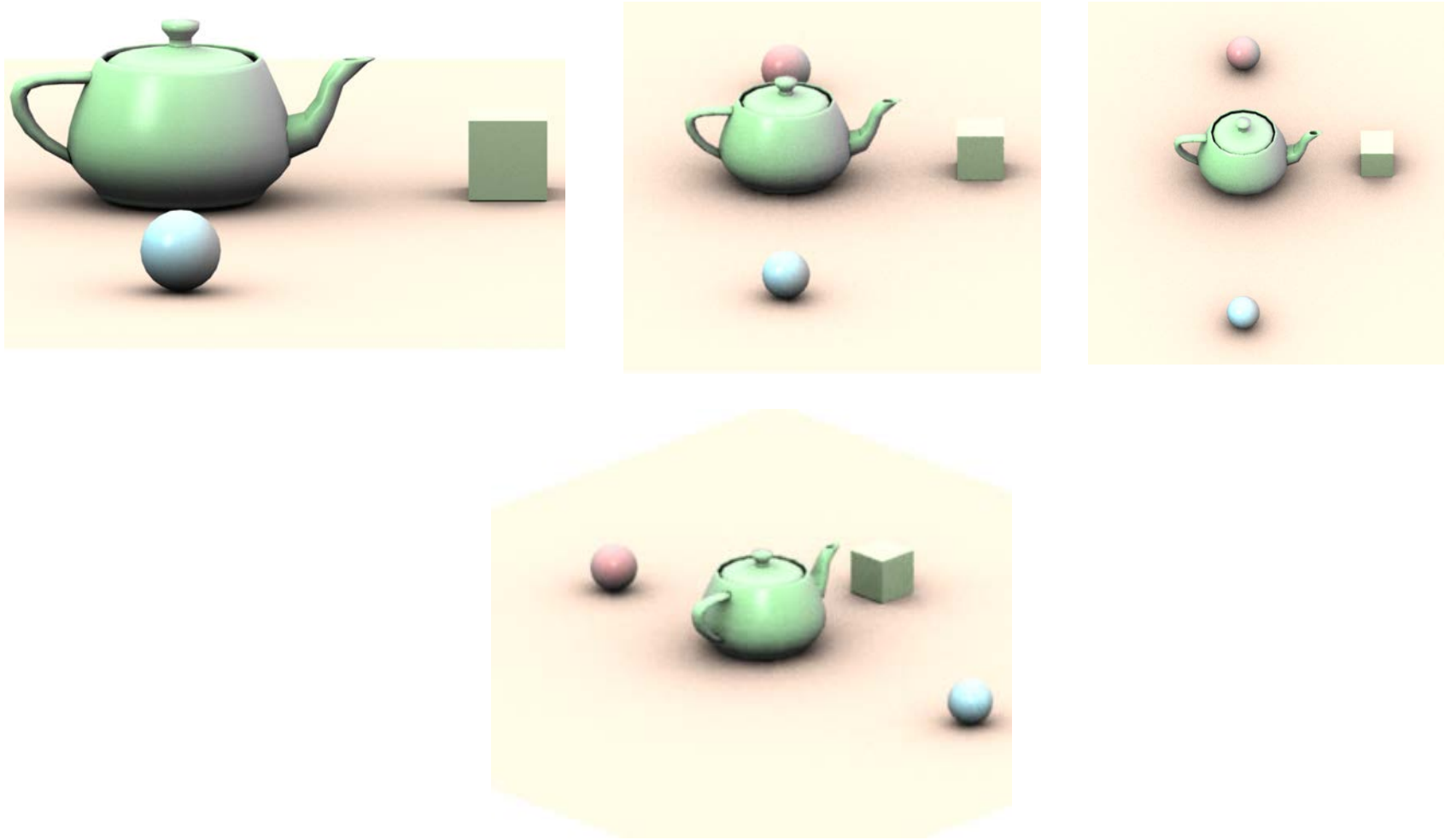
Viewer API

map3d.js Library




```
var map = new map3d.map("mapcanvas");

var layer = new map3d.imageLayer([
  "http://t1.3dmaps.ch/tiles/teatime",
  "http://t2.3dmaps.ch/tiles/teatime",
  "http://t3.3dmaps.ch/tiles/teatime",
  "http://t4.3dmaps.ch/tiles/teatime"])  });

layer.addTo(map);

var teapot_marker = new map3d.marker("Green Teapot", [0,0,0]);
teapot_marker.addTo(map);

var cube_maker = new map3d.marker("Green Cube", [80.5, 11.5, 10.5]);
cube_maker.addTo(map);
```

Different prerenderings for different pitch/view direction

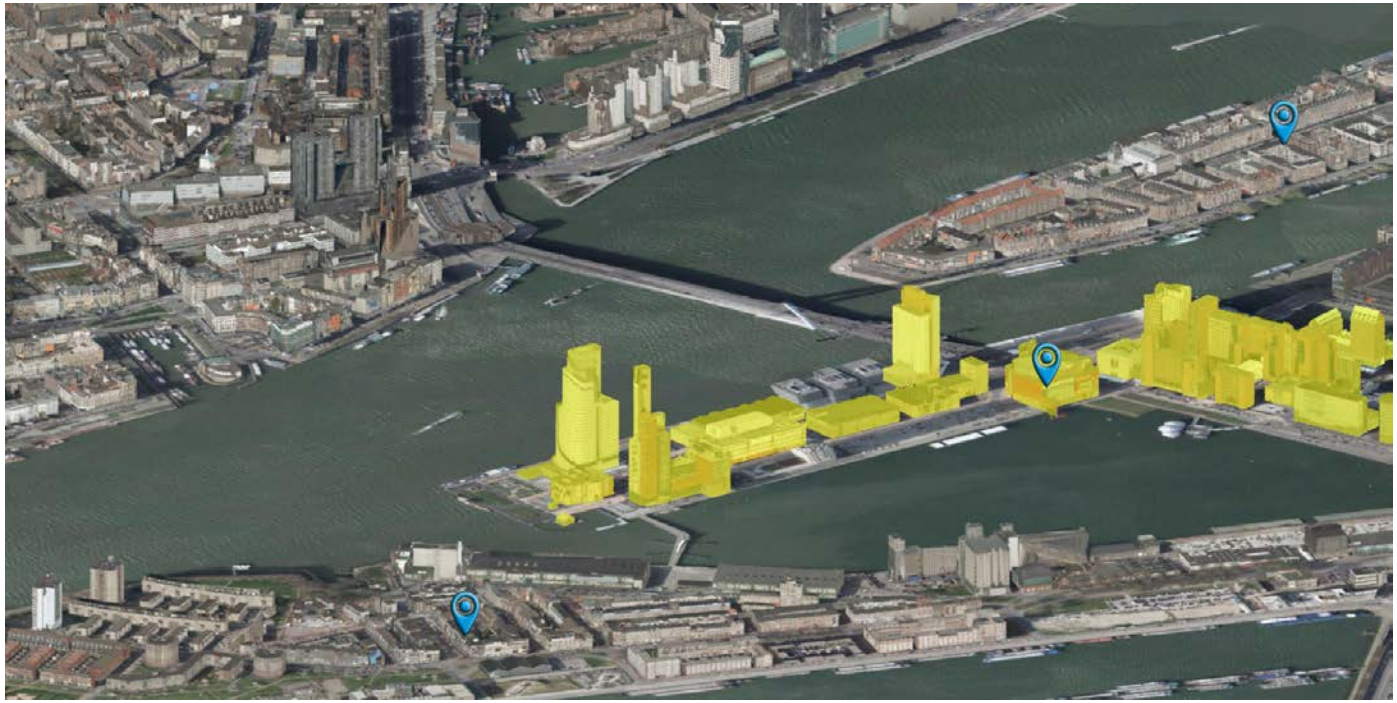


Every Prerendering needs storage, but with todays cloud storage pricing this is not really an issue anymore!

Why a teapot if we have (open) Geo Data ?!!

Use case 1: Rotterdam Dataset

90 CityGML files with a total size of 2.72 GB
26'474 textures with a size of 1024x1024, an uncompressed total data volume of around 77 GB
Orthophoto uncompressed 430 GB



Use case 2: The Roman city of Augusta Raurica

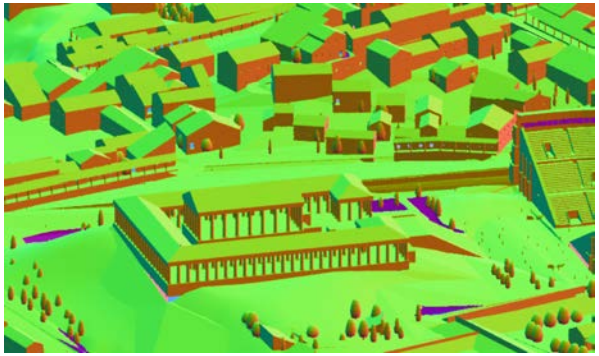
A digital reconstruction of the historical Roman City of Augusta Raurica, created at the institute of Geomatics Engineering at the FHNW. 3D-Printed to create a bronze model.

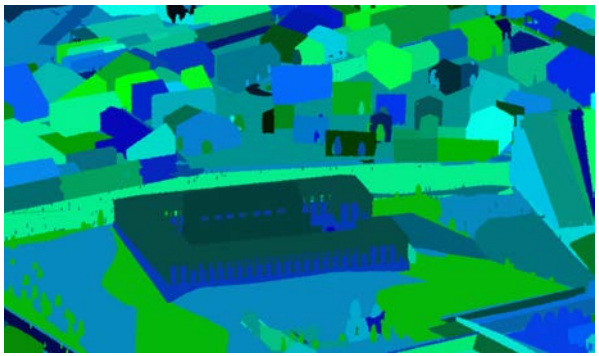
The 3D Model

About 4000 geospatial objects (buildings, roads, vegetation features, terrain, …) at three levels of detail.

3D Geometry & Textures around 1 GB

Prerendering the Model: Color Map, Normal Map, Id-Map, Depth Map



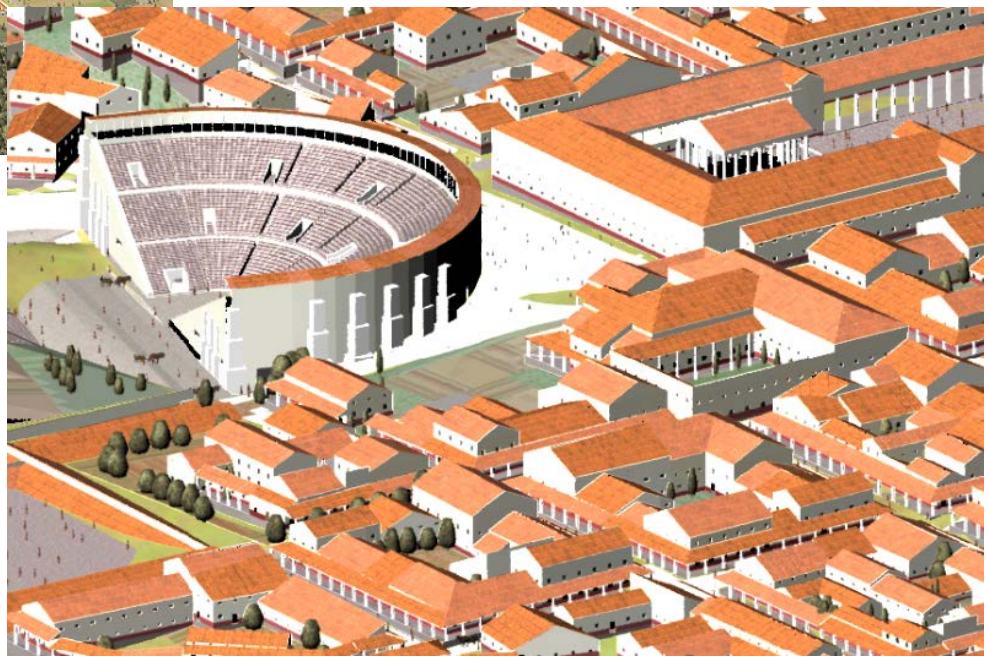


Dynamic Lighting

Normal Map: for Object Identification: Highlighting, special effects, …

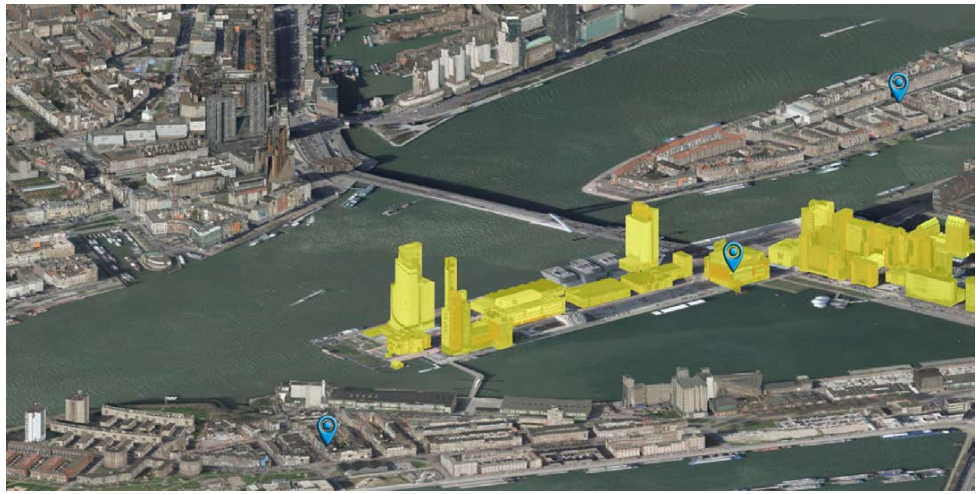
Depth Map: for 3D Position, special effects, …

3D View in the (mobile) webbrowser with dynamic Lighting



The Viewer

- The viewer basically uses the same concepts as a “2D Map Viewer”
- map3d.js supports WebGL
  There is also a pure canvas version available as fallback
- Operations like “highlighting” are highly customizable. Basically it is an image processing operation which runs on the GPU (WebGL Version). If there is no WebGL available, the operation is done using JavaScript.

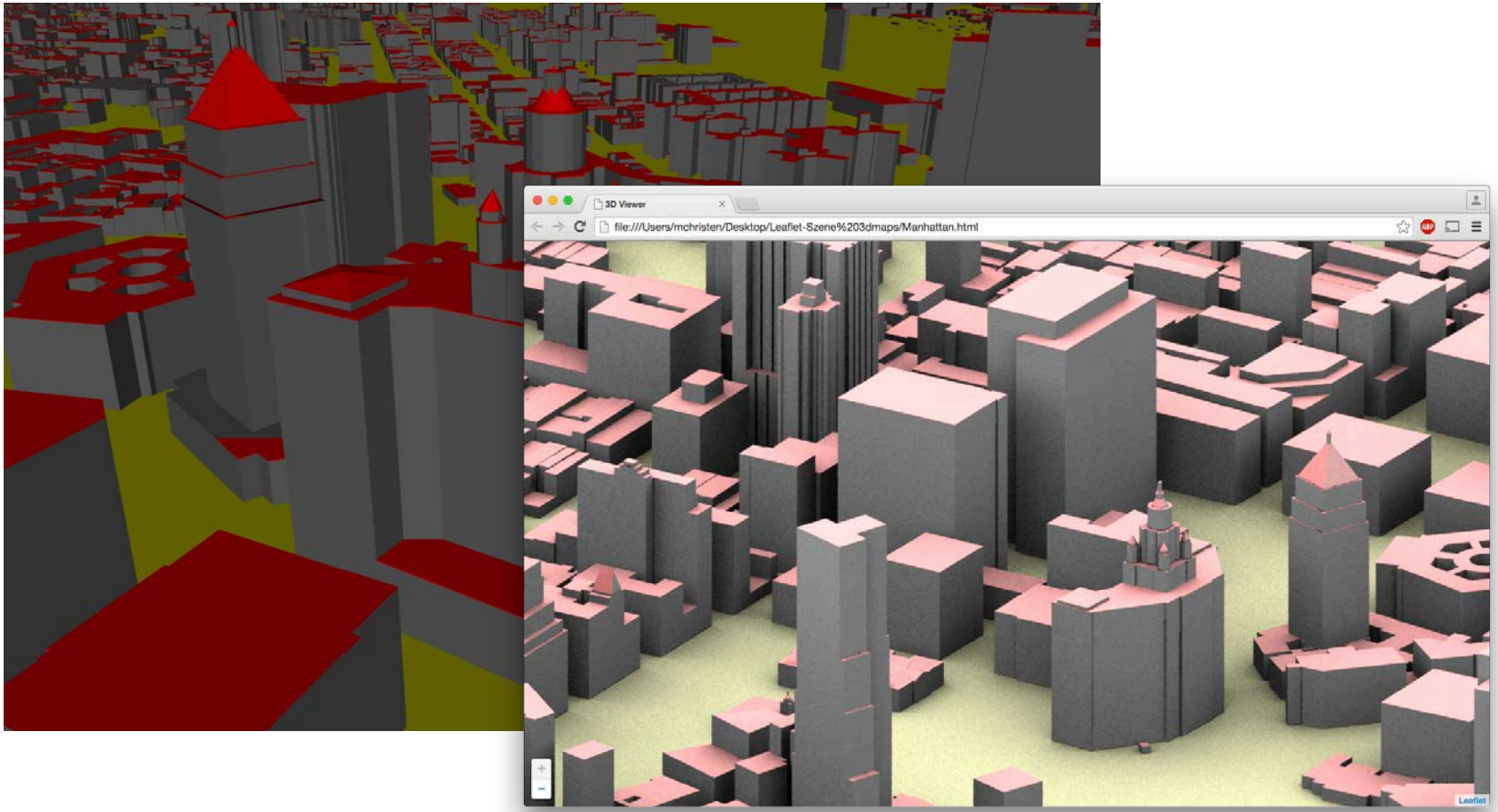


Outlook (1)

- Implement more effects and lighting models,
  dynamic snow/water/etc. using depth map & normal map
- Layer management (include point clouds, mix different layers using depth map
- Add realtime content (“mix real 3D Object”) using depth-map
- Release map3d.js as Open Source (around Q2/2016)

Outlook (2)

More Rendering Effects, for example Screen Space Ambient Occlusion (SSAO)



BTh, Daniel Rettenmund 2015

Outlook (3)

OpenWebGlobe 2

- “isometric maps” will be one of many features of OpenWebGlobe 2
- “isometric maps” will be the default 3D Viewer in OpenWebGlobe 2
- Switch to “real 3D” anytime
- State is saved:
  - The best matching viewpoint is selected when switching
  - If you highlight an object in “isometric mode”, it will be highlighted in “Real 3D” mode too.

n|w

Conclusion

3 ways to use 3D Graphics in Python were presented:

1. Using Python & Blender (including Blender Game Engine)
2. Using Low a Level API (OpenGL/WebGL)
3. Using Python to process 3D Views

Which approach is the best for Python ? This really depends the application domain.

I am quite sceptic with 2. - If you want to create a complex 3D game, I don't recommend using Python at this time.

I believe Python is great for 1. and 3.

Next Meetup: February 9th, 2016 in Muttenz (near Basel) 18:00 to 21:00

http://www.pybasel.ch

University of Applied Sciences and Arts Northwestern Switzerland
School of Architecture, Civil Engineering and Geomatics

python™

GeoPython 2016

Basel, Switzerland

June 22 - 24, 2016

http://www.geopython.net

n|w

Q&A