

# Master Thesis

---

## **Development of a framework for designing a map style across several scales and its implementation**

submitted by      Laura Eichler  
born on            21.10.1992    in      Suhl

submitted for the academic degree of  
Master of Science (M.Sc.)

Date of Submission      05.11.2018

Supervisors              Prof. Dipl.-Phys. Dr.-Ing. habil. Dirk Burghardt  
                                 Institute of Cartography, TU Dresden

Mathias Gröbe  
Institute of Cartography, TU Dresden

Dr.-Ing. Christian Murphy  
Chair of Cartography, TU Munich

---

## Statement of Authorship

Herewith I declare that I am the sole author of the thesis named  
„Development of a framework for designing a map style across several scales and its  
implementation“  
which has been submitted to the study commission of geosciences today.  
I have fully referenced the ideas and work of others, whether published or unpublished. Literal or analogous citations are clearly marked as such.

Dresden, 05.11.2018

Signature

---

## Acknowledgements

First I'd like to express my gratitude to one of my supervisors Mathias Gröbe for the support, guidance and patience throughout the master thesis. Your open door, encouraging feedback and engagement kept me on track and have been beyond helpful. Thank you sincerely!

To Juliane Cron many thanks for the dedication and support in organizing the master programme and still be helpful with individual issues. It was always appreciated and takes great influence on the programme.

I am immensely grateful to be part of the Cartography master programme and have enjoyed my time, thanks to the people involved. To be among the #CartoSquad16 has been an amazing experience that will not be forgotten.

Furthermore, I would like to thank my partner Falco who supported me in my studies even away from Dresden and through stressful times by always trying to lift some weight from my shoulders and make me smile.

Lastly, it is beyond words on how thankful I am for the support of my parents, especially my mother. I am joyous for the luxury to have studied a field that holds my compassion.

Thank you all.

## Table of Contents

Statement of Authorship .....	2
Acknowledgements .....	3
Table of Contents.....	1
List of Figures .....	4
List of Tables .....	6
List of Abbreviations .....	7
1 Introduction .....	8
1.1 Terms in Web Mapping .....	8
1.2 Interactivity .....	9
1.3 Development and Use of Web Maps.....	9
1.4 Research Objectives and Outline.....	12
1.5 Thesis Structure .....	13
2 Theoretical Background .....	14
2.1 Map Design Components .....	14
2.1.1 Colours .....	14
2.1.1.1 Colour Dimensions .....	14
2.1.1.2 Colour Models .....	16
2.1.1.3 Web Safe Colours .....	16
2.1.1.4 Colour in Map Design .....	18
2.1.2 Typography .....	21
2.1.2.1 Availability and Encoding.....	23
2.1.2.2 Typeface in Map Design .....	24
2.2 Multi-Scale Maps.....	26
2.2.1 Tiled Maps.....	27
2.2.2 Theoretical Principle behind Tiled Maps.....	29
2.2.3 Raster Tiles .....	31
2.2.4 Vector Tiles.....	32
2.2.5 Projection.....	36



2.2.6	Multi-Scale Map Design Approaches .....	37
3	Methodology & Implementation .....	40
3.1	Reviewing Existing Map Styles.....	40
3.1.1	Map Continuity & Colour Design.....	40
3.1.2	Typography .....	44
3.1.3	Icons & Symbols .....	44
3.1.4	Objects and features .....	45
3.2	Visualization Components .....	46
3.3	Workflow .....	47
3.3.1	Editorial Approach .....	47
3.3.1.1	Content.....	50
3.3.2	Available Hosted Data and Tile Sources .....	50
3.3.2.1	OpenMapTiles / MapTiler Cloud .....	51
3.3.2.2	Mapbox .....	52
3.3.2.3	Thunderforest.....	54
3.3.2.4	Other & Custom.....	55
3.3.3	Applications for the Process of Multi-Scale Map Design .....	55
3.3.3.1	Google Maps APIs Styling Wizard .....	56
3.3.3.2	Maputnik .....	57
3.3.3.3	Mapbox Studio .....	62
3.3.3.4	MapTiler Customize Tool.....	64
3.3.3.5	Esri ArcGIS Vector Tile Style Editor (beta) .....	65
3.3.3.6	Result Comparison Tools .....	68
3.3.4	Client Applications .....	70
3.3.4.1	MapboxGL JS .....	72
4	Result & Discussion.....	74
4.1	Finished Map Design .....	74
4.2	Layer Order and Zoom Dependence .....	78
4.3	Map continuity analysis.....	80
4.4	Colour blindness .....	81

---

4.5	Restrictions .....	82
5	Conclusion .....	83
	References .....	85
	Appendix I: Overview of Websafe colors.....	91
	Appendix II: ScaleMaster Operators .....	92
	Appendix III: Worksheets to Aid Editorial Process.....	94
	Appendix IV: Layer Schema of OpenMapTiles and Mapbox.....	95
	Appendix V: Comparison of Finished Map Styles .....	101

## List of Figures

Figure 2.1: upper scale shows decreasing saturation values from left to right (100%, 80%, 60%, 40%, 20%); lower scale shows decreasing lightness value from left to right (100%, 80%, 60%, 40%, 20%) .....	15
Figure 2.2: Additive colour theorem, big bubbles as primary, small as secondary colours...	16
Figure 2.3: simultaneous contrast shows, how the grey area is perceived as slightly darker on the left than the gray area on the right even though being the same gray .....	19
Figure 2.4: The lines resemble colour-blind confusion. Colours, that are two or more zones apart are easier to discern. (Brewer 2016, p. 171) .....	20
Figure 2.5: terms of typography illustrated .....	22
Figure 2.6: visual variables regarding typeface differentiating nominal phenomena and qualitative data (Muehlenhaus 2015, p. 113).....	24
Figure 2.7: examples of fonts that work well on web maps (Muehlenhaus 2014, p. 119)....	26
Figure 2.8: tile schema visualized as pyramid; smallest scale with fewest tiles on top, tile numbers increase with scale .....	27
Figure 2.9: Tile coordinates of a web map at zoom level 3 (Schwartz 2018).....	29
Figure 2.10: Overview of tile creation and retrieval process in Raster and Vector Tiles (adapted from Mathias Gröbe).....	30
Figure 2.11: Vector tiling principle (Gaffuri, 2012, p. 91) .....	33
Figure 2.12: Maptiler vector map („streets“) with building details of Dresden Cathedral on z17 .....	34
Figure 2.13: Web Mercator projection combined with Tissot’s Indicatrix (done in QGIS) ....	37
Figure 2.14: Example section of a ScaleMaster diagram (Roth, et al., 2011, p. 33).....	38
Figure 3.1.: Overview of map complexity throughout zoom levels 4 to 16.....	41
Figure 3.2: snippets of OSM carto web map style, top to bottom: z7, z8, z11, z13.....	42
Figure 3.3a (above): Google Maps default colour scheme (2018).....	43
Figure 3.3b (middle): Bing Maps default colour scheme (2018).....	43
Figure 3.3c (below): (German) OpenStreetMap default colour scheme (2018) .....	43
Figure 3.4: Visualization components overview .....	47
Figure 3.5: Workflow of Methodology .....	47
Figure 3.6: Example of work sheet aiding web map content assignment (see appendix III) 50	
Figure 3.7: OMT polygon, line and point layer schema visualized as sunburst diagrams; detailed version found in appendix IV .....	51

Figure 3.8: Mapbox polygon, line and point layer schema visualized as sunburst diagrams; detailed version found in appendix IV .....	52
Figure 3.9: Mapbox Terrain Layer Schema .....	53
Figure 3.10: Mapbox Traffic Layer fields.....	54
Figure 3.11: GUI of Google Maps APIs Styling Wizard .....	56
Figure 3.12: GUI of Maputnik (Github version) .....	58
Figure 3.13: Inspect mode on Zurich main station .....	61
Figure 3.14: GUI of Mapbox Studio.....	62
Figure 3.15: Map view change when adding a new layer.....	63
Figure 3.16: Value Size with stops at z7 (11.5px) and z15 (20px); currently at z11.1 having a size of 15.92px. ....	64
Figure 3.17a (left): Maptiler CUSTOMIZE GUI.....	65
Figure 3.17b (right): MapTiler Edit Style (advanced editing) GUI .....	65
Figure 3.18: User Interface of Esri ArcGIS Vector Tile Editor (beta) .....	66
Figure 3.19: Edit modes of Esri ArcGIS Vector Tile Editor (beta); Edit Layer Styles (Category & Drawing Order), Edit Colours, Edit Icon and Patterns.....	67
Figure 3.20: left showing symbolization of roads as white lines, right showing same features with grey casing .....	69
Figure 4.1: <i>Terrain</i> Map snippet displaying all land cover layers.....	75
Figure 4.2a: building fill layer at z15 with duplicate building shadow fill layer (translated 3, 1).....	77
Figure 4.2b: building fill extrusion layer at z16.....	77
Figure 4.3: OSM Liberty vector style showing fill extrusion layer in z14.5 .....	77
Figure 4.4: Overview of map complexity throughout zoom levels 4 to 16.....	80

## List of Tables

Table 2.1: web safe colours expressed differently .....	17
Table 2.2: example table about zoom levels (OSM Wiki, retrieved 30 July 2018) .....	28
Table 3.1: Colour palette for web map design .....	49
Table 3.2: used sources in web map .....	68
Table 3.3: Metropolis font weight usage .....	70
Table 3.4: Overview of client applications supporting custom styles .....	72
Table 4.2: Addition and removal of feature layers dependent on zoom level .....	79
Table 4.3: Comparison of <i>Terrain</i> in two different simulated colour blindness .....	81

## List of Abbreviations

API – Application Programming Interface  
CSS – Cascading Style Sheet  
DEM – Data Elevation Model  
GUI / UI – Graphical User Interface / User Interface  
HEX – Hexadecimal  
HTML – Hyper Text Markup Language  
JS – JavaScript  
JSON – JavaScript Object Notation  
LoD – Level of Detail  
MBtiles – Mapbox tiles  
MVT – Mapbox Vector Tile  
OMT – OpenMapTiles  
OSM – OpenStreetMap  
PBF – Protocolbuffer Binary Format  
RGB – Red Green Blue  
SVG – Scalable Vector Graphic  
VMT – Vector Map Tiles  
Z\* / z\* - Zoom level, e.g. z3 means zoom level 3

## 1 Introduction

Web maps and their applications are a major part of our daily lives nowadays, whether for navigational purposes, displaying and distributing online data or business online presence (Hennig, 2016). Cartographers have developed various styling guidelines for all types of maps, but they lack the specialty of web maps as they have specific features other than print or on-screen maps. By now, a web map entails interactivity and a series of scales (called “zoom levels”) that can be dynamically switched, so that basically an individual map has to be designed for every zoom level. This thesis is focused on web maps mentioning elements of multimedia cartography when necessary. It aims to offer guidelines on zoom-level design on a deeper level than is has been mentioned before (Peterson, 2015; Brewer & Bittenfield, 2007), and applying them on an example base map for a thematic map.

### 1.1 Terms in Web Mapping

A *web map* is a medium in the World Wide Web, which combines geographical and other data to engage the user in exploring and delving into a topic. Usually a web map consists of a base map and additional layers which include the geo-referenced data or interactive applications. Web maps can be static, i.e. represent an image of a map, or interactive which implies, that the user can interact with the map in form of selecting layers of different data sources or use exploration techniques such as *panning* and *zooming* (Möller, 2016).

The *base map* (or *reference map*) in terms of web mapping are maps accessible through a public interface and works as foundation for building new map services, often called mashups (Schmidt & Jörg, 2012). It is the most important layer of a web map and helps the user to provide geographical context to data or orientation in the surrounding while navigating. Additional layers overlay the base map. Depending on the purpose of the map, the base map can be subtle to give additional layers more importance or it can be more prominent if needed, e.g. when used in navigation. The purpose of the map additionally determines which reference information is displayed, e.g. geological information is not needed on a map about election outcomes (Dent, et al., 2009).

## 1.2 Interactivity

*Panning*, as mentioned before, is part of the user interface. It lets the user move the map, if it is bigger than what can be displayed on the screen. It is usually done by dragging a map via mouse click or keyboard (Muehlenhaus, 2014).

*Zooming*, is part of the user interface as well. This exploration technique enables the ability to change the scale of a map and dive into it to explore more detailed regions or get an overview of the overall map. So called “slippy maps”, referring to modern web maps with (pre-) rendered tiles, use these techniques, i.e. zooming and panning, by default. The particular scale of such an interactive web map is called *zoom level* or *level of detail (LoD)* and it determines what is seen on the map: the lowest zoom level is 0 (z0) and displays the whole world, while zoom level 15 (z15) can show already details of a small village regarding raster tile maps (Peterson, 2015; Mackaness, 2007).

*Rotating* happens on a web map when it is not north-oriented. Interactive web maps, such as Google Maps, offer the feature as a user interface option. Dynamic rotation is seen when web maps are used for navigation: the maps rotates relative to the moving direction (Muehlenhaus, 2014).

## 1.3 Development and Use of Web Maps

With the rise of computer technology and the introduction of the Internet a new media platform was created which re-defined, among others, the mapping process entirely. Early web maps were basically hypermaps, that offered links to further information, but no other functionalities because development was not yet advanced (Muehlenhaus, 2014). MapQuest<sup>1</sup> (1994), acquired by AOL (2000) and later Verizon (2015), is considered as one of the first pioneers in online mapping (Peterson, 2015; Harlan, 2015). The web service was one of the first to offer directions and location requests, but was limited (Purvis, et al., 2007). Access to data and increasing software technologies enabled the production of high quality maps followed by the demand of the use of updated and most recent data.

---

<sup>1</sup> <https://www.mapquest.com>



Roughly a decade after the beginning of the Internet Google Maps was published in February 2005 by Google Inc. and revolutionized the approach of web mapping techniques (Peterson, 2012): a map embedded into a search engine. It offered not only answers to locating and navigation issues but also satellite images for all regions of the world (Purvis, et al., 2007). Furthermore, Google Maps introduced interface functionalities such as the pan-and-zoom toolbar and used raster tiles early on (Gibson & Erle, 2006). With its API, an framework for everyone, it is possible to create a map application tailored to their needs by creating layers and displaying them on the base map of Google Maps (Muehlenhaus, 2014).

The foundation stone of interactive web mapping was laid and other developers followed: December 2005 the Microsoft opponent *Windows Live Local*, today known as *Bing Maps*, was released (Dayley, 2005). Spring 2006 the idea of open geospatial data for everyone was spread and later *OpenStreetMap* (OSM) was launched. Over the next months and years many contributors offered data sets and funding to OSM, e.g. *Yahoo!* Satellite imagery was the base for online mapping in the beginning (Anon., 2018). A community of mappers and those who wanted to be set themselves to map the world and keep the data free, open and updated. These volunteered geographic information (Goodchild, 2007) provide the foundation for numerous applications as OSM is used by plentiful developers and organizations.

*OpenLayers*<sup>2</sup>, *LeafletJS*<sup>3</sup> and *TangramJS*<sup>4</sup> are examples for mapping frameworks based on JavaScript libraries that use web maps established by said companies. *Esri* established its own online web GIS called *ArcGIS online*<sup>5</sup> in 2007 to create, analyse and use web maps combined with several functions adopted from their desktop software *ArcGIS* (Esri, 2006/2007). Another mapping platform called *Mapbox*<sup>6</sup> lets the user create their own custom designed maps and integrate them via APIs in their applications, using the data provided by OpenStreetMap.

---

<sup>2</sup> <https://openlayers.org/>

<sup>3</sup> <https://leafletjs.com/>

<sup>4</sup> <https://mapzen.com/products/tangram/>

<sup>5</sup> <http://www.esri.com/software/arcgis/arcgisonline>

<sup>6</sup> <https://www.mapbox.com/>

Subsequently, the shift from desktop web maps passed over to handheld devices, i.e. smartphones, tablets and even eBook readers, so that new requirements for web maps emerged. Web maps used on mobile devices cleared the way for new functionalities, especially location based services (LBS), where the location of the user is found via GPS (Global Positioning Service) and used. They also allow the user again to be as mobile as with paper maps.

The uses of web maps are diverse, but there are two main purposes where web maps are mostly applied. For one, locating a place and/or finding the most effective route from location points. Many business websites integrate a map marking their location on it<sup>7</sup> giving directions to the customers. Another use case would be public transport, where they combine search results with map visualizations<sup>8</sup>. On the other hand, web maps are used to display and distribute data. Developers use web maps with the help of mapping frameworks and bring together two components: the web map as a base layer and a layer which contains the geo-referenced data<sup>9</sup>.

Map purpose dependent scales aside, web maps use multi-scale maps that offer various level of details depending of the zoom level, as we know from Google Maps, Bing Maps, OSM and other various tiled maps. Samsonov (2011) brings the challenge accompanied by multi-scale map design on point: “Multiscale mapping is among the most prominent and problematic areas in modern cartography. Casting away the limits of fixed-scale maps cartographers go into detail of multiresolution databases design, real-time generalization, map layer structure, scale-dependent behaviour and symbolization, while trying to keep representation clear and credible.”

Zoom-level based design or multi-scale map design is mentioned throughout literature but still in development in regards on specific guidelines. Operators and tools are offered, nevertheless (Roth, et al., 2011; Brewer & Buttenfield, 2007; Samsonov, et al., 2013),

---

<sup>7</sup> Most chain stores offer a map with their locations so the user is able to find the store nearest to them, e.g. IKEA: [https://www.ikea.com/ms/de\\_DE/campaigns/IKEA\\_Standorte.html](https://www.ikea.com/ms/de_DE/campaigns/IKEA_Standorte.html)

<sup>8</sup> Dresden’s public transport DVB (<https://www.dvb.de/en-gb/>) shows connection route results also in a map.

<sup>9</sup> “The food capitals of Instagram” (<https://cewe-photoworld.com/instagram-food-capitals/>) shows a thematic map about the world’s favorite food filtered by Instagram tags. The base map is a simple light world map, while the data layer is prominent with contrasting colours.

guidelines on how to use them in the most appropriate way, are still in development and research only scratched the surface so far.

#### 1.4 Research Objectives and Outline

There are some guidelines of multi-scale mapping (Roth, et al., 2011; Brewer & Bittenfield, 2007; Samsonov, et al., 2013; Peterson, 2015) but seemingly focused on the generalization operations used to design a map and much less on the styling itself (see chapter 2.2.6). This thesis aims to bring together the ongoing research on multi-scale mapping in combination with traditional cartographic techniques as well as specialized methods regarding web development. More precisely, the main goal of this thesis is to provide a framework of styling rules exclusively developed to assist in styling multi-scale maps.

To accomplish that, an intense literature review is necessary to build a foundation of traditional cartographic design specification and to find out, which are transferable on web map design. Furthermore, some technical background insight is required to investigate the limits of existing techniques and the state of the art.

The specific research objectives of this thesis are:

- Process step to create a multi-scale web map design
- Components of multi-scale web map design

The following research questions will be answered:

1. What is important to pay attention to when designing a multi-scale map?
  - a. What is necessary to consider when designing maps for the Internet?
  - b. How can a harmonized map behaviour be achieved?
  - c. When is it reasonable to add a feature group?
  - d. Which base map styles exist and how do they use multi-scale as a styling component?
  - e. Do the existing base map styles achieve a continuous map appearance through all the zoom levels? How can it be investigated?

2. How can an individual map style be created?
  - a. Which tools are available?
  - b. What are the limits of these tools?
  - c. What techniques offer the most liberty in styling a web map?
  - d. Is the trend towards vector tile mapping justified?
  - e. Is the methodology applicable regardless of the technique or does one technique have a significance influence on the methodology?

### 1.5 Thesis Structure

The thesis is outlined in five chapters and appendices. The first chapter covers the introduction outlining thematical background, relevant terms used throughout the thesis and identifying research objectives. The second chapter provides theoretical background about general and web map design and applying them to multi-scale map design. Different forms of tiled maps are investigated and compared. The state of the art of multi-scale map design is presented. The third chapter contains the methodology with implementation. First established web map designs are reviewed followed by the proposition of a workflow to create a web map style. Available data sources and tools are inspected and compared to determine the most suitable combination supporting the implementation of a custom web map style. The fourth chapter is dedicated to the discussion of the result and further evaluation. In the end, chapter five gives an overall conclusion and addresses the previous defined research objectives followed by an outlook of future research questions.

## 2 Theoretical Background

### 2.1 Map Design Components

Similar to paper and desktop maps, choosing between certain style usages of colours and typeface determine the quality of a map (Dent, et al., 2009) and are important tools to know when attempting to style a map. The composition of these elements aggregate to legibility and usefulness to the map reader, so that the cartographer's task is to provide information in an understandable way.

#### 2.1.1 Colours

Selecting colours is the most important step when designing a map (Brewer, 2016). Robinson (1967, p. 50-61) as well as Tyner (2010, p. 62) list reasons why colour should not be seen as an afterthought in map design but rather as one of the intermediate stages:

- Colour attracts attention
- Colour leads the eye
- Colour affects the perceptibility on the map
- Colour is part of the visual levels of a map and unifies the design

To choose a colour some preliminary considerations are attached. Even though colours are a product of physical energy, they impact a user psychologically and emotionally (Dent, et al., 2009) based on associations with experiences made in life, personal preferences and cultural influences. The map designer needs to know the meaning of the colour, to prompt the desired reaction to a map (Atzl, 2016) and it is also necessary to understand the three dimensions of colours: hue, saturation and value in order to develop a sustainable map design.

##### 2.1.1.1 Colour Dimensions

Every colour can be split into hue, lightness/value and saturation. These elements are also factors of other colour models called HSV/HSL, which are generally preferred by computer scientists but not further relevant for web programming.

“*Hue* is the perceptual dimension of colour that we associate with colour names, such as red and yellow.” (Brewer 2016, p.130). Each hue is placed in the visible spectrum of wavelengths ranging from long wavelength (red) to short wavelength colours (blue). It should be mentioned that purple and magenta are products of mixing red and blue as they are not part of the visible spectrum. Monitors and screens are able to only produce a reduced set of the visible wavelengths (RGB), while the sunlight contains a full range of colours.

*Saturation* describes the colour effect, in other words, the brightness of a hue. The highest saturation is found in primary colours, as they are pure and unmixed. Mixing colours results in reduced saturation, moreover mixing colours with achromatic colours (white, grey, black) takes out the most saturation of the hue. The component is also termed as colourfulness, purity, chroma and intensity. “[It] can be understood by comparing a colour to a neutral grey. With the addition of more and more pigment of a colour, it will begin to appear less and less grey, finally achieving a full saturation or brilliance.” (Dent, et al., 2009, p.252).



Figure 2.1: upper scale shows decreasing saturation values from left to right (100%, 80%, 60%, 40%, 20%); lower scale shows decreasing lightness value from left to right (100%, 80%, 60%, 40%, 20%)

*Lightness* and *value* are both terms used in literature for the same colour component. It specifies how light or dark a colour hue is and “describes how much light appears to reflect (or emit) from an object compared to what looks white in the scene” (Brewer 2016, p.131). A shade is created when black is added, similarly a tint is created by mixing the hue with white. Naturally darker colour hues (blue, purple, red) result in better nuanced shade colours while bright hues (yellow, green) allow for better tinting (Muehlenhaus, 2014). In mapping techniques lightness is used to represent ranking of mapped information. Light hues are associated with low data values and dark colours with high data values.

### 2.1.1.2 Colour Models

To create colours technically reading it out on digital output device a colour model is required. It describes the value through coordinates in a three-dimensional space of a colour space. Depending on the presentation mode, whether the item (picture, poster, map) is viewed on a screen or printed, the colour models are either of additive (RGB) or subtractive (CMY) nature. The latter is applied when mixing colour pigment and therefore used for printed maps. As this thesis focuses on on-screen maps only, the relevant colour model is the additive one (see fig. 2.2), where colours are based on three main spectral colours red, green and blue, producing all other colours by adding light energy and projecting one above the other (imagine light rays). The colours are described with values within 0 and 255 of each primary colour: (R, G, B). For example, white is defined as (255, 255, 255) while pure red as (255, 0, 0). Therefore, a total of around 16.7 Mio colours are set in RGB (8-bit).

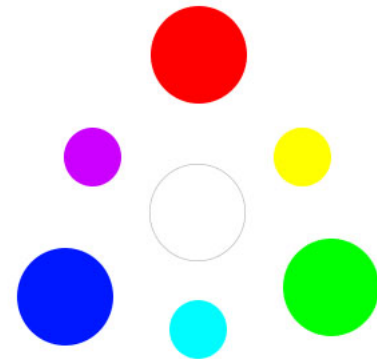


Figure 2.2: Additive colour theorem, big bubbles as primary, small as secondary colours

our model is the additive one (see fig. 2.2), where colours are based on three main spectral colours red, green and blue, producing all other colours by adding light energy and projecting one above the other (imagine light rays). The colours are described with values within 0 and 255 of each primary colour: (R, G, B). For example, white is defined as (255, 255, 255) while pure red as (255, 0, 0). Therefore, a total of around 16.7 Mio colours are set in RGB (8-bit).

General rules of the colour model are (Weber, 2008):

- primary colours cannot be generated by mixing colours
- secondary colours are composed of two primary colours (see fig. 2.2)
- tertiary colours consist of one primary colour as well as one secondary colour, that are complementary<sup>10</sup>

### 2.1.1.3 Web Safe Colours

“Computer monitors attempt to produce the same colour but generate colours with slight variations because they have various settings, such as video card resolution, colour calibration, and the ambient light condition that may impact the colour displayed.” (Dent, et al., 2009, p.257). Not every digital hardware device is able to display every colour (Kraak

<sup>10</sup> Complementary means, that the colours are opposite from each other on a colour circle and resulting in white when added together, e.g. green and magenta in RGB.

& Ormeling, 2003). Regarding the web map user, the map designer can only be sure that the web page is opened with any monitor and any browser. There is no knowledge about hardware components or calibration, much less the operating system or web browser. It needs to be kept in mind that the average internet user does not have high end graphic cards so that not all colours can be transmitted correctly (Weber, 2008).

To ensure a consistent and correct colour display on any screen an informal, but generally accepted colour palette for web browsers was introduced by browser developers *Netscape* – *web safe colours*, consisting of 216 different 24-bit-colours describing a 6x6x6-cube within the RGB colour space, that are available on both operating systems Mac and Windows. The RGB values are set in intervals of 51 to establish a uniform coverage of the initially 16.7 Mio colours. Therefore, values of 0, 51, 102, 153, 204 and 255 are defined to fit the criteria of a browser independent colour. These colours are especially qualified for backgrounds, buttons, lines, text, simply put, all homogeneous areas - so perfect for map features. They are not suitable for smooth colour hues and images (Weber, 2008). Henceforth, only web safe colours are going to be used in this thesis.

The most common method for representing colour in web pages is *RGB Hexadecimal (HEX)*. “[RGB HEX] breaks RGB decimal values down into six-digit, three-byte number codes that can be used with HTML, CSS, SVG [...], and many other applications.” (Muehlenhaus 2014, p.93). A hexadecimal notation always begins with the pound / hashtag sign (#) followed by two digits representing the red value, two digits for green and the last two expressing the amount of blue. The hexadecimal system uses therefore numbers from 0 to 9 (decimal: 0-9) and the letters A to F (decimal: 10-15) to describe 16 possible values. Regarding the web safe colour space only six expressions are relevant and listed in the table below. An overview of all 216 web safe colours can be seen in the palette in appendix (I). It also includes HEX codes for each colour.

<b>Decimal</b>	0	51	102	153	204	255
<b>Hexadecimal</b>	00	33	66	99	CC	FF
<b>percentage</b>	0%	20%	40%	60%	80%	100%

Table 2.1: web safe colours expressed differently



#### 2.1.1.4 Colour in Map Design

Choosing a colour for a map is influenced by many factors. The following is a list of recommendations on what to consider while electing colours. Base map colour schemes evolved from print map colour schemes so that many guidelines are similar to traditional design rules. As already mentioned, colours are perceived subjectively by users based on experiences and personal associations, so that the choosing process of colours should not be influenced by what the user might like, but by own preferences (Muehlenhaus, 2014).

Large online mapping services (e.g. Google, OSM) have similar standardizations in colour schemes with slight differences and are known by most users. It is favourable to orient on the existing styles, because “maps are always easier to read, when users do not have to spend time learning a set of new symbology and colour references. Familiarity helps map users pick up on visual cues of what they are viewing.” (Muehlenhaus 2014, p.102). That being said, there are colour connotations that most map readers assign unconsciously and should be kept in mind. Most obvious example would be to not assign the colour blue to any feature other than water bodies<sup>11</sup> (Peterson, 2015). On the other hand, colour connotations are not universal, meaning colours do not have the same meaning in all parts of the world. Cultural background can assort colours differently than they are assigned purely by symbology (Robinson, et al., 1995). For example, using the colour green in thematic maps to symbolize thematic data, it could stand for “fertility and paganism in Europe, sacred by Muslims [or] mourning and unhappiness in Asia” (Krygier und Wood 2016, p. 266), but is mostly recognized as features related to nature.

Consider the map purpose or rhetorical appeal of the map to decide if it is necessary to create a wholly new colour scheme. If it stands out it certainly attracts attention. Additionally, apart from the purpose of the map keep in mind the environment in which the map is implemented, i.e. is the style corresponding to the web site, the mobile app or operating system? “[The] map communication will be more effective if your design fits within the cohesive whole of the packaging [...]” (Muehlenhaus 2014, p.102). The so called *colour echoing* by using colours appearing in the map also in elements outside the

---

<sup>11</sup> Other colour associations could be white for ice, green for environment/nature, beige for elevation or terrain.

map ensures that the map is not compete with and instead complemented (Peterson, 2015).

Dent et al. (2009, p. 258) mention a study from 1970<sup>12</sup> where subjects were asked about colour combinations of object and background - object being the foreground or figure of focus. Results regarding lightness values included:

- high lightness contrast is important to achieve a pleasant object-background combination
- the background colour should either be light or dark, a value in-between produce poor results
- “[...] an object colour must stand out from its background colour by being definitely lighter or darker. This is the single most important finding of the study for the cartographer”.

Peterson (2015) also speaks about the figure-ground rule, where certain objects need to be separated from its surroundings although the task of the background is to guarantee geographic context, which is as important as emphasizing the foreground. The author as well as Traun (2016) mention examples such as contrasting colours, or border highlighting and later on speaks more detailed about contrast explaining that high saturated coloured objects shift the viewer’s focus to the important object in the foreground separating it from the opposing saturated background.

Likewise, colour interactions designate the appearance of a colour, being dependent to the colour surrounding. An optical illusion can happen, where an object colour is perceived differently regarding the background colour, even though being the same and reverse, see fig. 2.3. This is called *simultaneous contrast*. “Carefully consider the visual difference between different colours on your map. If you intend for your map to



Figure 2.3: simultaneous contrast shows, how the grey area is perceived as slightly darker on the left than the gray area on the right even though being the same gray

---

<sup>12</sup> H. Helson and J. Lansford. 1970. „The Role of Spectral Energy of Source and Background Colour in the Pleasantness of Objects“. In *Applied Optics* (9: 1513 - 62)

distinguish specific data from other data, use colours that have high visual difference.” (Krygier und Wood 2016, p.264).

Based on this, the logical structure of the data should match the perceptual structure of the colour palette, meaning that data sets can be arranged in a sequential, diverging or qualitative way (Brewer, 2016). To ensure readability, colours reflecting the data should not correspond badly to the colours of the base map. Also “[...] the human eye can only distinguish between five shades of the same colour (hue).” (Peterson 2015, p.106). This is primarily important regarding qualitative data visualizations, as in choropleth maps, but can also be applied on base maps. Resulting from the previous statement, when categorizing features within a feature class, they should not have more than five divisions.<sup>13</sup>

It is almost always mentioned in literature and important to remember that there could be users that have a colour vision deficiency, also called colour blindness (Peterson, 2015; Brewer, 2016; Dent, et al., 2009; Muehlenhaus, 2014; Robinson, et al., 1995). “Approximately 4 percent of the population has some degree of colour vision impairment (approximately 8 percent of men and less than 1 percent of women).” (Brewer 2016, p.169). Colour vision deficiency describes the inability to see certain colours or colour differences, the rarest form being only able to see black



Figure 2.4: The lines resemble colour-blind confusion. Colours, that are two or more zones apart are easier to discern. (Brewer 2016, p. 171)

and white, the most common one being red-green deficiency. In-between is the blue-yellow deficiency located, where no distinguishing between these two colours can be made (Peterson, 2015). Hence, there are colours that should never be combined. Brewer (2016) also recommends colour pairings that can be used regarding most common colour vision impairments<sup>14</sup> and further proposes a diagram of colours combined with confusion lines to determine colour pairings that are sufficient for colour blind people (see fig. 2.4). Most

<sup>13</sup> Example: The category or feature class is green areas, the feature types are park, forest, nature reserves, national park, etc.

<sup>14</sup> red/blue, red/purple, orange/blue, orange/purple, brown/blue, brown/purple, yellow/blue, yellow/purple, yellow/gray, blue/gray (Brewer 2016, p. 169)

troublesome colours however, seem to be pure reds and greens as they are not detectable from another for people with red-green vision. Therefore, colours should be tinted with some amount of blue or yellow to avoid this problem. It is advisable to either start with a colour palette that is visible to the whole audience (Colour Oracle<sup>15</sup>) or to check the map after finishing with a tool / plugin to revise the colouring (Peterson, 2015). The open-source GIS-software QGIS includes a tool to simulate different colour blindness (View > Preview Mode > Simulate Colour Blindness ...).

### 2.1.2 Typography

“Fonts are the personality of a map. They may be serious and authoritative or carefree and inviting. Whatever their tone, they need to be legible in the challenging contexts that are characteristic of mapping.” (Brewer 2016, p. 86). With this quote Brewer (2016) may be hinting at the *type personality*. Dent, et al. (2009) introduce this term claiming that a typeface can create a certain mood and attitude. Krygier & Wood (2016) also assert that words on a map are essential for maps. Without it, maps are as simple and powerless as pictures. A designer and cartographer should have basic understanding about how a typeface can affect a map and how to make the most of the map design. Functions of lettering on a map are explained by Robinson et al. (1995, p. 406) :

- Literal symbol: the letters form the name of a term designated to an object
- Locative symbol: the position of the label indicates the type of object it is representing; wide spacing are used for areal or lineal phenomena, otherwise it can support point feature positions
- Nominal symbol: variation of the type style creates differentiations

Tyner (2010, p. 43) addresses the purposes of text on a map as well: to label, to explain, to direct or point, or to establish hierarchy or show size. “[A typeface] is a set of letters and numbers with a unique design.” (Krygier & Wood 2016, p. 238), also known as glyphs. Even though *font* is the more common used term, a letter style group, i.e. *Arial* or *Times New Roman*, are actually referred as *typeface*. Font describes the styling parameters

---

<sup>15</sup> free colour blindness simulator, that shows relevant colours in real-time (Windows, macOS, Linux): <http://colouroracle.org>

of a typeface, e.g. Arial 10-point bold. Even though these descriptions are traditionally accurate, in literature and on the web, it seems, that people do not make a distinction between these two terms (Peterson, 2015), as it is not as seriously taken in the thesis as well. Fonts are pre-installed on the computer and available for purchase or download on the Internet<sup>16</sup>.

The figure below (2.5) gives an overview of the general terms in typography, as they shape the typeface and are included into the font styling (Peterson, 2015). Regarding web maps it should be pointed out that “[t]he best fonts for screens have generous x-heights, spacing between letters, internal letter widths, and consistent thicknesses.” (Muehlenhaus 2014, p. 115). Often, the map type is small. Thus, map makers use fonts with large *x-heights*.



Figure 2.5: terms of typography illustrated

Traditionally, serifed fonts are mostly used for body texts, which have more than one sentence, and are read at close range. Krygier & Wood (2016) suggests that serif type connotes tradition, dignity and solidity. Sans-serif on the other hand are used for wide-ranged reading and short texts, such as titles and implies newness, precision and authority (Krygier & Wood, 2016). Because of the low resolution monitors had, only sans-serif fonts were used for web maps. Muehlenhaus (2014, p. 114) argues, that especially backlit displays (i.e. LED) result in bad legibility. The digital devices seemed to blur the spacing between letters of serif fonts. With the advancement of technology however, devices nowadays have a considerably higher resolution than it used to be, so that web map designers are not restricted to only using sans-serif font families (Peterson, 2015), see chapter 2.1.2.2.

<sup>16</sup> e.g.: <http://dafont.com>, <http://www.fontspace.com/> or <https://www.fontsquirrel.com/>

### 2.1.2.1 Availability and Encoding

A considerable issue that can be caused by using fonts on the web is the character encoding. Using anything other than basic English texts, users may not be able to read what is written. When character encoding is indicated, it aims to ensure that non-Latin characters, e.g. Umlauts and Asian characters, are displayed properly. In addition to problems with the display of the words, search engines can not initialize it. Characters are stored in a *character set*, each character associated with a number, also called code point (Ishida, 2016). To establish multi-lingual displays of text the Unicode Consortium<sup>17</sup> proposed *Unicode* which is a single set of characters with the attempt to include all characters needed for any writing system in the world (Ishida, 2016). It “is a standard method for representing written language in computers” (Gillam, 2002) and is supported by all major browser applications. UTF-8 encoding is mostly used on web pages (Ishida, 2016) with the HTML snippet `<meta charset="utf-8">`. This is observable when reviewing different page-sources of websites<sup>18</sup>. As mentioned earlier, font is a set of glyphs and it usually does not cover every existing character but rather a subset. When choosing fonts, it should be kept in mind, that not every user has access to the same fonts as the developer. While some plug-ins (e.g. Oracle Java) bring a wide range of their own typefaces, HTML and CSS can only access the fonts that are installed on the device that is being used to view the web map. If the chosen font is not available, a fall back font will be selected or a square box (□) is displayed causing depletion to the careful design of the map (Muehlenhaus, 2014). To counter this problem, it is possible to provide an ordered list of fonts in CSS. Another way to include fonts that are not necessarily installed is established by embedding font files, that are hosted online or stylesheets, such as Google Fonts referring to a library. The latter supports web developers by offering an easily use of any font listed when a style sheet is included in the header of an HTML file. Google also “has been developing a font family called Noto, which aims to support all languages with a harmonious look and feel.” (Google Noto Fonts<sup>19</sup>).

---

<sup>17</sup> <http://www.unicode.org/>

<sup>18</sup> [view-source:https://tu-dresden.de/](https://tu-dresden.de/) (line 5), [view-source:https://www.tum.de/](https://www.tum.de/) (line 7), [view-source:https://www.mapbox.com/](https://www.mapbox.com/) (line 3) or [view-source:https://www.apple.com/](https://www.apple.com/) (line 49)

<sup>19</sup> <https://www.google.com/get/noto/>

### 2.1.2.2 Typeface in Map Design

A map should include few fonts, traditionally being one serif and sans-serif font family. Combinations of two sans-serif or two serif fonts should be avoided (Krygier & Wood, 2016). When choosing the two fonts it should be ensured, that they complement each other, e.g. both be informal or modern (Brewer, 2016). Some fonts are designed to be used together, e.g. Stone Sans and Stone Serif or Cambria and Calibri (Krygier & Wood, 2016; Peterson, 2012). There are various sources on the Internet which show a complementing pairing of fonts. Google Fonts, for example, always suggests popular pairings when looking up a typeface and offering live examples on how the composition works. When using two fonts consider what it is applied on. Peterson (2012) also dedicates a whole chapter to the composition of fonts giving use cases and application examples. “Often, a serif font is used to label hydrographic and other physical features, and a sans-serif is used to label cultural features.” (Brewer 2016, p. 90). Nevertheless, using only a single font should be preferred altogether, if possible.

Similarly to paper maps font styles and differing properties should symbolize certain characteristics. Krygier & Wood (2016) argues that type can be used on a map to differentiate qualities as well as quantities and hierarchy or both, see fig. 2.6. Various typefaces are usable for qualitative characteristics of the data. Additionally, type colour hues imply different nominal properties of the object group. Font size changes within the map indicate quantitative dimensions of a feature, e.g. big fonts representing continent names while small fonts label villages (Muehlenhaus, 2014). To achieve a noticeable difference of font sizes, a 3 point difference should be used for medium and larger font sizes and simultaneously a 2 point difference for small size fonts, but type sizes of less than 10 point are hard to read on screens (Krygier & Wood, 2016). Font weight also implies quantitative characteristics (Muehlenhaus, 2014). Bold type signifies importance and makes texts with less saturation, such as grey text, more legible, but may create a plump look (Krygier & Wood, 2016).

Qualitative	Quantitative
Font, <b>Font</b> , Font Hue, <b>Hue</b> , Hue Style, <i>Style</i> Arrangement	CASE, case Size, Size, <i>Size</i> Value, Value, <i>Value</i> Weight, <b>Weight</b>

Figure 2.6: visual variables regarding typeface differentiating nominal phenomena and quantitative data (Muehlenhaus 2015, p. 113)

Capitulatory, Kraak and Ormeling (2003, p. 105f) list these important requirements of letter types on maps:

- Differentiation of importance: they should be able to convey a hierarchy between important and less important features, which can be achieved by variations of font weight, size, spacing, colour value, and upper-/lowercase style
- Differentiation of categories showing nominal data, which can be set by variations of colour value and style
- Usable for all kinds of features (point, line, polygon)

Resolution can affect the legibility. The typeface should maintain readability when set in coarse resolution or in front of a patterns, as labels spread over different features or get interrupted by lines and colours. A contrasting outline (halo, letter casing) cannot only be used for decorative purposes but also help to achieve a better legibility. Brewer (2016, p.100) describes the importance precisely: “Halos are best when they are subtle and relatively thin, especially when used for smaller text and unobtrusive breaks, although occasionally a bolder halo may be useful. [...] The goal when selecting a halo size is to clean up small pieces of line or other content that show between letters while masking a little of the underlying map information as possible.”. Regarding the colour, the outline should be set in the colour of the background or one colour of the background pattern, if possible. However, this task can be challenging when many colours are found in the background (e.g. satellite maps). Additionally, regarding the resolution issue, decorative typefaces on the map should be avoided altogether, claimed by Dent, Torguson und Hodler (2009) as well as by Krygier et al. (2016), but there are exceptions that a certain decorative font can emphasize the artistic style of a web map.

The use of uppercase letters appears often for labels of large features, such as countries, continents or mountain ranges. Less important features remain in mixed-case lettering labels. (Dent, et al., 2009; Krygier & Wood, 2016). When using labels with only uppercase letters the developer should be aware of cases where sans-serif types can irritate the



user: the word “Illinois” shows a capital I followed by two lower-case l, which can be hard to read when the spacing is too small (Brewer, 2016).

Ultimately, Muehlenhaus (2014) offers an overview of fonts with arguments that work well on web maps and are available for the majority of users, see fig. 2.7.

## 2.2 Multi-Scale Maps

As mentioned before multi-scale mapping (called “zoom-level [map] design” by Peterson (2015)) is part of the ongoing research in modern digital cartography (Roth, et al., 2011). It is the design of a map across a range of scales while maintaining a continuous composition and legibility as scale changes. It holds an increasing importance for cartographers of the digital age and invites innovation (Roth, et al., 2011). Knowing operators and methods for these map types is advantageous and time- saving in advance. As Brewer & Buttonfield (2007) pointed out, there are decisions to make when taking on the design of a multi-scale map. First, one needs to know what kind of data and resolution will be used within zoom levels and then contemplate at which zoom levels (or *anchor levels*) will be changes regarding the symbolization and the geometry, along with the introduction of new compiled data. The following chapter aims to provide knowledge about the technical background as well as introduce beneficial tools and design questions regarding multi-scale mapping.

Verdana  
Century Gothic  
Arial  
Helvetica  
Trebuchet MS  
Tahoma  
Corbel  
Myriad Pro  
Georgia

Figure 2.7: examples of fonts that work well on web maps (Muehlenhaus 2014, p. 119)

### 2.2.1 Tiled Maps

A *tiled web map* (previously mentioned as *slippy map*) consists of *tilesets*, which “is a collection of raster or vector data broken up into a uniform grid of square tiles at 22 preset zoom levels.” (Mapbox, 2018). To optimize the map’s retrieval performance, it is cut into these tiles. The origin of the map view as well as its tile index (usually upper left tile) and the zoom level are mandatory to calculate a list of the tiles that are necessary to draw the map. Because only the part of the map is requested that is currently looked upon, tiles allow to browse a large amount of raster and vector data with-

out rendering the whole map image, which could slow the experience and performance. Fig. 2.8 shows how a map is divided the larger the zoom level, sometimes referred as level of detail (LoD) and often illustrated as a pyramid. The amount of tiles to fill a map is also needed. After the calculation the client’s task is to retrieve the particular tiles (formula 2.1).

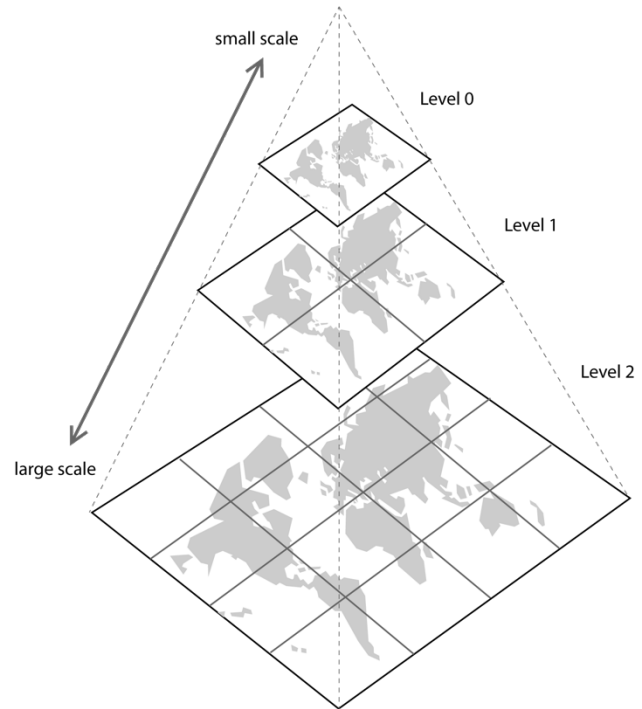


Figure 2.8: tile schema visualized as pyramid; smallest scale with fewest tiles on top, tile numbers increase with scale

$$\text{map width} = \text{map height} = 2^{\text{zoom level}} \text{ tiles}$$

Formula 2.1: The number of tiles increases exponentially, according to the zoom level

To illustrate the extend of zoom levels, the following table (2.2) presents the approximate scale of a zoom level starting with z0 as a single image which is divided in four tiles on the next level etc. The tile itself usually has an extend of 256 by 256 pixel, sometimes even 512 by 512 pixel if the tiles are set in high quality (Diamond, 2016).

<b>Z</b>	<b># of Tiles</b>	<b>Tile width</b> ° of longitudes	<b>~ Scale</b> on Screen	<b>Examples of representative areas</b>
<b>0</b>	1	360	1:500m	whole world
<b>1</b>	4	180	1:250m	
<b>2</b>	16	90	1:150m	subcontinental area
<b>3</b>	64	45	1:70m	largest country
<b>4</b>	256	22.5	1:35m	
<b>5</b>	1 024	11.25	1:15m	large African country
<b>6</b>	4 096	5.625	1:10m	large European country
<b>7</b>	16 384	2.813	1:4m	small country, US state
<b>8</b>	65 536	1.406	1:2m	
<b>9</b>	262 144	0.703	1:1m	wide area, large metropolitan area
<b>10</b>	1 048 576	0.352	1:500k	metropolitan area
<b>11</b>	4 194 304	0.176	1:250k	city
<b>12</b>	16 777 216	0.088	1:150k	town, or city district
<b>13</b>	67 108 864	0.044	1:70k	village, or suburb
<b>14</b>	268 435 456	0.022	1:35k	
<b>15</b>	1 073 741 824	0.011	1:15k	small road
<b>16</b>	4 294 967 296	0.005	1:8k	street
<b>17</b>	17 179 869 184	0.003	1:4k	block, park, addresses
<b>18</b>	68 719 476 736	0.001	1:2k	some buildings, trees
<b>19</b>	274 877 906 944	0.0005	1:1k	local highway and crossing details

Table 2.2: example table about zoom levels (OSM Wiki<sup>20</sup>, retrieved 30 July 2018)

“Each tile is given XY coordinates ranging from (0, 0) in the upper left to ( $2^{\text{level}}-1$ ,  $2^{\text{level}}-1$ ) in the lower right. For example, at level 3 the tile coordinates range from (0, 0) to (7, 7) [...]” (Schwartz, 2018). Fig. 2.9 shows level 3 tile coordinates. A single tile is available through a single URL using HTTP GET request (Sample & Ioup, 2010). The most common request style uses references to the zoom level (z) and the tile coordinates (x,y)<sup>21</sup>:

```
http://tile.{host-address.com}/{z}/{x}/{y}.png22
```

<sup>20</sup> [https://wiki.openstreetmap.org/wiki/Zoom\\_levels](https://wiki.openstreetmap.org/wiki/Zoom_levels)

<sup>21</sup> [https://wiki.openstreetmap.org/wiki/Tile\\_servers](https://wiki.openstreetmap.org/wiki/Tile_servers) gives an overview of different tile web servers including a link to a tile retrieval

<sup>22</sup> example: <https://tile.openstreetmap.org/4/8/5.png>.



Figure 2.9: Tile coordinates of a web map at zoom level 3 (Schwartz 2018)

As soon as the tiles are retrieved the client's task is to arrange them so that a map can be viewed (see next chapter). Usually, the tile retrieval is extended by a border of tiles, where the adjoining tiles (one or two tiles deep) are also fetched and arranged so that the user's movement on the map is not interrupted by another tile retrieval and therefore improving performance (Sample & Ioup, 2010).

### 2.2.2 Theoretical Principle behind Tiled Maps

The principle along with technical components is drawn out in figure 2.10. The tiled map is requested by the client. The client's tasks are counted among the calculation on what tiles are needed to draw the map, fetching the tiles and arranging them together (Sample & Ioup, 2010). The client is able to make a request via hypertext transfer protocols (HTTPs, see previous chapter) and, therefore, must have a connection to the server(s), either locally or over the Internet. It displays and interacts with the web map. Web browsers, i.e. Google Chrome, Mozilla Firefox or Microsoft Edge, are clients (Sterling Quinn, 2018). A database stores the GIS data, e.g. PostgreSQL<sup>23</sup>. If data is requested, it is transmitted to the geospatial server (GIS or map server) which holds the software and processor

<sup>23</sup> <https://www.postgresql.org/>

power to illustrate the map, including functions of feature queries and GIS analysis operations and application of a slicer to generate the map tiles. The tiles of the map consist of images (usually PNG files) or vector data. On that account, the tiles need to either be pre-rendered and stored for distribution or rendered on-the-fly. Therefore, the map server either retrieves the pre-rendered tiles from the database or vector and style data that is then passed on to the web server and client to process. The web server's (proxy server's) task is to protect the user's network from virulent data transfer, such as viruses, by using firewalls as well as responding to the client's request. The data forwarded from the geospatial server also encodes to ensure security. GeoServer<sup>24</sup>, for example, is an open-source geospatial server combined with a web server. Other commonly known map servers (also called *Internet Map Servers, IMS*) are UMN MapServer<sup>25</sup>, Mapnik<sup>26</sup>, Esri's ArcIMS<sup>27</sup> and GLG Map Server<sup>28</sup>.

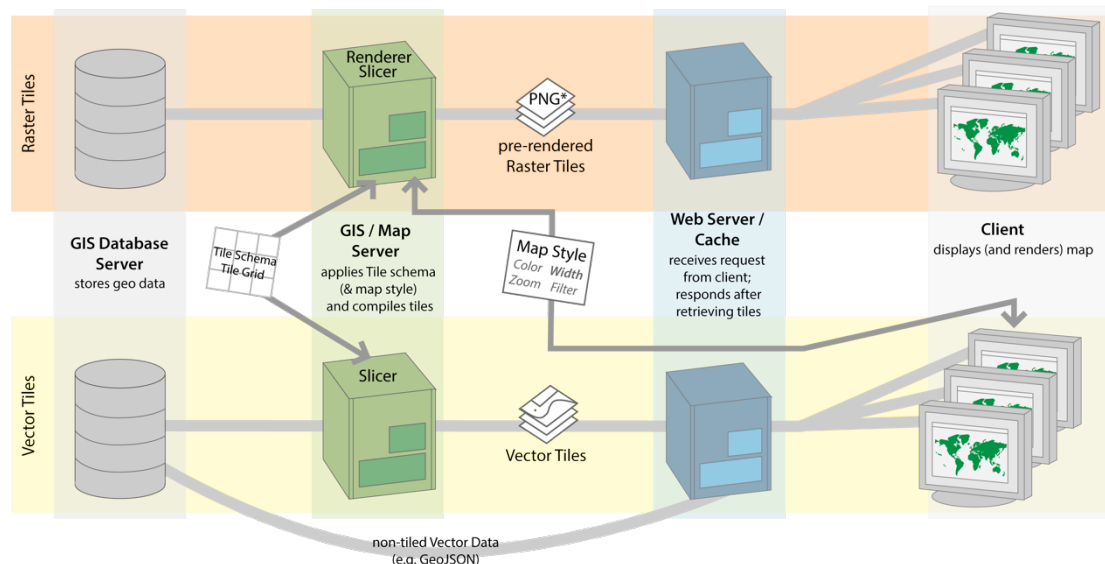


Figure 2.10: Overview of tile creation and retrieval process in Raster and Vector Tiles (adapted from Mathias Gröbe)

A big advantage of the tiling technique is, that the grid images are cacheable. *Tile caching* or *map caching* refers to the process of a pre-generated series of map tiles throughout a range of map scales (Fu, 2015), meaning as soon as a tile is downloaded (either pre-

<sup>24</sup> <http://geoserver.org/>

<sup>25</sup> <https://mapserver.org/>

<sup>26</sup> <https://mapnik.org/>

<sup>27</sup> <http://www.esri.com/news/arcuser/1000/arcims.html>

<sup>28</sup> [http://www.genlogic.com/map\\_server.html](http://www.genlogic.com/map_server.html)

rendered or on-demand) and saved (cached), it can be delivered to the user(s) faster, resulting in short set-up time inside the client, i.e. browser (Onmaps, 2018). The web server's response is therefore fast, retrieving the pre-processed data from the cache and reducing pressure on GIS database and map server. It applies primarily to the most popular used tiles (Peterson, 2012), as they are requested by more clients. Fu (2015, p. 123) also lists the advantages of map caching, being performance, scalability cartographic quality and user experience. According to Quinn & Dutton (2018) tiled maps are the model which could serve complex web applications while being accessed by thousands of users.

### 2.2.3 Raster Tiles

Traditionally, web maps are based on raster files created on a web server and delivered to the client upon request (Esri, 2006). A raster image is divided into small, manageable areas, which are stored in a database, usually sorted by the zoom level. When creating an image raster system, the image tiles are pre-rendered, because reformatting, scaling and projecting of the raster data are performed during the tile creation process (see previous chapter). Due to the amount of time the pre-rendering process can take, updating the data and changing the style is immensely time-consuming (Sample & Ioup, 2010). Updating the raster tiles in bulk can result in outdated maps. The GIS and map servers provide a “high-end cartographic capabilities” such as advanced label placement and symbology, but cannot be changed once it is rendered so that the developer has no influence on the user experience, such as changing the language or rotating labels when the device is rotated. On the other hand, encoding issues of characters are eliminated, because only the developer needs to have the character encoding and font file available. Raster tiles are fast to transmit and interpretable by most common mapping software application, so that they can be used by a wide range of applications and devices (ArcGIS Online, 2018) and are especially useful for satellite imageries and DEMs (TileLayers, 2018). Additionally, raster tiles promise a good performance on a wide range of applications as the use of computing power is low in comparison to vector tiles, creating smooth panning which in return results in a positive user experience. On modern mobile devices raster tiles do not accomplish the resolution requirements as screens of smartphones and tablets are much higher than the usual tile's resolution of 96dpi, so that the map appears to be pixelated on

occasion. With the extend of LoDs and the number of tiles, the required storage increases as well because image files require more storage and take longer to download compared to text (Peterson, 2012). Individual map styles need to be pre-rendered, so that the same data in different styles adds exponentially using up more storage space. To give an illustrative example, Geofabrik offers a tool, that summarises a snippet of tiles outlining number of tiles and storage for each zoom level when a map extend is chosen. The default example<sup>29</sup> presents the storage of raster tiles of Germany's extent, which already consists of more than 320GB on 13 zoom levels. A bounding box including the whole world is increases up to 18TB. In case of OSM not every tile is pre-rendered, as it turned out that only approx. 1.79% of the tiles are viewed (OSM Wiki<sup>30</sup>, retrieved 20 Aug 2018). This is due to the majority of tiles situated at zoom level 18 with most tiles showing nothing of interest, such as ocean.

#### 2.2.4 Vector Tiles

"Vector tile layers deliver map data as vector files (for example, PBF format) and include one or more layers that are rendered on the client based on a style delivered with the layer." (TileLayers, 2018). Other vector formats are MVT, JSON or GeoJSON and SVG. Contrarily to raster tiles, the data is stored as vector representation, i.e. points, lines and polygons, occasionally combined with attribute data that can hold additional information, such as house numbers or place names. They are interpreted by the client's application and rendered as they are requested by using WebGL<sup>31</sup>. Because of that, performance of the map user's system may be reduced while relieving developers hardware accelerations (Milner, 2017). With the caching of requested tiles this issue can be improved. Fig. 2.11 illustrates the principle of vector tiling (Gaffuri, 2012, p. 91) and Sample & Ioup (2010, p. 194) outlines the process on how vector tiles are created:

---

<sup>29</sup> <http://tools.geofabrik.de/calc/>

<sup>30</sup> [https://wiki.openstreetmap.org/wiki/Tile\\_disk\\_usage](https://wiki.openstreetmap.org/wiki/Tile_disk_usage)

<sup>31</sup> The JavaScript API WebGL renders 2D and 3D graphics supported by most modern web browsers without an additional use of plugins. (<https://en.wikipedia.org/wiki/WebG>)

1. Tile must be chosen
2. Bounding box of tile needs to be calculated
3. Requesting the features as vector data within the bounding box
4. Vector data needs to be rendered to tile
5. Arranging the tile images

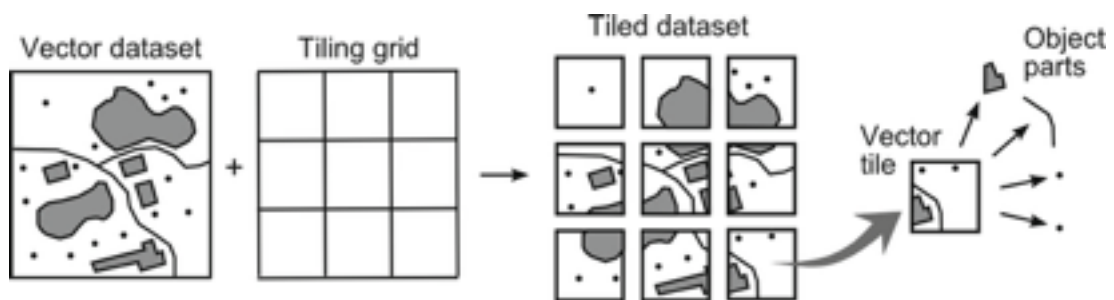


Figure 2.11: Vector tiling principle (Gaffuri, 2012, p. 91)

The smaller file size results in lower costs for storing and serving the tiles, but the file size of a vector tile is much larger than that of a raster tile for detailed maps and higher zoom levels. The main obstacle is performance problems due to the use of too detailed vector data and therefore a higher processing power at the end of the user, which explains why raster tile maps are still vigorously used. With the technological improvement in the memory, processing and connectivity of customer devices, the trend towards vector maps continues and eventually takes over web maps as a technique (Gaffuri, 2012; Fischer, et al., 2016). Furthermore, a LoD appropriate to the zoom level works against the cause of it. Multi-scale vector databases and MRDBs are the subsequent step for this purpose. Multiple representation databases (MRDB) are compilations of the same geographical data stored in a database and linked with the characteristic of generalization for the use of a particular (range of) scale (Roth, et al., 2011; Hahmann & Burghardt, 2010). More precisely, “[a] MRDB consists of various representation levels with different degrees of geometric or semantic abstraction providing a set of different views of the same object [...]” (Sarjakoski, 2007, p. 25). Vector tiles are cost and time efficient when it comes to updating the data and map styles, so that even real time data, e.g. traffic data, is possible to be rendered on-the-fly. Smooth transitions between zoom levels and less pixilation add to the benefits of vector tiles.





are integrated<sup>35</sup>, made possible by vector tile maps. Also a practical function enabled by vector tiles is the dynamical change of languages<sup>36</sup>.

Further technical background on vector tiles can be found in the vector tile specification (VTS), which is an extensive and detailed documentation by Mapbox on how vector tiles are encoded<sup>37</sup>. It provides information about file formats, extensions, structure (geometry and attribute encoding), projections and bounds. The first version was released in April 2014. Currently version 2.1 (released January 2016) is used (Mapbox Vector Tile Specification<sup>38</sup>). Earlier this year (March 2018), developers of Mapbox announced the “Vector Tile 3.0 Specification” in which they plan to include new ideas and break major changes, such as including 3D data and improved tile set metadata documentation, called TileJSON spec (Thompson, 2018). TileJSON provides information about the tile set’s metadata and references the map tile’s location (Github: Mapbox TileJSON spec<sup>39</sup>). Mapbox vector tiles are formatted according to Google Protocol Buffers (PBFs), which possess the characteristics of a language-neutral and platform-neutral extensible mechanism used for serialized structured data (Github: Vector Tile Specification<sup>40</sup>). A vector tile bears the filename .mvt and a tile set is stored in .mbtiles.

As a consequence, vector tile maps are the future of web mapping providing a better user experience and enabling innovative functions that are yet to be standardized and invented (de la Torre, 2018). Therefore, the practical part of this thesis is focused on vector tiles rather than raster tiles.

---

<sup>35</sup> [https://www.google.com/maps/place/Madison+Square+Garden/@40.7503664,-73.9936821,18.92z/data=!3m1!5s0x89c259ae1546fb27:0x93ba42deb43c8368!4m12!1m6!3m5!1s0x89c259ae15b2adcb:0x7955420634fd7eba!2sNew+York+Pennsylvania+Station!8m2!3d40.7505681!4d-](https://www.google.com/maps/place/Madison+Square+Garden/@40.7503664,-73.9936821,18.92z/data=!3m1!5s0x89c259ae1546fb27:0x93ba42deb43c8368!4m12!1m6!3m5!1s0x89c259ae15b2adcb:0x7955420634fd7eba!2sNew+York+Pennsylvania+Station!8m2!3d40.7505681!4d-73.9935187!3m4!1s0x89c25a21fb011c85:0x33df10e49762f8e4!8m2!3d40.7505045!4d-73.9934387)

[73.9935187!3m4!1s0x89c25a21fb011c85:0x33df10e49762f8e4!8m2!3d40.7505045!4d-73.9934387](https://www.google.com/maps/place/Madison+Square+Garden/@40.7503664,-73.9936821,18.92z/data=!3m1!5s0x89c259ae1546fb27:0x93ba42deb43c8368!4m12!1m6!3m5!1s0x89c259ae15b2adcb:0x7955420634fd7eba!2sNew+York+Pennsylvania+Station!8m2!3d40.7505681!4d-73.9935187!3m4!1s0x89c25a21fb011c85:0x33df10e49762f8e4!8m2!3d40.7505045!4d-73.9934387)

<sup>36</sup> MapTiler enables the dynamical change of languages on their vector tile maps (drop down):

<https://www.maptiler.com/maps/#streets//vector/3/16.85/48.7>

<sup>37</sup> <https://github.com/mapbox/vector-tile-spec/tree/master/2.1>

<sup>38</sup> <https://www.mapbox.com/vector-tiles/specification/>

<sup>39</sup> <https://github.com/mapbox/tilejson-spec/tree/master/2.2.0>

<sup>40</sup> <https://github.com/mapbox/vector-tile-spec/tree/master/2.1>

### 2.2.5 Projection

The *Web Mercator* projection used in web maps is a rectangular projection and was also brought by Google Maps (Muehlenhaus, 2014). While the distortion are insignificant on large scales, the landmasses on small scales are way out of proportions, owing to the fact, that the projection “defines the underlying geographic coordinates using WGS84 but projects them as if they were defined on a sphere” (Battersby, et al., 2014), resulting in simpler and faster calculations. Other than the Mercator projection itself, the Web Mercator is technically non-conformal, even though a visual difference cannot be seen, so “for most general purpose mapping the distortions to local angles are minimal” (Battersby, et al., 2014), which is important when displaying aerial imagery to avoid distortions of buildings (Schwartz, 2018). Contrarily, on a global scale, it is rather inefficient for visualizing thematic situations (Kessler, et al., 2017), as the Mercator projections distorts the land mass shapes immensely<sup>41</sup> but is used by the majority of online map providers none the less (Battersby, et al., 2014).

Because of the structure of map tiles (size usually being 256 by 256px), the map needs to be rectangular to be divided evenly into tiles so that a square aspect ratio is used at a maximal latitude of approx. 85.05 degrees (Schwartz, 2018). Another attribute of the projection is, that there are no special restriction (Jansen & Adams, 2010): when zooming out of the OSM map<sup>42</sup> it is noticeable, that the map is repeated in west and east direction. The projection is also called *Google Mercator* (EPSG:900913), *Spherical Mercator* or *Pseudo Mercator* (EPSG:3857)<sup>43</sup>. Figure 2.13 shows the projection in combination with the Tissot’s Indicatrix to emphasize the distortion.

---

<sup>41</sup> An effective visualization of the distortion of the Web Mercator can be explored on <https://thetruesize.com/> where country shapes can be compared simultaneously.

<sup>42</sup> <https://www.openstreetmap.org/#map=0/-1/0>

<sup>43</sup> see <https://epsg.io/3857>

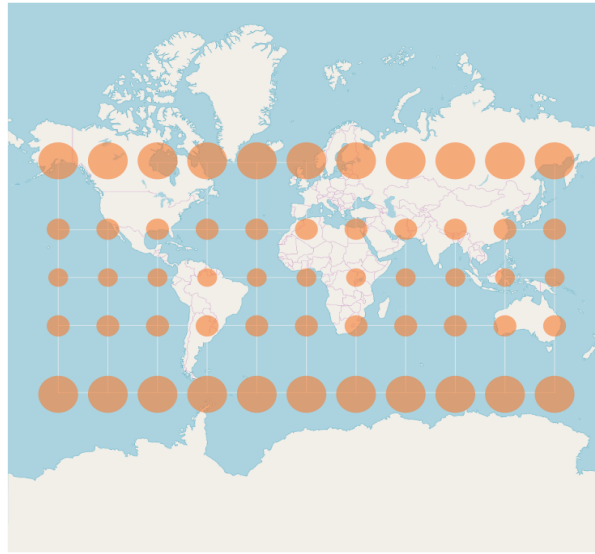


Figure 2.13: Web Mercator projection combined with Tissot's Indicatrix (done in QGIS)

Recently, Google Maps introduced an innovative function called *3D globe mode* which occurs when exploring the map in smaller scales. The map does not remain flat but gets spherical when zooming out until the whole globe is pictured. In doing so, the immense distortions near the poles get cancelled out, e.g. Greenland is no longer pictured as big as Africa (Liptak, 2018), which is a big advantage. An adaption for thematic world maps is questionable as the comparison of two opposite situated regions are not easily accomplished.

#### 2.2.6 Multi-Scale Map Design Approaches

The state of the art regarding techniques to design a multi-scale map expose that they are not quite automated and need a lot of manual work as well as thinking ahead on how to design the map (Samsonov, et al., 2013). Brewer and Buttonfield (2010) studied the workload that constituted by multi-scale map design by partitioning the design process in “modifying display” (focused on styling the features) and “modifying geometry” (focused on generalizing the data) with the result, that both should not be treated individually but rather seen as complimentary as case studies revealed that the overall workload is reduced.

There are few tools developed to assist in the complex task in designing a multi-scale map (Roth, et al., 2011; Samsonov, et al., 2013). One being ScaleMaster, which was already introduced in 2007 (Brewer, et al., 2007) and advanced later on (Roth, et al., 2011). The

ScaleMaster diagram is a “conceptual schematic for organizing, maintaining, and sharing the scale-dependent design specifications of a multiscale mapping project.” (Roth, et al., 2011, p. 29). It was a collaboration to offer a tool used in classrooms and a guide for multi-scale mapping. The diagram is divided into map themes, which are subdivided into feature types. The rows of these feature types are set at anchor levels (decision points) where operators are used to change a property of the feature type, see fig. 2.14.

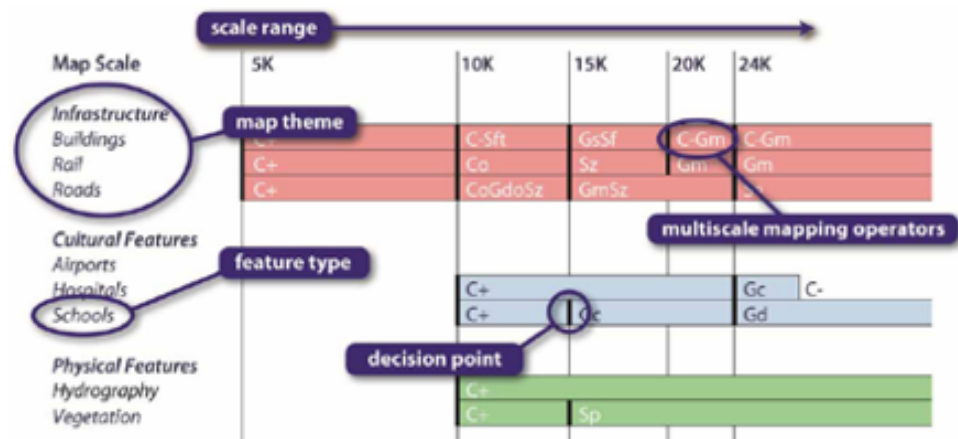


Figure 2.14: Example section of a ScaleMaster diagram (Roth, et al., 2011, p. 33)

With the help of abbreviations, operators are set at the decision points, which is a modification that happens when scales are changed, e.g. adjusting transparency. Algorithms specify the operator's changes, e.g. transparency is increased by 20%. Roth, et al. (2007) also offers a detailed list of the operators and their abbreviation including description and use cases illustrated by example images. The appendix (II) shows a list of the operators which are categorized in content, geometry, symbol, and label, and are more aligning to a cartographer's view rather than a computation-/automation-oriented approach. The content category define decisions made to arrange features on the map, e.g. in-/exclusion of feature types. Geometry operators influence the components of features, that form a feature: points, lines, polygons. The symbol operators form the styling parameters used to symbolize a map feature. To add typographical objects to the map, the label operators are used. The decision-making is not automated, operators are selected by the developer manually (Samsonov, et al., 2013). Most geometry operators can be applied automatically, but the symbolization operators are left to be applied manually by the cartographer.

A ScaleMaster extension (ScaleMaster 2.0) has been proposed by Touya & Girres (2013) and takes on the automatic generalization processes in multi-scale maps. While concentrating on the geometry generalization and by extension content operators, it is limited by different indicators, including the symbol design and multi-themes processes. Samsonov, et al. (2013) on the other hand offered an approach for automatic symbolization from one detailed level to others. They state, that “[s]ymbolization of objects is not always considered as generalization operators” (p. 1), but is necessary to contemplate when geometric and content transformations arise while zooming between scales. The theoretical framework for an automatic symbol translation through multiple LoDs were applied on case studies and concluded in certain drawbacks, which led to believe that the Multimapper-concept is well-thought of but still needs to be investigated further. This is due to the transformation of features that take effect on one layer without consideration of the other layers resulting in disagreements symbolization-wise.

Whereas the previous approaches offered tools and frameworks to create a design concept of multi-scale mapping, they are not merged with guidelines on symbolization techniques and web-dependent design issues. This thesis is not focused on the generalization processes, as they were largely discussed in previous papers and rather aims to provide a bridge between the technical set up and general design decisions.

### 3 Methodology & Implementation

#### 3.1 Reviewing Existing Map Styles

Existing web map styles are widely established throughout the Internet. The review of the styles aims to help gather used techniques in styling multi-scale web maps and show what web map users are already used to and recognize unconsciously. The web map styles that are analysed going forward are Google Maps, Bing Maps and OSM carto.

##### 3.1.1 Map Continuity & Colour Design

The biggest challenge of multi-scale mapping persists in the continuity of a map design which extends through all zoom levels without a provoking break in aesthetics. To grade on how well existing multi-scale map publishers dealt with the difficulty, maps were investigated based on the map complexity and its changing between zoom levels.

Map complexity describes the map content quantity and density (Töpfer, 1974). The numerical map complexity defines the number of objects in the map as “number of objects per  $\text{cm}^2$ ”. The graphical map complexity indicates the area loading in %, whereas the visual map complexity focuses on the drawing quantity and its optical effect on the human eye considering colouring, contrast to the surrounding and optical illusions that influence the visual impact (Ogrissek, 1983). Since the access to the feature data is not given and the map area changes not only by location but also by scale when applying the concept on web maps, only the visual map complexity is taken into account. Because of the unavailable access to the features on the map merely screenshots of the same region (in this case Europe to inner city of Dresden) are used to examine the visual map complexity. The output data - the map snippet’s histogram divided by colour schema (RGB) - is used to show the number of pixel of a certain colour value and examined the drawing amount, as it was traditionally done with colour-separated copy templates of a physical map (Töpfer, 1974). The snippet is then set to greyscale to explore the mean value of the picture. Therefore, information can be distilled on how the features are distributed and how cluttered the map snippet may be. The histogram describes the distribution of values, in this case pixel grey values. Low values describe dark pixel and high values bright ones. The values range from 0 to 255. The mean value of the histogram of each zoom level is calculated

together with the standard deviation. The mean value described the location of a distribution, meaning, the average value of each histogram. The standard deviation is wanted to specify if the distribution is close to the mean or spread out in a wider range. Figure 3.1 illustrates the differences between the map styles of OpenStreetMap (green), Google Maps (blue) and Bing Maps (violet)<sup>44</sup>. It seems exaggerated as the y-axis bounds begin with 200 and end with 240. Nevertheless, it emphasizes the changes between zoom levels depending on the style and provides further details between the changes. If the y-axis was segmented 0 to 255, which are the only possible values, it would not provide sufficient distinguishability.

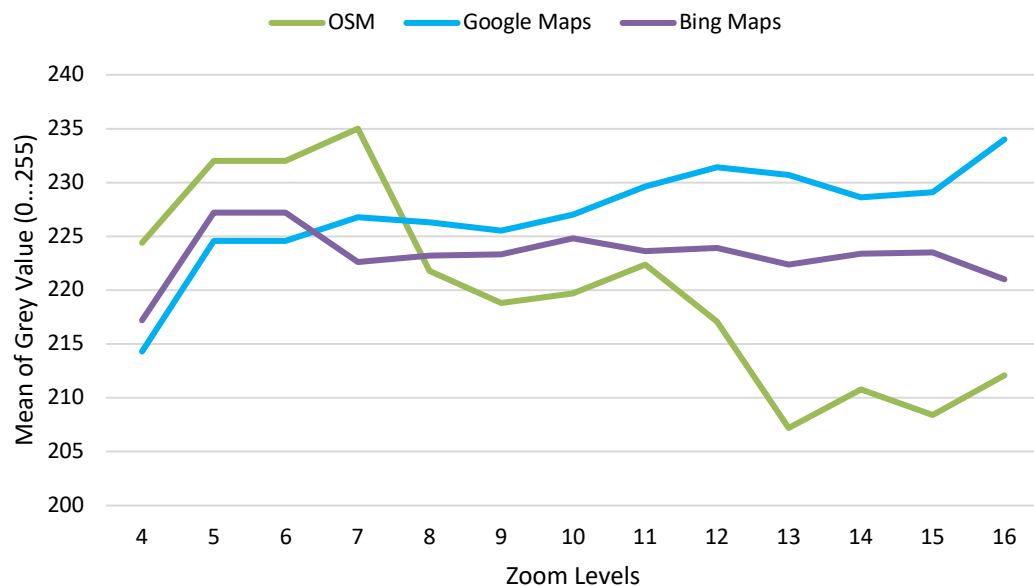


Figure 3.1.: Overview of map complexity throughout zoom levels 4 to 16

<sup>44</sup> The following links give a complete overview for every map snippet from zoom level 4 to 16 with graphs and values (Mathias Gröbe):

OSM: <https://wwwpub.zih.tu-dresden.de/~s6845890/kartenstil/kartenbelastung-osm-carto.html>

Google Maps: <https://wwwpub.zih.tu-dresden.de/~s6845890/kartenstil/kartenbelastung-google-maps.html>

Bing Maps: <https://wwwpub.zih.tu-dresden.de/~s6845890/kartenstil/kartenbelastung-bing-maps.html>



While every line graph jumps from zoom level 4 to 5, which is due to the decrease of water area on the map snippet, Bing maps is able to maintain a steady level of map complexity and (grey) colour distribution throughout the investigated zoom levels. Google maps style seem to lighten up as the zoom level advances, which can also be seen while zooming through the map: smaller zoom levels show lots of vegetational features and the background colour changes from green to beige and finally grey. OSM, on the other hand, shows some significant differences. The first drop from zoom level 7 to 8 illustrates the change from a bright simple map style to a more colourful map filled with many more features and information as seen in figure 3.2. The map complexity increases, more features are displayed, which require different symbolizations and results in a lower mean value. The

same effect happens at the second drop between zoom level 11 and 13. The colours are more saturated and therefore darker. In addition, more text is displayed, which also adds darker pixel counts.

With analysing the map complexity, it is necessary to consider the colour palette, as the composition and number of colours influence the map users experience as well, see chapter 2.1.1. The colours of the following palettes are ordered by: background colour, green features, roads, water, buildings.



Figure 3.2: snippets of OSM carto web map style, top to bottom: z7, z8, z11, z13

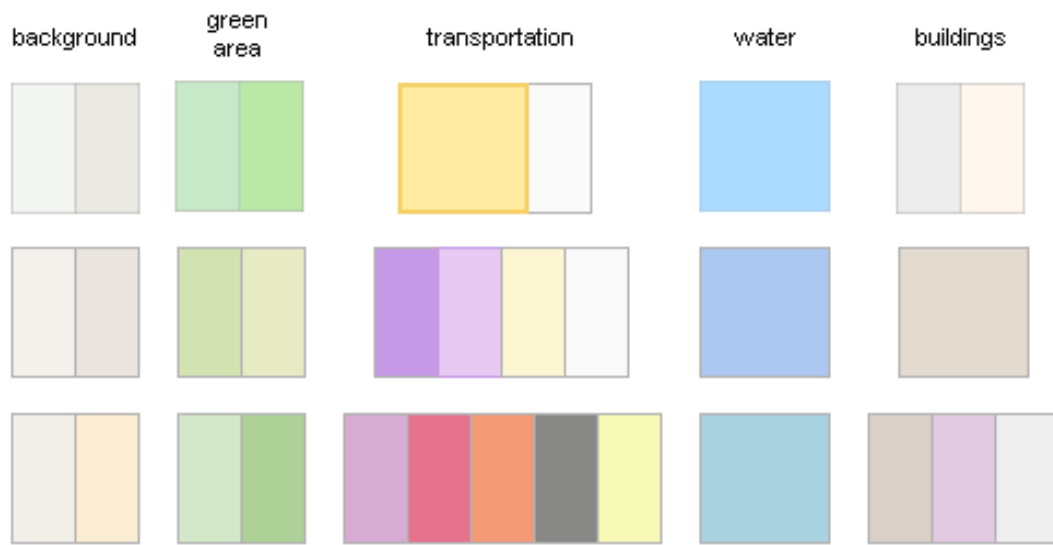


Figure 3.3a (above): Google Maps default colour scheme (2018)

Figure 3.3b (middle): Bing Maps default colour scheme (2018)

Figure 3.3c (below): (German) OpenStreetMap default colour scheme (2018)

Google Maps, as mentioned before, is probably the leading map service when it comes to usage, since it was the first to ever offer such a service (see chapter 1.3.). Throughout its development, the style of Google Maps has shifted many times, but today's colour scheme can be seen in fig. 3.3a. The colours are few with changing saturation throughout the zoom levels, so that the user is not going to be overwhelmed by a lot of features. Bing Maps shows few colours as well, except for the roads: the unusual colours, categorize the roads in higher zoom levels and are not normally associated with this type of features (see fig. 3.3b). They are used probably in order to distance themselves from Google Maps. OpenStreetMap (OSM) on the other hand includes a bigger colour palette, due to presenting many more features in comparison to the other service providers. Fig. 3.3c only displays part of it. When zooming into the map the colours of certain features change many times and on higher zoom levels it gets quite overwhelming.

All in all, it can be said, that a map style with a satisfactory level of continuity would have a graph line keeping the same level or provide a steady de- or increase of the mean value. Peaks and drops illustrate a break in the map style which is undesirable.

### 3.1.2 Typography

When examining the typeface selections it is obvious that only sans-serif ones are used. Since the major 2013 update, Google adopted the home-brand Roboto font for labelling (Graham-Smith, 2013), which is described as a font “with friendly features and open curves. [...] This makes for a more natural reading rhythm [...]” (Robertson, 2018). With the large geometry, different font weights and styles a clear hierarchy is created to distinguish feature labelling. Non-Latin characters, i.e. Cyrillic and Greek letters are supported. For other languages, e.g. Arabic or various Asian character sets, Google uses Noto where Roboto is not supported.

Microsoft created its own typeface called Segoe UI<sup>45</sup>, which is part of the corporal design. Naturally it is used for labelling in Bing Maps (Microsoft- Bing Maps Team, 2012). The Unicode range includes Cyrillic, Greek, Hebrew, Vietnamese among others. Again, font weights, colour distribution and the use of halos create an effective hierarchy and legibility for feature groups. Fall back fonts are Arial and Helvetica.

OpenStreetMap carto style uses Noto font to cover the most Unicode range. DejaVu Sans is used for systems that do not support Noto and other fall back typefaces are Hanazono and Unifont<sup>46</sup>. Font weights are used to distinguish feature types whereas font sizes create hierarchical elements. Occasionally halos enhance the legibility on busy backgrounds.

### 3.1.3 Icons & Symbols

Google Maps symbol design for POIs established a distinctive design with their marker icons. Symbols appear white on a colourful background. The colour depends on the classification of the POI. Transportation seem particularly different to other POIs supporting local symbols such as the London Underground or the S-Bahn in Germany. The same applies to road signs, which are different depending on the location. For example, German Autobahn or US highway road signs resemblance is identical to real life road signs. Google considers cultural differences by these distinctions and supports the national preferences without forcing a universal standard that may not be accepted by a large part of the world.

---

<sup>45</sup> <https://docs.microsoft.com/de-de/typography/font-list/segoe-ui>

<sup>46</sup> [https://wiki.openstreetmap.org/wiki/Standard\\_tile\\_layer#Fonts](https://wiki.openstreetmap.org/wiki/Standard_tile_layer#Fonts)

Bing Maps uses symbols for every point based feature. Capitals are symbolized by red squares, state capitals by red dots and lesser settlements by black dots. The level of detail in design of road numbers is profound. Depending on the location, the road sign conforms to the national standard supporting cultural individuality, even though the map view can get busy with all these road signs. The individuality continues with transportation themed icons, such as train stations, underground stations (e.g. London) and subway stations (e.g. New York City). POIs have a homogeneous design with a symbol classified by colour on a white circle creating a satisfactory composition with the other map elements.

Apart from POIs and road signs are no point features symbolized by icons in OSM carto style. The more than 200 POI icons are colour coded corresponding to their classification, which helps distinct the theme when the map view is cluttered by many POIs. Their design is well-matched and look uniform. The road signs are not individually designed and remain the same regardless their national position.

In summary it can be said, that in case of including POIs with many classification it is useful to colour code them to ensure intuitive distinction by the map viewer. The challenge to provide a simple pictogram for an object that is interpretable by map users from different nations and cultures is ongoing, even though certain symbols are commonly practiced.

#### 3.1.4 Objects and features

Regarding the consideration on what feature groups should be added to draw an effective web map Google Maps', OSM's and Bing Maps' distribution was considered. When creating a web map style the purpose should always be kept in mind. While Google Maps and Bing Maps aim to give an overview of the real world with navigation elements, OSM rather tries to display as much information as possible. Corresponding feature groups that all three web maps illustrate are:

- Water
- Boundaries
- Settlements
- Roads & rails
- Forests

- Buildings
- Certain POIs

When paying attention to further established web map styles, i.e. *CartoDB Positron / Dark Matter* or *Esri World Street Map* and *Esri World Topo Map* display the same feature groups among others so it leads to believe that a basic web map should at least show the listed features to a certain degree.

### 3.2 Visualization Components

While analysing map styles, different visualization techniques were examined to symbolize feature groups. To bring them to order and give an well-structured overview, aspects were drawn in an chart dividing the topic in three categories (fig. 3.4). Geometry holds visualization components regarding size, e.g. of an icon or font, and width, e.g. of line features and polygon or text halos. 3D remarks to the extrusion of features and the rendered shadows while Icon describes the shape of a point symbol. Position holds the rotation and translation properties. Colour attributes were explained in detail in chapter 2.1.1. In the overview opacity and colour change are listed as well, as they influence the colour behaviour massively. Opacity, when applied to multiple features, that can overlay each other, can create colour combinations in certain use cases. The gradual colour change possible by vector tile techniques can produce an impressive effect. Relationship is the most important part of multi-scale map design. The layer order is the fundamental step when including map layers. Most often the order seems to be the same: background, land cover and land use, water, buildings, roads, boundaries and icons/labels. Water should be drawn on top of land cover and land use, as ponds and lakes can be concealed. *Point to Polygon* describes the conversion of point features to polygon features and the level it needs to happen, e.g. when capitals should not be symbolized by an icon anymore but rather the residential land cover polygons. Same goes for *Polygon to 3D*. Sometimes it is rational to first include the polygon layer as such and then imitate an extrusion to not overwhelm the map user with abrupt emerging polygons. Another aspect worth considering is the map view. Vector tile maps allow for tilted maps with 3D visualizations and rendered data enabling a wholly new map view experience for the user.

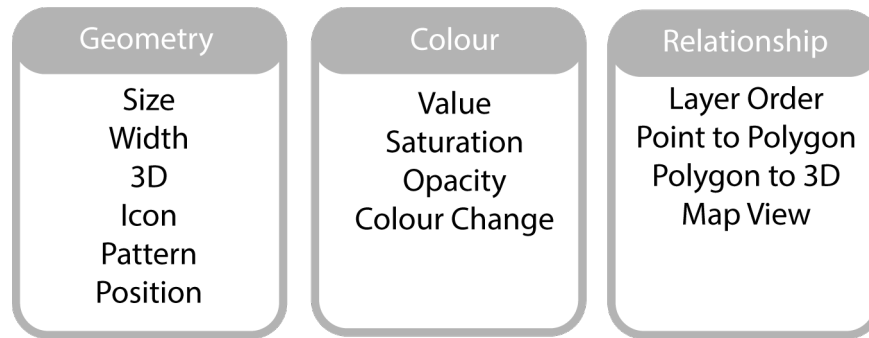


Figure 3.4: Visualization components overview

### 3.3 Workflow

The workflow to design a web map style is proposed as the following figure 3.5. First it needs to be clear what the intention of the web map style would be and therefore preliminary considerations regarding content, medium, and styling components need to be made, following by the selection of available data suitable for the desired content. The styling of the web map and its feature takes up the most time of the process while the implementation of the style in the desired medium format would be the final step.



Figure 3.5: Workflow of Methodology

#### 3.3.1 Editorial Approach

A basic map for thematic maps should be designed to test the workflow. Two versions were considered, both focused on slightly different use cases. While a web map style for geographical and touristic approaches may need elevation data and natural land cover visualized (style will be called *Terrain*), a base map for thematic maps displaying additional data as an overlay, it may only need the basic components to not draw attention away from the data included on the map (style will be called *Grey*). As early as in 1972 Imhof claims that a base map set in monochromatic colours (shades of grey) are outdated (Imhof 1972, p. 47). Reason to use a monochromatic colour scheme is to emphasize the thematic elements of the map and to not distract the user from them. It is true, that the background features should not use strong colours, but that does not necessarily restrict the developer

from using colours entirely, because colours give depth to the map. *Grey* is also practiced to examine the implication on only using a single data source.

Important considerations must be taken before working on a map, as they can save time and work in advance and are essential to organizing. The following list highlight the typical components of a map editorial plan.

1. Purpose

What is the map made for, what context is the map created for, is it going to be a stand-alone map, a base map for visualization, an artistic map, etc. ?

*providing a solid base map for various thematic mapping techniques giving geographical context without intruding or distracting from data visualization*

2. Targeted users or audience

Who is going to use the web map, where will it be published?

*developers, who want to display geo-references data on web maps*

3. Presentation area

It is a multi-scale map so different extends from world to continent to urban view are available; Is there a certain focus of a world's part?

*It is a multi-scale map, therefore different extends from world to continent to urban view are available; no special focus on any part of the map*

4. Content

On multi-scale maps the content is LoD dependent. Certain features are added or withdrawn depending on the zoom levels. What features are necessary and serve the purpose best?

*For further detail see sub chapter (LoDs are not equivalent to zoom levels)*

*LoD1: landmass, water, country boundaries*

*LoD2: waterways, capitals*

*LoD3: highways, subtle state boundaries*

*LoD4: landcover, lakes, big cities, primary roads*

*LoD5: secondary roads, smaller cities, rail*

*LoD6: tertiary roads, towns*

*LoD7: remaining roads, tram rails, villages, suburbs*

*LoD8: building polygons, POIs?*

5. Data Sources

What data sources are affordable or open-source, what sources include the necessary features?

*Terrain:*

- OpenMapTiles: water, boundaries, country labels, roads, rail, capitals, cities, towns, buildings
- Mapbox Terrain: landcover
- Mapbox TerrainRGB: hillshade

*Grey: OpenMapTiles for all layers*

## 6. Colours

Which colours and colour ranges are serving the purpose? Which colours should be avoided?

*web-safe colours with a lot of transparency, no garish colours distracting the map user*

Back-ground	Light grey		#EEEEEE
Water	blue		#99CCFF, #6699CC, #336699
Boundaries	Dark grey		#999999
Land cover	From yellow to green with high transparency		#CCCC66, #99CC99, #669966, #336633; Ice: #3366CC
Roads	White, later dark grey out-line		#FFFFFF, #999999
Rail	Dark grey later with white dash		#CCCCCC, #999999, #FFFFFF
Buildings	Grey with high transparency		#CCCCCC

Table 3.1: Colour palette for web map design

## 7. Fonts &amp; Language

What font family is used for which labeling? Certain languages demand special characters which restricts the use of some fonts.

*Metropolis, Arial Unicode MS as fall back; English, default glyph sheet*

## 8. Icons

Are icons and symbols to be included? If so, what sources are used or are they custom designed?

*for cities, using custom icons and spritesheet*

## 9. Technology

There are different tools available for designing web maps, which one fits the desired criteria?

*See chapter 3.3.3*

## 10. Publication

is it going to be a file or source code for further distribution or a whole web map compiled by a client

*JSON file for further distribution, implemented in MapboxGL JS for demonstration*



### 3.3.1.1 Content

Due to the nature of multi- scale maps, in which features appear and disappear on certain LoDs, complex web maps with a large number of layers or function groups, the overview can easily be lost, so that a clear content structure is useful. To deal with the issue work sheets are proposed as an aid to not lose focus of the content. OpenMapTiles layers serve as an example of demonstration. Appendix III previews the remaining work sheets.

The header section gives information about the layer regarding feature type, visibility of zoom levels, and sources.

Classes and subclasses are listed as a check list either in the header section or as a separate sheet if there are too many options. Second part of the work sheet is a table of available zoom levels as rows and columns depicting sources and fields to fill out about adding and removing features as well as making notes for designing aspects, see fig. 3.6. These sheets aim to aid the concept of web map design and proved to be a great help.

**Worksheet Features Zoom-Levels**

**Layer:** boundary\_line, **Point**  
 Boundary regarding administrative borders, such as countries, states, counties, etc.  
 Sources: OSM, etc.  
 Filtered by admin\_level, values 2,4  
 Filtered by admin\_level, values 2,4

**Feature Type:** boundary\_line  
**Feature Name:** boundary\_line  
**Feature Class:** boundary\_line  
**Feature Subclass:** boundary\_line

Z	L	S	Add	Remove	Styling
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

**Feature List:**

Feature Type	Feature Name	Feature Class	Feature Subclass
Country	country	country	
State	state	state	
County	county	county	
City	city	city	
Village	village	village	
Hamlet	hamlet	hamlet	
Neighborhood	neighborhood	neighborhood	
Island	island	island	
Island Labeling	island_labeling	island_labeling	

Figure 3.6: Example of work sheet aiding web map content assignment (see appendix III)

### 3.3.2 Available Hosted Data and Tile Sources

The following chapters provide an overview of existing tile data that can be used as data sources, all of which are based on OSM and free of charge for development purposes. The more technical approach with self- hosted vector tiles is only mentioned as the thesis focuses on the design process.

When a large number of layers are included, potential filter fields and value can become overwhelming. A sunburst diagram arrangement has been developed for aid, separating polygon from line and point layers (fig. 3.7, 3.8). The most inner ring holds the layer name, fields are listed in the middle ring and possible values are shown on the outer ring. Detailed versions are available in the appendix (IV). The available TileJSON URL is added. If an access key is required it is indicated by [personalKey].

### 3.3.2.1 OpenMapTiles / MapTiler Cloud

OpenMapTiles' vector tile data (*OpenMapTile v3*) is hosted on MapTiler Cloud. Currently version 3 is used. In order to use the tiles a registration is necessary on the MapTiler Cloud website. The user is assigned a key with which the tile sets can be accessed. Layers include among others building, landcover, transportation, POIs and water. The OTM website offers a detailed documentation of layers<sup>47</sup>, explaining sources and fields/values, as well as a tool to inspect the features beforehand<sup>48</sup>.

TileJSON: [https://maps.tilehosting.com/data/v3.json?key=\[personalKey\]](https://maps.tilehosting.com/data/v3.json?key=[personalKey])

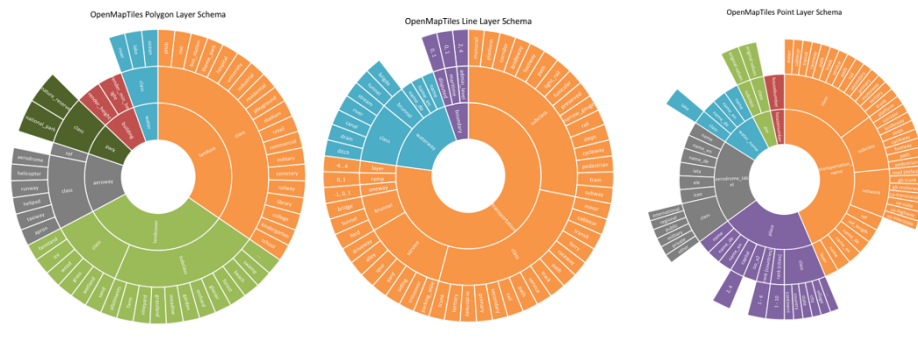


Figure 3.7: OMT polygon, line and point layer schema visualized as sunburst diagrams; detailed version found in appendix IV

MapTiler's vector tile set *Contours* is available from z9 to z14 and includes MultiLineStrings with the properties of height and nth\_line for individual styling of the lines.

TileJSON: [https://maps.tilehosting.com/data/contours.json?key=\[personalKey\]](https://maps.tilehosting.com/data/contours.json?key=[personalKey])

*Landcover* consists of a polygon layer which provides global landcover from z0 to z9. The polygons are divided into classes with possible values being: crop, grass, scrub, forest and tree.

TileJSON: [https://maps.tilehosting.com/data/landcover.json?key=\[personalKey\]](https://maps.tilehosting.com/data/landcover.json?key=[personalKey])

<sup>47</sup> <https://openmaptiles.org/schema/>

<sup>48</sup> <https://openmaptiles.org/inspect/#0.47/-0/0>

Raster tile sets (*Hillshades*, *RGB Terrain*, *Satellite*, *Satellite Mediumres*) as well as a 3D Model of the world's terrain (*Terrain – Quantized Mesh*) are also available on MapTiler Cloud.

#### TileJSON URLs

*Hillshade*: [https://maps.tilehosting.com/data/hillshades.json?key=\[personalKey\]](https://maps.tilehosting.com/data/hillshades.json?key=[personalKey])  
*RGB Terrain*: [https://maps.tilehosting.com/data/terrain-rgb.json?key=\[personalKey\]](https://maps.tilehosting.com/data/terrain-rgb.json?key=[personalKey])  
*Satellite*: [https://maps.tilehosting.com/data/satellite.json?key=\[personalKey\]](https://maps.tilehosting.com/data/satellite.json?key=[personalKey])  
*Satellite Mediumres*: [https://maps.tilehosting.com/data/satellite-medium-res.json?key=\[personalKey\]](https://maps.tilehosting.com/data/satellite-medium-res.json?key=[personalKey])  
*Terrain – Quantized Mesh*: [https://maps.tilehosting.com/data/terrain-quantized-mesh.json?key=\[personalKey\]](https://maps.tilehosting.com/data/terrain-quantized-mesh.json?key=[personalKey])

### 3.3.2.2 Mapbox

Mapbox's tile sets are available through a source ID. The URL, that is pasted as a TileJSON URL, consists of `mapbox://[mapid]`. An access key is needed to use the data sources, which can be requested upon registration on Mapbox<sup>49</sup>. A detailed documentation explains layer references and schema<sup>50</sup>.

*Mapbox Streets v8* - currently version 5 to 8 are available – provides the basic vector tile set including many layers, similar to *OpenMapTiles v3*. Some layers include mixed feature types, which is why some appear in more than one diagram, e.g. *aeroway* being line and polygon features.

TileJSON: `mapbox://mapbox.mapbox-streets-v8`

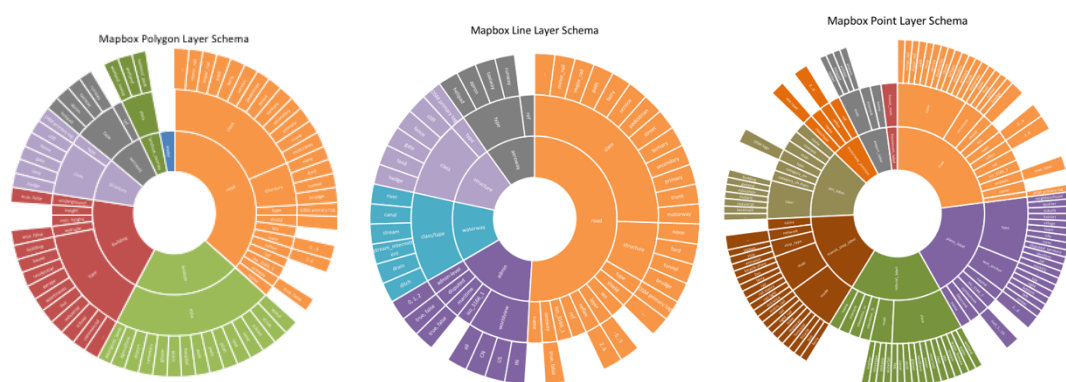


Figure 3.8: Mapbox polygon, line and point layer schema visualized as sunburst diagrams; detailed version found in appendix IV

<sup>49</sup> <https://www.mapbox.com/signup/>

<sup>50</sup> <https://www.mapbox.com/vector-tiles/>

*Mapbox Terrain* (version 2) vector tile set contains hillshades, elevation contours as well as landcover data, all of which are polygon features.

TileJSON: `mapbox://mapbox.mapbox-terrain-v2`



Figure 3.9: Mapbox Terrain Layer Schema

*Mapbox Traffic* vector data is based on *Mapbox Streets* road geometry and provides constantly updated congestion and traffic data. It consists of a single line feature layer (*traffic*).

TileJSON: `mapbox://mapbox.mapbox-traffic-v1`

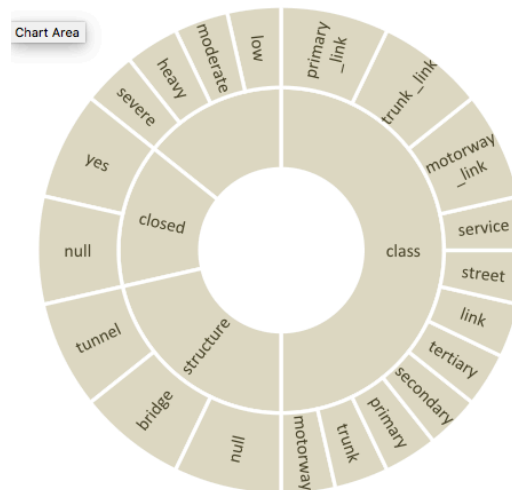


Figure 3.10: Mapbox Traffic Layer fields

*Mapbox Terrain (RGB coded RasterDEM)* and *Mapbox Satellite* are raster tile layers offered by Mapbox which can be included as well.

#### TileJSON URLs

*Mapbox Terrain (RGB)*: `mapbox://mapbox.terrain-rgb`

*Mapbox Satellite*: `mapbox://mapbox.satellite`

### 3.3.2.3 Thunderforest

Thunderforest's vector tile data seem to be highly detailed and require much computing power on the user's side when included as a source. OpenCycleMap<sup>51</sup>, for example, uses Thunderforest data. Two different tile sets are available: *Outdoors* and *Transport*. The documentation is not as consistent as the previous providers, there is only a list of implemented layers and a short description for each tile layer<sup>52</sup>.

#### TileJSON URLs

*Outdoors*: `https://tile.thunderforest.com/thunderforest.outdoors-v1.json?apikey=[personalKey]`

*Transportation*: `https://tile.thunderforest.com/thunderforest.transport-v1.json?apikey=[personalKey]`

<sup>51</sup> <https://www.opencyclemap.org/>

<sup>52</sup> <https://www.thunderforest.com/docs/vector-maps-api/>

#### 3.3.2.4 Other & Custom

Mapzen was a mapping platform, similar to Mapbox, offering free tools and services. It shut down services in January 2018<sup>53</sup>. Their products remain open-source on Github and long-term support for Mapzen maps, vector and terrain tiles is given in form of .mvt files by Nextzen that can be incorporated with TangramJS, a JavaScript library to render web maps and originally a service of Mapzen, see next chapter.

There are various other sources that offer vector tile sets, but they are commercial. Mapcat<sup>54</sup> offers among other services raster and vector tiles with regularly updated data, customization and supporting native languages.

Custom data can be hosted with TileservGL<sup>55</sup>, an open-source map server with the ability to render vector data into raster tiles for web and mobile applications featuring GL styles. These can be used locally (offline) or on a personal web server. Tippecanoe<sup>56</sup> is an open-source command-line utility to generate vector tile sets (MBtiles) from GeoJSON feature collections. Mapbox developers offer the tool for the possibility to convert big data to smaller files suitable for an upload to Mapbox studio. Tiler<sup>57</sup>, called a “no nonsense Vector Tile pipeline”, turns vector data, such as Shapefiles or GeoJSON, into vector tiles (Milner, 2017). With its command-line utility it is similar to the previous tools.

#### 3.3.3 Applications for the Process of Multi-Scale Map Design

To investigate what is possible in multi-scale map design and where the limits are, a suitable tool has to be chosen. There are several tools available to create a multi-scale map style differing in customizations and functions. The investigated tools are solely online platforms that do not require an installation on the computer system. They are generating JSON files or saving the style on the account server that describe the styling of objects shown on the map and are using mainly vector map data.

---

<sup>53</sup> <https://mapzen.com/blog/shutdown/>

<sup>54</sup> <https://www.mapcat.com/planpricing-2/>

<sup>55</sup> <https://tileserv.readthedocs.io/en/latest/index.html>

<sup>56</sup> <https://github.com/mapbox/tippecanoe>

<sup>57</sup> <https://github.com/geovation/tiler>

### 3.3.3.1 Google Maps APIs Styling Wizard

Google Maps APIs Styling Wizard<sup>58</sup> is a platform to generate JSON files that describe the styling of objects. The accessed data is available by Google Maps data in vector format, meaning changes are instantly rendered and displayed. At the starting situation it is possible to choose from uploading an existing JSON file or between established base map styles: Standard, Silver, Retro, Dark, Night and Aubergine. Their object density can be adjusted via slider regarding their feature types roads, landmarks and labels. This is automatically done and does not offer any high customization. Advanced settings on the other hand lets the developer decide further geometry style properties, including colour, weight, saturation and lightness. Apart from the colour property, all *Stylers* (as called on the platform) are set by a slider, so that fine settings are not available and slider points are pre-set by the application. Object's visibility can be set to hidden, shown, inherited or simplified, the latter also only on pre-set conditions. Apart from the geometry settings, label properties of each object are also adjustable in advanced settings with the same conditions. The features are ordered alphabetically in categories. The output of the style is either a JSON code which can be included by using *Google Maps JavaScript API* or a Google Maps URL for the *Google Maps Static API* with all changed objects, which can potentially lead to a very long URL to be implemented.

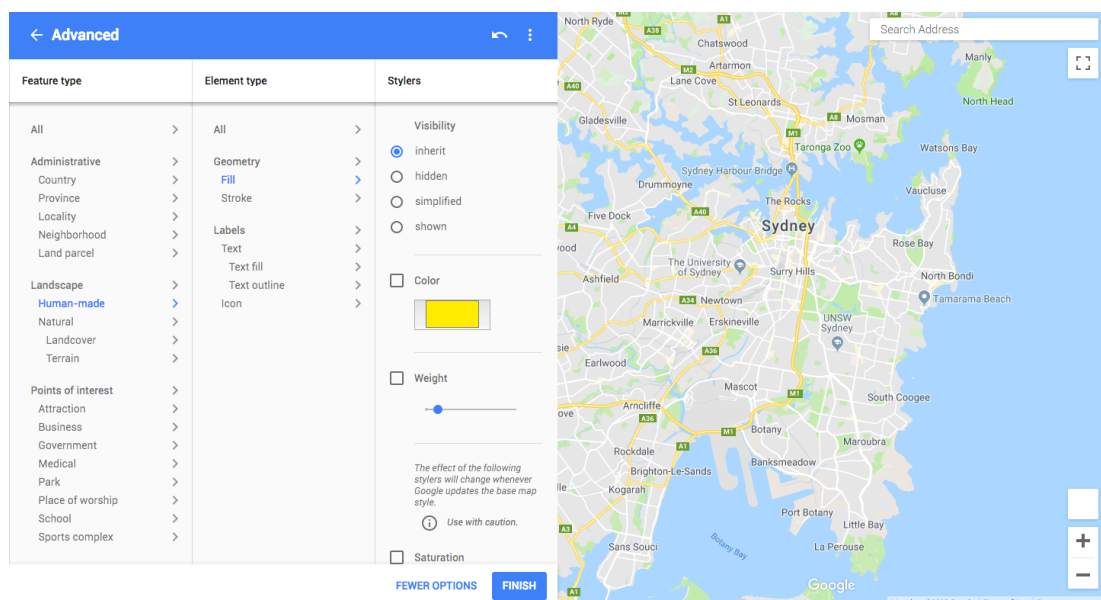


Figure 3.11: GUI of Google Maps APIs Styling Wizard

<sup>58</sup> <https://mapstyle.withgoogle.com/>

While Google Maps APIs Styling Wizard is a suitable tool for beginners to adapt a Google Map as desired, it has a lot of limits and does not offer any customization in relation to multi-scale mapping, as they are no parameters to set anchor levels and adjust properties according to them. Important to note: it is not possible to change the font family, icons and layer content of the map, which leads to some major restriction in customizing the base map. Also the initial induction is not helped by guides or tips, which is probably due to the few function the platform offers, but it is still missing an introduction to get started.

Snazzy Maps<sup>59</sup> is an online community where user are able to offer their finished Google Maps design to other users in form of JSON. The on-site customizing tool offers further adjustment and few more styling parameters, such as Gamma value and inverting of colours, but offers basically the same options as Google Maps APIs Styling Wizard.

### 3.3.3.2 Maputnik

Maputnik is an open-source and free visual editor using the Mapbox GL style specification developed by Lukas Martinelli. It is possible to use the online editor or install the editor as an application on the computer and use it locally. Even though the online editor is accessible on OpenMapTiles<sup>60</sup>, the Github<sup>61</sup> version is more advanced and maintained periodically. Being an open-source tool, an elaborate documentation offers thorough explanations to functions and shows sample applications. Figure 3.12 shows the graphical user interface of the style editor. To the far left the layers are listed while the next column enables styling options for the selected layer. The header navigation holds buttons for data sources, upload/export, settings, mode switch and help.

---

<sup>59</sup> <https://snazzymaps.com/>

<sup>60</sup>

<sup>61</sup> <https://maputnik.github.io/editor/>



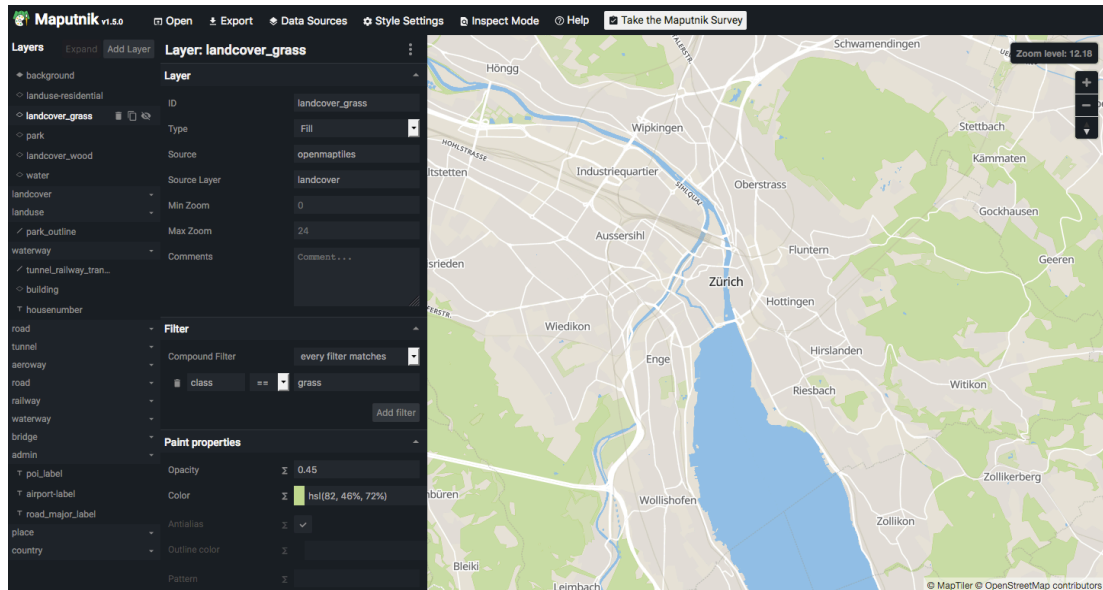


Figure 3.12: GUI of Maputnik (Github version)

Sources, that can be used by default, include Mapbox Street vector tiles and OpenMap-Tiles vector tiles, which are based on OSM. Recently added default sources are Thunderforest Transport and Outdoors vector tiles which are both highly detailed and therefore demand more processing power. Additional data sources are addable in form of vector/raster TileJSON or XYZ URLs, raster DEMs and GeoJSON data, see previous chapter. Therefore, a mix of both vector tile set and raster tiles set is possible, e.g. vector polygons and lines combined with hillshade raster. Within the style settings the URLs for the glyphs and sprites. Glyphs are used for custom fonts, sprites are single images containing various icons as PNG coupled with an corresponding JSON file (Mapbox Help Glossary, 2018). Both files need to have the same name and to be hosted on a secure server (HTTPS). The JSON file constructed like the following code snippet, using width, height and pixel ratio to describe the extent of the icons and x and y as the upper left corner pixel coordinates of the icon on the PNG file. The spritesheet is included with the url to the folder and file name excluding the file extents:

`https://example-domain.com/path/to/folder/sprite.`

```
{  "icon-name": {
    "height": 10,
    "width": 10,
    "pixelRatio": 1
    "x": 0,
    "y": 0
  },
  "icon2-name": {
    [...]
  }
}
```

Similar to Google Maps Styling Wizard it is possible to either start with an upload of a JSON style file, choose from existing styles or start from scratch. The added layers can be arranged as wanted to overlay features as desired. The higher the position in the list, the lower it appears in the layer hierarchy on the map. In order to add layers, the developer needs to know the layer reference which is often listed in documentations and schemas, see next chapter. The particular layer is highly customizable depending on the type. The source, minimal and maximal zoom as well as filter functions are always given to be edited.

*Background*, as the name suggests, describes the background of the map and usually contains the styling of the land masses because the ocean layer frames them. Only colour, opacity and a pattern can be altered for this layer type.

*Fill* layers are used for polygons of all kinds. Additionally to the previous settings, fill outline colour and translation adjustments can be made.

*Lines* are styled with colour, opacity and width. A pattern can either be applied by a symbol, similar to the previous types, or a dash array pattern. The line style is additionally characterized by blurs, offset and gap. The latter creates a line casing leaving the inside blank/transparent. Cap property (butt, round, square), join (bevel, round, miter) and miter as well as round limit are numbered among layout properties.

*Symbol* type layers are used for point based information. For instance, point data (e.g. points of interest) or specific placements on lines, such as labels for street names or river names, are use cases. The point can either be styled with an icon, some text (if the layer has corresponding properties) or a combination of both, e.g. a hospital showing a cross symbol plus hospital name. Because of the complexity of label and symbol placement a lot of customization features are given. Regarding text appearance not only colour opacity

are available, size, font and halo are adjustable as well. Label placement is set through anchor (e.g. left), offsets and rotations. Same applies to icon placement which are based on objects defined in a sprite sheet. There are many more settings that can be explored, the mentioned ones are the rather obvious ones to use.

*Raster* type layers are also supported by Maputnik and require a raster layer. As explained in previous chapters, raster layers are pre-rendered tiles, so that the adjustment are limited by image processing components such as opacity, brightness, saturation and contrast. This type of layer is mostly used for satellite imagery.

*Circle* is similar to symbol type a format that is used for point based data visualization in form of circle symbols variable in colour, opacity, stroke width an opacity, blur, radius, translation and pitch. Therefore, it is possible to symbolize points without the need of sprite sheets.

*Fill extrusion* are used for polygons that contain a numerical property. Most common use is the extrusion of buildings based on their height. Apart from the usual symbol settings, extrusion can be coloured or patterned. The height is then either set at an universal value or a field attached to the layer.

*Hillshade* type layers are based on digital elevation models (DEM) which are rendered in raster. Illumination, exaggeration as well as colours for shadow, highlight and accent are valuables that are adjustable.

*Heatmap* layers underlie scattered points as GeoJSON data and visualize them typically in temperature colours depending on their clustering degree. Maputnik offers only radius, intensity, opacity and weight to edit the layer. The adjustment of colours is still in development<sup>62</sup>.

Nearly every adjustment component is coupled with the possibility to edit it zoom dependent. If enabled, at least two stops need to be set and a gradual change from one stop to another is applied on the map. To give an example: the fill layer of ocean is at z2 a bright blue and at z6 a darker blue. Between z2 and z6 the bright blue gets darker with

---

<sup>62</sup> <https://github.com/maputnik/editor/issues/265#issuecomment-366511333>

increasing zoom until the wanted dark blue tint is reached on level z6. Same goes for opacity, icon size, font size, etc. As many stops as zoom levels can be added.

Apart from the map real time rendering (map mode), it is possible to switch to the inspect mode, which shows all features added to the map. Selected and filtered objects are highlighted. With the help of mouseover, additional information are offered such as classes, names in different languages and other important information linked to the layer / object. Unfortunately the popup is not scrollable so that especially place features cannot fully surveyed, because too many fields are listed (see fig. 3.13). Another convenient function is offered when clicked on the map: a popup window is opened with the layers that include features on the clicked location, which can then be selected over the popup. In case of a web style with a lot of layers, this can be crucial to work efficiently.

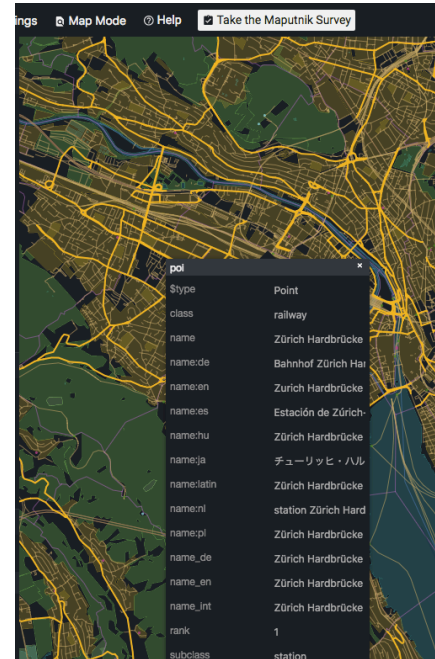


Figure 3.13: Inspect mode on Zurich main station

Because there is no registration needed to use the style editor, it is not possible to save the style on an online platform, it is only exportable. The current state, however, is saved in cache of the browser, so that even after closing the browser it is still available. To use the style, an API key for the various sources is necessary, meaning if for example Mapbox layers are used, a public Mapbox access key is needed in order to be able to use and even display the tiles for customization. OpenMapTiles tiles can be accessed via MapTiler Cloud access keys which is the tile hosting provider for OpenMapTiles data. A sample access key is given to work on a style in Maputnik but asked for as soon as an export of the style is requested. In the settings of the style it is also possible to choose between MapboxGL JS and Open Layers 3, which are JavaScript libraries for creating web map applications (see chapter 3.3.4). These options are set to choose between renderer the map style is going to be compiled. The exported JSON file can then be included on web applications with the help of said display engines. The style

JSON file underlies the *Mapbox Style Specification*<sup>63</sup> defining the attributes connected to the drawing of a map, i.e. data sources, drawing order and visual appearance of the data.

### 3.3.3.3 Mapbox Studio

Mapbox Studio<sup>64</sup> is another style editor for vector data suitable for web maps in browsers or applications on mobile devices. Naturally, the style JSON conforms to the Mapbox Style Specification. Presumably, Maputnik and Mapbox Studio are somehow linked, as Lukas Martinelli is part of the cartography team of Mapbox<sup>65</sup>. The majority of functionalities offered by Mapbox Studio are highly similar to those of Maputnik. In order to use it a registration on Mapbox is necessary. A detailed documentation and explanation of the style editor is established<sup>66</sup>.

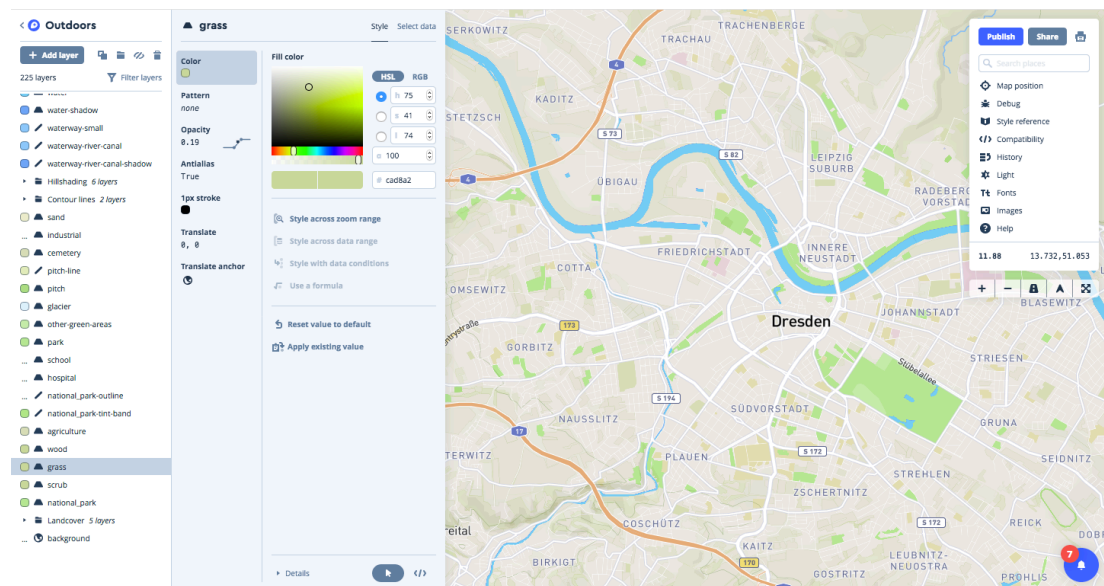


Figure 3.14: GUI of Mapbox Studio

As can be seen in figure 3.14 the interface structure is similar to the other editors with layers being situated on the left and styling options opening up next to them as soon as a layer is selected. A toolbar is included on the right side of the interface. Unlike Maputnik, the layer order is reversed which means that the lower the layer is listed the lower the features are shown on the map starting with the background layer at the bottom and labels

<sup>63</sup> <https://www.mapbox.com/mapbox-gl-js/style-spec>

<sup>64</sup> <https://www.mapbox.com/studio/>

<sup>65</sup> <https://www.mapbox.com/about/team/lukas-martinelli/>

<sup>66</sup> <https://www.mapbox.com/studio-manual/reference/styles/>

at the top. When adding a new layer, Mapbox' data sources are listed and offer all available feature layers. A mix of vector and raster tile set is also possible here. Custom data can be uploaded in form of MBTiles, KML, GPX, GeoJSON, Shapefiles and CSV, but hosted vector tiles by other providers are not supported. The new layer are then defined by type (fill, line, symbol, background) and can be filtered by available field values. The drop down list of available filter fields and possible values is an enormous help for the developer, so that a prior study of the documentation is not required and can happen intuitively. At the same time, the map view changes from the rendered map to the mode where all features are included (see fig. 3.15). Even text labels are displayed so that a preview of labeling can already be viewed. The view can also be requested on existing layers by switching from *Style* to *Select Data* on the layer pane.

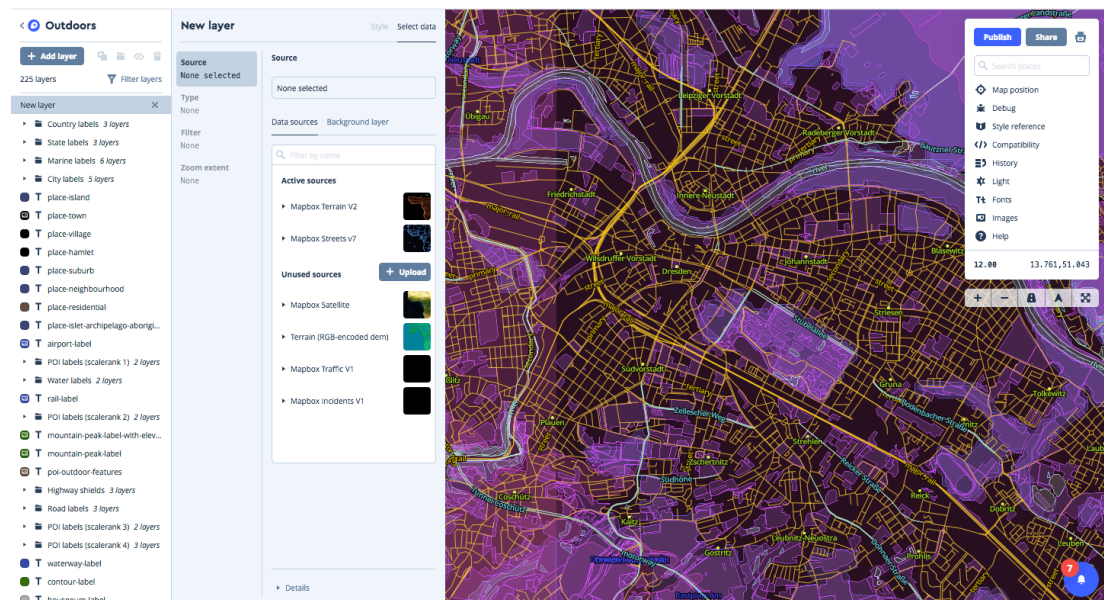


Figure 3.15: Map view change when adding a new layer

The majority of features and settings set out in Maputnik can be found in Mapbox Studios, such as zoom- dependent styling and in- depth customization. The toolbar on the right side offers additional customization, such as upload of SVG images and icons or even fonts. Mapbox lets the user include own fonts from the start (Maps Mania, 2018), allowing the creation of distinctive web map styles. Debug and Compatibility checks are useful for more skilled developers with an advanced agenda. The finished style is then saved and published on Mapbox, enabling the possibility to download a style.json file with all used icons and fonts or use a Mapbox URL to include the style on web pages.



The user experience on Mapbox Studio is generally pleasant, as the user can intuitively explore the many features or find help in the detailed documentation if necessary. The tidy yet expansive GUI relieves the user from the vast number of functions by offering a clean and innovative design with written out labels and hints when hovering. Convoluted menus and panes as well as useful previews of settings create a clear overview that contributes to the usability, e.g. when zooming the map zoom dependent values change simultaneously as well as a slider animation showing when changes are to be expected (see fig. 3.16).

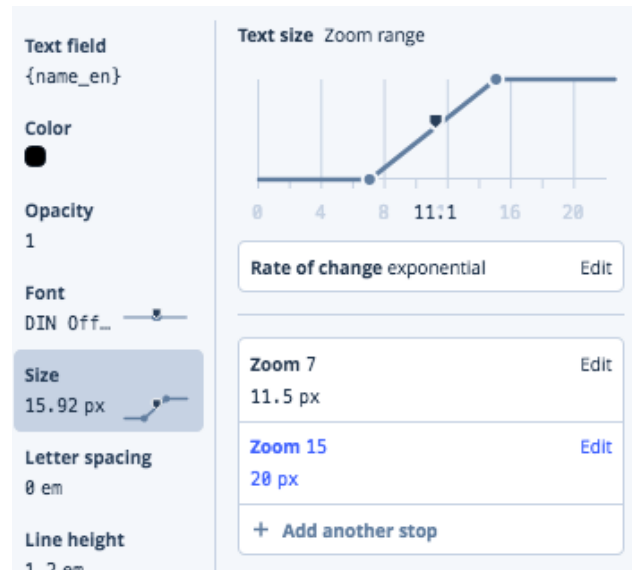


Figure 3.16: Value Size with stops at z7 (11.5px) and z15 (20px); currently at z11.1 having a size of 15.92px.

### 3.3.3.4 MapTiler Customize Tool

The MapTiler Costumize tool style editor lets the developer edit an existing style in terms of colours, font and language. With a click on the help button on the lower right side a short tutorial is given explaining the interface and functions. Further explanation can be found in the “How-to” section<sup>67</sup>. Apart from the comparably few adjustments that can be made, on the right upper corner “Advanced Editing” can be selected which then forwards to a map style editor which is highly identical to Maputnik, which seems natural as MapTiler and OpenMapTiles are linked providers (see fig. 3.17b). Yet a distinguishable difference is the selection of sources: even though MapTiler offers a variety of datasets – contours, hillshade, landcover, satellite (mediumres), terrain (RGB and quantized mesh), OpenMapTiles – no other providers are possible to include. Custom data can be added if an upgraded account is available. Exactly like Mapbox, with MapTiler the user is able to save the style on MapTiler Cloud. The style is not exportable – solely the map is published and hosted in the cloud, so that the style can be used with Mapbox GL JS, OpenLayers

<sup>67</sup> <https://www.maptiler.com/how-to/completely-change-the-map-design/>

and Leaflet GL. Accordingly, no self-hosting of the JSON file is necessary. Theoretically it is possible to host the vector tile style created in Maputnik on MapTiler Cloud to avoid self-hosting it, but has issues when uploading styles with certain sources and custom sprites.

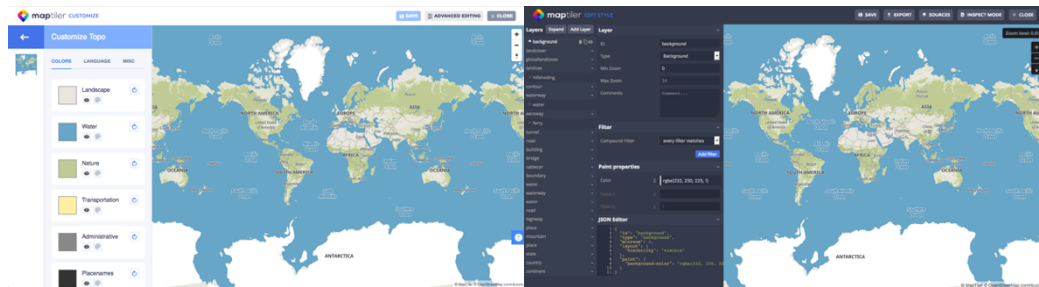


Figure 3.17a (left): Maptiler CUSTOMIZE GUI

Figure 3.17b (right): MapTiler Edit Style (advanced editing) GUI

### 3.3.3.5 Esri ArcGIS Vector Tile Style Editor (beta)

The Esri ArcGISOnline service<sup>68</sup> introduced an editor for vector tiles in May this year (Foster, 2018) and is still in beta version as of now, which means there are changes and improvements based on user feedback going to made. To use the tool a registration at ArcGISOnline is required. With a lot of functionalities the UI stands out from the other tools that were introduced. After selecting an existing Esri style the overview is divided into four parts, see fig. 3.18. The map views shows how the style appears on different zoom levels, defaults being z2, z6, z10 and z15. Every map view includes the zoom utility buttons, the current zoom number display and a search function to look for a place or address. The mini maps can also be hidden. When hovering with the mouse over the map, on the bottom of the page an information is given what layer is selected by the mouse. With a click, the layer can be selected and directly changed in the layer edit menu.

<sup>68</sup> <https://developers.arcgis.com/vector-tile-style-editor/>



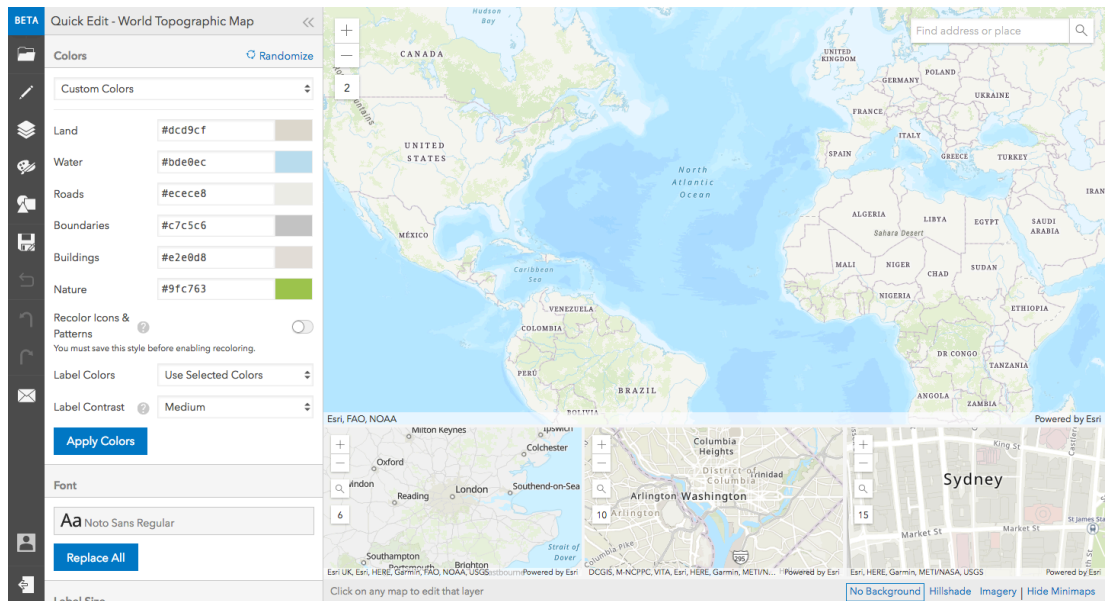


Figure 3.18: User Interface of Esri ArcGIS Vector Tile Editor (beta)

By default, the *Quick Edit* toolbar is opened (fig. 3.18), which allows instantaneous adjustments regarding the general colour scheme of layer groups, label sizes and fonts as well as road widths. If more in-depth customization is wanted, the sidebar on the left side offers different edit modes. *Edit Layer Styles* is a list of all existing layers with extensive property settings. It is sortable by category with expandable sub-categories or by drawing order giving a preview of the colour and indicating the zoom level extend with vertical bars, see fig. 3.19. Layers are distinguished by type (area, line, point) and offer therefore different setting options. *Visibility* adjusts at which zoom levels the feature group is displayed. *Position* can translate an object group and *Appearance* adjusts colour properties, pattern or opacity. Everything can be adjusted depending on zoom levels except for patterns. Additional to these settings, line type layers are also adjustable in terms of width, blurring and dash patterns. Point type layers on the other hand are arranged by *Visibility*, *Icon* and *Text Appearance* as well as *Symbol Position* and *Text Layout*. A simpler edit mode is *Edit by colour*, which lists all colours that are specified and are changeable easily. After clicking a colour, an overview of all layers using the colour is given. That way, many layers can be changed at once. Custom icons and patterns can be easily uploaded and existing ones adjusted, see fig. 3.19.

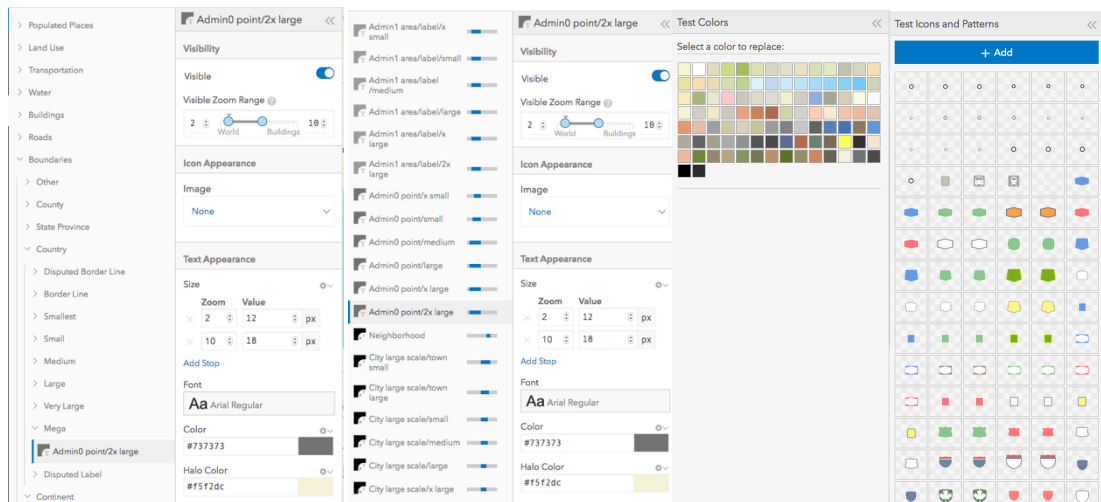


Figure 3.19: Edit modes of Esri ArcGIS Vector Tile Editor (beta); Edit Layer Styles (Category & Drawing Order), Edit Colours, Edit Icon and Patterns

The tool is compared to the other ones neatly arranged and less overwhelming when first used. However, certain drawbacks are too limiting in order to be relevant in the long run. There is no documentation except for the community site on GeoNet<sup>69</sup>, offering an introduction tutorial after further clicking through articles. The tutorial itself<sup>70</sup> only gives an basic opening to the tool which can also be explored intuitively. Questions asked by users are seldom answered, which lets the developer much to be desired in regards of communication. The biggest disadvantage is the inability to add or remove layers, so that after deciding the base map the map content is set. Without knowing the base map's contents it is hard determine in the beginning, which base map fits best for the intended purpose. This characteristic undermines the customization of a user-specific map design. Filtering feature groups is therefore difficult and changing a language is also not supported. When finished, the style is saved on the ArcGISOnline account but how to use it for further applications is not easily accessible, only mentioned after intensive research. The custom style can only be used with the ArcGIS API for JavaScript using the unique id of the style when saved<sup>71</sup>.

<sup>69</sup> <https://community.esri.com/community/arcgis-vector-tile-style-editor/pages/overview>

<sup>70</sup> <https://developers.arcgis.com/labs/arcgisonline/style-a-vector-basemap/>

<sup>71</sup> <https://developers.arcgis.com/labs/javascript/display-a-styled-vector-basemap/>

### 3.3.3.6 Result Comparison Tools

Google Styling Wizard limits the user too much and is therefore not considered as a valuable option for implementing the styling strategy. Esri ArcGIS Vector Tile Editor (beta) establishes a well-balanced interface with a lot of functionalities but is still highly limited compared to other tools. Although Mapbox Studio offers more convenience, functionalities and a quick work in, it limits the user from using various vector data sources, a functionality that also MapTiler does not provide. On the other hand, Maputnik does provide it. By being an open-source project, Maputnik is developer friendly and offers a detailed documentation with ongoing maintenance, communication and development. Even though the interface and its various services requires some time to get used to, where there are limitations by other tools, Maputnik offers a solution. The easy set-up and use without any previous registration is another advantage. Therefore, Maputnik will be used for implementation.

After the preparation is done, the predefined components are implemented with the help of Maputnik. Starting with an empty map, data sources were included in order to be able to add a layer. The table below lists all used sources.

<b>OpenMapTiles</b>	<i>Terrain &amp; Grey</i>	Water, Boundaries, Transport, Building	<a href="https://free.tilehosting.com/data/v3.json?key={accessKey}">https://free.tilehosting.com/data/v3.json?key={accessKey}</a>
<b>Mapbox Terrain</b>	<i>Terrain</i>	Land cover	<a href="https://mapbox.com/data/v3.json?key={accessKey}">mapbox://mapbox.mapbox-terrain-v2</a>
<b>Mapbox Terrain DEM</b>	<i>Terrain</i>	DEM	<a href="https://mapbox.com/data/v3.json?key={accessKey}">mapbox://mapbox.terrain-rgb</a>

Table 3.2: used sources in web map

Following, the layers were added, first being the background layer and the water (`class=ocean,river,lake`), to display the land masses. The water colour changes throughout the zooming; change stops are: z3 (`#99CCFF`), z7 (`#6699CC`) and z13 (`#99CCFF`). The RGBcoded raster DEM by Mapbox was rendered just above the background layer with an exaggeration of 0.5 and black as the shadow and accent colour, white as the highlight colour. Two separate boundaries layer were added, one depicting the country borders as solid lines starting at z1 and one drawing state borders as dotted lines starting at z4, both getting wider while zooming in.

For the *Terrain* style global land cover was added in separate layers classified by Mapbox: crop, grass, scrub, wood and ice. The previous declared colours were assigned with an opacity of 0.1. The *Grey* style is using land cover likewise, but supplied by OpenMapTiles land cover layer classified to woods, so that only forests are drawn on the map. Apart from the land cover and hillshade, the same design adjustments are made for *Grey*.

The symbolization of the transportation layers (different classified roads plus rail) were more complex than the previous layers. They were drawn on top of the land cover as white lines first, widening the line and transforming to a darker grey all the while adding an almost identical layer with smaller width in white. Figure 3.20 demonstrates the two different visualizations for the same features. A similar effect is achieved with rails with first showing grey lines and then adding a white dashed inner line to the grey casing.

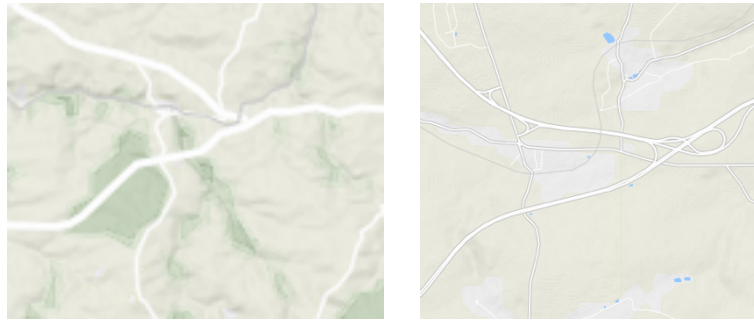


Figure 3.20: left showing symbolization of roads as white lines, right showing same features with grey casing

Two building fill layers aim to create a pleasant user experience in combination with an additional 3D rendered fill extrusion layer. One building fill layer holds the original position in light grey, the other building fill layer underneath is slightly translated (3,1) in dark grey intending a shadow simulation. First buildings as simple polygons in grey with a maximum opacity of 1 and a fading out effect followed by a fading in of light grey 3D rendered buildings dependent on their `render_height` value with an opacity of 0.3.

As the last step labels were added. *Metropolis* serves as the font family with different styles to differ the classes and characterisations. The table below lists all assigned font stylings. Except for Water (#003366), all text is set as dark grey (#333333, #666666) with a text size between 10.5 (suburbs) to 16 (continents). Occasionally a halo of light grey

(#CCCCCC) is added to increase the legibility. Arial Unicode MS is used as a fall back in case of technical issues or server problems when rendering the map.

Metropolis Regular	Cities
Metropolis Light	Continent (uppercase), Suburbs (italic)
Metropolis Extra Light	Towns, Water (italic)
Metropolis Semi Bold	Countries (uppercase), Capitals
Metropolis Medium	State Capitals
Metropolis Thin	Transportation

Table 3.3: Metropolis font weight usage

Capitals, state capitals, cities and towns were assigned icons to add texture to the map design and underline the importance of these features. From z4 to z8 symbols are used for different cities. Capitals receive a small square (4 by 4px), based on the traditional atlas designs, where capitals and metropolises were symbolized by squares. State capitals, cities and towns are symbolized by dots differing by size. There are no icons used other than the mentioned. It was considered to use a circle with a star cut-out for the capitals, but it broke up the map design and caught too much attention.

### 3.3.4 Client Applications

In certain use cases it is important to know which client is available to be able to use the base map and further develop additional applications. These are libraries based on JavaScript (APIs) to rendering interactive web maps. A code snippet demonstrates how to implement a style.json file to easily generate an interactive custom web map.

### Google Maps JS API

The API is accessible with Google API key and lets the user embed Google Maps on a website or mobile devices. The style is directly implemented in the JavaScript code without including an external file.

```
<script>
var map = new google.maps.Map(document.getElementById('map'), {
  center: {lat: 13.72, lng: -51.03},
  zoom: 12,
  styles: [{
    featureType: 'road',
    elementType: 'geometry',
    stylers: [{colour: '#38414e'}]
  }, {
    featureType: 'road',
    elementType: 'geometry.stroke',
    stylers: [{colour: '#212a37'}]
  }, {
    featureType: 'road',
    elementType: 'labels.text.fill',
    stylers: [{colour: '#9ca5b3'}]
  }
]);
</script>
```

### MapboxGL JS

Mapbox GL JS<sup>72</sup> is an open-source library based on WebGL embedding customizable maps in web. The Mapbox access token is an requirement to use the API.

```
<script>
mapboxgl.accessToken = '[personalKey]';
var map = new mapboxgl.Map({
  container: 'map',
  style: 'http://URL-TO-STYLE.json',
  center: [13.723776, 51.0290301],
  zoom: 12.24
});
</script>
```

### OpenLayers

OpenLayers is a well-established JavaScript library that creates web maps in web browsers from different sources adding various features. In order to display the vector style an OpenLayers plugin called *ol-mapbox-style* is needed.

```
<script>
var style = 'https://URL-TO-STYLE.json';
var map = new ol.Map({
  target: 'map',
  view: new ol.View({
    center: ol.proj.fromLonLat([13.723776, 51.0290301]),
    zoom: 12
  })
});
olms.apply(map, style);
</script>
```

### LeafletJS

LeafletJS - another open-source JavaScript library for web browsers - allows the display of web maps and additional information. There is a plugin called *MapboxGL Leaflet* that supports the compiling of style.json, but as of now is experimental.

```
<script>
var map =
L.map('map').setView([13.723776,
51.0290301], 12);
var gl = L.mapboxGL({
  style: http://URL-TO-STYLE.json'
}).addTo(map);
</script>
```

<sup>72</sup> <https://www.mapbox.com/mapbox-gl-js/api>

### Esri ArcGIS API for JavaS- cript

The API enables the possibility to create 2D and 3D data visualizations and maps. A registration on ArcGISOnline is required to access a personal key and save a custom style with a unique ID.

```
<script>
require([
  "esri/Map",
  "esri/views/MapView",
  "esri/Basemap",
  "esri/layers/VectorTileLayer",
  "dojo/domReady!"],
function(Map, MapView, Basemap,
VectorTileLayer) {
  var basemap = new Basemap({
    baseLayers: [
      new VectorTileLayer({
        portalItem: {
          id: "UNIQUE-ID"
        })
      ]
    });
  var map = new Map({
    // basemap: "topo-vector",
    basemap: basemap
  });
  var view = new MapView({
    container: "viewDiv",
    map: map,
    center: [13.72, 51.03],
    zoom: 10
  });
});
</script>
```

### TangramJS

The library renders 2D and 3D maps with WebGL supporting any GeoJSON, TopoJSON and vector data such as single files and tilesets. It was developed as a Leaflet plugin, so LeafletJS is required in order to display the map. A separate scene file written in YALM holds information about data sources, layers and styling rules, similar to the style JSON file, but follows a different schema<sup>73</sup>.

```
<script>
var map = L.map('map');
var layer = Tangram.leafletLayer({ scene: 'URL-TO-SCENE.yaml' });
layer.addTo(map);
</script>
```

Table 3.4: Overview of client applications supporting custom styles

#### 3.3.4.1 MapboxGL JS

The setup of the frontend of a web map with the custom style is comparable easy. The exported JSON file is used to draw a web map with MapboxGL JS by using the following code<sup>74</sup>:

<sup>73</sup> Demo of a scene file used for TangramJS: <https://github.com/tangrams/simple-demo/blob/master/scene.yaml>

<sup>74</sup> On <http://laura-lou.com/webmaps/master/> the web map can be seen with additional functions, such as a dynamic zoom level display and style switch.


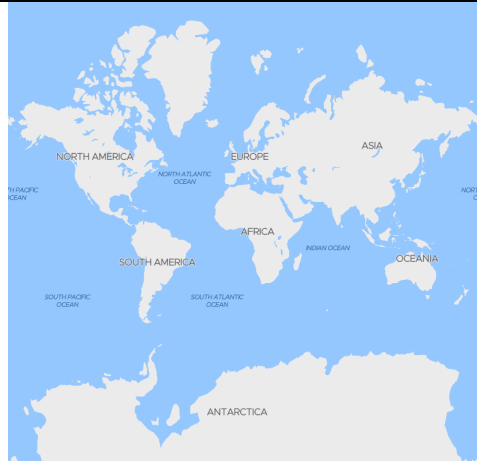
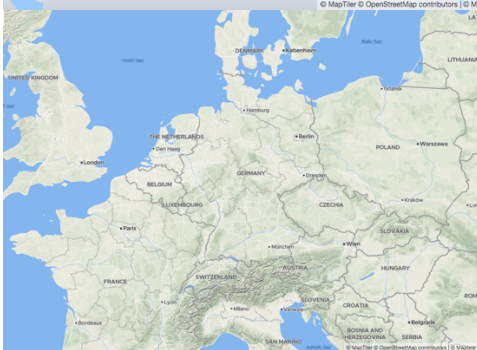

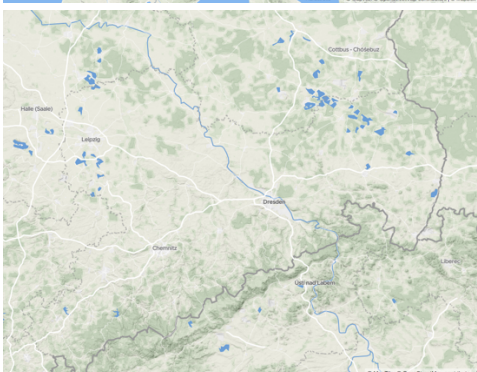
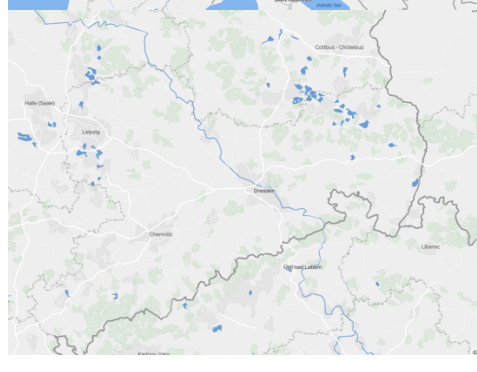
```
<html>
<head>
  <meta charset='utf-8' />
  <script src='https://api.tiles.mapbox.com/mapbox-gl-
js/v0.48.0/mapbox-gl.js'></script>
  <link href='https://api.tiles.mapbox.com/mapbox-gl-
js/v0.48.0/mapbox-gl.css' rel='stylesheet' />
  <style>
    #map {
      position: absolute;
      top: 0;
      bottom: 0;
      width: 100%;
    }
  </style>
</head>
<body>
  <div id='map'></div>
  <script>
    mapboxgl.accessToken = 'personalKey';
    var map = new mapboxgl.Map({
      container: 'map',
      style: 'URL-TO-STYLE.json',
      center: [13.74, 51.05],
      zoom: 11
    });
  </script>
</body>
</html>
```



## 4 Result & Discussion

### 4.1 Finished Map Design

Table 4.1 gives an overview of both map styles compared on different zoom levels, a more detailed version with all zoom levels from z0 to z16 is listed in the appendix (V). Apart from the land cover and hillshade, the same layers were used in both styles. Forests were included into the *Grey* style, because without any polygon features it appeared too cold and identical to already existing web map styles (e.g. CartoDB Positron<sup>75</sup>).

zL	Terrain	Grey
z0		
z4		
z7		

<sup>75</sup> <https://carto.com/blog/getting-to-know-positron-and-dark-matter/>



The used colours are web safe to ensure a correct rendering on the majority of hard ware terminals. Colours associated to the features and established in cartographic symbolizations aid the intuitive recognition process by the map reader. Therefore, water was set as blue, forests as green and glaciers as light blue. *Terrain* features like crop were visualized as yellow, grass as a bright green and scrub as a slightly darker green. So that a gradient from the most underneath features (crop) to the layers on top (forest) can be achieved. The remaining elements are styled with a combination of white and grey to provide enough contrast to the colours with readability but remain as unobtrusive as possible. Low opacity (0.1) enabled an interesting colour mix when features overlap and blend with the light grey background. The ice features on top of the DEM enhance the mountain visualizations. A colourful yet unobtrusive background is provided, see figure 4.1.



Figure 4.1: *Terrain* Map snippet displaying all land cover layers

Water was divided into polygon features and line features while providing the same symbolization. Ocean and lake polygons were included from  $z_0$ , while river polygons were

visible from z12, as before river line provide data in smaller scales. The changing point of the line to polygon transition was difficult to decide as the data between these layers did not correspond in many regions, i.e. rivers, that were symbolized as a line feature may not be included as a polygon feature until later on. Ultimately, z11 to z12 were set as the changing point, by fading out line features while simultaneously fading in polygon features to simulate a smooth transition of feature types. The water's colour is changing gradually from z3 (brighter blue) to z7 (slightly darker blue) to z13 (back to brighter blue). It begins with a brighter blue because in smaller scales the ocean mass takes up the majority of the screen and big areas should be visualized brighter to not put a strain on the map users view. According to Imhof (1972) if vibrant colours were to be restricted to small areas an expressive map structure emerges. Because of this when ocean areas become smaller and lakes as well as rivers appear on the map, the blue turns darker. Further zooming in transforms the lakes and rivers again to bigger features whereas the colour changes back to a lighter blue.

One of Maputnik's numerous functions and an innovative advancement of vector tiles is the rendering of 3D polygons. The layer type *Fill extrusion* allows to set an extrusion height and base. Applying this technique on the layer buildings enabled by the deposited value of `render_height` and `render_min_height`, interesting and realistic shapes are created. While labels work with the field value, this is not working with the extrusion height and base as a number is expected. However, when inserting the following code snippet the extrusion of the polygons works as desired.

```
[...]
"paint": {
  "fill-extrusion-colour": "rgba(153, 153, 153, 1)",
  "fill-extrusion-height": {
    "property": "render_height",
    "type": "identity"
  },
  "fill-extrusion-base": {
    "property": "render_min_height",
    "type": "identity"
  },
  "fill-extrusion-opacity": 0.3
}
[...]
```

A Workaround this issue highlights a certain drawback of Maputnik and requires further development. Another approach on shadow casting would be a second duplicate layer in a darker tone with a minimal translation, e.g. 3 and 2. That was included for the simple fill polygons of buildings to create a smooth transition between both layers. Figure 5.2a shows both building fill layer and building shadow fill layer that establish a minimal 3D effect, that later on get transitioned into real 3D rendering with values as shown in figure 5.2b.

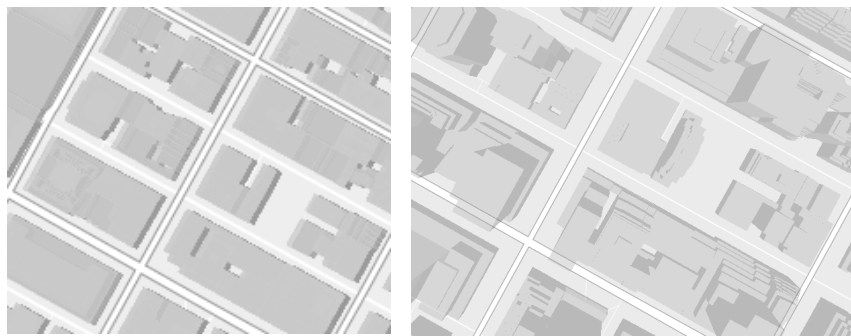


Figure 4.2a: building fill layer at z15 with duplicate building shadow fill layer (translated 3, 1)

Figure 4.2b: building fill extrusion layer at z16

It is possible to only add the 3D building layer from z14 and render shadows that way but the map can get too busy on smaller scales. Figure 4.2 depicts roughly the same area as the previous pictures in 4.3, symbolized with the *OSM Liberty* style. The map seems too busy and cluttered and depending on the view angle the street names and POI labels are hardly readable. Furthermore, the shadow rendering in smaller scales generate jagged lines which decreases the map use experience and legibility as well.



Figure 4.3: OSM Liberty vector style showing fill extrusion layer in z14.5

Metropolis is used as the only font, Arial Unicode MS as a fall back. To not imitate existing map styles, it was refrained from using Noto font or Roboto. The decision of font weight and size aims to provide clarity, readability and contextual labelling. Halos support the legibility on busy backgrounds. The font weight and size corresponds to the significance of the feature group creating a distinct hierarchy with a tidy appearance.

## 4.2 Layer Order and Zoom Dependence

The layer order resulted to the following (top being the first layer to draw and situated the most underneath):

- Background
- Hillshade / DEM
- Global land cover
- Water
- Boundaries
- Transportation
- Buildings
- Labeling

The background of the map is created by the combination of background colour, hillshade and overlapping global landcover due to low opacity values. The vibrant colour mix provide a solid basis for further features. Water is drawn above the land cover because there are instances where land cover mask overlap or wholly cover a water feature. Naturally, transportation features are situated above the land cover and water features. Buildings are drawn on top of these as the fill extrusion can overlap the roads when shifting the map view located on high buildings. When examining default Maputnik styles (OSM Bright, Klokantech basic, Mapbox basic, etc. ) it can be seen, that the layer order is similar as the proposed one.

The decision at which zoom level a layer should be added is mostly made by the vector data provider as predefined selections were done. For example, capital labels do not show up until z4 even though minZoom is given at 0. The developer is therefore already restricted from own design decision. Nevertheless, the following table suggests zoom levels on when to first add (+) and remove (-) feature groups that proved to be appropriate. The result was accomplished by comparing established styles and own experiences while designing the multi-scale map style.

<b>z0</b>	+ background + water (ocean) + continent labels + major ocean labels + landcover + hillshade
<b>z1</b>	+ country boundaries + two-letter country code label
<b>z2</b>	+ state boundaries for major countries, e.g. USA, Canada, Brazil or Australia + two letter state labels for major countries - continent labels - two-letter country code label + country labels
<b>z3</b>	+ waterways (river) + sea water labels + state boundaries - two letter state labels for major countries + state labels
<b>z4</b>	+ city icons and labels
<b>z5</b>	+ roads
<b>z6</b>	+ lake water labels
<b>z7</b>	+ highway labels + water (river polygon)
<b>z8</b>	+ rail - waterways (river)
<b>z9</b>	- country labels
<b>z10</b>	+ land use
<b>z11</b>	+ POIs
<b>z12</b>	+ road labels + road orientation roads - city labels + suburb labels
<b>z13</b>	+ land use label + buildings
<b>z14</b>	+ contour lines and labels + footway + building extrusion

Table 4.2: Addition and removal of feature layers dependent on zoom level

### 4.3 Map continuity analysis

As before the established styles *Terrain* and *Grey* were analysed by the map complexity method outlined in chapter 3.1.1. Screenshots from z4 to z16 of identical areas were used to investigate the complexity with the help of the grey value's mean<sup>76</sup>. Figure 4.4 Displays the values comparing to the previous styles. The jump from z4 to z5 is again very prominent, which is due to the decreasing water area when zooming. While the grey line stays comparably steady between the grey values 225 and 235, it shows that the other style (*Terrain*) gets brighter with every zoom level until the peak at z13. From z14 buildings are displayed on the map in grey decreasing the mean value by casting darker shadows in z15 resulting in even darker patches. This is better resolved in *Grey* as residential areas were included in z6 before buildings in z14, while simultaneously drawing back the residential land cover. Comparing both styles, it is obvious that *Grey* seems to be the more continuous style of the both. Even though *Terrain* exhibits a peak after a steady rise for 9 zoom levels, it offers a better continuity than OSM carto style, which is also noticeable when examining the map style. To further analyse the map continuity different sample areas should be included, such as low urban and mountain areas.

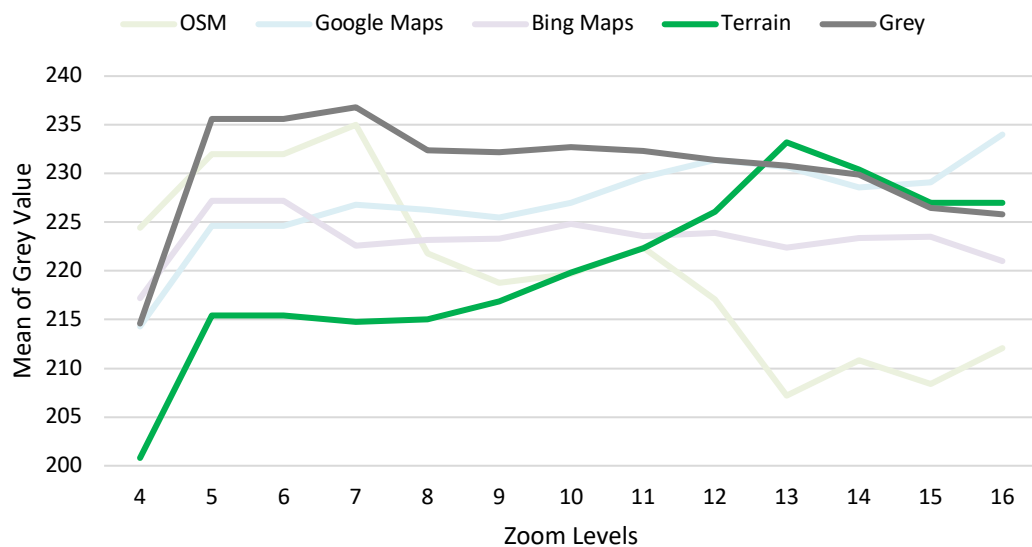


Figure 4.4: Overview of map complexity throughout zoom levels 4 to 16

<sup>76</sup> Screenshots and further information can be examined on  
<https://wwwpub.zih.tu-dresden.de/~s6845890/kartenstil/kartenbelastung-laura-terrain.html>  
<https://wwwpub.zih.tu-dresden.de/~s6845890/kartenstil/kartenbelastung-laura-grey.html>



#### 4.4 Colour blindness

By simulating different colour blindness diseases, legibility and colour distinction are still distinguishable. An overview of the changes between different colour blindness diseases and the original version is given in the following table. In this case only *Terrain* is used as demonstration as this style shows to most changes. The colour vision deficiency were simulated with the desktop app of Colour Oracle (mentioned in chapter 2.1.1.4).











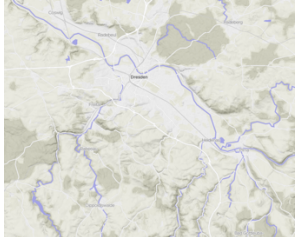




zL	Original <i>Terrain</i>	Deuteranopia	Tritanopia
z0.5			
z3			
z6			
z10			
z14			

Table 4.3: Comparison of *Terrain* in two different simulated colour blindness



As can be seen, for people with Deuteranopia the water colour is slightly off but still interpretable. Vegetation features remain visually distinct. Same applies to Tritanopia vision, even though land cover areas appear as a grey scale. Therefore, it can be assumed that the developed style is usable for people with colour vision deficiency.

#### 4.5 Restrictions

Even though the system in place enable innovative and individual designs, there are certain drawbacks that hinder the development process, one being the unavailable access to the vector data. The developer is dependent on the vector tile creator and their choices when it comes to generalization and changes between zoom levels. For example, in the surroundings of Dresden in z8 to z9 the river landscape changes so drastically that it can create confusion for map reader. These “geometry jumps” are individual in different places and responsible by the vector tile data. The degree of feature’s generalization is authorized by the vector data provider and cannot be manipulated with the proposed design framework. To be able to filter the data by additional thematical and numerical properties, e.g. area size, number of inhabitants or length would be a valuable addition, but also remaining a task for the vector data creator for providing additional data values. Development of an expression support is ongoing in regards of Maputnik<sup>77</sup>. Custom GeoJSON data is filterable by such expressions if the information is provided. The creation of custom vector tiles would also be a possible albeit elaborate solution.

As for styling property limits, Maputnik allows the developer a big amount of liberties and there is not much to be desired. To underline text was the only thing that appeared missing, when creating the web map style. To add symbols or icons and glyphs could be included more user friendlier. To figure out how to implement custom icons with sprite sheets is not beginner-friendly and turned out more complicated than necessary. A documentation for the step would already be helpful.

---

<sup>77</sup> <https://github.com/maputnik/editor/issues/223>

## 5 Conclusion

The main objective of this thesis was to develop a strategy to create a web map style across several scales. Vector tile maps emerged as the suitable medium to easily create custom styles due to their current growth regarding technical advancement and innovative adaptations. Elements of traditional and web cartography were reviewed and applied to multi-scale mapping. Available data sources were explored as well as the state of the art of tools to customize web maps. Maputnik fit the most criteria allowing the highest liberties and still maintain a user friendly interface that enables autonomous use. The resulting web map styles that were created accomplish the desired intention to serve as a thematic base map while still offering the possibility as a stand-alone map. The proposed workflow is clear and holds the advantage of being adaptable to different tools and data sources supporting even more customization.

The specialty of multi-scale map design lies in the change between map scales entailing map features' appearing and disappearing dependent on zoom levels. The challenge to provide a map style across the scales that stays continuous without visual breaks adds to the peculiarity of multi-scale map design. Traditional and web map design techniques can be certainly applied to the design concept but still a demand of specially tailored design rules exists. Visualization components divided into categories of geometry, colour and relationship were proposed to offer guidelines for the specialty of multi-scale web map design. A balanced map appearance across zoom levels is investigable by examining the map complexity. A harmonized map can be achieved with steady map complexity, meaning the continuous behaviour of features balanced to the appearance of new layers. Blatant colour changes or the addition of too many layers at once create breaks within the styles. Google Maps, for example, is using multi-scale as an active part of their design by changing colours and creating a dynamic map view change gradually from global view to flat and back. Google's ongoing development and technical advancement inspires for the possibilities of vector tile maps.

Shortcomings of available data restrict the developer and lets improvement to be desired. A documentation with exact descriptions is indispensable. Therefore future advancement should focus on more detailed documentation and schemas for layers, outlining generalization stages including minimal or maximal zoom levels and major changes in feature geometries.

Further research is proposed to focus on an automated ways on designing multi-scale web maps, as the manual workload is still immense. Approaches have been made, , e.g. Mapbox Cartogram<sup>78</sup>, allowing to upload a photo and selecting its colours, but lets much to be desired.

The development of a dynamic legend would be an innovative inclusion increasing insight for the map user in case of many layers and furthermore offering new possibilities for multi-scale mapping.

---

<sup>78</sup> <https://www.mapbox.com/cartogram>

## References

- Anon., 2018. *History of OpenStreetMap*. [Online]  
Available at: [https://wiki.openstreetmap.org/wiki/History\\_of\\_OpenStreetMap](https://wiki.openstreetmap.org/wiki/History_of_OpenStreetMap)  
[Accessed 25 June 2018].
- Anon., 2018. *Tile Layers*. [Online]  
Available at: <http://doc.arcgis.com/en/arcgis-online/reference/tile-layers.htm>  
[Accessed 12 July 2018].
- ArcGIS Online, 2018. *Tile Layers*. [Online]  
Available at: <http://doc.arcgis.com/en/arcgis-online/reference/tile-layers.htm>  
[Accessed 31 July 2018].
- Arnberger, E., 1977. *Thematische Kartographie. Mit einer Kurzeinführung über Automation in der thematischen Kartographie..* Braunschweig: Westermann.
- Atzl, C., 2016. How to design web maps that users like?. In: *Online-Karten im Fokus - Praxisorientierte Entwicklung und Umsetzung*. Berlin/Offenbach: Wichmann, pp. 40-44.
- Battersby, S. E., Finn, M. P., Usery, E. L. & Yamamoto, K. H., 2014. Implications of Web Mercator and Its Use in Online Mapping. *Cartographica*, Issue 49 (2).
- Brewer, C. A., 2016. *Designing better maps - A guide for GIS users*. 2nd Edition ed. Redlands: Esri Press.
- Brewer, C. A. & Buttenfield, B. P., 2007. Framing Guidelines for Multi-Scale Map Design Using Databases at Multiple Resolutions. *Cartography and Geographic Information Science*, 34(1), pp. 3-15.
- Brewer, C. & Buttenfield, B., 2007. Framing Guidelines for Multi-Scale Map Design Using Databases at Multiple Resolutions. *Cartography and Geographic Information Services*, 34(1), pp. 3-15.
- Brewer, C. & Buttenfield, B., 2010. Mastering map scale: balancing workloads using display and geometry change in multi-scale mapping. *Geoinformatica*, April, 14(2), pp. 221-239.

Brewer, C., Battenfield, B., Frye, C. & Acosta, J., 2007. *Scalemaster: Multi-Scale Mapmaking from Multiple Database Resolutions and for Multiple Map Purposes*, s.l.: s.n.

contributors, W., 2018. *Web Map Tile Service*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Web\\_Map\\_Tile\\_Service](https://en.wikipedia.org/wiki/Web_Map_Tile_Service)  
[Accessed 12 July 2018].

Dayley, A., 2005. *Windows Live Local powered by Virtual Earth!!*. [Online]  
Available at: <https://blogs.bing.com/maps/2005/12/07/windows-live-local-powered-by-virtual-earth>  
[Accessed 25 June 2018].

de la Torre, J., 2018. *Using Mapbox Vector Tiles in CARTO for Maps & Location Apps*. [Online]  
Available at: <https://carto.com/blog/using-mvt-in-carto/>  
[Accessed 31 July 2018].

Dent, B. D., Torguson, J. S. & Hodler, T. W., 2009. *Cartography - Thematic Map Design*. 6th Edition ed. New York: McGraw-Hill.

Esri, 2006/2007. *ArcGIS Online Services Introduced*. [Online]  
Available at: <http://www.esri.com/news/arcnews/winter0607articles/arcgis-online.html>  
[Accessed 3 August 2018].

Esri, 2006. *Comparing Vector and Raster Mapping for Internet Applications*. s.l.:Esri White Paper.

Fischer, L., Gert, E. & Hebestreit, N., 2016. Mobile Karten. In: S. Hennig, ed. *Online-Karten im Fokus*. Berlin, Offenbach: Wichmann, pp. 89-104.

Gaffuri, J., 2012. Towards Web Mapping with Vector Data. In: *Geographic Information Science. GIScience 2012. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, pp. 87-101.

Gibson, R. & Erle, S., 2006. *Google Maps Hacks*. Sebastopol: O'Reilly Media, Inc.

Goodchild, M. F., 2007. *Citizens as Sensors: the world of volunteered geography*, s.l.: s.n.

Hahmann, S. & Burghardt, D., 2010. Linked Data - A Multiple Representation Database at Web Scale?. In: *Proceedings of the 13th ICA Workshop on Generalization and Multiple Representation*. Zurich: s.n.

Harlan, C., 2015. 'Does MapQuest still exist?' Yes, it does, and it's a profitable business.. [Online]

Available at: [https://www.washingtonpost.com/business/economy/does-mapquest-still-exist-as-a-matter-of-fact-it-does/2015/05/22/995d2532-fa5d-11e4-a13c-193b1241d51a\\_story.html?utm\\_term=.d7753e24aef1](https://www.washingtonpost.com/business/economy/does-mapquest-still-exist-as-a-matter-of-fact-it-does/2015/05/22/995d2532-fa5d-11e4-a13c-193b1241d51a_story.html?utm_term=.d7753e24aef1)

[Accessed 3 August 2018].

Hennig, S., 2016. *Online-Karten im Fokus - Praxisorientierte Entwicklung und Umsetzung*. Berlin/Offenbach: Wichmann.

Imhof, E., 1972. *Thematische Kartographie*. s.l.:De Gruyter.

Jansen, M. & Adams, T., 2010. *OpenLayers - Webentwicklung mit dynamischen Karten und Geodaten*. München: Open Source Press.

Kessler, F. C., Battersby, S. E., Finn, M. P. & Clarke, K. C., 2017. Map projections and the Internet. In: *Choosing a map projection*. s.l.:Springer, pp. 116-148.

Kraak, M.-J. & Ormeling, F., 2003. *Cartography - Visualization of Geospatial Data*. 2nd Edition ed. Essex: Pearson Education Limited.

Krygier, J. & Wood, D., 2016. *Making Maps - A visual Guide to Map Design for GIS*. 3rd Edition ed. s.l.:Guilford Publications.

Möller, M., 2016. Vom GIS zum Web-Mapping. In: *Online-Karten im Fokus - Praxisorientierte Entwicklung und Umsetzung*. Berlin/Offenbach: Wichmann, pp. 19-29.

Mackaness, W. A., 2007. Understanding Geographic Space. In: W. A. Mackaness, A. Ruas & L. T. Sarjakoski, eds. *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Oxford, Amsterdam: Elsevier, pp. 1-10.

Mapbox, 2018. *tileset / Mapbox*. [Online]

Available at: <https://www.mapbox.com/help/define-tileset/>

[Accessed 5 April 2018].

Masó, J., Pomakis, K. & Julià, N., 2010. *OpenGIS® Web Map Tile Service Implementation Standard*, s.l.: Open Geospatial Consortium Inc..

Mitchell, T., 2008. *Web Mapping mit Open Source-GIS-Tools*. 1st Edition ed. Köln: O'Reilley Verlag.

Muehlenhaus, I., 2014. *Web Cartography*. Boca Raton: CRC Press.

Onmaps, 2018. *WMTS und TMS - Karten in höchster Geschwindigkeit*. [Online]  
Available at: <https://onmaps.de/tile-map-service-tms-web-map-tile-service-wmts.html>  
[Accessed 12 July 2018].

Peterson, G. N., 2012. *Cartographer's Toolkit - Colours, Typography, Patterns*. Fort Collins: PetersonGIS.

Peterson, G. N., 2015. *GIS Cartography*. 2nd Edition ed. Boca Raton: CRC Press.

Peterson, M. P., 2012. Online Mapping with APIs. In: *Online Maps with APIs and WebServices*. Berlin Heidelberg: Springer Verlag, pp. 3-12.

Purvis, M., Sambells, J. & Turner, C., 2007. *Google Maps Anwendungen mit PHP und Ajax*. 1st Edition ed. Heidelberg: mitp.

Quinn, S. & Dutton, J. A., 2018. *Why tiled maps? | GEOG 585: Web Mapping*. [Online]  
Available at: <https://www.e-education.psu.edu/geog585/node/706>  
[Accessed 27 June 2018].

Robinson, A. H. et al., 1995. *Elements of Cartography*. 6th Edition ed. New York: John Wiley & Sons, Inc..

Roth, R. E., Brewer, C. A. & Stryker, M. S., 2011. A typology of operators for maintaining legible map designs at multiple scales. *Cartographic Perspectives*, Issue 68, pp. 29-64.

Sample, J. T. & Ioup, E., 2010. *Tile-Based Geospatial Information Systems - Principles and Practices*. Berlin, Heidelberg: Springer.

Samsonov, T., 2011. Multiscale Hypsometric Mapping. In: *Advances in Cartography and GIScience*. Berlin Heidelberg: Springer, pp. 497-520.

- Samsonov, T., Podolsky, A. & Yurova, N., 2013. *Multimapper - Prototype System for Designing Multi-Scale Maps*, s.l.: s.n.
- Sarjakoski, L. T., 2007. Conceptual Models of Generalisation and Multiple Representation. In: W. A. Mackaness, A. Ruas & L. T. Sarjakoski, eds. *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Oxford, Amsterdam: Elsevier, pp. 11-35.
- Schmidt, M. & Jörg, W., 2012. basemap.at - Creating a Harmonised Web Basemap for Austria. In: M. Jobst, ed. *Service-Oriented Mapping 2012*. Wien: JOBSTmedia Management Verlag, pp. 143-149.
- Schulz, T., 2014. *Der statistische Atlas : Untersuchungen zu klassifikatorischen, inhaltlichen, gestalterischen, technischen und kommunikativen Aspekten*. Dresden: Inst. für Kartographie .
- Schwartz, J., 2018. *Bing Maps Tile System*. [Online]  
Available at: <https://msdn.microsoft.com/en-us/library/bb259689.aspx>  
[Accessed 12 July 2018].
- Slocum, T. A., McMaster, R. B., Kessler, F. C. & Howard, H. H., 2009. *Thematic Cartography and Geovisualization*. 3rd Edition ed. Upper Saddle River(NJ): Daniel Kaveney.
- Sterling Quinn, J. A., 2018. *GEOG 585 Open Web Mapping*. [Online]  
Available at: <https://www.e-education.psu.edu/geog585/node/519>  
[Accessed 13 July 2018].
- Thompson, B., 2018. *Announcing Vector Tile 3.0 Specification Development*. [Online]  
Available at: <https://github.com/mapbox/vector-tile-spec/issues/103>  
[Accessed 31 July 2018].
- Thompson, B., 2018. *HD vector maps open standard*. [Online]  
Available at: <https://blog.mapbox.com/hd-vector-maps-open-standard-335a49a45210>  
[Accessed 31 July 2018].
- TileLayers, 2018. *Tile Layers*. [Online]  
Available at: <http://doc.arcgis.com/en/arcgis-online/reference/tile-layers.htm>  
[Accessed 12 July 2018].



- Touya, G. & Girres, J.-F., 2013. ScaleMaster 2.0: a ScaleMaster extension to monitor automatic multi-scales generalizations. *Cartography and Geographic Information Science*, 40(3), pp. 192-200.
- Traun, C., 2016. Wahrnehmungsorientierte Kartengestaltung. In: S. Hennig, ed. *Online-Karten im Fokus*. Berlin Offenbach: Wichmann, pp. 71-88.
- Tyner, J. A., 2010. *Principles of Map Design*. New York: Guilford Press.
- Weber, H., 2008. *Handbuch Farbkombinationen Webfarben - Die richtige Farbkombination für jedes Design*. Heidelberg: REDLINE GmbH.

## Appendix I: Overview of Websafe colors

**ZoNi.Web! - Mala Web škola**  
**WEB SAFE COLOR CHART**

Compiled by Professional Web Design - <http://junior.apk.net/~jbarta/>

These are the grays:

These are the closest (more or less) to the 16 basic Windows colors:

## Appendix II: ScaleMaster Operators

List of operators used in the ScaleMaster diagram (Roth, et al., 2011)

Operator	Abbr.	Short description	Use case
<b>Content</b>			
Add	C+	Insert feature type once a certain scale is reached	Include country boundary lines
Eliminate	C-	Remove feature as soon as a certain scale is reached, where purpose is not justified	Remove building polygons
Reclassify	Cc	Alter feature's organization based on attributes	green area is split in national area and forest
Reorder	Co	Change of the order how features are placed on one another	
<b>Geometry</b>			
Aggregate	Gg	Multiple related features are encapsulated to a representative feature	
Collape	Gc	feature is replaced by a representative feature of smaller dimension	Polygon of a city's residential area is replaced by a point symbol in smaller scales
Merge	Gm	Feature is replaced by a representative feature of the same dimension	
Displace	Gd	Location of a feature is adapted to maintain overall typography	imitate shadow of polygon features by adding a duplicate displaced layer underneath
Exaggerate	Gx	Emphasizing characteristic aspect of feature	
Simplify	Gs	Reducing points that draws a feature	detail of features not needed on smaller scales
Smooth	Go	Deletion of variations in geometry	detail of features not needed on smaller scales
<b>Symbol</b>			
Adjust colour	Sc	Colour is altered to maintain readability	colour is brightened the more you zoom out
Enhance	Se	Beautifying features to accentuate feature relationships	
Adjust Iconicity	Si	Changing iconic look without changing feature dimension	Include pattern to forest area on larger scales

Adjust Pattern	Sp	To maintain legibility, pattern of fill or stroke is changed	Change simple geometric shapes in more complex iconic shapes
Rotate	Sr	Orientation of symbol is changed	Updated wind orientation on point symbols
Adjust Shape	Ss	Changing shape of feature without changing feature dimension	
Adjust Size	Sz	Changing symbol size without changing feature dimension	City point symbols enlarged when zoomed in
Adjust Transparency	St	Adjustment to the symbol opacity are made	Opacity of residential increased when zooming in
Typify	Sf	Multiple related features are replaced by representative, more scattered arrangement of symbols	

**Label**

Add Label	L+	Once scale is reached, label is inserted	add road labels
Eliminate Label	L-	Once scale is reached, label is removed	remove country labels
Adjust Appearance	La	Styling parameters are adjusted to a set of labels	enlarge font when zooming in
Adjust position	Lp	Placement of label is changed in relating to the symbol	capital city labels positioned centred after icon disappeared

## Appendix III: Worksheets to Aid Editorial Process

Worksheet Features: Zoom-Level			
<b>Layer: boundary, line</b> Lines spanning administrative borders, such as zone line, state, county. Name: Nationalities, OSM ID: Filtered by address, (area), values 1-4		<b>Layer: point, Point</b> Labels for places such as continents, i.e. Asia or capitals, i.e. Paris. Source: OSM ID # Filtered by address, values: area,	
address_way 1      address_way 2 address_way 3      address_way 4		continent country state city town	
		village hamlet suburb neighborhood isolated_dwelling	
Zoom	Area	Feature	Style
18	continent		
19			
20			
21			
22			
23	line		
24			
25			
26			
27			
28	boundary		
29			
30			
31			
32			
33			
34			
35			
36			
37			

<b>Worksheet Features Zoom-Levels</b>			<i>Polygons of buildings Label for measurements</i>
Layer: Invis.Lidings:	<b>Polygon_Poin</b>		
Layer: Measurements:	<b>Point</b>		
Source: CDD editor zoom:None			
ZL	S	Name	Doping
V13			
V14			
V15			
V16			
V17			
V18			

Layer: poi.: **Polygon\_Poin**      Label and Polygons of Points of Interest, such as shops, facilities, ...  
Name: CDD editor

(Use by clients and subscribers from back page)

ZL	S	Name	Doping
V12			
V13			
V14			
V15			
V16			
V17			
V18			

[illegible]

Worksheet Features Zoom-Level				
<b>Layer: Isocurves, Polygon</b> physical material on the surface of Earth Source: <a href="#">Isocurves</a> , <a href="#">Color LUT</a> Filtered by: Isocurves and subcategories (see back page)			<b>Layer: Isocurves, Polygon</b> shown and used by: <a href="#">Isocurves</a> Source: <a href="#">Isocurves</a> (A-B, <a href="#">GDW</a> (2016)) Filtered by: Isocurves and subcategories (see back page)	
<b>Layer: isostatic_packs, Point</b> shown and used by: <a href="#">Isostatic</a> Source: <a href="#">GDW</a> (2016) Filtered by: <a href="#">Isostatic</a> , <a href="#">isostatic_packs</a>			<b>Layer: packs, Polygon</b> shown: <a href="#">GDW</a> (2016) Filtered by: <a href="#">Isostatic</a> with values <a href="#">isostatic_packs</a> , <a href="#">isostatic_packs</a>	
20c	1	name	Resource	Display
01	Isocurves (physical material)	01		
02				
03				
04				
05				
06				
07	Isocurves (physical material)	07		
08				
09				
10				
11				
12				
13				
14				
15				
16				
17	Isocurves (physical material)	17		
18				
19				
20				

Layer Landscape classes & subclasses			
<b>natural</b>	<b>terrace</b>	<b>cruciant</b>	<b>woodland</b>
wood	park	grassland	hedg
woodland	garden	grass	viney
grassland		thicket	oak_woodland
glacier	<b>bedstone</b>	heaps	heath
lake_nak	artificial	afforestation	woodland
creek	form	recreation_ground	urban
beach	formal	park	suburban
sand	artificial		suburban
	park(hobby)		suburban

Layer Landscape classes & subclasses		
<b>artificial</b>	<b>bedstone</b>	<b>terrace</b>
rock_formation	artificial	ice
school	ceremony	theme_park
university	iceberg	
kindergarten	recreational	<b>terrace</b>
college	commercial	thicket
library	industrial	park
hospital	water	playground

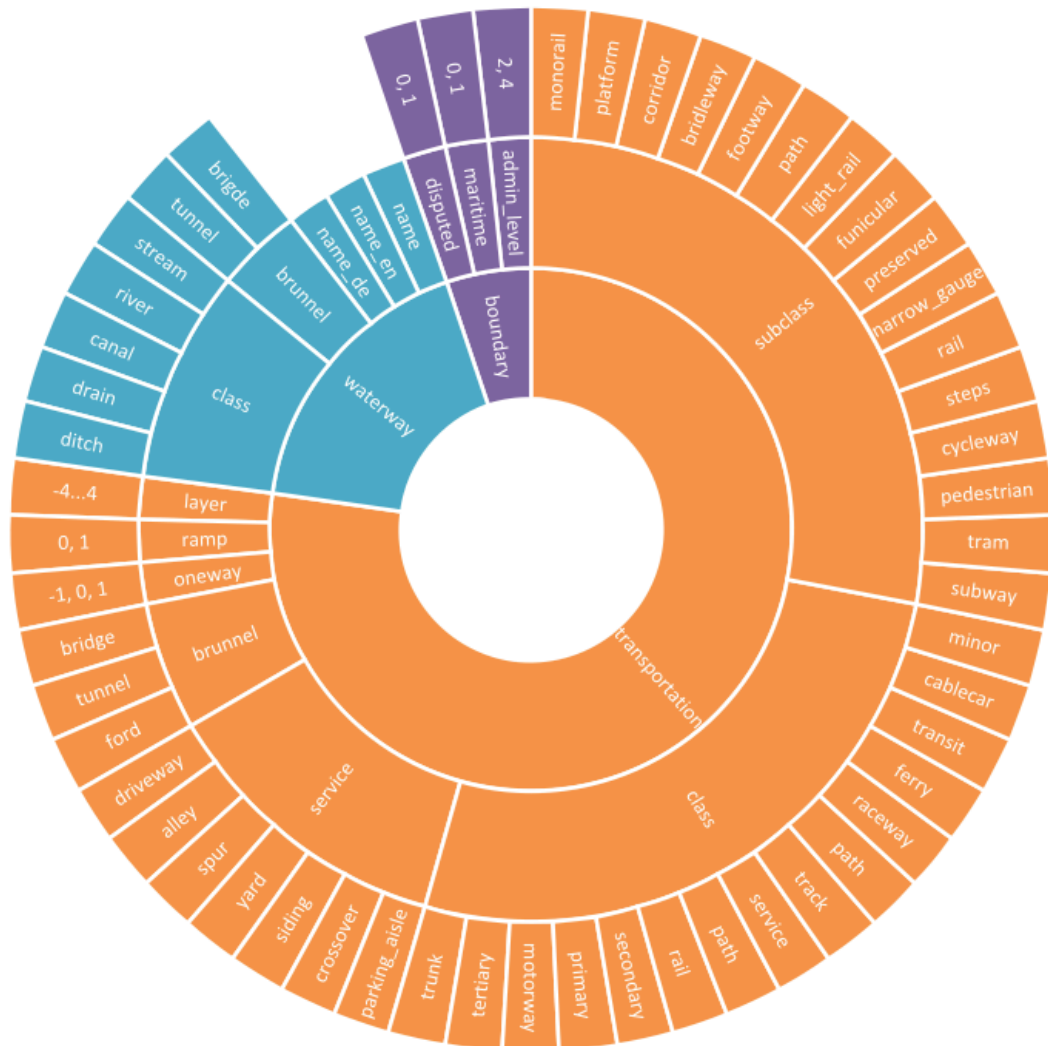
[illegible][illegible]

## Appendix IV: Layer Schema of OpenMapTiles and Mapbox

## OpenMapTiles Polygon Layer Schema



## OpenMapTiles Line Layer Schema



## OpenMapTiles Point Layer Schema





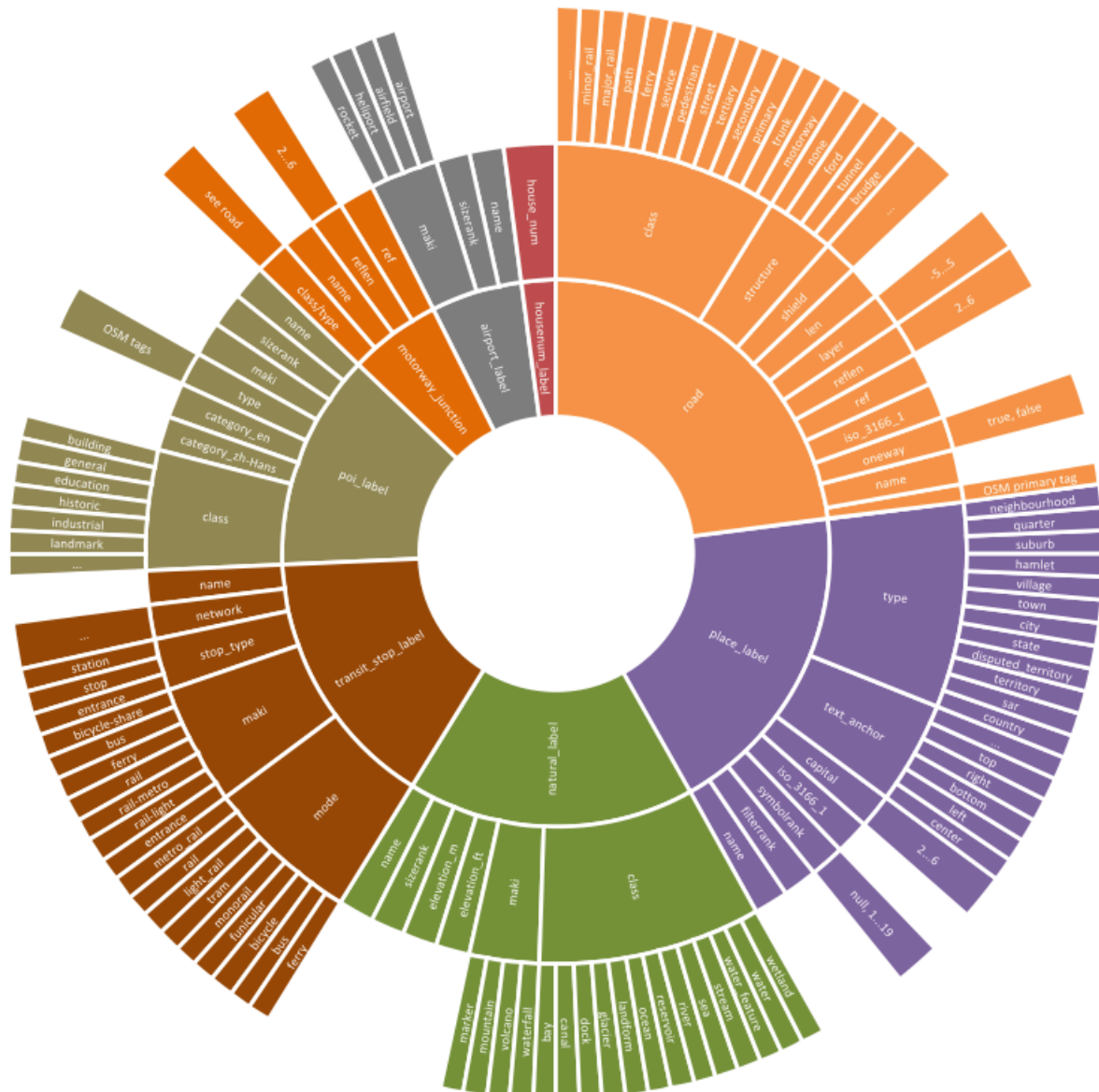
## MapBox Polygon Layer Schema















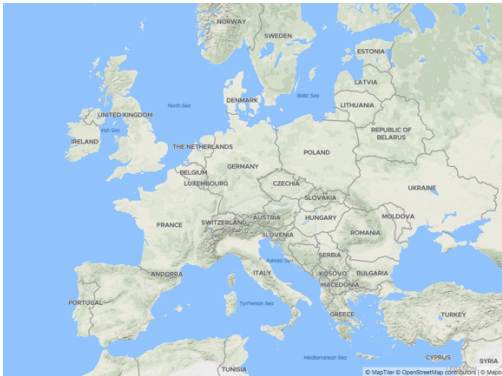

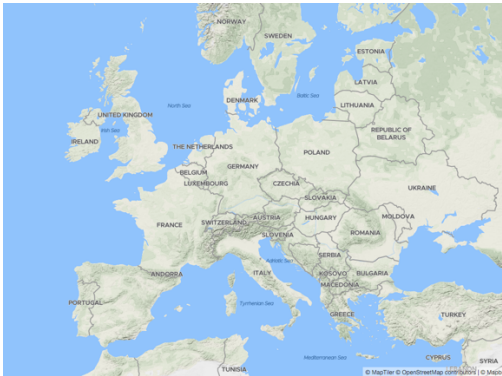

## Mapbox Line Layer Schema



## Mapbox Point Layer Schema

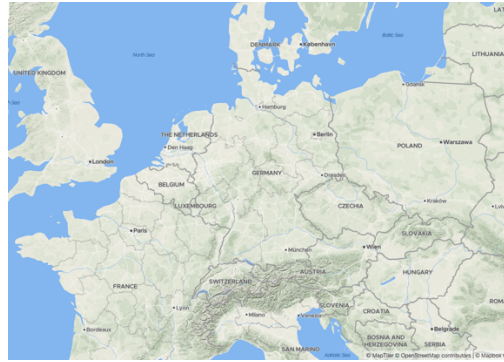


Appendix V: Comparison of Finished Map Styles

ZL	Terrain		Grey	
z0				
				
z1				
				
z2				
				
z3				
				



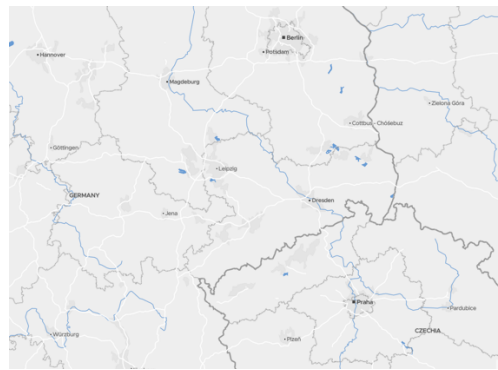
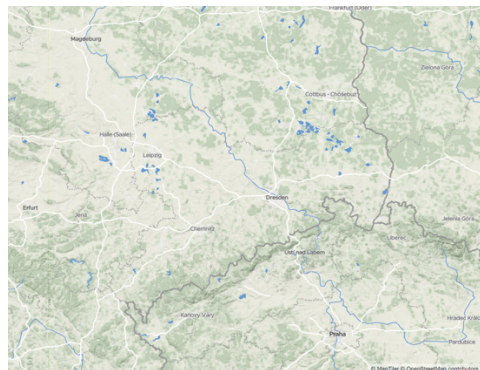
z4



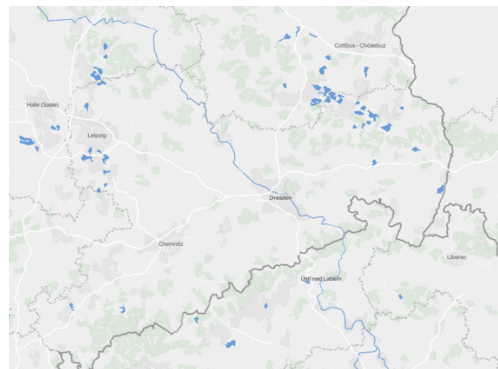
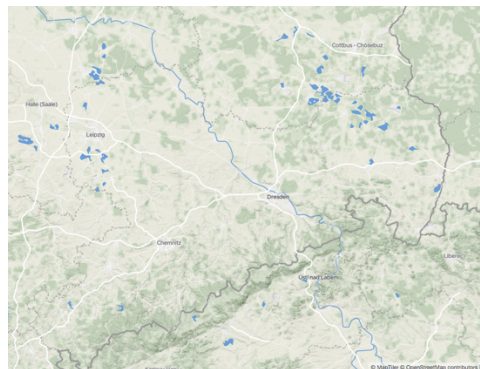
z5



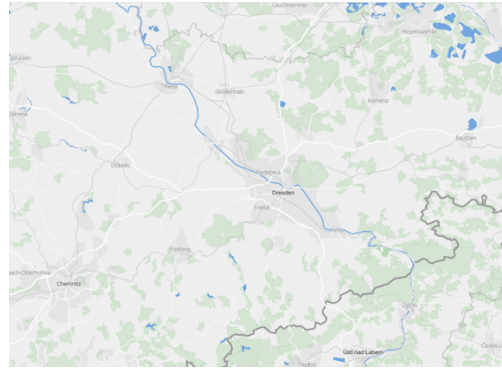
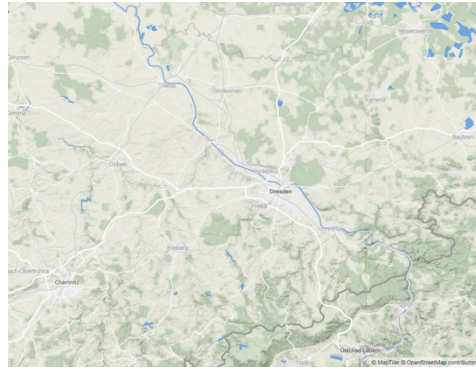
z6



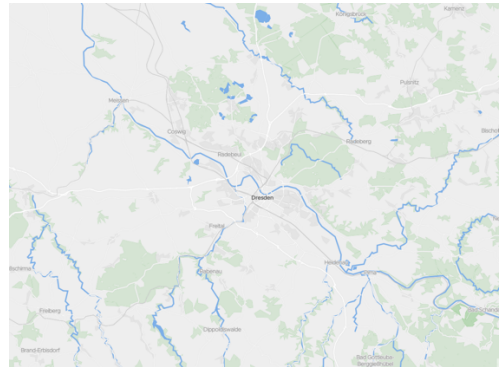
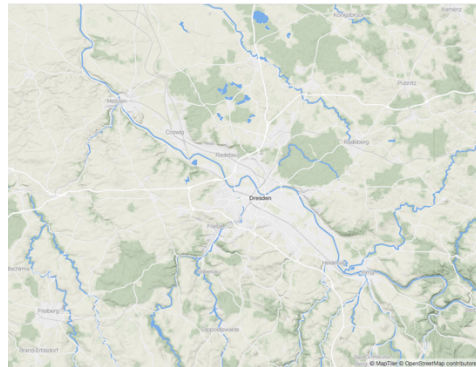
z7



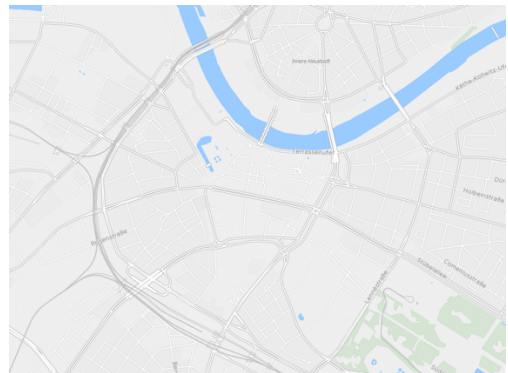
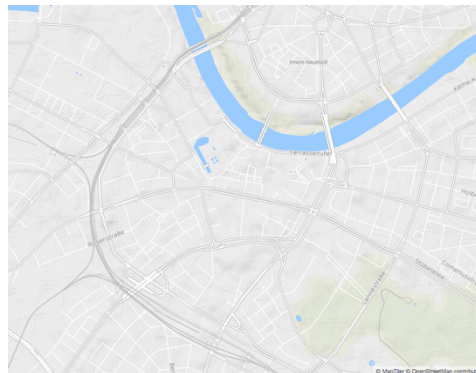
z8



z9

z  
10z  
11



z  
12z  
13z  
14z  
15

**z**  
**16**

