3dfier documentation

version 1.2.2

Last generated: October 10, 2019

Table of Contents

Getting started with 3dfier

Getting started with 3dfier

This command does not generate any model but is to verify if the program is beeing properly executed. For running 3dfier with data go to Examples (page 19)

3dfier is a command-line utility that does not have a graphical user interface (GUI). This means you need to run it from Command Prompt . You can do this by opening the Command Prompt (press windows button+R, type cmd and press enter) and dragging the 3dfier program into the Command Prompt window. If you now press enter the 3dfier program should output:

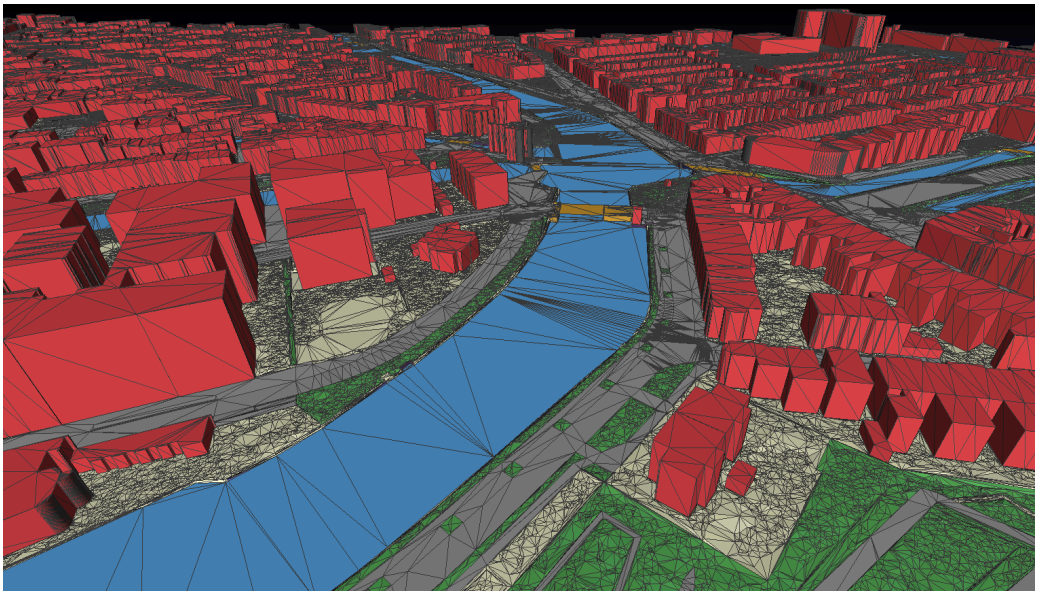
```
3dfier Copyright (C) 2015-2018  3D geoinformation research grou
p, TU Delft
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; for details run 3dfier with the '--li
cense' option.

ERROR: one YAML config file must be specified.

Allowed options:
  --help                        View all options
  --version                     View version
  --license                     View license
  --OBJ arg                     Output
  --OBJ-NoID arg                Output
  --CityGML arg                 Output
  --CityGML-Multifile arg       Output
  --CityGML-IMGeo arg           Output
  --CityGML-IMGeo-Multifile arg Output
  --CityJSON arg                Output
  --CSV-BUILDINGS arg           Output
  --CSV-BUILDINGS-MULTIPLE arg  Output
  --CSV-BUILDINGS-ALL-Z arg     Output
  --Shapefile arg               Output
  --Shapefile-Multifile arg     Output
  --PostGIS arg                 Output
  --PostGIS-PDOK arg            Output
  --PostGIS-PDOK-CityGML arg    Output
  --GDAL arg                    Output
```

Introduction

The open-source tool for creation of 3D models

Developped by the **3D Geoinformation group** at **Delft University of Technology**, 3dfier tries to fill the gap for simply creating 3D models. It takes 2D GIS datasets (e.g. topographical datasets) and "3dfies" them (as in "making them three-dimensional"). The elevation is obtained from a point cloud (we support LAS/LAZ at this moment), and the semantics of every polygon is used to perform the lifting. After lifting, elevation gaps between the polygons are removed by "stitching" together the polygons based on rules so that a watertight digital surface model (DSM) fused with 3D objects is constructed. A rule based stucture is used to extrude water as horizontal polygons, create LOD1 blocks for buildings, smooth road surfaces and construct bridges (floating 3D polygons).

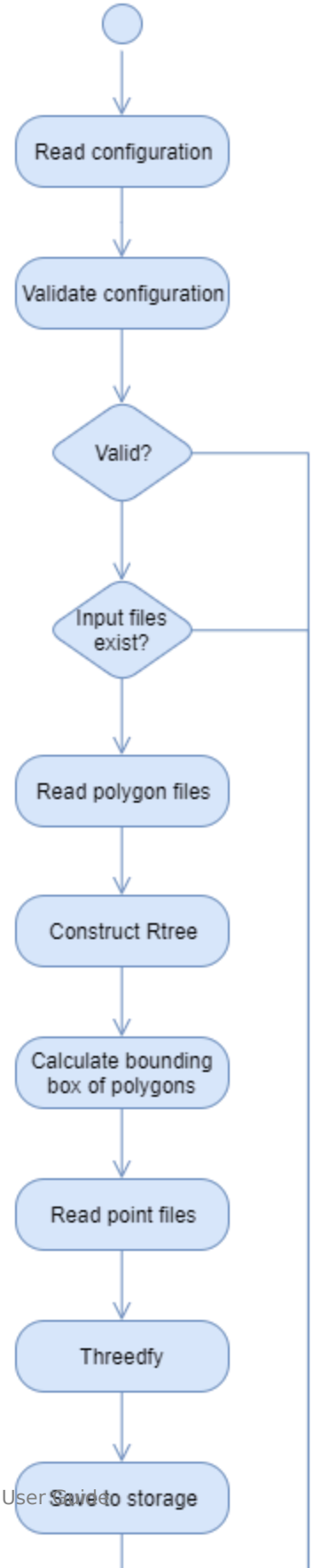
Our aim is to obtain one model that is error-free, so no intersecting objects, no holes (the surface is watertight) and buildings are integrated in the surface.



Data flow through the program

Explain different steps in the flow diagram. Show what data goes were and link to following flow diagrams.

3dfier general flow



Supported file formats

Input formats

1. Polygons

2. Point cloud

Output formats

CityJSON

CityJSON

OBJ

CityGML

CSV

Contents of a release

Release binaries content

Description of files in the zip file

Resources

Description of the contents of the resources repository

Known issues

Missing DLL’s

If you see an error message about missing MSCVRxxx.dll instead of this text you may need to do one of the following:

- MSCVR100.dll is missing: Install the Microsoft Visual C++ 2010 Redistributable
- MSCVR120.dll is missing: Install the Microsoft Visual C++ 2013 Redistributable
- MSCVR140.dll is missing: Install the Microsoft Visual C++ 2015 Redistributable

Install 3dfier on Windows

Install using binaries

There exists a ready-to-use version of 3dfier for Windows 64-bit (page 0). Download and extract the files to any given folder and follow the instructions in the Get started guide (page 0).

Note that this means that this version cannot be used when having a 32-bit (x86) Windows operating system, in that case you should build from source. Make sure to download all installers in 32-bit and set Visual Studio solution configurations to x86.

Compile from source

This guide will talk you through the compilation of 3dfier on Windows 10 64-bit using Visual Studio 2017 (steps are identical for Visual Studio 2015).

There are some steps to be taken to prepare the build environment for 3dfier. Most important is installing software to compile and downloading the libraries 3dfier is dependent of.

1. Running installers

First you will need to download and install in their default directorties:

1. Visual Studio Community 2017 . Install at least the C++ part.
2. CMake , download and install `Windows win64-x64 Installer`. Add variable to the PATH during installation.
3. Boost precompiled binaries . Pick the latest version (`boost_1_71_0-msvc-14.0-64.exe` at the time of writing). If you build on VS2015 get *mscv-14.0*, for VS2017 get *mscv-14.1*. Install boost using the installer.
4. OSGeo4W , download the 64-bit installer . From this package you will need to install at least GDAL.
5. CGAL , download `CGAL-4.12-Setup.exe` and install. Select *GMP and MPFR precompiled libs*, *Environment variables to set CGAL_DIR* and *Add CGAL/auxilary/gmp/lib to the PATH* during setup.

2. Compilation of dependencies

Next, we need to download and compile Yaml-cpp and LAStools.

Yaml-cpp

Download yaml-cpp 0.5.3 and extract to e.g. `C:\dev\yaml-cpp`. There are two options of getting the Visual Studio project files using CMake:

1. using CMake GUI (tutorial here).
2. using command line. Open a Command prompt (press windows button+R, type cmd and press enter) and navigate to the yaml-cpp directory:

```
cd C:\dev\yaml-cpp
```

Generate the Visual Studio build files with

```
mkdir vs_build
cd vs_build
cmake .. -G "Visual Studio 15 2017 Win64"
```

After generation open the Visual Studio file `YAML_CPP.sln`. Set the solution configuration to `Release` in the main toolbar. From the menu bar select Build and click `Build Solution`.

LAStools

Download LAStools and extract to e.g. `C:\dev\lastools`. There are two options of getting the Visual Studio project files:

1. Go to the folder where you extacted LAStools using file explorer (C:\dev\lastools). Enter subfolder LASlib and open `LASlib.dsp`. Now Visual Studio will automatically open and asks for a one time upgrade. Choose Yes to proceed with the upgrade. After the upgrade save and close the solution.
2. Use CMake as explained previous for Yaml-cpp (page 12) to generate the Visual Studio solution files.

After generation open the Visual Studio file `LASlib.sln`. Set the solution configuration to `Release` in the main toolbar. From the menu bar select Build and click `Build Solution`.

3. Set environment variables

Go to `Control Panel > System > Advanced system settings > Environment Variables` and add the following user variables. Note that the version numbers may be different!

- `BOOST_ROOT` = `C:\boost_1_71_0`
- `BOOST_LIBRARYDIR` = `C:\boost_1_71_0\lib64-msvc-14.0`
- `CGAL_DIR` = `C:\dev\CGAL-4.12`
- `GDAL_ROOT` = `C:\OSGeo4W64`
- `LASLIB_ROOT` = `C:\dev\lastools\LASlib`
- `LASZIP_ROOT` = `C:\dev\lastools\LASzip`
- `OSGEO4W_ROOT` = `C:\OSGeo4W64`
- `YAML-CPP_DIR` = `C:\dev\yaml-cpp`

Go to `Control Panel > System > Advanced system settings > Environment Variables` and add the following directory to Path.

- `C:\OSGeo4W64\bin`

4. Compile 3dfier

Download and extract the source code from the menu on the left or fork directly from GitHub. Browse to the vs_build folder and open the Visual Studio file `3dfier.sln`.

If in any case the Visual Studio solution is not working its possible to generate them from the CMake files directly as explained previous for Yaml-cpp (page 12).

5. Run 3dfier!

If all is good you should now be able to run 3dfier! Go to the Get started guide (page 0) and start producing models.

Help, Visual Studio complains that some file can not be found!

Check whether the directories and files specified in the environment variables are correct. Also check these places in Visual Studio:

- the include folders in `Project > Properties > C/C++ > General > Additional Include Directories`
- the library folders in `Project > Properties > Linker > General > Additional Library Directories`
- the libraries files in `Project > Properties > Linker > Input > Additional Dependencies`

Make sure each directory or file exists on your drive. For example: you may need to change a version number somewhere.

Install 3dfier on Ubuntu

1. Adding *ubuntugis* repositories

To install *GDAL* on Ubuntu 16.04 LTS it is probably the easiest to add one of the *ubuntugis* repositories, either *ubuntugis-stable* or *ubuntugis-unstable*. Both contains *GDAL* >= 2.1.

E.g. add the *ubuntugis-stable* repository by running:

```
sudo add-apt-repository ppa:ubuntugis/ppa
sudo apt-get update
```

2. Install dependencies

CGAL (`libcgal-dev`), *Boost* (`libboost-all-dev`) and *yaml-cpp* (`libyaml-cpp0.5v5`) are part of the *Ubuntu Universe* repository.

Once you have all the repos added, you can use a package manager, e.g. `apt` or *Synaptic* to install them.

E.g. using apt-get:

```
sudo apt-get install libcgal-dev libboost-all-dev libyaml-cpp
0.5v5
```

3. Run the build script

LASzip, *libLAS* and *3dfier* need to be built manually in this order, and this is what the build script can do for you. It downloads and compiles these three packages, and takes care that *libLAS* is compiled with *LASzip* support and that *3dfier* can find the executables of both. We try to take care that the download links are up to date with the upcoming releases. However, if you notice that the links are outdated, just update the version number in the script.

To download and run the build script do:

```
wget https://raw.githubusercontent.com/tudelft3d/3dfier/build_ubuntu_2/ressources/build_ubuntu1604.sh
sudo build_ubuntu1604.sh /opt
```

Where `/opt` is the directory where *3dfier* will be installed and *LASzip*, *libLAS* downloaded and compiled. *LASzip*, *libLAS* is installed into `/usr/local`, hence you'll probably need root privilage to run the script.

A note on GRASS GIS and libLAS\ If you already have GRASS installed from the *ubuntugis-(un)stable* PPA, you probably also have libLAS (`liblas-c3, liblas3`) installed, as GRASS depends on them. However, this *libLAS* install is without *LASzip* support as *LASzip* is not part of the *ubuntugis* PPA. Therefore, you will need to remove the GRASS and *libLAS* libraries first, then compile *libLAS* with *LASzip* support (with this script). Then you can install GRASS again from the *ubuntugis* PPA, it will be compiled with *libLAS* that now supports `.laz`.

Install 3dfier using a Docker image

The Dockerfile for building 3dfier (`Dockerfile_builder`) contains the complete instruction for setting up a build environment. If you don't want to install the required dependencies on your local machine, you can use this docker image to build and run 3dfier. If you want to build 3dfier locally, then please look into the Dockerfile for the libraries that you'll need.

Using Docker to compile and run 3dfier

The builder image (`Dockerfile_builder`) does not contain the source code of 3dfier, neither the executable. We use this image for compiling 3dfier from different branches during development and testing. Nevertheless, it is suitable for 3dfying any data set.

As an example, the complete process of building the docker image, compiling 3dfier and running looks like the following.

1. Build the docker image

```
$ docker build -t 3dfier:builder -f Dockerfile_builder .
```

2. Prepare a compiler script for 3dfier

The contents of `build_3dfier.sh`

```
#! /bin/sh

rm -rf /opt/3dfier_src/build; \
cd /opt/3dfier_src && mkdir build && cd ./build; \
cmake \
-DLIBLASC_LIBRARY=/opt/libLAS-1.8.1/build/lib/liblas_c.so \
-DLIBLAS_INCLUDE_DIR=/opt/libLAS-1.8.1/build/include \
-DLIBLAS_LIBRARY=/opt/libLAS-1.8.1/build/lib/liblas.so \
-DLASZIP_INCLUDE_DIR=/opt/laszip-src-2.2.0/build/include \
-DLASZIP_LIBRARY=/opt/laszip-src-2.2.0/build/lib/liblaszip.so \
-DCGAL_DIR=/opt/cgal-releases-CGAL-4.12/build \
-DCMAKE_BUILD_TYPE=Release ../3dfier \
; \
make
```

3. Compile 3dfier in a container

If we set up an out of source build system in a root directory `3dfier_src` with the following sturcture,

```
/3dfier_src
├── 3dfier
├── build
```

then we execute the commands from the root dir. The `3dfier_src` dir is *bind-mount*-ed to the container where 3dfier is complied by running `build_3dfier.sh`. The script will place the executable into the `build` directory in `3dfier_src`.

```
$ docker run --rm -it -v "$(pwd)":/opt/3dfier_src 3dfier:builde
r /opt/3dfier_src/build_3dfier.sh
```

4. Running 3dfier in a container

Then the previously built executable is mounted to the container and executed there. Note that the data and config files are also mounted in a similar fashion.

```
$ docker run --rm -it -v "$(pwd)":/opt/3dfier_src 3dfier:builde
r /opt/3dfier_src/build/3dfier /opt/3dfier_src/3dfier/example_d
ata/testarea_config_docker.yml --CSV-BUILDINGS /opt/3dfier_src/
test.csv
```

Example data

Prepare example data

For this example we use BGT_Delft_Example.zip (page 0) from the GitHub repository located in `3dfier/resources/Example_data/`. Create a folder with 3dfier and the depencency dll's and add the `example_data folder`.

Folder layout

Running with the example data

Opening the Command Prompt (press windows button+R, type cmd and press enter) and navigate to the folder where 3dfier is located.

Now go into the example data folder using `cd example_data` and run `..\3dfier.exe testarea_config.yml --OBJ output\myfirstmodel.obj`. Now 3dfier will start processing and when finished it produced its first 3D model!

The output file can be found here `example_data\output\testarea.obj` and console output should be as follows.

```
3dfier Copyright (C) 2015-2018  3D geoinformation research grou
p, TU Delft
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; for details run 3dfier with the '--li
cense' option.

Config file is valid.
Reading input dataset: bgt\bgt_waterdeel.sqlite
        Layer: waterdeel
        (3 features --> Water)
Reading input dataset: bgt\bgt_ondersteunendwaterdeel.sqlite
        Layer: ondersteunendwaterdeel
        (0 features --> Water)
Reading input dataset: bgt\bgt_onbegroeidterreindeel.sqlite
        Layer: onbegroeidterreindeel
        (81 features --> Terrain)
Reading input dataset: bgt\bgt_wegdeel.sqlite
        Layer: trafficarea
        (151 features --> Road)
Reading input dataset: bgt\bgt_ondersteunendwegdeel.sqlite
        Layer: auxiliarytrafficarea
        (0 features --> Road)
Reading input dataset: bgt\bgt_pand.sqlite
        Layer: buildingpart
        (160 features --> Building)
Reading input dataset: bgt\bgt_begroeidterreindeel.sqlite
        Layer: plantcover
        (126 features --> Forest)
Reading input dataset: bgt\bgt_scheiding.sqlite
        Layer: scheiding
        (57 features --> Separation)
Reading input dataset: bgt\bgt_kunstwerkdeel.sqlite
        Layer: kunstwerkdeel
        (1 features --> Separation)
Reading input dataset: bgt\bgt_overigbouwwerk.sqlite
        Layer: overigbouwwerk
        (0 features --> Separation)
Reading input dataset: bgt\bgt_overbruggingsdeel.sqlite
        Layer: bridgeconstructionelement
        (3 features --> Bridge/Overpass)

Total # of polygons: 570
Constructing the R-tree... done.
Spatial extent: (84,616.468, 447,422.999) (85,140.839, 447,75
```

```
0.636)
Reading LAS/LAZ file: ahn3\ahn3_cropped_1.laz
        (640,510 points in the file)
        (all points used, no skipping)
        (omitting LAS classes: 0 1 )
[==================================================] 100%
Reading LAS/LAZ file: ahn3\ahn3_cropped_2.laz
        (208,432 points in the file)
        (all points used, no skipping)
        (omitting LAS classes: 0 1 )
[==================================================] 100%
All points read in 1 seconds || 00:00:01
3dfying all input polygons...
===== /LIFTING =====
===== LIFTING/ =====
=====  /ADJACENT FEATURES =====
=====  ADJACENT FEATURES/ =====
=====  /STITCHING =====
=====  STITCHING/ =====
=====  /BOWTIES =====
=====  BOWTIES/ =====
=====  /VERTICAL WALLS =====
=====  VERTICAL WALLS/ =====
Lifting, stitching and vertical walls done in 0 seconds || 00:0
0:00
=====  /CDT =====
=====  CDT/ =====
CDT created in 0 seconds || 00:00:00
...3dfying done.
OBJ output: output\myfirstmodel.obj
Features written in 0 seconds || 00:00:00
Successfully terminated in 2 seconds || 00:00:02
```

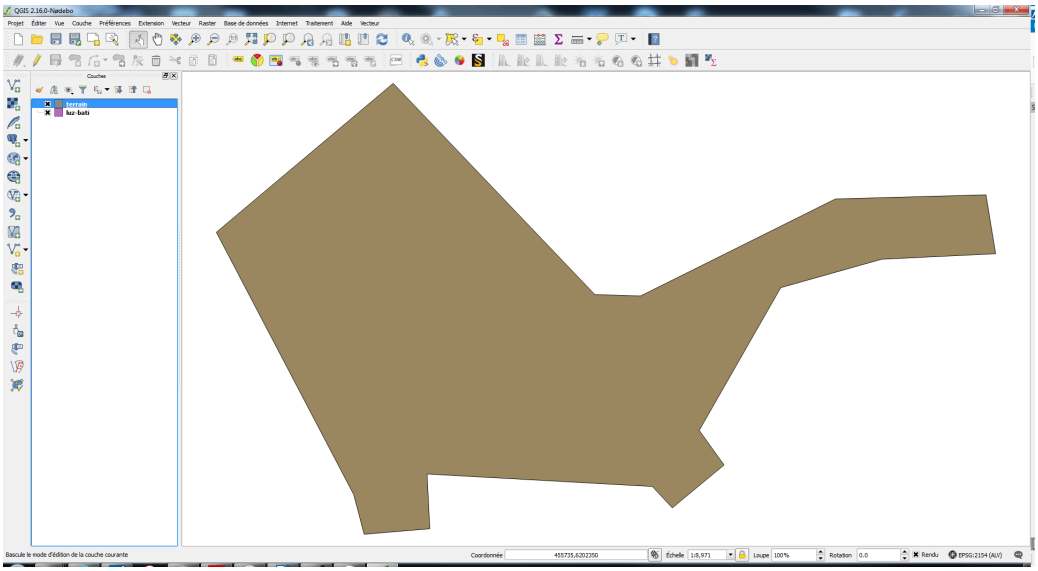
Data preperation overview

Generate terrain polygon

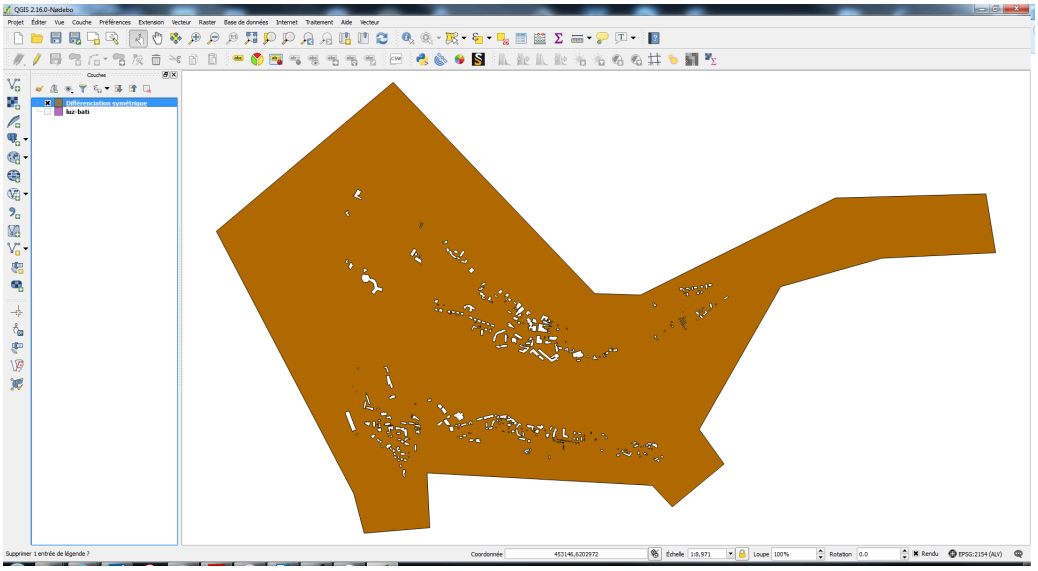
Without a 2D polygon describing terrain extent, 3dfier is not able to create a terrain from a point cloud without. The solution is to create a polygon of the area of interest save this as a shapefile.

We make sure the terrain and buildings are stitched nicely and the buildings do not intersect with the terrain. We do this by using symetrical difference between the terrain polygon and the buildings. For nice tutorial about this see section *D. Symmetrical Difference* at GrindGIS .

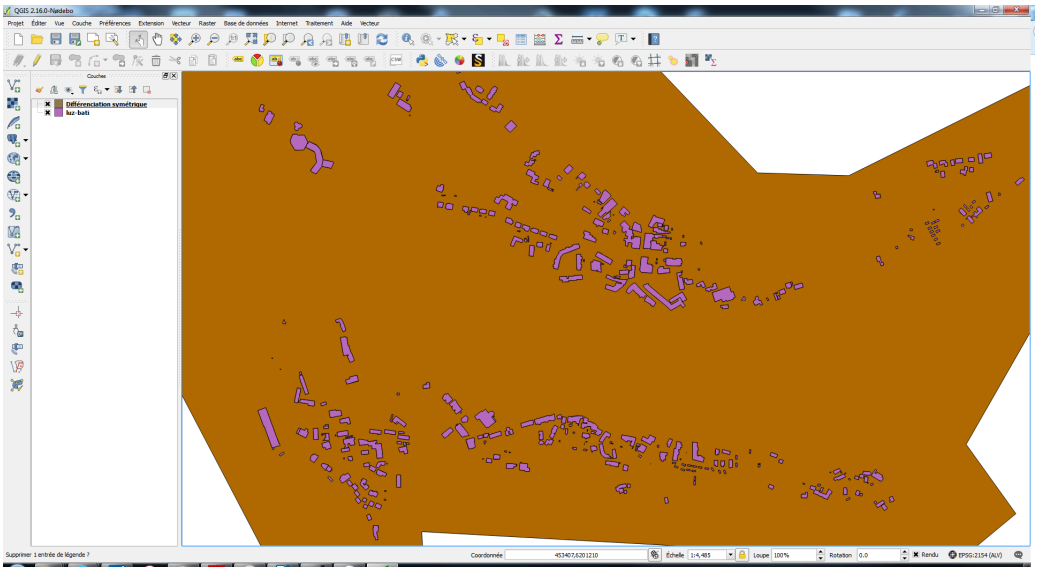
Terrain



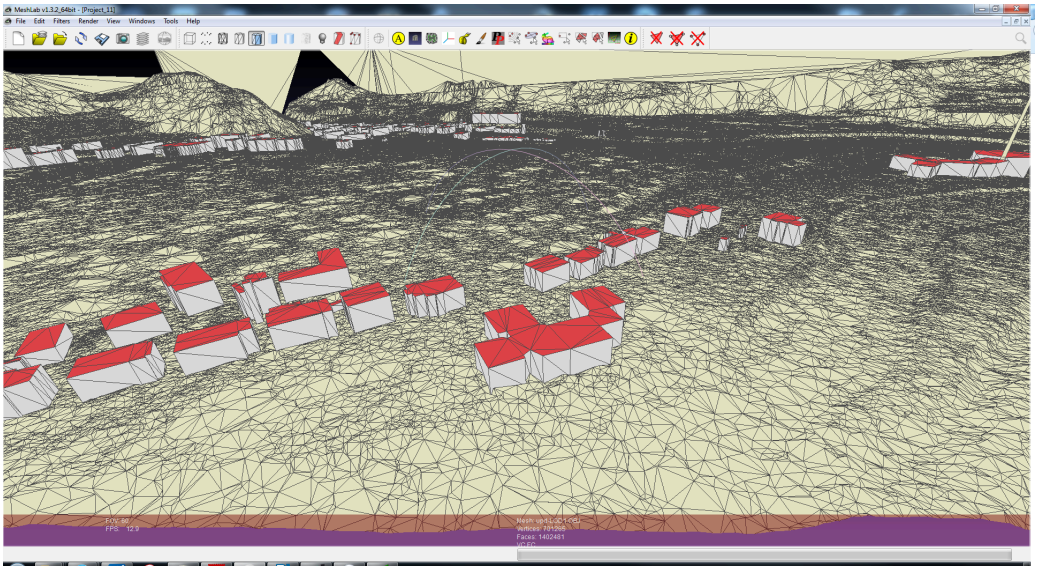
Symmetrical Difference operation result



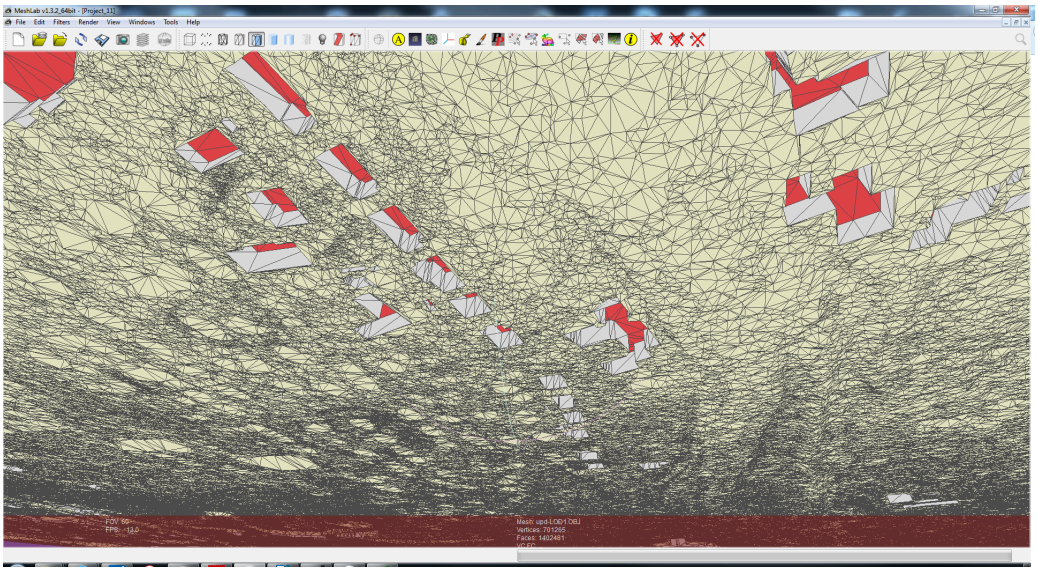
2D terrain + buildings



3D model as OBJ in Meshlab



Viewpoint below the 3D model



Thanks to @antoinebio for the images supplied in issue #48 .

Building footprints from OpenStreetMap

This guide is about extracting building footprints as a Shapefile of polygons from the OpenStreetMap dataset, in order to use them as input for 3dfier.

Download OSM data

First, you need to download the OSM data for your area of interest:

1. Through your browser, visit the OpenStreetMap website
2. Zoom at the area you want to work on
3. Press the `Export` button on the top
4. From the left panel, you may select to further specify the area you want to download (via the `Manually select a different area` option)
5. When finished, press the `Export` button from the left panel.

You should be asked to download an `.osm` file. Just store it somewhere on your computer.

Extract buildings through QGIS

Originally, the data from OpenStreetMap are just geometries with key-value pairs assigned to them. You can easily filter the buildings from all geometries inside an `.osm` file.

1. Open QGIS.
2. From the menu, select `Layers` -> `Add Layer` -> `Add Vector layer...`.
3. Select the `.osm` file with the area downloaded.
4. When prompted about the layer you want to add, you can only select the `multipolygons` layer (it's OK if you add all of them, but buildings are on this layer).
5. From the `Layers Panel`, right-click on the multipolygons layer you've just added and select `Filter...`.

6. On the dialog, provide the following expression: `"building" is not null` (essentially, what that means is that all polygons without a *building* key, will be filtered out). You should now see only the buildings on the map.
7. From the `Layers Panel`, right-click on the multipolygons layer, again, and now select `Save as...`.
8. On the dialog, select the `ESRI Shapefile` format and provide the output file. You may also want to reproject the geometries on another CRS, now, if the elevation data you are going to use on 3dfier are not on WGS 84 (EPSG:4326).

When you save, you should have a shapefile with footprint of the buildings for this area, at the CRS you specified. You can use this, now, as an input for 3dfier.