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

Workshop:

Creating Virtual Globe Applications using the OpenWebGlobe SDK

04.03.2012

Benjamin Loesch, Martin Christen
Fachhochschule Nordwestschweiz

These slides for download:

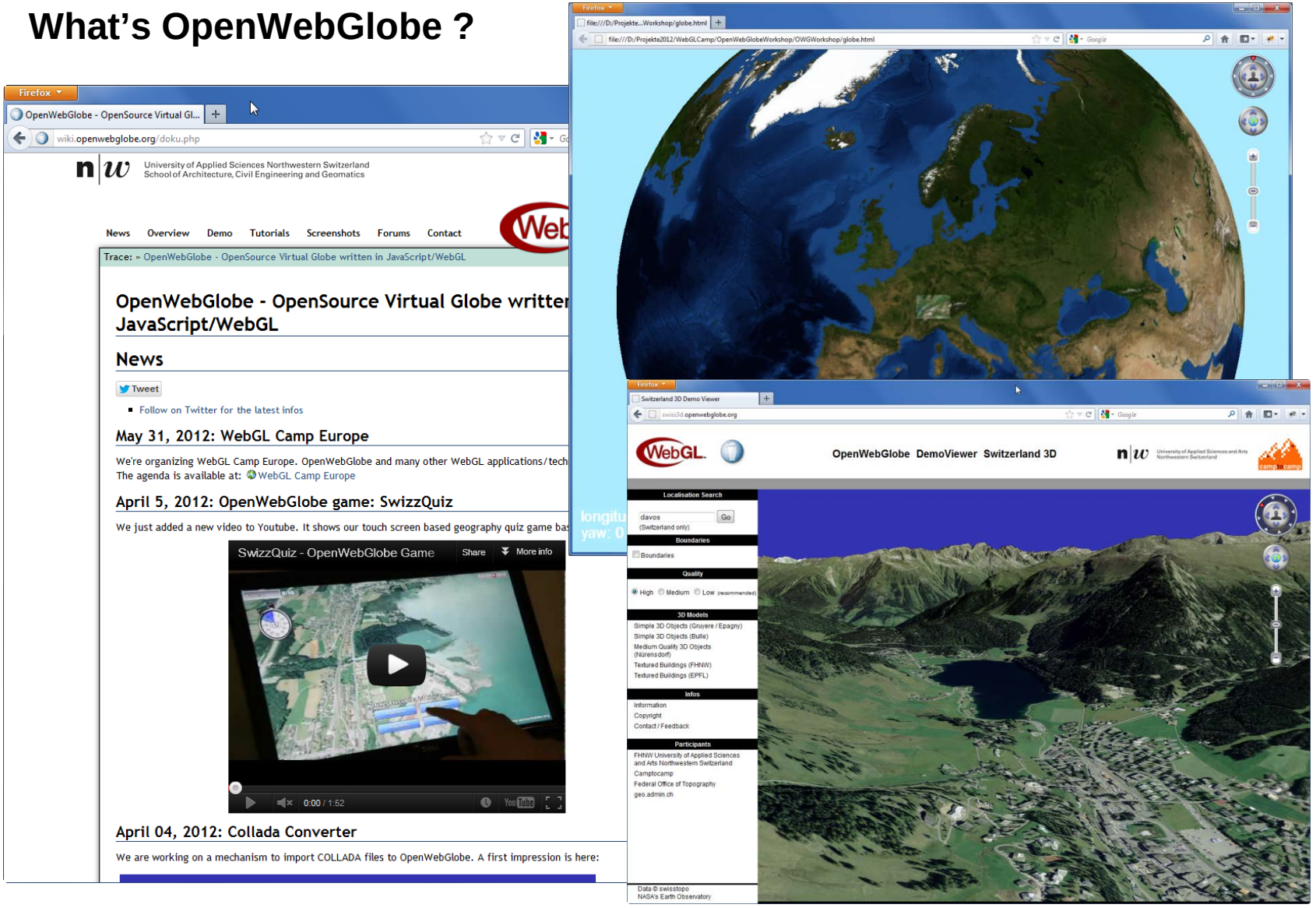
www.openwebglobe.org/workshopdata/slides.pdf

OpenWebGlobe Function Reference

http://www.openwebglobe.org/workshopdata/ref/

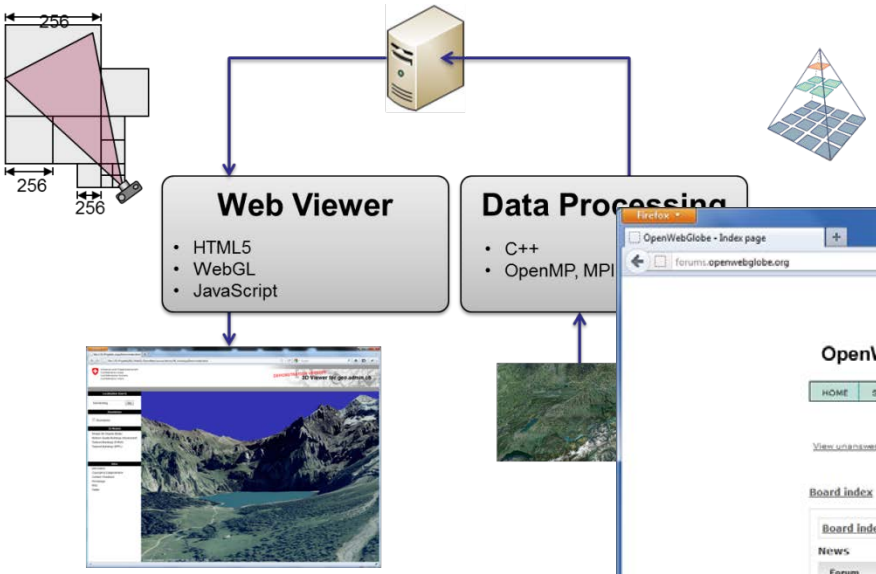
University of Applied Sciences Northwestern Switzerland
School of Architecture, Civil Engineering and Geomatics

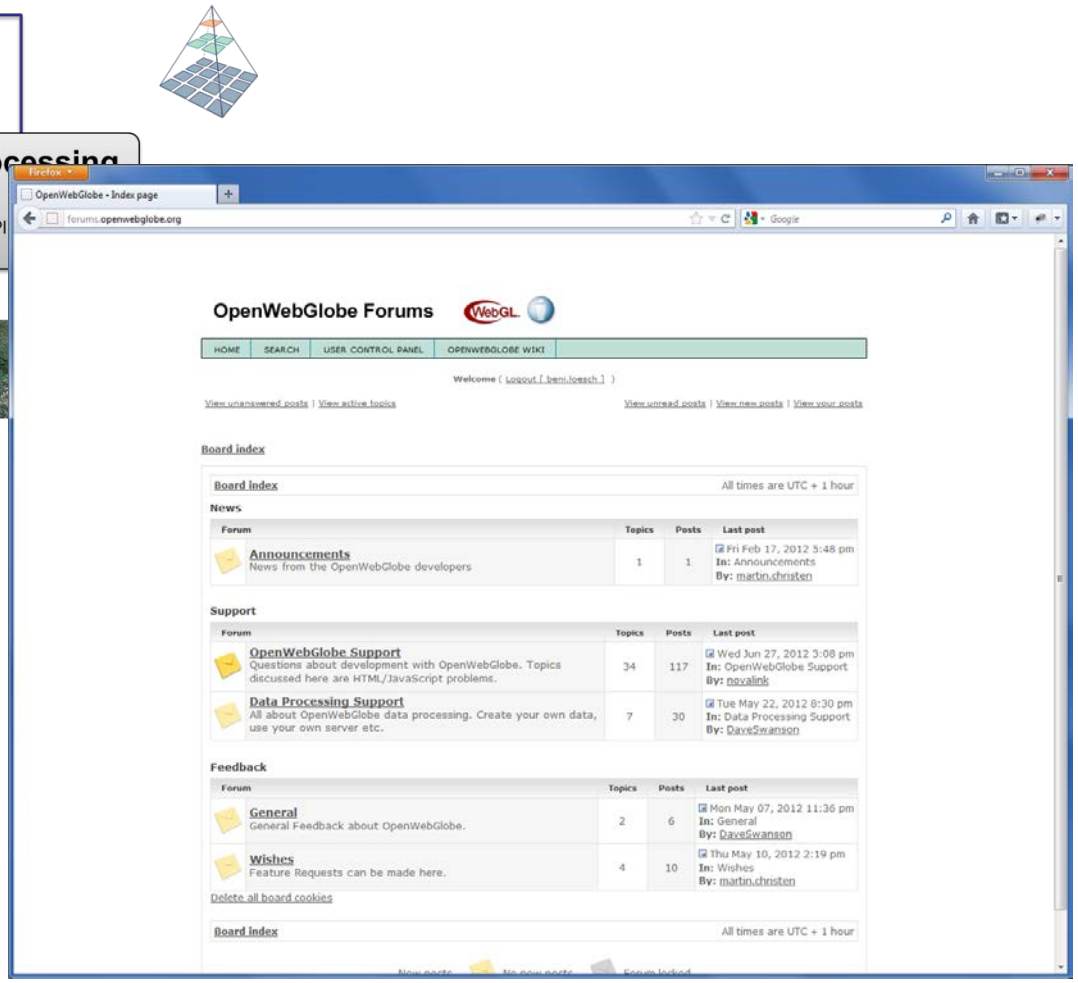
What's OpenWebGlobe ?



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

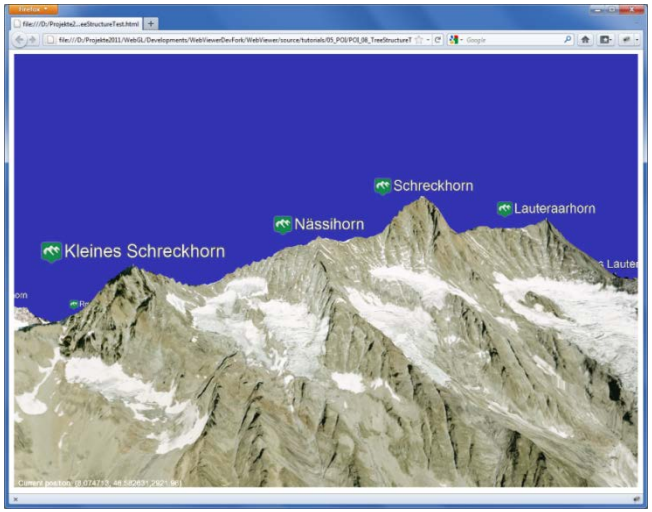
What's OpenWebGlobe ?

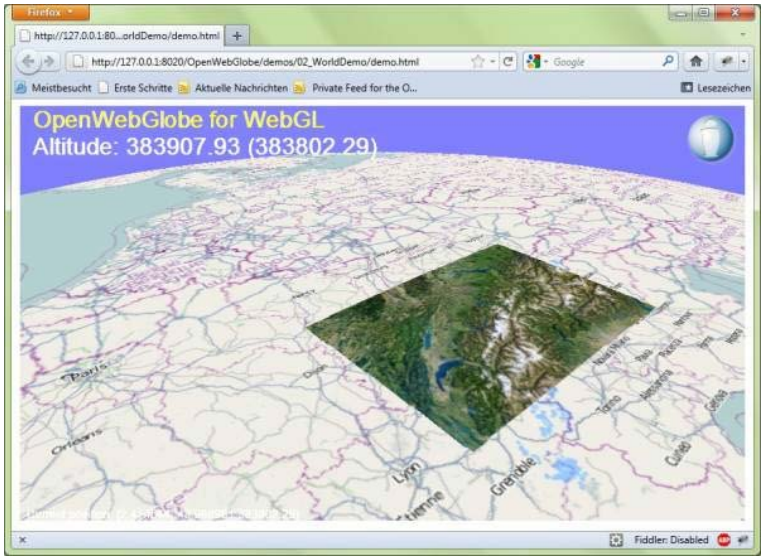




University of Applied Sciences Northwestern Switzerland
School of Architecture, Civil Engineering and Geomatics

What's OpenWebGlobe ?

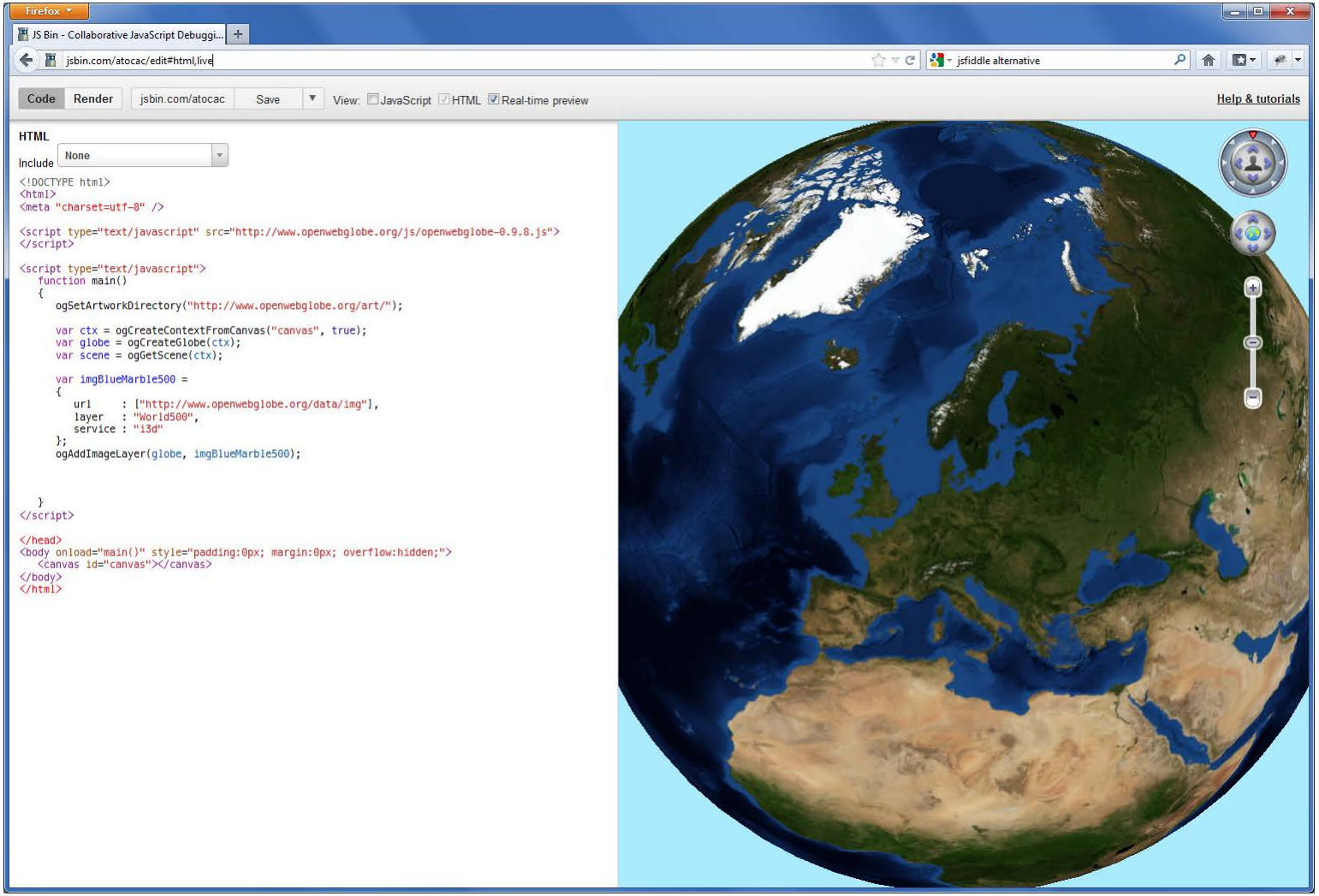




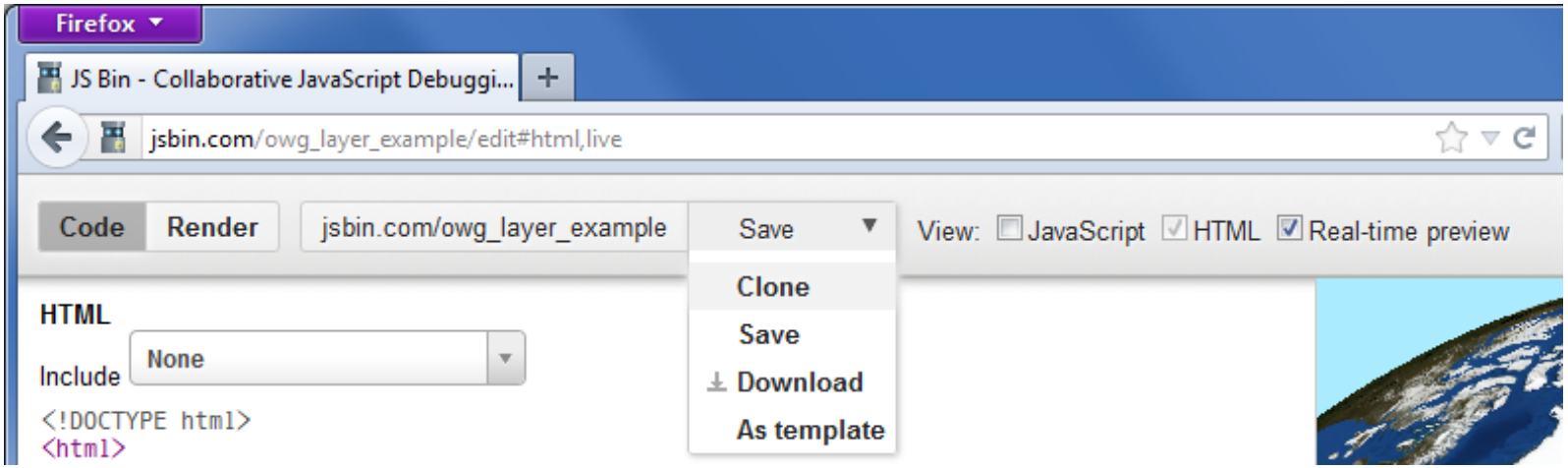
Workshop Content:
Creating Virtual Globe Applications using the OpenWebGlobe SDK

1. **Setting up a simple virtual globe application**
2. **Different image sources**
3. **Navigation**
4. **Define points of interest POIs**
5. **3d models**
6. **Integrate dynamic content**
7. **Further Developement**

Before we start: Introduction to jsBin



Before we start: Introduction to jsBin



1. Setting up a simple virtual globe application

1. Setting up a simple virtual globe application

```
<!DOCTYPE html>
<html>
<meta "charset=utf-8" />

<script type="text/javascript" src="http://www.openwebglobe.org/js/openwebglobe-0.9.8.js"></script>

<script type="text/javascript">
   function main()
   {
      ogSetArtworkDirectory("http://www.openwebglobe.org/art/");

      var ctx = ogCreateContextFromCanvas("canvas", true);
      var globe = ogCreateGlobe(ctx);

      var imgBlueMarble500 =
      {
         url     : ["http://www.openwebglobe.org/data/img"],
         layer   : "World500",
         service : "i3d"
      };
      ogAddImageLayer(globe, imgBlueMarble500);

   }
</script>

</head>
<body onload="main()" style="padding:0px; margin:0px; overflow:hidden;">
   <canvas id="canvas"></canvas>
</body>
</html>
```

1. Setting up a simple virtual globe application

```
var ctx    = ogCreateContextFromCanvas("canvas", true);
var globe = ogCreateGlobe(ctx);
```

ogCreateContextFromCanvas(sCanvasId, fullscreen, *cbfInit, cbfExit, cbfResize*)
Convenience function to create context

ogCreateGlobe(context_id)
create globe (WGS84 world)

1. Setting up a simple virtual globe application

```
var imgBlueMarble500 =
{
    url     : ["http://www.openwebglobe.org/data/img"],
    layer   : "World500",
    service : "i3d"
};
ogAddImageLayer(globe, imgBlueMarble500);

var elvSRTM_CH =
{
    url     : ["http://www.openwebglobe.org/data/elv"],
    layer   : "SRTM",
    service : "i3d"
};
ogAddElevationLayer(globe, elvSRTM_CH);
```

ogAddImageLayer(globe_id, options)
Add an image layer to the globe

ogAddElevationLayer(globe_id, options)
Add an elevation layer to the globe

1. Setting up a simple virtual globe application

```
ogSetArtworkDirectory("http://www.openwebglobe.org/art/");
```

1. Setting up a simple virtual globe application

```
<!DOCTYPE html>
<html>
<meta "charset=utf-8" />
<script type="text/javascript" src="http://www.openwebglobe.org/js/openwebglobe-0.9.8.js"></script>
<script type="text/javascript">

    function main()
    {
        ogSetArtworkDirectory("http://www.openwebglobe.org/art/");

        var ctx    = ogCreateContextFromCanvas("canvas", true);
        var globe = ogCreateGlobe(ctx);
        var scene = ogGetScene(ctx);

        var imgBlueMarble500 =
        {
            url     : ["http://www.openwebglobe.org/data/img"],
            layer   : "World500",
            service : "i3d"
        };
        ogAddImageLayer(globe, imgBlueMarble500);

        var elvSRTM_CH =
        {
            url     : ["http://www.openwebglobe.org/data/elv"],
            layer   : "SRTM",
            service : "i3d"
        };
        ogAddElevationLayer(globe, elvSRTM_CH);
    }
</script>

</head>
    <body onload="main()" style="padding:0px; margin:0px; overflow:hidden;">
    <canvas id="canvas"></canvas>
</body>
</html>
```

Exercise – A simple virtual globe application

1. Use jsBin or your own IDE and create a simple virtual globe application

http://jsbin.com/owg_layer_example

Please clone source code before start working

2. Try to add another image layer

```
var imgLandsatCH =  {
   url: ["http://www.openwebglobe.org/data/img"],
   layer: "LandsatCH",
   service: "i3d"
};
```

3. Try to add an elevation layer

```
var elvSRTM_CH =
{
   url     : ["http://www.openwebglobe.org/data/elv"],
   layer   : "SRTM",
   service : "i3d"
};
```

Exercise – A simple virtual globe application

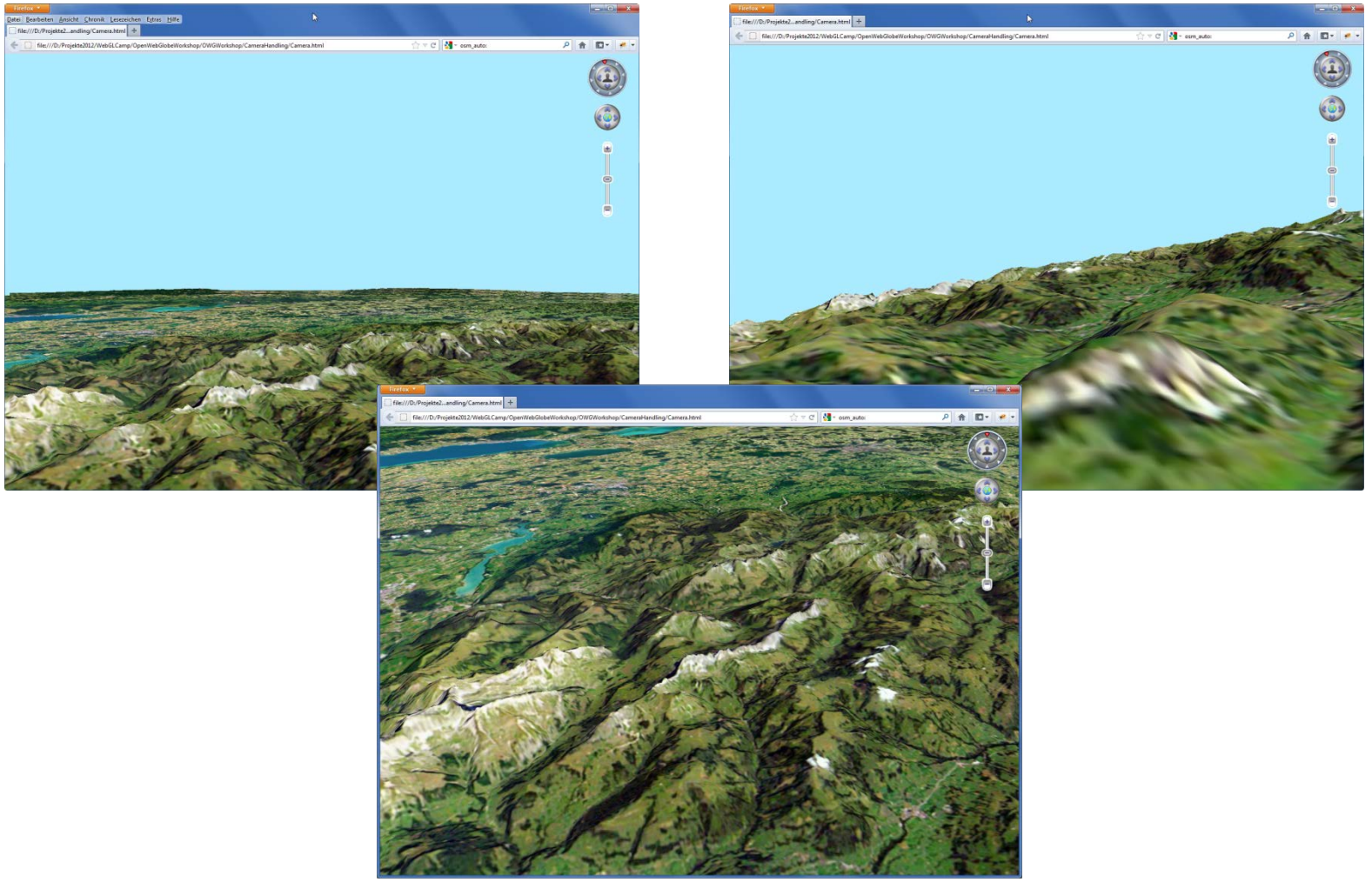
4. Try to add OpenStreetMap Data from WMS Server or OSM Tiles

```
var wmsserver =
{
   url: ["http://129.206.228.72/cached/osm"],
   layer: "osm_auto:all",
   SRS: "EPSG:3857",
   service: "wms"
};
```

```
var imgOpenStreetMap =
{
   url     : ["http://a.tile.openstreetmap.org",
              "http://b.tile.openstreetmap.org",
              "http://c.tile.openstreetmap.org" ],
   service : "osm"
};
ogAddImageLayer(globe, imgOpenStreetMap);

// (4) Set Color of north, south pole to match the "OpenStreetMap Color"
ogSetNorthpoleColor(globe, 181/255,208/255,208/255);
ogSetSouthpoleColor(globe, 241/255,238/255,232/255);
```

2. Navigation in OpenWebGlobe

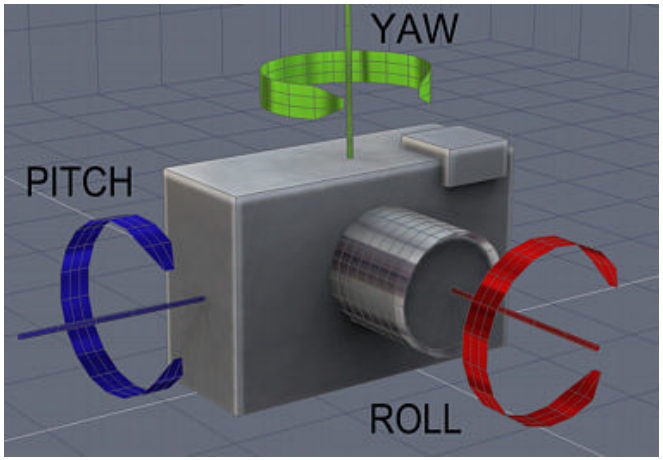


2. Navigation in OpenWebGlobe: A code example

```
cam1 = ogGetActiveCamera(scene);
ogSetPosition(cam1,7.23,46.43,10000);
ogSetOrientation(cam1,100,0,0);

cam2 = ogCreateCamera(scene);
ogSetPosition(cam2,6.23,46.43,10000);
ogSetOrientation(cam2,90,0,0);

ogSetActiveCamera(cam1);
```



2. Navigation in OpenWebGlobe : Functions for Camera Handling

ogGetActiveCamera(scene_id)
Get the active camera

ogCreateCamera(scene_id)
Create a new camera object

ogSetPosition(camera_id, lng, lat, elv)
Set camera position.

ogSetOrientation(camera_id, yaw, pitch, roll)
Set the camera orientation

ogSetActiveCamera(camera_id)
Set a camera object active

ogGetPosition(scene_id)
get the position of the active camera.
returns {“Longitude” ,”Latitude”, ”Elevation”}

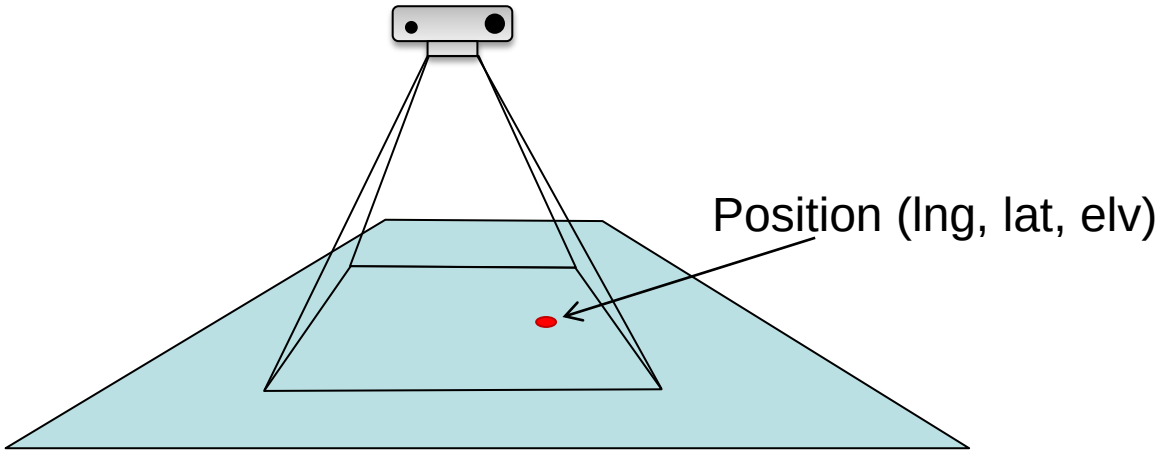
ogGetOrientation(scene_id)
get the orientation of the active camera.
returns {“Yaw” ,”Pitch”, ”Roll”}

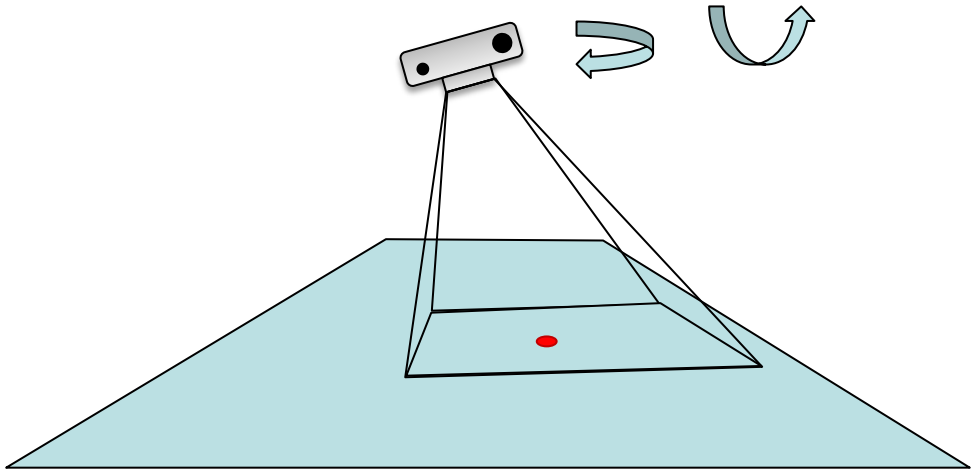
ogGetNumCameras(scene_id)
returns the number of defined cameras.

ogGetCameraAt(scene_id, index)
returns the camera at index.

ogLookAt(scene_id, lng, lat, elv)
Turn the active camera to look at a position.

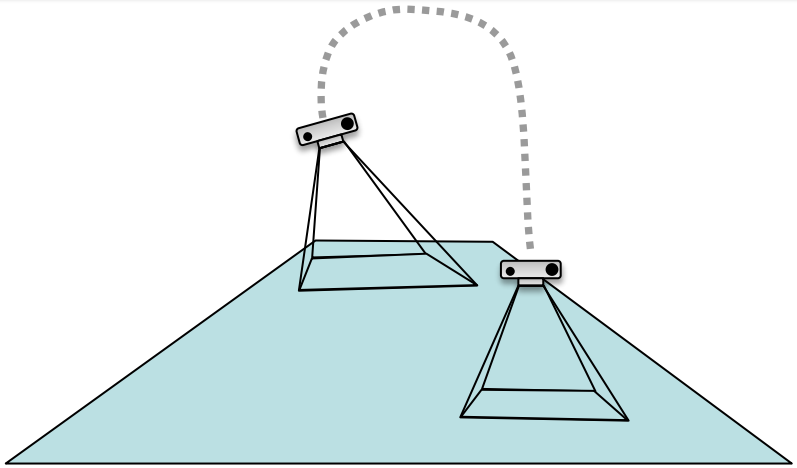
2. Navigation in OpenWebGlobe: ogLookAt





2. Navigation in OpenWebGlobe : FlyTo Functions

ogFlyTo(scene_id, lng, lat, elv, *opt_yaw, opt_pitch, opt_roll*)
flies the camera to a specific position.



ogSetFlightDuration(scene_id, timespan)
Set the duration of the FlyTo-animation in [ms].

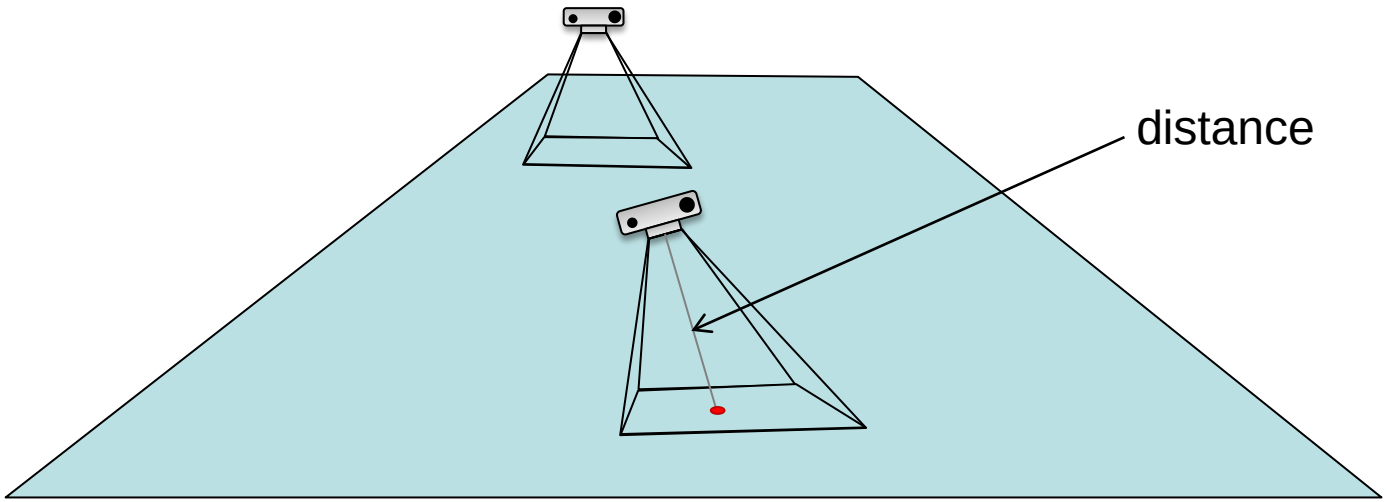
ogSetInPositionFunction(context_id, cbfInPosition)

ogSetFlyToStartedFunction(context_id, cbfFlyToStarted)

ogStopFlyTo(scene_id)

2. Navigation in OpenWebGlobe: FlyTo Functions

ogFlyToLookAtPosition(scene_id, lng, lat, elv, distance, *opt_yaw, opt_pitch, opt_roll*) flies the camera to a specific look-at position.



2. Navigation in OpenWebGlobe : Exercise

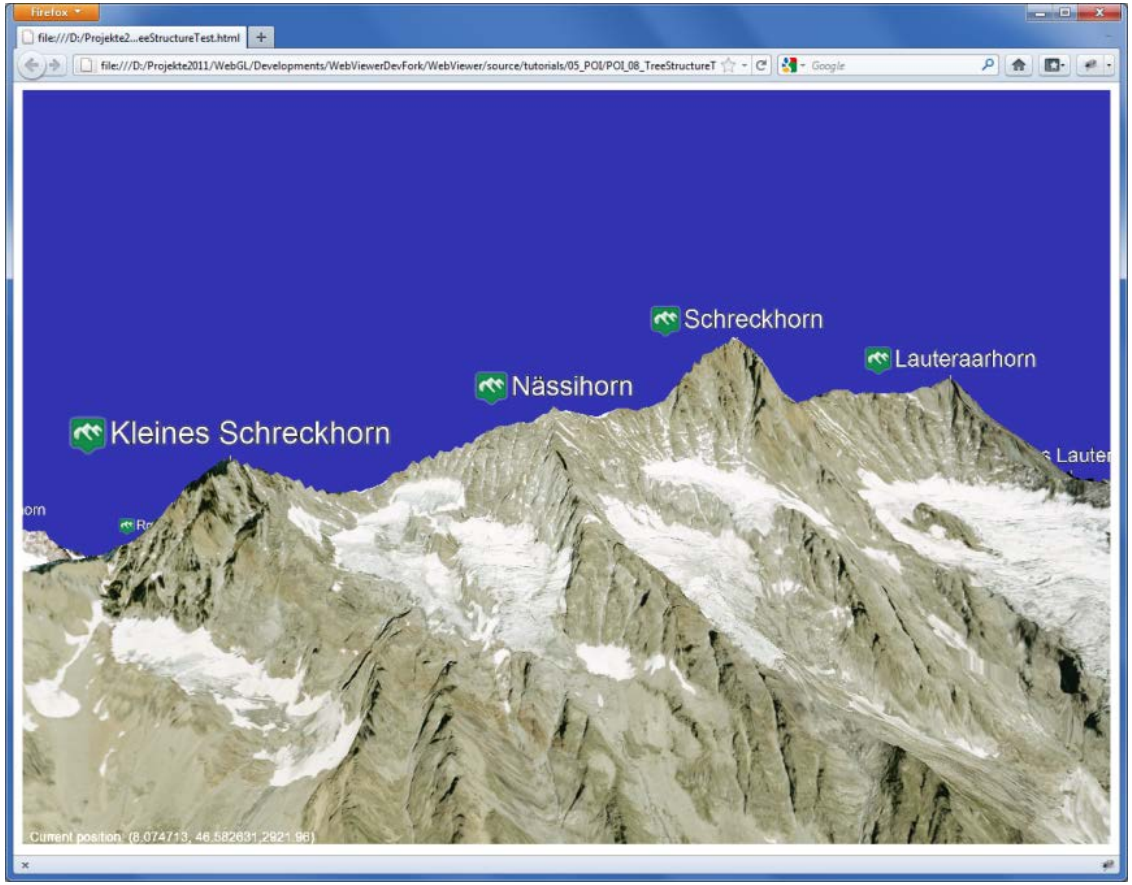
1. Use jsBin or your own IDE to test different navigation functions

http://jsbin.com/owg_navigation

Please clone source code before start working

2. Implement the missing functions

3. Define Points of Interest

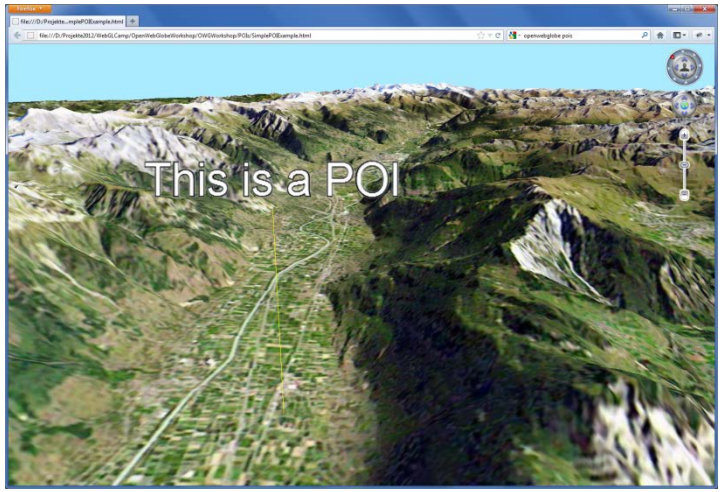


3. Define Points of Interest: A code example

```
//--- POI related code ----------------------
poilayer = ogCreatePOILayer(globe,"mylayer");

var poioptions =
{
    "text"      :   "This is a POI",
    "position"  :   [45.1234,7.1234,1000],
    "size"      :   50,
    "flagpole"  :   true
};

var poi = ogCreatePOI(poilayer,poioptions);
```



3. Define Points of Interest: Create a POI Layer

ogCreatePOILayer(world_id, layername, *textstyle, iconstyle*)
Creates a POI Layer

ogHidePOILayer(poilayer_id)
Hides a POI Layer

ogShowPOILayer(poilayer_id)
Shows a POI Layer

```
var customTextstyle =
  {
    "font"    : "Arial",
    "fontSize"  : 32,
    "backgroundColor" : [0,0,0,0],
    "lineWidth" : 3,
    "strokeColor" : [0,0,0,1],
    "shadowOffsetX" : 0,
    "shadowOffsetY" : 0,
    "shadowBlur" : 0,
    "shadowColor" : [0,0,0,0],
    "fontColor" : [1,0,0,1]
  };
```

```
var customIconstyle =
  {
    "width" : 32,
    "height" : 32,
    "border" : 0,
    "backgroundColor" : [0,0,0,0],
    "shadowOffsetX" : 0,
    "shadowOffsetY" : 0,
    "shadowBlur" : 1,
    "shadowColor" : [0,0,1,1]
  };
```

3. Define Points of Interest: Create a POIs

ogCreatePOI(poilayer_id, options)
Create a POI

ogHidePOI(poi_id)
Hide POI

ogShowPOI(poi_id)
Show previously hidden POI

ogPickPOI(scene_id, mx, my)
Pick a poi.

ogChangePOIIcon(poi_id, url)
Change POI Icon

ogChangePOIPositionWGS84(poi_id, lng, lat, elv)
Change POI Positions

ogChangePOISize(poi_id, size)
Change POI Size

ogChangePOIText(poi_id, text)
Change text of POI

```
var poioptions =
 {
    "icon"       :   "http://bit.ly/MYbha5",
    "text"       :   "This is a POI",
    "position"  :   [45.1234,7.1234,1000],
    "size"       :    50,
    "flagpole"  :   true,
    "flagpoleColor"   : [0,0,1,1],
    "visibilityRange" : [10,100]
 };
```

3. Define Points of Interest: POI Styles

3. Define Points of Interest: POI picking

```
function main()
{
    //...
    ogSetMouseDownFunction(context,onMouseDown);
    //...
}


function onMouseDown(ctx, button, mx, my)
{
    var scene = ogGetScene(ctx);
    var pickedpoi_id = ogPickPOI(scene,mx,my);
    //...
}
```

3. Define Points of Interest: OpenWebGlobe Events

ogSetMouseDownFunction(context_id, cbfMouseDown)
Set callback function for mouse down event
callback_function(context_id, button, mx, my)

ogSetMouseMoveFunction(context_id, cbfMouseMove)
Set callback function for mouse move event
callback_function(context_id, mx, my)

ogSetMouseUpFunction(context_id, cbfMouseUp)
Set callback function for mouse up event
callback_function(context_id, button, mx, my)

ogSetKeyDownFunction(context_id, cbfKeyDown)
Set callback function for key down event
callback_function(context_id, keycode)

3. Define Points of Interest: Exercise

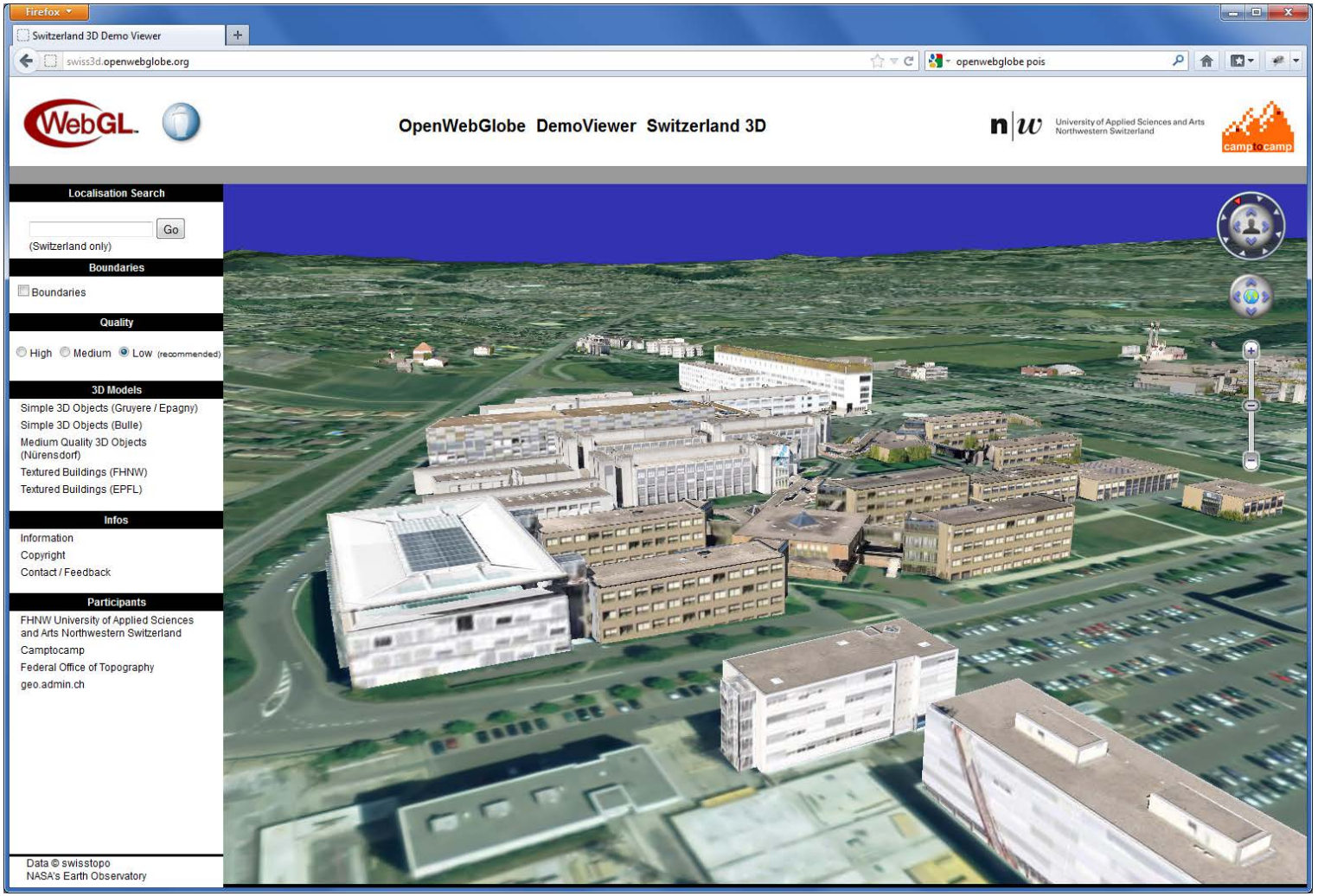
1. Use jsFiddler or your own IDE to create POIs

http://jsbin.com/owg_poi

Please clone source code before start working

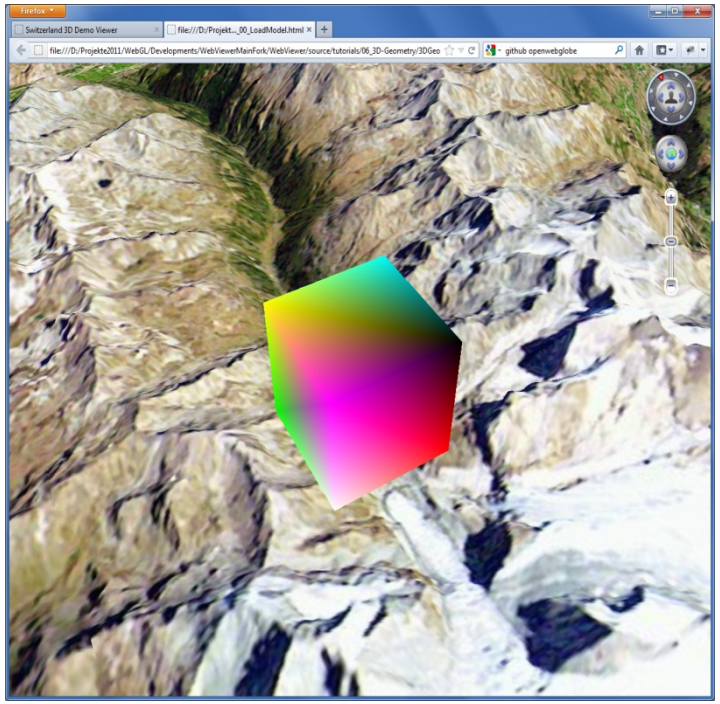
2. Try out POI picking: change the POI size on click

4. 3d models


Firefox
Switzerland 3D Demo Viewer
swiss3d.openwebglobe.org
openwebglobe pois
WebGL
OpenWebGlobe DemoViewer Switzerland 3D
University of Applied Sciences and Arts Northwestern Switzerland
camptocamp
Localisation Search
Go
(Switzerland only)
Boundaries
Boundaries
Quality
High Medium Low (recommended)
3D Models
Simple 3D Objects (Gruyere / Epagny)
Simple 3D Objects (Bulle)
Medium Quality 3D Objects (Nürensdorf)
Textured Buildings (FHNW)
Textured Buildings (EPFL)
Infos
Information
Copyright
Contact / Feedback
Participants
FHNW University of Applied Sciences and Arts Northwestern Switzerland
Camptocamp
Federal Office of Topography
geo.admin.ch
Data © swisstopo
NASA's Earth Observatory

4. 3d models: The OpenWebGlobe Model Description

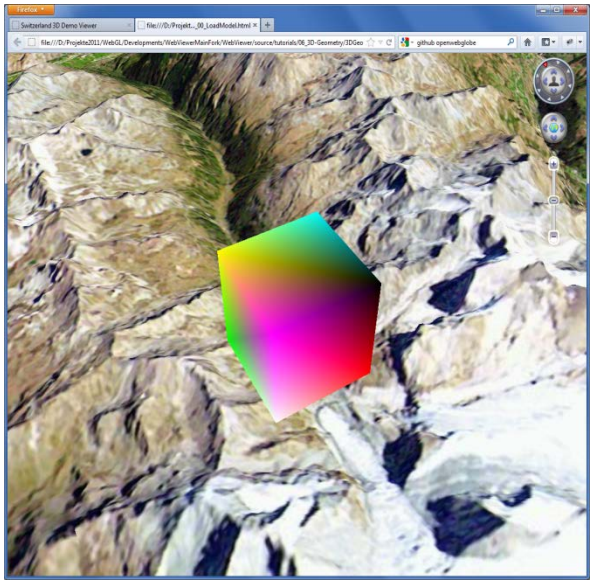
```
[
   [
    {
      "id"                 :  "1",
      "Center"             :  [7.438637,46.951081,3400],
      "VertexSemantic":  "pc",
      "Vertices"  :  [      -100,-100,-100,1.0,1.0,1.0,0.0,
                            -100,-100, 100,1.0,0.0,0.0,0.2,
                             100,-100,-100,0.0,1.0,0.0,0.2,
                             …
                            -100, 100,-100,1.0,0.0,1.0,1.0,
                            -100, 100, 100,0.0,0.0,0.0,1.0
                            ],
      "IndexSemantic"  :  "TRIANGLES",
      "Indices"  :  [ 3,1,0,
                      5,3,2,
                            …
                      3,5,1,
                      4,2,6]
    }
   ]
 ]
```



4. 3d models: Import a model

```
//create a geometry layer
geometrylayer = ogCreateGeometryLayer(world,"buildings");

//load the 3d model
var geometry1 = ogLoadGeometryAsync(geometrylayer,"cubepc.json");
```



4. 3d models: Model related functions

ogCreateGeometryLayer(world_id, layername)
Create a Geometry Layer Object

ogHideGeometryLayer(layer_id)
Hide GeometryLayer

ogShowGeometryLayer(layer_id)
Show hidden GeometryLayer

ogCreateGeometry(layer_id, jsonobject)
Create a Geometry Object

ogLoadGeometryAsync(layer_id, url)
Load a geometry from a JSON url

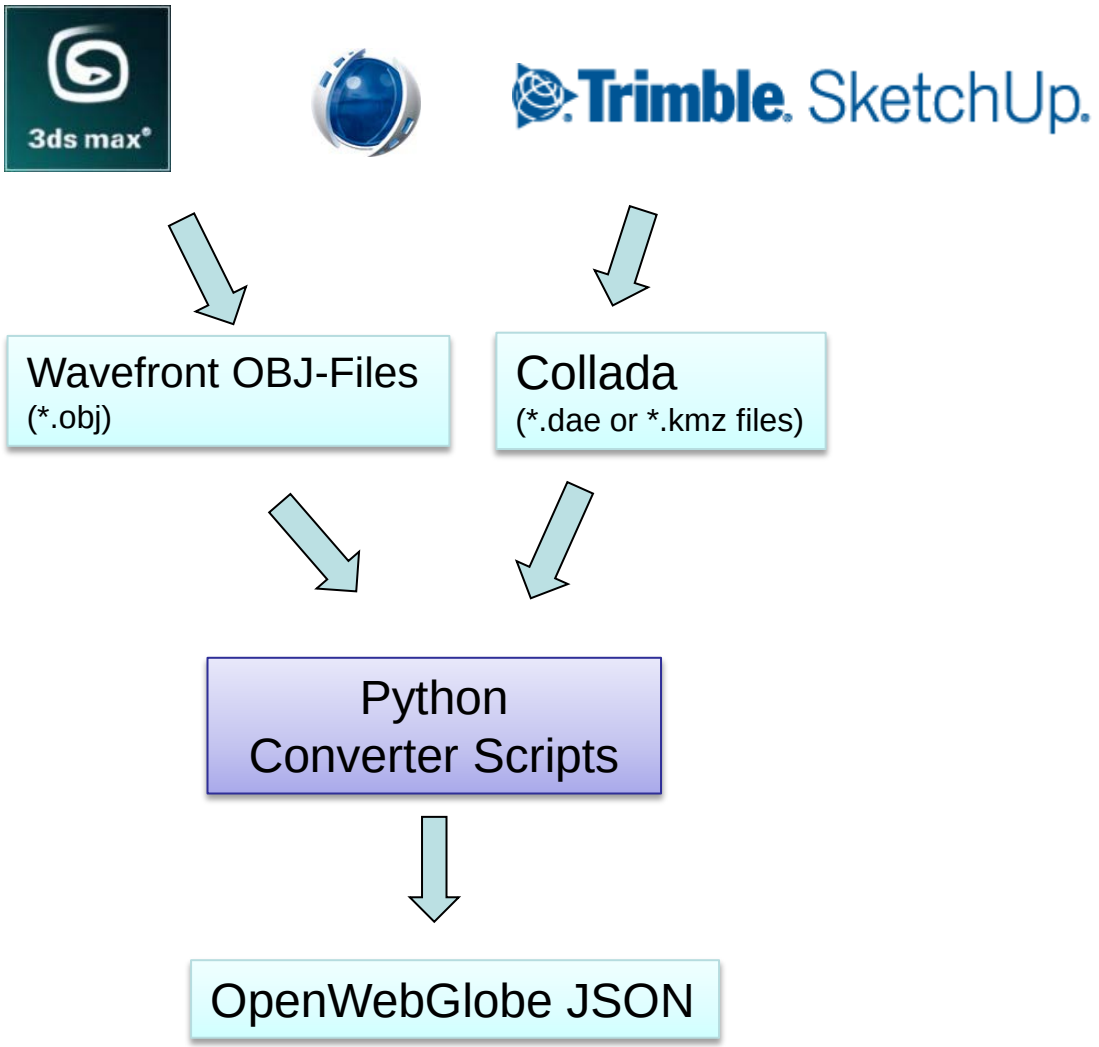
ogHighlightGeometry(geometry_id, r, g, b, a)
Highlight a geometry

ogSetGeometryPositionWGS84(geometry_id, lng, lat, elv, yaw, pitch, roll)

ogPickGeometry(scene_id, mx, my)
Returns the id of a picked geometry or -1

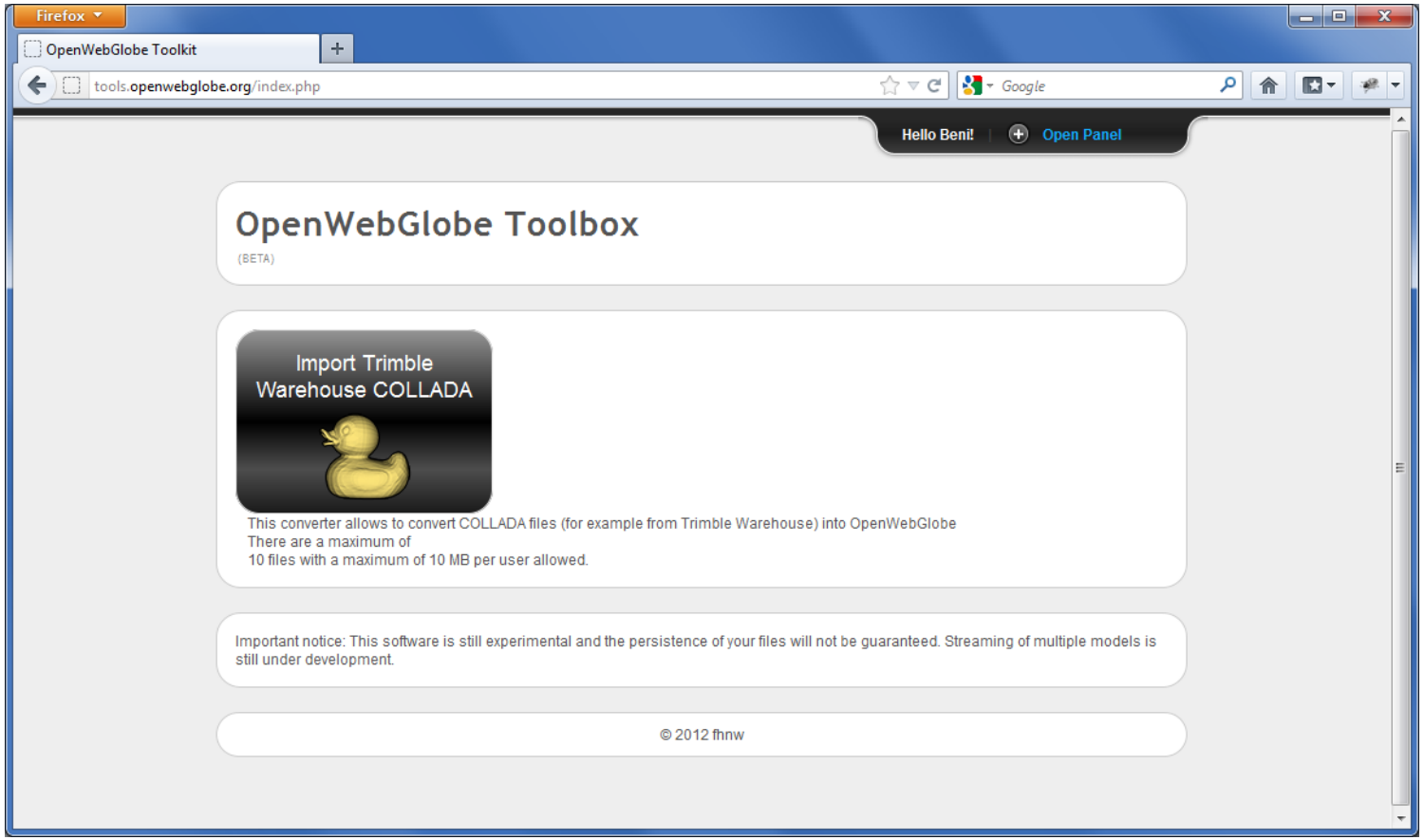
4. 3d models: ogHighlightGeometry

4. 3d models: Convert a 3d model



4. 3d models: Convert a 3d model

http://tools.openwebglobe.org



3. Models Exercise: Exercise

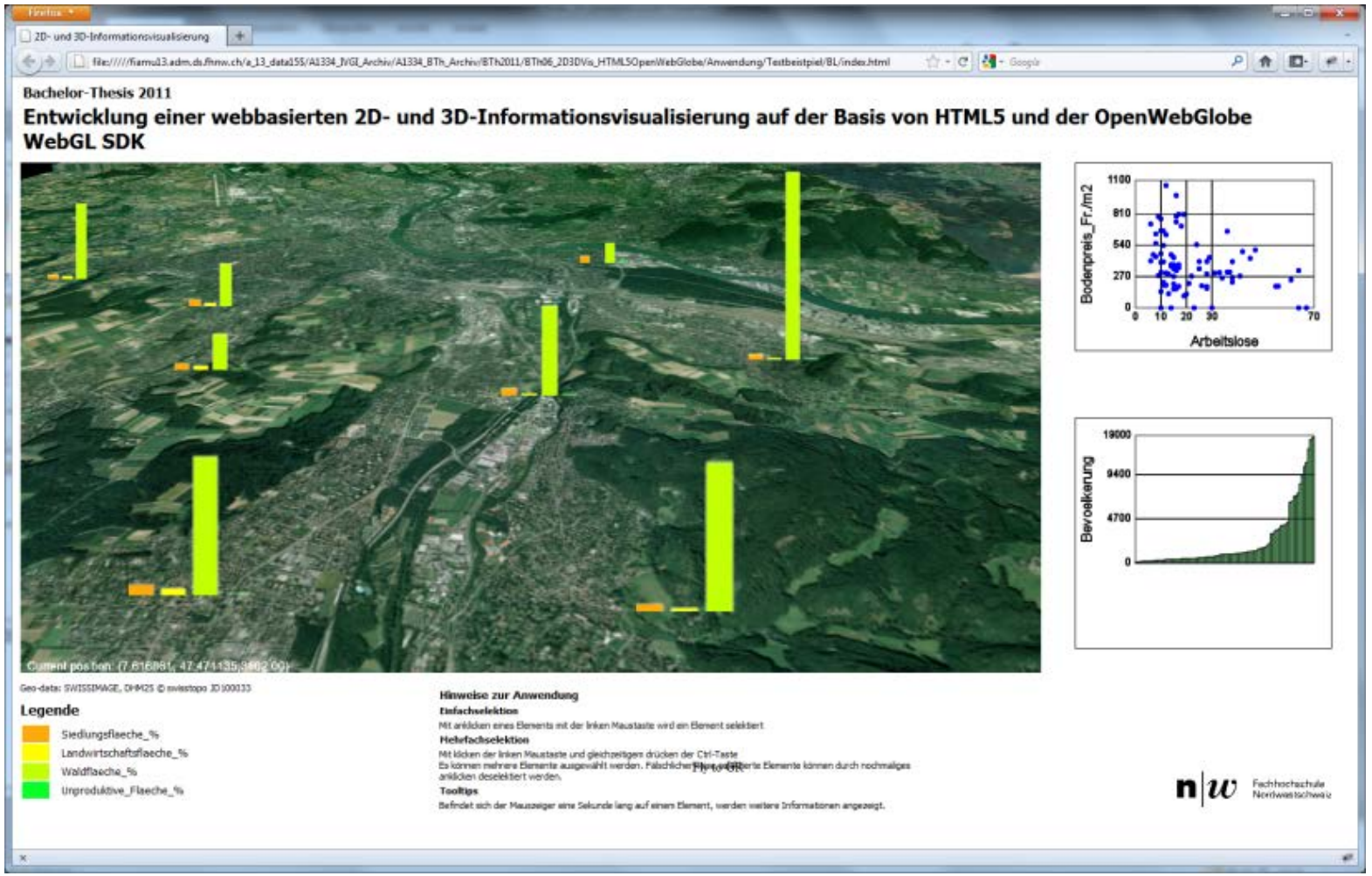
1. **Convert a 3d Model from 3d Warehouse**

   http://sketchup.google.com/3dwarehouse/
   http://tools.openwebglobe.org

1. **Watch the example html file you get from the conversion**

2. **Write code to highlight the model as soon you click on it**

5. Integrate dynamic content



5. Integrate dynamic content: Related Functions

ogCreateBillboardLayer(world_id, layername)
Creates a Billboard Layer

ogShowBillboardLayer(billboardlayer_id)
Shows a Billboard Layer

ogHideBillboardLayer(billboardlayer_id)
Hides a Billboard Layer

ogCreateBillboardFromCanvas(layer_id, canvas, lng, lat, elv)
Creates a Billboard using a HTML5 canvas element as content

ogUpdateBillboard(billboard_id, canvas)
updates the content of the billboard

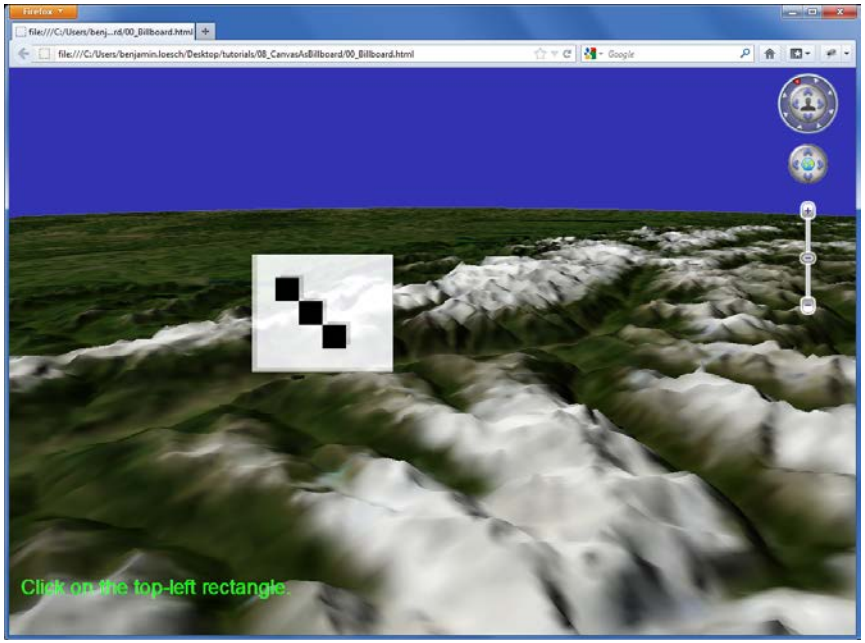
ogShowBillboard(billboard_id)
shows the billboard

ogHideBillboard(billboard_id)
hides the billboard

ogPickBillboard(mx, my, my)
returns an object if a billboard is picked

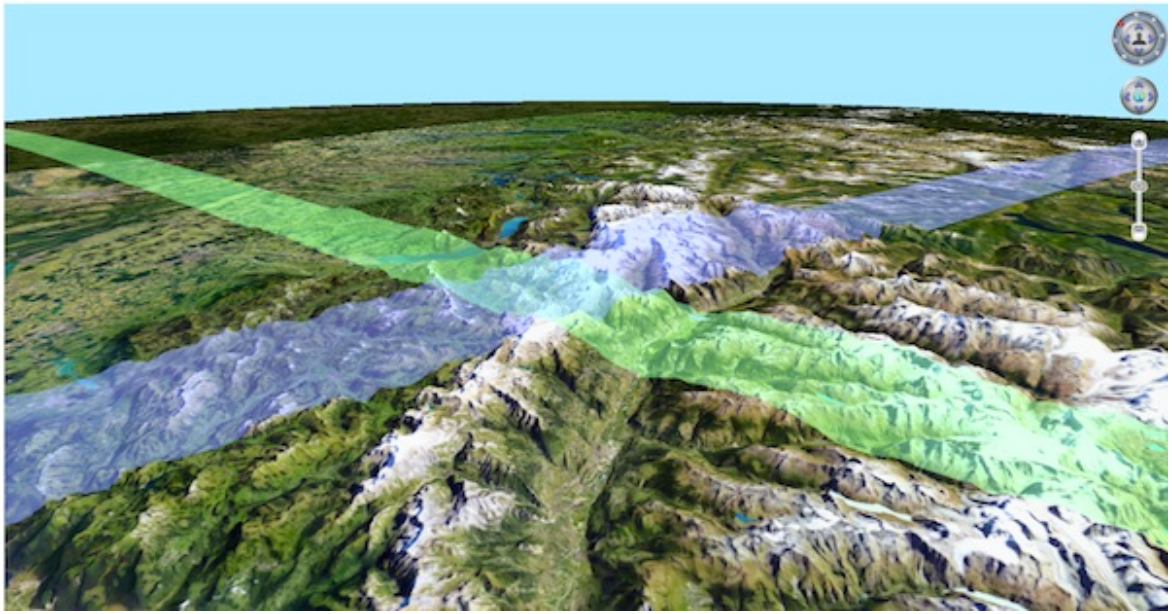
5. Integrate dynamic content: Demos

- A Simple Canvas & Picking Demo
- Paint into the Canvas
- Videos on a Canvas



What we left out

- Data Processing - https://github.com/OpenWebGlobe/DataProcessing
- Vector Layer (GeoJSON Support)

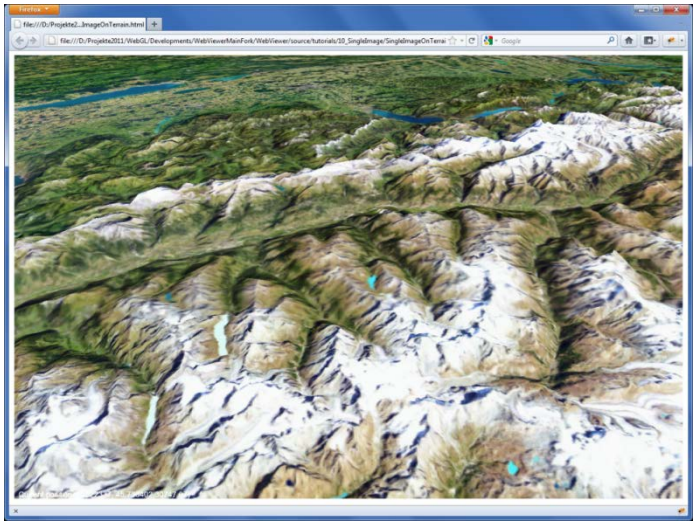


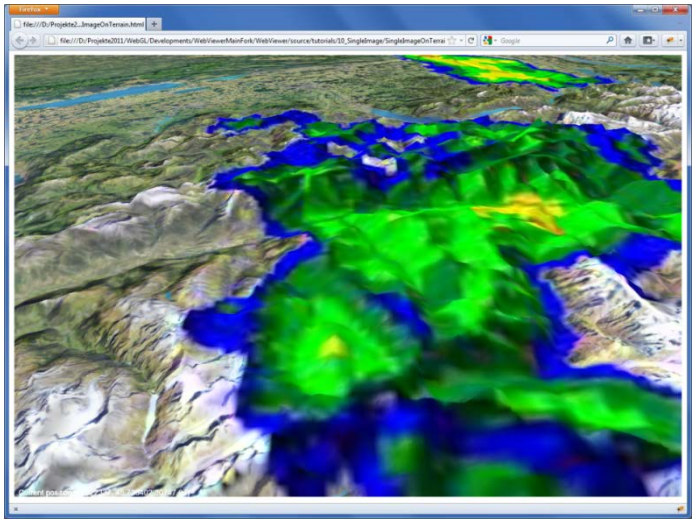
What we left out

- Data Processing - https://github.com/OpenWebGlobe/DataProcessing
- Vector Layer (GeoJSON Support)
- PointClouds

What we left out

- Data Processing - https://github.com/OpenWebGlobe/DataProcessing
- Vector Layer (GeoJSON Support)
- PointClouds
- Image Overlays

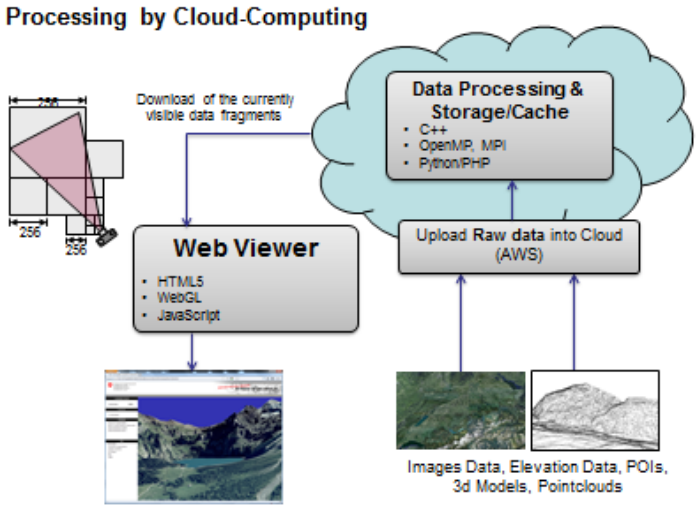




OpenWebGlobe – What's next ?

Data Processing

- Processing by Cloud-Computing
- Extension of the conversion tools at **tools.openwebglobe.org**
- Streaming of contents (3d models, POIs, Rich Point Clouds)
- Custom tile generation service



OpenWebGlobe – Whats next ?

WebViewer

- Streaming of contents (3d models, POIs, Point Clouds)
- Further work on Navigation

If you want to know more …

www.openwebglobe.org

Other useful functions

ogToCartesian(scene_id, lng, lat, elv)
Transforms WGS84 to geocentric cartesian coordinates.

ogToWGS84(scene_id, x, y, z)
Transforms geocentric cartesian to WGS84 coordinates.

ogWorldToWindow(scene_id, x, y, z)
get screen coordinates from cartesian world coordinates

ogCalcDistanceWGS84(lng0, lat0, lng1, lat1)
Calculate metrical distance between to points in WGS84

ogLockNavigation(scene_id)
Lock Navigation

ogUnlockNavigation(scene_id)
Lock Navigation

Other useful functions

ogSetBackgroundColor(context_id, r, g, b, a)
Set background color

ogSetRenderQuality(world_id, quality)
Set Render Quality of world.

ogPickGlobe(scene_id, mx, my)
Pick globe.

ogBlitTexture(texture_id, x, y, opt_options)
Blit texture to screen (2D)

ogDrawText(context_id, text, x, y)
Blits a text on screen (2D)