Cartography M.Sc.

Master thesis

MapColPal – a color palette generation and testing tool for thematic maps

Valerian Lange

2022

UNIVERSITY OF TWENTE.
ITC

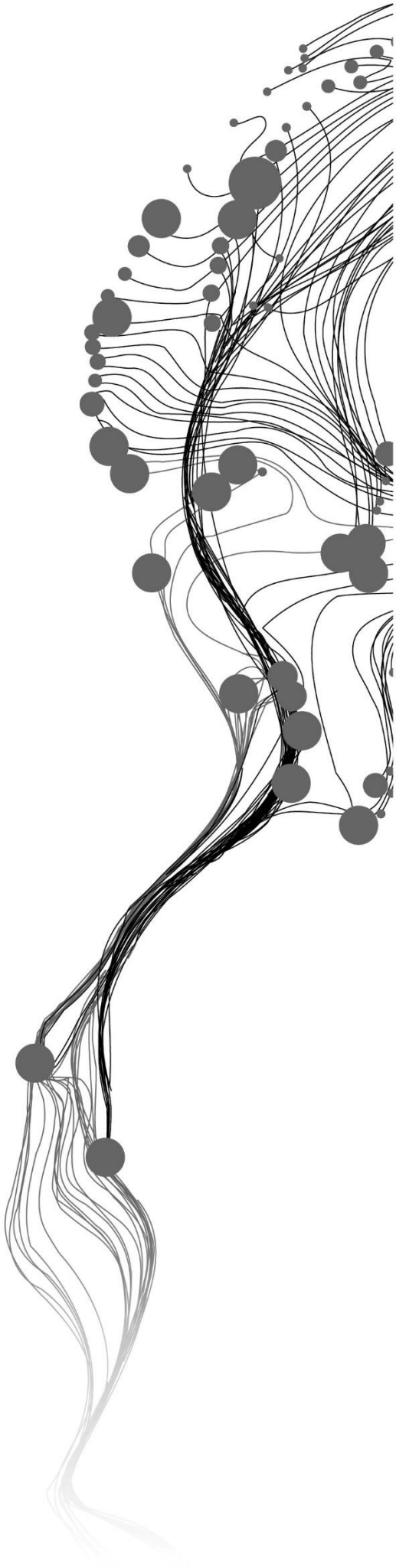
Statement of Authorship

Herewith I declare that I am the sole author of the submitted Master's thesis entitled:

"MapColPal – a color palette generation and testing tool for thematic maps"

I have fully referenced the ideas and work of others, whether published or unpublished. Literal or analogous citations are clearly marked as such.

Vienna, September 2022 Valerian Lange



MapColPal – a color palette generation and testing tool for thematic maps

VALERIAN LANGE
Enschede, The Netherlands, September 2022

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Cartography M.Sc.

SUPERVISORS:
Dr. P. Raposo

THESIS ASSESSMENT BOARD:
Dr. rer. nat. F. Mocnik
M.Sc. E. P. Bogucka (Reviewer, TU Munich)

MapColPal

a color palette generation and testing tool for thematic maps

Valerian Lange

thesis for the M.Sc. Cartography
September 20, 2022

Abstract

Color is a crucial part of cartographic visualization while simultaneously posing many challenges to the cartographer working on it. Creating and testing self-made color palettes for maps instead of relying on standard palettes requires manual effort, time, and expertise. Tools to aid in the cartographic design process with a limited scope and high depth became known as *cartographic brewers* Brewer (2003), with ColorBrewer (Harrower & Brewer, 2003) as one of the most influential examples among them. ColorBrewer helps to work with and understand properties of color palettes more easily. And yet, there are limitations to it: It features only a selection of pre-created color palettes for one map layer at a time and presents these palettes only applied to a choropleth map. What could a tool look like which improves on previous applications like ColorBrewer to assist cartographers in choosing color palettes for thematic maps?

In this thesis, MapColPal, a web-based color palette generation and testing tool for thematic maps, was designed, built, and evaluated. It provides a new take on the old problem of selecting colors for maps in a way suiting the data, human perception, and aesthetic preferences. MapColPal tackles this problem by deriving color palettes in a structured way from a shared set of seed colors, visualizing each update immediately, as well as offering user interaction and palette testing at all steps along the process. It builds on previous tools like ColorBrewer and combines their ideas with insight from recent literature and modern technology.

keywords: color palette generation, color palette testing, thematic cartography, cartographic design, requirements engineering, prototyping, web development

Contents

Contents 10

List of Figures 12

List of Tables 13

Glossary with Acronyms 14

1 Introduction 15
- 1.1 Research objective . . . 16
- 1.2 Research questions . . . 16
- 1.3 Thesis outline . . . 17

2 Background on color science 19
- 2.1 Perception of color . . . 19
- 2.2 Color use and harmony . . . 26
- 2.3 Color for maps and simultaneous contrast . . . 28
- 2.4 Digital cartography and web map applications . . . 30

3 Related work 32
- 3.1 ColorBrewer . . . 36
- 3.2 Colorgorical . . . 36
- 3.3 Chroma.js color palette helper . . . 37

4 Methodology 38
- 4.1 Requirement engineering . . . 38
- 4.2 Prototyping . . . 40
- 4.3 Heuristic evaluation . . . 41

5 Implementation 43
- 5.1 Identified criteria for thematic map color palettes . . . 43
- 5.2 Identified criteria for cartographic color palette tools . . . 44
- 5.3 Requirements . . . 45
- 5.4 Iterative prototyping process . . . 49

6 Results 56
- 6.1 Completed proof of concept . . . 56
- 6.2 Requirements check . . . 62
- 6.3 Heuristic evaluation . . . 64

6.4 Sample results 67

7 Conclusion and outlook **70**

References **73**

A Heuristics form **79**

B Screenshots of intermediate prototypes **87**
B.1 Wireframe 87
B.2 Technical capability test 90
B.3 First coded user interface prototype 90
B.4 Proof of concept implementation - First merge 91
B.5 Proof of concept implementation - Third merge 91
B.6 Proof of concept implementation - Fifth merge 93
B.7 Proof of concept implementation - Seventh merge 96

C Proof of concept - Responsive views **99**

D Screenshots of usage scenarios **102**
D.1 Scenario 1 - sequential palette generation 102
D.2 Scenario 2 - diverging palette generation 103
D.3 Scenario 3 - qualitative palette generation 104

List of Figures

2.1 Color gradient with varying hue and constant value and saturation in HSV using the sRGB color space . . . 26
2.2 Color gradient with varying hue and constant lightness and chroma in Oklab color space . . . 26

5.1 Wireframe . . . 49
5.2 Technical capability test . . . 50
5.3 First coded user interface prototype . . . 51
5.4 Implementation - First merge . . . 52
5.5 Implementation - Third merge . . . 53
5.6 Implementation - Fifth merge . . . 54
5.7 Implementation - Seventh merge . . . 54

6.1 Proof of concept - Main view . . . 56
6.2 Proof of concept - Layers panel . . . 59
6.3 Proof of concept - Test panel . . . 60
6.4 Proof of concept - Export panel . . . 61
6.5 Proof of concept - Tutorial modal . . . 61
6.6 Proof of concept - About modal . . . 62
6.7 Proof of concept - Success alert . . . 62
6.8 Proof of concept - Error alert . . . 62
6.9 Proof of concept - Export alert . . . 62
6.10 Scenario 1 - MapColPal input colors . . . 67
6.11 Scenario 1 - Sequential MapColPal palette . . . 67
6.12 Scenario 1 - ColorBrewer YlGnBu palette . . . 68
6.13 Scenario 2 - Diverging MapColPal palette . . . 68
6.14 Scenario 2 - ColorBrewer PiYG palette . . . 68
6.15 Scenario 3 - MapColPal input colors . . . 69
6.16 Scenario 3 - Qualitative MapColPal palette . . . 69
6.17 Scenario 3 - ColorBrewer Dark2 palette . . . 69

List of Tables

1.1 Relation between thesis outline and research questions 18

2.1 Estimated occurrences of color vision deficiencies (in %) 22
2.2 Hierarchy of scales 26

3.1 Overview of related work 33

Glossary with Acronyms

CIE	International Commission on Illumination
CVD	color vision deficiency
L cone	long-wavelength sensitive cone
M cone	middle-wavelength sensitive cone
S cone	short-wavelength sensitive cone
GIS	geographic information system
GUI	graphical user interface
JND	just-noticeable difference
RO	research objective
RQ	research question

CHAPTER 1

Introduction

> More delicate than the historians' are the map-makers' colors. (Bishop, 2011, from the poem *The Map*, first published in *North & South* 1946)

Color is crucial within cartography: it is part of multiple visual variables (Bertin, 2011, p. 42), it is an aesthetic element (Brown & Feringa, 2003, p. 127), and it is included in most textbooks on cartography (e.g., Brown & Feringa, 2003; Brewer, 2016). And yet, suitable color use in a map is challenging. The colors should be aesthetically pleasing to make the map attractive and enjoyable to read, but also need to be fitting to the map's theme and also match the kind of data that is displayed. Otherwise, an ill-fitted color palette may make the map unappealing and, even worse, distort the meaning of the data, instead of facilitating comprehension. Additionally, colors that work well for one map might not be suitable for another. A typical mistake is usage of the same color palette for every mapping need (Harrower & Brewer, 2003, p. 27).

When in the process of finding a suitable color palette for a given map, cartographers can either choose from standards supplied by the software they intend to use, for example a geographic information system (GIS) may offer a selection built-in, or they can rely on palettes crafted and tested by experts. In cartography, the most popular example of this is likely the color schemes provided by the ColorBrewer application (Harrower & Brewer, 2003). Using one of these pre-defined color palettes leaves the cartographer with a limited, finite set of options. This, in turn, not only limits the cartographer's control and agency over the map design, it can lead to data-distortion as a result, severely impacting the quality of the resulting map. Moreover, existing tools like ColorBrewer are not designed to be used effectively in situations where specific colors are required, e.g. a brand color (Smart, Wu, & Szafir, 2020, p. 1215). Of course, there is also the possibility to adapt an existing palette or craft a completely new one. This process of creating and testing such a self-made color palette instead of relying on a standard choice requires manual effort, time, and expertise (K. Lu et al., 2021, p. 475). Most textbooks and tools to date focus on providing advice for choosing and applying a pre-defined color palette (see chapter 2 for an overview). Also, it is possible to do this prototyping of a new palette within a GIS, but such systems offer little help in this demanding procedure.

Therefore, the need for a current review of research on color palettes within cartography and a summary of this information was identified, to provide cartogra-

phers with the necessary knowledge to be able to craft their own color schemes. Furthermore, to provide aid in the creation process, generation algorithms, which are already utilized in different disciplines (see chapter 3), can be adapted to fit the needs of cartography and data visualization, like providing a set of related color palettes for visualizations on the same topic that shall be displayed together, considering a basemap's colors for the palette generation, and taking multiple map layers into account. For example, in the case of a point symbol map overlaid on top of a choropleth map. Utilizing a generation algorithm provides creative inspiration and a base for guaranteed consideration of color design principles and can lead to outcomes that are preferred to standard palettes in direct comparison (Gramazio, Laidlaw, & Schloss, 2017, p. 529). A fitted generation algorithm can then be combined with the summarized advice and a palette choosing and testing interface similar to ColorBrewer. This will be further improved to also allow for different geometry types, multiple map scales, and color vision deficiency simulation, to create a unique new tool for cartographic color palette generation and testing that will be able to provide the cartographer with fitting palettes for varied map designs.

1.1 Research objective

The general research objective (RO) of this thesis is to **design, build, and evaluate a tool to assist cartographers in choosing suitable color palettes for thematic maps**.

For the cartographer to be able to choose a color scheme well, the tool combines relevant information on color palette design with a palette generation algorithm and a testing environment to visualize and assess the palette in relevant situations without the need to export a potential palette to other applications. The tool is focused on web maps to be displayed on screen displays like computer monitors and smartphones. In terms of what combinations of map layers are supported, the proof of concept shall support a basemap overlaid with a choropleth map layer which can in turn be overlaid with a point symbol map layer. For map scales, a city-scale (Web Mercator zoom level 11, rounded scale 1:300000) and a country-scale (Web Mercator zoom level 4, rounded scale 1:37000000) view shall be provided.

1.2 Research questions

To reach the research objective, the following research questions (RQs) were defined:

RQ 1 What criteria are necessary to decide whether to use a color palette for a thematic map considering not only choropleth maps, but also proportional point symbol maps and multi-layered combinations?

RQ 2 What color palette generation and testing tools exist already? How can a new tool improve upon the existing ones?

RQ 3 What requirements exist for a tool implementing these criteria and improving upon the existing tools?

RQ 4 How can these requirements be implemented in a proof of concept?

RQ 5 Does the proof of concept fulfill the requirements set before?

1.3 Thesis outline

As the intended research objective is a prototype of a tool, the general philosophy and style of work will follow the system design type of research. System design is defined as research "[...] where the researcher designs a system (database, visualization, modelling ...) and shows that it is somehow 'better' than previous designs; this includes design of algorithms and methods" (Rossiter, 2018a, 35). Rossiter (2018a, p. 64) names important parts of a thesis using this mode of research: Firstly, establishing the need for a design. This gets explicitly addressed in chapter 1 and is then built upon in chapter 2 and 3. Secondly, inspecting existing designs to determine their deficiencies. This is being handled in chapter 3 in detail. Thirdly, proposing a design and specifying its innovations. This is being done in chapter 5. And lastly, showing that the design fulfills its promises. This is the content of chapter 6 before a general conclusion is taken in chapter 7.

To handle each of these aspects and answer the research questions accompanying each chapter and in the end reach the research objective, scientific methods need to be chosen to work on this in a controlled and comprehensible manner. As research questions 1 and 2 deal with existing knowledge and its synopsis, a literature review was identified as the method to answer these questions. Research question 3 deals with the theoretical concept for the proof of concept, or in other words, the design to be implemented as well as its innovations. To define these theoretical aspects, requirements engineering was chosen as a fitting method to answer this research question. The fourth research question deals with the realization of the requirements identified in RQ 3 and uses prototyping combined with additional considerations for the software developing to implement these.

Introduction. Introduces the topic of research and the research objective.

Background on color science. This chapter builds the knowledge foundation upon which the first research question will be answered during implementation.

Related work. This chapter reviews existing tools and calculations for color palette generation and testing.

Methodology. This chapter describes the approach chosen to fulfill the research objective.

Implementation. This chapter describes the process of creating the requirements for the proof of concept as well as the actual proof of concept itself.

Results. In this chapter the final proof of concept is presented and evaluated.

Conclusion and outlook. This chapter discusses how far the research objective was achieved and how to proceed from here.

The structure of this thesis was chosen in relation to the research questions. Table 1.1 lists which research questions are addressed in which chapter.

Table 1.1: Relation between thesis outline and research questions

Chapter	Relevant research question(s)
Introduction	RO
Background	RQ 1
Related work	RQ 2
Methodology	RO
Implementation	RQ 1, RQ 2, RQ 3, RQ 4
Results	RQ 5
Conclusion and outlook	RO

CHAPTER 2

Background on color science

To be able to answer the first research question and identify relevant criteria for color palette usage in thematic maps, an overview of the current body of knowledge on this topic is summarized from cartography and relevant other disciplines in this chapter. It is structured from general to specific and follows the parts of the research question: First, the necessary foundations for discussing color and color palette design are laid out before moving to the specifics for thematic maps. Then, the more implementation-focused aspects of web map applications are considered including user interface design for such applications.

2.1 Perception of color

The most vital term to define for this research is the concept of color. And as this research does not feature a solely artistic view on color but wants to leverage the concept in the context of data visualization, the important aspects are color appearance and perceived color. One possible definition is stated by the International Commission on Illumination (CIE), which are well-known for the definition of multiple color spaces (Fairchild, 2013, p. 79-83): In the International Lighting Vocabulary, they define perceived, or also perceptual, color as a "characteristic of visual perception that can be described by attributes of hue, brightness (or lightness) and colourfulness (or saturation or chroma)" (CIE, 2020). Thus, color is seen as a part of visual perception. It also relies on the terms hue, brightness, lightness, colorfulness, saturation, and chroma, which will be regarded in the following. Fairchild (2013) also emphasizes that while color can be difficult to define, these attributes upon which its definition is relying are easier to define more precisely and are also exceedingly important for color appearance modeling (p. 88).

To start with visual perception, while it relies on the electromagnetic spectrum of light, it is not proper to state that a specific wavelength within this spectrum, or a specific object, are a given color. It is more fitting to state that those stimuli are perceived to be of a specific color when viewed under a given set of conditions (Fairchild, 2013, p. xix). Human visual perception is influenced by rods and cones, the two types of retinal photoreceptors. They do not always differ in their shape, which the names are derived from, but they have an important difference in visual function: Rods al-

low for vision when there is low luminance, or in other words only a small amount of light received by the eyes, while cones allow for vision in high luminance situations, and both are active in intermediate levels of luminance. When only the rods are providing vision, it is called *scotopic vision*. Similarly, when only the cones are active, it is called *photopic vision*, and *mesopic vision* is used for the intermediate situations where both are supplying vision (Fairchild, 2013, p. 8). While only one kind of rod receptor exists, there are three types of cone receptors with partly overlapping areas of spectral responsivity and peak responsivities at different points within the visual spectrum. They are therefore referred to as *long-wavelength* (L cone), *middle-wavelength* (M cone) and *short-wavelength* (S cone) *sensitive cones*. Color vision is achieved through the three types of cones. Rods are incapable of color vision. Cones utilize three kinds of pigment for sensing these different wavelengths, therefore vision including all three types of pigment is known as *trichromatic vision* (Bleicher, 2012, p. 11). Two modes of color vision can be discerned, *light-color perception*, visual effects produced by homogenous, non-varying light which are attributed to the interaction of the light with our eyes, and *object-color perception*, where perceived light is intercepted by an object which thereby appears to possess a color. The latter can only occur when more than one object is present (Evans, 1964, p. 1468). The overlap in responsivity between the kinds of cones differs from physical imaging systems, which are built with non-overlapping areas of responsivities for practical reasons, and this difference is part of the reason why precise color reproduction is difficult to realize (Fairchild, 2013, p. 9).

The different qualities of perception that together define a certain perceived color are also relevant for the definition of color. *Hue* is considered an "attribute of a visual perception according to which an area appears to be similar to one of the colours red, yellow, green, and blue, or to a combination of adjacent pairs of these colours considered in a closed ring" (CIE, 2020). This definition also introduces the concept of hues ordered in a circle, which is also known as a color wheel. A color wheel is a common element in color theory to describe color combinations geometrically and is also used in user interfaces to visualize or choose a color for an action (Tan, Echevarria, & Gingold, 2018, p. 1). Hue is also used to discern perceived colors into *chromatic colors*, which possess hue, and *achromatic colors*, those without hue (CIE, 2020). *Brightness* is defined as an "attribute of a visual perception according to which an area appears to emit, transmit or reflect, more or less light" (CIE, 2020) and *lightness* builds on this as it is the "brightness of an area judged relative to the brightness of a similarly illuminated area that appears to be white or highly transmitting" (CIE, 2020). As lightness needs a reference area for this relative measurement, it is restricted to related colors. These are defined as belonging to an area relative to other colors as opposed to unrelated colors which are perceived as not belonging or as being isolated from other colors (CIE, 2020). *Colorfulness* is an "attribute of a visual perception according to which the perceived colour of an area appears to be more or less chromatic" (CIE, 2020). *Saturation* in turn is the "colourfulness of an area judged in proportion to its brightness" (CIE, 2020) and lastly *chroma* is the "colourfulness of an area judged as a proportion of the brightness of a similarly illuminated area that appears grey, white or highly transmitting" (CIE, 2020). So while colorfulness is a property of its own, saturation relates to the brightness of an area as well while chroma relies on the comparison to a reference area along the lines of lightness.

A perceived color is never described as e.g. greenish and reddish at the same time. This led to the foundation of *opponent colors theory*, which states that some hues are connected to the same pigments and receptors for processing within the human

body and can therefore not be perceived at the same time. Opponent colors theory can be used to explain visual afterimages (Bleicher, 2012, p. 11), the loss in distinction for specific hue pairs in color vision deficiencies, and observations made regarding simultaneous contrast, where objects on a green background appear more red (Fairchild, 2013, p. 19). This theory further developed in the middle of the last century, and red-green and yellow-blue opponent mechanisms are important for all color appearance models (Fairchild, 2013, pp. 20–21).

Acuity

Visual acuity is the eye's ability to see detail; the sharpness of what is being visually perceived (Holtzschue, 2017, p. 234). It is varying per person, over the field of view, for different amounts of light energy received and also for different wavelengths of the visible spectrum (CIE, 2020). While technically the amount and wavelength of a light perception can be measured, the human eye perceives the spectrum of light and the colors connected with this perception not as separate, but as one continuous flow where one color blends into the next (Holtzschue, 2017, p. 44). The point at which two stimuli are just barely perceivable as being different is known as a *just-noticeable difference* (JND) (Fairchild, 2013, p. 41). It is an important concept for defining palette colors for visualization because colors for two different classes should be at least one JND apart, so that the classes are distinguishable. The JND can be different for a targeted object depending on the context, so it cannot be simply statically derived per color and object size, but also distance to other objects and their composition needs to be considered (M. Lu et al., 2022, p. 718). In terms of hue, there are more perceivable variations in reds and blues and less in yellows and greens (Kraak, Roth, Ricker, Kagawa, & Le Sourd, 2020, p. 42).

Preference

If accurate representation of data is not the only goal of a visualization, choosing colors which are appealing to the target audience becomes a relevant factor in choosing palette colors as well. To find an objective answer to the question 'What colors are preferential to a certain group of people?' is difficult though. Generally speaking, previous research found that a majority of test subjects, in humans as well as animals, favor blue and dislike yellow hues (McManus, Jones, & Cottrell, 1981, p. 665). When narrowing down the group of people, the differences in preference get less clear. While cultural biases, and learned responses as part of a group, seem likely to exist in color preference too, there is little documentation and explanation for them (Fairchild, 2013, pp. 361–362). The range of difference in individual color preference within a culture, and variations because of the depicted content of an image, are more apparent than the difference between cultural mean preference levels, and exceed their range (Fernandez, Fairchild, & Braun, 2005, p. 104). Therefore, if the general preference of blue hues is not specific enough for a given application, studies with the actual visualization, to account for variation in preference due to image content, and members of the actual target group, to account for individual preference, are required to achieve meaningful preference measurements.

Color vision deficiencies

Nobody has exactly the same optical perception and visual functionality also changes within a lifetime. If someone misses certain qualities of color vision, they are pop-

ularly known as 'colorblind', but as in the majority of cases there is only a reduction in color discriminability and not a complete loss of color vision, more appropriate terms are "color defective" (Hunt & Pointer, 2011, p. 13) or "color vision deficient" (Fairchild, 2013, p. 32). The terms *color vision deficient* and *color vision deficiency* CVD will be used for this thesis. Color vision can change within a lifetime, for example related to diseases, but inherited color vision deficiencies account for most cases. Because the genes for photopigments are present on the X chromosome, persons with only one X chromosome are more likely to be born color vision deficient. Hunt and Pointer (2011) combined statistics for the occurrence of color vision deficiencies in people of European descent from multiple surveys and arrives at around 4% occurrence for the total population, which splits up into 8% for persons with one X chromosome and 0.4% for persons with two X chromosomes (p. 14).

Table 2.1: Estimated occurrences of color vision deficiencies in populations of European descent (in %)

Type	**One X chromosome**	**Two X chromosomes**
Protanopia	1.0	0.02
Deuteranopia	1.1	0.01
Tritanopia	0.002	0.001
Cone monochromatism	0 (very rare)	0 (very rare)
Rod monochromatism	0.003	0.002
Protanomaly	1.0	0.02
Deuteranomaly	4.9	0.38
Tritanomaly	0 (rare)	0 (rare)
Total	8.0	0.4

Data by Hunt & Pointer, 2011, p. 14

As can be seen in table 2.1, the most common kind of color vision deficiency by far is deuteranomaly, followed by deuteranopia, protanopia and protanomaly. Deuteranomaly and protanomaly are kinds of anomalous trichromacy, so the color perception is changed, but still trichromatic. With deuteranomaly, affected persons experience reduced discrimination of the reddish and greenish contents of colors. Protanomaly leads to a similar experience, but reddish colors also appear to be more dim than usual (Hunt & Pointer, 2011, pp. 13–14;Fairchild, 2013, pp. 32–33). With deuteranopia and protanopia on the other hand, certain photopigments are missing completely which leads to a dichromatic color vision. Deuteranopes, persons with deuteranopia, as well as protanopes, persons with protanopia, are missing a red-green opponent mechanism completely and therefore cannot distinguish reddish and greenish hues at all. For protanopes reddish colors appear dimmer, similar to persons with protanomaly (Hunt & Pointer, 2011, pp. 13–14;Fairchild, 2013, pp. 32–33).

A person with average color vision cannot experience what the world looks like to a person with a color vision deficiency. It is possible though to visualize the hues that become indistinguishable due to the loss or reduction in color differentiation (Fairchild, 2013, p. 33). As color vision deficiencies are quite common, as was illustrated before, people about to make critical color appearance or color matching decisions should be screened and briefed beforehand to enable them to make informed decisions. Prevalent tests include pseudoisochromatic plates, like the Ishihara test,

and the Farnsworth-Munsell 100-Hue test (Fairchild, 2013, p. 36). To allow persons with a color vision deficiency to perceive a visualization as intended, it is recommended to avoid colors not suitable for the specific CVD in question and in general keep the amount of classes to the least possible number and use strong contrasts (Weninger, 2015, pp. 109–110).

Color models

To facilitate working with colors, standardized descriptions of colors are required. *Color models* allow for this by specifying a space within a coordinate system in which each possible color represents a point (Silva, Sousa Santos, & Madeira, 2011, p. 320). A *color space* on the other hand can be defined as a "geometric representation of colour in space" (CIE, 2020), so it is the space of possible colors that a color model is applied to. Therefore, a color space is bound to the color model it is based on, while a color model can be represented within multiple possible color spaces. Various color models and color spaces exist with their own properties and can be more or less appropriate for a certain task, thus relevant ones to the domain of this research are introduced in this section.

Color models can be either device dependent or device independent (Silva et al., 2011, pp. 320–321), and they are closely related to the notions of color order systems and color appearance models. The former use basic colorimetry to specify visual stimuli and collect those within a color order system (Fairchild, 2013, p. 97). The latter on the other hand are defined as "[…] any [models] that [include] predictors of at least the relative color appearance attributes of lightness, chroma, and hue" (Fairchild, 2013, p. 200). Color order systems supply data and specification techniques, but no mathematical framework to allow for extension of the system. So color order systems can be a relevant precursor to color appearance models, but no replacement (Fairchild, 2013, p. 97). Because of these characteristics, color appearance models are better suited to create color palettes based on human perception in.

A color space is called *uniform* if it is a "colour space in which equal distances are intended to represent threshold or suprathreshold perceived colour differences of equal size" (CIE, 2020). Out of three colors A, B, and C, if color B is twice as chromatic as color A and color C is four times as chromatic as color A, the geometric distance between A and B should be the same as between B and C in a perceptually uniform color space. A perceptually uniform color space relies on its context and therefore standard illuminants are defined to generalize this context. Illuminants build upon the concept of a light source, an actual emitter of visible energy, and abstract that into standardized tables of spectral power distribution values typical for a certain light source. Well-known examples include the CIE illuminants A (for incandescent light), D65 (for daylight) and F2 (for fluorescent light) (Fairchild, 2013, p. 59).

RGB

The RGB color model, named after its red, green, and blue primary colors, is an additive color model, which means varying intensities of the three primary colors get added together to represent the possible colors within the color model. It is a device-dependent color model, because the same intensities of the R, G, and B components can appear differently on different devices. While the RGB color model is based in human perception, it is not a perceptually uniform color model (Zeileis et al., 2020, p. 8).

When each of the primary colors gets mapped to an axis in space, RGB color space can be represented as a three-dimensional cube. Working with amounts of red, green, and blue components per color and orienting in a cubical space is considered unintuitive and impractical for color selection (Brown & Feringa, 2003, p. 39), which are the major disadvantages of the RGB color model apart from not being perceptually uniform.

The gamma-corrected standard RGB, sRGB, is the currently defined standard color space for the World Wide Web (CIE, 2020). This implementation of the RGB color model adopts gamma correction to standardize the perception of colors represented in sRGB. Otherwise, it shares the same general properties with the RGB model (Zeileis et al., 2020, p. 8).

HSL and HSV

HSV and HSL are transformations of one of the RGB color spaces, that allow to specify RGB colors with different components with the goal of making them more intuitive to work with. The HSL color model is a transformation using hue, saturation, and lightness components. HSV on the other hand utilizes hue, saturation, and value components (Zeileis et al., 2020, pp. 8–9).

They are used in applications which use RGB, but a more intuitive way of color specification is wanted, for example in a color picker within the application. HSV and HSL still share the same properties with RGB though, so they are also not perceptually uniform, therefore their simple and intuitive design, and the naming of the color components, can be misleading. Because of this, their use is discouraged by experts, but as of now, they are still in use on the internet and in GIS applications (Brown & Feringa, 2003, p. 43).

Munsell

The Munsell system is a color order system which is especially used in the United States of America, and was developed by the artist Albert H. Munsell in the early twentieth century to aid education. In this system color appearance is specified by three variables: hue, value and chroma. The definition of these three values match the definitions currently used by the CIE, which are given above, with Munsell value referring to lightness (Fairchild, 2013, p. 99).

The Munsell system is still accepted as a standard, but as a color order system, the Munsell system is missing a mathematical framework as well as being not wholly perceptually uniform. Also, the three-dimensional color solid that the variables are applied to is asymmetric, it is neither fully a cylinder nor shaped as a double cone (Brown & Feringa, 2003, pp. 32–34).

Many modern color appearance models share similarities with the Munsell system though, so it is an important historical predecessor to them (Brown & Feringa, 2003, p. 37).

CIE color spaces

The CIE introduced the XYZ color system in 1931, which is still a standard for color description. It is based on the human optical system and offers the advantage of allowing spectrophotometer measurements to be plotted onto a two-dimensional graph that can be derived from its color space, a so-called *chromaticity diagram* (Brown &

Feringa, 2003, p. 44). One unit within the CIELAB color space is approximately corresponding to one JND (Szafir, 2018, p. 393).

The system is based on the same primaries as RGB, but uses artificial primaries which correspond to one of the types of retinal cone each respectively. All of this has certain advantages for color science, but as the luminance is only represented as a separate number in the chromaticity diagram, the idea to derive a three-dimensional color space from the XYZ color system arose (Brown & Feringa, 2003, pp. 44–46). Two such color spaces were created by the CIE itself, CIELAB and CIELUV. They made chromaticity diagrams mostly obsolete, as these extend tristimulus colorimetry into "[...] three-dimensional spaces with dimensions that approximately correlate with the perceived lightness, chroma, and hue of a stimulus" (Fairchild, 2013, p. 79). They are well-known and especially CIELAB is an established standard among color appearance models in use today (Brown & Feringa, 2003, pp. 47–48, Fairchild, 2013, p.210). These color spaces allow for uniform color difference measurements (Fairchild, 2013, p. 79). Yet, CIELAB also has its drawbacks, for example an incorrect hue shift in the blue hue range. Therefore, it is recommended to use it merely as a benchmark to compare more sophisticated color appearance models to (Fairchild, 2013, pp.209–210).

CIELAB and CIELUV can also be transformed into a cylindrical form, CIELCH, also known as HCL(Zeileis et al., 2020, p. 9), which uses lightness, chroma and hue components. The respective versions for CIELAB and CIELUV are also referred to as $CIELCH_{ab}$ and $CIELCH_{uv}$ (Zhou & Hansen, 2016, p. 2059).

HSLuv

One newer, in 2012, developed color space is the HSLuv space, which was developed as an alternative to HSL in color pickers and is based on $CIELCH_{uv}$ (Boronine, 2012). It specifies colors with hue, saturation, and lightness components. The hue and lightness components are the same as in $CIELCH_{uv}$, while the saturation component is defined as in $CIELCH_{uv}$, but rescaled relatively to the most saturated sRGB color possible with the same hue and lightness components. The hue component shares the same hue distortion problem with the CIE color spaces, due to the definition of the saturation component, loses its perceptual uniformity and does not vary smoothly (Ottosson, 2021).

Oklab

Another recently developed color space is the Oklab color space. It is designed to be easy to use, perceptually uniform and allowing for even transitions between two colors. It assumes a D65 illuminant, which is also used in sRGB and therefore makes it well-suited for working with it on the internet (Ottosson, 2020).

Figures 2.1 and 2.2 show two color gradients with varying hue while keeping the other color components stable. Figure 2.1 uses the HSV color space for this, while figure 2.2 uses the Oklab color space for the transformation. When viewing these images digitally, figure 2.1 using HSV has shifts in apparent lightness even though the HSV value is constant. Oklab performs much better in comparison, which also serves as a general comparison between perceptually not-uniform and uniform color spaces.

A transformation into polar coordinates is possible and results in lightness, chroma, and hue components for color specification, similar to CIELAB and CIELCH (Ottos-

son, 2020). This transformation will be called Oklch in this thesis. Because of these properties, and the disadvantages of the other color spaces presented before, the Oklch color space was chosen as the primary color space for the implementation of the proof of concept.

Figure 2.1: Color gradient with varying hue and constant value and saturation in HSV using the sRGB color space. Source: Ottosson, 2020

Figure 2.2: Color gradient with varying hue and constant lightness and chroma in Oklab color space. Source: Ottosson, 2020

2.2 Color use and harmony

The general method to visualize data using color is to apply a color palette or *colormap* to the data, which is defined as "[...] a mapping from data values to colors that generates visual structures for the data" (Zhou & Hansen, 2016, p. 2051). Colormaps can be either discrete or continuous. A discrete colormap works with classified data or single values that are each assigned a color. Continuous colormaps define a series of colors along the data range and the colors for values in-between are interpolated. A continuous colormap is represented by a curve in a color space (Bujack et al., 2018, p. 926).

Table 2.2: Hierarchy of scales

Scale	**Properties**	**Operations**
Nominal	Naming	Counting
Ordinal	Order, unequal intervals	Ranking
Interval	Equal intervals, arbitrary zero	Addition, substraction
Ratio	Natural zero, equal intervals	Multiplication, division

Own table, adapted from Fairchild, 2013, p. 43

Decisions regarding color palette design, like this one between a continuous or discrete colormap, also depend on the kind of data and the resulting perceived scales to be displayed. For this, the hierarchy of scales, as depicted in figure 2.2 is important knowledge to base decisions on. For example, nominal data should not be encoded with a color palette that conveys the impression of an ordered succession to the reader, as the original data does not have this kind of order or rank to it. Note,

that the mathematical operations per level of scaling are adding to the possible operations of all scale types lower in the hierarchy (Fairchild, 2013, p. 43).

Bujack et al. (2018, p. 924) summarize the following design rules for perceptual colormaps from the literature:

- Order (intuitive, natural, easy to remember)
- Discriminative power (separation, sensitivity, just noticeable differences, distinct color levels, color space utilization/ exploitation, perceptual range / resolution, discriminability)
- Uniformity (equidistant differences, separation, associability, separability, linearity, equal values shall be mapped to equal colors)
- Smoothness (continuity, no boundaries, no Mach bands, low curvature no sharp bends)
- Equal visual importance
- Robust to vision deficiencies
- Robustness to contrast effects
- Robustness to shading on 3D surfaces Background sensitivity
- Device independence (do not leave the gamut)
- Aesthetically pleasing
- Intuitive / natural color choices
- Use different colormaps for different variables
- Separation of values into low, medium, and high
- Avoid rainbow
- Highlighting of prominent values

Bujack et al. (2018) highlight especially order, discriminative power and uniformity as important. This coincides with mentions of preservation of order (if existing in the data) and correctly conveying uniformity among values by Levkowitz (1996, p. 97) and Weninger (2015, pp. 109–110). Although papers and books containing advice on the scientific use of colormaps exist, usage of colormaps unfit for scientific data visualization, for example not perceptually uniform rainbow colormaps which therefore distort the perception of the visualized data, is still prevalent among scientists as well as the public (Crameri, Shephard, & Heron, 2020, p. 2). Thus, a need for further information on correct color use in data visualization can be identified.

To aid the comprehension of the data at hand and also to be aesthetically pleasing, colors in a colormap are chosen to be *harmonious*. Itten defines color harmony as

> [...] the craft of developing themes from systematic color relationships capable of serving as a basis for composition. Since it would be impossible to catalogue all combinations here, let us confine ourselves to developing some of the harmonic relationships. Color chords may be formed of two, three, four, or more tones. We shall refer to such chords as dyads, triads, tetrads etc. (1970, p. 72)

These systemic color relationships are also known as harmony templates. Especially hue templates, operating within a color wheel, are the most widely-used model for color harmony (O'Donovan, Agarwala, & Hertzmann, 2011, p. 2). Templates are seen as being equally harmonious among each other and to be independent of its rotational placement on the color wheel. Also, they are defined independently of the color space the wheel is placed in, so they include an element of uncertainty by definition and are often rather seen as a starting point than a strict rule (O'Donovan et al., 2011, p. 2). Itten (1973) proposes sets of two to six hues equidistant on a color wheel to be harmonious. Matsuda (1995) defined hue templates with sectors on a color wheel.O'Donovan et al. (2011, p. 2) evaluated these and templates used by color picker applications and did not find higher color preferences when templates were used. This study only examined general color themes and had no specific considerations for data visualization or cartography. At the same time, O'Donovan et al. (2011) note the trend of colors harmonizing when they have the same hue, similar saturation and contrasting lightness. Colors being harmonious with lightness and saturation de- or increasing systematically towards one end of the colormap is also mentioned by Weninger (2015, pp. 109–110). For this thesis, hues will be seen as *harmonious*, when they are equidistant on a color wheel that represents the polar hue component of a perceptually uniform color space. For example, the hue angles of the CIELch or OKLch color spaces can be used to map harmonious colors. Palette colors will be considered harmonious, if they have a systematic lightness and saturation (or chroma, depending on the color space used) shift and a visual hierarchy emerging from this value change that matches the order and scale of the data visualized.

2.3 Color for maps and simultaneous contrast

Cartography is defined as "[...] the art, science, and technology of making and using maps" (Kraak et al., 2020, p. V). When designing a color palette for map use, some phenomena need to be considered that are especially important in cartography.

One such phenomenon is *simultaneous contrast*. Simultaneous contrast is the effect that the perception of a color may be altered due to surrounding colors. Because in maps not every combination of neighboring data values is always known upfront, for example in automatically updated digital maps, simultaneous contrast is especially relevant to cartographic color design (Kraak et al., 2020, p. 42). Simultaneous contrast is related to the perception of transparency. A colored element nested within uniformly colored surroundings appears increasingly transparent the more similar the colors of the element and its surroundings are (Ekroll & Faul, 2013, p. 350).

Another important topic is the consideration of layered information for the color design of a map, as most maps consist of multiple map layers. A map layer describes a geographic dataset or an abstraction of one. A map layer can be either a *structure layer*, a group of features based on the same topological data structure and logically related, or a *thematic layer*, a group of features belonging to the same theme. The latter are not explicitly existing but can be derived from structure layers (Hoop, Oos-

terom, & Molenaar, 1993, p. 140). To consider the layered structure of the map for its color design, Brewer (2016, p. 37) mentions that a basemap is background information in a map, and therefore needs to be different enough in lightness compared to the main map layers to allow the main content to stand out against the basemap. A basemap is a map layer meant to provide context to the main information contained in a map (Brewer, 2016, p. 21). This also hints at related and helpful concepts: *visual hierarchy* and the *figure-ground relationship* within a visualization. Visual hierarchy is the order in which map elements are perceived and should be designed to ensure that important information is perceived first (Kraak et al., 2020, p. 49). Brewer (2016, p. 111) lists color lightness as one characteristic to use to establish visual hierarchy. Kraak et al. (2020, p. 42) recommend designing a map's visual hierarchy in grayscale with only lightness in mind first, before building up chroma and hue for emphasis of the most important features. The figure-ground relationship is a similar concept from psychology and when applied to a map the ground is background information, like basemaps and less important map layers, and should be less chromatic and lighter, while the figure is the more important layers and foreground information, and should be more chromatic and darker (Brown & Feringa, 2003, p. 139).

Visual variables

An important concept within cartography and related sciences is the notion of *visual variables* as first introduced by Bertin in 1967. If given a graphical mark with a fixed position on a two-dimensional plane, this mark can be varying in multiple modes which are called visual variables. Bertin defined these as size, value, texture, color, orientation, and shape (2011, p. 42). Brewer (2016) names eight visual variables: size, lightness, (pattern) spacing, saturation, hue, shape, orientation, and arrangement (p. 194). While Bertin (2011) includes color only in two variables, 'value' and 'color', which refer to lightness and hue according to the CIE definitions, Brewer (2016) includes 'lightness', 'saturation', and 'hue', which are named according to the CIE definitions. So in both approaches, color is represented in multiple visual variables, although the organization in the newer publication is more easy to work with from the perspective of color science and will be used for this thesis. In general, the concept of visual variables helps to conceptualize the different qualities a color palette has to fulfil in visualization and how to use these qualities correctly depending on the context: Hue can be used to differentiate qualitative data, lightness is better suited for visualizing rank-ordered phenomena, and saturation (or chroma) can help establish the visual hierarchy with more important information being more chromatic (Brown & Feringa, 2003, pp. 132–138).

Thematic cartography

Thematic cartography is the subsection of cartography focused on thematic maps and is the application domain for this research. A thematic map "[…] depicts the variation of one or sometimes several […] geographic phenomena, mapping spatial and attribute information together. Meeting the SDGs requires thematic mapping of indicator data. Thematic maps enable geographic imagination and spatial thinking, and often represent abstract or statistical concepts that cannot be observed directly" (Kraak et al., 2020, p. 58).

Different types of thematic map exist, e.g. graduated point symbol maps and choropleth maps, and utilize differing visual variables for depiction of information

(Kraak et al., 2020, pp. 58–59): A nominal map represents differences in mode or a binary value. A choropleth map shades enumeration units according to their respective attribute values. A proportional symbol map uses point symbols scaled according to their respective attribute values. A graduated point symbol map is similar, but uses fixed size intervals. And an isoline map represents a gradient of interpolated values between sampled attribute values. A map can contain multiple kinds of thematic map within it, as each map layer can be of one of these types.

2.4 Digital cartography and web map applications

Web mapping is a kind of multimedia mapping and "[...] is dynamic and interactive (most of the time) and makes use of or connects to many different types of media" (Muehlenhaus, 2014, p. 19). Web mapping is the technological context for the implementation of the proof of concept and also the target medium for many maps nowadays.

Digital maps as a target medium influence both the design of a map or tool to be developed and the color space to design a color palette in. As a goal of digital maps is consistency of colors when being viewed on a range of possible output devices, colors are necessary that are visible within the color space supported by the screen in question (Weninger, 2015, pp. 109–110).

User interface and usability

As this thesis is less concerned with the *usability* of existing tools and primarily focuses on the aspect of color palette generation and testing for thematic maps, this section just aims to lay down the core concepts of usability to make sure not to degrade the proposed tool from existing solutions.

Usability is defined as a product's ease of use. Usability is important for the usage of the product to lead to a successful and satisfactory result, which is the *user experience* (Kraak et al., 2020, p. 112). To ensure usability, a well-designed *user interface* is important, through which the user can interact with and manipulate elements on the screen. Graphical user interfaces (GUIs) with the possibility of immediate interaction are important in cartography because of its visual nature (Kraak et al., 2020, p. 94).

Shneiderman (1996, p. 337) lists the following core interaction concepts for user experience design:

1. **Overview**. Gain an overview of the entire collection.
2. **Zoom**. Zoom in on items of interest.
3. **Filter**. Filter out uninteresting items.
4. **Details-on-demand**. Select an item or group and get details when needed.
5. **Relate**. View relationships among items.
6. **History**. Keep a history of actions to support undo, replay, and progressive refinement.
7. **Extract**. Allow extraction of sub-collections and of the query parameters.

The proof of concept will be implemented considering these concepts as well as possible. Additionally, Brewer (2003, p. 161) coins the term *cartographic brewer* for a tool with a limited scope and high depth to aid in the cartographic design process and states the following characteristics of such a brewer:

1. A selection of choices are offered for a specific representation challenge (a brewer is not simply a general lesson in principles).
2. Choices are organized by a set of mapping principles made explicit to the user (the choices are not merely listed in an unstructured catalog).
3. All of the choices offered are potentially suitable for problems for which the tool is used (there are no extreme choices or straw men in a symbol set).
4. All choices can be examined as categories are explored, encouraging the user to learn about criteria for applying the existing variety of choices.
5. Choices are not software specific (solutions can be implemented in multiple mapping applications).
6. Only basic skills in the use of mapping software are needed to implement the representations offered (programming skills are not required).
7. Representations are further augmented with tips on their suitability, though the user making a quick selection can ignore these details.
8. Users are encouraged to be critical of the choices offered, evaluating them with a display that will reveal potential shortcomings.

While these influence the general design requirements for a tool that is to be deemed a brewer, all of these aspects influence the user interface design as well.

CHAPTER 3

Related work

This chapter summarizes the research done on existing tools and calculations to be able to answer the second research question. 42 publications were identified as relevant to this thesis in terms of practical implementation and the underlying concepts. These are listed in table 3.1 with a description of the content related to this thesis as well as the category of apparent evaluation method that was utilized.

From these publications, at least seven of them deal in some fashion with the ColorBrewer tool. ColorBrewer also gets mentioned multiple times in the literature reviewed for chapter 2 (see e.g. Brewer, 2016; Kraak et al., 2020; Muehlenhaus, 2014; Weninger, 2015). Therefore, ColorBrewer was identified as the most central tool to this field of research in existence so far and will be regarded first. Afterwards, further publications from table 3.1 will be examined in more detail, based on which are the most relevant to the research objective. A discussion of the works presented here follows in the beginning of chapter 5.

Table 3.1: Overview of related work

Publication	Relevant content	Apparent evaluation used
Aisch, 2013	Chroma.js Color Palette Helper.	None.
Aisch, 2018	Automatically check visualizations for non-colorblind safe colors.	None.
Aisch, 2019b	Chroma.js Color Palette Helper.	None.
Brewer, 1991	Model to predict simultaneous contrast.	Human subject experiments.
Brewer, 1997	Model to predict simultaneous contrast.	Human subject experiments.
Brewer, 2003	ColorBrewer: web tool for selecting color schemes for thematic maps.	Not described in this publication.
Bujack et al., 2018	Framework for assessment of continuous colormaps.	None.
Gardner, 2005	Design recommendations to improve ColorBrewer.	Human subject experiments.
Gramazio et al., 2017	Colorgorical: web tool for creating discriminable and aesthetically preferable categorical color palettes.	Theoretical evaluation, human subject experiments.
Gresh, 2008	Perceptual correction of continuous colormaps.	Theoretical evaluation.
Harrower & Brewer, 2003	ColorBrewer: web tool for selecting color schemes for thematic maps.	Not described in this publication.
Heer & Stone, 2012	Creation of a color naming model, usage for color palette evaluation, remarks for ColorBrewer.	None.
Hu et al., 2012	Harmonious color scheme generation with an interactive tool, using hue templates.	Human subject experiments.
Hu et al., 2013	Preferential color scheme generation with an interactive tool.	Human subject experiments.
Lindner & Süsstrunk, 2013	Color palette generation from words, using hue templates.	Human subject experiments.
K. Lu et al., 2021	Palettailor: color palette generation using three scoring functions and adapting to data or user preferences and filters, remarks for ColorBrewer and Colorgorical.	Human subject experiments.

Continued on next page

Table 3.1: Overview of related work - continued

Publication	Relevant content	Apparent evaluation used
Machado, Oliveira, & Fernandes, 2009	Equations for simulating color vision deficiencies.	Human subject experiments.
Meier, 1988	Computer-aided design for color in user interface design, using a knowledge base.	
Meier, Spalter, & Karelitz, 2004	Multiple interactive color palette tools, partly using hue templates / harmony rules.	None.
Nuñez, Anderton, & Renslow, 2018	Optimize continuous colormaps also regarding CVDs.	Theoretical evaluation.
O'Donovan et al., 2011	Evaluation of color datasets in regards to color compatibility, partly using hue templates.	Human subject experiments.
Peng & Chou, 2019	Color palette design from affect words / sentiment analysis.	None.
Petroff, 2021	Color sequence development, balancing aesthetics and accessibility, using a data-driven approach with machine-learning aesthetic-preference models combined with accessibility constraints that address color-vision deficiencies, grayscale printing, minimum contrast with the background, and color saliency.	Theoretical evaluation.
Post & Goode, 2020	Tool to help identify if color combinations are distinguishable for with and without CVDs, using confusion lines for CVD sight.	Human subject experiments.
Robertson & O'Callaghan, 1986	Generation of univariate and bivariate color sequences.	Theoretical evaluation.
Samson, 1985	Computer-aided design for color in cartography, using a knowledge base.	
Schloss & Palmer, 2011	Tests harmonious and preferred color combinations.	Human subject experiments.
Shamoi, Inoue, & Kawanaka, 2014	FHSI - fuzzificated HSI color space.	Theoretical evaluation.
Shugrina, Lu, & Diverdi, 2017	Playful Palette: interactive artistic color mixer.	Human subject experiments.

Continued on next page

Table 3.1: Overview of related work - continued

Publication	Relevant content	Apparent evaluation used
Smart et al., 2020	Color Crafting: generate color ramps using design mining and unsupervised clustering.	Human subject experiments.
Sochorová & Jamriška, 2021	Digitally augment analog/real life pigment color mixing, using Kubelka - Munk color model.	Theoretical evaluation.
Stone, 2012	Relevance of size of objects for color design, varying luminance easier to distinguish for small sizes.	None.
Stone, Szafir, & Setlur, 2014	noticeable difference model to describe just-noticeable differences (JNDs) and color difference as a function of size.	Human subject experiments.
Szafir, 2018	Color difference models size dependent and independent.	Human subject experiments.
Tan et al., 2018	Image decomposition and palette extraction, using hue templates, color harmonization.	Human subject experiments.
Tokumaru, Muranaka, & Imanishi, 2002	Color harmony and hue and tone templates.	Theoretical evaluation.
Waldin et al., 2019	Cuttlefish: Adapt color scheme in use to the current view and scale of the visualization.	
Wijffelaars, Vliegen, van Wijk, & van der Linden, 2008	Univariate lightness ordered/sequential palette generation.	Human subject experiments.
Yuan et al., 2021	InfoColorizer: interactive color palette advice/generation for infographics.	Human subject experiments.
Zeileis, Hornik, & Murrell, 2009	Deriving color palettes in HCL color space.	
Zeileis et al., 2020	colorspace: R package for working with and generating color palettes in the HCL color space.	
Zheng et al., 2022	Derive color palettes for categorical data visualization from images.	

3.1 ColorBrewer

ColorBrewer is a collection of color palettes to be used in thematic maps (and de facto used in general in data visualization (see e.g. K. Lu et al., 2021; Gramazio et al., 2017)) and a tool to visualize these online to help with selecting a palette from this collection. To this end, the tool provides the user with a selection interface to choose the number of classes and the type of data to be visualized as well as allowing for filtering the available palettes by further qualities like suitability for printing or for color vision deficient persons (Brewer, 2016, p. 205). The main user interface consists of a side panel on the left side with the adjustable controls and a choropleth map on the right side. The map visualizes a chosen color palette on example data that cannot be adjusted. The background color can be changed as well as borderlines and the transparency of the choropleth map overlay (*ColorBrewer source code*, 2021). The palettes are created by following conceptual arcs through a color space, although they were not actually plotted in one color space but used expert knowledge combining cyan, magenta, yellow and black (CMYK) color mixing as is used in printing as well as the Munsell color order system. The diverging color palettes arc over lighter colors in the middle in perceptual color space, the multi-hue sequential palettes change more in hue in the middle of the palette and more in lightness at the ends of the palette. The qualitative palettes, apart from the 'Paired' and 'Accent' palettes, keep saturation and lightness the same while contrasting in hues (Harrower & Brewer, 2003, pp. 30–31). For varying class numbers, ColorBrewer systematically derives palettes from shared sets of colors (Harrower & Brewer, 2003, pp. 36–37). ColorBrewer deviates in its user interface from the hierarchy of scales with nominal, ordinal, interval and ratio scales and instead simplifies to three types of color palette: qualitative, sequential and diverging. While the qualitative palettes are suited for nominal data and the sequential palettes are suited for ordinal, interval, and ratio data, the diverging palettes are a special case, as they are also meant to be used with ordinal, interval, and ratio-scale data, but with the difference of highlighting a critical value in the data, for example zero or the mean or median value (Harrower & Brewer, 2003, pp.29–31). The current version of the web application is implemented in HTML, CSS and JavaScript. As ColorBrewer consists only of premade palettes, the tool does not require a backend running calculations, just the web frontend presenting the palettes (*ColorBrewer source code*, 2021).

In 2005, Gardner evaluated the ColorBrewer color schemes in terms of how well they accommodate map readers with impaired color vision and found the majority of the ColorBrewer color schemes to be accommodating these readers. For the schemes which are not or might not be accommodating, Gardner proposed user interface changes to communicate this to the ColorBrewer user (Gardner, 2005, p. 88). Brewer (2006) recommends using an additional tool ('Daltonize' on vischeck.com) apart from ColorBrewer to better check color palettes for CVD suitability.

3.2 Colorgorical

Colorgorical is an online tool for qualitative color palette generation. It utilizes an algorithm that ensures at least JNDs between all palette colors while also allowing for aesthetically preferable palette generation and user input to use only a certain hue or lightness range for generation. The aesthetical preference can be influenced by setting the range filters and adjusting the importance of perceptual distance, name difference, pair preference, and name uniqueness, which are then used as weights

within the generation algorithm (Gramazio et al., 2017). The main user interface consists of a side panel on the left side with the adjustable controls and the main page on the right side, which consists of text information and after generation of colors a row per generated palette, which houses example data visualized as a choropleth map, a bar chart and a scatter plot using the color palette. The example data cannot be adjusted. A small button allows for the display of additional graphs plotting the palette colors' CIELab lightness, and a and b components, as well as CIELch chroma and hue (Gramazio, 2016). Colorgorical is implemented with a backend for palette construction using NumPy and C combined with a frontend for user interaction using HTML, CSS and JavaScript with the D3 and Bootstrap libraries (Gramazio, 2016).

3.3 Chroma.js color palette helper

The Chroma.js color palette helper is a web tool to help create sequential and diverging color palettes. It utilizes interpolation through the perceptually uniform CIELAB color space and automatic checks after palette generation to test for suitability for color vision deficient viewers. The CVD test checks whether the difference between two colors is similar to the difference under normal vision, therefore it does not check whether the classes in general are always at least one JND apart (Aisch, 2019a). The user interface is only one main screen with one column with four steps from top to bottom which lead through the generation and export process. The currently generated palette is visualized as a bar showing the palette colors next to each other and graphs for the lightness, saturation and hue components of the palette colors. There is no visualization on example data (Aisch, 2019a). The tool is implemented using just a frontend which runs all necessary scripts on client-side with HTML, CSS and JavaScript using the Chroma.js and Svelte libraries.

CHAPTER 4

Methodology

In the following the methods chosen to answer the research questions will each be described in their own section, as well as argued to why they were chosen and how exactly they were applied in the context of this thesis.

For the overview of the existing theoretical knowledge on the subject in chapter 2 and the review of related applied research and software development projects in chapter 3, a literature review was conducted. A literature review is defined as

> "[the] selection of available documents (both published and unpublished) on the topic, which contain information, ideas, data, and evidence written from a particular standpoint to fulfil certain aims or express certain views on the nature of the topic and how it is to be investigated, and the effective evaluation of these documents in relation to the research being proposed" (Hart, 2009, p. 13).

A literature review can only live up to this definition if it was conducted systematically. The literature reviews undertaken for the chapters 2 and 3 were conducted following the principles established in (Rossiter, 2018b, pp. 31–46): The search strategy included keyword searches in digital databases which were then followed up with the 'spider' approach to identify further publications from the publications previously identified as relevant. For this, ConnectedPapers, a tool for bibliometric network analysis as described by Kammerer, Göster, Reichert, and Pryss (2021) proved useful and enabled a time-efficient literature review.

4.1 Requirement engineering

As the research objective includes the design of a new system, a method is needed to formulate what is necessary for this system to be innovative, necessary and relevant based on the previous work in the field in a structured way. The research field concerned with this is called *requirements engineering*, seen as a part of engineering but also traversing disciplinary boundaries in the form of multidisciplinary requirements engineering (see Crowder & Hoff, 2022). In a general way it can be defined as "[...] the subset of systems engineering concerned with discovering, developing,

tracing, analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction" (Dick, Hull, & Jackson, 2017, p. 9) or as "[...] concerned with the elicitation, evaluation, specification, analysis and evolution of the objectives, functionalities, qualities and constraints to be achieved by a software-intensive system within some organizational or physical environment" (van Lamsweerde, 2009, p. xxi). The latter definition already includes the mention of a software-intensive system and as the system to be designed in this thesis is completely software-based, there are more concrete definitions of requirements engineering especially for within software engineering. Based on and adding to a definition by Zave (1997, p. 315), Laplante (2018) defines requirements engineering as "[...] the branch of **engineering** concerned with the *real-world goals* for, functions of, and constraints on **systems**. It is also concerned with the relationship of these factors to *precise specifications* of **system** behavior and to their *evolution over time and across families of* ***related systems***"(pp. 2–3). Good requirements are supposed to be feasible, valid, unambiguous, verifiable, modifiable, consistent, complete and traceable (Berenbach, Paulish, Kazmeier, & Rudorfer, 2009, pp. 9–13).

For software development, a differentiation can be made between user requirements and system requirements (Sommerville, 2016, p. 102):

1. *User requirements* [emphasis added] are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality.
2. *System requirements* [emphasis added] are more detailed descriptions of the software system's functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented.

Requirements can be further concretized by subdividing them into *functional* and *nonfunctional requirements*. The former are defined as "[...] statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do" (Sommerville, 2016, p. 105) and the latter as "[...] constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services" (Sommerville, 2016, p. 105) and consist of product requirements, organizational requirements, and external requirements.

Product requirements define or restrict the software behavior at runtime (Crowder & Hoff, 2022, p. 132). *Organizational requirements* on the other hand encompass vast system requirements which result from the context of the development and the later use. Examples are requirements that define use cases for the system, and development process requirements like a specific programming language that has to be used. And finally *external requirements* are factors from outside the system that influence it and the process of its development. Examples include ethical or legal requirements. These categories are in practice not as clearly separated and might be

depending on one another, but still provide a useful order to stick to (Sommerville, 2016, p. 105–109).

Understandings of the word requirement can differ (Laplante, 2018, p. 3), for the use in this thesis, a requirement is seen as the specification of how a goal, or a high-level objective in other words, should be accomplished by the proposed system (see Laplante, 2018, p. 4). The requirements for the proof of concept development for this thesis are structured into user and system requirements, with the latter subdivided in functional or nonfunctional requirements. Nonfunctional system requirements will be further split into product, organizational, and external requirements.

4.2 Prototyping

Written requirements are an important part of any software implementation, but the lack of a visual explanation can lead to misinterpretation, which is why a proof-of-concept consisting of wireframes and prototypes is a valuable support in communication requirements and showing how a concept is intended to work (Warfel, 2009, pp. 5–6). The process of creating these wireframes and prototypes is called *prototyping* and can be defined as "[…] externalizing and making concrete a design idea for the purpose of evaluation" (Muñoz, Miller-Jacobs, Spool, & Verplank, 1992, p. 579). A *wireframe* is "[…] a visual representation of the functional page structure. They visually communicate what functional pieces are present on a page and their relationship to each other. Wireframes are typically in black and white or shades of gray" (Warfel, 2009, p. 6). Wireframes are also known as mockups (Arnowitz, Arent, & Berger, 2007; Bähr, 2017). Even wireframes can only further the understanding to a certain extent though, which is where *prototypes* come in: "A prototype is a representative model or simulation of the final system. Unlike requirements documents and wireframes, prototypes go further than show and tell and actually let you experience the design" (Warfel, 2009, p. 6). Thereby, prototypes help demonstrate the feasibility of a software product and provide insight into the possible organization and structure of the final product (Sommerville, 2021, p. 26). For this thesis, in the beginning of the prototyping process, a wireframe is being created before moving on to coding a functional prototype.

Arnowitz et al. (2007, pp. 21–25) name the following four phases for effective prototyping:

1. **Plan**. Determine prototyping needs and plan the prototyping process.
2. **Specification**. Determine prototyping methods and tools.
3. **Design**. Formulate design criteria, and create the prototype.
4. **Results**. Review the prototype, validate the design, and implement the design.

For this thesis, the planning phase consists of establishing the criteria for thematic map color palette creation and for the tool itself, and the derivation of the requirements for the prototype from these. This phase is described in the first part of chapter 5.

The specification phase is described in the second part of chapter 5 as it is rather a continuous process than a step to be done with because of the iterative nature of the prototyping for this thesis: There was not a single prototyping method to be specified, but instead the process started with paper and wireframing prototyping methods before moving on more and more to working on a coded prototype. The design

phase with making design decisions and creating the actual prototypes is described in detail in the second part of chapter 5 as well. The results phase with a review of the prototype and an evaluation of the design is the content of chapter 6 and the further development of the design after this thesis is discussed in 7.

4.3 Heuristic evaluation

Apart from reviewing whether the requirements could be met or not, a second evaluation method was added to also evaluate the usability of the proof of concept. *Heuristic evaluation* is a method from usability engineering to identify usability problems within the design of user interfaces. The key feature is to work with a small group of evaluators who examine the interface and assess its conformance to accepted usability principles, the eponymous *heuristics* (Nielsen, 1994a). Heuristic evaluation is among the most actively researched and used usability inspection methods, and it can be utilized early on in the software development cycle to allow for discussion of interface changes while the product is not yet fully developed (Hollingsed & Novick, 2007). Also, it is suited for all development stages, requires only low training, and has low costs (Wilson, 2014, p. XIII). Nielsen (1994b) states that usability specialists normally lead to better results, but are not necessarily required as the method is easy enough to apply. For all of these reasons, heuristic evaluation was chosen as the second evaluation method for this thesis.

While a single evaluator is prone to miss a lot of the possible usability problems in an interface, working with a small panel of evaluators significantly increases the proportion of found problems. Nielsen (1994a) recommends working with three to five evaluators, as the additional information gained per additional evaluator decreases with each evaluator more. For this thesis, five evaluators and me participated in the heuristic evaluation, which should lead to about 78 percent of possible usability problems being detected (Nielsen, 1994a).

Including me, four evaluators have an education in cartography. These are therefore not usability experts, but have some knowledge in the field and at the same time add knowledge from a user perspective as cartographers are also the target group of the developed tool. The other two evaluators are a person with communications background and an artistic researcher, both of which added a second perspective on the user interface aspects and the color palette generation aspects of the application each.

Nielsen (2020) names ten general principles for interaction design:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design

9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

The form used for the heuristic evaluation was derived from these ten principles by assigning heuristics specific to the context of the proof of concept to each principle. As a last section in the form, three usage scenarios were added to see if the evaluators could achieve actual results using the application from a user perspective. The main body of the heuristics form is provided in appendix A, while the complete, formatted Excel spreadsheet sent to the evaluators is provided with the thesis.

CHAPTER 5

Implementation

This chapter describes how the proof of concept was implemented to fulfil the research objective. It is structured into sections for each of the research questions 1 to 4 and answers them one by one.

5.1 Identified criteria for thematic map color palettes

Research question 1, 'What criteria are necessary to decide whether to use a color palette for a thematic map considering not only choropleth maps, but also proportional point symbol maps and multi-layered combinations?', aims to encompass the decision process that the proof of concept is being built to assist. The criteria to answer this question are derived from the theoretical background in chapter 2 and are unweighted and therefore in random order here. They are phrased as binary questions that can be answered with yes or no. The important criteria for this decision are...

1. Is the palette suited for the kind of data to be displayed? Are the visual variables which vary within the palette suitable to represent the variation in the data?
2. Does the palette consider human perception to not skew the comprehension of the data?
    - 2.1 Are the palette colors notably different from one another?
    - 2.2 Are the palette colors correctly spread out in a perceptually uniform color space to represent the data?
    - 2.3 Is the palette considering simultaneous contrast (and similar phenomena)?
    - 2.4 Is the palette considering color vision deficiencies in its design? Are the other criteria also being met for viewers with color vision deficiencies?
3. Is the palette aesthetically pleasing?
4. Are the palette colors harmonious (according to the definition in chapter 2)?

5. Is the palette supporting the position of the map layer within the visual hierarchy of the map?

6. Is the palette suitable for the deployment situation of the map?

7. Was the palette tested before use to ensure meeting the other criteria?

For this thesis, a color palette meeting these criteria will be considered suitable for use in a map.

5.2 Identified criteria for cartographic color palette tools

While the section before focuses on the color palette itself, this section is directed at important aspects for the implementation of a cartographic color palette tool with the aim of improving over existing research and tools. In doing so, research question 2, 'What color palette generation and testing tools exist already? How can a new tool improve upon the existing ones?', will be answered and the practical side of the implementation is considered. This builds upon the review of related work in chapter 3.

Discussion of related work

As ColorBrewer is the most well-known cartographic color palette tool and uses a screen layout, that e.g. also Colorgorical uses as well (side panel with controls on left side plus right side for visualization), this general layout of the application will be taken over for the proof of concept, to create a sense of familiarity for everybody who worked with ColorBrewer before and to ensure a certain baseline of user interface quality.

In the same sentiment, the three categories 'sequential', 'diverging', and 'qualitative' as used by ColorBrewer for possible palettes seem like a good choice and are a better distinction than for example using the hierarchy of scales directly. Also, this distinction of possible palettes is well-established, e.g. the Chroma.js color palette helper also uses the terms 'sequential' and 'diverging' as well as textbooks like (Brewer, 2016). As the proposed tool is supposed to improve over existing tools, it should be able to generate palettes of all three categories.

Colorgorical is a powerful tool for the generation of qualitative color palettes and has a lot of algorithm input customization with intuitively designed controls. Also, it is possible to generate multiple palettes and compare them to each other. It does need a server backend though, which makes the application infrastructure more complex. And the used algorithm is very specific to qualitative palettes, cannot be easily adjusted to also output sequential and diverging palettes. Furthermore, the given testing environment does not offer customization options apart from the option to toggle a certain visualization on and off and does not offer additional information for color vision deficiency support or information how to read and properly work with the given graphs.

ColorBrewer as well as Colorgorical only offer a choropleth map as the only cartographic option, so it would be better to include different kinds of thematic map to allow for more different usage scenarios. For the proof of concept, at least a point symbol map and a choropleth map should be supported, as well as viewing both as layers of the same map. Other color palette tools, like the Chroma.js color palette helper offer no cartographic visualization at all. Also, none of the reviewed color

palette tools offer the option of displaying the chosen palette on example data in front of changing basemaps. ColorBrewer only has the option to display grayscale terrain as a background.

The testing environment in the Chroma.js color palette helper is helpful with the graphs for lightness, saturation and hue, as well as the automatic CVD testing and the possibility to also apply a CVD view simulation to the palette for manual evaluation of the generated color palette.

Color harmony templates like used by Tokumaru et al. (2002) can be used to ensure a certain level of harmony in the generated color palettes. Basemap color extraction could be implemented similar to Tan et al. (2018) and Zheng et al. (2022). If no libraries to work with color vision deficiencys in a web environment are found during prototyping, CVD testing could be implemented with the equations from Machado et al. (2009).

Six reviewed works only considered color schemes for general design purposes and not specifically for data visualization and were thus less relevant for this thesis (see Hu et al., 2012, 2013; Lindner & Süsstrunk, 2013; Meier et al., 2004; O'Donovan et al., 2011; Peng & Chou, 2019).

Criteria for cartographic color palette tools

The criteria are formulated to add to the criteria for thematic map color palettes, therefore e.g. color harmony is not mentioned here again. They are unweighted and presented in random order. The criteria are phrased as binary questions that can be answered with yes or no. The final important criteria are…

1. Does the tool feature an easy-to-understand screen layout and user interface similar to the ColorBrewer design?
2. Does the tool offer palette generation and testing for the types 'sequential', 'diverging', and 'qualitative'?
3. Does the tool offer example visualization for relevant types of map?
4. Does the tool offer example visualization in front of a selection of basemaps?
5. Does the tool offer additional graphs to help evaluate the color palette visually?
6. Does the tool run automatic tests and algorithms to ensure a certain quality level for the generated output? (Especially regarding JNDs and CVDs)

5.3 Requirements

The criteria for thematic map color palettes and the criteria for cartographic color palette tools were combined to derive the requirements for the proof of concept and thereby answer research question 3, 'What requirements exist for a tool implementing these criteria and improving upon the existing tools?'.

The user requirements 1 - 7 and the functional system requirements 1 - 7 are ordered in the same way as the seven criteria for thematic map color palettes, so the connection from a criterion to its corresponding user requirement and in turn the more detailed system requirement is easy to make. The criteria for cartographic color palette tools influenced the same requirements and if necessary, additional requirements were added.

User requirements for the proof of concept

The general objective of the tool stated as a high-level goal is: *The tool assists cartographers in choosing suitable color palettes for thematic maps.*

1. The tool generates color palettes suited for a chosen kind of data to be displayed (sequential, diverging, and qualitative) by utilizing changes in lightness, hue and chroma accordingly.
2. The tool generates color palettes that consider human perception to not skew the comprehension of the data.
    - 2.1 The tool generates color palettes with colors notably different from one another.
    - 2.2 The tool generates color palettes with colors spread out in a perceptually uniform color space befitting the distribution of data.
    - 2.3 The tool generates color palettes considering simultaneous contrast and similar phenomena.
    - 2.4 The tool generates color palettes considering color vision deficiencies.
3. The tool generates color palettes which please aesthetically.
4. The tool generates color palettes consisting of harmonious colors.
5. The tool generates palettes for multiple map layers at once in a systematical fashion. This is facilitated through palette derivation from a set of seed colors as well as shifts in lightness and chroma per information layer.
6. The tool generates color palettes fitting the deployment situation of the map.
7. The tool provides a testing environment and information to allow the user to evaluate the previous requirements, as well as running automated tests and communicating the results of these.
8. The tool is easy to use, easily available online and can be executed on a variety of target devices.

Functional system requirements for the proof of concept

1. The tool offers modes for the mentioned three kinds of data and changes settings accordingly internally to work with proper visual variables; e.g. for categorical data the lightness for all colors shall be the same and only the hue should be varied.
2. *Functional system requirements for color palette generation considering human perception*
    - 2.1 The tool checks the difference between palette colors and ensures it is at least a JND. The result of this test is given back to the user as a visual feedback.
    - 2.2 The tool derives color palettes from so-called seed colors by interpolating between them through a perceptually uniform color space to ensure correct distances between the colors within the palette.

2.3 Based on imported user data to be displayed, layer opacities, symbol sizes, and the currently chosen color palette, the tool checks for problematic simultaneous contrast and gives the user a visual feedback.

2.4 The tool checks that colors are still noticeably different when viewed with a color vision deficiency and gives a visual feedback about the result of this check.

3. The tool utilizes randomness and the option to regenerate seed colors to allow the user to find a palette aesthetically preferable to them or their target group. A color picker and simultaneous updating of all displayed colors accordingly allow for on-the-fly quick changes to the generated colors to fit them to preference if the general direction is already likeable to the user. Seed color hue template implementation and filters allow for more targeted generation directly as well. Seed colors should also be able to be generated from a basemap or one or multiple start colors, as well as allow for loading of seed colors.

4. Hue templates with equidistant hues are to be implemented as an option in seed color generation.

5. The tool first generates so-called seed colors from which then multiple palettes can be derived to aid in generating a uniting theme for multiple map layers. Lightness and chroma shifts from lighter and less chromatic colors in the back to darker and more chromatic colors in the front (or vice versa for palettes to be utilized on a dark background).

6. The tool is specialized on thematic maps in a web environment in typical lighting situations. The tool considers different deployment scenarios within this context by considering a variety of basemaps and background colors for the palette generation.

7. *Functional system requirements for the testing environment*

7.1 The changes applied per mode are also visualized to the user immediately.

7.2 *Functional system requirements for testing a color palette considering human perception*

7.2.1 The user receives feedback on how different colors are from one another perceptually using graphs to visualize the steps in lightness, hue and chroma within the palette.

7.2.2 The user receives feedback on how colors are spread out in the color space by using graphs to visualize the steps in lightness, hue and chroma within the palette.

7.2.3 The tool visualizes two different scale datasets to the user and allows for random data regeneration or own data import to see unlimited possible class combinations. A range of freely available basemaps is supplied as well. The tool also features an opacity slider to allow for testing at different opacities per layer and to toggle layer visibility on and off in general.

7.2.4 The tool also simulates the view of a person with a CVD and visualizes this view to the user to allow for evaluation by the user themselves.

7.3 The seed colors, the generated palettes and changes to them, e.g. applied modifiers, are always directly visualized to the user to evaluate their aesthetics.

7.4 The seed colors, the generated palettes and changes to them, e.g. applied modifiers, are always directly visualized to the user to evaluate their harmony.

7.5 The tool offers the user to adjust settings regarding the visual hierarchy.

7.6 The variety of basemaps and background colors is also available in the given testing environment as well.

8. The tool's user interface is designed after ColorBrewer to help with orientation for cartographers familiar with this or similar existing tools. Further, the tool utilizes intuitive metaphors and standards often used in user interface design. The generated palettes should be easily exported in multiple formats to ease continued work with them. The tool features a responsive user interface which adapts to a range of target devices.

Nonfunctional system requirements for the proof of concept

Product requirements

1. As no user data is collected and the tool runs directly on the target device without a server backend, no security concerns arise. This also keeps the chance of reliability failures apart from bugs and the initial loading of the page low as the program can be run offline as long as the website was loaded fully while still online.

2. Performance requirements are supposed to be low, the implementation in general should allow for loading times to be longer on less powerful devices instead of a complete system failure. Also, using a frontend only approach keeps reaction times comparably small, especially with slow internet connections, as the software doesn't need to communicate back and forth outside the target device.

Organizational requirements

1. An operational process requirement is, that the system shall be useable by cartographers globally and with varying skill sets, therefore a web-based approach is chosen as the most including.

2. A development process requirement resulting from this is, that a programming language, that can be executed on the web should be chosen. For this, JavaScript was selected and during the third iteration of prototyping further specified to TypeScript, a superset of JavaScript.

3. The development environment and process standards are to use Visual Studio Code for the prototyping, together with the Prettier formatting extension, and with the Svelte for VS Code extension (added during the third iteration of prototyping).

4. The operating environment should support a long and cheap product life span during and after the thesis. For this, also a frontend only approach is preferable, as well as using only necessary external libraries and web standard technologies.

External requirements

1. As no user data is being collected, and no cookies are used, there are no specific ethical requirements.

2. The research context influences the prototyping, because the proof of concept has to be able to be implemented in a short time period and has to be adaptable to new insights, and possible feedback by the thesis supervisor.

5.4 Iterative prototyping process

After the requirements were fixed, the prototyping process began to develop the actual proof of concept from the requirements and answer research question 4, 'How can these requirements be implemented in a proof of concept?'.

The prototyping process is composed of multiple iterations with intermediate prototypes each, before reaching the working proof of concept as a web application in the last iteration. To keep the thesis concise, only selected screenshots in smaller scale will be shown in this section. For a full overview of screenshots for all intermediate prototyping results, refer to appendix B.

First iteration: Wireframe of the intended proof of concept

The first iteration began with brainstorming and first drafts on paper, which were then transferred and refined into a first digital wireframe. This digital version was created using Figma, a web-based design platform, and Untitled UI, a user interface design kit for this platform, to allow for fast wireframing while at the same time achieving a comparably polished result. The wireframing was helpful in deciding how to lay out the user interface roughly for the first versions of the proof of concept. The main view of the wireframe is depicted in figure 5.1.

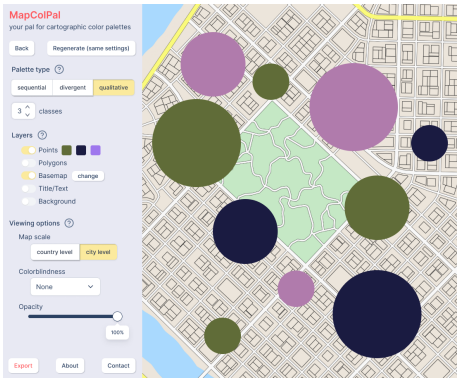


Figure 5.1: Wireframe

Second iteration: Technical capability testing

After the wireframe was done, and I also found JavaScript libraries that appeared promising for the implementation of the proof of concept, the next step was to do simple technical tests to better gauge whether I think the technologies and my own coding skills allow for an actual coded proof of concept. This iteration followed the idea of an *early rapid prototype*, a prototype helping to plan out concepts, make them easier to understand, and increase confidence in the project planning later on (Arnowitz et al., 2007, p. 16).

Figure 5.2: Technical capability test

In this first coded prototype, depicted in figure 5.2, I combined color interpolation through a perceptually uniform color space with chroma.js and a simple graphic visualizing these colors with D3.js. Implementing this first prototype worked well, therefore I moved on to the next, knowing that at least a simple tool for sequential palettes using color interpolation should be possible to implement.

Third iteration: First coded user interface prototype

This iteration also followed the principle of an early rapid prototype and had the goal to fix the rest of the technologies to be used in the proof of concept, at least the ones that were clear would be needed: A library to work with basemaps as D3.js does not easily facilitate that, a frontend framework to be able to store state and update related parts of the application on state change as well as facilitate easier development, a user interface component library to not have to design every interface control entirely on my own.

For this test, I used Svelte as a frontend framework, because it seemed to be a good choice for a small-scale application with less boilerplate code than e.g. React would require. Together with Svelte, I tested switching to TypeScript, as this supports type checking which eases development by allowing error messages during compiling and not only during runtime. Using Svelte also made testing of a specific programming approach possible: Encapsulated components, where the HTML, CSS and TypeScript code for each component are written together in one '.svelte'-file. As a UI component library, I tested daisyUI, which in turn is based on Tailwind CSS and PostCSS. Apart from D3.js and chroma.js as in the last iteration, I now also tested Leaflet and OpenLayers for working with basemaps. The implemented prototype for this iteration is depicted in figure 5.3.

Also during this iteration, the name MapColPal was brainstormed for the tool as a shortened combination of the words 'map', 'color', and 'palette', and at the same time a pun with the word 'pal' as well. During this test I decided to use Leaflet for the proof of concept, as it was easier to get D3.js and Leaflet to work together than D3.js and OpenLayers. To easily use different basemaps within Leaflet, the plugin Leaflet-providers was added to the libraries in use. As this then all worked without problems, I decided to continue working with these technologies.

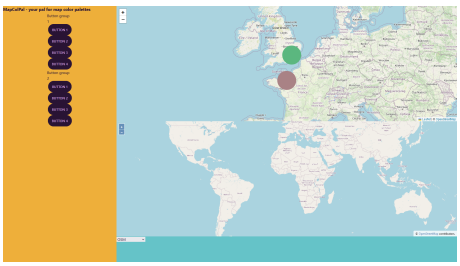


Figure 5.3: First coded user interface prototype

Identified components of the proof of concept

During this iteration, one decision to make was, how to divide the planned application into parts to structure the code. For this, the concept of encapsulated components was applied and the layout of the wireframe was used as an orientation. The application was divided into the following components:

1. Side panel
2. Map view
    a) OpenLayers test map
    b) Leaflet test map

This basic structure of dividing the app into the side panel and the map view was kept after this iteration and adapted to house more components as they were being programmed, the final structure is being presented in chapter 6.

Fourth iteration: Proof of concept implementation

After having created the wireframe to know how the user interface could look like generally and having coded two early rapid prototypes to fix the technological base of implementation, I started working on the actual proof of concept.

It was developed iteratively and organically over time in an agile manner, without setting myself clear milestones for what order to code in. Even though I worked on it alone, the proof of concept was created using the Git version control system, so each committed change to the code is documented with a commit message. The commit messages are loosely structured into four categories:

- New: Adds new features to the codebase.
- Fix: Fixes a bug or broken feature.
- Refactor: Rewrites a part of the code to improve readability and maintainability, and reduce complexity.
- Merge: Update the 'main' branch with the new commits from the 'prototyping' branch.

In total, there were 165 commits. The categories were not used mutually exclusively, so a commit can also be tagged with two categories at the same time. 114 of

the commits included the category 'New', 29 commits each included the categories 'Fix' or 'Refactor', and 9 commits were of the category 'Merge'. I merged the 'prototyping' branch into the 'main' branch, when I felt like having done some important changes, therefore these provide good points to pick for in-between-versions to exemplary summarize the development process here. As some merge points were in short succession, every second merge will be described here with the last being presented in more detail in the Results chapter.

Screenshots are taken at full-screen resolution on a full HD, 1920 x 1080 pixel, monitor as this is the size the user interface was created at in the beginning before optimizing the display at smaller resolutions later on in the process.

First merge

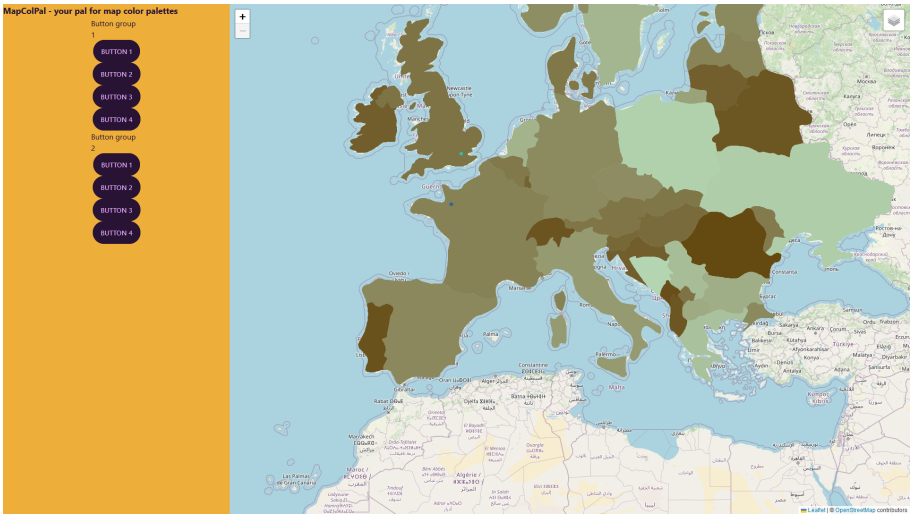


Figure 5.4: Implementation - First merge

The application at the first merge, as depicted in figure 5.4, is visually and functionally still similar to the first coded user interface prototype. The main map display on the right side of the screen features a zoomable and draggable overlay implemented with D3.js with country polygons and manually placed points in front of a Leaflet basemap, which can be chosen from two options. The user interface elements in the side panel on the left are still placeholders only. The data used for the country polygons was from Project Linework (projectlinework.org), already with a loose focus on Europe. The colors applied to the polygons and points are regenerated randomly on reload of the website.

Third merge

At the third merge, the application is already much closer to the final design, as depicted in figure 5.5 and also implements all the features planned in the wireframe visually. The side panel was subdivided into three different sub-panels. Firstly, the 'start' panel for seed color generation, the 'options' panel for derivation of color palettes from these seed colors and for viewing them, and the 'export' panel to export the generated color palettes. A progress indicator in the top of the side panel visualizes

which step the user is currently in. Within the code repository, Svelte stores were introduced, to allow for exchange of values between components and at the same time be able to store them. The seed colors can now be regenerated on button click as well and the export panel also is functional already. All other user interface elements are not functioning yet. For the data, the polygons were changed to Natural Earth data (naturalearthdata.com) and filtered in GIS software to only contain member states of the European Union. This was done so that the test data, and area depicted, is not completely randomly chosen. The European Union was chosen, because this thesis is written within a Master's program which is co-funded by the Erasmus+ program of the European Union.

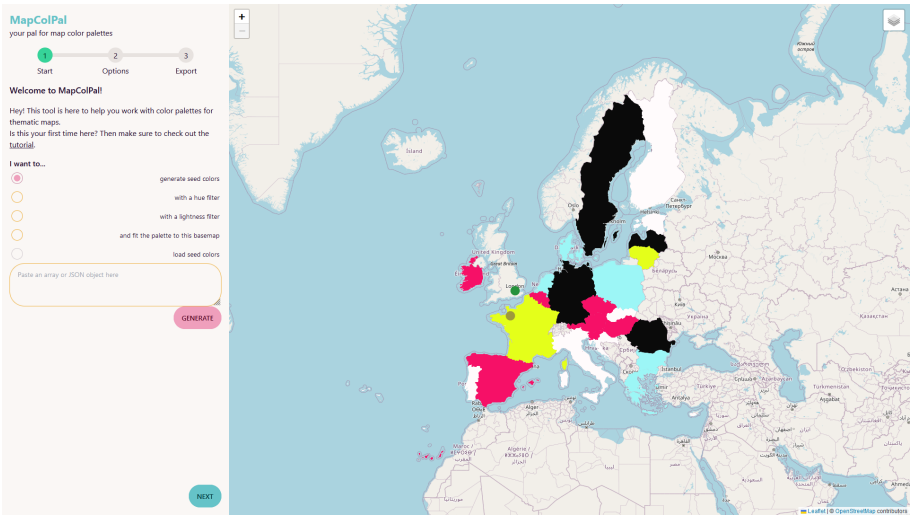


Figure 5.5: Implementation - Third merge

Fifth merge

For the fifth merge, as depicted in figure 5.6, a favicon was added that is meant to be simple and represent the shape and color from the progress indicator at the top of the side panel as this is a visually present element in the MapColPal interface. The fifth merge added alert messages on error during import of seed colors or after a successful import. Modals, pop-ups which open on button click, were added. At this point one of them was intended for additional settings the user could set and the second one for additional information about the application. Capitals from Natural Earth were added as point data for the country level visualization and Vienna districts and swimming pools were added as polygon and point data on city level, as Vienna is a city I like and the geodata is freely and easily available. The size of the displayed points is relative to the generated data values. The basemap can now be chosen from a control in the side panel. Seed colors can be loaded from a text input or from the URL of the website. Other generation controls apart from the number of colors are not working yet. The 'options' panel was split into two panels, the 'layers' panel with options regarding the structure layers within the map and the color palette generation for them, and the 'test' panel for future palette testing facilities. The 'test' panel is still mostly empty for now, the 'layers' panel offers the user to choose between coun-

try or city level for the map, to generate new random data for each layer, and to toggle layers on or off and change their opacities. Also, the background color can be set.

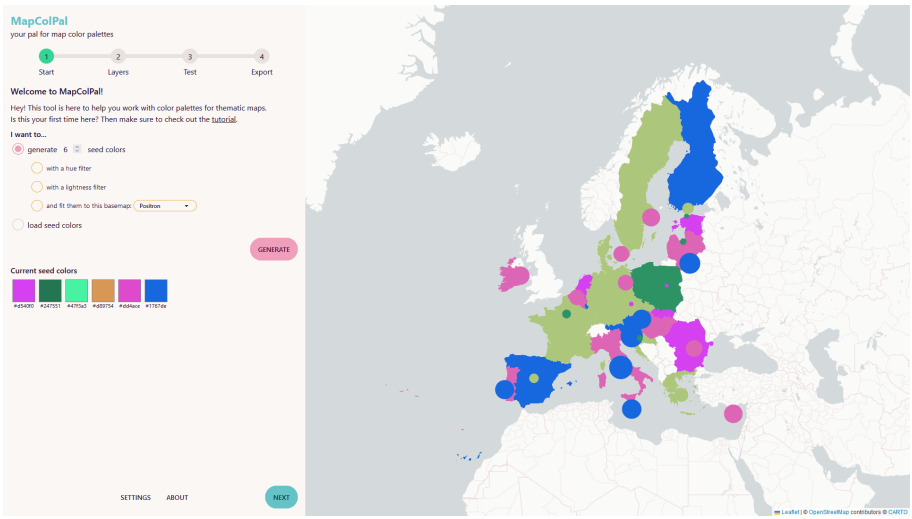


Figure 5.6: Implementation - Fifth merge

Seventh merge

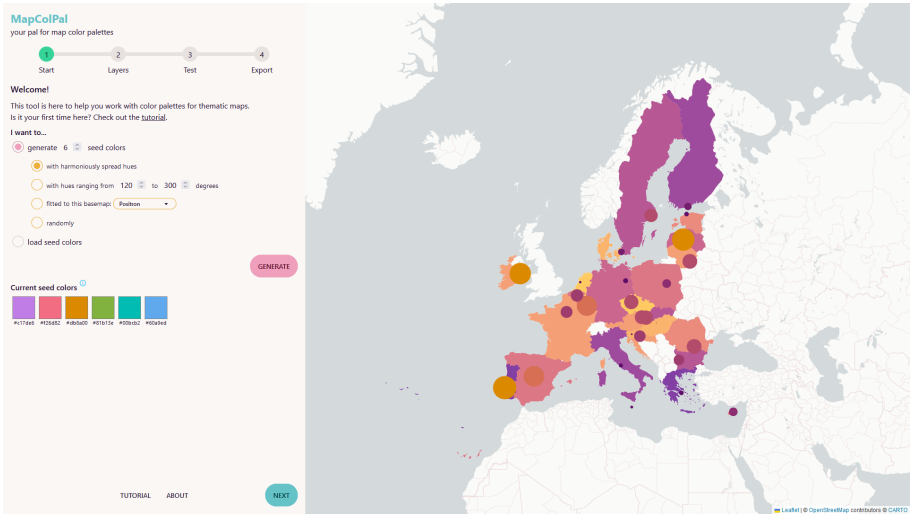


Figure 5.7: Implementation - Seventh merge

For the seventh merge, as depicted in figure 5.7, the seed colors are now automatically updated in the URL as the application is used. In terms of the code, the seed colors are now handled not only as an array of hexadecimal color codes, but as an array of objects containing also an ID, which allows for easier working with them. The former 'settings' modal was now changed to host a tutorial as the 'settings' modal seemed to stay unnecessary for now, as there are no additional options that need

space to put them. The tutorial follows the same order as the sub-panels, although no actual content was added yet. Also, buttons for additional information are spread out throughout the app. The seed colors are now finally not directly applied to the data anymore, but instead a palette is derived from the seed colors depending on user input within the 'layer' panel. Drag and drop elements were added to choose which seed colors to use for palette generation for a layer and for the order they should be used in. They are based on the visual metaphor of a deck of cards. Further options in the 'layer' panel include lightness optimization which differs per palette type, color curve smoothing which utilizes Bézier interpolation through CIELAB color space, a button to reverse the input color order, and size options for the point layer. Now all points can also be set to a fixed size instead of changing with the data. The user interface is conditional, for example if a layer is toggled off, all unneeded controls are set to be invisible as well. Now, seed colors can not only be generated randomly, but also with a hue template to spread them out equally depending on the number of seed colors to be generated, and if selected then map them to a certain hue range. The basemap seed color derivation is also functional now, and a loading animation was added to signal that the asynchronous process is still running. The basemap color derivation first merges, for performance reasons only a part of, all loaded basemap tiles into one image, then extracts the color from each pixel and quantizes the colors to reduce to a small palette before deriving the mean color from this palette. This mean color is then used as the first color in the hue template as described before. The color quantization was implemented following Sniurevicius (2022). The 'test' panel is now also functional and allows for CVD view simulation, and shows a lightness graph for each layer. The 'export' panel is now more formatted and offers to export colors for each generated palette and for the seed and input colors.

Ninth merge

The ninth merge was the final one and while it will be presented in more detail in chapter 6, the changes included in this merge will be discussed here. So far the user interface contained a mix of flat shading and drop shadows, which were now unified into a simplified shading using colored CSS borders inspired by Tanaka contours with an imaginary light source in the upper left corner of the screen. All informational texts and the tutorial were added in as well as overall the user interface and content was polished and unified.

CHAPTER 6

Results

This chapter sets out to answer the last research question 'Does the proof of concept fulfill the requirements set before? And can it fulfil the research objective?' by presenting the completed proof of concept and then evaluating it. The evaluation is on one hand done by checking the proof of concept against the requirements set before. On the other hand, the results of the heuristic evaluation are being discussed afterwards. Finally, sample resulting palettes are presented and compared to ColorBrewer palettes.

6.1 Completed proof of concept

The proof of concept is a responsive web application for color palette generation and testing for thematic maps, as shown in figure 6.1.

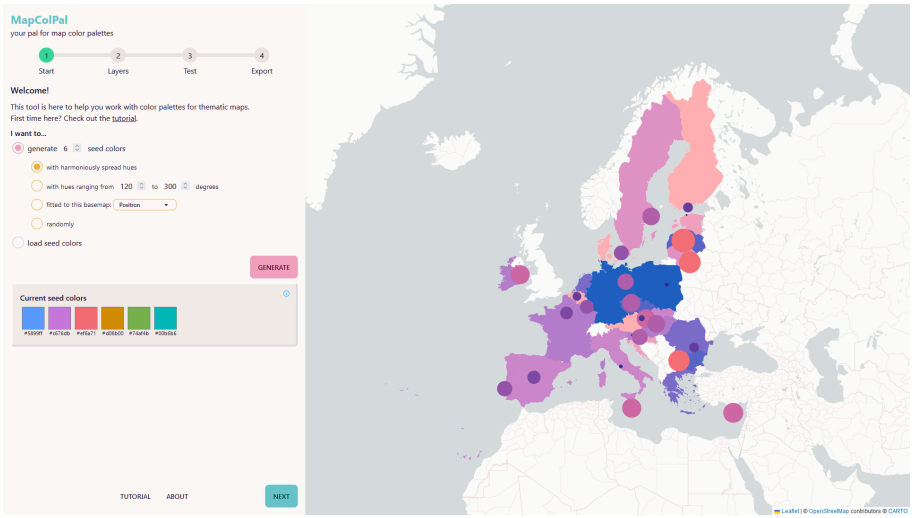


Figure 6.1: Proof of concept - Main view

It allows to either generate seed colors according to a hue template or randomly, or to load colors in hexadecimal RGB form. When generation, the first color in the hue template can be derived from the currently displayed basemap, or the colors can be mapped to a certain range of hues. Seed colors can be further adapted by the user on click wherever they are shown in the user interface.

These seed colors are then used to derive a color palette per map layer. This can be influenced by choosing which seed colors to use and in what order, the number of classes and a palette type from the options 'sequential', 'diverging', and 'qualitative'. The seed colors chosen as input then get interpolated between to derive a palette. Depending on the chosen palette type, the lightness is optimized fittingly: For sequential palettes, it ranges from low to high or vice versa, for diverging palettes from low to high to low or vice versa, and for qualitative palettes, the lightness is set to the same level. The exact middle value and range to use for this lightness correction can be chosen by the user. A Bézier curve interpolation between the input colors can be applied to smooth the resulting color palette. If only one seed color is used for sequential palette derivation, lightness steps of this color are generated. In the same situation for diverging palettes, the complementary color, the color on the opposite side of the color wheel, is chosen for the other end of the color palette, while diverging palettes with only one or two input colors use an achromatic color for the middle of the palette.

Afterwards, the created color palettes can be viewed with a CVD simulation and evaluated using a graph showing the steps in lightness within the palettes. This way, the user can verify that a palette applies lightness in a way befitting the data type and distribution, and that it is discriminable also when viewed with a CVD.

Finally, the palettes can be exported as a JavaScript array, with or without quotation marks around each color element, and formatted as hexadecimal RGB colors, as the RGB format used in CSS, as CIELAB components, or as CIELCH components.

During this whole process, the generated palettes get displayed on a sample map in real-time. They are shown as a point symbol map layer and/or a choropleth map layer in front of a chosen basemap or background with a chosen color. Every map layer can be shown or not and the opacity can be varied. The data values are randomly generated and can be regenerated to view the same palette in different conditions, as well as changing the map scale between a city-scale (Web Mercator zoom level 11, rounded scale 1:300000) and a country-scale (Web Mercator zoom level 4, rounded scale 1:37000000). The point symbols can be changed in size and can be either of a fixed size or sized relatively to the data.

Additional information on everything is presented in form of a tutorial modal, a modal with metadata, and contextual pop-ups.

Technologies used

The proof of concept uses the following technologies and libraries:

- **HTML, CSS, TypeScript, Node Package Manager, Vite, and Svelte**. The basic technologies the application is based on.
- **PostCSS, Tailwind CSS, and daisyUI**. The libraries used for the user interface.
- **D3.js**. To display and interactively update the map point and polygon data layers.

- **Leaflet, and Leaflet-providers.** To display the basemap tiles.
- **Merge-images.** To merge the basemap tiles into one canvas element to then extract its colors.
- **Color-blind**. To simulate CVD vision.

Component hierarchy

The proof of concept uses the following code structure:

1. Stores
2. Components
    a) Map view
    b) Controls
        i. Side panel
        ii. Start panel
        iii. Layers panel
        iv. Test panel
        v. Export panel
    c) Elements
        i. Exported colors
        ii. Help dropdown
        iii. Input colors
        iv. Lightness graphs
        v. Modal
        vi. About modal
        vii. Tutorial modal

The 'stores' hold information that needs to be accessed and manipulated by multiple other components. The 'components' house all components with parts visible in the user interface, such as the 'map view' which displays the map on the right side of the application. The 'controls' house the main user interface control groups. The 'side panel' is the parent component for the other four panels which are displayed within this parent component. The 'elements' house all other components of smaller scope. The 'exported colors' are used within the export panel, the 'help dropdown' is used throughout all side panels. The 'input colors' and 'lightness graphs' are used in the layers panel and the test panel respectively and are extracted as their own components because they are each used twice, once for the point layer and once for the area layer of the map. The 'modal' is the parent element for the 'about' and the 'tutorial' modal.

User interface elements

The user interface is designed after the flow of the process as described before. It mainly consists of the side panel housing all important user controls on the left side of the screen, and the map view on the right side of the screen. On a device with a small screen the map gets displayed below the side panel. Additional screenshots of the user interface at different screen resolutions can be found in appendix C.

Map view

The map view informs the user about the current settings and allows viewing the generated palettes. It is depicted in figure 6.1 on the right side of the application's interface.

Start panel

The 'start' panel contains all controls related to seed color generation and loading. It is depicted in figure 6.1 on the left side of the application's interface.

Layers panel

The 'layers' panel, depicted in figure 6.2, holds all controls related to color palette derivation from seed colors, as well as the major map view options, such as the ability to change layer opacities for example.

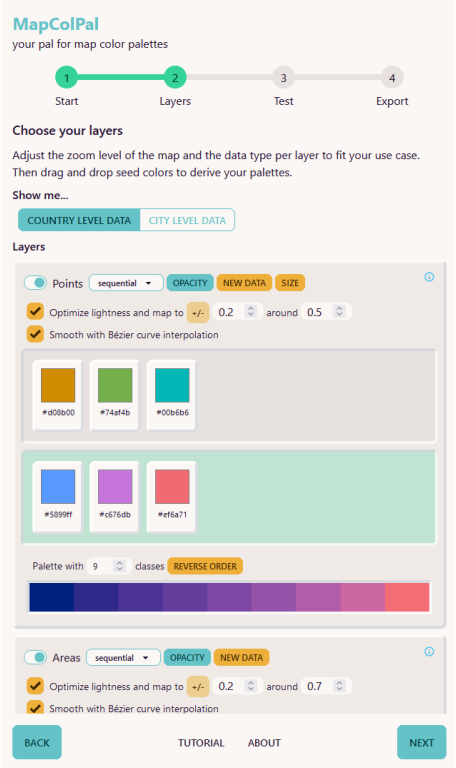


Figure 6.2: Proof of concept - Layers panel

Test panel

The 'test' panel, shown in figure 6.3, contains the control for the CVD view simulation, and the lightness graphs with the ability to switch between a relative and absolute scale.

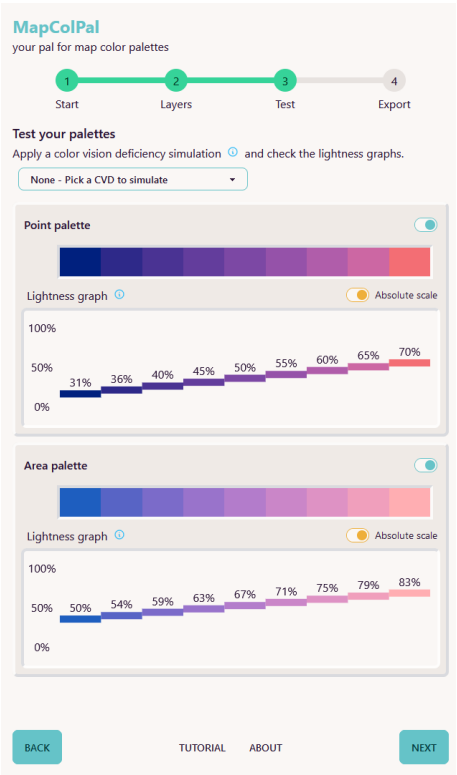


Figure 6.3: Proof of concept - Test panel

Export panel

The 'export' panel, depicted in figure 6.4, contains customization options related to palette export as well as the export button per collection of colors and a text window which allows for manual copying of the colors as well.

Figure 6.4: Proof of concept - Export panel

Tutorial and About modals

The 'tutorial' and 'about' modals, depicted in figures 6.5 and 6.6 respectively, contain additional information for new users and users interested in knowing more about the application in detail.

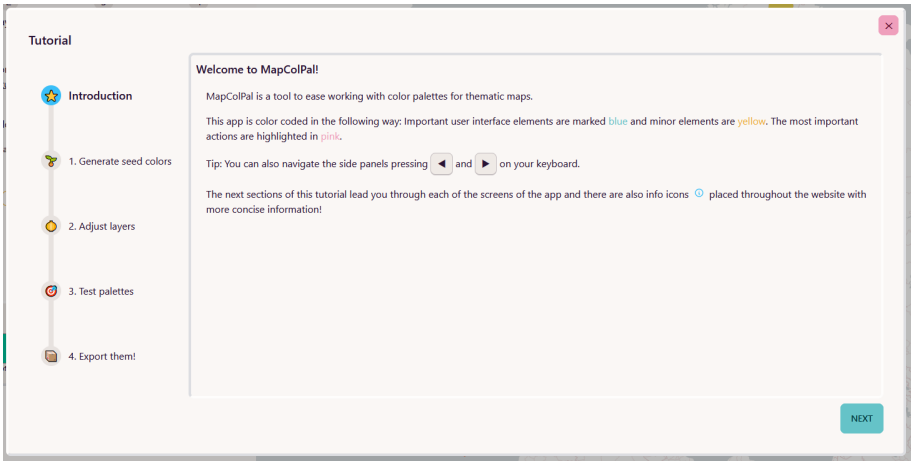


Figure 6.5: Proof of concept - Tutorial modal

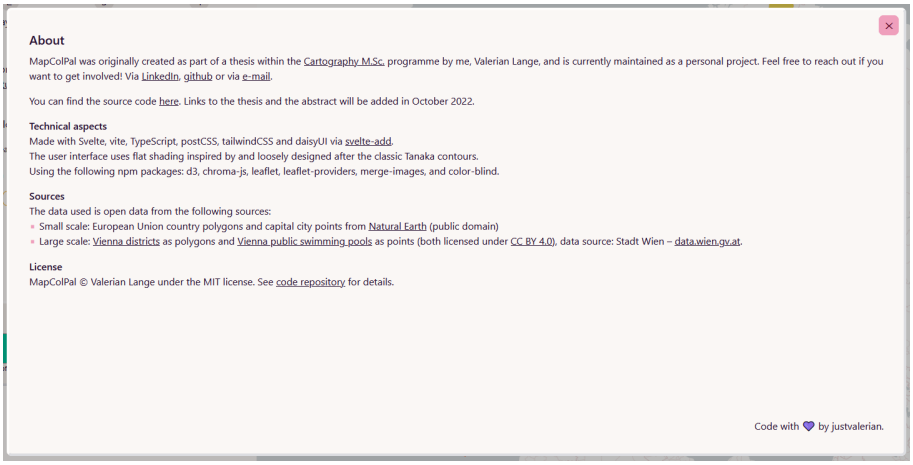


Figure 6.6: Proof of concept - About modal

Error and success alerts

The alert messages which appear and disappear on the user interface automatically as required, give feedback for the import and export steps of the workflow. The success message shown in figure 6.7 appears on successful seed color import, the error alert depicted in figure 6.8 appears on unsuccessful seed color import, and the message shown in figure 6.9 appears on button press to export colors by copying them to the clipboard.

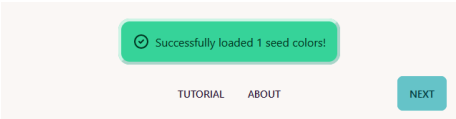


Figure 6.7: Proof of concept - Success alert

Figure 6.8: Proof of concept - Error alert

Figure 6.9: Proof of concept - Export alert

6.2 Requirements check

The finished proof of concept will now be compared to the set requirements in chapter 5. For this comparison, the functional system requirements have been selected, as

they are more detailed and implementation-focused compared to the user requirements, and comparing both here would yield overlapping results. The nonfunctional system requirements are not discussed here in detail, as they were shaping the prototyping process more than the final outcome and were adhered to during this process. The tool was tested in the Firefox and Chrome browsers in the current versions for September 2022.

- ✔ Requirement 1 is being met by the proof of concept, it can generate color palettes of all three kinds.

  Functional system requirements for color palette generation considering human perception

  - ✖ Requirement 2.1 is not being met, currently no automated check for color difference is implemented.
  - ✔ Requirement 2.2 is being met.
  - ✖ Requirement 2.3 is not being met. To consider visual phenomena like simultaneous contrast in palette generation requires the import of the data used in the end product and more extensive analysis, as the data and its surroundings, layer opacities, symbol sizes and the current color palette need to be considered. Given the non-functional system requirements and restrictions, for the proof-of-concept the focus was then instead put on providing an extensive testing environment. Especially important for this testing requirement is the possibility to see different combinations of data, scale, geographic distribution and basemap. Therefore, the focus for the current tool was on providing the possibility to see random combinations of classes next to each other and at different scale levels and for multiple geographic datasets in front of different basemaps.
  - ✖ Requirement 2.4 is not being met, there is no automated check for CVD compatibility.

- ? Requirement 3 has mostly been met, but seed color generation with a given start color is not yet implemented in the user interface.
- ✔ Requirement 4 is being met.
- ? Requirement 5 is being partly met. Instead of implementing an automatic distribution of the map layers along the possible range in lightness, the users can edit the values themselves. Chroma shifts have not been implemented yet.
- ✔ Requirement 6 is being met. Seed color generation based on the background color was purposefully not implemented to not clutter the interface unnecessarily. Instead, it could be implemented as a sub-feature of the generation with a given start color which is part of requirement 3.

  Functional system requirements for the testing environment

  - ✔ Requirement 7.1 is being met, all changes to the controls are visualized immediately.

    Functional system requirements for testing a color palette considering human perception

    - ? Requirement 7.2.1 is being partly met. Only graphs for lightness are implemented currently.

- ? Requirement 7.2.2 is being party met. Only graphs for lightness are implemented currently.
- ? Requirement 7.2.3 is mostly met. Only import of data by the user is not implemented yet, see the check of requirement 2.3 for more information. Also, the opacity setting for the basemap resets, when the basemap is changed or moved, and needs to be set again.
- ✔ Requirement 7.2.4 is being met.

- ✔ Requirement 7.3 is being met.
- ✔ Requirement 7.4 is being met.
- ✔ Requirement 7.5 is being met.
- ✔ Requirement 7.6 is being met.

- ✔ Requirement 8 is being met. An example of a utilized metaphor is a card-based drag and drop interface for rearranging the seed colors for palette generation.

Eleven of the 19 functional system requirements were fully met, five of them are partly met, and three are not being met currently.

6.3 Heuristic evaluation

The results of the heuristic evaluation were compiled into one Excel spreadsheet which is provided with the thesis. The evaluators were pseudonymized with numbers: evaluator 1 is me, the author of this thesis, evaluators 2-4 are the evaluators with cartography background, evaluator 5 is the artist and artistic researcher, evaluator 6 is the evaluator with communications background. If nothing else is mentioned, a 'passed' heuristic was marked as passing by all evaluators.

1. *Visibility of system status*
   - ✔ Heuristic 1.1 passed.
   - ? Heuristic 1.2 passed for all but one evaluator, evaluator 6 tended towards passed, but was unclear on what the heuristic is meant to judge.
   - ✔ Heuristic 1.3 passed, evaluator 2 mentioned visible loading times for the 'Imagery' basemap.
   - ✔ Heuristic 1.4 passed.
2. *Match between system and the real world*
   - ? Heuristic 2.1 passed for four evaluators and was unclear for two. Evaluator 4 notes that some concepts are advanced for non-cartographers or cartographers focused on design, evaluator 6 noted from a non-cartographer perspective, that the application seems mostly understandable for them, but that some words are unclear to them.
   - ? Heuristic 2.2 passed for all but evaluator 6, who found the drag and drop capabilities to be unintuitive and suggested a hand symbol when hovering a drag and drop element as a possible solution.
3. *User control and freedom*

- ✔ Heuristic 3.1 passed.
- ✔ Heuristic 3.2 passed. Evaluator 1 noted that changes to the seed colors can only be reverted via the browser 'undo' button. Evaluator 4 noted that a possible improvement could be to make the progress indicator steps clickable to switch panels with this element as well.

4. *Consistency and standards*

   - ✔ Heuristic 4.1 passed.
   - ✔ Heuristic 4.2 passed.
   - ✔ Heuristic 4.3 passed.
   - ✔ Heuristic 4.4 passed. Evaluator 3 was unclear on what the heuristic is meant to judge.

5. *Error prevention*

   - ✔ Heuristic 5.1 passed.
   - ✖ Heuristic 5.2 passed for all but evaluator 1, who mentioned a bug where the basemap seed color generation runs into an endless loop, when a basemap does not have tiles available for a given zoom level, which requires switching to a different side panel and back to the start panel to get the button to reset and the operation to stop.
   - ✔ Heuristic 5.3 passed.

6. *Recognition rather than recall*

   - ✖ Heuristic 6.1 passed for all but evaluator 1. When generating seed colors with a given hue range, the user needs to remember or look up the corresponding hue degrees manually in the tutorial. Similar remarks were made under heuristic 11.1 by evaluator 4 and 5.
   - ? Heuristic 6.2 passed for all but evaluator 4, who needed multiple trues to get to know the tool without referring the tutorial and recommends showing it automatically on first opening of the website. Also, the tutorial is missing a 'finish' button.

7. *Flexibility and efficiency of use*

   - ✔ Heuristic 7.1 passed.
   - ? Heuristic 7.2 passed for all but evaluator 5, who remarked that both the point layer palette and area layer palette are visible in the export panel even when one was disabled in the previous panels.

8. *Aesthetic and minimalist design*

   - ✔ Heuristic 8.1 passed.
   - ✔ Heuristic 8.2 passed.

9. *Help users recognize, diagnose, and recover from errors*

   - ✔ Heuristic 9.1 passed. Three evaluators marked it as unclear, but only because no errors were encountered.

- ✔ Heuristic 9.2 passed. Three evaluators marked it as unclear, but only because no errors were encountered.

10. *Help and documentation*

- ✔ Heuristic 10.1 passed.
- ✔ Heuristic 10.2 passed.

11. *User stories*

- ✖ Heuristic 11.1 was not passed by evaluator 2 and marked as unclear by evaluator 4, who were either unable to achieve the desired result or only with trial and error. Evaluator 4 connected their troubles with this user story to the problem described here under heuristic 6.1.
- ✔ Heuristic 11.2 passed.
- ✔ Heuristic 11.3 passed. Evaluator 3 mentioned that restricting users to correct options in the sense of a cartographic visualization guide could be helpful. Evaluator 5 needed a moment to realize that the basemap needs to be disabled from view to see the chosen background color.

12. *Other*

    Evaluator 2 stated, that it might be helpful to disallow panning the map on the right side and also remarked about a clickable progress indicator as described under heuristic 3.2.

Heuristics 1.2, 4.4, 9.1, and 9.2 should be rephrased or receive additional explanation in a similar heuristic evaluation, as they were not clear enough for one evaluator, or in the case of heuristics 9.1 and 9.2 multiple evaluators. With the strict definition of marking a heuristic as unclear or not passed as soon as one evaluator marked them as such, 20 heuristics are passing, five are unclear, and three are not passing. Two of the not passing heuristics were marked as such only by evaluator 1 with in-depth knowledge.

The mentioned visible loading times for the 'Imagery' basemap are not directly related to the application, but most likely due to the tile server of that basemap tile service or the internet connection of the evaluator, as the basemaps are the only part of the application that requires internet while using the application. After the heuristic implementation, the suggestion with the hand cursor symbol for heuristic 2.2, when hovering the drag and drop elements, was implemented directly, as well as a grabbing hand symbol on click and also a pointer symbol when hovering the seed color inputs that allow to change the color, as this was a missing indicator as well. Heuristic 5.2 is due to a basemap tile server misfunctioning as well, but could be fixed easier by either deactivating that basemap option or setting a timeout for the function and then displaying an error message. Heuristic 6.1 could be fixed by showing addition information when the control is selected. Heuristic 6.2 can be fixed by implementing the suggested solution with 'JavaScript localStorage', which is similar to a cookie, but places only settings to be used on the client's side and could be as simple as one boolean value to check if the tutorial was viewed before. Heuristic 7.2 can be fixed easily by adding the same controls as for the layer panel and the test panel.

6.4 Sample results

After these two forms of evaluation, sample palettes are generated after the example of randomly chosen ColorBrewer palettes of each of the three palette types. These scenarios are executed as examples of what results are possible when using the tool in its current state. Using ColorBrewer as a benchmark is meant to ensure that MapColPal is able to generate useable color palettes. The seed colors for these comparisons will all be generated using standard settings (6 seed colors with harmoniously spread hues) and were not altered afterwards. The comparisons are done using MapColPal. For the ColorBrewer palettes, they are exported using ColorBrewer's export feature and then imported via the text import of MapColPal. Then the lightness correction and Bézier curve interpolation are deactivated, and all seed colors are used as palette input colors. Full screenshots of the scenarios are in appendix D and allow to see the palettes applied to a sample map each.

Scenario 1: Create a sequential color palette similar to the ColorBrewer YlGnBu palette with 5 classes.

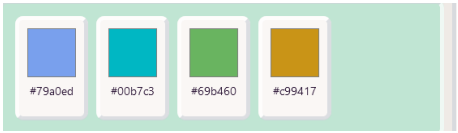


Figure 6.10: Scenario 1 - MapColPal input colors

For this comparison four of the six seed colors were used and arranged from blue over green to yellow hues, as shown in figure 6.10. With the 'sequential' mode selected, smoothing and lightness optimization enabled, and the latter set to a range of '0.4' around '0.8' lightness, the results depicted in figure 6.11 were achieved.

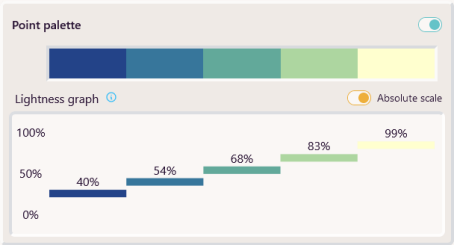


Figure 6.11: Scenario 1 - Sequential MapColPal palette

When comparing this to the ColorBrewer YlGnBu palette, as shown in figure 6.12, the overall look of the palette is very similar, with the ColorBrewer palette being slightly more chromatic. The lightness steps are also similar, with the ColorBrewer palette making bigger steps at the edge of the palette and smaller in the middle, while MapColPal has equally sized lightness steps throughout the whole palette.

The exported MapColPal palette as hexadecimal RGB codes in a JavaScript array is ['#234388', '#37769b', '#62a99a', '#add6a0', '#ffffcf'].

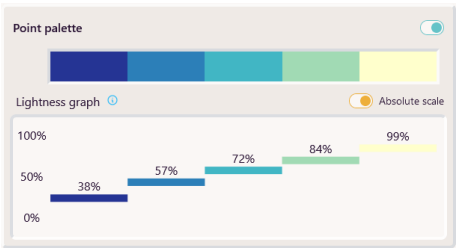


Figure 6.12: Scenario 1 - ColorBrewer YlGnBu palette

Scenario 2: Create a diverging color palette similar to the ColorBrewer PiYG palette with 5 classes.

For this comparison, seed colors were regenerated until a suitable pink color was generated, then only this color was used as an input color for the palette generation. With the 'diverging' mode selected, smoothing disabled and lightness optimization with a range of '0.3' around '0.9' lightness enabled, the results depicted in figure 6.13 were achieved.

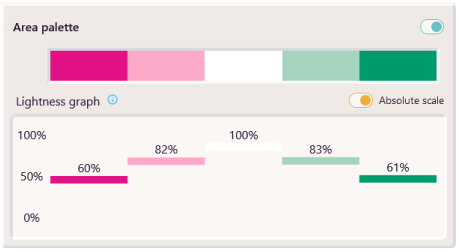


Figure 6.13: Scenario 2 - Diverging MapColPal palette

When compared to the ColorBrewer PiYG palette, as shown in figure 6.14, the palettes are similar overall, with the hues being slightly different due to the randomness of the seed color generation. The palettes appear to be equally chromatic, the lightness steps in the ColorBrewer palette are again bigger on the outer sides of the palette.

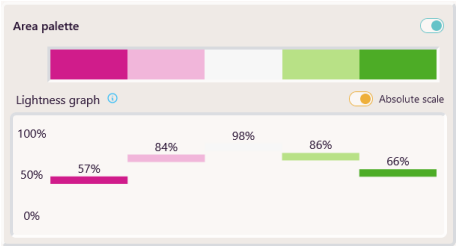


Figure 6.14: Scenario 2 - ColorBrewer PiYG palette

The exported MapColPal palette as hexadecimal RGB codes in a JavaScript array is ['#e21286', '#fca9c9', '#ffffff', '#a6d3bd', '#009b6d'].

Scenario 3: Create a qualitative color palette similar to the ColorBrewer Dark2 palette with 8 classes.

Figure 6.15: Scenario 3 - MapColPal input colors

For the last comparison, eight seed colors were generated to match the number of classes. All eight seed colors were used as input colors and arranged to match the order of the ColorBrewer palette, as shown in figure 6.15. With the 'qualitative' mode selected and lightness optimization enabled, set to a lightness of '0.65', the results depicted in figure 6.16 were achieved.

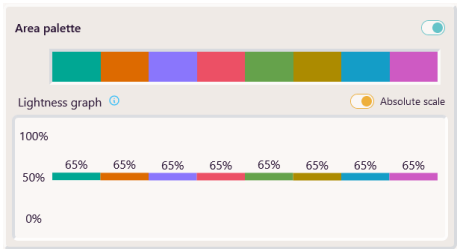


Figure 6.16: Scenario 3 - Qualitative MapColPal palette

In comparison to the ColorBrewer Dark2 palette, as depicted in figure 6.17, there are comparably more differences than in the other scenarios. While the Dark2 palette has varying lightness in a range of almost 30 percent, in MapColPal all palette colors receive the same lightness. Also, as all colors are equally chromatic, colors like the gray in the Dark2 palette are only possible in MapColPal with manual adjustment at the moment.

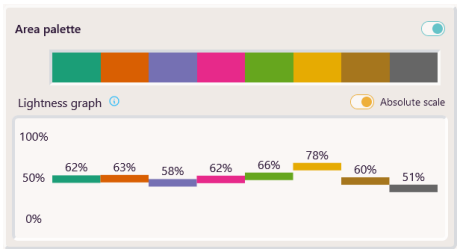


Figure 6.17: Scenario 3 - ColorBrewer Dark2 palette

The exported MapColPal palette as hexadecimal RGB codes in a JavaScript array is ['#00a793', '#dd6a00', '#8a76fc', '#ec5065', '#65a24b', '#ac8b00', '#149dc7', '#ce5ac3'].

CHAPTER 7

Conclusion and outlook

The research objective of this thesis is to **design, build, and evaluate a tool to assist cartographers in choosing suitable color palettes for thematic maps**. To reach this objective, five research questions were phrased and answered within the previous chapters of this thesis.

To answer RQ 1, criteria for suitable color palettes for thematic maps were identified.

To answer RQ 2, criteria, which a cartographic color palette tool should fulfil to improve over previous designs, were identified from a review of related work.

To answer RQ 3, requirements were derived from both kinds of criteria, which can serve as a foundation for software development.

To answer RQ 4, building on these requirements, a proof of concept was coded in an iterative prototyping process.

To answer RQ 5, the proof of concept was evaluated against the requirements that it was designed after, as well as being subject of a heuristic evaluation to further evaluate its user interface and usability, and used to generate color palettes which then were compared to ColorBrewer color palettes.

Based on the conducted evaluation, the proof of concept fulfilled the majority of the requirements and the heuristics. The tool offers cartographers a unique experience in working with colors for maps with options for the design of fitting color palettes that previous tools and pre-defined color palettes do not offer. It provides a new take on the old problem of selecting colors for maps in a way suiting the data, human perception, and aesthetic preferences. The proof of concept tackles this problem by deriving color palettes in a structured way from a shared set of seed colors, visualizing each update immediately, as well as offering user interaction and palette testing at all steps along the process.

The last part of the evaluation showed, that the tool in its current state can be used to achieve ColorBrewer-like results while offering more testing options and considering the user's creative freedom and contextual requirements. The main drawback of the current tool is, that it still requires the user to have in-depth knowledge, as the automated test features of the tool could not be implemented yet. Also control over a color's chroma is limited. This is offset by the color picker functionality which allows for manual adjustments, as long as the user knows what they are doing. Therefore,

the research objective is seen as achieved. Improvement is possible for color generation and testing tools based on MapColPal and further research in this area. Both will be discussed in the following two sections.

Ideas for further development

Apart from the improvement ideas stated in the last chapter, next possible steps in developing the application are:

- Refactoring the code further, adding more comments and documentation, and restructuring the components.
- Saving all application state in the URL, to allow sharing a link which not only contains the seed colors, as well as enabling full 'undo' and 'redo' capabilities using web standards.
- Enable automated chroma (and hue) adjustments and allow for user control over these adjustments.
- Allow for different palette class distributions apart from equal interval classes.
- The import of user data could be enabled to see a possible color palette more closely to the final deployment scenario, which would also allow implementing calculations regarding simultaneous contrast.
- Enabling bivariate color palette generation per map layer.

Further, the following notions specifically for the testing environment exist:

- Example visualization with full map layouts using procedural generation to generate example title, text, and legend elements, as well as the layout itself.
- Support deriving pseudo-random data from seed values to allow for reproduction of the same sample visualization.
- Offer multiple kinds of example visualization per data type, e.g. hexagonal binned data.
- Offer additional monitoring views, e.g. for hue and chroma.

Lastly, for the user interface, the ensuing ideas exist:

- Optimize the mobile view in the responsive design.
- Add a comparison function for direct comparison of multiple color palettes.
- Allow the user to choose the color space the application utilizes.
- Add a custom color picker that also uses the specified color space instead of relying on the browser standard color picker.
- Allow for export within the currently used color space.
- Show filters currently applied to palettes in the export panel.

- Add a toggle to switch between a shortened tutorial and the full-length detailed version.
- Allow easily switching which palette gets assigned to the point layer and which to the area layer.
- Change the automated disappearance of the side panel scrollbar based on further user feedback.
- Allow for basic display options like changing the basemap or switching between country and city scale on all panels.
- Gray out the basemap selection control in the start panel, if the option is not selected.
- Add an educational walkthrough video.
- Add a button to switch the theme to a dark theme.
- Enable option to add or remove layers.

Further research needs

Further development of the MapColPal proof of concept into a fully fleshed out tool should go hand in hand with more extensive evaluation of the tool and the concepts behind it. For this, in-depth user testing is recommendable. This testing could be directed at confirming or falsifying the results of the evaluation conducted within this thesis, or at using the tool for a specific application domain. For example, the suitability of the tool for color decisions in mapping sustainable development goals could be examined. Widening the scope of the tool to cover thematic maps more extensively is another promising direction of research, for example by including considerations for printed maps or other kinds of thematic map. Continuing research could also target transfer of the concepts and criteria the proof of concept was developed on to other map types, for example, developing a color palette helper for topographic maps.

At the same time, I also encourage revitalizing the idea of cartographic brewers in all parts of cartography. I see a lot of potential in using the improving capabilities of the web and web-based applications for help in decision processes in general and especially within cartographic design. The proof of concept could also be further developed into a generalized framework to be used as a base for such applications using the same ideas of bundled user controls and a map view to visualize changes in real-time.

This thesis offers important contributions for working with colors in cartography. The development of the proof of concept can be seen as a first step towards more advanced research and tools in this domain.

References

Aisch, G. (2013). *Mastering multi-hued color scales with chroma.js.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801080621/https://www.vis4.net/blog/2013/09/mastering-multi-hued-color-scales/

Aisch, G. (2018). *I wrote some code that automatically checks visualizations for non-colorblind safe colors. here's how it works.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801082159/https://www.vis4.net/blog/2018/02/automate-colorblind-checking/

Aisch, G. (2019a). *Chroma.js color palette helper: A tool for creating nice, percerptually correct and colorblind-safe color palettes. https://github.com/gka/palettes.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801075558/https://github.com/gka/palettes

Aisch, G. (2019b). *Update for the chroma.js palette helper.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801080840/https://www.vis4.net/blog/2019/06/chroma-js-color-palette-tool-update/

Arnowitz, J., Arent, M., & Berger, N. (2007). *Effective prototyping for software makers* (1st ed.). Elsevier.

Bähr, B. (2017). *Prototyping of user interfaces for mobile applications*. Cham: Springer International Publishing.

Berenbach, B., Paulish, D. J., Kazmeier, J., & Rudorfer, A. (2009). *Software & systems requirements engineering: In practice*. New York, NY, USA: McGraw-Hill.

Bertin, J. (2011). *Semiology of graphics: Diagrams, networks, maps: Translated by william j. berg* (1st ed., Vol. 62). Redlands, Calif.: Esri Press.

Bishop, E. (2011). *Poems* (1st ed.). New York: Farrar Straus and Giroux.

Bleicher, S. (2012). *Contemporary color: Theory & use* (2nd ed.). Clifton Park, NY: Delmar Cengage Learning.

Boronine, A. (2012). *Color spaces for human beings.* Retrieved 16.09.2022, from https://web.archive.org/web/20220610174227/https://www.boronine.com/2012/03/26/Color-Spaces-for-Human-Beings/

Brewer, C. A. (1991). *The prediction of surround-induced changes in map color appearance* (Unpublished doctoral dissertation). Michigan State University.

Brewer, C. A. (1997). Evaluation of a model for predicting simultaneous contrast on color maps. *The Professional Geographer*, *49*(3), 280–294.

Brewer, C. A. (2003). A transition in improving maps: The colorbrewer example. *Cartography and Geographic Information Science*, *30*(2), 159–162.

Brewer, C. A. (2006). *Updates to colorbrewer resources.* Retrieved 17.09.2022, from https://web.archive.org/web/20220602090428/http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_updates.html

Brewer, C. A. (2016). *Designing better maps: A guide for gis users* (2nd ed.). Redlands, California: Esri Press.

Brown, A., & Feringa, W. (2003). *Colour basics for gis users*. Harlow: Prentice Hall.

Bujack, R., Turton, T. L., Samsel, F., Ware, C., Rogers, D. H., & Ahrens, J. (2018). The good, the bad, and the ugly: A theoretical framework for the assessment of continuous colormaps. *IEEE transactions on visualization and computer graphics*, *24*(1), 923–933.

CIE. (2020). *International lighting vocabulary (ilv): Cie s 017/e:2020* (2nd ed.). International Commission on Illumination (CIE).

Colorbrewer source code. (2021). Retrieved 17.09.2022, from https://github.com/axismaps/colorbrewer/

Crameri, F., Shephard, G. E., & Heron, P. J. (2020). The misuse of colour in science communication. *Nature communications*, *11*(1).

Crowder, J. A., & Hoff, C. W. (2022). *Requirements engineering: Laying a firm foundation* (1st ed. 2022 ed.). Cham: Springer International Publishing and Imprint Springer.

Dick, J., Hull, E., & Jackson, K. (2017). *Requirements engineering* (4th ed.). Cham: Springer International Publishing.

Ekroll, V., & Faul, F. (2013). Transparency perception: the key to understanding simultaneous color contrast. *Journal of the Optical Society of America. A, Optics, image science, and vision*, *30*(3), 342–352.

Evans, R. M. (1964). Variables of perceived color. *Journal of the Optical Society of America*, *54*(12), 1467–1474.

Fairchild, M. D. (2013). *Color appearance models* (3rd ed.). Chichester, West Sussex: Wiley.

Fernandez, S. R., Fairchild, M. D., & Braun, K. (2005). Analysis of observer and cultural variability while generating "preferred" color reproductions of pictorial images. *Journal of Imaging Science and Technology*, *49*(1), 96–104.

Gardner, S. D. (2005). *Evaluation of the colorbrewer color schemes for accommodation of map readers with impaired color vision* (Unpublished doctoral dissertation). Pennsylvania State University.

Gramazio, C. C. (2016). *Colorgorical source code.* Retrieved 17.09.2022, from https://github.com/connorgr/colorgorical

Gramazio, C. C., Laidlaw, D. H., & Schloss, K. B. (2017). Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE transactions on visualization and computer graphics*, *23*(1), 521–530.

Gresh, D. L. (2008). *Self-corrected perceptual colormaps* (No. RC24542).

Harrower, M., & Brewer, C. A. (2003). Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, *40*(1), 27–37.

Hart, C. (2009). *Doing a literature review: Releasing the social science research imagination*. London: Sage.

Heer, J., & Stone, M. (2012). Color naming models for color selection, image editing and palette design. In J. A. Konstan (Ed.), *Proceedings of the sigchi conference on human factors in computing systems.* ACM. Retrieved from http://portal.acm.org/citation.cfm?id=2207676&CFID=94747092&CFTOKEN=43686596

Hollingsed, T., & Novick, D. G. (2007). Usability inspection methods after 15 years of research and practice. In D. Novick (Ed.), *Proceedings of the 25th annual acm international conference on design of communication* (pp. 249–255). ACM.

Holtzschue, L. (2017). *Understanding color: An introduction for designers* (5th ed.). Hoboken: Wiley.

Hoop, S., Oosterom, P., & Molenaar, M. (1993). Topological querying of multiple map layers. In G. Goos, J. Hartmanis, A. U. Frank, & I. Campari (Eds.), *Spatial*

information theory a theoretical basis for gis (Vol. 716, pp. 139–157). Springer Berlin Heidelberg.

Hu, G., Pan, Z., Zhang, M., Chen, Yang, W., & Chen, J. (2012). An interactive method for generating harmonious color schemes. *Color Research & Application*, *39*(1), 70–78.

Hu, G., Zhang, M., Pan, Z., Lin, L., Rhalibi, A. E. L., & Song, J. (2013). A user-oriented method for preferential color scheme generation. *Color Research & Application*, *40*(2), 147–156.

Hunt, R. W. G., & Pointer, M. (2011). *Measuring colour* (4th ed.). Chichester: Wiley.

Itten, J. (1970). *The elements of color: A treatise on the color system of johannes itten, based on his book the art of color*. New York: Van Nostrand Reinhold Co.

Itten, J. (1973). *The art of color: The subjective experience and objective rationale of color*. New York, NY: Van Nostrand Reinhold.

Kammerer, K., Göster, M., Reichert, M., & Pryss, R. (2021). Ambalytics: A scalable and distributed system architecture concept for bibliometric network analyses. *Future Internet*, *13*(8), 203.

Kraak, M.-J., Roth, R. E., Ricker, B., Kagawa, A., & Le Sourd, G. (2020). *Mapping for a sustainable world*.

Laplante, P. A. (2018). *Requirements engineering for software and systems* (Third edition ed.). Boca Raton: CRC Press Taylor & Francis Group.

Levkowitz, H. (1996). Perceptual steps along color scales. *International Journal of Imaging Systems and Technology*, *7*, 97–101.

Lindner, A. J., & Süsstrunk, S. (2013). Automatic color palette creation from words. In *Proceedings of the is&t 21st color and imaging conference* (pp. 69–74). Retrieved from https://api.semanticscholar.org/CorpusID:15979145

Lu, K., Feng, M., Chen, X., Sedlmair, M., Deussen, O., Lischinski, D., … Wang, Y. (2021). Palettailor: Discriminable colorization for categorical data. *IEEE transactions on visualization and computer graphics*, *27*(2), 475–484.

Lu, M., Lanir, J., Wang, C., Yao, Y., Zhang, W., Deussen, O., & Huang, H. (2022). Modeling just noticeable differences in charts. *IEEE transactions on visualization and computer graphics*, *28*(1), 718–726.

Machado, G. M., Oliveira, M. M., & Fernandes, L. A. F. (2009). A physiologically-based model for simulation of color vision deficiency. *IEEE transactions on visualization and computer graphics*, *15*(6), 1291–1298.

Matsuda, Y. (1995). *Color design*. Asakura Shoten.

McManus, I. C., Jones, A. L., & Cottrell, J. (1981). The aesthetics of colour. *Perception*, *10*, 651–666.

Meier, B. J. (1988). Ace: A color expert system for user interface design. In M. Green (Ed.), *Proceedings of the 1st annual acm siggraph symposium on user interface software* (pp. 117–128). ACM.

Meier, B. J., Spalter, A. M., & Karelitz, D. B. (2004). Interactive color palette tools. *IEEE computer graphics and applications*, *24*(3), 64–72.

Muehlenhaus, I. (2014). *Web cartography: Map design for interactive and mobile devices*. Boca Raton, FL: CRC Press.

Muñoz, R., Miller-Jacobs, H. H., Spool, J. M., & Verplank, B. (1992). In search of the ideal prototype. In *Chi '92: Proceedings of the sigchi conference on human factors in computing systems* (pp. 577–579).

Nielsen, J. (1994a). *How to conduct a heuristic evaluation.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801112221/https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

Nielsen, J. (1994b). *Summary of usability inspection methods.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801112114/https://www.nngroup.com/articles/summary-of-usability-inspection-methods/

Nielsen, J. (2020). *10 usability heuristics for user interface design.* Retrieved 01.08.2022, from https://web.archive.org/web/20220801111113/https://www.nngroup.com/articles/ten-usability-heuristics/

Nuñez, J. R., Anderton, C. R., & Renslow, R. S. (2018). Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PloS one*, *13*(7), e0199239.

O'Donovan, P., Agarwala, A., & Hertzmann, A. (2011). Color compatibility from large datasets. In H. Hoppe (Ed.), *Acm siggraph 2011 papers.* ACM.

Ottosson, B. (2020). *A perceptual color space for image processing.* Retrieved 15.09.2022, from https://web.archive.org/web/20220915200056/https://bottosson.github.io/posts/oklab/

Ottosson, B. (2021). *Okhsv and okhsl: Two new color spaces for color picking.* Retrieved 18.09.2022, from https://web.archive.org/web/20220918115450/https://bottosson.github.io/posts/colorpicker/

Peng, Y.-F., & Chou, T.-R. (2019). Automatic color palette design using color image and sentiment analysis. In *2019 ieee 4th international conference on cloud computing and big data analysis (icccbda)* (pp. 389–392). IEEE.

Petroff, M. A. (2021). *Accessible color sequences for data visualization.* Retrieved from http://arxiv.org/pdf/2107.02270v2

Post, D. L., & Goode, W. E. (2020). Palette designer: A color-code design tool. *Displays*, *61*.

Robertson, P., & O'Callaghan, J. (1986). The generation of color sequences for univariate and bivariate mapping. *IEEE Computer Graphics and Applications*, *6*(2), 24–32.

Rossiter, D. G. (2018a). *Research concepts & skills - volume 1: Concepts: Version 3.4.*

Rossiter, D. G. (2018b). *Research concepts & skills - volume 2: Skills: Version 3.7.*

Samson, L. (1985). *Graphic design with color using a knowledge base* (Unpublished doctoral dissertation). Simon Fraser University.

Schloss, K. B., & Palmer, S. E. (2011). Aesthetic response to color combinations: preference, harmony, and similarity. *Attention, perception & psychophysics*, *73*(2), 551–571.

Shamoi, P., Inoue, A., & Kawanaka, H. (2014). Perceptual color space: Motivations, methodology, applications. In *2014 joint 7th international conference on soft computing and intelligent systems (scis) and 15th international symposium on advanced intelligent systems (isis)* (pp. 1354–1359). IEEE.

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 ieee symposium on visual languages* (pp. 336–343). IEEE Comput. Soc. Press.

Shugrina, M., Lu, J., & Diverdi, S. (2017). Playful palette: An interactive parametric color mixer for artists. *ACM Transactions on Graphics*, *36*(4), 1–10.

Silva, S., Sousa Santos, B., & Madeira, J. (2011). Using color in visualization: A survey. *Computers & Graphics*, *35*(2), 320–333.

Smart, S., Wu, K., & Szafir, D. A. (2020). Color crafting: Automating the construction of designer quality color ramps. *IEEE transactions on visualization and computer graphics*, *26*(1), 1215–1225.

Sniurevicius, Z. (2022). *Extracting a color palette from an image with javascript.* Retrieved

19.09.2022, from https://web.archive.org/web/20220918235320/https://dev.to/producthackers/creating-a-color-palette-with-javascript-44ip

Sochorová, Š., & Jamriška, O. (2021). Practical pigment mixing for digital painting. *ACM Transactions on Graphics*, *40*(6), 1–11.

Sommerville, I. (2016). *Software engineering: Global edition* (10th ed.). Boston and Columbus and Indianapolis and New York and San Francisco and Hoboken and Amsterdam and Cape Town and Dubai and London and Madrid and Milan and Munich and Paris and Montreal and Toronto and Delhi and Mexico City and São Paulo and Sydney and Hong Kong and Seoul and Singapore and Taipei and Tokyo: Pearson.

Sommerville, I. (2021). *Engineering software products: An introduction to modern software engineering: Global edition*. Hoboken, NJ: Pearson.

Stone, M. (2012). In color perception, size matters. *IEEE Computer Graphics and Applications*, *32*(2), 6–11.

Stone, M., Szafir, D. A., & Setlur, V. (2014). An engineering model for color difference as a function of size. In *Color and imaging conference* (pp. 253–258). Retrieved from https://api.semanticscholar.org/CorpusID:11337646

Szafir, D. A. (2018). Modeling color difference for visualization design. *IEEE transactions on visualization and computer graphics*, *24*(1), 392–401.

Tan, J., Echevarria, J., & Gingold, Y. (2018). *Palette-based image decomposition, harmonization, and color transfer.*

Tokumaru, M., Muranaka, N., & Imanishi, S. (2002). Color design support system considering color harmony. In *Fuzz-ieee'02* (pp. 378–383). IEEE.

van Lamsweerde, A. (2009). *Requirements engineering: From system goals to uml models to software specifications*. Hoboken, N.J. and Chichester: Wiley.

Waldin, N., Waldner, M., Le Muzic, M., Gröller, E., Goodsell, D. S., Autin, L., … Viola, I. (2019). Cuttlefish: Color mapping for dynamic multi-scale visualizations. *Computer Graphics Forum*, *38*(6), 150–164.

Warfel, T. Z. (2009). *Prototyping: A practitioner's guide*. Brooklyn, NY: Rosenfeld Media.

Weninger, B. (2015). A framework for color design of digital maps: An example of noise maps. In J. Brus, A. Vondrakova, & V. Vozenilek (Eds.), *Modern trends in cartography* (pp. 103–116). Springer International Publishing.

Wijffelaars, M., Vliegen, R., van Wijk, J. J., & van der Linden, E.-J. (2008). Generating color palettes using intuitive parameters. *Computer Graphics Forum*, *27*(3), 743–750.

Wilson, C. (2014). *User interface inspection methods: A user-centered design method*. Waltham, MA, USA: Morgan Kaufmann.

Yuan, L., Zhou, Z., Zhao, J., Guo, Y., Du, F., & Qu, H. (2021). Infocolorizer: Interactive recommendation of color palettes for infographics. *IEEE transactions on visualization and computer graphics*, *PP*.

Zave, P. (1997). Classification of research efforts in requirements engineering. In *Proceedings of the third ieee international symposium on requirements engineering, january 6-10, 1997, annapolis, maryland, usa* (pp. 315–321). IEEE Computer Society Press.

Zeileis, A., Fisher, J. C., Hornik, K., Ihaka, R., McWhite, C. D., Murrell, P., … Wilke, C. O. (2020). colorspace : A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software*, *96*(1).

Zeileis, A., Hornik, K., & Murrell, P. (2009). Escaping rgbland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, *53*(9), 3259–3270.

Zheng, Q., Lu, M., Wu, S., Hu, R., Lanir, J., & Huang, H. (2022). Image-guided color

mapping for categorical data visualization. *Computational Visual Media*, *8*(4), 613–629.

Zhou, L., & Hansen, C. D. (2016). A survey of colormaps in visualization. *IEEE transactions on visualization and computer graphics*, *22*(8), 2051–2069.

APPENDIX A

Heuristics form

Table A.1: Heuristics form

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
example	Example	example	This is an example.	unclear	Passed should be 'yes', 'unclear' or 'no'. Further comments in this column. Definitely comment when 'no' or 'unclear' to clarify why.
1	Visibility of system status	1.0			General comments on visibility of system status here.
1	Visibility of system status	1.1	The user always sees which side panel page they are on and which comes next.		
1	Visibility of system status	1.2	Seed colors, input colors and palette colors are always visualized when mentioned.		
1	Visibility of system status	1.3	Changes to the app settings are immediately applied visually.		
1	Visibility of system status	1.4	On color import and export, the user receives a short notification.		
2	Match between system and the real world	2.0			General comments on match between system and the real world here.

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
2	Match between system and the real world	2.1	The app uses easy-to-understand language familiar to cartographers.		
2	Match between system and the real world	2.2	User interface elements fit the intuitive expectations set to them (e.g. a toggle toggles something on and off).		
3	User control and freedom	3.0			General comments on user control and freedom here.
3	User control and freedom	3.1	The user can adjust every setting as often as they want.		
3	User control and freedom	3.2	The user can go back and forth between side panel pages in case they want to reset a previous setting.		
4	Consistency and standards	4.0			General comments on consistency and standards here.
4	Consistency and standards	4.1	The user interface follows industry conventions (e.g. a button looks like a button).		

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
4	Consistency and standards	4.2	The user interface within the app remains consistent (e.g. buttons for the same action look the same).		
4	Consistency and standards	4.3	Each setting has a default option pre-selected.		
4	Consistency and standards	4.4	The app doesn't use unlabeled icons.		
5	Error prevention	5.0			General comments on error prevention here.
5	Error prevention	5.1	No critical (app-breaking) errors can occur.		
5	Error prevention	5.2	No errors without warning messages or that can't be undone can occur.		
5	Error prevention	5.3	User interface elements are big enough to be easy to interact with.		
6	Recognition rather than recall	6.0			General comments on recognition rather than recall here.

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
6	Recognition rather than recall	6.1	The user doesn't need to remember required information, important information for a given decision or action is visible in the interface.		
6	Recognition rather than recall	6.2	Help is offered in context, not only a tutorial is available.		
7	Flexibility and efficiency of use	7.0			General comments on flexibility and efficiency of use here.
7	Flexibility and efficiency of use	7.1	The side panel page can be changed by using the keyboard arrow keys.		
7	Flexibility and efficiency of use	7.2	The user interface is conditional and only shows information relevant to the currently selected settings.		
8	Aesthetic and minimalist design	8.0			General comments on aesthetic and minimalist design here.
8	Aesthetic and minimalist design	8.1	The user interface is focused on essential features and information.		

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
8	Aesthetic and minimalist design	8.2	The most important features are prioritized and highlighted, there is a clear visual hierarchy.		
9	Help users recognize, diagnose, and recover from errors	9.0			General comments on help for error recognition, diagnose and recovery here.
9	Help users recognize, diagnose, and recover from errors	9.1	Errors are communicated with a visible, red notification.		
9	Help users recognize, diagnose, and recover from errors	9.2	Error messages are formulated in easy-to-understand language without technical jargon.		
10	Help and documentation	10.0			General comments on help and documentation here.
10	Help and documentation	10.1	Aside the contextual help (see 6.2), there is also a help documentation which can be used to quickly find more information on a specific topic.		

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
10	Help and documentation	10.2	The help offers concrete advice (e.g. steps to be carried out).		
11	User stories	11.0			General comments on the user stories here.
11	User stories	11.1	Attempt the following and comment on your experience: Export a sequential palette using only red to yellow hues.		
11	User stories	11.2	Attempt the following and comment on your experience: Export a diverging color palette featuring 5 classes for a point symbol map on city level.		
11	User stories	11.3	Attempt the following and comment on your experience: Export a qualitative color palette featuring 6 classes for an choropleth map on country level. Display in front of a dark background.		

Continued on next page

Table A.1: Heuristics form - continued

Category Nr.	Category	Heuristic Nr.	Heuristic	Passed	Comments
12	Other	12.0			General comments on anything else here.

APPENDIX B

Screenshots of intermediate prototypes

B.1 Wireframe

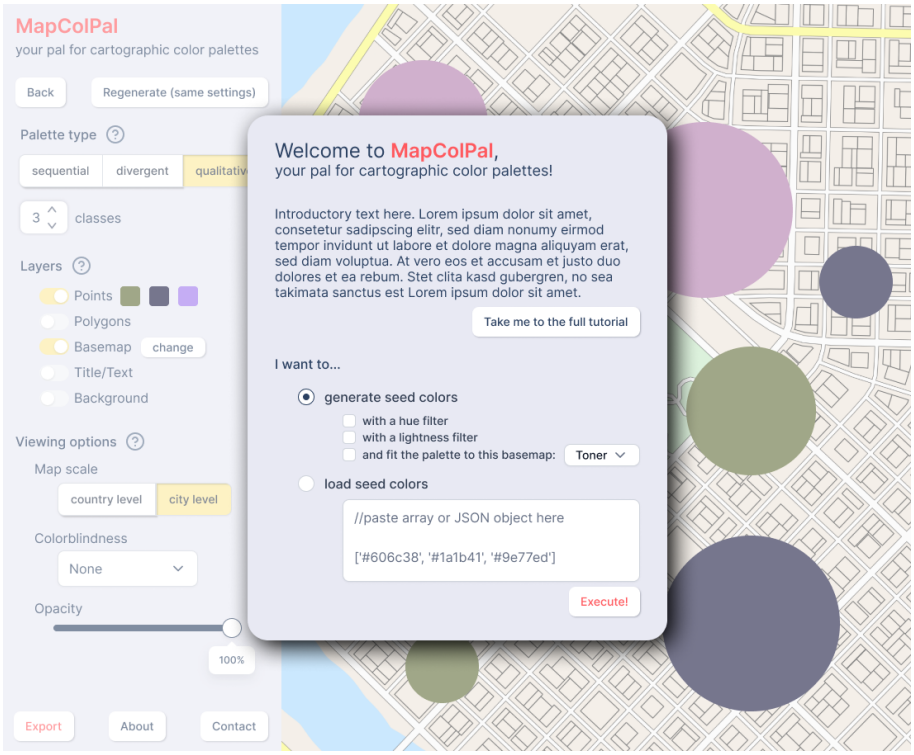


Figure B.1: Wireframe - Start view

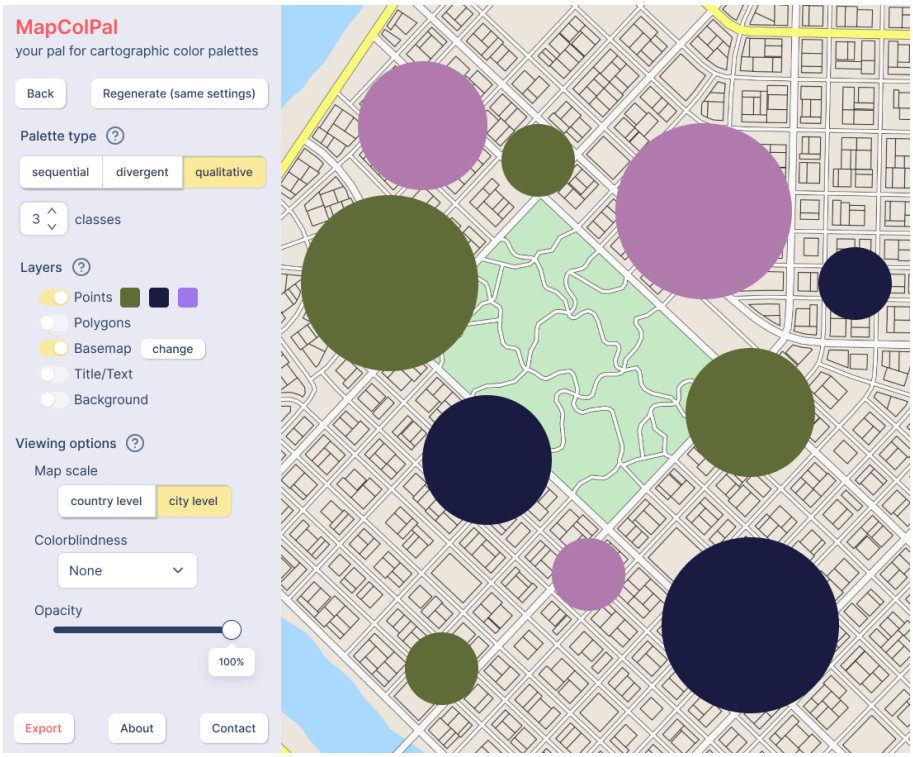


Figure B.2: Wireframe - Main view

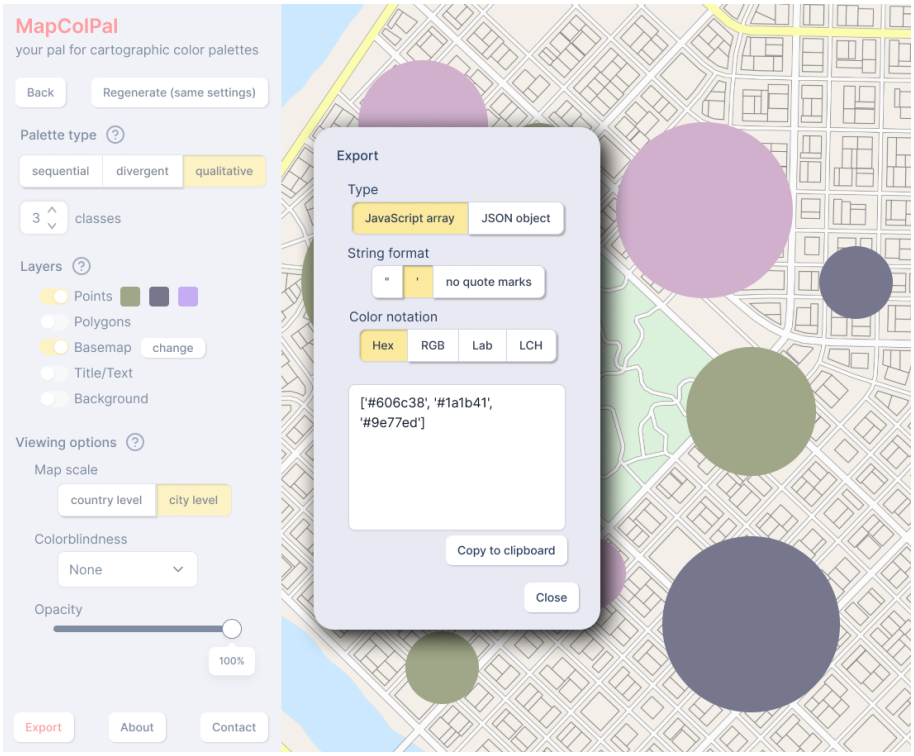


Figure B.3: Wireframe - Export view

B.2 Technical capability test

Figure B.4: Technical capability test

B.3 First coded user interface prototype

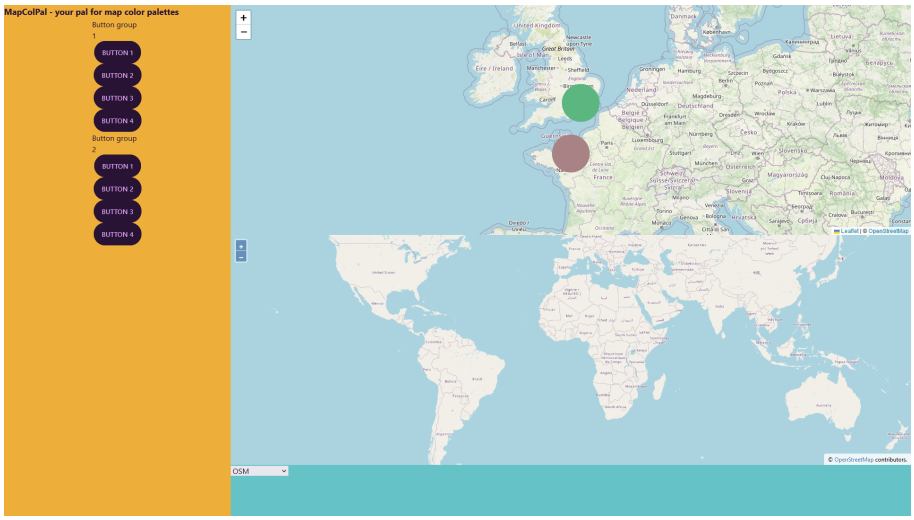


Figure B.5: First coded user interface prototype

B.4 Proof of concept implementation - First merge

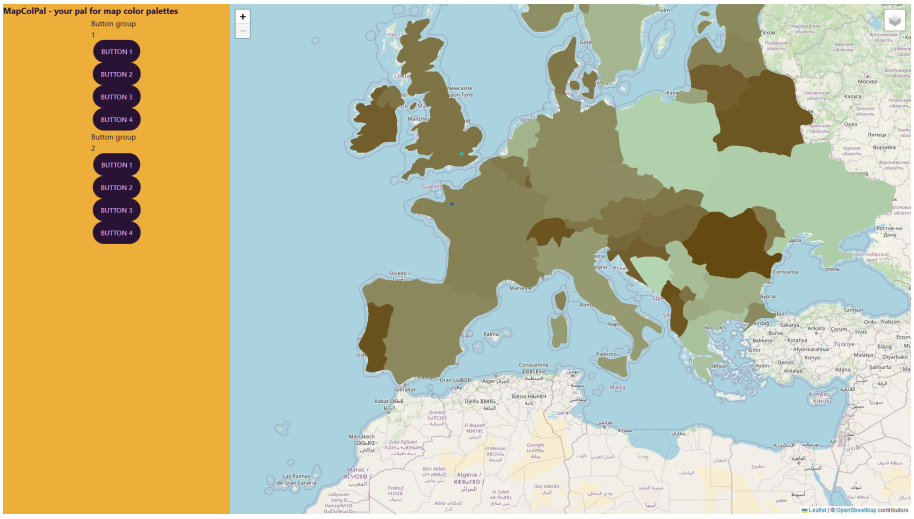


Figure B.6: Implementation - First merge

B.5 Proof of concept implementation - Third merge

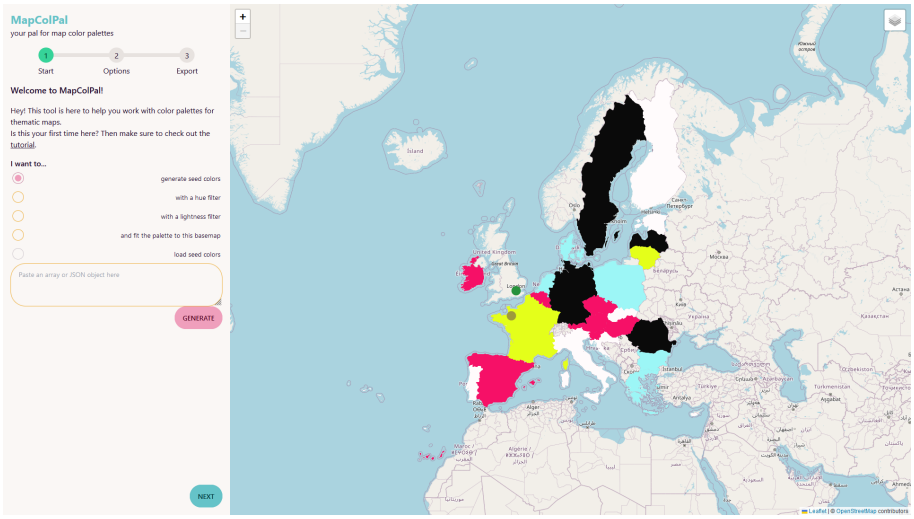


Figure B.7: Implementation - Third merge - Start panel

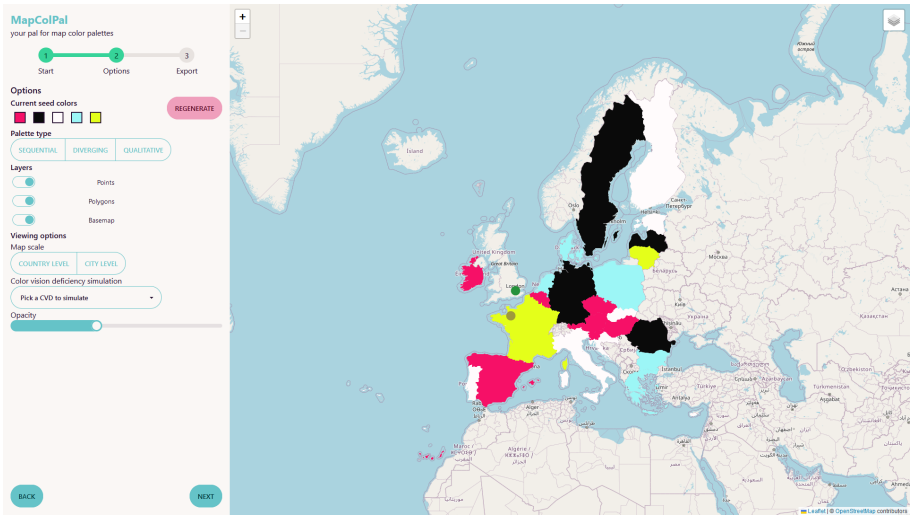


Figure B.8: Implementation - Third merge - Options panel

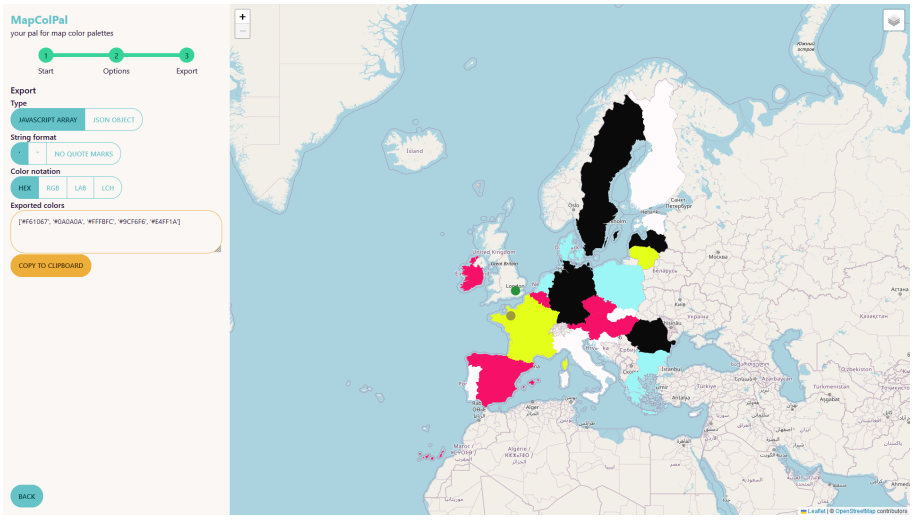


Figure B.9: Implementation - Third merge - Export panel

B.6 Proof of concept implementation - Fifth merge

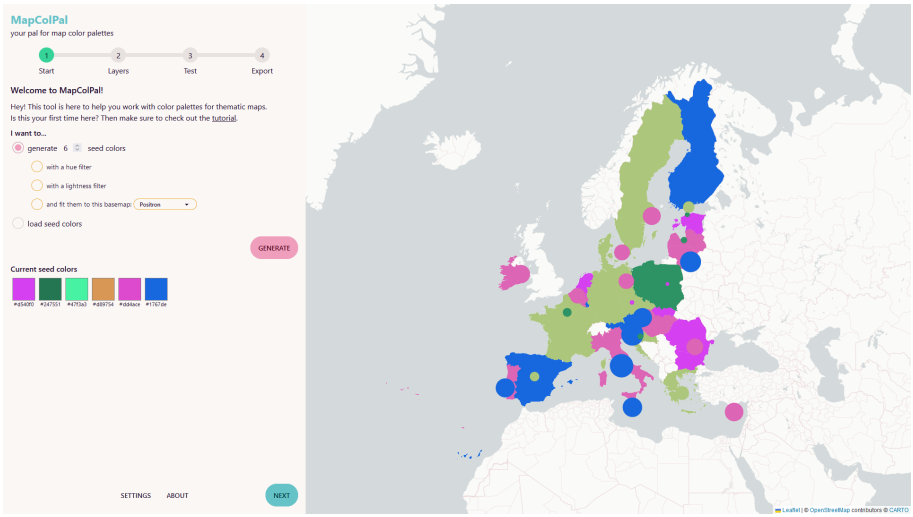


Figure B.10: Implementation - Fifth merge - Start panel

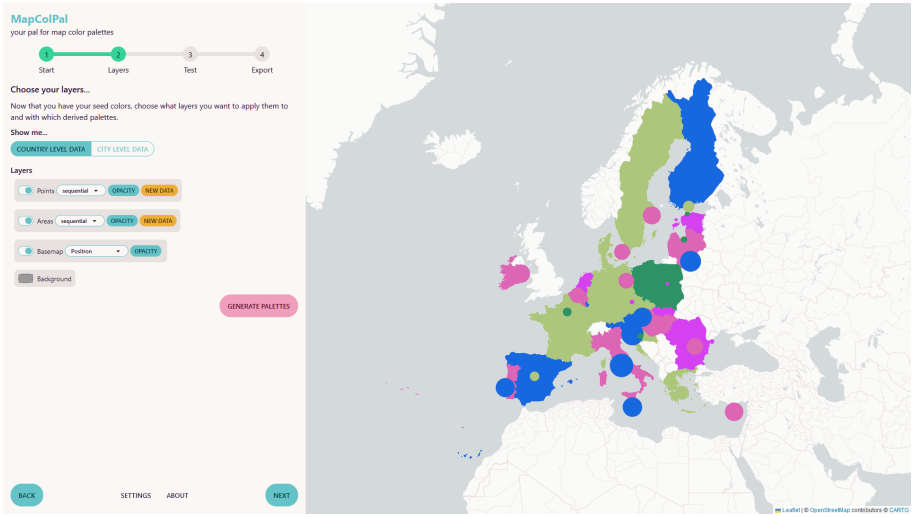


Figure B.11: Implementation - Fifth merge - Layers panel

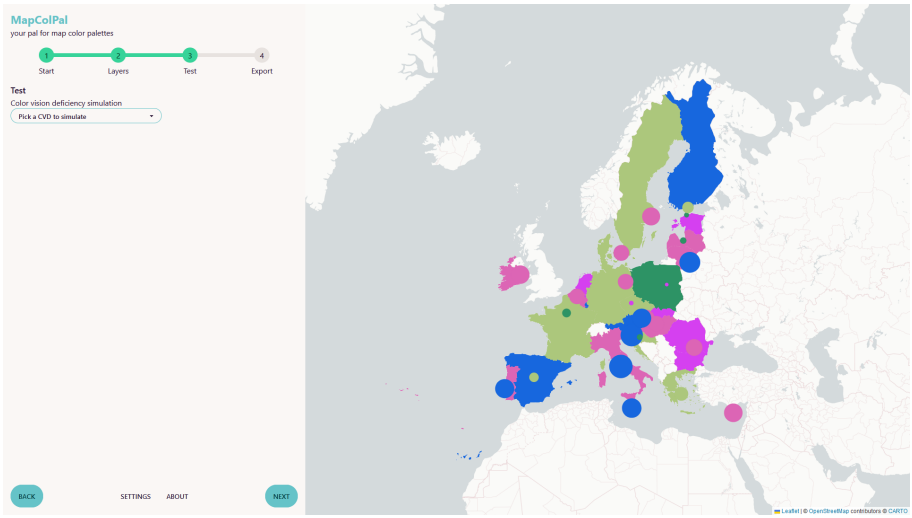


Figure B.12: Implementation - Fifth merge - Test panel

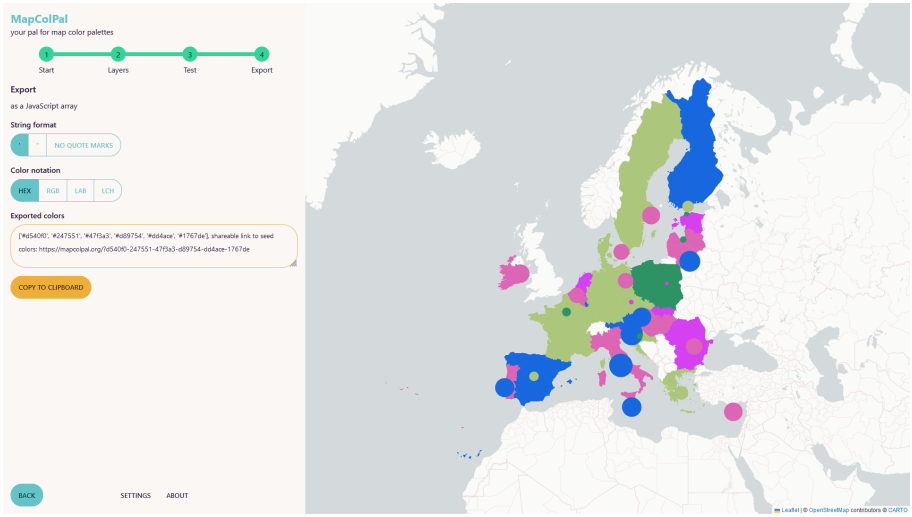


Figure B.13: Implementation - Fifth merge - Export panel

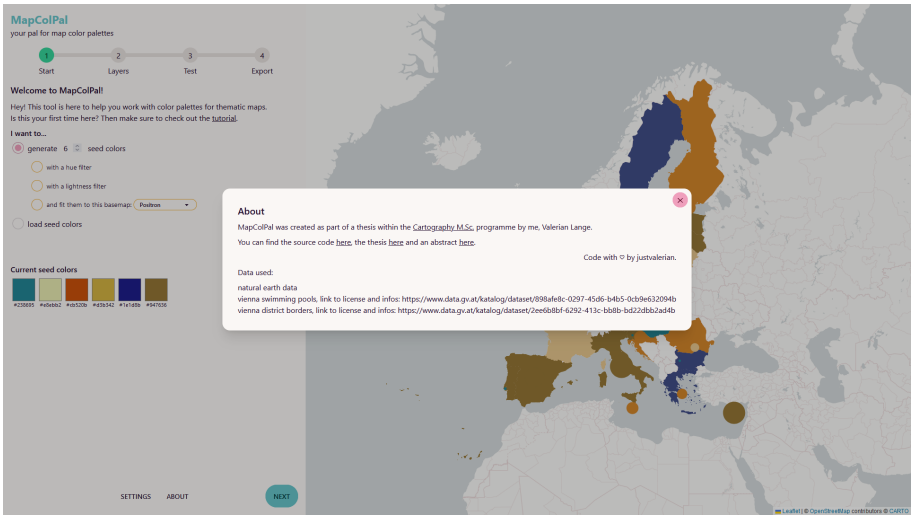


Figure B.14: Implementation - Fifth merge - About modal

B.7 Proof of concept implementation - Seventh merge

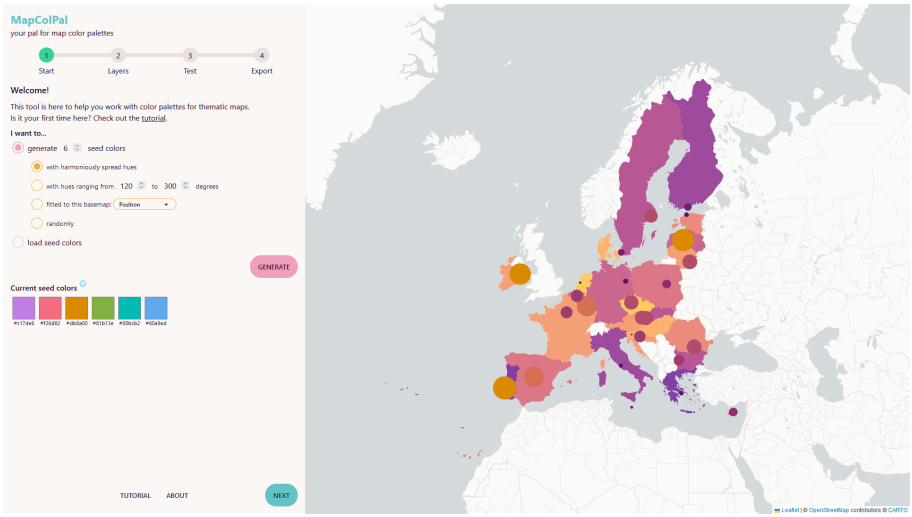


Figure B.15: Implementation - Seventh merge - Start panel

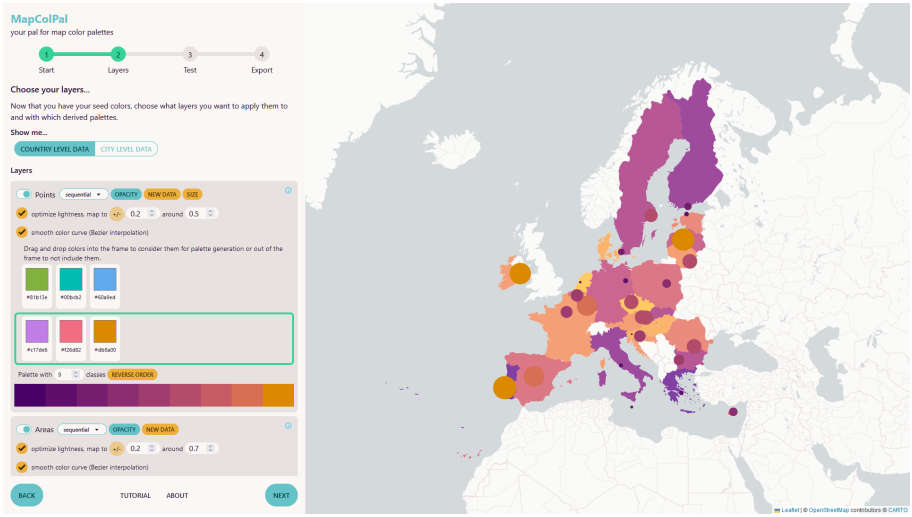


Figure B.16: Implementation - Seventh merge - Layers panel

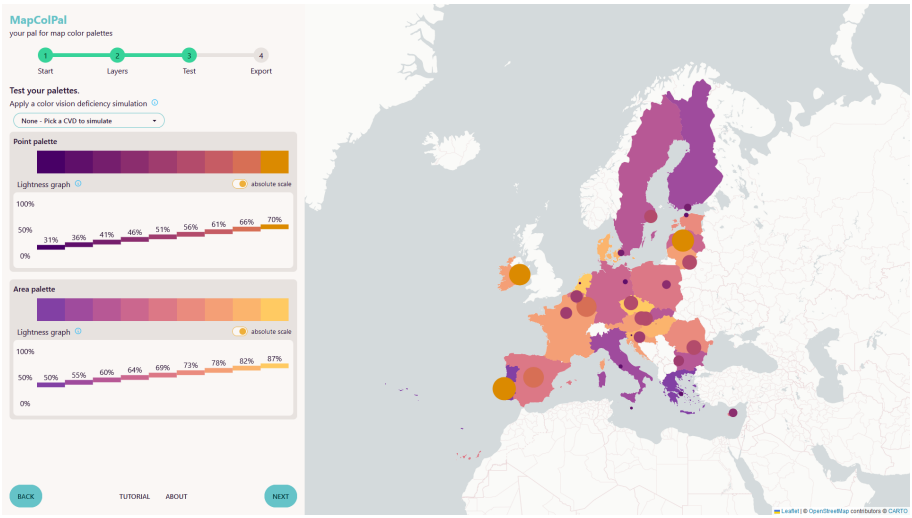


Figure B.17: Implementation - Seventh merge - Test panel

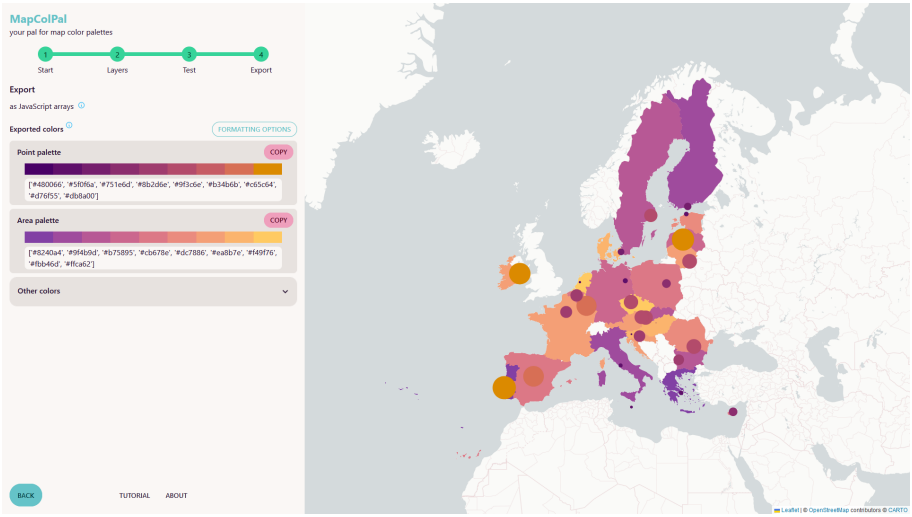


Figure B.18: Implementation - Seventh merge - Export panel

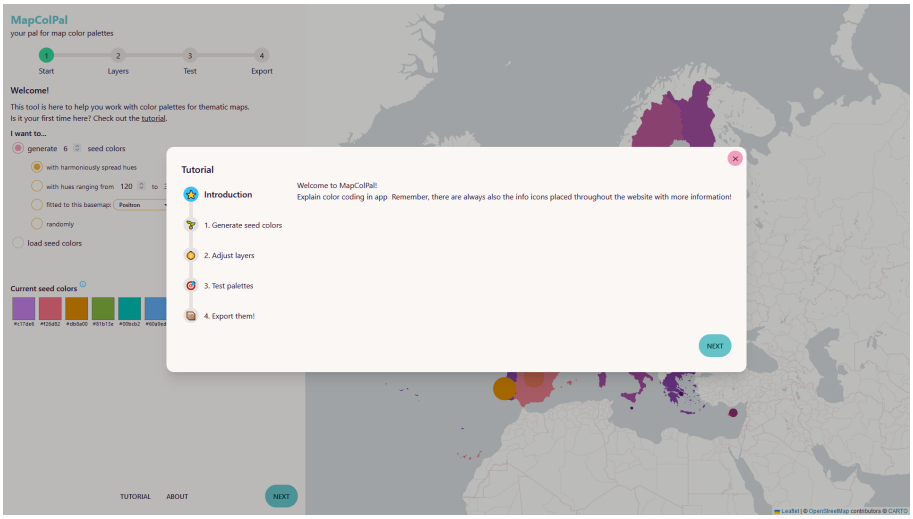


Figure B.19: Implementation - Seventh merge - Tutorial modal

APPENDIX C

Proof of concept - Responsive views

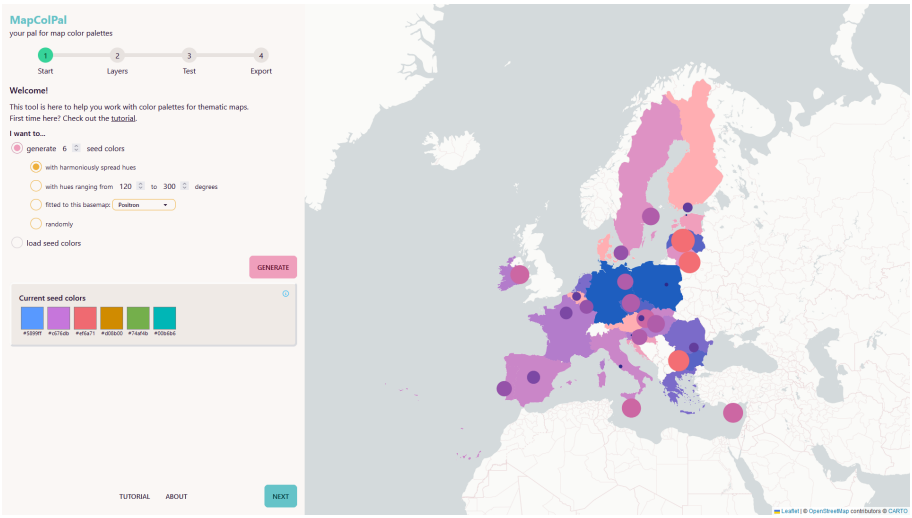


Figure C.1: Proof of concept - Main view on a full HD screen

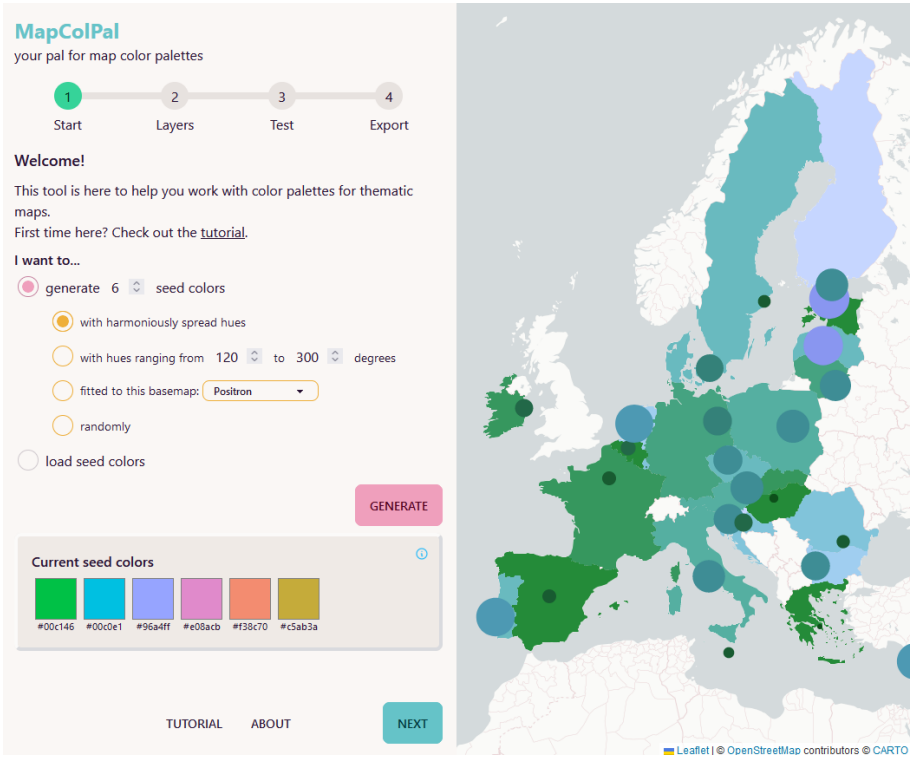


Figure C.2: Proof of concept - Main view on a tablet screen

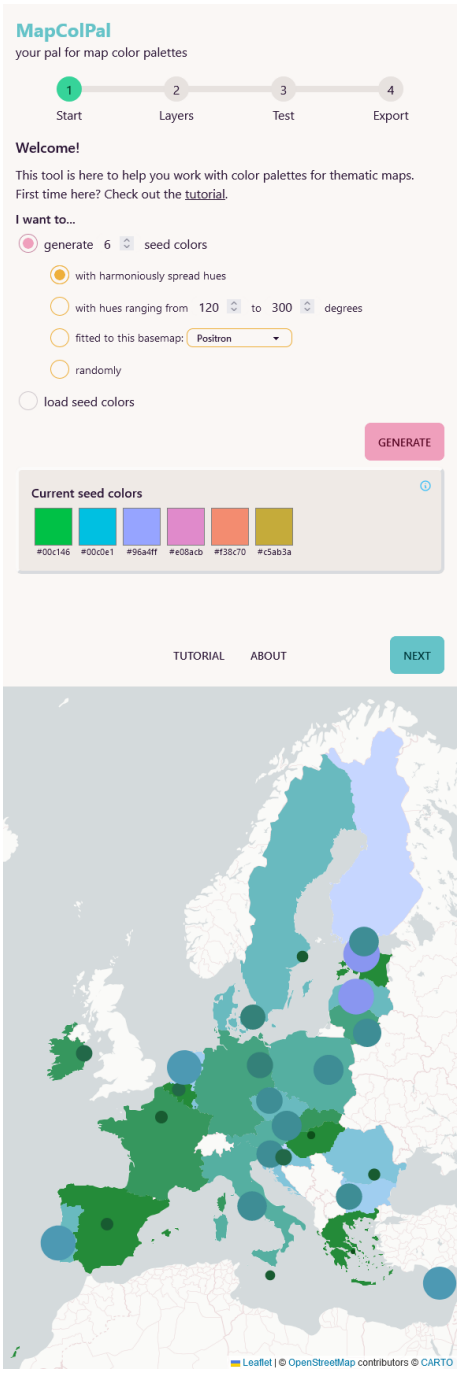


Figure C.3: Proof of concept - Main view on a small screen

APPENDIX D

Screenshots of usage scenarios

D.1 Scenario 1 - sequential palette generation

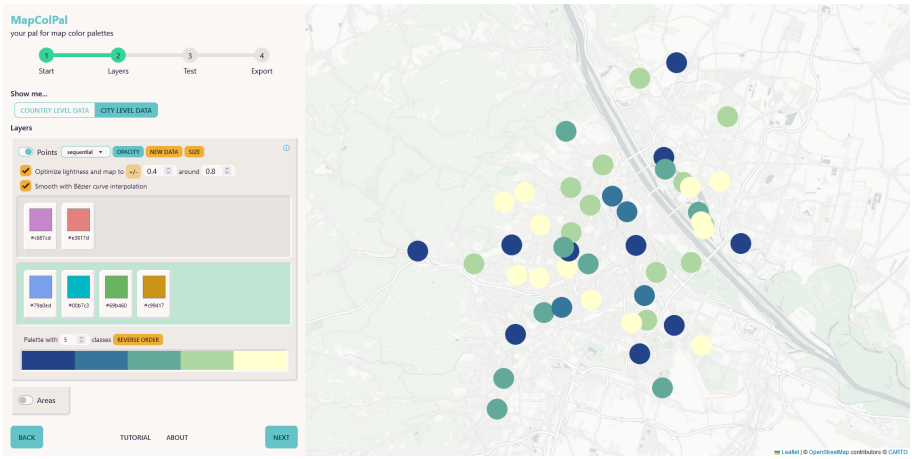


Figure D.1: Scenario 1 - MapColPal sequential palette - Full screenshot

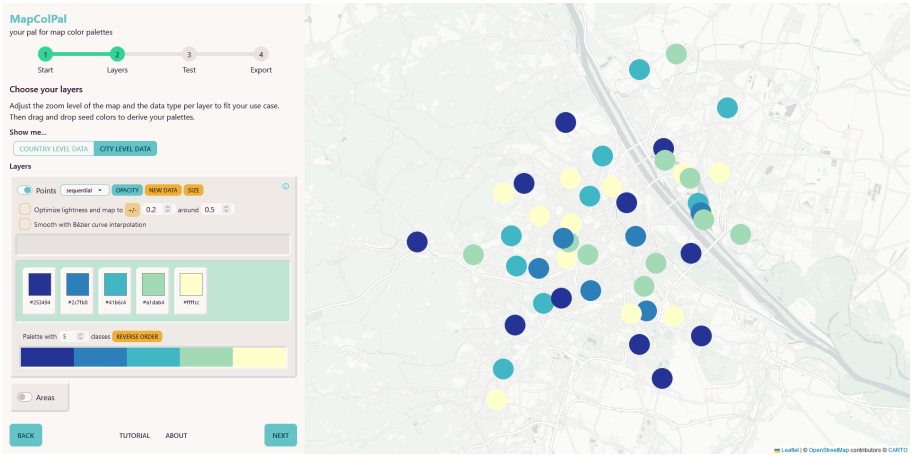


Figure D.2: Scenario 1 - ColorBrewer YlGnBu palette - Full screenshot

D.2 Scenario 2 - diverging palette generation

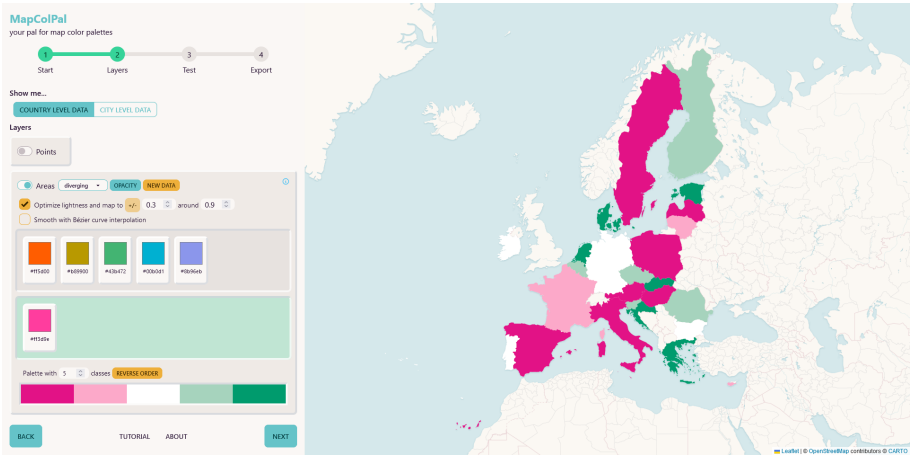


Figure D.3: Scenario 2 - MapColPal diverging palette - Full screenshot

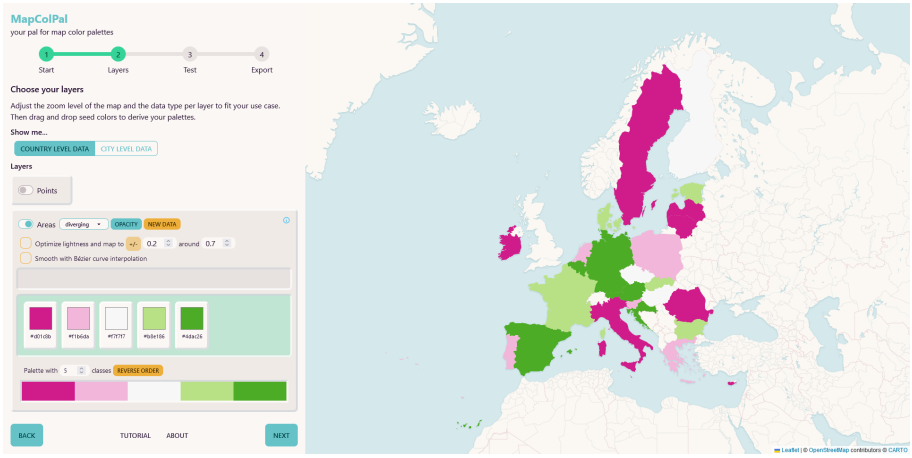


Figure D.4: Scenario 2 - ColorBrewer PiYG palette - Full screenshot

D.3 Scenario 3 - qualitative palette generation

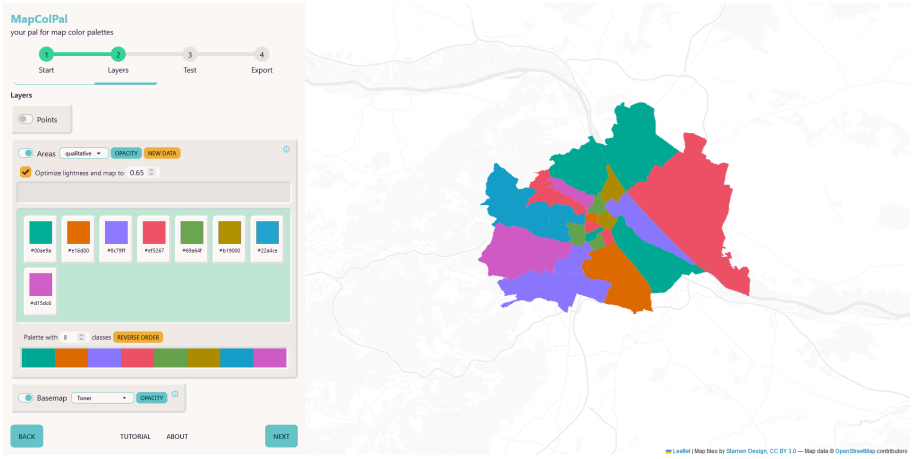


Figure D.5: Scenario 3 - MapColPal qualitative palette - Full screenshot

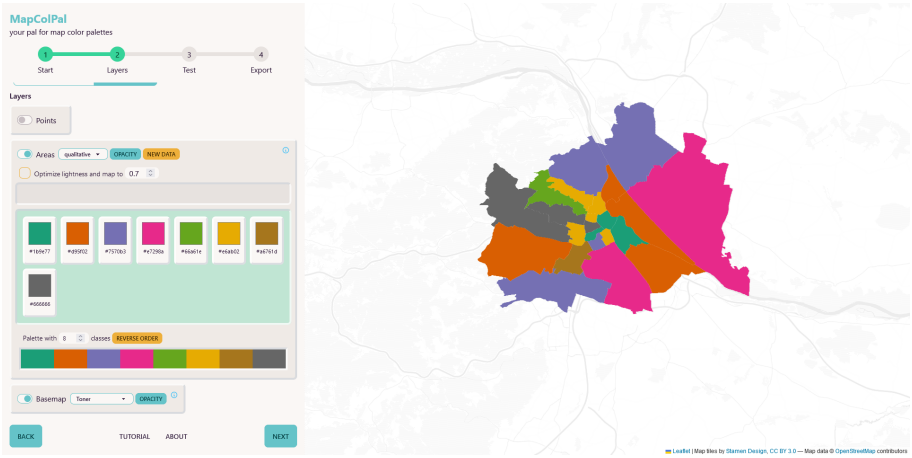


Figure D.6: Scenario 3 - ColorBrewer Dark2 palette - Full screenshot