Maps Re-imagined: Digital, Informational, and Perceptional Experimentations in Progress

Contents

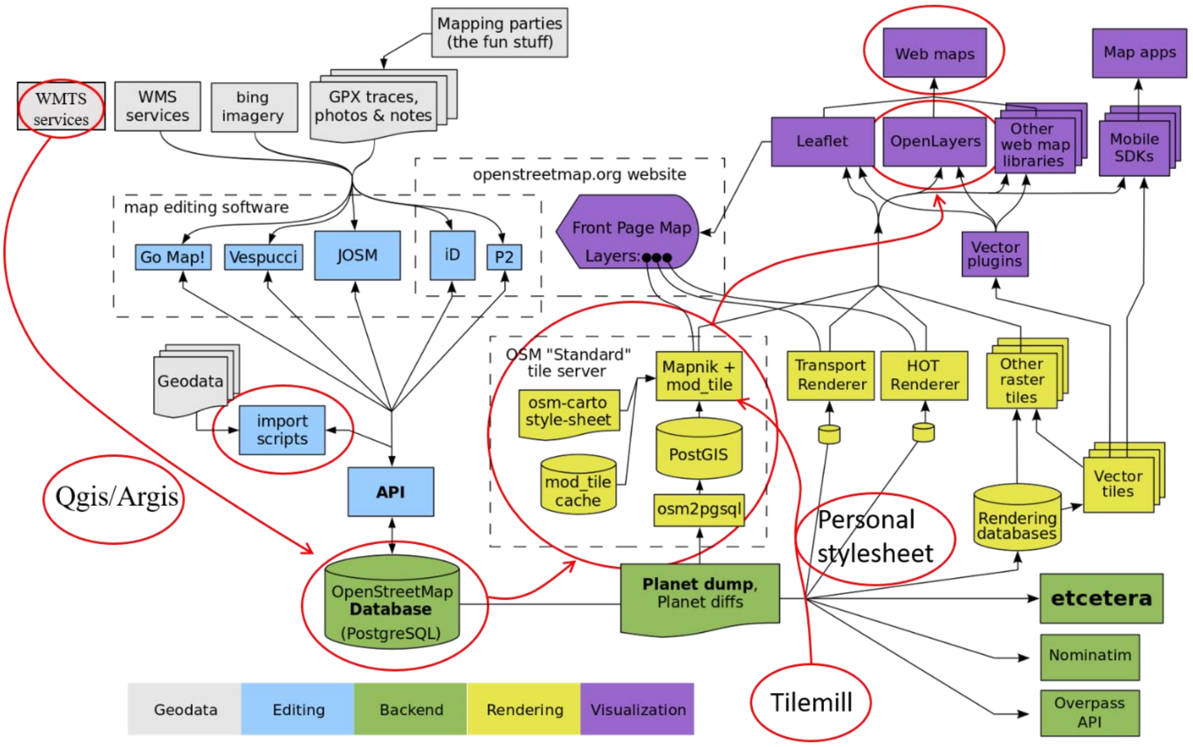
Objectives

In this tutorial, you will first learn thecurrent state of the ancient map, as well as the protection dilemma faced, and then explain how to convert the ancient map file into a vector file through QGIS and execute its key values according to the format. OpenStreetMap. Convert and write via Ogr2osm (https://github.com/pnorman/ogr2osm) written in Python and convert the shapefile to the .osm file format.

The next step is to show you how to write a graphical style sheet, write the .osm stylesheet through the open source TileMill program, and output it as a Mapnik XML file format.

Finally, learn how to set up OSM's tileserver on Ubuntu Linux, how to upload self-contained .osm and style sheets, and how to render and then apply them to Openlayers.

The following figure is the process diagram of this technical document.



Software Requirements

This section mainly introduces the features of the software used in the remap process, and provides individual installation methods, but does not explain the installation process and application instructions in detail.

QGIS
QGIS is a free and open-source cross-platform desktop geographic information system (GIS) application that supports viewing, editing, and analysis of geospatial data, QGIS functions allow users to analyze and edit spatial information, and QGIS supports both raster and vector layers; vector data is stored as either point, line, or polygon features. And you can install it by searching for "QGIS" or via the link below. https://www.qgis.org/en/site/
However, you can also use the Arcgis produced by ESRI to digitize the ancient maps.

Ogr2osm
Ogr2osm is a Python script which converts any OGR supported vector format into .osm format, with an emphasis on converting polygon boundaries into relations.

Unlike most (if not all) other scripts, such as shp2osm and polyshp2osm, it can save a processing step by converting from any coordinate system supported by PROJ.4 to the system used by OpenStreetMap, (EPSG:4326).

The current version is maintained by Paul Norman, released under a MIT license, and You can find more detail in this website. http://www.github.com/pnorman/ogr2osm.

TileMill
TileMill is an open source map design studio, developed by a community of volunteer open source contributors. Original development on the project was led by Development Seed and Mapbox as part of the 2010 Knight News Challenge.
This project will use the tilemill to design the stylesheets for the digital maps.
And you can learn more about the details of the work through the links below.
https://tilemill-project.github.io/tilemill/docs/crashcourse/introduction/

Ubuntu
Ubuntu is a free and open-source Linux distribution based on Debian.Ubuntu is officially released in three editions: Desktop,Server, and Core (for IoT devices and robots). Ubuntu is a popular operating system for cloud computing, with support for

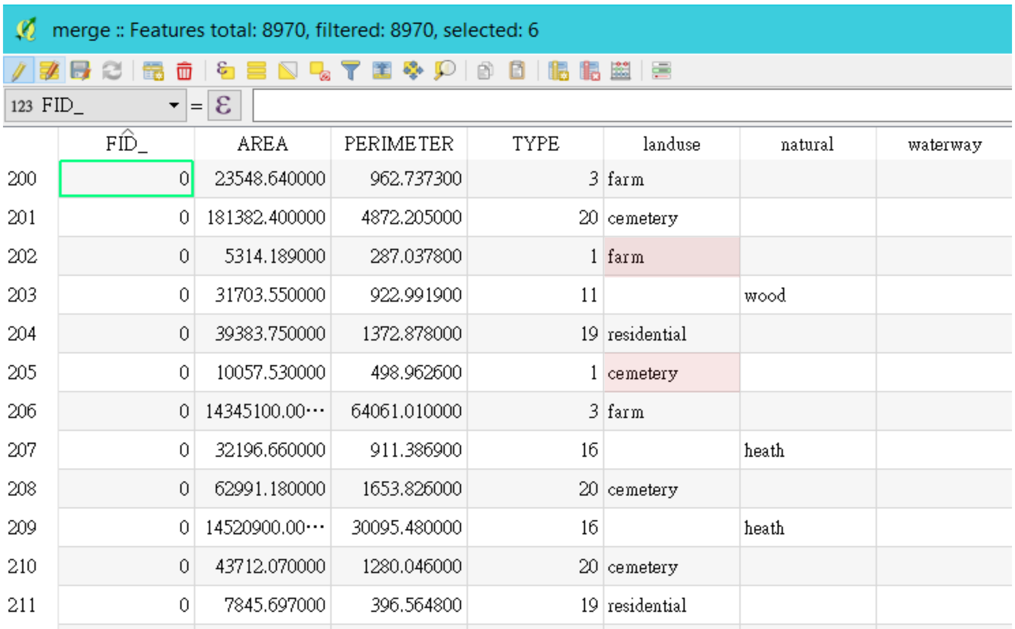
OpenStack. In this project, we only need to install the Ubuntu (16.04.2 LTS) version, and you can install it through the following link.
https://www.ubuntu.com/download/alternative-downloads

OpenLayers
OpenLayers is an open-source (provided under the 2-clause BSD License) JavaScript library for displaying map data in web browsers as slippy maps. It provides an API for building rich web-based geographic applications similar to Google Maps and Bing Maps. OpenLayers supports GeoRSS, KML (Keyhole Markup Language), Geography Markup Language (GML), GeoJSON and map data from any source using OGC-standards as Web Map Service (WMS) or Web Feature Service (WFS).

Remap shapefiles in format of OpenStreetMap

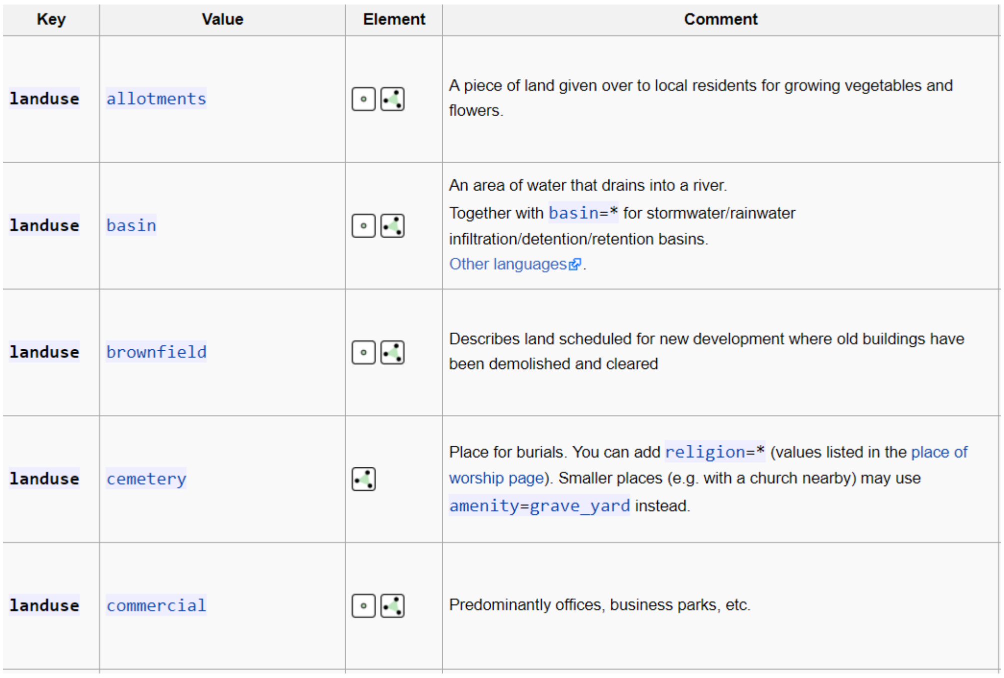
In this step, you need to import the ancient map files or import WMTS maps, and edit the features and the attribute table through QGIS. If you are a beginner in GIS, you can follow the documentation on the website to find out the detailed workflow.
https://docs.qgis.org/3.4/en/docs/user_manual/

Most features can be described using only a small number of tags.These tags need to be recorded in the Key-Value of the attribute table and follow the naming rules of OpenStreetMap. You can learn more about the rules for key-value writing through the links below. https://wiki.openstreetmap.org/wiki/Map_Features
The following picture illustrates the process of writing and the k-value classification of OpenStreetMap



merge :: Features total: 8970, filtered: 8970, selected: 6

	FID_	AREA	PERIMETER	TYPE	landuse	natural	waterway
200	0	23548.640000	962.737300	3	farm		
201	0	181382.400000	4872.205000	20	cemetery		
202	0	5314.189000	287.037800	1	farm		
203	0	31703.550000	922.991900	11		wood	
204	0	39383.750000	1372.878000	19	residential		
205	0	10057.530000	498.962600	1	cemetery		
206	0	14345100.00…	64061.010000	3	farm		
207	0	32196.660000	911.386900	16		heath	
208	0	62991.180000	1653.826000	20	cemetery		
209	0	14520900.00…	30095.480000	16		heath	
210	0	43712.070000	1280.046000	20	cemetery		
211	0	7845.697000	396.564800	19	residential		



Key	Value	Element	Comment
landuse	allotments	⊡ ⊞	A piece of land given over to local residents for growing vegetables and flowers.
landuse	basin	⊡ ⊞	An area of water that drains into a river. Together with basin=* for stormwater/rainwater infiltration/detention/retention basins. Other languages.
landuse	brownfield	⊡ ⊞	Describes land scheduled for new development where old buildings have been demolished and cleared
landuse	cemetery	⊞	Place for burials. You can add religion=* (values listed in the place of worship page). Smaller places (e.g. with a church nearby) may use amenity=grave_yard instead.
landuse	commercial	⊡ ⊞	Predominantly offices, business parks, etc.

Using Python to convert shapefiles to an osm document

When we complete the ancient map digitization and give the key values specified by OpenStreetMap. Thwn, We convert the shapefiles through Ogr2osm to convert it into the .osm format.

Ogr2osm requires gdal with python bindings. Depending on the file formats you want to read you may have to compile it yourself but there should be no issues with shapefiles.

On Ubuntu you can run the following command to get the software you need.

```
sudo apt-get install -y python-gdal python-lxml
```

Then, to install ogr2osm and download the default translations the following command can be used.

```
git clone --recursive https://github.com/pnorman/ogr2osm
```

To update

```
cd ogr2osm
git pull
git submodule update
```

You can find more usage details of ogr2osm.py in this website. http://www.github.com/pnorman/ogr2osm.

And you can quickly convert .shp to .osm file format by executing the following command.

```
pthon ogr2osm.py your shapefile path -o your output path
```

The following codes shows the example of the .osm file.

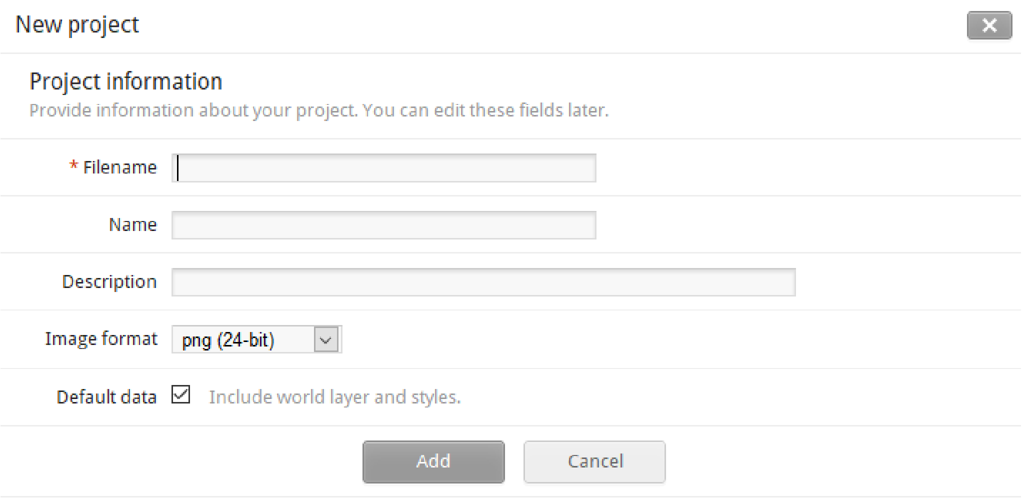
```
<?xml version="1.0"?>
<osm version="0.6" upload="false" generator="uvmogr2osm">
<node id="1826" visible="true" version="1" changeset="1" timestamp="2017-08-30T11:05:08Z" user="HSU" uid="70362" lat="23.258194741796974" lon="120.21515318097568" >
   <tag k="landuse" v="cemetery"/>
</node>

<way id="66" visible="true" version="1" changeset="66" timestamp="2017-08-30T11:05:08Z" user="HSU" uid="70362">
   <nd ref="10341" />
   <nd ref="10342" />
   <nd ref="10343" />
   <tag k="highway" v="path"/>
</way>
<relation id="-86310" visible="true">
   <member ref="-86311" role="outer" type="way" />
   <member ref="-86317" role="inner" type="way" />
   <member ref="-86318" role="inner" type="way" />
   <member ref="-86319" role="inner" type="way" />
   <member ref="-86320" role="inner" type="way" />
   <member ref="-86322" role="inner" type="way" />
   <member ref="-86324" role="inner" type="way" />
   <member ref="-86325" role="inner" type="way" />
   <tag k="type" v="multipolygon" />
   <tag k="waterway" v="river" />
   <tag k="landuse" v="" />
   <tag k="natural" v="water" />
</relation>
```

Installing TileMill& Design the Maps Stylesheet

After completing the .osm format conversion, you will then design the stylesheet for the map. In this section, you will use TileMill to design and preview and modify the style during the design process.

In our project, we are designing a stylesheet by installing the Windows version of Tilemill.And you can also follow the installation process of other platforms through the links below.
https://tilemill-project.github.io/tilemill/docs/install/
Once you have completed the installation and initial setup of the Tilemill, you can start designing the style sheet.
Next, we will explain the basic process of its design, and briefly introduce the commonly used functions and their syntax usage.

In the first step, you must create a new project to manage different stylesheets.

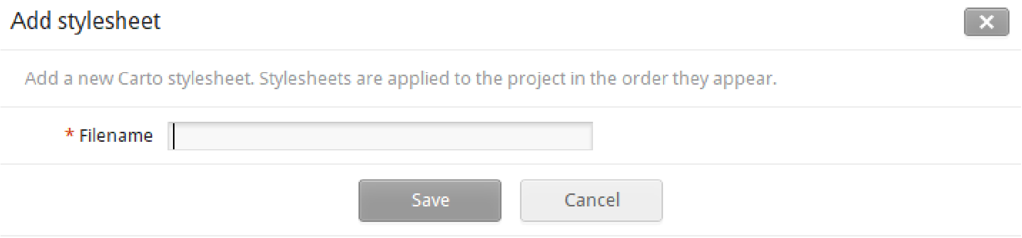


New project

Project information
Provide information about your project. You can edit these fields later.

* Filename	
Name	
Description	
Image format	png (24-bit)
Default data	☑ Include world layer and styles.

And it can be roughly divided into four layers: "background", "point", "line", "polygon".Among them, "point", "line", "polygon" can correspond to the three attributes of the description object in OpenStreetMap.
Therefore you can click "Add stylesheet" to add different style layers for point, line and polygon layers.

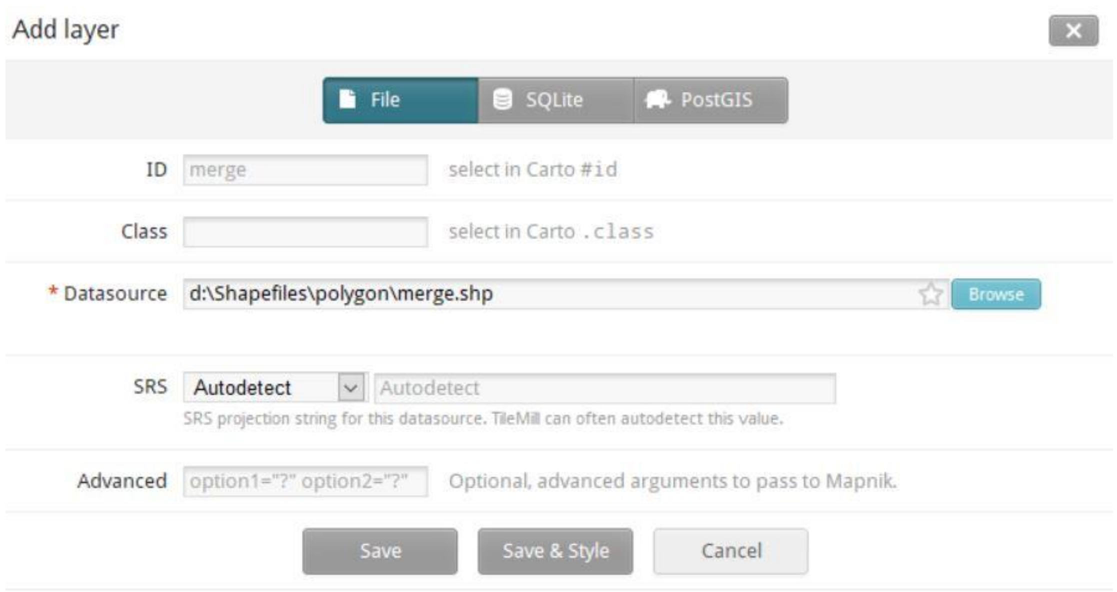


Add stylesheet

Add a new Carto stylesheet. Stylesheets are applied to the project in the order they appear.

* Filename

Then you can import the vector layers through the "Add Layers" at the bottom left to preview the stylesheet.



ID: merge — select in Carto #id

Class: — select in Carto .class

* Datasource: d:\Shapefiles\polygon\merge.shp

SRS: Autodetect — Autodetect
SRS projection string for this datasource. TileMill can often autodetect this value.

Advanced: option1="?" option2="?" — Optional, advanced arguments to pass to Mapnik.

Next we will explain the details of the stylesheet syntax.
In the "point" section.The pattern size can be adjusted for different zoom levels, and the pattern can be set by "marker-file: url ('your path of the marker's picture') ".
In addition, "marker-allow-overlap: true " is to explain whether to allow layer coverage

```
#temple {
    marker-width:60;
    marker-file: url('your path of the marker’s picture');
    marker-opacity:0.8;
    marker-allow-overlap:true:
    [zoom = 14]{
        Marker-width:30;
    }
    [zoom = 15]{
        Marker-width:60;
    }
}
```

In the "line" section, it is mainly divided into "case" to adjust the line thickness and color, "fill" to adjust the fill level of the line, and "dash" is the adjustment of the dotted line.

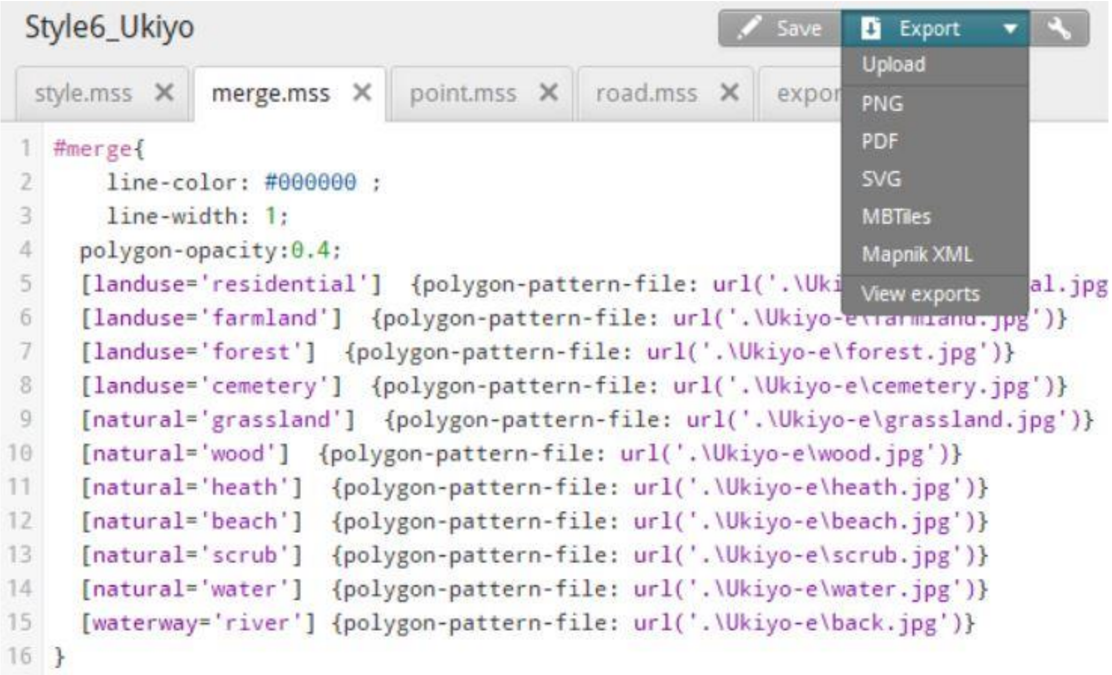
```
#primaryway {
    [zoom < 15]{
    ::case {
        line-width: 2;
        line-color:#000000;
    }
    ::fill {
        line-width: 1;
        line-color:#fff;
    }
    }
    [zoom >= 15]{
    ::case {
        line-width: 4;
        line-color:#000000;
    }
    ::fill {
        line-width: 2;
        line-color:#fff;
    }
    }
}
```

In the "polygon" section, the polygon edges are adjusted through "line-color" and "line-width", "polygon-opacity" adjusts the transparency, and "polygon-pattern-file" selects the polygon style.In addition, you can fill in simple colors with "polygon-fill"

```
    #merge{
            line-color: #000000  ;
            line-width: 1;
        polygon-opacity:0.4;
        [landuse='residential'] {polygon-pattern-file: url(' your path of the polygon’s
picture ')}
        [landuse='farmland'] {polygon-fill: #fff}

}
```

You can design your own unique stylesheet according to the details described in the official documentation.
https://tilemill-project.github.io/tilemill/docs/crashcourse/styling/

After the customization of the stylesheet is completed, the output can be converted into a Mapnik XML file via built-in conversion.



```
#merge{
    line-color: #000000 ;
    line-width: 1;
  polygon-opacity:0.4;
  [landuse='residential']  {polygon-pattern-file: url('.\Uki              al.jpg
  [landuse='farmland']  {polygon-pattern-file: url('.\Ukiyo-e\farmland.jpg')}
  [landuse='forest']  {polygon-pattern-file: url('.\Ukiyo-e\forest.jpg')}
  [landuse='cemetery']  {polygon-pattern-file: url('.\Ukiyo-e\cemetery.jpg')}
  [natural='grassland']  {polygon-pattern-file: url('.\Ukiyo-e\grassland.jpg')}
  [natural='wood']  {polygon-pattern-file: url('.\Ukiyo-e\wood.jpg')}
  [natural='heath']  {polygon-pattern-file: url('.\Ukiyo-e\heath.jpg')}
  [natural='beach']  {polygon-pattern-file: url('.\Ukiyo-e\beach.jpg')}
  [natural='scrub']  {polygon-pattern-file: url('.\Ukiyo-e\scrub.jpg')}
  [natural='water']  {polygon-pattern-file: url('.\Ukiyo-e\water.jpg')}
  [waterway='river'] {polygon-pattern-file: url('.\Ukiyo-e\back.jpg')}
}
```

And then you can upload it to the Osm tileserver that will be set up next to provide a new style of application.

Building a tile server (16.04.2 LTS)

In this part, We will describe how to install, setup and configure all the necessary software to operate your own tile server. The step-by-step instructions are written for Ubuntu Linux 16.04.2 LTS (Xenial Xerus).

You need to set up the tileserver according to the process described on this website: https://switch2osm.org/manually-building-a-tile-server-16-04-2-lts/

In the process of installation, you only need to pay attention to the following steps, then you can update the osm files and personal stylesheets completed in the previous work to your tileserver.

1. Loading data

In this part, you need to replace the data path introduced in the documentation, and replace it with the osm file you have completed.

```
  osm2pgsql -d gis --create --slim   -G --hstore --tag-transform-script
~/src/openstreetmap-carto/openstreetmap-carto.lua -C 2500 --number-processes 1 -S
~/src/openstreetmap-carto/openstreetmap-carto.style ~your osm file path
```

2. Setting up your webserver

In this section, you must change the rendering settings via the command

```
  sudo nano /usr/local/etc/renderd.conf
```

When you enter the "renderd.conf" interface, in addition to following the steps in the documentation, you can add your new stylesheet with the following code:.

```
[Your stylesheet name]
URI=/which url do you want/
TILEDIR=/var/lib/mod_tile
XML=your path of the stylesheet .xml file export by Tilemill
HOST=localhost
TILESIZE=256
MAXZOOM=20

[default](this is the default stylesheet)
URI=/default/
TILEDIR=/var/lib/mod_tile
XML=/home/bcfuture/src/openstreetmap-carto/mapnik.xml
HOST=localhost
TILESIZE=256
MAXZOOM=20
```

In addition, you can also use "renderd.conf" to control the version and url of your stylesheet.

After completing the above work steps, you can enjoy the results of different stylesheets of your own digital maps.

Connecting to Openlayers

And if you want to use osm tileserver further, such as providing different styles of map services. You can use OpenLayers to complete the content and functions of the subsequent platform service.

And this is the map service that the author has initially completed and set up. The website is as follows:
https://bcfuture.github.io/tileserver/Switch.html

Also, you can find more example and tutorials in this website:
https://openlayers.org/en/latest/examples/