

Creating a Web Map with Texture and Scale-dependent Symbolization using TileMill & Mapbox

What you need

- ✓ TileMill v 0.10 and above
- ✓ Internet access
- ✓ MapBox account (explained below)
- ✓ Data package for this tutorial
- ✓ Any text editor
- ✓ Photoshop or GIMP (or any other equivalent that let you create texture and export them as .png)

Tutorial overview

This tutorial will show you how to prepare a web map with a natural look, using CartoCSS and TileMill and how to embed it in a simple web page. Part A will demonstrate how to use TileMill to create beautiful map with texture and visual effect. Then, part B will show how to export your project into MBtiles that can be uploaded to MapBox. And finally, part C will show you how to use your tiles and to serve them on your webpage using the Mapbox-Leaflet library.

MapBox Account

A MapBox account allows you to serve tiles through their web service, but it also offers customizable map to use in combination with your own data. The free option allows you to have 3000 request per months to your maps.

- 1) Go to www.mapbox.com
- 2) Click on **Sign Up** and follow the instructions
- 3) Keep your Username and password at hand for the part B of this tutorial.

(A) Creating the map

- 1) If not already done, download TileMill at <https://www.mapbox.com/tilemill/> and open the program.
- 2) In the starting window, click on **+ New Project** (see Figure 1).

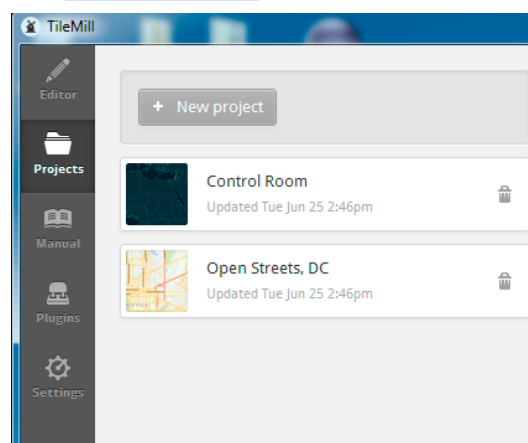


Figure 1: Create a new project

- 3) Fill out the parameters and click **Add** (see Figure 2).

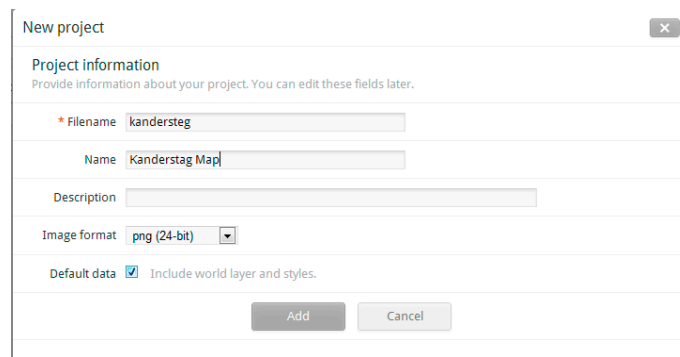


Figure 2: Parameters

4) Now you are in the editor section and TileMill loaded a layer with the countries for you and symbolized it. You can see the code in the right panel. The language used to define the style is called CartoCSS. If you know CSS, you can notice similarities. In the bottom-left corner you have four icons that let you access the layers of your map (layer stack icon), the references for CartoCSS (the bracket icon), the available fonts and the template options (see Figure 3).

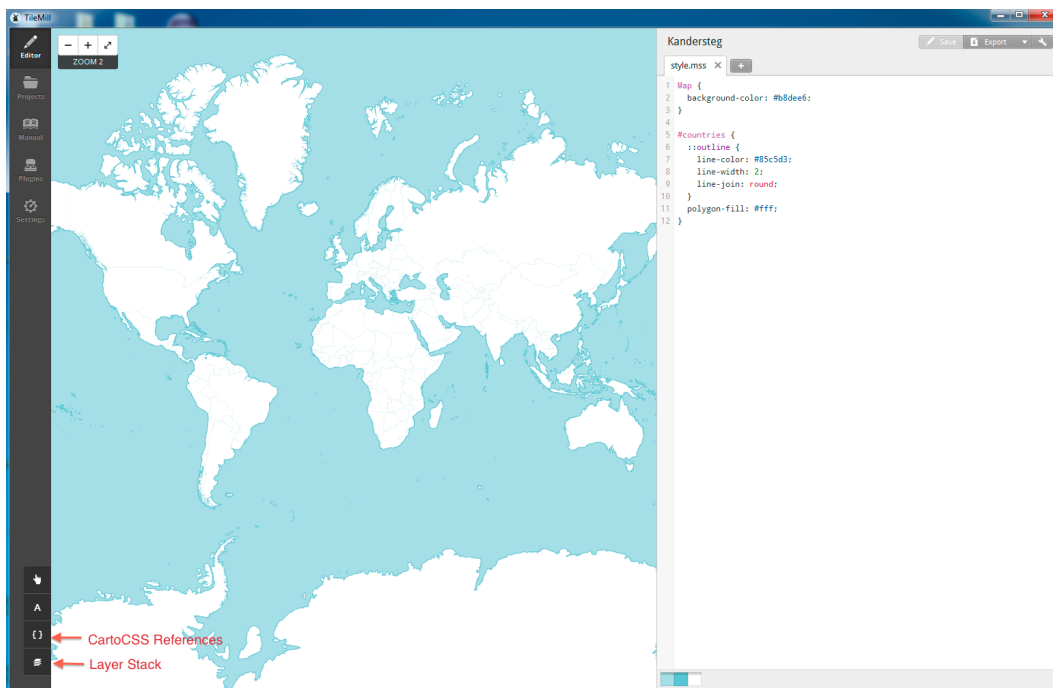


Figure 3: TileMill Overview

5) Click on the icon representing the layer stack, click **+ Add Layer** (see Figure 4).

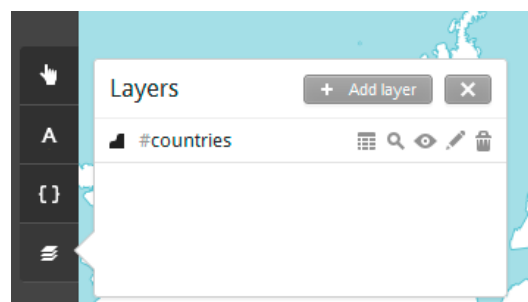


Figure 4: Add layer

6) Browse to load the *kandersteg_landuse.shp* layer (see Figure 5)

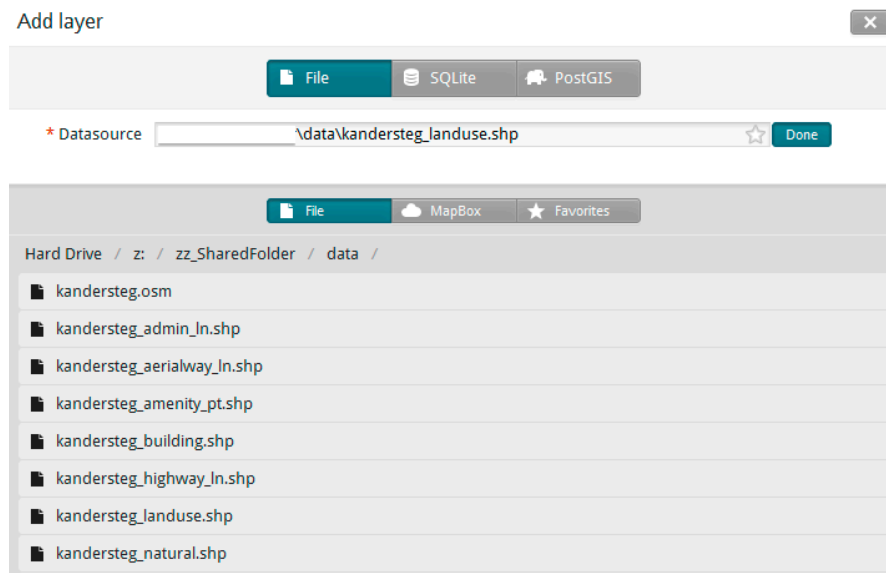


Figure 5: Load a layer

7) Give a name (ID) to the layer. Don't forget to click **Save and Style** when ready to load the shapefile (see Figure 6). You might need to zoom in to see the features.

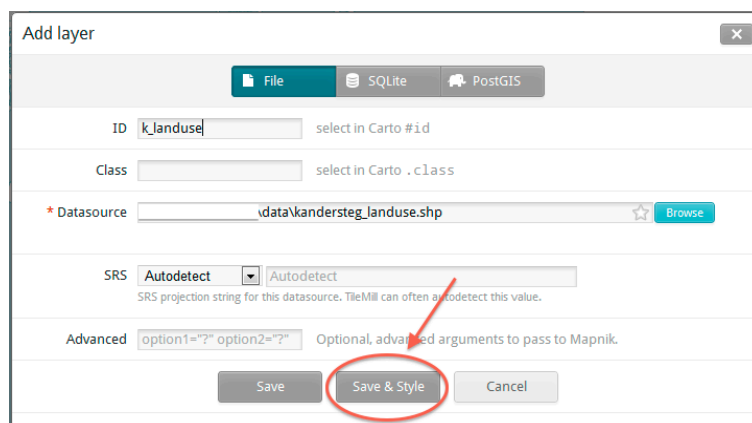


Figure 6: Name your layer, then Save and Style it

8) At that point, you can erase the layer #countries in the left panel with the layer by clicking on the garbage bin icon. You can erase the corresponding style in the right panel as well.

9) Now, in the top right corner, click on the icon for the map settings. Zoom to zoom level 13 to see your data and, enter the settings as shown below. It defines the zoom level to which the map will be available (13 to 16), as well as its extent (7.6262,46.4406,7.8181,46.5422) and its center (7.6724, 46.4851,13) (see Figure 7). And click **Save**.

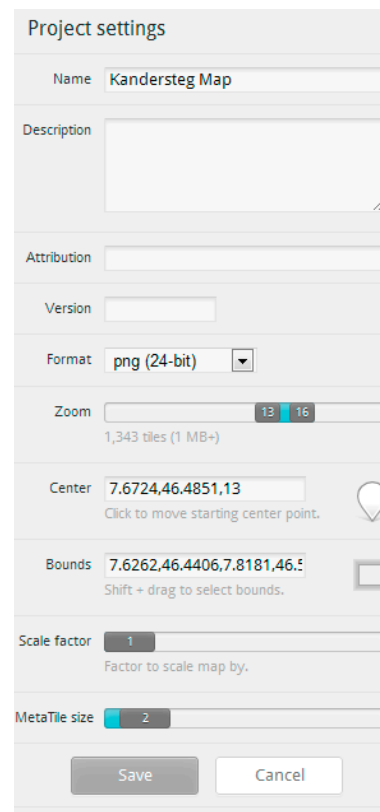


Figure 7: Project settings

10) Now, have a closer look to the CartoCSS and the style define for the landuse layer. The # direct to the layer name to which the style applies. In the round brackets, you find the parameters that define the color and width of the line as well as the fill (color) and opacity of the polygon. Now, we will define a different color for the each different landuse category. To do that, you need to use the following muster ['attribute_name' = 'attribute_value'] { style parameters } (see Figure 8). If you need to check a specific CartoCSS parameter, do not forget you can access the documentation by clicking on **{ }** in the bottom left corner.

```

Kandersteg Map
style.mss x +
1 Map {
2   background-color: #b8dee6;
3 }
4
5 #k_landuse {
6   ['landuse'='forest'] {
7     polygon-fill:#739C70;
8     line-color: #2e602b;
9     line-width:1.5;
10  }
11 }
12
    
```

Figure 8: CartoCSS first try

11) Define the polygon-fill for all the categories (see Figure 9).

```

Kandersteg Map
style.mss x +
1 Map {
2   background-color: #fffff9;
3 }
4
5 #k_landuse {
6   ['landuse'='forest']{
7     polygon-fill:#739C70;
8     line-color:#2e602b;
9     line-width:1.5;
10  }
11  ['landuse'='residential']{
12    polygon-fill:#d9d2d9;
13  }
14  ['landuse'='railway']{
15    polygon-fill:#c6abab;
16  }
17  ['landuse'='farm']{
18    polygon-fill:#d9d2d9;
19  }
20  ['landuse'='meadow']{
21    polygon-fill:#d3f0bc;
22  }
23  ['landuse'='scout_camp']{
24    polygon-fill:#d3f0bc;
25  }
26  ['landuse'='cemetery']{
27    polygon-fill:#c7b1a5;
28  }
29 }
30
    
```

Figure 9: Style the landuse layer

12) Now, we will see a useful parameter: `polygon-smooth` to make the polygons less angular. You can place the parameter at the top of the style definition, so it will be applied to all the categories (see Figure 10).



Figure 10: Polygon-smooth

!!! Because you also have a line define for the forest category, you also need to use the `line-smooth` parameter (see Figure 11) with it, otherwise you will see a mismatch between the polygon-fill and the line !!!

```
#k_landuse {polygon-smooth:0.5;
  ['landuse']='forest' ){
    polygon-fill:#739C70;
    line-color:#2e602b;
    line-width:1.5;
    line-smooth: 0.5;
  }
  ['landuse']='residential' ){
    polygon-fill:#d9d2d9;
```

Figure 11: Line-smooth

- 13) Add the layer kandersteg_natural.shp and define the style for the category natural=glacier, natural=water, natural=grassland (see Figure 12).

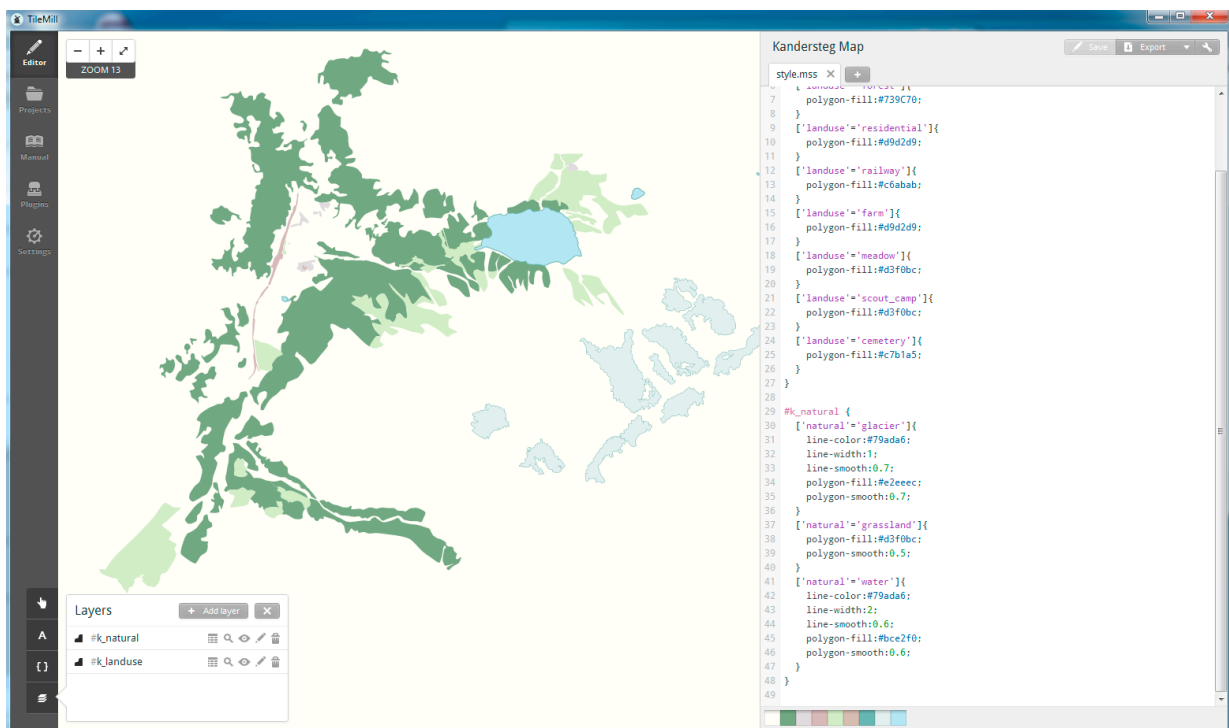


Figure 12: natural layer

- 14) Delete the line parameters for the landuse=forest category.
 15) Add the layer kandersteg_highway_In.shp. Define the symbolization and look at it more in details (see Figure 13):

- a) The dash-array parameter:

```
line-dasharray : 6 , 3 ;
```

- b) Using ::fill and ::case parameters for roads in order to draw a border to the road:

```
::case {
  line-width : 5 ;
  line-color : # FFE200 ;
}
::fill {
  line-width : 2 ;
  line-color : #FFF5A5 ;
```

```
}

```

- a) Defining how the lines are joined and how the angles are drawn with the line-join and line-cap parameters:

```
line-join : round ;
```

```
line-cap : square ;
```

- b) Combination of category selection:

```
[ 'highway'='primary' ], [ 'highway' = 'primary_link' ]
```

- c) Zoom-level dependent symbolization:

```
[zoom >14] [ 'highway' = 'path' ]
```

- d) The gamma parameter for antialiasing:

```
line-gamma : 0.5 ;
```

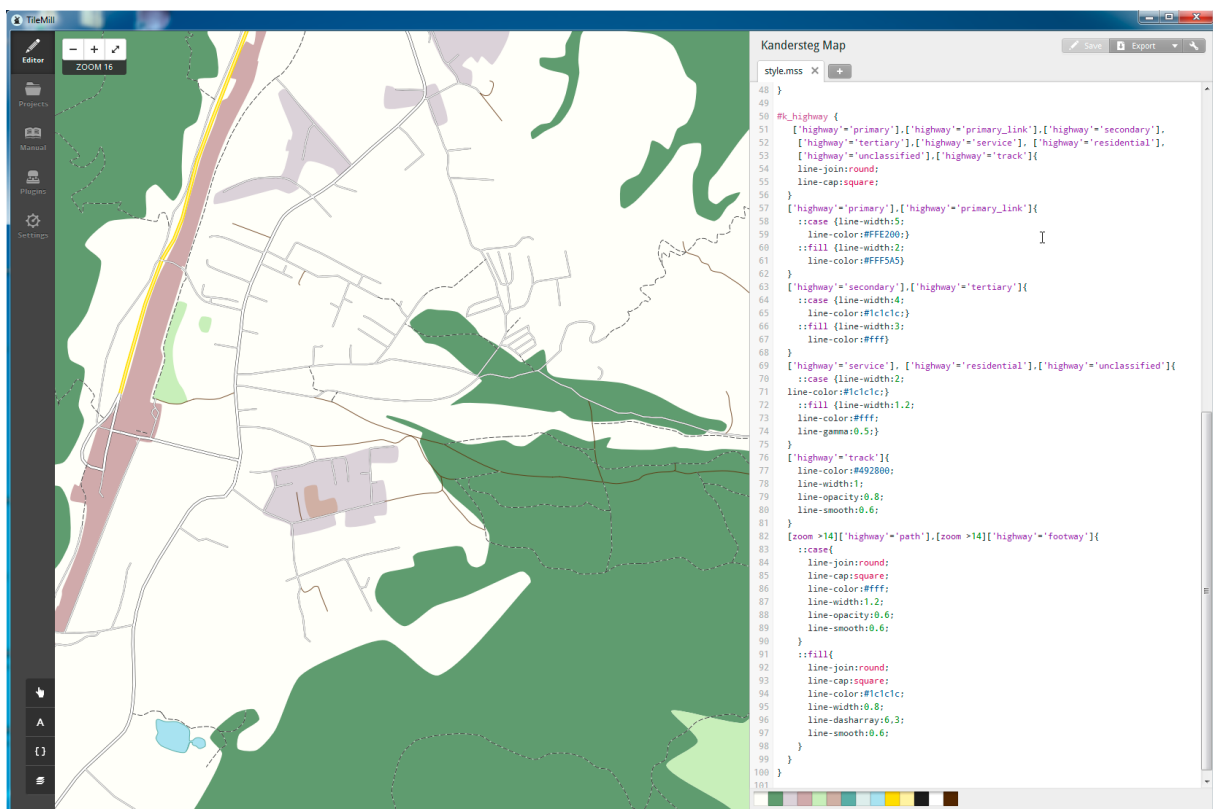


Figure 13: highway layer

- 2) Now, let see how to add some texture to your symbolization. Open Photoshop or GIMP and create a a new document sized 100 x 100 px. Color it in a green hue and add some texture effects to create the feeling of forest texture.
- 3) Note: with GIMP, use the Bucket Fill Tool & Filters; with Photoshop, use the Paint Bucket Tool, Layer Styles and Filters.
- 4) Export as png and save the image in Documents/Mapbox/project/name-of-your-project/images/ and name it forest01.png .

>>> You will need to create the images folder, but by saving the images inside the TileMill project folder, you can relatively reference to it. <<<

- 5) To apply the texture to your symbolization, you can use the following parameter:

```
polygon-pattern-file:url("images/forest_texture.png");
```


- 1) To that, you can add other parameter to refine how your pattern behave with the other layers. The comp-op (combination operation) is very useful is you have relief in the background)

 polygon-pattern-opacity: 0.8;

 polygon-pattern-comp-op: multiply;

>>>> To know more about the different options for the comp-op parameter, you can visit this page: <http://www.mapbox.com/tilemill/docs/guides/comp-op/> <<<<

- 2) Do the same for the grassland and meadow class.
- 3) You can also prepare a background pattern if you want and don't already have a relief and for that the parameter is:

 background-image: url("images/background01.png")

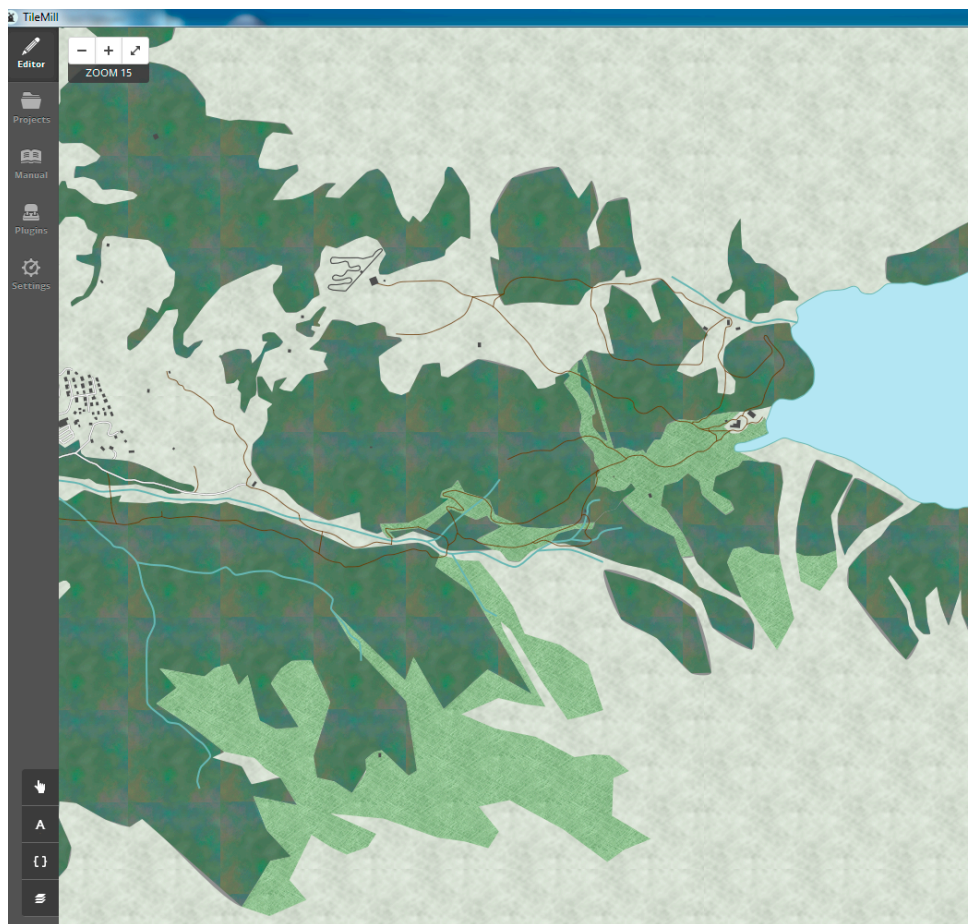


Figure 14: texture-pattern

- 4) Load some extra layers: kanderstaeg_building, kandersteg_railway_In (line), kandersteg_piste_In, and kandersteg_aerialway_In and kandersteg_waterway_In (line) and symbolize them (see Figure 15).


```

104
105 #k_railway {
106   line-width:1.2;
107   line-color:#3f3f3f;
108 }
109
110 #k_building{ [zoom >14]{
111   polygon-fill:#3f3f3f;
112 }
113 }
114
115 #k_waterway {
116   line-width:2;
117   line-color:#79ada6;
118   line-smooth: 0.5;
119 }
120
121 #k_piste {
122   line-join: round;
123   line-cap:round;
124   line-smooth:0.6;
125   ['piste_type' = 'downhill']['piste_diff'='easy']{
126     line-width:3;
127     line-color:#1066C1;
128   }
129   ['piste_type' = 'downhill']['piste_diff'='intermediate']{
130     line-width:3;
131     line-color:#C11010;
132   }
133   ['piste_type' = 'nordic']{
134     line-width:2;
135     line-color:#229E6A;
136   }
137   ['piste_type' = 'sled']{
138     line-width:2;
139     line-color:#684075;
140   }
141 }
142
143 #k_aerialway {
144   |:line, |:hatch { line-color: #1c1c1c; }
145   ::line { line-width:2; }
146   ::hatch {
147     line-width: 6;
148     line-dasharray: 2, 24;
149   }
150 }
151

```

Figure 15: railway, building, waterway, piste and aerialway layers

- 5) For the aerialway, you can use the parameter `:: hatch` to draw small perpendicular segment to the main line (see Figure 16).



Figure 16: hatch parameter

- 6) In order to define the drawing order of the layers, you need to open the tab Layers in the left bottom corner, click on the respective layer icon and drag it to a new spot (see Figure 17).

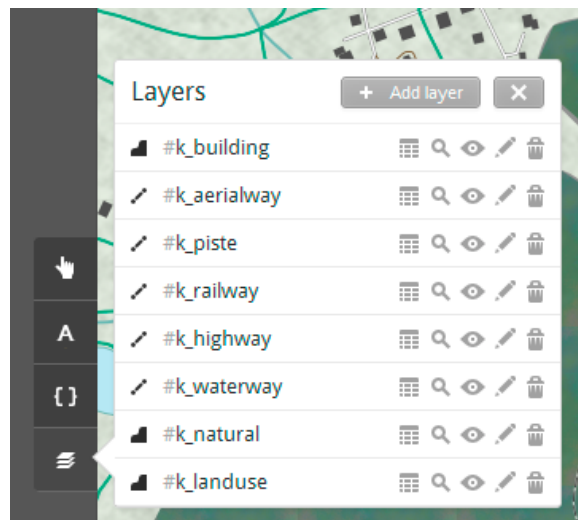


Figure 17: Layer stack

- 7) We still have to see how style point geometries. Load the layer kandersteg_tourism_pt and observe the default symbolization. Points are symbolize using the marker class. By default, a circle is used for point symbolization.

```
#k_tourism {
  marker-width:6;
  marker-fill:#f45;
  marker-line-color:#813;
  marker-allow-overlap:true;
}
```

- 8) You can also use SVG icons and we will use them to symbolize the hotels and the information stands with specific icon. In the tutorial data folder "images", you will find icon for the hotel sand the information stands, copy them into your images folder within your project folder (same as with images for the textures).

```
#k_tourism [[zoom>14]['tourism']='hotel']{
  marker-file:url("images/lodging-18.svg");
  marker-fill:#df534c;
}
  [[zoom>14]['tourism']='information']{
  marker-file:url("images/star-18.svg");
  marker-fill:#df534c;
}
}
```

>>> *These icons have been developed in order to be used with the marker-fill parameter. If you want to know more about the Maki (developed by Mapbox), you can go here <https://www.mapbox.com/maki/>. <<<*

- 9) When you are happy with your map, you can go the next part.

(B) Export to MapBox

- 1) Click on the parameter icon in the top right corner to define the parameters for the export. Chose the following:

Zoom: 13 to 16
 Center: 7.6794,46.4965,14
 Bounds: 7.5893,46.4303,7.8106,46.5398
 Scale factor: 1

>>> *It is important to limit the number of zoom levels for which the map is available for question of tile package size. <<<*

>>> *The scale factor will scale your symbolization. <<<*

- 2) Click **Save**.
- 3) Click on **Export > Upload** and enter your Mapbox account information. verify that the information are correct and click upload.
- 4) When finished uploading, you can click on **View** to see your map in the browser.

>>> *You also have the possibility to export you map in another format. If working with your own Tile Server, you can pick MBTiles. You can also export your map using the well-known pdf and png or as drawing format svg. <<<*

- 5) You need to do one more thing to get your map ready to be served in a webpage. You need to create a project on your MapBox account. Go on your Mapbox account, if not already, go to **Data**, select your map and click on **+Create project**.
- 6) It will bring you to the position 0,0. Click on the magnifying glass and type in "Kandersteg" to be brought to the location of your map. Set the center and zoom level right and then under **Project > Settings**, give a name to your project. Click **Save**.

(C) Integration in a webpage

- 1) Create an html page in a text editor, link the mapbox library and the style sheet.

```
<!DOCTYPE html SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Kansdersteg</title>
    <meta name='viewport' content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no' />
    <script src='https://api.tiles.mapbox.com/mapbox.js/v1.6.2/mapbox.js'></script>
    <link href='https://api.tiles.mapbox.com/mapbox.js/v1.6.2/mapbox.css' rel='stylesheet' />
```

- 2) Add a style for the body and the map element

```
<style>
body { margin:0; padding:0; }
#map { position:absolute; top:0; bottom:0; width:100%; }
</style>
```

- 3) Create and initiate the map.

```
<script>
// Create a map in the div #map
L.mapbox.map('map', 'djana.isiphjnf');
</script>
</head>
```

```
<body>
```

- 4) Add the map element in the body

```
<div id="map"></div>
<script>
  // Create a map in the div #map
  var map = L.mapbox.map('map', 'djana.i5iphjnf');
</script>
</body>
</html>
```

- 5) That's it! For more hints and capabilities of the Mapbox-TileMill combination, read the next section.

Comments

The Mapbox.js library used in this example builds on top of Leaflet.js and extends the Leaflet classes. Mapbox has a very well done and complete documentation on their website, with complex examples. You should definitely check that page if you want to create more complex symbolization: <https://www.mapbox.com/mapbox.js> . This also means that you have all the functionality of Leaflet within the MapBox.js API.

Another point to consider is the interactivity. You can already define the legend as well as other interactivity aspect in TileMill, see <https://www.mapbox.com/tilemill/docs/crashcourse/tooltips/>.

Other Links that might be useful:

TileMill Crash Course:

<https://www.mapbox.com/tilemill/docs/crashcourse/introduction/>

Leaflet.js website:

<http://leafletjs.com/>

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.