Processing Geo-Data using the OpenWebGlobe Tools

Martin Christen
martin.christen@fhnw.ch

Version 0.9.1

FHNW University of Applied Sciences and Arts Northwestern Switzerland

Contents

Abstract

The OpenWebGlobe data processing algorithms have been developed with a focus on scalability to very large data volumes for all supported data types, including imagery, map and terrain data. We adapted the algorithms to support as many cores as possible and came up with a set of OpenWebGlobe processing commands.

All commands run on normal computers (regular laptops and work stations) and on high performance compute clusters (HPCC), including cloud services.

One important aspect is compatibility to other services, therefore we choose to use the OpenStreetMap tile layout for our tile storage. This makes it very easy to also use our processed image data in 2D-Applications like OpenLayers.

1 Introduction

The OpenWebGlobe processing tools are considered beta. The tools are still work in progress and being actively developed. In section 7 you will find planned features for future releases. This is the documentation for the OpenWebGlobe processing tools version 0.9.1.

This documentation is work in progress. The latest version of this documentation is available at `https://github.com/downloads/OpenWebGlobe/DataProcessing/dataprocessing.pdf`.

1.1 Why Data Processing is Required

A virtual globe can have several data categories such as image data, elevation data, points of interest, vector data, 3D objects, and point clouds. Before streaming over the internet this data must be preprocessed. This preprocessing usually comprises a transformation from a local to a global reference system, creation of pyramid layers or level of detail, tiling of the data, and optionally compression and encryption of the data.[1]

Because visualization may consist of terabytes to petabytes of orthophoto and elevation data, out-of-core rendering with a level of detail approach must be used. Data can be streamed over the internet, a local network or a local hard drive.[1]

1.2 Quadtree structure

OpenWebGlobe tiles are indexed using quadtree keys. Each quadkey number identifies a single tile at a single zoom level.

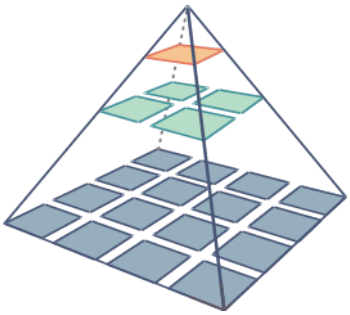

Figure 1: Zoom Levels for Level of Detail

In OpenWebGlobe the Mercator projection[6] is used to map image and elevation data to a square area. The Mercator projection is mainly used to minimize distortions in processed images and elevation data.

The maximum latitude is chosen so that the resulting map fits into a square. For the spherical Mercator projection this maximum latitude is approximately 85.05 degrees.

Tiles can be accessed using **tile coordinates**.

This projection and tile system is also used by other popular web maps like Google Maps[3], Bing Maps[4] or OpenStreetMap[5].

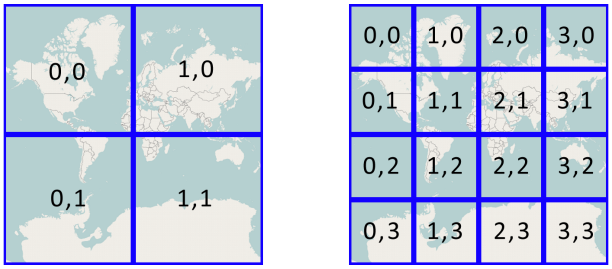

Figure 2: Tile Coordinates at different zoom levels

1.3 Rendering on Ellipsoid

An ellipsoidal geodetic reference model is required, in order to minimise geometric transformation errors and to enable position accuracies within the Virtual Globe at the submeter level.[1] The default spatial reference system in OpenWebGlobe is WGS84, therefore the globe is rendered using the semi major axis $a = 6378137.0$ and the semi minor axis $b = 6356752.314245$. It is possible to change those parameters in the visualization SDK from version 0.9.

2 Processing Workflow - General Data Processing

A general data processing workflow has been created[2] to simplify data processing from small to very large datasets.

1. The first step in data processing is knowing the extent of your dataset. You can use the tool called "ogCalcExtent" to calculate the extent of your data in WGS84 and tile coordinates. This is described in section 2.1.

2. Once you know the extent of your data you can create a new layer. This is done using the "ogCreateLayer" tool. You find more information in section 2.2.

3. After creation of the layer you can start adding data. This is done using the "ogAddData" tool, as shown in section 2.3.

4. If you create elevation data you have to use the "ogTriangulate" tool at this step to create geometry from the previously added data. You learn more about this tool in section 2.5. For image data this step is not required.

5. When you are finished adding data, level of details must be calculated. This can be done using the "ogResample" tool, as shown in section 2.6.

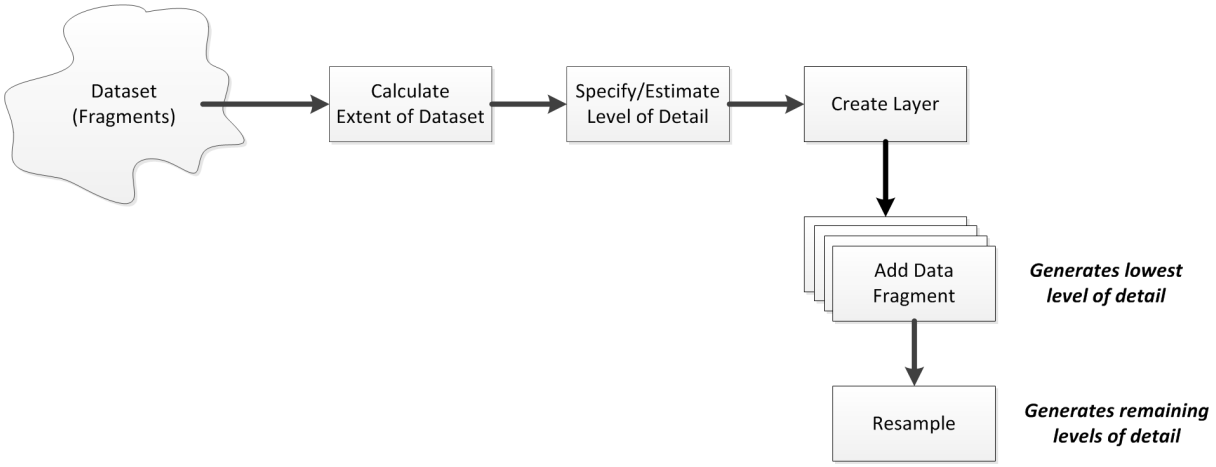

Figure 3: Workflow for data processing

2.1 Calculate Extent - ogCalcExtent

We assume data is given in fragments. For example an image data may consist of 24 GeoTiff images, each around 200 MB (orthophoto mosaic). The data has an rectangular extent in WGS84 coordinates which can be converted to tile coordinates.

ogCalcExtent is a convenience function. It is not really required for processing but it helps retrieving the tile boundary and zoom level of your dataset.

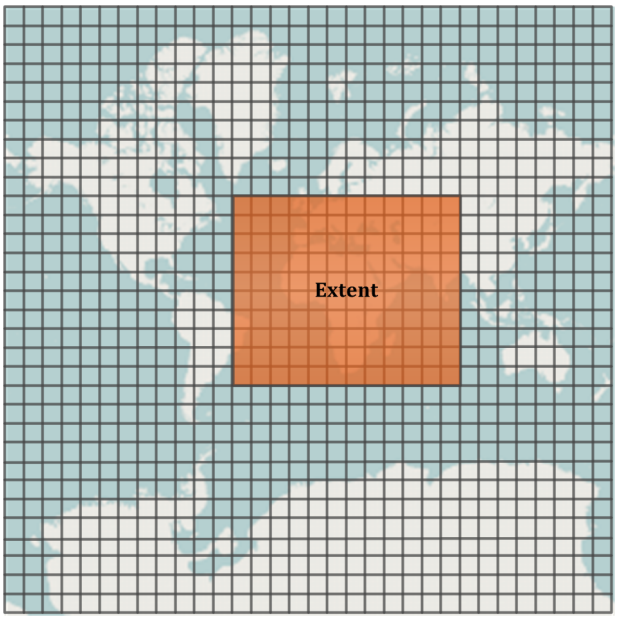

Figure 4: Example of an extent in tile coordinates

I recommend trying the step by step tutorial in section 4. This is probably the best way to get started.

2.1.1 Case 1: Calculating Tile Extent from WGS84

Option	Description
–maxlod	level of detail (zoom level) for tile coordinates
–wgs84	wgs84 coordinates in the form: lng_0 lat_0 lng_1 lat_1 (with $lng_1 > lng_0$ and $lat_1 > lat_0$)

Table 1: Command Arguments when tile coordinates from WGS84

Example:

```
ogCalcExtent --maxlod 15 --wgs84 7.6 45.0 8.1 46.1
```

Result:

```
Tile Coords: (17075, 11644)-(17121, 11787)
```

2.1.2 Case 2: Calculating Tile Extent from a List of Files

Option	Description
–srs	spatial reference system (in the form EPSG:xxxx)
–input	space separated list of files

Table 2: Command Arguments for Calculating Extent using a List of Files

Example:

```
ogCalcExtent --srs EPSG:21781 --input C:/data/myData00.tif C:/data/myData01.tif
```

2.1.3 Case 3: Calculating Tile Extent from Files in a Directory

This is the most common usage of the ogCalcExtent tool if all input files are in the same directory.

Option	Description
–srs	spatial reference system (in the form EPSG:xxxx)
–inputdir	space separated list of files
–filetype	the file extension of the input files

Table 3: Command Arguments for Calculating Extent from Files in a Directory

```
ogCalcExtent --srs EPSG:21781 --inputdir C:/data/mydatset/ --filetype tif
```

2.1.4 Level of Detail

The maximum level of detail (also called zoom level) must be specified when creating a layer. The tool "ogCalcExtent" gives you a recommended value. The higher the level of detail, the more files are generated and the processing will be slower. In table 4 you can see the pixel resolution at different level of detail. Be aware the resolution changes along latitude.

lod	Resolution [m]
1	78272
2	39136
3	19568
4	9784
5	4892
6	2446
7	1223
8	611
9	306
10	152
11	76
12	38
13	19
14	9.6
15	4.8
16	2.39
17	1.19
18	0.6
19	0.3
20	0.151
21	0.074
22	0.037
23	0.019

Table 4: Level of Detail Pixel Resolution at Latitude 0 (for image data)

2.2 Create Layer - ogCreateLayer

You can create a new layer using the ogCreateLayer function. This must be done for each dataset.

Option	Description
–name	name of the layer (ASCII, no special chars)
–lod	desired level of detail
–extent	tile boundary tx_0 ty_0 tx_1 ty_1 for elevation/image data. The 4 values are space separated and $tx_0 < tx_1$ and $ty_0 < ty_1$
–type	The type of layer. This can be "image" or "elevation". (More types will be added in future releases).
–force	[optional] If a dataset with same name already exists, it will be deleted and recreated.
–numthreads	[optional] Specify number of threads used to create the layer. This should be the number of cores of your CPU. In most cases this is not important as creating a new layer is a fast operation.

Table 5: Command Arguments for Creating a New Layer

2.3 Adding Data - ogAddData

2.3.1 Adding Image Data

Image Data can be added using the "ogAddData" tool. An example how to use this is provided in the tutorial in section 4. The possible parameters are shown in table 6.

Option	Description
–image [filename]	Use this flag when adding image data
–layer [layername]	Name of the image layer previously created using ogCreateLayer.
–srs [srsid]	spatial reference system of the image file, in the form EPSG:xxxxx.
–overwrite , –fill	use –overwrite to overwrite existing parts of the layer, use –fill to fill empty regions only with new data. In most cases you want to use the –fill option. When updating new data you would use –overwrite
–numthreads [num]	[optional] Specify number of threads used to add the data. This should be the number of cores of your CPU.

Table 6: Adding Image Data

2.3.2 Adding Elevation Data

Adding elevation data is basically the same like adding image data. The pararameters for adding elevation are shown in table 7. An example of adding elevation data is found in the tutorial in section 4.

Option	Description
–elevation [filename]	Specify the elevation file to be added
–layer [layername]	Name of the elevation layer previously created using ogCreateLayer.
–srs [srsid]	spatial reference system of the elevation file, in the form EPSG:xxxxx.
–overwrite , –fill	use –overwrite to overwrite existing parts of the layer, use –fill to fill empty regions only with new data. In most cases you want to use the –fill option. When updating new data you would use –overwrite
–numthreads [num]	[optional] Specify number of threads used to add the data. This should be the number of cores of your CPU.

Table 7: Adding Elevation Data

2.4 Adding Data in a Compute Cluster

To speed up processing very large datasets, ogAddData can be executed on different computers at the same time. Each Data Fragment must be added from a different node in your system.

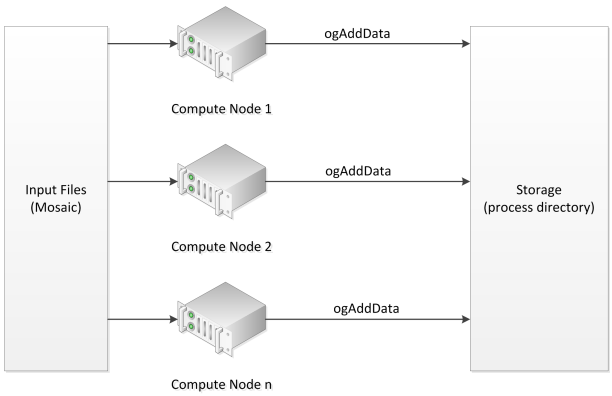

Figure 5: Calling ogAddData from different compute nodes

2.5 Triangulating Elevation Data - ogTriangulate

When adding elevation data is finished the data must be triangulated to create the 3D-Geometry JSON files. This is done using the "ogTriangulate" tool. Currently the tool

creates triangulated tiles using Delaunay Triangulation. In future releases it will be possible to choose other triangulation or raster algorithms.

Option	Description
–layer [layername]	Name of the elevation layer previously created using ogCreateLayer.
–triangulate	Use delaunay triangulation algorithm.
–maxpoints	[optional] specify the maximum allowed points per tile. The default value is 512. In most cases values should be between 256 and 512.
–numthreads [num]	[optional] Specify number of threads used for triangulation.

Table 8: Triangulating

2.6 Resampling - ogResample

The tools ogAddData and ogTriangulate only calculate the maximum zoom level. ogResample is a tool for calculating the remaining zoom levels.

2.6.1 Resampling Image Data

Option	Description
–layer [layername]	Name of the image layer.
–type [layertype]	use "image" here.
–numthreads [num]	[optional] Specify number of threads used for resampling.

Table 9: Parameters for Image Resampling

2.6.2 Resampling Elevation Data

Option	Description
–layer [layername]	Name of the image layer.
–type [layertype]	use "elevation" here.
–numpoints [num]	[optional] specify the maximum allowed points per tile. The default value is 512. In most cases values should be between 256 and 512.
–numthreads [num]	[optional] Specify number of threads used for resampling.

Table 10: Parameters for Elevation Resampling

3 Installing the Tools

3.1 Downloading prebuilt packages

Prebuilt packages are currently only available for the Windows platform. Because the processing tools have many dependencies to other software, building it yourself is very time consuming.

3.2 Configuring the Tools

Configuration is done in an XML file (setup.xml) where you specify the directory where data will be processed and a directory for logging. setup.xml is located in

```
C:\Program Files (x86)\OpenWebGlobeProcessing\setup.xml
```

If you don't want to modify the included setup.xml simply type:

```
cd "C:\Program Files (x86)\OpenWebGlobeProcessing"
mkdir process
mkdir log
```

If you want to edit the XML:

```
<ProcessingSettings>
   <processpath>c:/data/bla/</processpath>
   <logpath>c:/logs/</logpath>
</ProcessingSettings>
```

The "processpath" must point to an **existing directory** where data will be written (high performance storage)

The "logpath" is the path where log files are written. It must point to an existing directory.

4 Tutorial: Image and Elevation Data Processing

In this tutorial an image and an elevation dataset will be processed and put on the globe. Along with the processing tools there is an example dataset of the Bugaboo Provincial Park (British Columbia, Canada) which was produced by Tyler Mitchell and Will Cadell, Timberline Forest Inventory Consultants, British Columbia, Canada. From DEM and a fused Landsat texture based on data from http://geobase.ca. This dataset is freely available, therefore it is copied as a tutorial dataset along with the OpenWebGlobe data processing utilities.

The image data is in "data/bugaboo/bugaboo_img.jgw" and the elevation data is located in "data/bugaboo/bugaboo_dem.tif".

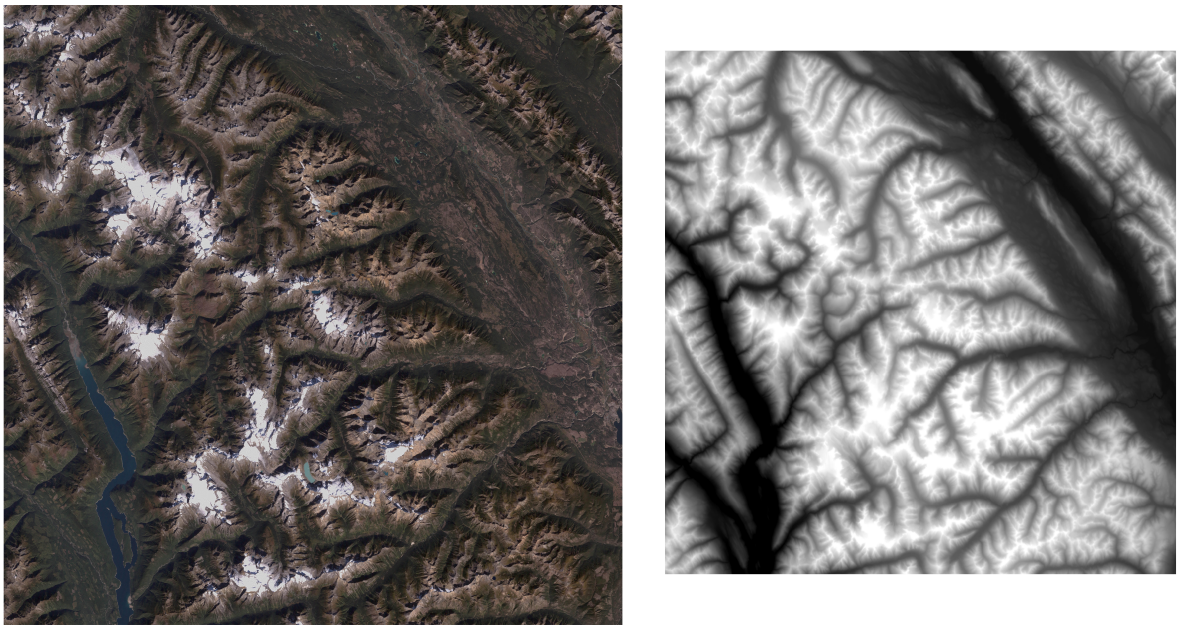

Figure 6: Image and Elevation Data of Bugaboo Provincial Park.

4.1 Processing the Image Data

4.1.1 Calculating Extent

The example image data is in NAD27 / UTM zone 11N projection, which corresponds to EPSG code 26711. The image data must be in a supported raster data format with 3 channels (RGB). Refer to table 14 for all supported formats.

```
ogCalcExtent --srs EPSG:26711 --inputdir data\bugaboos\ --filetype jpg
```

Option	Description
–srs	spatial reference system. EPSG:26711 is used here.
–inputdir	the directory where data is located. For this example a relative path is used and pointing to the included demo data.
–filetype	The file type is jpg. A correspoinding .jgw file is present too.

Table 11: Parameters of ogCalcExtent used in this example

The output of this will be

```
SRS epsg-code: 26711
[OK]   data\bugaboos\bugaboo_img.jpg
GATHERED BOUNDARY (Mercator):
        ulx: -0.6508961467967627
        lry: 0.3239096059373637
        lrx: -0.6444645967523146
        uly: 0.330477474424119
BOUNDARY in WGS84:
       lng0: -117.1613064234173
       lat0: 50.25377317536166
       lng1: -116.0036274154166
       lat1: 51.00368256032537
 pixelsize: 9.882724674077332 m
LEVEL OF DETAIL 1: Tile Coords: (0, 0)-(0, 0)
LEVEL OF DETAIL 2: Tile Coords: (0, 1)-(0, 1)
LEVEL OF DETAIL 3: Tile Coords: (1, 2)-(1, 2)
LEVEL OF DETAIL 4: Tile Coords: (2, 5)-(2, 5)
LEVEL OF DETAIL 5: Tile Coords: (5, 10)-(5, 10)
LEVEL OF DETAIL 6: Tile Coords: (11, 21)-(11, 21)
LEVEL OF DETAIL 7: Tile Coords: (22, 42)-(22, 43)
LEVEL OF DETAIL 8: Tile Coords: (44, 85)-(45, 86)
LEVEL OF DETAIL 9: Tile Coords: (89, 171)-(91, 173)
LEVEL OF DETAIL 10: Tile Coords: (178, 342)-(182, 346)
LEVEL OF DETAIL 11: Tile Coords: (357, 685)-(364, 692)
LEVEL OF DETAIL 12: Tile Coords: (714, 1371)-(728, 1384)
LEVEL OF DETAIL 13: Tile Coords: (1429, 2742)-(1456, 2769)
LEVEL OF DETAIL 14: Tile Coords: (2859, 5484)-(2912, 5538)
LEVEL OF DETAIL 15: Tile Coords: (5719, 10969)-(5825, 11077)
LEVEL OF DETAIL 16: Tile Coords: (11439, 21938)-(11650, 22154)
LEVEL OF DETAIL 17: Tile Coords: (22878, 43877)-(23300, 44308)
LEVEL OF DETAIL 18: Tile Coords: (45757, 87755)-(46600, 88616)
LEVEL OF DETAIL 19: Tile Coords: (91515, 175511)-(93201, 177233)
LEVEL OF DETAIL 20: Tile Coords: (183030, 351022)-(186402, 354466)
LEVEL OF DETAIL 21: Tile Coords: (366061, 702045)-(372805, 708932)
LEVEL OF DETAIL 22: Tile Coords: (732123, 1404090)-(745611, 1417864)
**************************************
RECOMMENDATION (MINIMUM LOD):
IF THIS IS ELEVATION: LOD=17: Tile Coords: (22878, 43877)-(23300, 44308)

IF THIS IS IMAGE: LOD=13: Tile Coords: (1429, 2742)-(1456, 2769)
**************************************
```

4.1.2 Creating the Image Layer

The recommended zoom levels would be 13, but we want to have a little bit more and choose 16. For zoom level 16 the tile coords are 11439 21938 11650 22154. With this information we can create the image layer.

```
ogCreateLayer --name bugaboos --lod 16 --extent 11439 21938 11650 22154 --type image
              --force
```

Option	Description
–name	The name of the layer is simply called ”bugaboos”. This will be the directory name inside of the ”process” dir.
–lod	we choose the maximum level of detail 16
–extent	the extent at lod 16, previously calculated with ogCalcExtent.
–type	this is an image layer, so we use ”image”.
–force	if we already have a ”bugaboos” layer it will be deleted. Be careful with this option!

Table 12: Parameters of ogCreateLayer used in this example

The ogCreateLayer command will output the following:

```
[date_time]: Creating all required subdirectories...
0212130230321131
0212132130002030
[date_time]: creating LOD directory: process/bugaboos/tiles/1
[date_time]: creating LOD directory: process/bugaboos/tiles/2
[date_time]: creating LOD directory: process/bugaboos/tiles/3
[date_time]: creating LOD directory: process/bugaboos/tiles/4
[date_time]: creating LOD directory: process/bugaboos/tiles/5
[date_time]: creating LOD directory: process/bugaboos/tiles/6
[date_time]: creating LOD directory: process/bugaboos/tiles/7
[date_time]: creating LOD directory: process/bugaboos/tiles/8
[date_time]: creating LOD directory: process/bugaboos/tiles/9
[date_time]: creating LOD directory: process/bugaboos/tiles/10
[date_time]: creating LOD directory: process/bugaboos/tiles/11
[date_time]: creating LOD directory: process/bugaboos/tiles/12
[date_time]: creating LOD directory: process/bugaboos/tiles/13
[date_time]: creating LOD directory: process/bugaboos/tiles/14
[date_time]: creating LOD directory: process/bugaboos/tiles/15
[date_time]: creating LOD directory: process/bugaboos/tiles/16
[date_time]: calculated in: 0.406 s

[date_time]: All required subdirectories created...
```

in your processing directory you will find the tile structure and the file ”layersettings.json”, which contains the following:

```
{
   "name" : "bugaboos",
   "type" : "image",
   "format" : "png",
   "maxlod" : 16,
   "extent" : [11439, 21938, 11650, 22154]
}
```

At this time all directories are empty, as data has not yet been added.

4.1.3 Adding Data

Now we are ready to add the image data. This is done using:

```
ogAddData --numthreads 4 --layer bugaboos --image data\bugaboos\bugaboo_img.jpg
          --srs EPSG:26711 --fill
```

If you have several image mosaics you would call ogAddData for each mosaic.
This operation is time consuming. The result look like this:

```
[datetime]: Logging started
[datetime]: Forcing number of threads to 4
[datetime]: calculated in: 522.771 s
```

At this time the lowest level of image data is finished processing. There are around 45000 tiles on this harddisk now (211x216). On my laptop with a quad-code CPU (Intel Core i7-2720QM @ 2.2 GHz) a tile was generated in 0.01 seconds (87 tiles per second).

The OpenWebGlobe processing tools are optimized for multiprocessing, therefore it is interesting to see what happens when changing the number of threads. Of course this was done after deleting all files and running the processing again using 1,2 and 3 threads. The results are listed in table 13.

Threads	Total Time [s]	Tiles per Second
1	1139.3	40
2	671.1	71
3	534.7	85
4	522.8	87

Table 13: Processing the data with 1 to 4 threads on an Intel Core i7-2720QM @ 2.2 GHz

4.1.4 Resampling

The next step is calculating the remaining zoom levels. This can be done using the "ogResample" tool.

```
ogResample --layer bugaboos --type image --verbose
```

In the directory "processed/bugaboos/tiles/" you can look at the tiles. Each zoom level has its own directory.

```
[datetime]: Logging started
[datetime]:
Resample Setup (Image Layer):
     name = bugaboos
   maxlod = 16
   extent = 11439, 21938, 11650, 22154

[datetime]: Processing Level of Detail 15
[datetime]: Processing Level of Detail 14
[datetime]: Processing Level of Detail 13
[datetime]: Processing Level of Detail 12
[datetime]: Processing Level of Detail 11
[datetime]: Processing Level of Detail 10
[datetime]: Processing Level of Detail 9
[datetime]: Processing Level of Detail 8
[datetime]: Processing Level of Detail 7
[datetime]: Processing Level of Detail 6
[datetime]: Processing Level of Detail 5
[datetime]: Processing Level of Detail 4
[datetime]: Processing Level of Detail 3
```

```
[datetime]: Processing Level of Detail 2
[datetime]: Processing Level of Detail 1
[datetime]: calculated in: 823.982 s
```

Now the tiles are finished processing.

4.2 Processing the Elevation Data

The workflow for processing elevation data is pretty much the same like the one for images. The exception is that elevation data must be converted to geometry and therefore a triangulation step is necessary.

4.2.1 Calculating Extent

```
ogcalcextent.exe --verbose --srs EPSG:26711 --inputdir data\bugaboos\ --filetype tif
```

```
SRS epsg-code: 26711
[OK]   data\bugaboos\bugaboo_dem.tif
GATHERED BOUNDARY (Mercator):
        ulx: -0.6508957532005574
        lry: 0.3239035952061771
        lrx: -0.6444587127773991
        uly: 0.3304770724421868
BOUNDARY in WGS84:
       lng0: -117.1612355761003
       lat0: 50.25308139584989
       lng1: -116.0025682999319
       lat1: 51.00363702834048
 pixelsize: 39.53005418648766 m
LEVEL OF DETAIL 1: Tile Coords: (0, 0)-(0, 0)
LEVEL OF DETAIL 2: Tile Coords: (0, 1)-(0, 1)
LEVEL OF DETAIL 3: Tile Coords: (1, 2)-(1, 2)
LEVEL OF DETAIL 4: Tile Coords: (2, 5)-(2, 5)
LEVEL OF DETAIL 5: Tile Coords: (5, 10)-(5, 10)
LEVEL OF DETAIL 6: Tile Coords: (11, 21)-(11, 21)
LEVEL OF DETAIL 7: Tile Coords: (22, 42)-(22, 43)
LEVEL OF DETAIL 8: Tile Coords: (44, 85)-(45, 86)
LEVEL OF DETAIL 9: Tile Coords: (89, 171)-(91, 173)
LEVEL OF DETAIL 10: Tile Coords: (178, 342)-(182, 346)
LEVEL OF DETAIL 11: Tile Coords: (357, 685)-(364, 692)
LEVEL OF DETAIL 12: Tile Coords: (714, 1371)-(728, 1384)
LEVEL OF DETAIL 13: Tile Coords: (1429, 2742)-(1456, 2769)
LEVEL OF DETAIL 14: Tile Coords: (2859, 5484)-(2912, 5538)
LEVEL OF DETAIL 15: Tile Coords: (5719, 10969)-(5825, 11077)
LEVEL OF DETAIL 16: Tile Coords: (11439, 21938)-(11650, 22154)
LEVEL OF DETAIL 17: Tile Coords: (22878, 43877)-(23300, 44308)
LEVEL OF DETAIL 18: Tile Coords: (45757, 87755)-(46601, 88617)
LEVEL OF DETAIL 19: Tile Coords: (91515, 175511)-(93203, 177234)
LEVEL OF DETAIL 20: Tile Coords: (183031, 351022)-(186406, 354469)
LEVEL OF DETAIL 21: Tile Coords: (366062, 702045)-(372812, 708938)
LEVEL OF DETAIL 22: Tile Coords: (732124, 1404091)-(745624, 1417876)
**************************************
RECOMMENDATION (MINIMUM LOD):
IF THIS IS ELEVATION: LOD=15: Tile Coords: (5719, 10969)-(5825, 11077)

IF THIS IS IMAGE: LOD=11: Tile Coords: (357, 685)-(364, 692)
**************************************
```

4.2.2 Creating the Elevation Layer

again we choose level of detail 16. As seem from the output of ogCalcExtent the tile coordinates are 11439 21938 11650 22154. This is exactly the same extent as the images. Usually elevation data has different extents.

Now we create the layer using the following command:

```
ogCreateLayer --name bugabooselv --lod 16 --extent 11439 21938 11650 22154
              --type elevation --force
```

```
[datetime]: Logging started
[datetime]: Target directory: process/bugabooselv
[datetime]: Creating all required subdirectories...
0212130230321131
0212132130002030
[datetime]: creating LOD directory: process/bugabooselv/tiles/1
[datetime]: creating LOD directory: process/bugabooselv/tiles/2
[datetime]: creating LOD directory: process/bugabooselv/tiles/3
[datetime]: creating LOD directory: process/bugabooselv/tiles/4
[datetime]: creating LOD directory: process/bugabooselv/tiles/5
[datetime]: creating LOD directory: process/bugabooselv/tiles/6
[datetime]: creating LOD directory: process/bugabooselv/tiles/7
[datetime]: creating LOD directory: process/bugabooselv/tiles/8
[datetime]: creating LOD directory: process/bugabooselv/tiles/9
[datetime]: creating LOD directory: process/bugabooselv/tiles/10
[datetime]: creating LOD directory: process/bugabooselv/tiles/11
[datetime]: creating LOD directory: process/bugabooselv/tiles/12
[datetime]: creating LOD directory: process/bugabooselv/tiles/13
[datetime]: creating LOD directory: process/bugabooselv/tiles/14
[datetime]: creating LOD directory: process/bugabooselv/tiles/15
[datetime]: creating LOD directory: process/bugabooselv/tiles/16
[datetime]: calculated in: 0.171 s

[datetime]: All required subdirectories created...
```

4.2.3 Adding Dataset

```
ogAddData --numthreads 4 --layer bugabooselv --elevation data\bugaboos\bugaboo_dem.tif
          --srs EPSG:26711 --fill
```

This command projects the data to the Mercator projectation and sorts the data spatially. Tiles are stored on the harddisk. The result of the tool is:

```
[datetime]: Logging started
[datetime]: Forcing number of threads to 4
[datetime]: calculated in: 108.054 s
```

4.2.4 Triangulation

The next step is triangulating the lowest zoom level, in our case this is 16.

```
ogTriangulate --layer bugabooselv --verbose --triangulate --maxpoints 256 --numthreads 4
```

"–triangulate" means it uses the delaunay triangulate algorithm. Currently there are no other algormithms for creating the tile geometry. In future it is planned to have different algorithms to create the 3D-Geometry. "–maxpoints" specifies the maximal allowed number of points per tile. Good values are between 256 and 512.

This operation is CPU-intense and takes a while to finish.

```
[datetime]: Logging started
[datetime]: changing maxpoints to 256
[datetime]: Forcing number of threads to 4
[datetime]:
Elevation Layer:
     name = bugabooselv
   maxlod = 16
   extent = 11439, 21938, 11650, 22154

[datetime]:
Extent mercator:   extent = -0.650909, 0.323883, -0.64444, 0.330505

[datetime]: calculated in: 841.971 s
```

In the directory process/boogabooelv/tiles/16 you find the .json geometries for the level of detail 16.

4.2.5 Calculating remaining Zoom Levels

Now the remaining zoom levels, in our case 1-15 must be generated. Like for image data this is done using the ogResample tool.

```
ogResample --layer bugabooselv --verbose --type elevation --maxpoints 256
```

Once this is finished you can delete the temp directory (process/bugabooselv/temp).

4.3 Visualization in OpenWebGlobe

As mentioned earlier, the processed data is located in the "process" directory by default. Now you have to run a webserver which can access this directory. OpenWebGlobe uses WebGL for textures and it is highly recommended to set the HTTP Access-Control-Allow-Origin header to support cross domain textures.

If you use the Apache webserver you can add a .htaccess file with the following contents:

```
<IfModule mod_headers.c>
   Header set Access-Control-Allow-Origin *
</IfModule>
```

Now you can create html code. The NASA Blue Marble dataset is taken from openwebglobe.org. Of course you could process blue marble dataset yourself now.

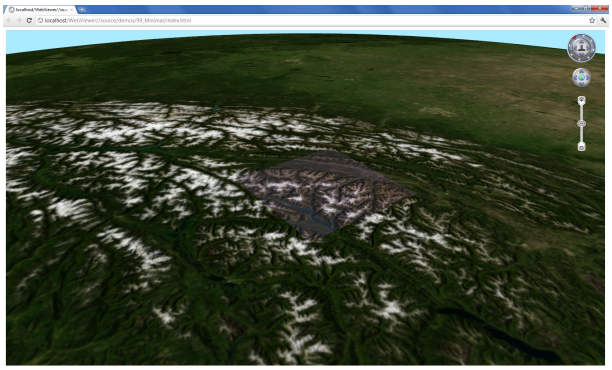

Figure 7: NASA Blue Marble layer combined with the bugaboos dataset

```
<html lang="en">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <head>
                <script type="text/javascript" src="openwebglobe-0.9.0.js"></script>
        </head>
        <body style="overflow:hidden;">
                <div style="width: 100%; height: 100%;">
                        <canvas id="canvas">
                        </canvas>
                </div>
        <script type="text/javascript">
        ogSetArtworkDirectory("http://www.openwebglobe.org/art/");
                var context = ogCreateContextFromCanvas("canvas", true);
                var globe = ogCreateGlobe(context);

        ogAddImageLayer(globe, {
```

```
            url: ["http://www.openwebglobe.org/data/img"],
            layer: "World500",
            service: "i3d"
                });
        ogAddImageLayer(globe, {
            url: ["http://localhost/DataProcessing/bin/process/"],
            layer: "bugaboos",
            service: "owg"
                });
        ogAddElevationLayer(globe, {
            url: ["http://localhost/DataProcessing/bin/process/"],
            layer: "bugabooselv",
            service: "owg"
                });
        </script>
        </body>
</html>
```

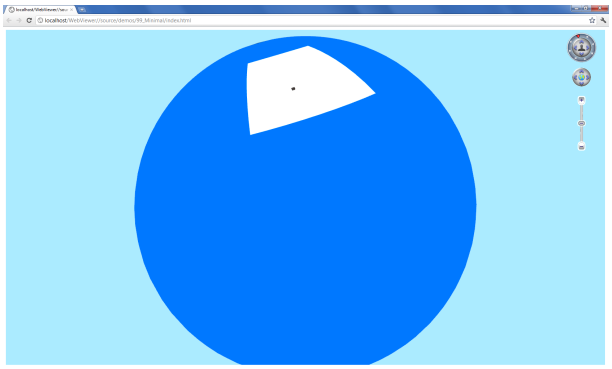

Figure 8: If you don't specify a global dataset you will see an empty world.

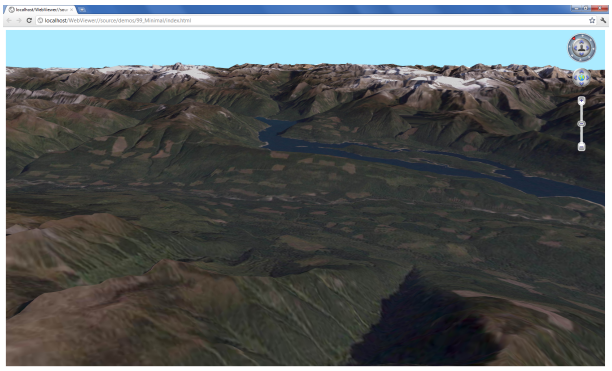

Figure 9: More detailed view of the processed data

5 Advanced Topics

5.1 Building from Source

5.1.1 Windows

Download prebuilt external dependencies from `http://www.openwebglobe.org/downloads/dataprocessing_external_win32.tar.gz`. This package is for Visual Studio 2010. We

do not support other compilers for Windows at this time.

5.1.2 Linux

Compile using the included Makefile. Currently not all tools compile under linux. We are working on this.

5.1.3 MacOS X

MacOS X is not yet supported in this release. A MacOS X build setup is planned later this year.

6 Supported Image and Elevation Formats

The OpenWebGlobe processing tools use the Geospatial Abstraction Library (GDAL) version 1.9.0, available at (www.gdal.org). It is possible to recompile GDAL to support more formats if necessary. The included version of the precompiled binaries supports the following raster formats. All data must be georeferenced to be used with OpenWebGlobe. For example if you have a .jpg image you need the corresponding world file (.jgw) or it will not be possible to add this data.

Long Format Name	Code
Arc/Info ASCII Grid	AAIGrid
ACE2	ACE2
ADRG/ARC Digitilized Raster Graphics (.gen/.thf)	ADRG
Arc/Info Binary Grid (.adf)	AIG
Magellan BLX Topo (.blx, .xlb)	BLX
Microsoft Windows Device Independent Bitmap (.bmp)	BMP
VTP Binary Terrain Format (.bt)	BT
Military Elevation Data (.dt0, .dt1, .dt2)	DTED
ECRG Table Of Contents (TOC.xml)	ECRGTOC
ESRI .hdr Labelled	EHdr
Erdas Imagine Raw	EIR
NASA ELAS	ELAS
ENVI .hdr Labelled Raster	ENVI
ERMapper (.ers)	ERS
EOSAT FAST Format	FAST
WMO GRIB1/GRIB2 (.grb)	GRIB
GRASS Rasters	GRASS
GRASS ASCII Grid	GRASSASCIIGrid
TIFF / BigTIFF / GeoTIFF (.tif)	GTiff
NOAA .gtx vertical datum shift	GTX
GXF - Grid eXchange File	GXF

HF2/HFZ heightfield raster	HF2
Erdas Imagine (.img)	HFA
Image Display and Analysis (WinDisp)	IDA
ILWIS Raster Map (.mpr,.mpl)	ILWIS
Intergraph Raster	INGR
USGS Astrogeology ISIS cube (Version 2)	ISIS2
USGS Astrogeology ISIS cube (Version 3)	ISIS3
Japanese DEM (.mem)	JDEM
JPEG JFIF (.jpg)	JPEG
KMLSUPEROVERLAY	KMLSUPEROVERLAY
NOAA Polar Orbiter Level 1b Data Set (AVHRR)	L1B
Erdas 7.x .LAN and .GIS	LAN
FARSITE v.4 LCP Format	LCP
Daylon Leveller Heightfield	Leveller
NADCON .los/.las Datum Grid Shift	LOSLAS
MBTiles	MBTiles
In Memory Raster	MEM
Vexcel MFF	MFF
Vexcel MFF2	MFF2 (HKV)
MG4 Encoded Lidar	MG4Lidar
EUMETSAT Archive native (.nat)	MSGN
NLAPS Data Format	NDF
NOAA NGS Geoid Height Grids	NGSGEOID
NITF	NITF
NTv2 Datum Grid Shift	NTv2
OZI OZF2/OZFX3	OZI
PCI Geomatics Database File	PCIDSK
PCRaster	PCRaster
NASA Planetary Data System	PDS
Swedish Grid RIK (.rik)	RIK
Raster Matrix Format (*.rsw, .mtw)	RMF
Raster Product Format/RPF (CADRG, CIB)	RPFTOC
RadarSat2 XML (product.xml)	RS2
Idrisi Raster	RST
SAGA GIS Binary format	SAGA
SAR CEOS	SAR_CEOS
ArcSDE Raster	SDE
USGS SDTS DEM (*CATD.DDF)	SDTS
SGI Image Format	SGI
Snow Data Assimilation System	SNODAS
Standard Raster Product (ASRP/USRP)	SRP
SRTM HGT Format	SRTMHGT
USGS ASCII DEM / CDED (.dem)	USGSDEM
GDAL Virtual (.vrt)	VRT

ASCII Gridded XYZ	XYZ
ZMap Plus Grid	ZMap

Table 14: Supported Raster Formats

7 Planned Features in Future Releases

Here is a list (in random order) of things that are currently in development or are planned to be implemented soon.

- Integration of Hillshading/Normalmaps (recent Masterthesis [7])
- Integration of OpenStreetMap Data processing (recent Masterthesis [7])
- Documentation & disk images for cloud integration (Amazon EC2)
- Tools for preprocessing point clouds
- Tools for preprocessing 3d objects (large scale)
- WMS support
- support different projection definitions, for example proj4 strings
- sparse quadtree support for image datasets
- Documentation: add a section about OpenLayer support
- efficient updating layers
- new tools for very large/global datasets

References

[1] Martin Christen and Stephan Nebiker. Large scale constraint delaunay triangulation for virtual globe rendering. In Thomas H. Kolbe, Gerhard König, and Claus Nagel, editors, *Advances in 3D Geo-Information Sciences*, Lecture Notes in Geoinformation and Cartography, pages 57–72. Springer Berlin Heidelberg, 2011.

[2] Martin Christen and Stephan Nebiker. Openwebglobe sdk, an open source high performance virtual globe sdk for open maps. In Manuela Schmidt and Georg Gartner, editors, *Proceedings of the 1st European State of the Map*, 2011.

[3] Google. Google maps, `http://maps.google.com`, 2012.

[4] Microsoft. Bing maps tile system, `http://msdn.microsoft.com/en-us/library/bb259689.aspx`, 2010.

[5] OpenStreetMap. Slippy map tilenames, http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames, 2012.

[6] J. P. Snyder. *Map Projections: A Working Manual*. U.S. Geological Survey Profesional Paper 1395. U.S. Geological Survey, http://pubs.er.usgs.gov/usgspubs/pp/pp1395, 1987.

[7] Robert Wüest. Paralleles pre-processing und optimiertes rendering globaler openstreetmap-daten in openwebglobe, 2012.