TU WIEN Informatics

# Interactive Web-Based Flood Map Visualization

BACHELORARBEIT

zur Erlangung des akademischen Grades

**Bachelor of Science**

im Rahmen des Studiums

**Medieninformatik und Visual Computing**

eingereicht von

**Christoph Michael Essler**
Matrikelnummer 01328166

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Ing. Dr.techn. Jürgen Waser

Wien, 8. Dezember 2022 [illegible]

Christoph Michael Essler Eduard Gröller

TU WIEN Informatics

# Interactive Web-Based Flood Map Visualization

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Bachelor of Science**

in

**Media Informatics and Visual Computing**

by

**Christoph Michael Essler**
Registration Number 01328166

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Ing. Dr.techn. Jürgen Waser

Vienna, 8th December, 2022 [illegible]

Christoph Michael Essler Eduard Gröller

# Erklärung zur Verfassung der Arbeit

Christoph Michael Essler

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. Dezember 2022

[illegible]

Christoph Michael Essler

# Danksagung

Ich bin höchst dankbar, für all die Unterstützung die ich in der gesamten Zeit bekommen habe, während ich diese Bachelorarbeit schrieb. Ich danke VRVis, und zwar insbesonders den Teamkolleg*innen beim Visdom Projekt, für die einzigartige Gelegenheit, in einem so nützlichen, aufregenden, und großen Projekt wie diesem arbeiten zu können, und für ihre verlässliche und hilfreiche Unterstützung sowie Aufsicht. Ich danke auch meiner Familie und meinen Freund*innen für die Unterstützung die sie alle auf ihre eigene Weise zur Verfügung gestellt haben. Ohne den Spaß und die Ratschläge, die wir in unserer langanhaltenden Gruppe von Freunden im Studium, bestehend aus Emilia Jäger, Omar Ismail, Elias Marold, Negin Sadeghi, und mir selbst, untereinander geteilt haben, wäre das Studium zweifelsohne viel schwieriger gewesen, und eine Vielzahl großartiger Erinnerungen wären nie entstanden.

Diese Bachelorarbeit wurde durch das Kompetenzzentrum VRVis ermöglicht. Die VRVis Forschungs-GmbH wird im Rahmen von COMET – Competence Centers for Excellent Technologies (854174) durch BMK, BMDW, Land Steiermark, Steirische Wirtschaftsförderung – SFG und Wirtschaftsagentur Wien – ein Fonds der Stadt Wien gefördert. Das Programm COMET wird durch die FFG abgewickelt.

# Acknowledgements

I am highly thankful for the support I had during the overall time of writing this thesis. I thank VRVis, most specifically the people associated with the Visdom project, for the unique opportunity to work in a project as useful, exciting, and big as this, and for their reliable and helpful support and supervision. I also thank my family and friends for the support they each provided in their own ways. Without the fun times and advice shared between my group of longtime studying friends, namely Emilia Jäger, Omar Ismail, Elias Marold, Negin Sadeghi, and myself, the bachelor studies would have been indubitably much more difficult, and a lot of great memories would not have been made.

This thesis was enabled by the Competence Centre VRVis. VRVis is funded by BMK, BMDW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174), which is managed by FFG.

# Kurzfassung

Überflutungen verursachen viele Todesfälle sowie finanziellen Schaden. Überflutungskarten stellen die Anfälligkeit einer Gegend für Fluten dar und können auf verschiedene Weisen dabei helfen, Schäden durch Fluten zu reduzieren - Sie sind bereits dafür in Verwendung, festzustellen, in welchen Gegenden das Bauen neuer Gebäude sicher ist. Diese Karten online verfügbar zu machen, könnte die Aufmerksamkeit der Öffentlichkeit bezüglich Überflutungen in spezifischen Gebieten erhöhen. Außerdem sind Karten durch Verfügbarkeit im Internet leicht erreichbar. Im Zuge dieser Arbeit wurde der Status Quo bezüglich verfügbarer online Flutkarten durch Recherche erörtert, und eine React-Komponente mit interaktiver online Karte für den Web Client eines Simulations- und Visualisierungssystems für Überflutungen implementiert, welche als Überflutungskarte verwendet wird. Die Karte stellt Fluten durch Polygone dar. Die Eckpunkte der Polygone werden vom Simulationssystem geliefert, und werden am Web Client über die Karte gelegt. In der Zukunft könnte die ausgearbeitete Komponente auch mit mehr Interaktivitätsmöglichkeiten ausgestattet werden, die mit dem Simulationssystem korrespondieren, und Simulationen bearbeiten könnte.

# Abstract

Floods cause many deaths as well as lots of financial damage. Flood maps encode areas susceptibility for floods and can be used in several ways that reduce loss from floods - they are already in use for determining in which areas it is safe to build new buildings. Making them available online might increase public awareness of the risks associated with floods in specific areas. Additionally, maps are easier to reach once they are available on the internet. In the course of this thesis, the current status quo concerning available online flood maps is determined by research, and a React-based web map component to visualize flood maps was developed for the web client of a flood simulation and visualization system. The map represents floods by polygons. The vertices of the polygons are provided by the simulation system, and are then rendered over the map on the web client. In the future, the developed component could include further interactivity that allows more correspondence with the simulation system, so that it could edit the simulations.

# Contents

CHAPTER 1

# Introduction

This chapter introduces the concepts this thesis is built on, explains the motivation behind the thesis and the advantages the development of an online flood map can bring, and explains flood maps and related concepts.

## 1.1 Motivation

A disaster is a large-scale catastrophe that affects a multitude of people. *Disaster management* is the effort to avoid, reduce, and recover from the negative consequences that a disaster causes. Many different projects, tools, and procedures exist when it comes to disaster management. Considering the points in time they are used in respect to the lifetime of a potential disaster, there are some differences between them. Some are useful in advance, for example for planning cities, while others may help in taking urgent actions in case there is an emergency.

*Flood management* is disaster management that focuses particularly on floods. It is clear that floods cause a lot of damage. Multiple cases of floods with damages in the millions and billions of euros are known [CEA07, p. 8], [KNK$^+$12], [Abf], [Cen20]. Other than property losses, floods can be lethal as well. Aside from the immediate and local consequences of floods, they can also have lasting and global consequences, such as disrupted supply chains [KNK$^+$12]. Despite the risks, flood-prone areas are very attractive for the forming of settlements [WRF$^+$11, p. 1872], and contain most of the populated areas [Was11, p. 1].

Climate change is expected to make heavy rain and other weather extremes more frequent [Int14, p. 552]. Insurance Europe sees the increase of extreme weather events as opposed to the relatively stable number of geophysical events such as earthquakes and volcano eruptions also as an indication that climate change will increase the frequency of extreme weather events [CEA07, p. 9].

Summarizing these points so far: Floods cause lots of damage in various ways, and it pays off to be prepared for them well in advance. However, it is also possible that flood mitigation efforts are deployed without a flood actually appearing afterwards. In this case the investment did not pay off, considering that available budgets around the world are limited. In 2003, Pearce [Pea03] showed the importance of “sustainable hazard mitigation” over “response and recovery” approaches in disaster management, and that disaster management needs to be integrated into community planning, and needs to allow public participation.

## 1.2 Flood Maps

*Flood maps* are a useful tool in various use cases. They help giving an overview of which areas are likely to be affected by flooding events and creating awareness of them among the general public. They are also of use to disaster management experts, in order to prepare flood mitigation plans and install flood prevention measures.

A web map is particularly suitable for a flood map if it is intended to be easily reachable by a large target user group (e.g., for spreading awareness of flood risk zones to non-experts). A *web map* in this case means a map application that is hosted on the internet and accessible via a browser. The advantage of web maps is their easy accessibility. Modern operating systems come with a preinstalled web browser, so visiting a website is fairly easy, and roughly the same process for different operating systems. This is an advantage of developing a web application - it is inherently cross-platform, and is, in terms of effort, significantly easier than developing a separate native client for multiple operating systems. Mobile operating systems also have web browsers. A web-based flood map visualization that is performant enough to run on a mobile device can also be taken to the site of a flood event. Furthermore, it could be connected to simulations for decision support.

## 1.3 Problem Statement

The aim of this thesis is to implement a web map component for the efficient visualization of easily accessible flood maps. The purpose for these maps is to display flood extents, in a way that is both understandable and available to the general public.

CHAPTER 2

# Related Work

This chapter summarizes some previous related work. Sections 2.1-2.5 show some previously or currently available web-based flood maps. Section 2.6 draws some conclusions from these works - where possible, these lessons are to be used for the web map developed as part of this thesis.

Nowadays, there are multiple *flood maps* that, roughly speaking, try to show how likely an area will be affected by floods. In 2009, de Moel, van Alphen, and Aerts [dvA09] have reviewed the state of European flood maps back then. They categorized the maps into several types: *Flood extent maps* show the boundaries of a specific (possibly hypothetical) flood event, such as a $HQ_{100}$, i.e. a flood that is expected roughly once every century. *Flood depth maps* are similar, but they do not only show the border, but also encode the depth of the water into the map. For *flood danger maps*, a value 'danger´ is calculated for each point on the map out of parameters, such as water depth and flow velocity. *Flood risk maps* take hazard information and combine it with information about possible consequences of floods in an area. An example would be a map displaying direct financial damage, though a map could also consider indirect financial damage, or other non-financial consequences. They also mention other types of maps, given by mapping other parameters, such as exposure, coping capacity, flow velocity, etc.

De Moel et al. [Dir07, Fed18] also refer to a EU (European Union) directive (2007/60/EC) that was adopted in late 2007. In this directive, a *flood hazard map* is defined to contain flood extent, water depth, relevant water flow, and flow velocity. These elements need to be shown for each of three scenarios: Extreme events with a low probability, “medium” events with a return period $\geq$ 100 years, and events with a high probability. Member states of the EU are obliged to create flood hazard and risk maps for the areas which the states identify as likely to have floods, and make them publicly available.

As discussed previously, flood maps are valuable tools in the fight against floods. They are useful for planning ahead of a flood event. Also, if available online, they make it easier for the public to inform itself about flood risks in their region. Hagemeier-Klose and Wagner [HW09] said in 2009, that "Furthermore, the high frequency of flood events in Europe and globally over the last years shows an increasing need to provide precise and extensive information to the general public and especially to people at risk so as to prevent future damages." The internet is a useful platform for the purpose of communicating risks to the public. It provides high platform independence and a wide reach [SMPF00] - many people are able to access a public resource on the internet. Experts in the area of flood management also gain advantages from having their tools platform independent. With relatively little effort, one can support a variety of devices and operating systems, although sometimes, adjustments must be made. For example, before an application designed for desktop computers can be properly used on a mobile device, adaptations may need to be made.

Whenever designing an application it is important to know who your target audience is. Hagemeier-Klose and Wagner [HW09] have done a formative evaluation of some web-based flood maps, and implemented their own as well. They differentiate between *laymen* and *experts*. According to them, a simple web map with real-time information and high readability seems to be enough for laymen, whereas experts will rather need a complex web GIS system. Their evaluation happens with regards to how well they communicate the information to the 'general public', which they assume to be laymen. Of the maps they reviewed, all were either too complex for the general public, or too simple for the experts. The formative evaluation consists of three parts: A creative workshop, an online survey on Bavaria's then current web-based flood map, and an analysis of multiple existing maps. In the creative workshop with 24 people of varying levels of expertise, the participants looked at 50 (real and fabricated) examples of flood hazard maps and discussed them. Participants were asked to rate them concerning 'readability', 'design and visualization', and 'content'. Among the results of this workshop were the following takeaways:

- The maps should be easily understandable and avoid technical terms.
- Legends and categories need to be "readable at first sight".
- Showing actual past flood events (additionally to the risk zones) strengthens the "local risk awareness".
- The color blue is commonly associated with water. Various shades of blue can be used to represent different zones.
- Which background is useful, depends on the content of the map - "The legally protected flood plain or the water depths within the one hundred year flood should be presented on the digital land register map, because the land owner must be able to recognise his own parcel of land. [...] When dealing with the extension of flood events with different return periods or with different hazard zones, a digital city plan or an orthophoto should be used as background because of the easier orientation possibilities for laymen." [HW09, p. 569]

- As interaction features, searching and zooming should be implemented, as well as showing background information on selecting an object.
- Gauge levels are particularly useful for communicating the intensity of a risk zone to the public, as laymen can easily make comparisons based on them.

The online survey conducted by Hagemeier-Klose and Wagner was about the web-based flood map provided by *Informationsdienst Überschwemmungsgefährdete Gebiete Bayern* (IÜG). Interesting findings include that many users that were previously affected by floods estimate their flood risk to be relatively low, and that users expected different kinds of flooding information from the service, though the extension of flood plains was the most frequently expected information.

Performance is another important aspect. If the system has to process a lot of data in each frame of a running application, it will cause *lag*, i.e., a noticable pause between frames. Roth conducted interviews with "geospatial professionals" about the user experience (UX) of "cartographic interaction" [Rot15]. In an attempt to figure out if a map application can be considered interactive, the issue of 'immediacy' is discussed. The interviews show that there is a wish for immediate response, and that more lag makes the map to be perceived as less interactive (p. 104).

Concerning the background of the map, it is important that the relevant data is visible. Thus, when a flood extent is shown, it should be clearly distinguishable from the background. In this case it would be a bad idea to use a background that shows bodies of water, especially in blue.

## 2.1 Web-based Floodplain Advisory Tool

The *"Web-based Floodplain Advisory Tool"* (WFAT) by Sugumaran et al. [SMPF00] seems to no longer be accessible at the point of writing. The article is from the year 2000, and it would not work well with modern browsers as it is a Java Applet, and Java Applets within the browser are now considered outdated and insecure. Back then, Java Applets were more common. WFAT was devised as a tool for "planners and other local decision makers in St. Charles County, Missouri" [SMPF00, p. 1262] with different layers including flood extent boundaries of $HQ_{100}$ and $HQ_{500}$.

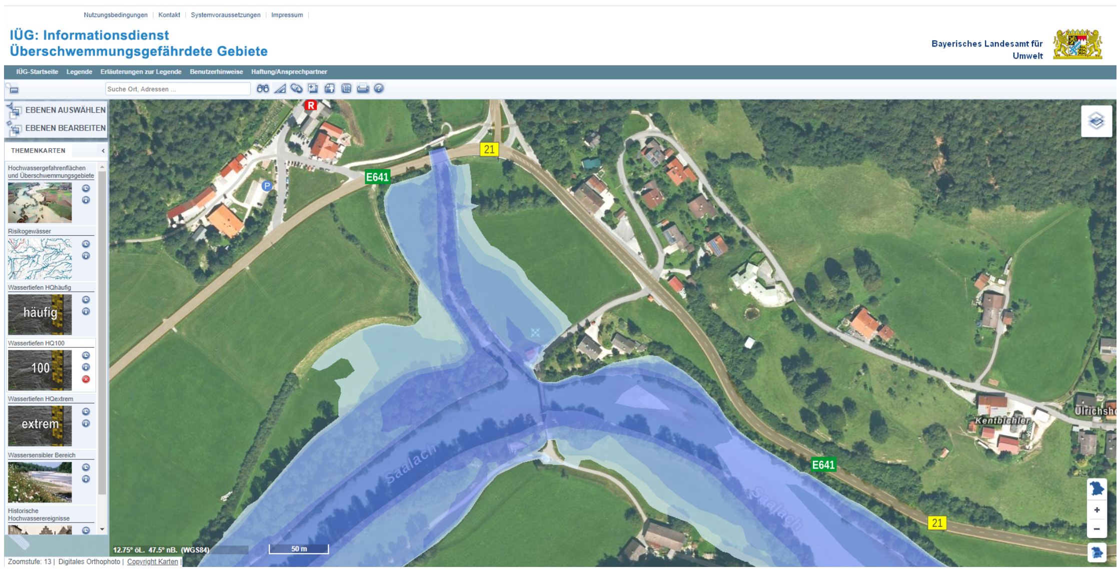

Nutzungsbedingungen | Kontakt | Systemvoraussetzungen | Impressum
IÜG: Informationsdienst Überschwemmungsgefährdete Gebiete
Bayerisches Landesamt für Umwelt
IÜG-Startseite
Legende
Erläuterungen zur Legende
Benutzerhinweise
Haftung/Ansprechpartner
Suche Ort, Adressen ...
EBENEN AUSWÄHLEN
EBENEN BEARBEITEN
THEMENKARTEN
Hochwassergefahrenflächen und Überschwemmungsgebiete
Risikogewässer
Wassertiefen HQhäufig
häufig
Wassertiefen HQ100
100
Wassertiefen HQextrem
extrem
Wassersensibler Bereich
Historische Hochwasserereignisse
E641
21
Saalach
Kentbichler
Ulrichshö
12.75° öL. 47.5° nB. (WGS84)
50 m
Zoomstufe: 13 | Digitales Orthophoto | Copyright Karten |

Figure 2.1: IÜG

## 2.2 Informationsdienst Überschwemmungsgefährdete Gebiete (IÜG)

The IÜG map [HW09] is a flood map for Bavaria. This map is no longer accessible since 30.04.2021. Visitors are encouraged to navigate to the Umwelt Atlas instead (see Section 2.3). It features several map backgrounds and data overlays. On the left side one has to pick what kind of map one wants to view, e.g. water depth of a $HQ_{100}$, or a historic floods map. Selecting the $HQ_{100}$ map defaults to a greyscale version of a common street map, which is useful since it contrasts well with the blue overlay. Alternatively one can choose other backgrounds using the control on the top right. Depicted in Figure 2.1 is an orthophoto background. Clicking on the map reveals extra information. Each map also allows different kinds of overlays.

Regarding performance, this map sometimes took time to display the overlays in the appropriate level of detail, for example when zooming into the map one could see the overlay bigger but pixelated. However while taking some time to load content, it still allowed the user to continue navigating without interrupting for the loading process. In other words, it was still possible to navigate the map about while it loads content.

The IÜG map also included search, measure, sharing, data loading, and printing features. It supported zooming by mouse wheel scrolling and panning by mouse dragging, as well as zooming via buttons. A miniature overview map of Bavaria could be toggled by the button on the bottom right.

The article by Hagemeier-Klose and Wagner [HW09] describes an earlier version of this application (see Section 2).

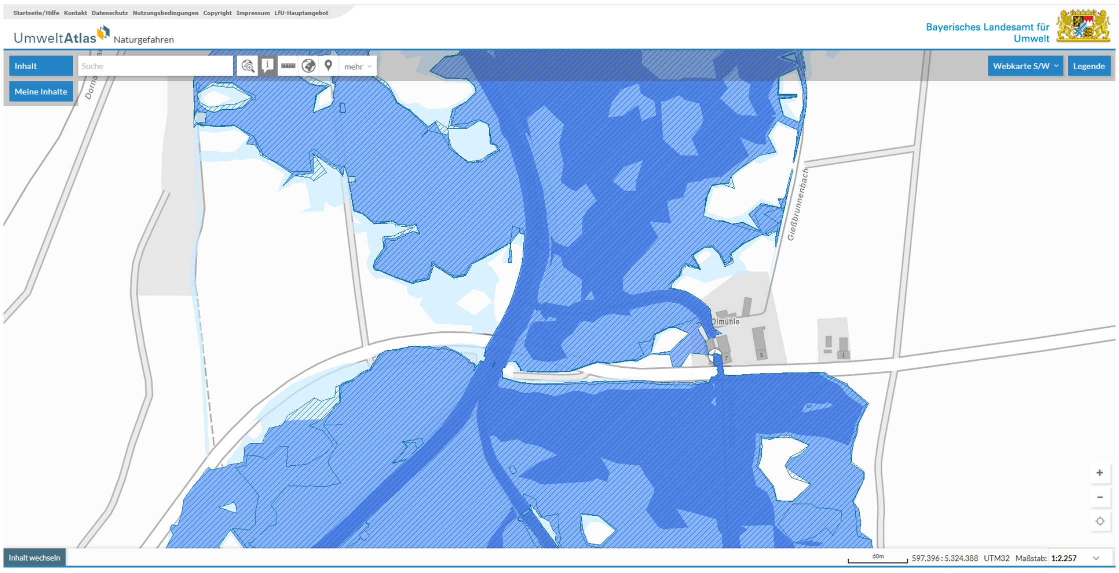

Startseite/Hilfe Kontakt Datenschutz Nutzungsbedingungen Copyright Impressum LfU-Hauptangebot
UmweltAtlas Naturgefahren
Bayerisches Landesamt für Umwelt
Inhalt
Suche
mehr
Meine Inhalte
Webkarte S/W
Legende
Gießbrunnenbach
Ölmühle
+
−
Inhalt wechseln
60m
597.396 : 5.324.388 UTM32 Maßstab: 1:2.257

Figure 2.2: Umwelt Atlas (see [Bay])

## 2.3 Umwelt Atlas Bayern

The Umwelt Atlas Bayern [Bay] provides a more current flood map for Bavaria. Navigation and look-and-feel is similar to the IÜG map. This map does not include a historic map unlike the IÜG map. Other than that it is very similar — sometimes, the overlays take time to adapt to the current zoom level, but the map remains usable during that time. It provides lots of overlay options, and a few background options. Again, clicking on a point reveals additional information about it. Navigation controls are similar to the IÜG map, a difference is that the zoom level is not directly displayed, however the scale (as ratio) is visible instead. Both maps feature a graphical scale. A miniature overview of Bavaria is available. Searching, measuring, data loading, printing, and sharing are also supported. Figure 2.2 displays risk zones for the three flood return periods that Umwelt Atlas and IÜG use ($HQ_{\text{häufig}}$, $HQ_{100}$, $HQ_{\text{extrem}}$) and some other active overlays on top of the greyscale street map.

## 2.4 Niederösterreich Atlas

The navigation of the Niederösterreich Atlas [Ene] is counterintuitive. By default, its zoom tool is selected, which means that dragging the mouse with the left mouse button causes the map to zoom to the selected rectangle, where one would usually use that interaction to pan - instead, in this mode, panning is performed by dragging with the right mouse button. If the panning tool is selected, one can use the left mouse button instead to navigate the map. It also has a tool for getting additional information, and it can also export it to various formats. It has some measuring, labelling, and printing tools. There also is a “meeting point” tool, which can be used to put a marker on the

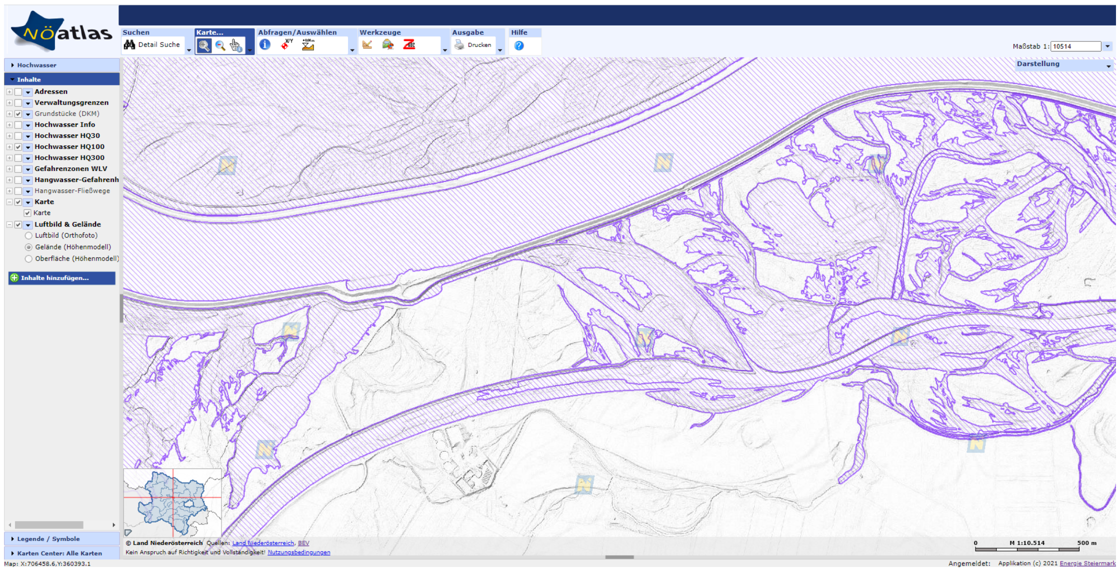


Figure 2.3: NÖ Atlas (see [Ene])

map and share it with someone. There are various overlays available, including HQ$_{30}$, HQ$_{100}$, HQ$_{300}$, and there are three background options: orthophoto, terrain and surface. It is also worth noting, that there is a loading dialog with progress bar every time the map is moved. The map can be moved while it is in the loading state. Sometimes when moving the map, some tiles of the map that were already loaded disappear during the loading process. There also is a miniature map of Lower Austria.

## 2.5 Flash Flood Risk Map for Upper Austria (FFRM)

The Flash Flood Risk Map for Upper Austria (FFRM) [Dip] was made by the Dipl.-Ing. Günter Humer GmbH. The map contains data within Upper Austria. Navigation is straightforward - dragging with the left mouse button held is panning, and mouse wheel scrolling is zooming. There also are buttons for zooming on the map. The map has been implemented with *Leaflet* [Vla] as mapping library. Users can choose between six background options, however only four of them work. The options “Greyscale” and “Streets” do not work. The remaining backgrounds are those provided by *basemap.at*. Basemap and Leaflet are discussed in more detail in Section 4.1.6, because they are used in the map implemented for this thesis. The problem with the selection of backgrounds that are available, (even with “Basemap Grau”, which one might expect to be in greyscale) is that they contain color, specifically both green and blue. There are three overlays available. The “Risk Map”, as seen in Figure 2.4, uses a scale from green over yellow to red. The “Maximum Water Depth” overlay, as seen in Figure 2.5, uses a scale from light blue over blue to purple. The “Stream Network” overlay uses blue lines. Considering the background and overlay options, there are contrast issues, e.g., the blue of the water

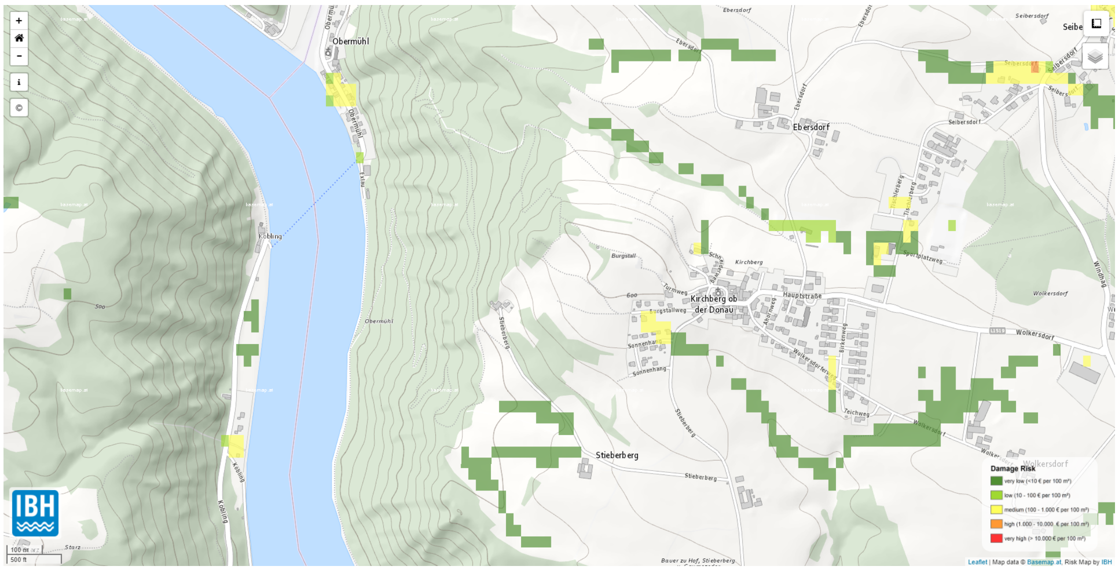
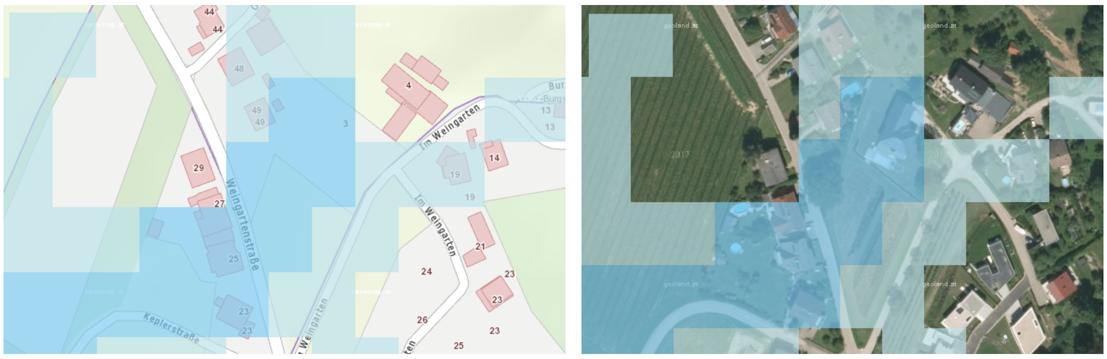

Obermühl
Eberdorf
Kirchberg ob der Donau
Stieberberg
Wolkersdorf
Seibersdorf
IBH
Damage Risk
🟩 very low (<10 € per 100 m²)
🟩 low (10 - 100 € per 100 m²)
🟨 medium (100 - 1.000 € per 100 m²)
🟧 high (1.000 - 10.000 € per 100 m²)
🟥 very high (> 10.000 € per 100 m²)
Leaflet | Map data © Basemap.at, Risk Map by IBH

Figure 2.4: FFRM Riskmap (see [Dip])



(a) Flood depth data on the street map.

(b) Flood depth data on the orthophoto map.

Figure 2.5: Grid data on the FFRM map in comparison to buildings in the background.

depth overlay on the water of the background, or the green of the risk map on terrain. There is a measuring tool and a graphical scale on this map. The zones on this map are grid based and not so much polygonal as in many other maps. The grid seems fairly coarse grained, as can be seen in Figure 2.5. The data acquirement process for the map is described in the work by [RH16].

## 2.6 Lessons Learned

For developing a simple flood extent web map, there are some valuable lessons to be learned from the web map examples in the previous subsections:

- Flood extent overlays should be in blue, as that color represents water intuitively.
- The background should not contain blue elements and bodies of water. (For distinction, overlay colors should not be similar to colors on the background)
- Web technologies that will be supported for a long time by major browsers form a more reliable base for the application. This is impossible to predict completely. The World Wide Web Consortium (W3C) defines standards for web applications [W3C], which include HTML, CSS, JavaScript Web APIs, and more. This makes HTML, CSS, and JavaScript good choices that have been in use on websites for a long time. However, sometimes newer Javascript libraries become popular and others become less maintained or even abandoned. Java Applets and web embedded Flash applications were previously popular, but are now widely considered outdated.
- People used to interactions with Google Maps might expect dragging with the left mouse button held down to pan the map, and scrolling the mouse wheel to zoom the map. For a simple map (i.e. not a complex GIS interface with multiple tools that the user can choose from), this configuration of interactions should be retained.
- A legend should be easily interpretable. A simple table showing the color in one column and a label for what the color represents is enough. The need for a legend is especially given if there are multiple zones (that should have sufficiently different colors each).

CHAPTER 3

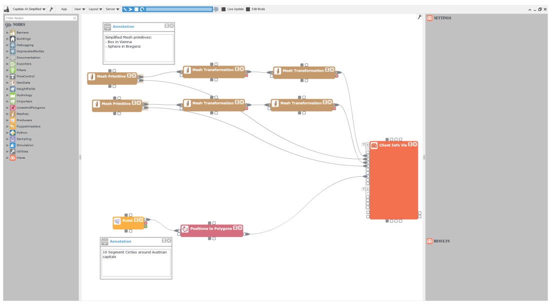
# Visdom Web Client

For the practical part of my thesis, a Map View for the web client to the Visdom system has been implemented. In Section 3.1, the features of Visdom and what the system looks like are summarized. Section 3.2 demonstrates the implemented web client features, whereas Section 4 describes implementational details and decisions, including the implementation process, tools/libraries used, problems that surfaced during development, and design decisions.

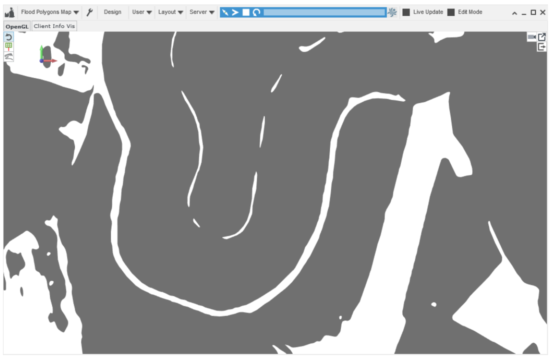
## 3.1 Framework: Visdom System

*Visdom* [Vis] 'is a powerful decision support tool for crisis management. It "combines visualization, simulation, and analysis techniques". The system consists of a server, a desktop client, and a web client that is currently in development. The client has two different modes: In the design mode (see Figure 3.1a), the project is planned using a node-link graph. The app mode depends on the project - which UI elements are shown, and what data is displayed on them depends on the nodes and their connections (see Figure 3.1b and 3.1c). This makes the application quite flexible. The project settings are synchronized to the server, which actually performs the simulations, and then the client displays results in its app view. Visdom models multiple variants of a scenario using the concept of World Lines [WFR$^+$10]. A World Line represents one of multiple related simulations. Whenever one wants to explore the simulation with a different parameter set at a specific point, they can "split" the scenario into multiple branches. This way, a tree is formed. A World Line is, in terms of graph theory, a specific version of a simulation represented in the path from the root of the tree to a single leaf.

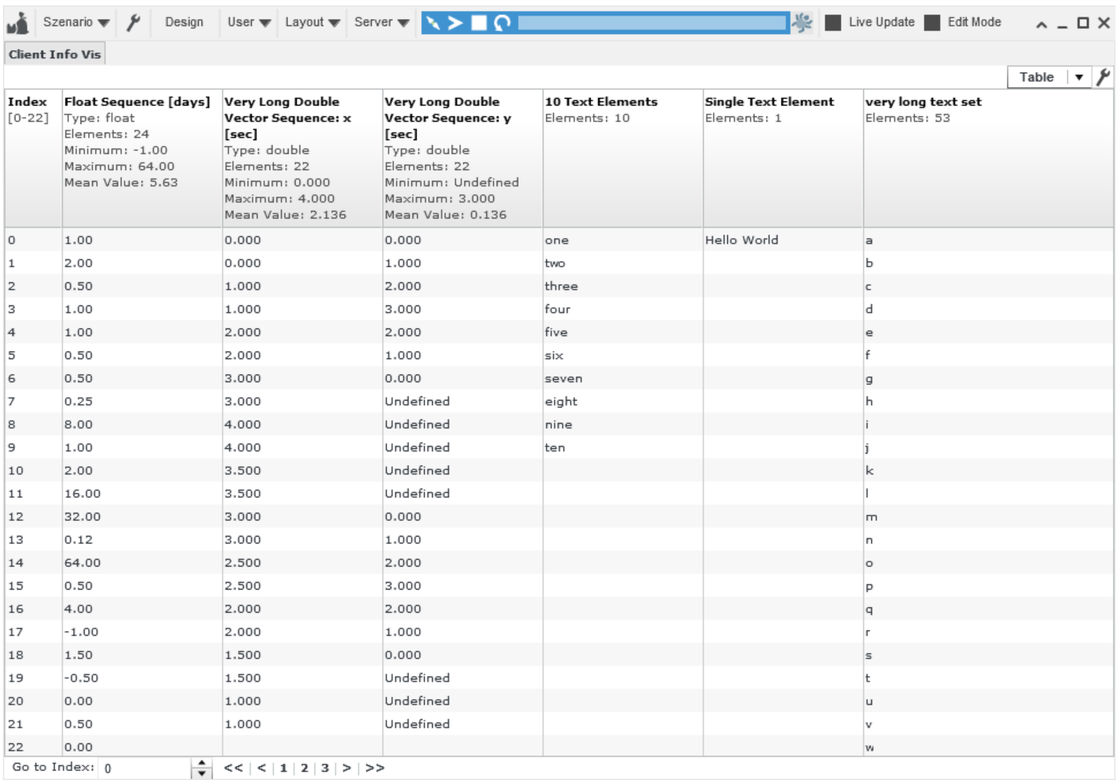
Visdom can be used for many different visualizations. Some examples of interactive 3D visualizations can be seen in Figure 3.2 and in Cornel's dissertation [Cor20]. The environment can be intuitively interacted with, e.g. sandbag barriers can be sketched into the environment [WKC18].



(a) Design view, showing the `capitalsAt simplified.visdom` project.



(b) OpenGL window, showing a part of the `HQ100` dataset.

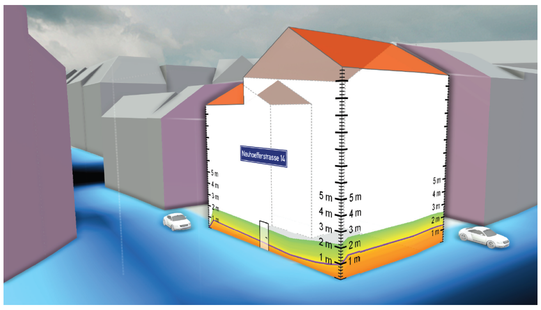


| Index [0-22] | Float Sequence [days] Type: float Elements: 24 Minimum: -1.00 Maximum: 64.00 Mean Value: 5.63 | Very Long Double Vector Sequence: x [sec] Type: double Elements: 22 Minimum: 0.000 Maximum: 4.000 Mean Value: 2.136 | Very Long Double Vector Sequence: y [sec] Type: double Elements: 22 Minimum: Undefined Maximum: 3.000 Mean Value: 0.136 | 10 Text Elements Elements: 10 | Single Text Element Elements: 1 | very long text set Elements: 53 |
|---|---|---|---|---|---|---|
| 0 | 1.00 | 0.000 | 0.000 | one | Hello World | a |
| 1 | 2.00 | 0.000 | 1.000 | two | | b |
| 2 | 0.50 | 1.000 | 2.000 | three | | c |
| 3 | 1.00 | 1.000 | 3.000 | four | | d |
| 4 | 1.00 | 2.000 | 2.000 | five | | e |
| 5 | 0.50 | 2.000 | 1.000 | six | | f |
| 6 | 0.50 | 3.000 | 0.000 | seven | | g |
| 7 | 0.25 | 3.000 | Undefined | eight | | h |
| 8 | 8.00 | 4.000 | Undefined | nine | | i |
| 9 | 1.00 | 4.000 | Undefined | ten | | j |
| 10 | 2.00 | 3.500 | Undefined | | | k |
| 11 | 16.00 | 3.500 | Undefined | | | l |
| 12 | 32.00 | 3.000 | 0.000 | | | m |
| 13 | 0.12 | 3.000 | 1.000 | | | n |
| 14 | 64.00 | 2.500 | 2.000 | | | o |
| 15 | 0.50 | 2.500 | 3.000 | | | p |
| 16 | 4.00 | 2.000 | 2.000 | | | q |
| 17 | -1.00 | 2.000 | 1.000 | | | r |
| 18 | 1.50 | 1.500 | 0.000 | | | s |
| 19 | -0.50 | 1.500 | Undefined | | | t |
| 20 | 0.00 | 1.000 | Undefined | | | u |
| 21 | 0.50 | 1.000 | Undefined | | | v |
| 22 | 0.00 | | | | | w |

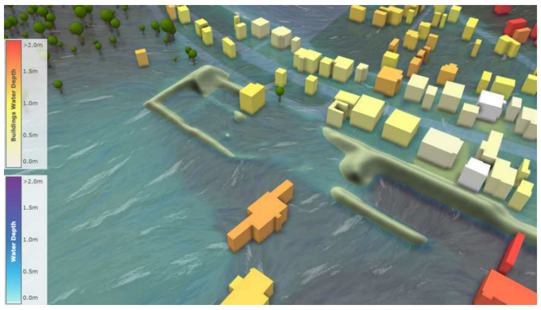
Go to Index: 0 << < 1 2 3 > >>

(c) "Client Info Vis" window, showing the table view with some sample data.

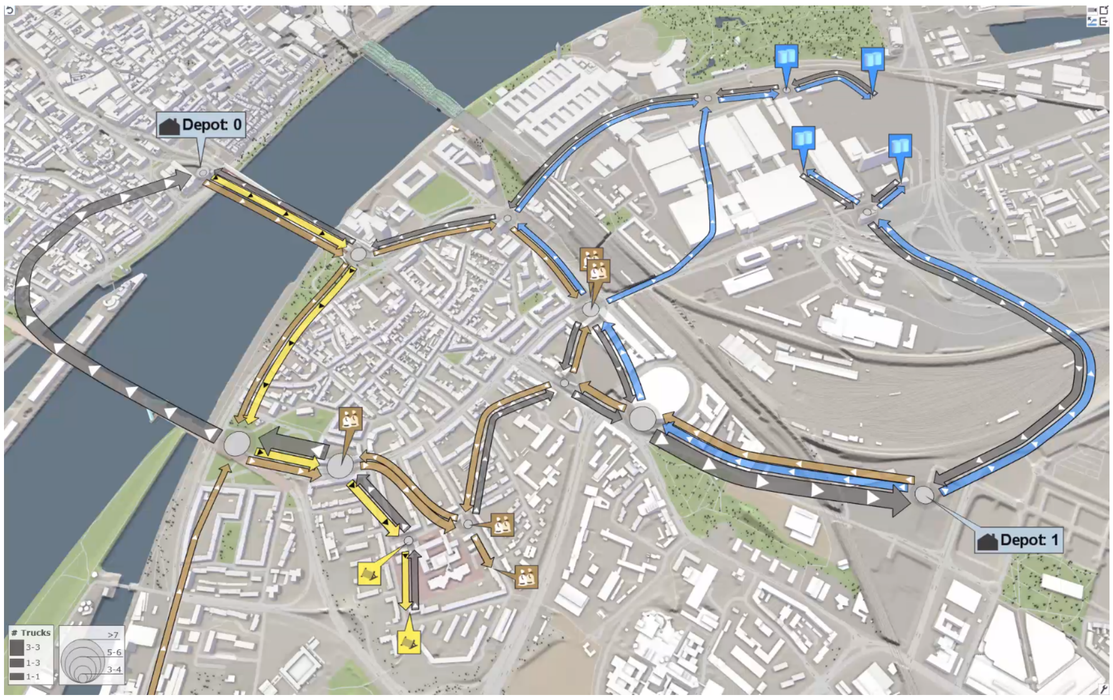
Figure 3.1: Screenshots of the Visdom desktop client.



(a) Colors on the facade of the house represent how likely it is to be exposed to water. (See Figure 2 in [CKS+15])



(b) Colors of the water and buildings represent how deep the water is at that point. (See Figure 4.18 in [Cor20])



(c) Composite flow map. (See [CKS+16][Com])

Figure 3.2: Some of the different visualizations that are possible in Visdom.

Visdom allows collaboration sessions on simulations between experts and decision-makers, in which questions may come up, for which simulation parameters need to be tweaked [WKC18]. The results therefore have to be there quickly. Visdom achieves this by using parallel GPU processing for its simulations. Since Visdom combines simulation, analysis, and visualization capabilities, it also saves time from running three separate tools and working with different tools. This workflow also supports designing sewer networks.

Visdom can be used for different simulations. A major point is surface water simulation, e.g., heavy rainfall or overflowing water bodies that cause floods. Sewer system simulations can be coupled with the surface water simulations [WKC18]. Crowd simulations, e.g., for evacuation scenarios are also supported.

Specific projects using Visdom include:

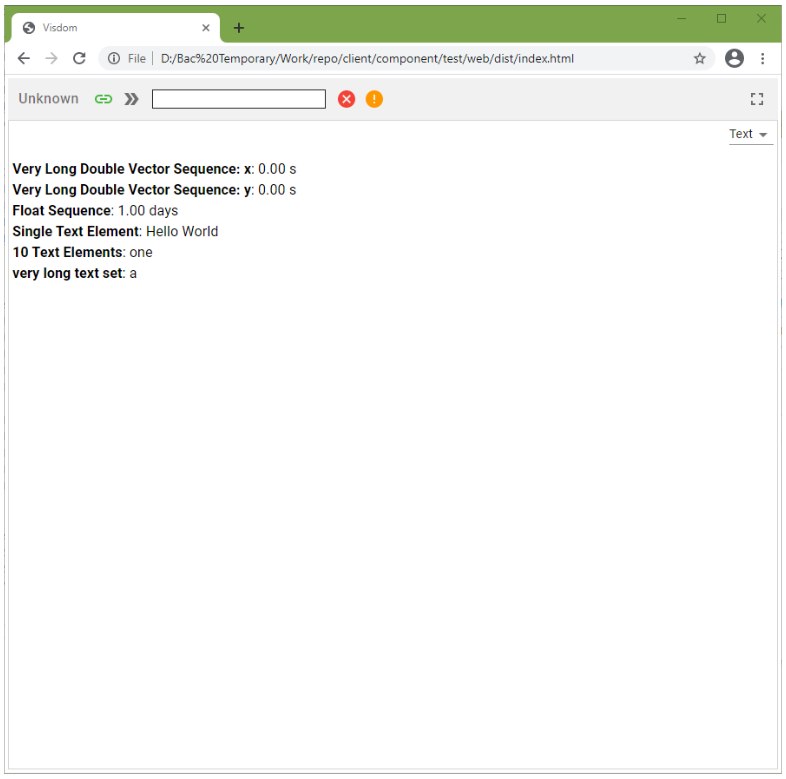
- Krösl‘s *Master of Disaster* [KSD+19] is a project that builds on Visdom. It is a VR software that connects to Visdom. It can be used for flood response training. It is cheaper and safer to train personell in a virtual environment rather than in a real one. It is also a strong way to convey the work of a flood manager and the personal risks of people in a flood to non experts.
- Decision support tool *FLOODVISOR* (see [VRV, Vis]) for the City of Cologne.
- Simulation based risk zone map *HORA 3* for the HORA (Natural Hazard Overview Risk Assessment Austria) website [BL, Vis].
- Dyke breach simulation for Marchfeld, Austria [Vis].
- Surface runoff water simulation for Amstetten, Austria [Vis].

## 3.2 State of the Web Client

This section demonstrates the current state of the web client, and describes the features that were added as part of this thesis.

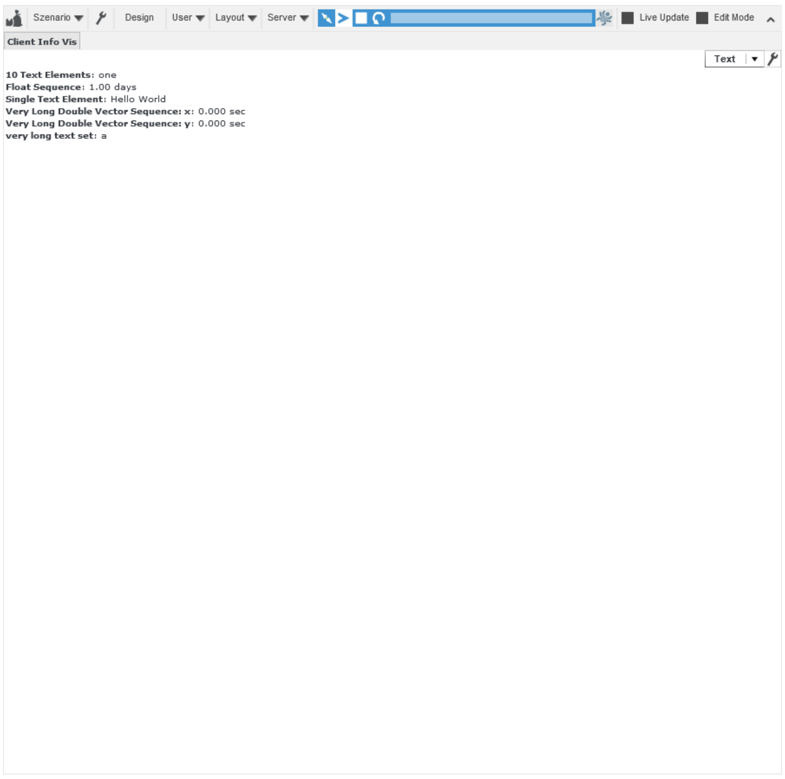
### 3.2.1 Text View

The text view displays the incoming numeric and text data as simple lines of text. Incoming data can each be a single value or a sequence of values. Each text line contains a label and the first value of the corresponding data. If the data is a vector, it is split into one text line per component. For example in Figure 3.3a, ‘Very Long Double Vector Sequence’ is a vector sequence with 22 elements and two components (x and y). Each label for vector data consists of the name associated with the data and the component. The text view on the web client has been ported from its desktop client equivalent (see Figure 3.3b).


Visdom
File | D:/Bac%20Temporary/Work/repo/client/component/test/web/dist/index.html
Unknown
Text

**Very Long Double Vector Sequence: x**: 0.00 s
**Very Long Double Vector Sequence: y**: 0.00 s
**Float Sequence**: 1.00 days
**Single Text Element**: Hello World
**10 Text Elements**: one
**very long text set**: a

(a) Text view on the web client



**10 Text Elements**: one
**Float Sequence**: 1.00 days
**Single Text Element**: Hello World
**Very Long Double Vector Sequence: x**: 0.000 sec
**Very Long Double Vector Sequence: y**: 0.000 sec
**very long text set**: a

(b) Text view on the desktop client

Figure 3.3: The text view on the Visdom web client in comparison to the one on the desktop client.

### 3.2.2 Table View

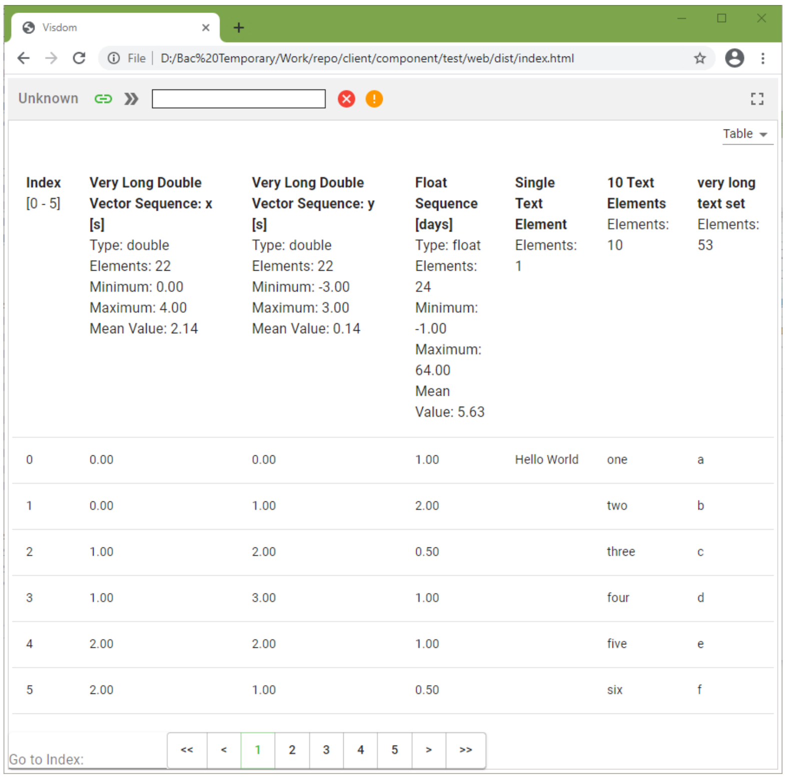
The table view (see Figure 3.4a) shows the same incoming data, but not only the first value like the text view does. Rather than that, the table view contains a table with a row for each element in the sequence. Vectors are again split into a column for each of their components. The table is paginated - it uses a custom paginator control, and requests from the server only the rows that fit on the page. The table view is ported from the desktop clients table view (see Figure 3.4b). The number of rows that fit onto a page are a very different between the two clients. This is not surprising, as the web clients table view uses Material-UI components, which use a larger text size and and big margins. If the browser window is not wide enough, there will be additional line breaks in the column headers, which causes the headers to take quite some space. Figure 3.5 shows the same table as before but on a significantly bigger window. If the table is too wide for the viewport, the table can be scrolled horizontally.

The paginator has up to five page number buttons, and buttons for navigating to the first, previous, next, and last page. Also there is a text box for entering a row index manually. These controls are all present on the desktop client table view as well, from which this table view has been ported. The paginator will not always have five page number buttons, as the table will not always have more than or equal to five pages. While on one of the first three pages, the paginator will show page number buttons from one to five (or less if less pages exist). Analogously, while on the last three pages ($n-2$ to $n$ where $n$ is the number of pages), the paginator will show buttons for pages $n-4$ to $n$ (if the pages exist). On other pages, the current page number $c$ will be centered in the page buttons, and there will be page buttons from $c-2$ to $c+2$ (again only if the pages exist). By following these rules, the paginator respects the minimum and maximum page number of the table, and has five page number buttons available whenever possible. An example of this behavior can be seen in Figure 3.6. Figure 3.6a shows the paginator on page 4, where the current page number is centered between the other page number buttons, two on each side. Figure 3.6b shows the paginator on the last page, in which case the current page number is to the right of four page number buttons. When the browser window is resized, the pagination is redone, and navigates to the fitting page.

The number box (see Figure 3.7) can be used for entering a specific row index. The client navigates to the page containing that row index. The number can be changed by typing directly into the box, or by using the small arrow buttons that appear when the box is focussed.

### 3.2.3 Map View

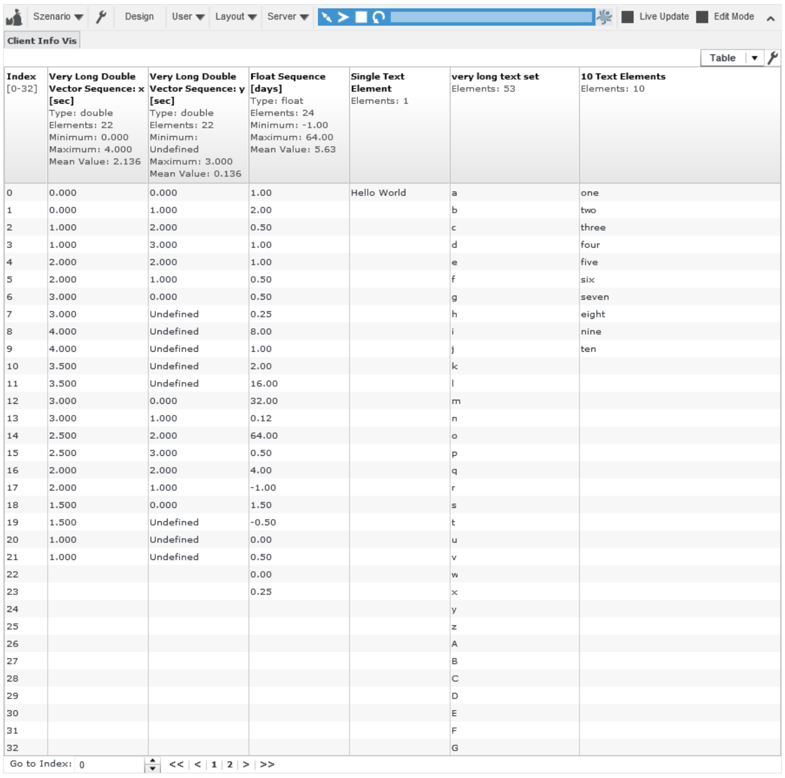
The map view (see Figure 3.8) is the main contribution of this thesis. While the other views were just ported from the desktop client to the web client, this view was not. Other than the table and text view, the map view does not take plain numbers, vectors, texts or sequences of those. It takes polygon data, and plots the polygons on a web map.


Visdom
D:/Bac%20Temporary/Work/repo/client/component/test/web/dist/index.html
Unknown
Table

| Index<br>[0 - 5] | **Very Long Double Vector Sequence: x [s]**<br>Type: double<br>Elements: 22<br>Minimum: 0.00<br>Maximum: 4.00<br>Mean Value: 2.14 | **Very Long Double Vector Sequence: y [s]**<br>Type: double<br>Elements: 22<br>Minimum: -3.00<br>Maximum: 3.00<br>Mean Value: 0.14 | **Float Sequence [days]**<br>Type: float<br>Elements: 24<br>Minimum: -1.00<br>Maximum: 64.00<br>Mean Value: 5.63 | **Single Text Element**<br>Elements: 1 | **10 Text Elements**<br>Elements: 10 | **very long text set**<br>Elements: 53 |
|---|---|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 1.00 | Hello World | one | a |
| 1 | 0.00 | 1.00 | 2.00 | | two | b |
| 2 | 1.00 | 2.00 | 0.50 | | three | c |
| 3 | 1.00 | 3.00 | 1.00 | | four | d |
| 4 | 2.00 | 2.00 | 1.00 | | five | e |
| 5 | 2.00 | 1.00 | 0.50 | | six | f |

Go to Index: << < 1 2 3 4 5 > >>

(a) Table view on the web client

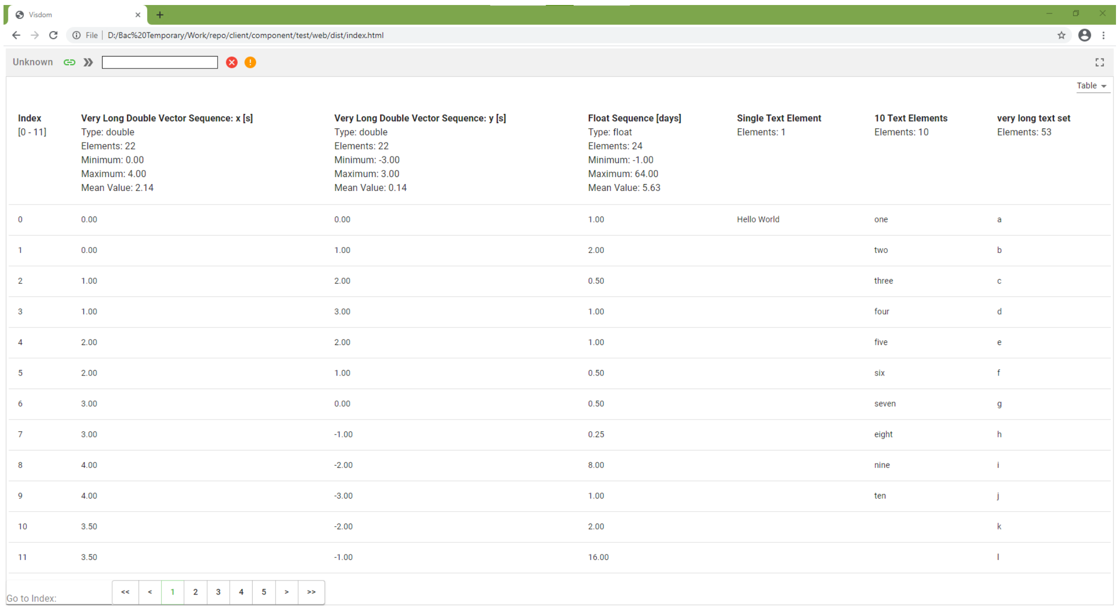


| Index<br>[0-32] | **Very Long Double Vector Sequence: x [sec]**<br>Type: double<br>Elements: 22<br>Minimum: 0.000<br>Maximum: 4.000<br>Mean Value: 2.136 | **Very Long Double Vector Sequence: y [sec]**<br>Type: double<br>Elements: 22<br>Minimum: Undefined<br>Maximum: 3.000<br>Mean Value: 0.136 | **Float Sequence [days]**<br>Type: float<br>Elements: 24<br>Minimum: -1.00<br>Maximum: 64.00<br>Mean Value: 5.63 | **Single Text Element**<br>Elements: 1 | **very long text set**<br>Elements: 53 | **10 Text Elements**<br>Elements: 10 |
|---|---|---|---|---|---|---|
| 0 | 0.000 | 0.000 | 1.00 | Hello World | a | one |
| 1 | 0.000 | 1.000 | 2.00 | | b | two |
| 2 | 1.000 | 2.000 | 0.50 | | c | three |
| 3 | 1.000 | 3.000 | 1.00 | | d | four |
| 4 | 2.000 | 2.000 | 1.00 | | e | five |
| 5 | 2.000 | 1.000 | 0.50 | | f | six |
| 6 | 3.000 | 0.000 | 0.50 | | g | seven |
| 7 | 3.000 | Undefined | 0.25 | | h | eight |
| 8 | 4.000 | Undefined | 8.00 | | i | nine |
| 9 | 4.000 | Undefined | 1.00 | | j | ten |
| 10 | 3.500 | Undefined | 2.00 | | k | |
| 11 | 3.500 | Undefined | 16.00 | | l | |
| 12 | 3.000 | 0.000 | 32.00 | | m | |
| 13 | 3.000 | 1.000 | 0.12 | | n | |
| 14 | 2.500 | 2.000 | 64.00 | | o | |
| 15 | 2.500 | 3.000 | 0.50 | | p | |
| 16 | 2.000 | 2.000 | 4.00 | | q | |
| 17 | 2.000 | 1.000 | -1.00 | | r | |
| 18 | 1.500 | 0.000 | 1.50 | | s | |
| 19 | 1.500 | Undefined | -0.50 | | t | |
| 20 | 1.000 | Undefined | 0.00 | | u | |
| 21 | 1.000 | Undefined | 0.50 | | v | |
| 22 | | | 0.00 | | w | |
| 23 | | | 0.25 | | x | |
| 24 | | | | | y | |
| 25 | | | | | z | |
| 26 | | | | | A | |
| 27 | | | | | B | |
| 28 | | | | | C | |
| 29 | | | | | D | |
| 30 | | | | | E | |
| 31 | | | | | F | |
| 32 | | | | | G | |

Go to Index: 0 << < 1 2 > >>

(b) Table view on the desktop client

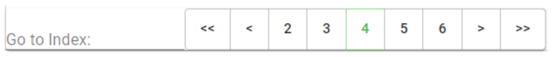
Figure 3.4: The table view on the Visdom web client in comparison to the one on the desktop client.



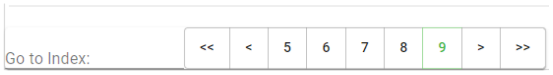
Visdom
File | D:/Bac%20Temporary/Work/repo/client/component/test/web/dist/index.html
Unknown
Table
Go to Index:
<< < 1 2 3 4 5 > >>

| Index [0 - 11] | Very Long Double Vector Sequence: x [s] Type: double Elements: 22 Minimum: 0.00 Maximum: 4.00 Mean Value: 2.14 | Very Long Double Vector Sequence: y [s] Type: double Elements: 22 Minimum: -3.00 Maximum: 3.00 Mean Value: 0.14 | Float Sequence [days] Type: float Elements: 24 Minimum: -1.00 Maximum: 64.00 Mean Value: 5.63 | Single Text Element Elements: 1 | 10 Text Elements Elements: 10 | very long text set Elements: 53 |
|---|---|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 1.00 | Hello World | one | a |
| 1 | 0.00 | 1.00 | 2.00 | | two | b |
| 2 | 1.00 | 2.00 | 0.50 | | three | c |
| 3 | 1.00 | 3.00 | 1.00 | | four | d |
| 4 | 2.00 | 2.00 | 1.00 | | five | e |
| 5 | 2.00 | 1.00 | 0.50 | | six | f |
| 6 | 3.00 | 0.00 | 0.50 | | seven | g |
| 7 | 3.00 | -1.00 | 0.25 | | eight | h |
| 8 | 4.00 | -2.00 | 8.00 | | nine | i |
| 9 | 4.00 | -3.00 | 1.00 | | ten | j |
| 10 | 3.50 | -2.00 | 2.00 | | | k |
| 11 | 3.50 | -1.00 | 16.00 | | | l |

Figure 3.5: Bigger web client window with the table view.



(a) Paginator on page 4

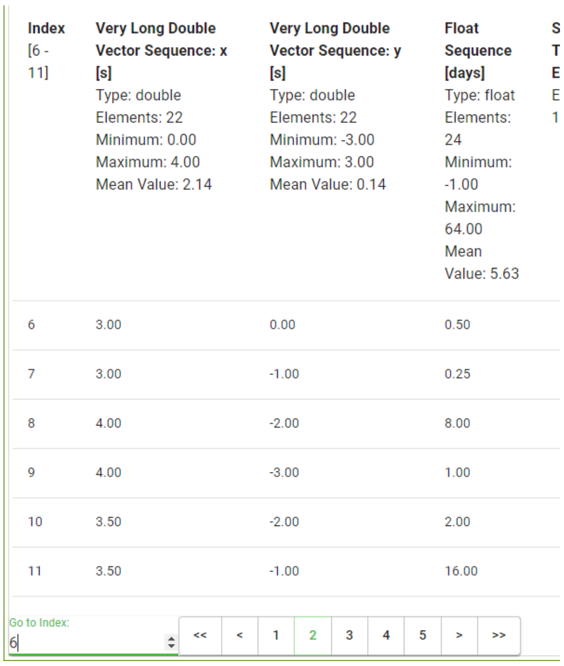


(b) Paginator on the last page

Figure 3.6: Paginator of the web clients table view.

The map supports two different map engines (each with different background tiles). Users select the map type from the dropdown box on the top right of the map view. The two supported engines are Leaflet [Vla] and Google Maps [Gooa]. Leaflet is the default setting. The Leaflet map is configured to use tiles from basemap.at [Bas] ("Gelände" and "Overlay" variants). This combination provides basic street map information, while not showing bodies of water. The Google Maps map has been assigned a custom style so that it does not show bodies of water either. Other than that, it is the default Google Maps map style. Water on map backgrounds was avoided, so that it does not visually interfere with the polygon overlays that the Visdom web client adds on top of the map background, in accordance with the guidelines laid out in Section 2.6. The Google Map also allows users to switch to a satellite image background, in which water is still visible. Hill shading can also be activated on the Google Maps street map. Figure 3.9 shows the Google Maps map with a polygonal overlay, and with hill shading activated. The polygons that are rendered can be concave and have holes, and are rendered with a transparent blue fill on both engines. To enable hill shading, the user must hover over the "Map" button in the toggle button group in the top left - upon hovering, a checkbox labelled "Terrain" is revealed, which then has to be checked.

The navigation features of the map view depend on the navigation capabilities of the

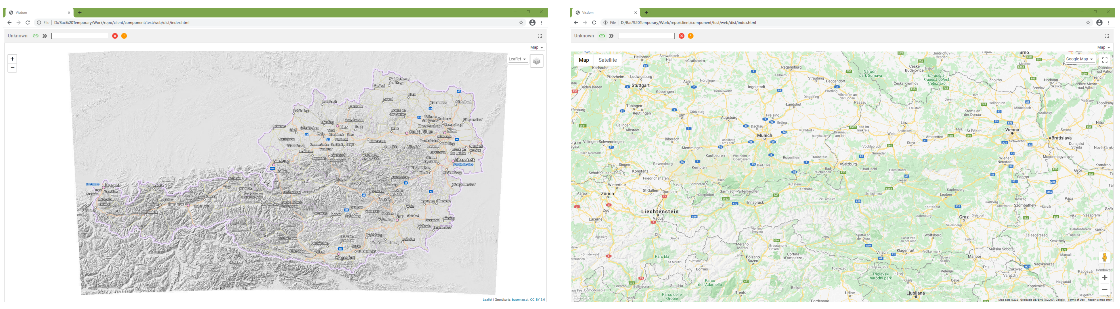


| Index [6 - 11] | Very Long Double Vector Sequence: x [s] Type: double Elements: 22 Minimum: 0.00 Maximum: 4.00 Mean Value: 2.14 | Very Long Double Vector Sequence: y [s] Type: double Elements: 22 Minimum: -3.00 Maximum: 3.00 Mean Value: 0.14 | Float Sequence [days] Type: float Elements: 24 Minimum: -1.00 Maximum: 64.00 Mean Value: 5.63 | S T E E 1 |
|---|---|---|---|---|
| 6 | 3.00 | 0.00 | 0.50 | |
| 7 | 3.00 | -1.00 | 0.25 | |
| 8 | 4.00 | -2.00 | 8.00 | |
| 9 | 4.00 | -3.00 | 1.00 | |
| 10 | 3.50 | -2.00 | 2.00 | |
| 11 | 3.50 | -1.00 | 16.00 | |

Go to Index:
6

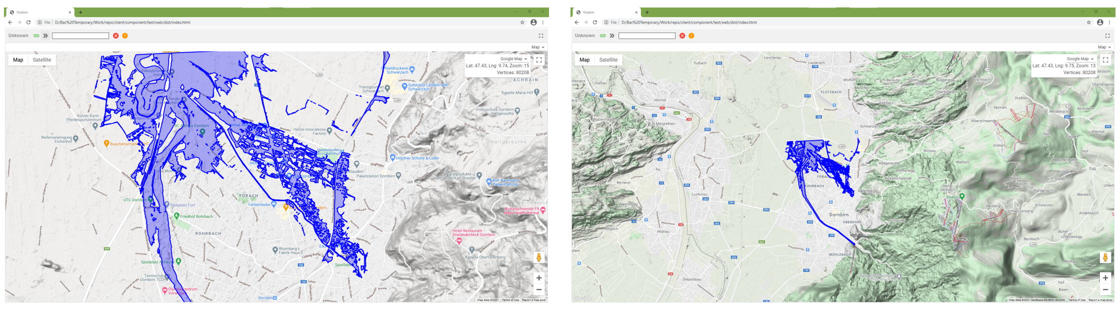
<< < 1 2 3 4 5 > >>

Figure 3.7: Focussed number box “Go To Index” in the paginator.



(a) Leaflet map (b) Google map

Figure 3.8: Austria on both map engines running in the map view of the web client.



(a) at zoom level 15 (b) at zoom level 13

Figure 3.9: Google Map mode of the map view with hill shading enabled. The polygons rendered on top of the map are loaded from the shapefile `vorarlbergPart.shp`. The map view can render concave polygons with holes.

map engines, which both support dragging with the left mouse button for panning and mouse wheel scrolling for zooming. Both also support double clicking for zooming in, and also have ‘+’ and ‘-’ buttons for zooming in and out respectively. The Google Maps map also has the “Street View” feature enabled. Both, Street View and satellite images, are not intentional features, but rather just carry over from using Google Maps as a library. Though no effort has yet been put into disabling these features for the Visdom web client, this might be the case in the future. Position and zoom level synchronize between Leaflet and Google Maps. This also makes it easier to compare the two maps for a specific location.

### 3.2.4 Integration with the System

For the web client to properly run, an instance of the Visdom server must be running and reachable by the web client. Also, if the client is to actually receive data from the server, the server must first be configured to run a Visdom project. To do that, an instance of the desktop client is required for loading a .visdom project onto the server. Once that is done, the desktop client is no longer required.

CHAPTER 4

# Implementation

## 4.1 Tools

The web client uses the following technologies, libraries, and tools:

### 4.1.1 TypeScript & React

Facebook‘s **React** [Reaa] framework is used for writing custom user interface (UI) elements that can be used and written with JSX, which provides an HTML-like structure. React can be used in combination with Javascript or TypeScript - for the Visdom web client, Microsofts **TypeScript** [Typ] is used as the programming language. It is an extension of the well known JavaScript language. TypeScript programs need to be compiled to JavaScript programs first before being usable by browsers. TypeScript adds static typing to common JavaScript, which is dynamically typed. React is used with Hooks, and additionally **Redux** [Red].

### 4.1.2 Leaflet & Google Maps

As mentioned in Section 3.2.3, **Leaflet** and **Google Maps** are used as map libraries for the map view.

### 4.1.3 Material-UI

The web client uses the **Material-UI** [Mata] React library for creating UIs that conform to Googles *Material Design* design language. Material-UI provides several controls such as buttons or tables that conform to Material Design out of the box.

### 4.1.4 Google-Maps-React & React Leaflet

The two libraries Google-Maps-React and React Leaflet are used for integrating Google Maps and Leaflet with the React framework. Other libraries for these purposes have been considered, and these were chosen with regards to their code maintenance statistics on the npm webpage [Npm] and whether they provide the required features. Recent and frequent updates indicate that a library is well maintained, which is important (see Section 2.6).

The **Google-Maps-React** library was made by the Fullstack React team. It exists on GitHub as companion code to their tutorial "How to Write a Google Maps React Component" [Ari]. The library is however freely usable and available as npm package [Goob]. The **React Leaflet** library [Reab] is available as a npm package [Reac].

### 4.1.5 Other Used Tools

Npm is used for installing libraries as npm packages and for running npm scripts, e.g. for building the application using Gulp. Microsofts Visual Studio Code is used for development - it allows debugging the code and integrates the aforementioned npm features. React Measure [Read] is used for detecting and reacting to resizing of the browser window or measuring the dimensions of components in general.

### 4.1.6 Issues With Specific Tools

The tools that are used have some issues and other characteristics that have to be considered.

#### Licensing & Pricing

Multiple products (map engines and tiles) were considered as underlying maps for the map view before finally settling on Leaflet and Google Maps. Licenses and pricing of the products make some less easily accessible than others. The considered map engines were OpenLayers [Ope], Leaflet [Vla], Mapbox GL JS [Map], and Google Maps [Gooa]. OpenLayers and Leaflet are engines that are free to use and have no particular default map tiles, but can use several kinds of map tiles which have their own pricing and licenses offered by different hosts. The basemap.at tiles are free to use under the "Open Government Data Österreich Lizenz CC-BY 4.0" license. The pricing and licensing issues were compared in a spreadsheet (last modified on 30th of April 2020 - more recent changes in those products are thus not represented). A list version of the spreadsheet can be found in appendix A.

#### Bugs & Limitations

The Google Maps map sometimes gets "stuck" in panning, in the sense that while panning, it might happen that one can no longer pan until the zoom level is changed. The Leaflet map is limited to Austria. Some of the available map tiles (i.e. rectangular parts of the map) surround Austria, but the quality of the tiles decreases outside the country borders. Further outside, there are no tiles that can load at all, which causes a 404 http message to appear in the browser console. The same goes for the zoom level with specifically 'Gelände' tiles that are used. To avoid unnecessary 404 messages and the background disappearing, the Leaflet map is limited to a maximum zoom level of 17. Zooming far into the outside of Austria near the border still results in these messages, e.g.:

```
    GET https://maps4.wien.gv.at/basemap/bmapoverlay/normal/
google3857/17/45103/70548.png 404 (Not Found)
```

### Problems With Comparing Two Different Map Engines

Comparing the two map types is very tricky. First of all, the map engines are not directly used per se - they are handled by the React bindings. Just by comparing the performance of the two map types, one cannot draw conclusions on the map engines, because the React bindings may negatively affect the performance, and they might affect the performance differently. Similarly, it has to be considered, that Leaflet and Google Maps are configured differently in terms of background tiles. The Leaflet map is set up with two tile sets that are superimposed, meaning there are two png images loaded per tile unit. Google Maps is set up with a custom style that removes water. This needs to be remembered when drawing any conclusions on the performance of the maps, especially with regards to tile loading.

If the two map types had events with the same triggers, it would also be easier to compare. For example, if both maps had events that trigger when the map and any overlays have loaded, one could compare the loading times of the maps. Google-maps-react has an "onReady" event: "When the <Map /> instance has been loaded and is ready on the page, it will call the onReady prop, if given. The onReady prop is useful for fetching places or using the autocomplete API for places." [Goob, Section "onReady"] This description does not say anything about whether that includes having all tiles loaded, or whether the overlay polygons have already been drawn. It also has an "onTilesLoaded" event which fires after it had to load tiles and is done loading them. Again this does not include any information about whether the polygons were drawn. React Leaflet on the other hand, provides an "onadd" event on the polygon component, that seems to fire after the polygons were drawn (although the documentation does not specify that), but in this case, the background tiles do not have to be loaded.

An easy way of comparing performance would usually be measuring how long a task takes to perform with the different scenarios, in this case map types. But map navigation tasks are highly depending on the user. Therefore it was decided to use FPS as a measure instead. This is still not a perfect solution, since users directly control the amount of time spent in any given part of the map. Since the different parts of the map are not equally easy to perform render, they thereby influence the distribution of high and low FPS values in a recording.

## 4.2 Implementation Steps

### 4.2.1 Implementing Data & Parsing

The first task at hand was, to parse messages sent from the Visdom server into objects that the web client can use. This is the task of the `DataParser`. The Visdom desktop client contains a DataParser as well, so this was mostly a task of porting parts of it to TypeScript. The web clients DataParser can parse any of the following data types:

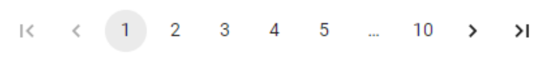
- Text — Strings and sequences of strings
- Vectors — Scalars or vectors of numbers (e.g., `float`, `double`, `int`, `uint`) and sequences of those
- Polygons — Sets of polygons: If polygons with holes are transmitted to the server, this is done by sending one polygon set with all the exterior polygons, each having their own id, and one set with all the interior polygons (i.e. the holes), each of whose id must match the exterior polygon that it is a hole in.
- Mesh — Triangle mesh: The web client only makes use of the vertex positions and the triangles (i.e., which vertices are connected into a triangle) - however, other mesh data such as texture coordinates are also parsed. These data were already included in the given information, so it was easy to parse them as well, and make them available for future versions.

### 4.2.2 Implementing Text & Table View

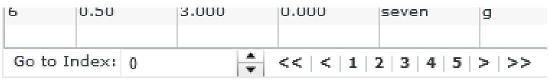
Porting the text view was a fairly easy task. There are only a few text elements to place so nothing sophisticated has happened here. The desktop version served as a template. Therefore the labels are bold and the values are not. If inbound data does not actually contain a value, the message "No Values" in italic is displayed where the value would be.

The table view is more complex than the text view. Because the table can include many entries, paging is required. This was a difficult task on two ends. On the UI side, the web client needs to figure out how much space is available and adapt the number of rows that are shown on a single page, as well as provide a paginator control for navigating between the pages. Apart from UI issues, the paging needed to be properly hooked up so that it actually gets the correct rows from the server.

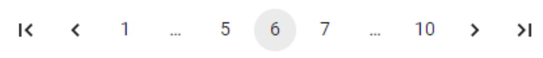
Material-UI contains a paginator control with some flexibility. By default, 'first' and 'last' arrow buttons are not shown, however they can be enabled. The first and last page number buttons always remain at the same location, no matter which page is selected, so this is similar to 'first' and 'last' arrow buttons. Figure 4.1 shows a variant of the Material-UI Pagination. To get even closer, one can increase the number of pages shown to the left and right of the current page. However, the Material-UI documentation mentions that for tables, the TablePagination control should be used instead of the Pagination



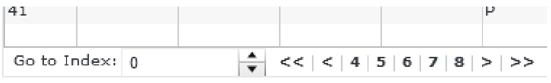
(a) One variant of the Pagination Material-UI control.



(b) Paginator of the desktop client.



(c) Material-UI variant on page 6.



(d) Desktop client on page 6.

Figure 4.1: The Material-UI Pagination control [Matb] can be configured to look more like the desktop clients paginator, by adding first and last page arrow buttons.

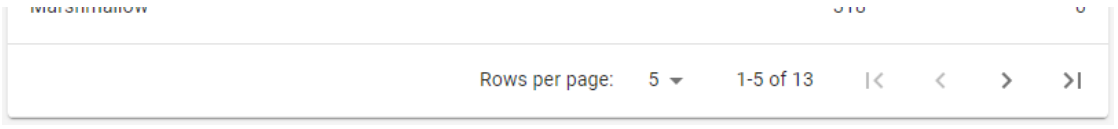
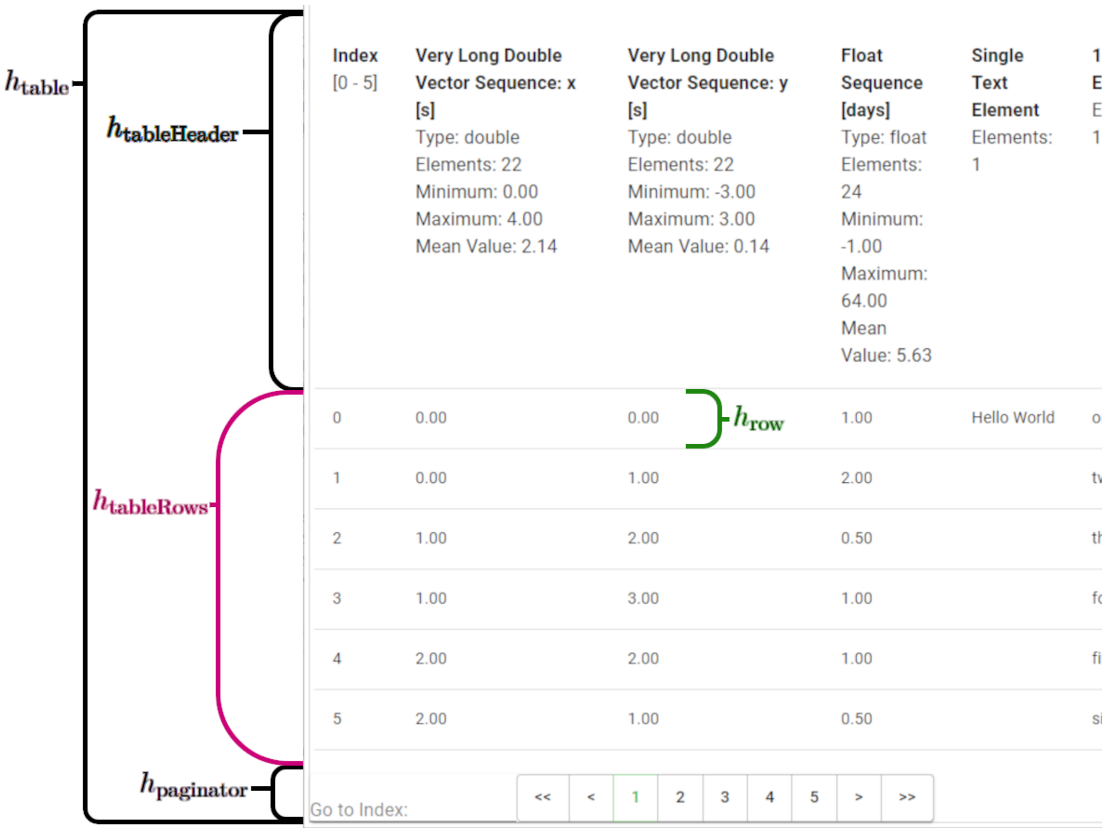


Figure 4.2: Material-UI TablePagination control. (Image source: [Matc])

control. The TablePagination is actually less adaptable to the layout and features of the desktop client, and lacks the page number buttons entirely. Both, Pagination and TablePagination, lack the 'Go to Index' number box. Instead, a custom Paginator control was composed out of Material-UI elements such as Buttons in a ButtonGroup, and an Input with an InputLabel, using a Paper element as container. The Paginator element requires several callback methods to be passed to it. These callbacks allow the Paginator to be fairly agnostic of what it is supposed to paginate. Any effects that happen to the table after navigating using the Paginator are defined with the callbacks. However, the table view is the only component that uses the Paginator right now.

The text view also interacts with pagination. The data being displayed is a single page with one row. No other rows are going to be displayed on the text view, so this is a reasonable shortcut in terms of communication over the network. Views have to use a workaround for dealing with pagination. The server will not send any data in case the page settings (i.e., page size and where the current page starts) are the same as the last time the page settings were changed. It will only return data, if the page settings have changed. This will often pose a problem when switching to a view, or when initially starting the application. An example where this is especially easy to see is the text view. After the text view was loaded, the page settings are always set to a page size of one and a starting index of zero (i.e., the first element). If one were to close the application here, and start the application afterwards (which defaults to the text view), it would again request a page size of one with the first element in the data being the starting element of the page. This means that the page settings would not change. The text view therefore would not receive any data. A simple workaround has been added to the text and map views. In the text view, the settings are set to a page size of two, and right afterwards are set back to a page size of one. In case of the table view, the workaround was not added; each time a table page is loaded (i.e. when the table view is switched to, or when



| Index [0 - 5] | Very Long Double Vector Sequence: x [s] Type: double Elements: 22 Minimum: 0.00 Maximum: 4.00 Mean Value: 2.14 | Very Long Double Vector Sequence: y [s] Type: double Elements: 22 Minimum: -3.00 Maximum: 3.00 Mean Value: 0.14 | Float Sequence [days] Type: float Elements: 24 Minimum: -1.00 Maximum: 64.00 Mean Value: 5.63 | Single Text Element Elements: 1 | 1 E E 1 |
|---|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 1.00 | Hello World | o |
| 1 | 0.00 | 1.00 | 2.00 | | tv |
| 2 | 1.00 | 2.00 | 0.50 | | tł |
| 3 | 1.00 | 3.00 | 1.00 | | fc |
| 4 | 2.00 | 2.00 | 1.00 | | fi |
| 5 | 2.00 | 1.00 | 0.50 | | si |

Go to Index: << < 1 2 3 4 5 > >>

Figure 4.3: Dimensions that are used in the calculation for how many rows fit on a table page.

page navigation happens), it only loads the correct page. For the map view, the pages have no specific meaning, as polygon and mesh data is not paged, however, it still needs to update page settings in order to get data, so the same workaround as in the text view is added. This combination means that the views always load the data if available, even the table view, which does not contain the workaround.

When using pagination on a table with a given height, the number $n_{\text{tableRows}}$ of rows that fit in the available space has to be calculated. Given the height $h_{\text{table}}$ of the entire table (including the header and paginator), the height $h_{\text{tableHeader}}$ of the header, and the height $h_{\text{paginator}}$ of the paginator, the height $h_{\text{tableRows}}$ that is available to the data rows (see Figure 4.3) is as follows.

$$h_{\text{tableRows}} = h_{\text{table}} - h_{\text{tableHeader}} - h_{\text{paginator}}$$

With the calculated available space, and the height $h_{\text{row}}$ of the individual rows, $n_{\text{tableRows}}$ can be found.

$$n_{\text{tableRows}} = \left\lfloor \frac{h_{\text{tableRows}}}{h_{\text{row}}} \right\rfloor$$

However, this results in the layout breaking in some edge cases, showing one line too many, which is then cut off by the paginator. The following formula was found to be working in those edge cases.

$$n_{\text{tableRows}} = \max\left(\left\lfloor \frac{h_{\text{tableRows}} - 1\text{px}}{h_{\text{row}}} \right\rfloor, 0\right)$$

$h_{\text{row}}$ and $h_{\text{paginator}}$ (static) are expected to be the same every time, even on different sessions. However, $h_{\text{tableHeader}}$ (dynamic) depends on the browser size and the incoming data. That is because column headers for vectors show minimum, mean, and maximum values in addition to the number of elements that are shown on every column header. $h_{\text{table}}$ (dynamic) depends on the window size of the browser as well. Incoming data can vary between sessions, but the window size can change at any time. Thus, the calculation needs to be dynamic, and has to be done whenever the browser is resized. The React Measure [Read] library helps with that.

Alternatively, the table could be built using a scrollable area, which would allow $n_{\text{tableRows}}$ to be set to any number, regardless of available space on the screen. The approach with $n_{\text{tableRows}}$ being variable mimics the behaviour of the desktop client and was chosen for that reason.

### 4.2.3 Porting Polygons and Meshes & Implementing Map View

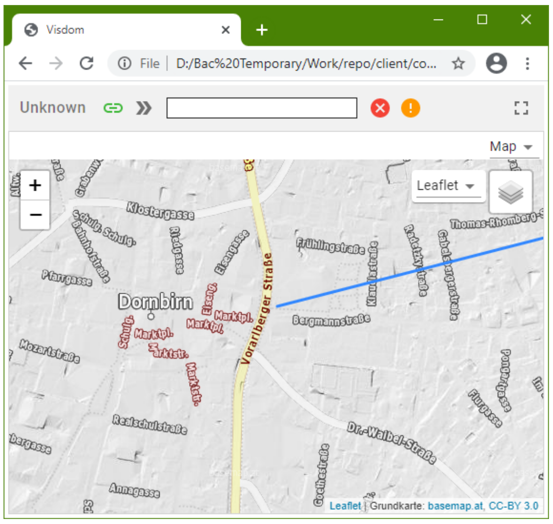
As opposed to text and vector type data, parsing polygon and mesh data was implemented using the visitor design pattern in the other components of the Visdom system. The visitor pattern was not replicated for the web client - instead, for simplicity, polygons and meshes are implemented like the other data types.

After the text and table view, the map view was developed last. The view started out as a simple debugging view that showed the polygons in a text representation. This view was then extracted into the DebugMapComponent, and GoogleMapComponent and LeafletMapComponent were added. A dropdown box at the top right (see Figure 4.4) can be used for switching between the map types. The DebugMapComponent has been removed and is thus not selectable in the dropdown box.

### 4.2.4 Improving Map View Performance By Discarding Polygons

To users, only polygons that are visible on screen can provide information directly. Polygons outside the map view bounds would be needlessly passed along to the map react components. These polygons can be discarded beforehand, in order to save valuable memory space, and consequently, calculation times.

Figure 4.5 illustrates the current approach. The blue rectangle represents the bounds of the map view. Bounding boxes of polygons are calculated beforehand, when assembling the vertices of the polygons from the parsed data. For all given polygons, the process is described in Algorithm 4.1.


Visdom
File | D:/Bac%20Temporary/Work/repo/client/co...
Unknown
Map
Leaflet
Leaflet | Grundkarte: basemap.at, CC-BY 3.0
(a) Dropdown box closed

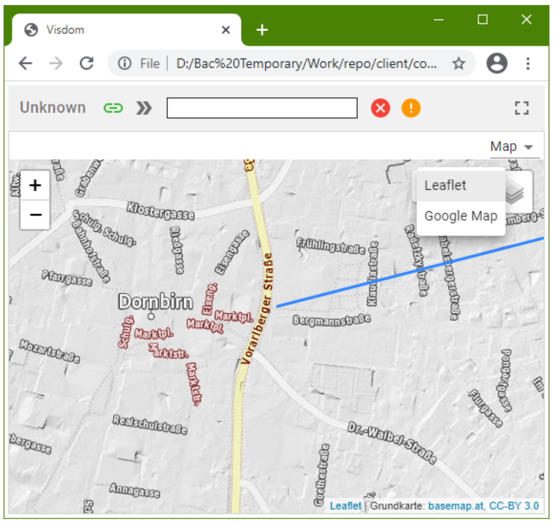
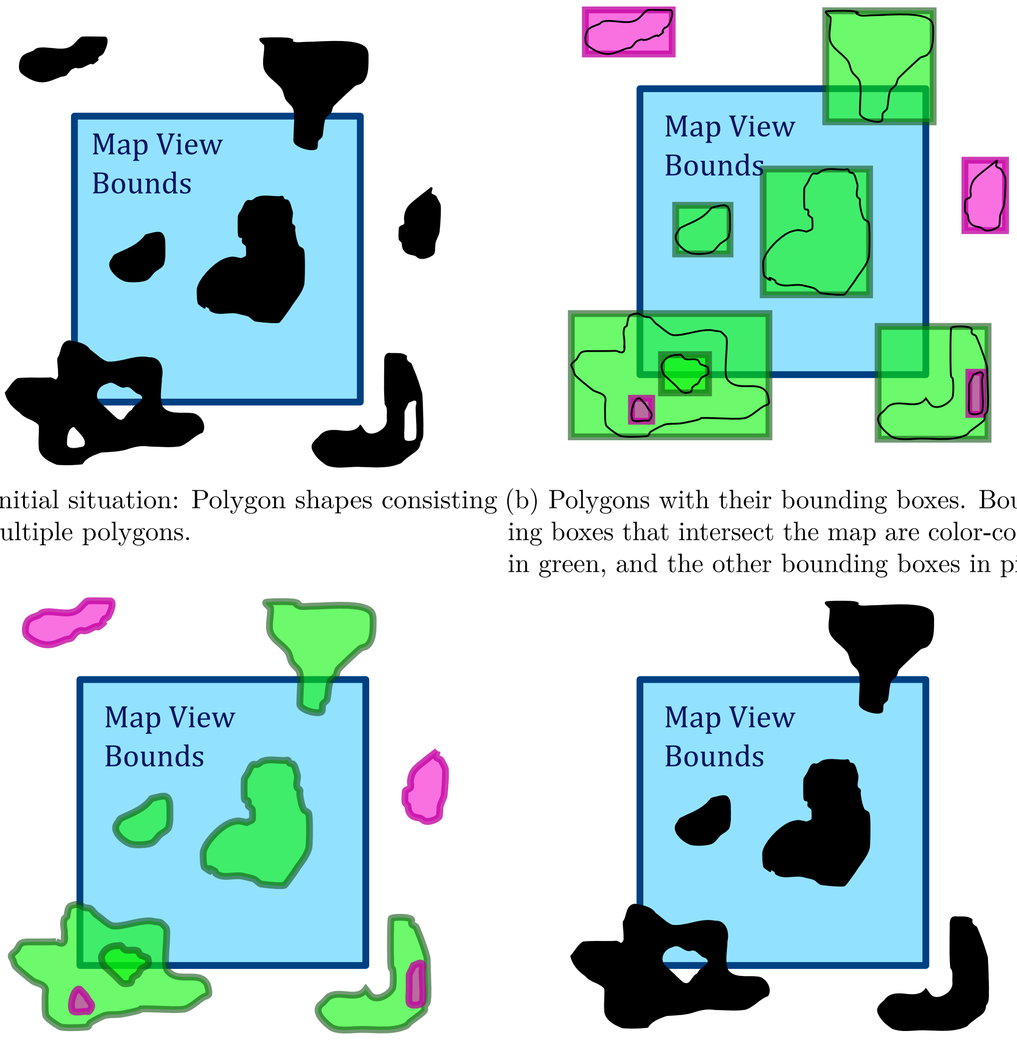

Visdom
File | D:/Bac%20Temporary/Work/repo/client/co...
Unknown
Map
Leaflet
Google Map
Leaflet | Grundkarte: basemap.at, CC-BY 3.0
(b) Dropdown box open

Figure 4.4: Map type selection dropdown box in the map view.

**Algorithm 4.1:** Finding the bounding box of a polygon

```
1  $north, east, south, west \leftarrow$ null;
2  foreach $Vertex\ v\ in\ polygon$ do
3      if $v.latitude > north$ then
4          $north \leftarrow v.latitude$;
5      end
6      if $v.longitude > east$ then
7          $east \leftarrow v.longitude$;
8      end
9      if $v.latitude < south$ then
10         $south \leftarrow v.latitude$;
11     end
12     if $v.longitude < west$ then
13         $west \leftarrow v.longitude$;
14     end
15 end
```



(a) Initial situation: Polygon shapes consisting of multiple polygons.

(b) Polygons with their bounding boxes. Bounding boxes that intersect the map are color-coded in green, and the other bounding boxes in pink.

(c) Polygons that are to be discarded are color-coded in pink

(d) Polygon shapes after the reduction

Figure 4.5: Example showing polygons and the bounds of a map view. If a polygon's bounding box does not intersect with the map, it is discarded.

As the vertices have to be iterated through once for assembling the polygon objects anyway, it is fairly efficient to determine the polygons bounding box here. Two rectangles $R_1$ and $R_2$ that are parallel to the axes intersect if the following condition is true, given that $\mathrm{n}(R_i)$, ... , $\mathrm{w}(R_i)$ are the coordinates (north, ... , west) of the rectangle $R_i$:

$$\neg\,[(\mathrm{s}(R_1) > \mathrm{n}(R_2)) \lor (\mathrm{w}(R_1) > \mathrm{e}(R_2)) \lor (\mathrm{n}(R_1) < \mathrm{s}(R_2)) \lor (\mathrm{e}(R_1) < \mathrm{w}(R_2))]$$

In more easily understandable terms, for two rectangles that are parallel to the axes to NOT intersect, one of them needs to be completely aside of the other in any direction.

Only the polygons with bounding boxes that intersect the map view bounds will be passed to the map components. All polygons discarded this way are invisible. As seen in Figure 4.5b, polygons may lie completely outside of the map view, but still have a bounding box that intersects the map view.

The intersection checks need to be done every time the map boundaries change. Map navigation (panning and zooming) and resizing the browser window cause map boundaries to change. React Measure is not required here, as the maps can already detect browser resizing. React Leaflet provides an ‘onresize’ event, and google-maps-react provides an ‘onBoundsChanged’ event, both of which fire on their respective map type whenever the browser resizes.

CHAPTER 5

# Evaluation

This chapter contains information about testing and evaluation during and after development. The evaluation consists of several user performed tasks, in which the user navigates the map in simple interactions like panning and zooming. The general aim of these tests is to determine whether the map is computationally performant enough to not be detrimental to the user experience. The evaluation as well as the measuring method are described in Section 5.2. Section 5.1 describes the files that were used during development and evaluation. Tthe process for cleaning the results of the evaluation is described in Section 5.3.

## 5.1 Data

This section describes the input data used for developing and testing the application. In Section 5.1.1, the `.visdom` files are explained, including `floodPolygonsMap.visdom`, which loads ESRI Shapefiles. The various Shapefiles used with it are listed in Section

| Project File | Meshes* | Polygons* | Vertices |
|---|---|---|---|
| capitalsAt.visdom | 3 | 9 | 6270 |
| capitalsAtSimplified.visdom | 2 | 9 | 612 |
| SyntheticSplitPolygon.visdom | 0 | 4 | 70 |
| floodPolygonsMap.visdom with... | | | |
| ...splitPolygon.shp | 0 | 19 | 14 679 |
| ...sparseVorarlberg.shp | 0 | 771 | 97 006 |
| ...vorarlbergBig.shp | 0 | 12 415 | 2 146 902 |
| ...vorarlbergPart.shp | 0 | 697 | 80 208 |

*Polygons within meshes are not counted. Multiple subpolygons of one polygon shape (i.e. several polygons with the same id) are counted individually.

Table 5.1: Project files and ESRI Shapefiles, and the data they contain..

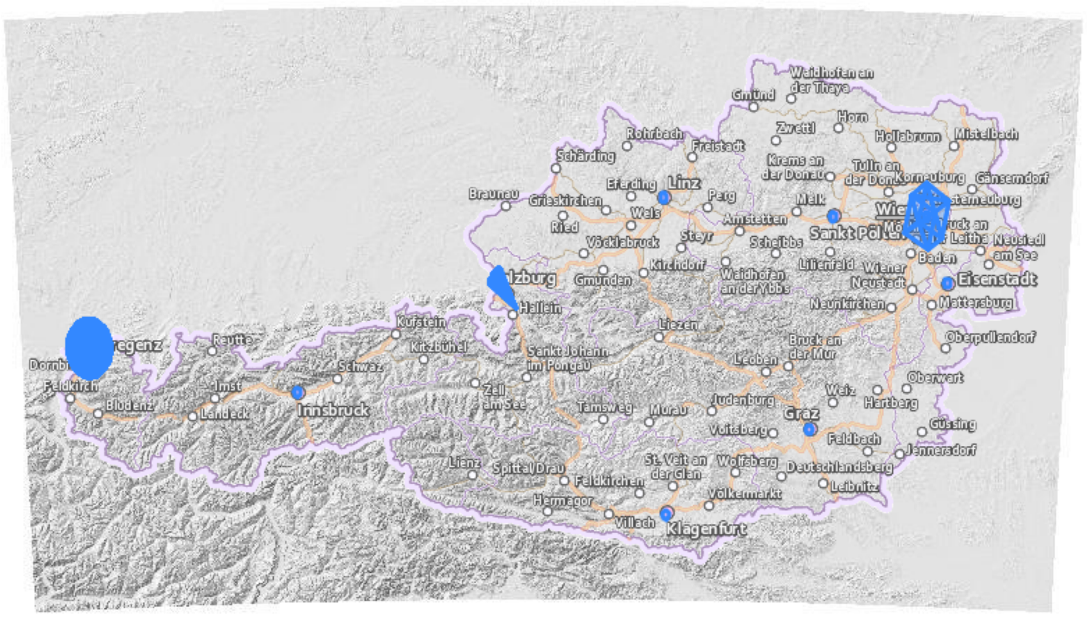


Figure 5.1: Project file capitalsAt.visdom in Leaflet mode

5.1.2. Table 5.1 lists both types of files with the data they include for the web client to handle.

### 5.1.1 Projects

`capitalsAt.visdom` (see Figure 5.1) has circles around the Austrian capitals, and three mesh primitives: A box in Vienna, a sphere in Bregenz, and a cone in Salzburg. The circles around the capitals have 20 segments each. The project was made to test polygon and mesh data rendering capabilities of the web client during development.

`capitalsAtSimplified.visdom` (see Figure 5.2) is similar to `capitalsAt.visdom`, but with 10 segmented circles instead of the 20 segmented ones, a less detailed sphere, and without the cone.

`SyntheticSplitPolygon.visdom` contains two polygons with the same id, each with a hole. The web client does not expect multiple polygons with the same id, but it will not have any problems with them, as long as they do not contain holes. The Google Map does not display such configurations correctly. Figure 5.3 shows the graphical bug. The map should show two circles with holes. However, one of the circles is filled solidly. The Leaflet map does not have any problem with the setup.

`floodPolygonsMap.visdom` loads an ESRI Shapefile and sends it to the web client as a polygon. Different Shapefiles were used in testing the application. Figure 5.4 shows the shapefile `vorarlbergPart.shp`.

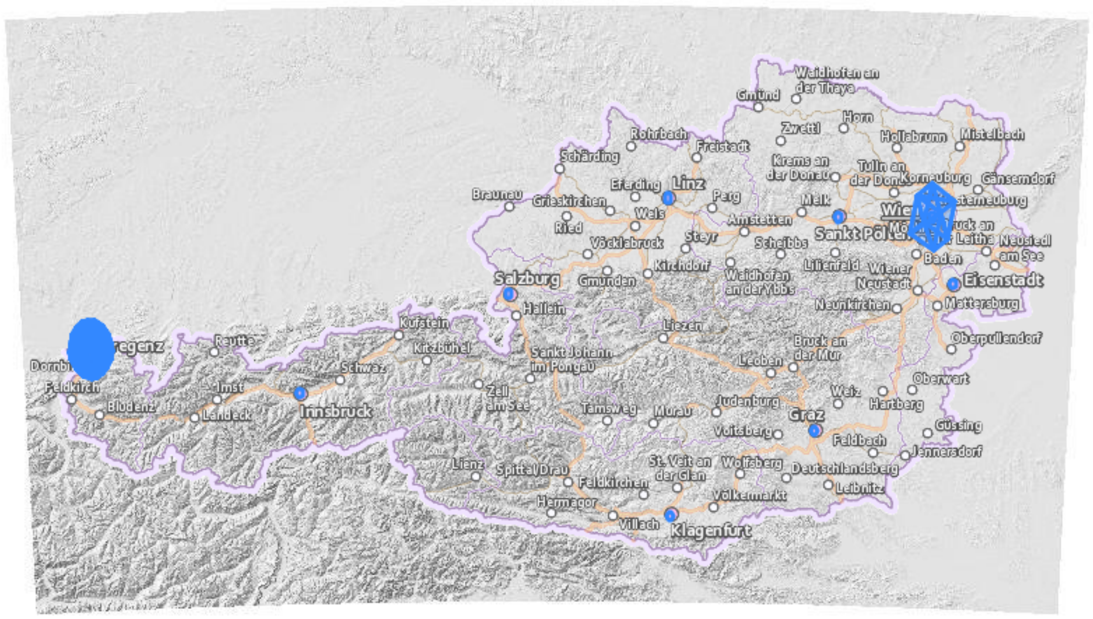


Figure 5.2: Project file capitalsAtSimplified.visdom in Leaflet mode

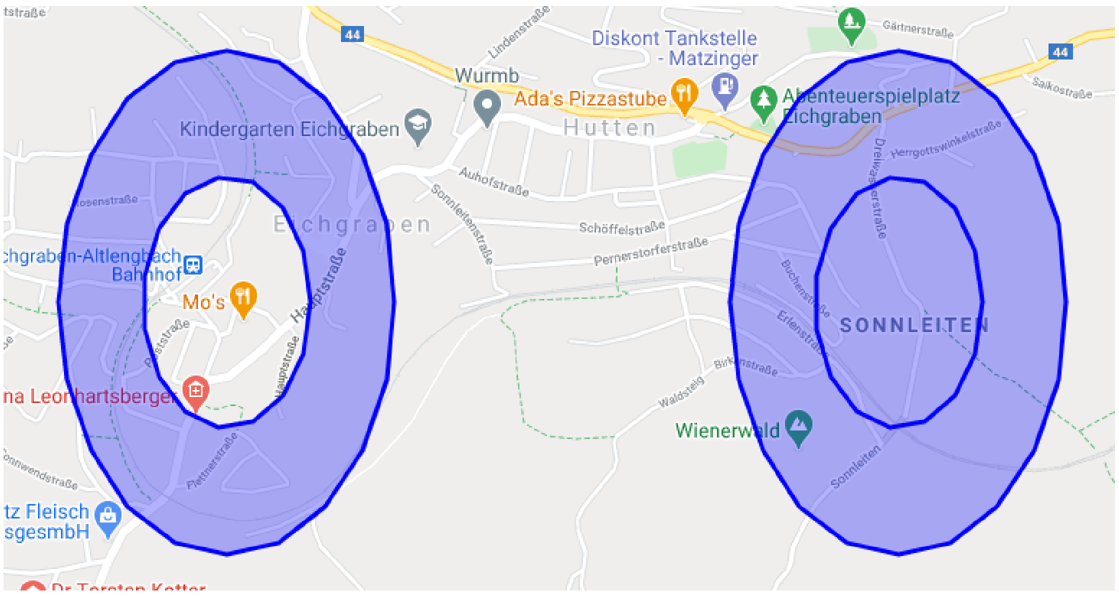


Figure 5.3: Project file SyntheticSplitPolygon.visdom in Google mode

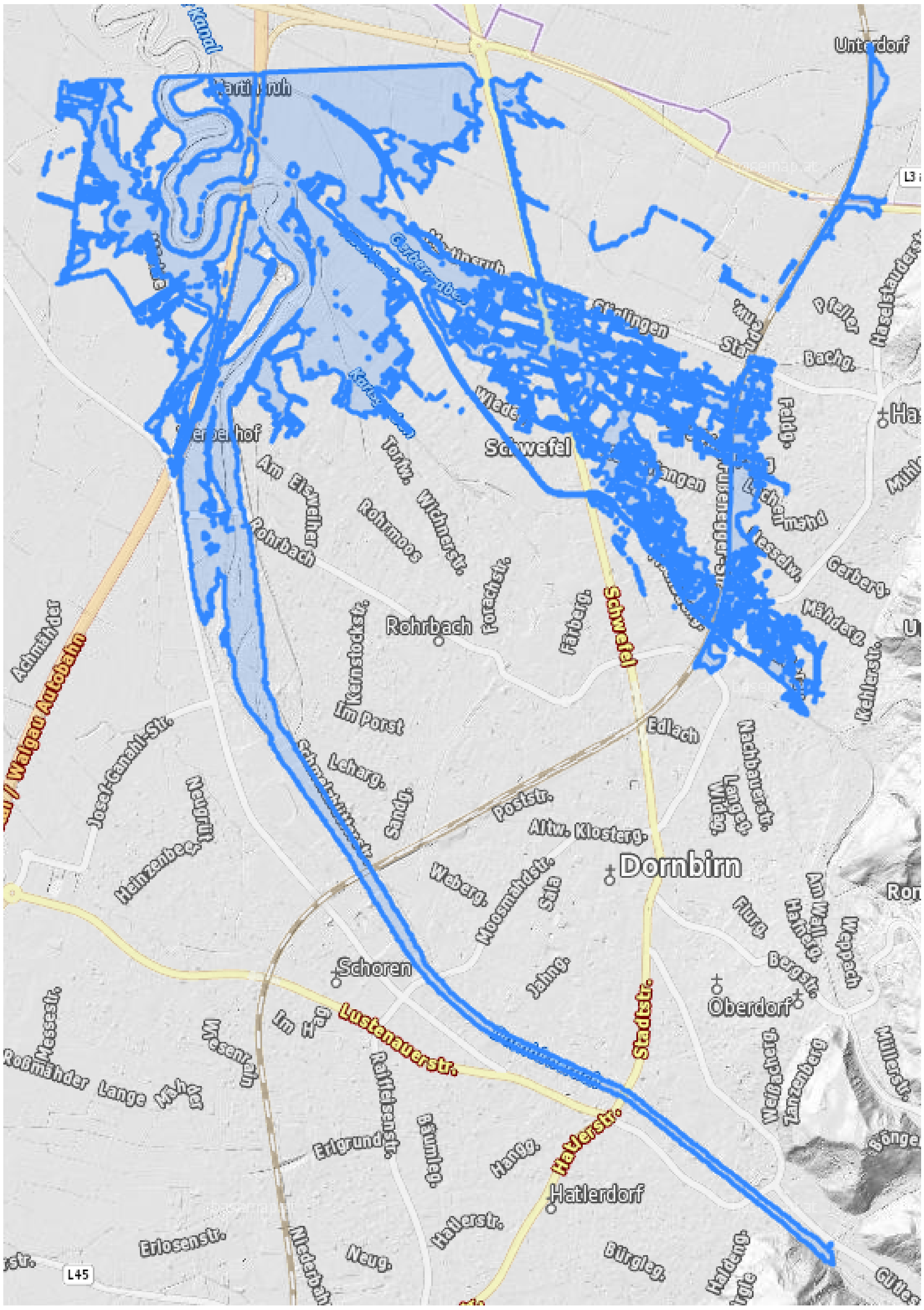


Figure 5.4: Project file floodPolygonsMap.visdom in Leaflet mode

### 5.1.2 Shapefiles

All of the Shapefiles that were used are different subsets of a dataset provided by VRVis, containing the flood extent boundaries of an $HQ_{100}$ in Vorarlberg.

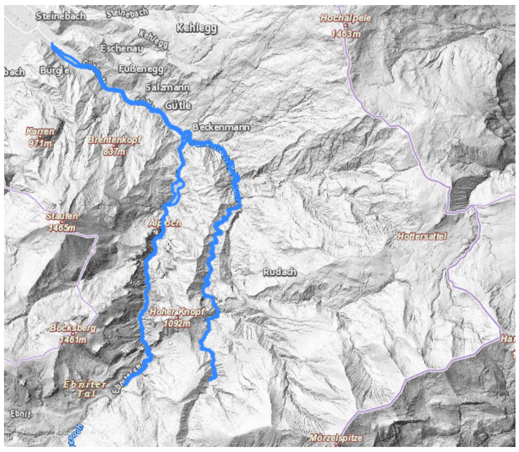
- `splitPolygon.shp` (see Figure 5.5a) includes polygons with duplicate ids.
- `sparseVorarlberg.shp` (see Figure 5.5b) consists of polygons chosen so that they are sparsely spread out over an area with some larger gaps between them.
- `vorarlbergBig.shp` (see Figure 5.5c) is a big shape file of roughly 51.008 KB and 2 146 902 vertices.
- `vorarlbergPart.shp` (see Figure 5.5d) is the default Shapefile in `floodPoly-gonsMap.visdom` and has 80 208 vertices.

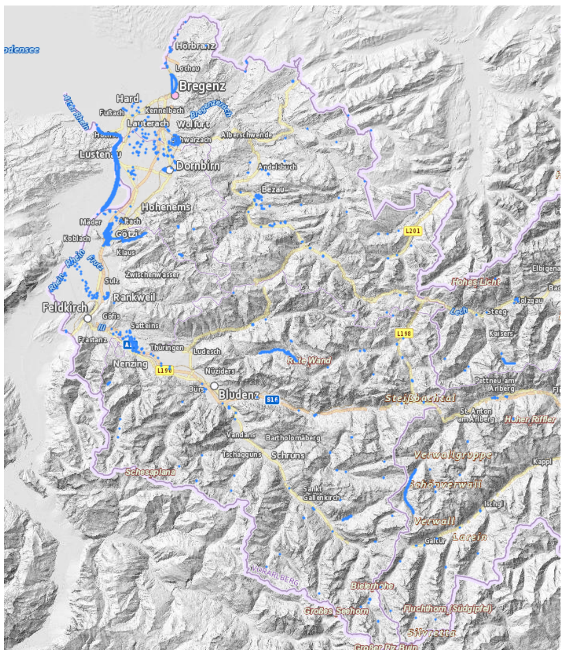
## 5.2 Test Procedure

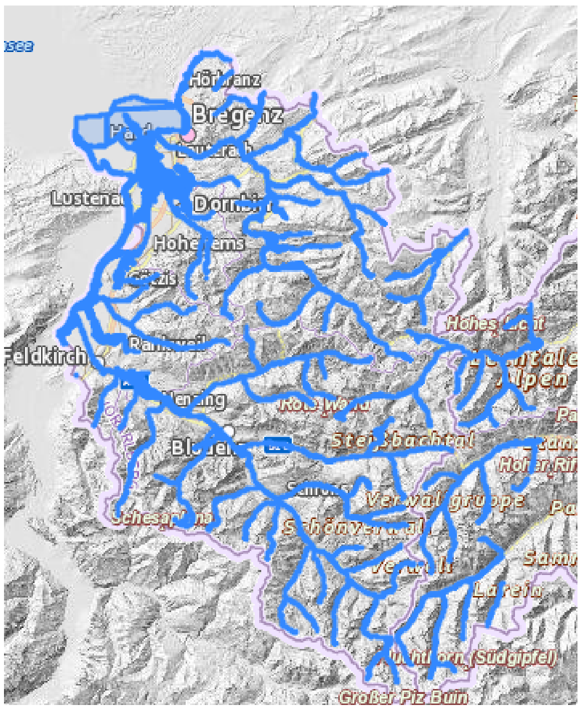
The developed application was evaluated with manual tests. The tasks are defined in Table 5.2. Some of these tasks are performed in the proximity of polygons on the map, whereas others are not. This distinction is important, because the it can demonstrate whether the rendering of polygon overlays has an influence on the clients computational performance. During the tasks, the frame times in milliseconds are recorded. The tasks have to be performed in multiple runs for each map type. All tasks in all runs are executed using the project file `floodPolygonsMap.visdom` and the shapefile `vorarlbergPart.shp`.

| # | Task | $Pos_{start}$ | $Pos_{end}$ | $z_{start}$ | $z_{end}$ | Scen. Type |
|---|---|---|---|---|---|---|
| 1 | Pan | Schwarzach | St. Anton | 14 | 14 | T → F |
| 2 | Pan | Bödele | Schwarzach | 11 | 11 | T |
| 3 | Pan | St. Anton | Vienna | 11 | 11 | F |
| 4 | Zoom out | St. Anton | St. Anton | 17 | 12 | F |
| 5 | Zoom out | Dornbirn | Dornbirn | 17 | 12 | T |
| 6 | Zoom in | St. Anton | St. Anton | 12 | 17 | F |
| 7 | Zoom in | Dornbirn | Dornbirn | 12 | 17 | T |
| 8 | Pan | St. Anton | Schwarzach | 14 | 14 | F → T |

Table 5.2: Tasks within the manual test procedure to be done with both map types. The different scenario types (labelled “Scen. Type” in the table) describe whether the task happens near polygons in the overlay (T) or not (F). “F → T” and “T → F” mean the scenario moves into or out of such an area respectively. “$Pos$” and “$z$” mean the position and zoom level respectively as pre- ($Pos_{start}$, $z_{start}$) and post-conditions ($Pos_{end}$, $z_{end}$).


(a) splitPolygon.shp


(b) sparseVorarlberg.shp


(c) vorarlbergBig.shp.shp

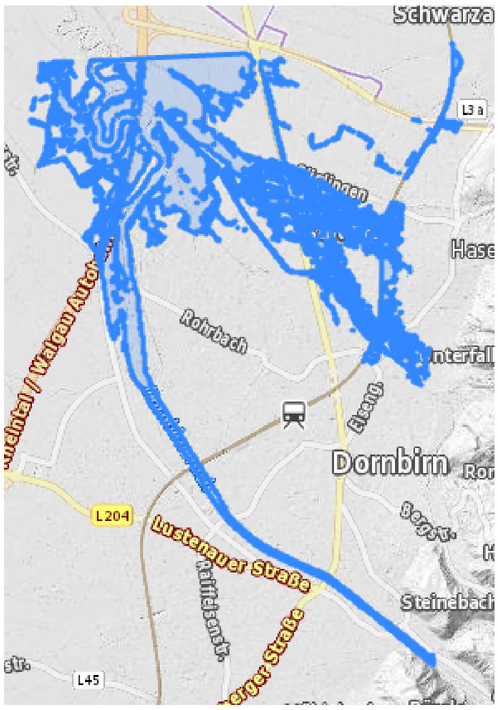

(d) vorarlbergPart.shp

Figure 5.5: ESRI Shapefiles

### 5.2.1 Recording & Measuring

The measurements are done using the Chrome DevTools Profiler. Before each task in each run, the profiling is started, and after the task, the profiling is stopped again. Each profiling is stored in a folder structure, seperating the Leaflet and Google Map profiles at the top level, then seperating run one and two - in the run folders, one profiling (as json file) per task is stored. After all profiles are done, the following JavaScript function is used to bring all frame durations from a profile into MATLABs array format:

```
    matlabize = () => {let arr = UI.panels.timeline
._flameChart._model._frameModel._frames.map((f) =>
f.duration); let st = "[" ; for(let i = 0; i < arr.length;
i++) {num = arr[i]; st += num; st += " " }; st += "]";
return st }
```

This function is adapted from an answer on Stack Overflow [Sta]. To use it, one needs to open the Chrome DevTools for the web app in a seperate window ($dt_1$), and then open another Chrome DevTools window ($dt_2$) for the previously opened $dt_1$. In $dt_1$, the 'Performance' tab has to be open, and a profile needs to be loaded. Then, in $dt_2$, in the 'Console' tab, when the `matlabize` function has been declared, the frame durations in milliseconds (in MATLABs array format) can be obtained by entering '`matlabize()`' into the console. This list of frames may contain long *idle frames*, in which no graphical updates happen. This may also be due to no interaction happening at all.

A measurement coming directly from the framework would be preferred. Unfortunately, there is no obvious solution to do that for comparing lag in two different libraries for React. While one can count the times a component rendered, that does not say much about how many graphical updates happened, because the two libraries, `react-leaflet` and `google-maps-react` seem to behave differently in this aspect. For example, when dragging, the Google Map only re-renders as a component when dragging stops, while the Leaflet Map re-renders many times during the dragging. In this example, the number of times the Leaflet Map re-renders during a set time interval is much more meaningful. In the end, the re-renderings within an interval are not meaningfully comparable. The same issue exists for measuring map load times, as already discussed in Section 4.1.6.

The following adaptions have been made to the application for the sole purpose of simplifying the testing purpose and making the process more streamlined and reproducable: Lines have been added between the points where the testing user has to pan. That means one line each for the connections described in Table 5.2. To make it easier to start each task, I tried to assign the number keys on the top of the keyboard to the start locations — pressing one of them should take the map to the related point. This did not work properly for multiple reasons, e.g., the Google Map never reacted to these key presses, and on the Leaflet Map, pressing the key with the digit 6 also caused a change in the zoom.

## 5.3 Cleaning the Data

The raw data, i.e., the results as mentioned in Section 5.2.1, were cleaned up with the steps described in this section to remove some influence of human error on the test results. That is, because the time between starting the profiling and actually starting the interaction with the map can take different durations every time. This can influence the outcome while not having to do with the maps performance. The first frame in which the event corresponding to the beginning of the interaction is determined and frames before that are discarded. This happens in order to eliminate the time before the actual interaction. For zoom events, the DevTools Timeline is searched for a wheel event, for pan events, it is searched for a mousedown event. The wheel event fires on scrolling the mouse wheel, whereas the mousedown event fires when the mouse button is pressed, which is required for a dragging interaction. Scrolling is indicative of zooming, and dragging is indicative of panning. Thus, these events were used. The frame that contains these events is considered the first frame of the interaction. Figuring out the last frame was more tedious: In this case the last fired event indicative of the interaction has to be either on the last considered frame or on a frame before it. It is possible that important frames for the interaction happen after the event fires. In a zooming task this is usually the case, because when scrolling the mouse wheel steps, the last 'wheel' event fires, but only afterwards does the map update to the proper zoom level. To account for this, the last graphical change is manually found by comparing the frames to each other. In panning interactions the event 'mouseup' can happen without any graphical differences afterwards - however it still might be the case that tiles of the map have to load. This is taken into consideration by manually looking for the last graphical change, however that introduces human error.

Furthermore, especially the Leaflet measurements beginning events ('mousedown' or 'wheel') lie within a frame that includes the idle time before the interaction. The frame might start at a significantly earlier time than the actual interaction. Thus the first frame has to be discarded, as its data is not comparable, and highly dependent on the map being used (Google or Leaflet Map) and the amount of time taken between starting the profiling and the actual interaction.

## 5.4 System Specifications

For testing (i.e., the procedure specified in the sections above), visualizations, and calculations the system described in Table 5.3 has been used.

## 5.5 Statistics and Visualization

This section describes the calculations and the software tool used for transforming the cleaned data into charts and comparable values. MATLAB R2020a is used for calculating and plotting them. The system specified in Section 5.4 was used. Among means and medians, the Concluding Chapter uses the ratio $r_{bf(t)}$ as a comparable value. Given a set $S$ of frames $f$ and a FPS threshold $t$:

$$S_{bf(t)} = \{f \in S : \mathrm{FPS}(f) < t\}$$

$$r_{bf(t)} = \frac{\left|S_{bf(t)}\right|}{|S|}$$

FPS($f$) here denotes how many frames of the same duration as a given frame $f$ would fit in a second. FPS are usually defined the other way around, as it describes the actual number of frames that happened within a given second. For the thesis, FPS values are directly calculated from the frame duration. This is done by first multiplying the frame duration in milliseconds by 1000 in order to convert them to seconds, i.e., the amount of seconds it took to render each frame. The FPS are the reciprocal of that value. Therefore, given a frame $f$ and its duration $d(f)$ in ms, the FPS of that frame are calculated using the formula $\mathrm{FPS}(f) = \frac{1}{d(f)*1000}$.

$S_{bf(t)}$ is the set of all frames in $S$ that have an associated FPS value lower than the given threshold $t$. The developer web pages of Google and Mozilla both talk about 60 FPS for good user experience: In the Chrome DevTools pages [Chr], the statement “Users are happy when animations run at 60 FPS.” can be found, while the MDN Web Docs pages say “The goal frame rate for in web site computer graphics is 60fps.” [mdn]. Since 60 FPS is considered a target for websites, one of the values of $t$ that are explored is 60. Other values of $t$ are 30 and 10. $r_{bf(t)}$ is the ratio of “bad frames” with FPS lower than $t$ to the total number of frames.

| | |
|---:|---|
| Operating system | **Windows 10 Home** |
| OS version | **10.0.19041 Build 19041** |
| CPU | **Intel® Core™ i7-6700HQ CPU @ 2.60GHz, 2592 Mhz, 4 Core(s), 8 Logical Processor(s)** |
| Architecture | **64 bit** |
| RAM | **16 GB** |
| GPU | **NVIDIA GeForce GTX 965M** |
| GPU driver version | **27.21.14.5241** |
| On-board graphics | **Intel® HD Graphics 530** |
| On-board graphics driver version | **22.20.16.4749** |
| CUDA driver version | **11.0** |
| CUDA runtime version | **10.1** |
| Device | **Omen by HP Laptop W8Z00EA#ABD** |
| Visdom server version | **2020.2002.0-904ed6eab3 Dev** |

Table 5.3: System Specifications of the machine for Testing.

CHAPTER 6

# Results

In this chapter, results are presented, and some interesting aspects of them are highlighted. Conclusions from these results will be drawn in Chapter 7. As described in Section 5.5, MATLAB is used for statistical calculations made here.

The durations in milliseconds for all frames that were considered part of the interactions can be seen in the tables in the appendix. The tables and charts in this section contain mean, median, and $r_{bf(t)}$ values calculated from the data. The tables in this chapter are rounded to two decimal places.

Tables 6.1 and 6.2 show the mean and median FPS values respectively of each task for each run. A row represents a run, and a column represents a task.

| | Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Run | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Leaflet Map 1 | 44.31 | 34.07 | 51.70 | 66.37 | 55.63 | 48.04 | 48.59 | 40.18 |
| Leaflet Map 2 | 45.93 | 39.88 | 19.67 | 63.31 | 54.56 | 42.66 | 40.57 | 44.33 |
| Google Map 1 | 45.47 | 75.56 | 52.71 | 36.69 | 32.47 | 42.33 | 41.06 | 48.01 |
| Google Map 2 | 50.01 | 41.96 | 55.13 | 40.75 | 24.73 | 53.04 | 41.86 | 45.89 |

Table 6.1: Mean FPS per task per run rounded to two decimal places. Higher is better.

| | Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Run | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Leaflet Map 1 | 58.63 | 52.68 | 42.06 | 59.75 | 59.83 | 57.70 | 39.28 | 51.28 |
| Leaflet Map 2 | 57.83 | 58.76 | 19.56 | 59.93 | 59.71 | 42.62 | 30.20 | 58.83 |
| Google Map 1 | 57.98 | 36.65 | 59.84 | 25.00 | 26.16 | 44.68 | 33.25 | 58.82 |
| Google Map 2 | 54.31 | 40.51 | 54.35 | 48.43 | 9.75 | 51.11 | 51.13 | 52.00 |

Table 6.2: Median FPS per task per run rounded to two decimal places. Higher is better.

| | Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Map | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Leaflet Map | 45.27 | 36.55 | 44.13 | 64.73 | 55.06 | 45.43 | 44.82 | 42.37 |
| Google Map | 47.61 | 55.65 | 53.83 | 39.00 | 28.77 | 48.77 | 41.48 | 47.02 |

Table 6.3: Mean FPS per task per map rounded to two decimal places. Higher is better.

| | Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Map | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Leaflet Map | 58.12 | 58.62 | 21.91 | 59.83 | 59.77 | 53.73 | 32.88 | 58.29 |
| Google Map | 55.70 | 39.23 | 59.28 | 38.80 | 17.20 | 50.38 | 39.32 | 57.38 |

Table 6.4: Median FPS per task per map rounded to two decimal places. Higher is better.

| Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 46.48 | 43.52 | 48.77 | 56.68 | 46.82 | 47.38 | 43.14 | 44.68 |

Table 6.5: Mean FPS per task rounded to two decimal places. Higher is better.

Tables 6.3 and 6.4 are similar to Tables 6.1 and 6.2, but use maps instead of runs as rows, i.e., the frames from the first and second run with each map are concatenated into one set. A row represents one of the two maps, and a column represents a task.

Tables 6.5 and 6.6 show the mean and median FPS for each task over all runs over both maps. This is useful for figuring out how difficult each individual task is for the system. Tables 6.7 and 6.8 show the mean and median FPS for each of the maps.

Table 6.9 shows the $r_{bf(t)}$ for each $t \in 10, 30, 60$ for each run and each task. Table 6.10 is similar, but uses maps instead of runs as rows, and also has an ‘all’ column per different $t$, and an ‘all’ row. The ‘all’ column contains the $r_{bf(t)}$ over all frames in a task, i.e., frames from both Google and Leaflet Map runs. The ‘all’ row contains the $r_{bf(t)}$ over all tasks with a specific map.

| Task Nr. | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 57.86 | 51.85 | 47.21 | 59.24 | 59.19 | 51.31 | 37.69 | 58.10 |

Table 6.6: Median FPS per task rounded to two decimal places. Higher is better.

| Leaflet | Google |
|---|---|
| 48.17 | 46.13 |

Table 6.7: Mean FPS per map rounded to two decimal places. Higher is better.

| Leaflet | Google |
|---|---|
| 58.71 | 52.65 |

Table 6.8: Median FPS per map rounded to two decimal places. Higher is better.

| | $r_{bf(10)}$ | | | | $r_{bf(30)}$ | | | | $r_{bf(60)}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Task | L1 | L2 | G1 | G2 | L1 | L2 | G1 | G2 | L1 | L2 | G1 | G2 |
| 1 | 0.12 | 0.06 | 0.05 | 0.04 | 0.37 | 0.33 | 0.34 | 0.32 | 0.70 | 0.69 | 0.66 | 0.65 |
| 2 | 0.44 | 0.30 | 0.36 | 0.38 | 0.44 | 0.30 | 0.36 | 0.38 | 0.70 | 0.65 | 0.55 | 0.69 |
| 3 | 0.09 | 0.08 | 0.04 | 0.03 | 0.43 | 0.97 | 0.23 | 0.24 | 0.70 | 0.97 | 0.52 | 0.64 |
| 4 | 0.03 | 0.03 | 0.19 | 0.17 | 0.14 | 0.08 | 0.55 | 0.43 | 0.59 | 0.50 | 0.76 | 0.76 |
| 5 | 0.07 | 0.06 | 0.38 | 0.51 | 0.20 | 0.18 | 0.50 | 0.66 | 0.54 | 0.62 | 0.79 | 0.83 |
| 6 | 0.12 | 0.16 | 0.18 | 0.12 | 0.40 | 0.44 | 0.37 | 0.30 | 0.61 | 0.71 | 0.74 | 0.68 |
| 7 | 0.22 | 0.23 | 0.34 | 0.28 | 0.44 | 0.50 | 0.46 | 0.38 | 0.70 | 0.73 | 0.78 | 0.74 |
| 8 | 0.11 | 0.14 | 0.10 | 0.13 | 0.46 | 0.34 | 0.28 | 0.34 | 0.74 | 0.68 | 0.64 | 0.67 |

Table 6.9: Bad Frame Ratios $r_{bf(t)}$ per task and run rounded to two decimal places. Ln is short for the $n_{th}$ Leaflet run; Gn analogously for Google. Lower is better.

| | $r_{bf(10)}$ | | | $r_{bf(30)}$ | | | $r_{bf(60)}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Leaflet | Google | all | Leaflet | Google | all | Leaflet | Google | all |
| 1 | 0.09 | 0.05 | 0.06 | 0.35 | 0.33 | 0.34 | 0.69 | 0.65 | 0.67 |
| 2 | 0.38 | 0.37 | 0.38 | 0.38 | 0.37 | 0.38 | 0.68 | 0.63 | 0.66 |
| 3 | 0.08 | 0.03 | 0.06 | 0.56 | 0.24 | 0.41 | 0.76 | 0.58 | 0.67 |
| 4 | 0.03 | 0.18 | 0.08 | 0.11 | 0.48 | 0.22 | 0.54 | 0.76 | 0.61 |
| 5 | 0.07 | 0.44 | 0.18 | 0.19 | 0.58 | 0.31 | 0.58 | 0.81 | 0.65 |
| 6 | 0.14 | 0.14 | 0.14 | 0.42 | 0.33 | 0.37 | 0.66 | 0.7 | 0.69 |
| 7 | 0.23 | 0.31 | 0.27 | 0.47 | 0.42 | 0.44 | 0.72 | 0.76 | 0.74 |
| 8 | 0.13 | 0.12 | 0.12 | 0.40 | 0.31 | 0.35 | 0.71 | 0.65 | 0.68 |
| all | 0.1 | 0.12 | | 0.33 | 0.35 | | 0.67 | 0.67 | |

Table 6.10: Bad Frame Ratios $r_{bf(t)}$ per task and map rounded to two decimal places. Lower is better.

CHAPTER 7

# Conclusion

As previously discussed, comparing the different map types is a nontrivial task. Looking at the tables and figures from the results section, we can see that in direct competition between Leaflet and Google Maps, such as the mean FPS (see Table 6.7), median FPS (see Table 6.8), and 'bad frame' ratios (see row 'all' in Table 6.10), Leaflet Maps performs slightly better or equally good as the Google Maps map (over all tasks). Looking at these values individually for each task (see Tables 6.3, 6.4, and 6.10), we can see Google Maps performing better than Leaflet Maps in some tasks. Either way, it is a very close call. As of now, FPS is a popular measure as far as animations and user experience are concerned. From this perspective Leaflet does perform slightly better.

Intuitively, it seems that the map performs worse with many vertices in memory. To simplify the wording of this discussion, the terms "difficult tasks" and "easy tasks" are used. User interactions in difficult tasks consist of navigation around points on the map near many vertices, whereas the interactions in easy tasks do not.

Three pairs of tasks are examined, each with a easy and a similar difficult task. They are similar in the sense that the users need to do essentially the same interaction but for different locations. These pairs are Tasks 2 (**d**ifficult) and 3 (**e**asy), 5 (d) and 4 (e), and 7 (d) and 6 (e). Of these pairs, the first one (2 and 3) is probably the one with the least similar tasks, as it is a panning task and the distances to pan in each task are not equal. The other tasks are zooming tasks, where the tasks in each pair have the same pre- and postconditions (other than the zoom levels). In Figures B.1 – B.5, we can see the mean FPS, median FPS, and $r_{bf(t)}$ of these pairs. We can see that in most comparisons, the lazy tasks perform better. This supports the assumption of large numbers of vertices in close proximity of the map view center decreasing the maps performance.

Figure B.6 shows the FPS at all frames (sorted in ascending order by FPS). The regions defined by the thresholds at 10, 30 and 60 FPS are difficult to see due to some frames having very high FPS values. Figure B.7 is similar, but clamps the FPS values to the range $[0, 70]$. This makes the zones more visible, while still visualizing how many frames were in which zone. The values at the very top of each graph do not match the actual FPS of those frames. Using the regions, we can quickly see how many frames in a specific run were significantly worse or better than one of the thresholds. For example, one can quickly see that more than half the frames in each run for Task 7 were below the target of 60 FPS. For both maps there are a few outlier frames with relatively high FPS. The graphs also show that some runs had significantly more frames in the recordings than other runs. Note that this does not mean that they necessarily took longer in time, seeing as each frame can last varying amounts of time.

File `capitalsAtSimplified.visdom` contains a sphere with many vertices. This file is demonstrative of a correlation between number of vertices in the React tree and the perceived lag in the application. Different zoom levels on the sphere mean different numbers of vertices being included in the tree. On a zoom level of 15 with  150-260 vertices, the Leaflet Map is very usable, whereas interaction with the Google Maps does not feel fluid at all on this level, possibly because of the noticeable time it takes until the new vertices are loaded. On zoom level 14 with ∼380 to slightly under 1200 vertices, the Leaflet Map has very noticeable lag - on the lower vertex counts this is not to a degree where the map is uncontrollable (mostly  5-7 FPS), but somewhere around 500 vertices it gets significantly worse to control. By 1000 vertices, the point where the lag on the Leaflet Map makes it a bad experience has been reached (mostly  3 FPS). As for the Google Map on the same zoom level, on the lower vertex counts, sometimes  4 FPS are reached, however between those frames are many frames with 60 FPS and higher. Similarly on the higher vertex counts, it often gets to approximately 2 FPS, but then has many much better frames in between. This phenomenon illustrates a problem with Google Maps as far as perceived lag and quantifiable data from the profiler is concerned: Values such as the mean or median FPS and the previously discussed bad frame ratio $r_{bf(t)}$ (see Section 5.5) are massively influenced by the many good frames between the few but significant bad frames.

The following is a synthetic example to further illustrate the problem: Given one frame that takes 5000 ms followed by 20 frames that take 1 ms each, users would very much notice the lag from the first frame, whereas the 20 other frames might not make much of a difference to them. The mean duration would be $\frac{5000+20}{21} \approx 239.0476$ ms. Converted to FPS, that makes  4.1833 FPS. In comparison to the 0.2 FPS that a frame of 5000 ms would have, that is quite a significant difference. The perceived lag might be higher than what the average FPS might suggest. This problem has yet to be solved, in order to have truly representative and comparable data - measuring perceived lag might solve this, for example with a big study with many independent users.

In Section 2.6, some lessons learned about the other maps were noted. The following paragraphs, review how some of the guidelines extracted from the related work were applied and how some are not yet fulfilled.

Comparing performance between very different maps, is to my knowledge not reasonable, as the maps are quite different, have different toolsets, or might take shortcuts by displaying overlays that were rendered into bitmaps beforehand. The Google and Leaflet Map types offer very similar functionality to the user, and are similarly suited for our web application, which is why the comparison of performance between them made sense.

As visible in all of the screenshots of the map view in this thesis, the overlays do not have any label or legend. The implementation can as of this point be used for all kinds of maps that are able to make use of polygon overlays. Though, since it was made with the intention of using it as a flood extent map, it should be extended with labels and legends. An option for keeping it use-case independent, is making labels and colors to be defined in the project file.

In Section 4.1.6, some issues regarding the individual map types were mentioned. As mentioned there, the basemap.at tiles are limited to Austria, therefore, for using the Leaflet map type outside of Austria, a different tile set will have to be used.

The polygon overlays are in blue, while blue has been avoided in the background. Thus, the problem of overlay colors being ambiguous is avoided. This has to still be tested by people with types of color blindness. The controls for the map are intuitive — they function much like in the popular tool Google Maps, and there are no complex drawing or measuring tools.

The level of detail shown on the map depends on the input data. As far as the web client is concerned, there is a technologically imposed limitation. Javascript can only represent numbers with a limited accuracy, as a finite number of bits can only represent so many numbers. The epsilon (`Number.EPSILON`), i.e., the smallest difference between two numbers, lies at approximately $2.22 * 10^{-16}$. It is possible that the libraries, especially those providing access to the maps, limit the level of detail further, however no information on this was found.

CHAPTER 8

# Future Work

One of the subsequent steps in the Visdom web client is a WebGL view. The WebGL view was the original end goal in this endeavour. It is supposed to implement a hybrid rendering strategy, where the Visdom web client and server render the same 3D scene. The server renders higher quality images, but as it will take time for the image to be rendered and transferred over a network connection, the client is responsible for rendering frames in between. This strategy would enable a practical compromise between fluid interactions and high quality images. The WebGL view would be the web clients equivalent of the OpenGL window on the desktop client.

More elaborate studies about this version or future versions of the web client's map view are needed, including a measure of performance that can be fairly compared between different map types. My assumption is, that the most significant result would be given by a large scale study about perceived lag with a large number of participants in disjoint groups - one for each map type. Then, questions like "Did you notice any annoying or distracting performance issues with the map when performing this task?" can be asked, and the percentage of people responding a certain way can be compared between questions and map types. A map type with a few frames with very low FPS can get better performance metric results by having many frames with high FPS. Asking users directly if what they perceived has been annoying or distracting might be a better approach, however it is depending on the questioned people's subjective judgement and patience, and would increase the amount of potential human error involved.

As far as performance increasing features go, the simple bounding box intersection check implemented on the web client is a good feature. Further performance improvements could come from more efficient communication with the server. If the web client's map bounds and zoom level are synchronized to the server, the server can then do two additional optimizations. Using the bounds, the server can take over the intersection check in order to avoid unnecessarily sending all polygons over the network and also reducing the workload on the web client. Using the zoom level, the server may decide to send a less detailed version of the polygon. This technique is called LOD (level of detail). For example, if a user zooms far out of the map, they will not be able to see a lot of details that are created with many vertices. However, if they zoom in close, they might be able to differentiate different vertices from each other. This means that, on low zoom levels, we do not need as much detail, and consequently less vertices. This might bring valuable performance improvements.

The ability to switch between multiple map types is in the current version of the web client for comparing different configurations. This feature can of course be kept for as long as it is deemed useful. As soon as a conclusive decision on which map system is better suited for the flood map use case has been reached, it is reasonable that the switching feature is removed.

APPENDIX A

# Maps Licensing & Pricing List

## A.1 Map Engines

- **Leaflet**
  - *Comments*: Currently used
  - *License & Limitations*: Leaflet © 2010-2021 Vladimir Agafonkin: `https://github.com/Leaflet/Leaflet/blob/master/LICENSE`
  - *Pricing*: Free
- **OpenLayers**
  - *Comments*: –
  - *License & Limitations*: –
  - *Pricing*: Free
- **Mapbox GL JS**
  - *Comments*: –
  - *License & Limitations*: –
  - *Pricing*: Map load = when Mapbox GL JS initializes. Tiers:
    montly loads (ml) $\leq$ 50000: Free
    50001 $\leq$ ml $\leq$ 100000: \$ 5 per 1000 loads
    100001 $\leq$ ml $\leq$ 200000: \$ 4 per 1000 loads
    200001 $\leq$ ml $\leq$ 1000000: \$ 3 per 1000 loads
    (for more monthly loads, the website does not name a price and requests to contact sales)

- **Google Maps**
  - *Comments*: currently used
  - *License & Limitations*: –
  - *Pricing*: Tiers:
    First $ 200 worth of requests are free
    Maps Static API: $ 2 per 1000 requests
    Maps Embed API: Free
    Maps JavaScript API: $ 7 per 1000 requests

## A.2 Map Tiles

- **Host own tiles**
  - *Comments*: Higher costs/efforts
  - *License*: N/A
  - *Pricing*: N/A
  - *Limitations*: N/A
- **OSM Carto**
  - *Comments*:
  - *License*: Copyright: `https://www.openstreetmap.org/copyright`
  - *Pricing*: Free / financed by donations
  - *Limitations*: Heavy use forbidden without permission. OSM tile policies: `https://operations.osmfoundation.org/policies/tiles/`
- **Wikimedia Maps**
  - *Comments*: Some bugs visible on tiles
  - *License*: OSM Copyright applies: `https://www.openstreetmap.org/copyright`
  - *Pricing*: Free / Wikimedia is financed by donations
  - *Limitations*: `https://foundation.wikimedia.org/wiki/Maps_Terms_of_Use`
- **Thunderforest tiles**
  - *Comments*: Some cut off labels
  - *License*:

  - *Pricing*: Tiers:
    150000 Tile Requests / Month (TRM): Free
    1500000 TRM: £ 95 / M
    15000000 TRM: £ 198 / M
    150000000 TRM: £ 395 / M
  - *Limitations*:

- **Humanitarian Map Style**
  - *Comments*: –
  - *License*: see `https://www.openstreetmap.fr/open-data/`
  - *Pricing*: Free / Financed by donations
  - *Limitations*: Hosted by OSM France, see license in url above

- **wmflabs (WikiMedia Foundation Labs)**
  - *Comments*: Includes OSM and other tiles
  - *License*: ?
  - *Pricing*: ?
  - *Limitations*: ?

- **Stamen Maps**
  - *Comments*: Different map styles, some less conventional (Toner, Watercolor) than others (Terrain)
  - *License*: "Except otherwise noted, each of these map tile sets are ©Stamen Design, under a Creative Commons Attribution (CC BY 3.0) license."
  - *Pricing*: Free / Financed by donations
  - *Limitations*: –

- **ÖPNVKarte / OpenBusMap**
  - *Comments*: Focusses on transit lines
  - *License*: Tiles: CC BY-SA 2.0; Server: see Limitations
  - *Pricing*: Tiles are free to use, servers are not (see limitations)
  - *Limitations*: "Even though the map is available under an open licence, that does not mean our server capacity can be used in your project. Especially websites with a lot of traffic must be hosted on your own servers. If you need help with that, I can provide you an offer."

- **Map1.eu**
  - *Comments*: Limited to Europe

  - *License*: CC BY-NC-ND 3.0
  - *Pricing*: –
  - *Limitations*: –

- **OpenTopoMap**
  - *Comments*: –
  - *License*: CC-BY-SA
  - *Pricing*: Free / requires registration
  - *Limitations*: `https://opentopomap.org/about`

- **OpenMapSurfer**
  - *Comments*: Discontinued
  - *License*: CC-BY 4.0
  - *Pricing*: Free Plan for up to 20000 requests per day and 2000 requests per minute
  - *Limitations*: –

- **MapTilesApi**
  - *Comments*: –
  - *License*: OdbL
  - *Pricing*: Free Plan for up to 10000 requests per day and 56 requests per second
  - *Limitations*: –

- **CARTO Maps**
  - *Comments*: –
  - *License*: –
  - *Pricing*: Limited use free for a year, see `https://carto.com/pricing/`
    Individual: Limited access (50000 map loads per month, limited API access, ...) for $ 199 per month
    Enterprise: Full access, cost not public.
  - *Limitations*: ?

- **Mapbox Maps**
  - *Comments*: –
  - *License*: `https://www.mapbox.com/pricing/`

  - *Pricing*: Vecter Tiles API: free up to 200000 TRM
    Static Tiles API: free up to 200000 TRM
    Raster Tiles API: free up to 750000 TRM
    Other APIs also available
  - *Limitations*: ?

- **Omniscale**
  - *Comments*: –
  - *License*: –
  - *Pricing*: No free plan - Volume-based and user-based pricing available
    € 29 per month (25 € if paid yearly) → 250000 TRM and 10000 WMS requests per month
  - *Limitations*: “Nutzung auch für interne oder kostenpflichtige Anwendungen, sowie für das Asset-Tracking”

APPENDIX B

# Data Visualization

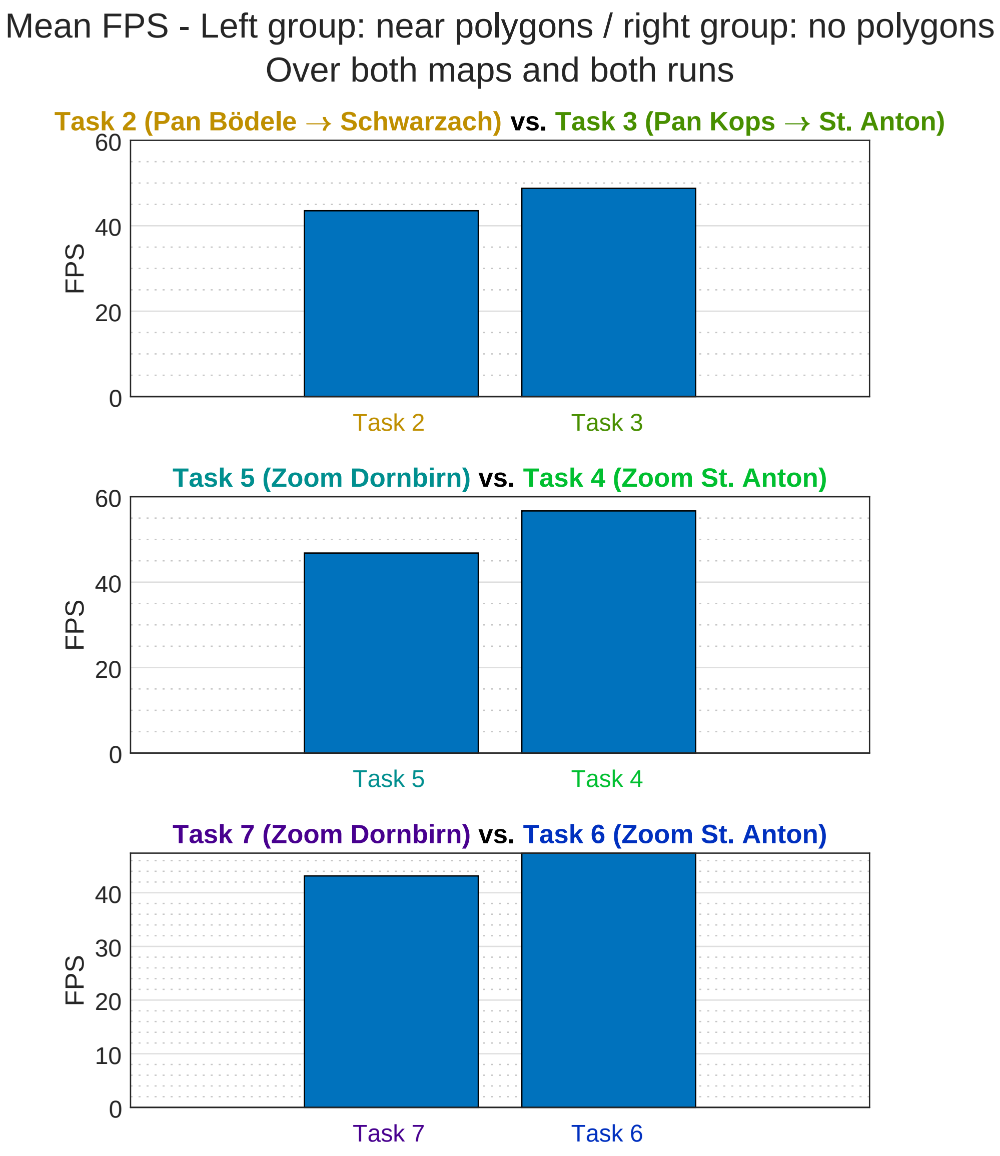


Figure B.1: Mean FPS of “difficult” tasks vs. “easy” tasks

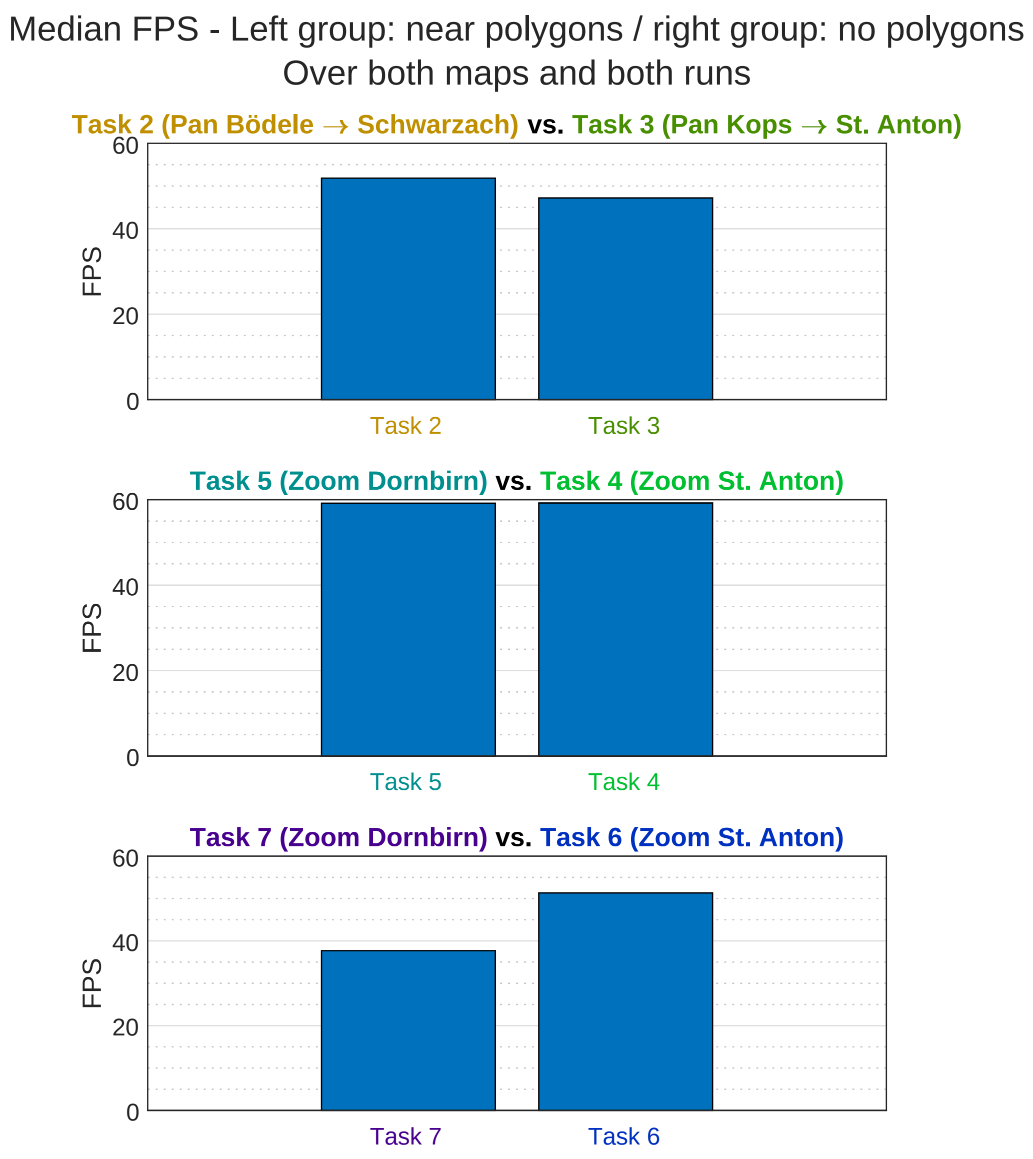


Figure B.2: Median FPS of “difficult” tasks vs. “easy” tasks

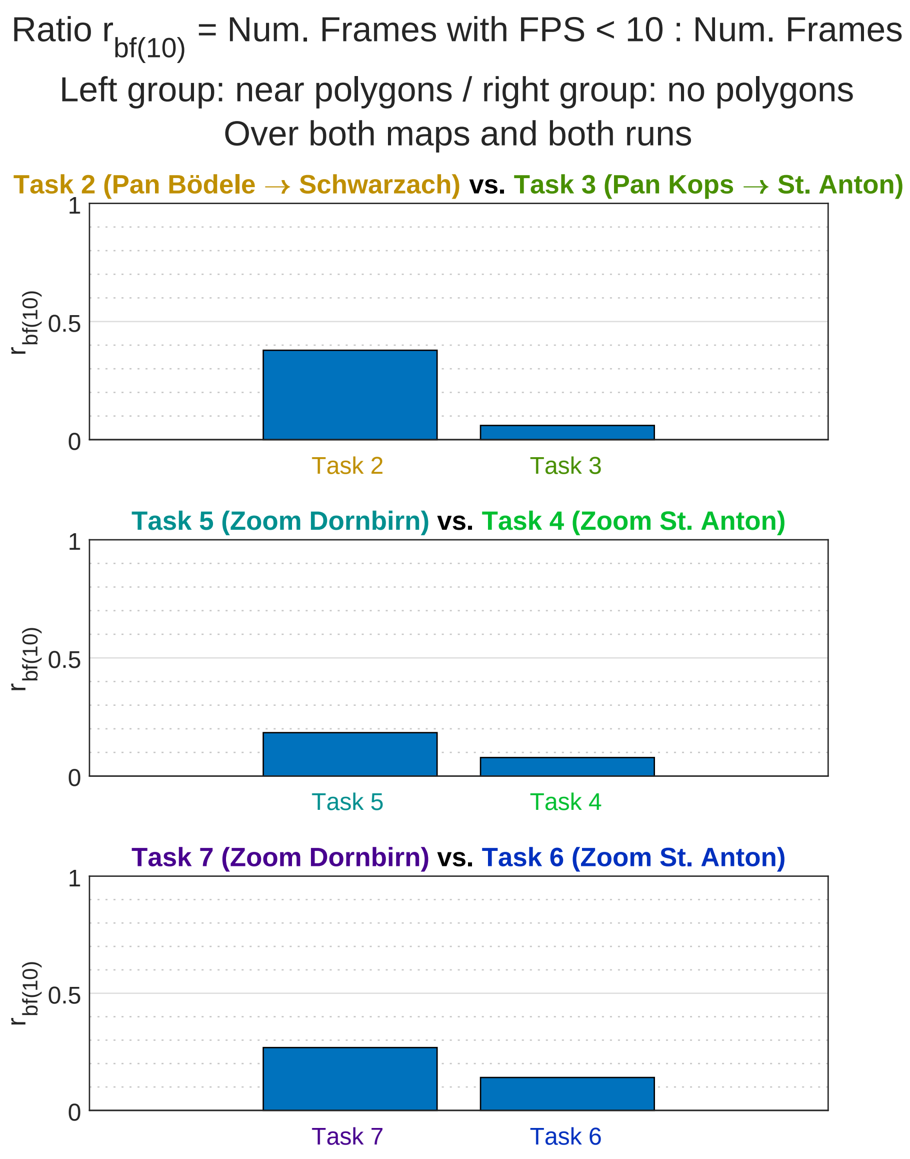


Figure B.3: $r_bf(10)$ of "difficult" tasks vs. "easy" tasks

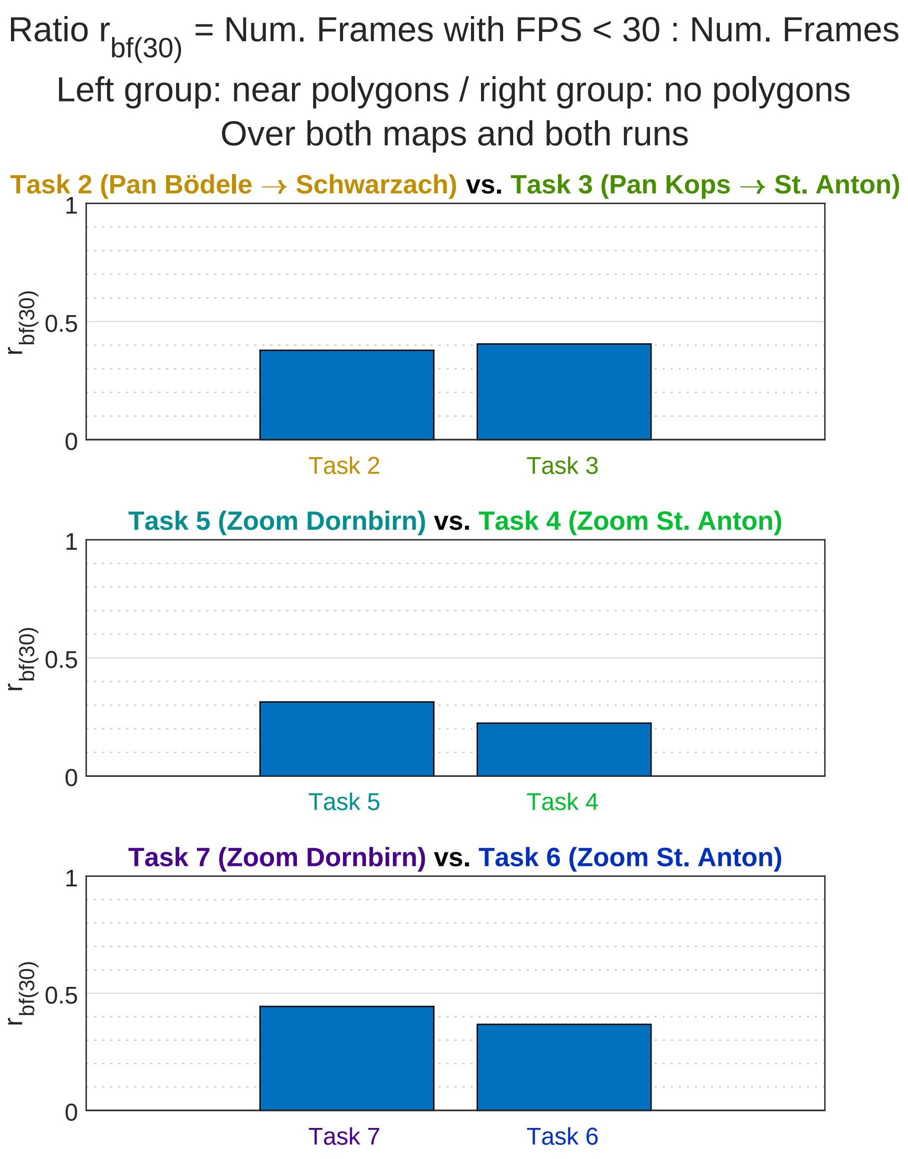


Figure B.4: $r_b f(30)$ of "difficult" tasks vs. "easy" tasks

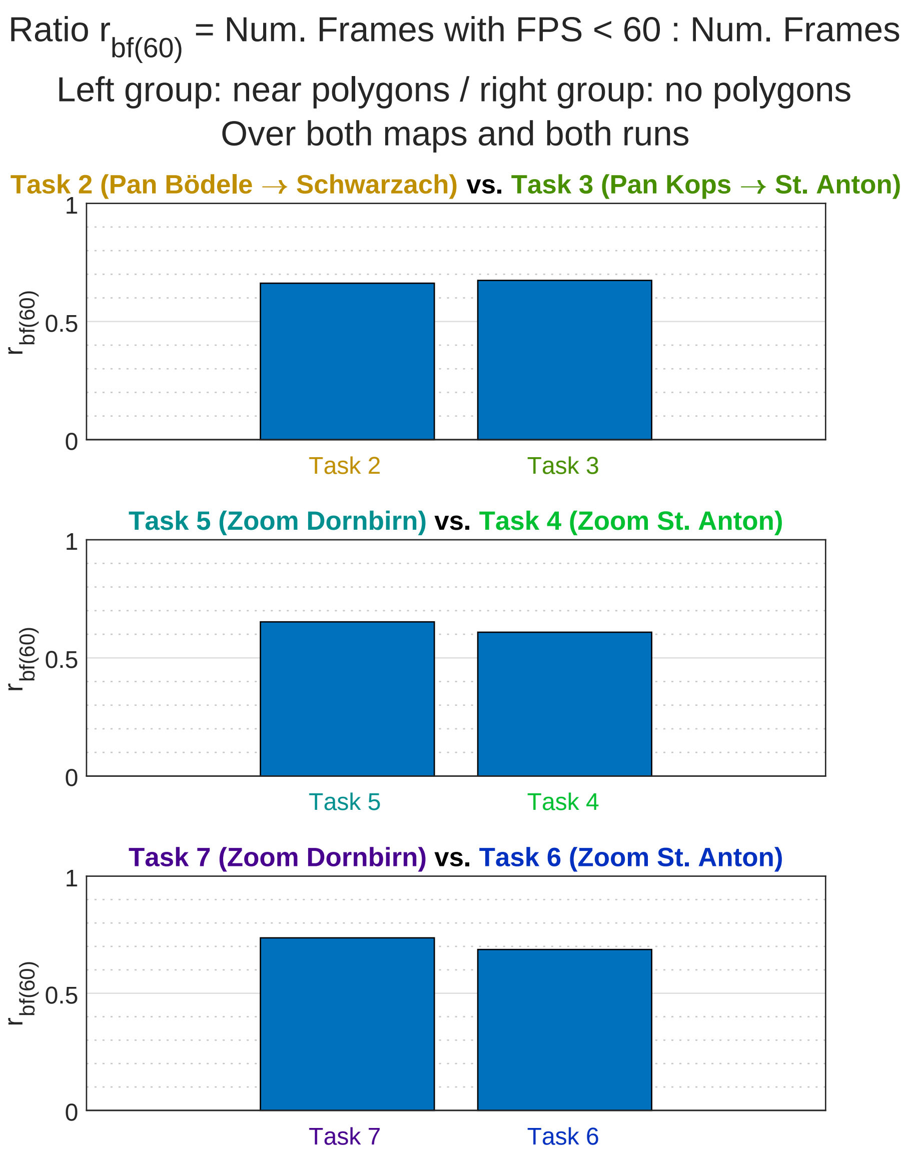


Figure B.5: $r_bf(60)$ of "difficult" tasks vs. "easy" tasks

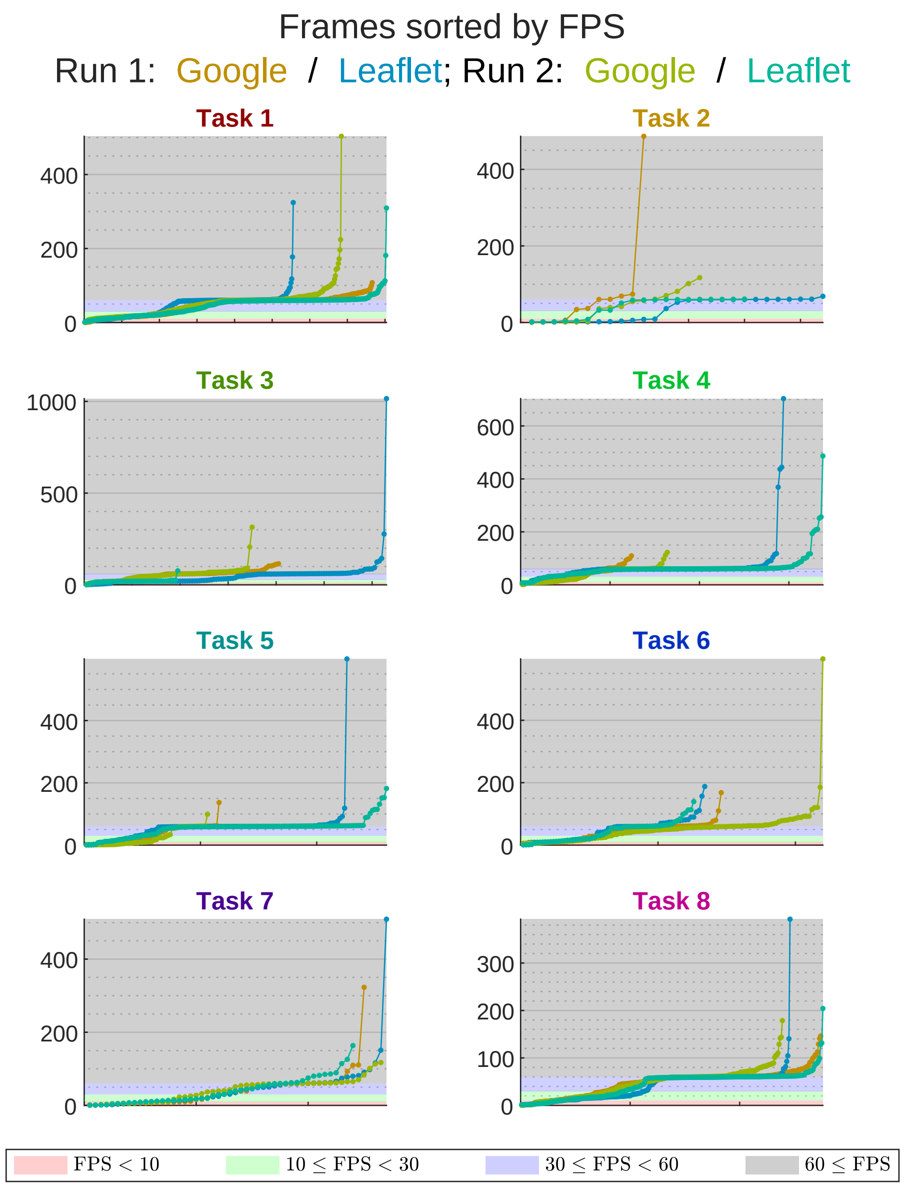


Figure B.6: FPS per task in ascending order.

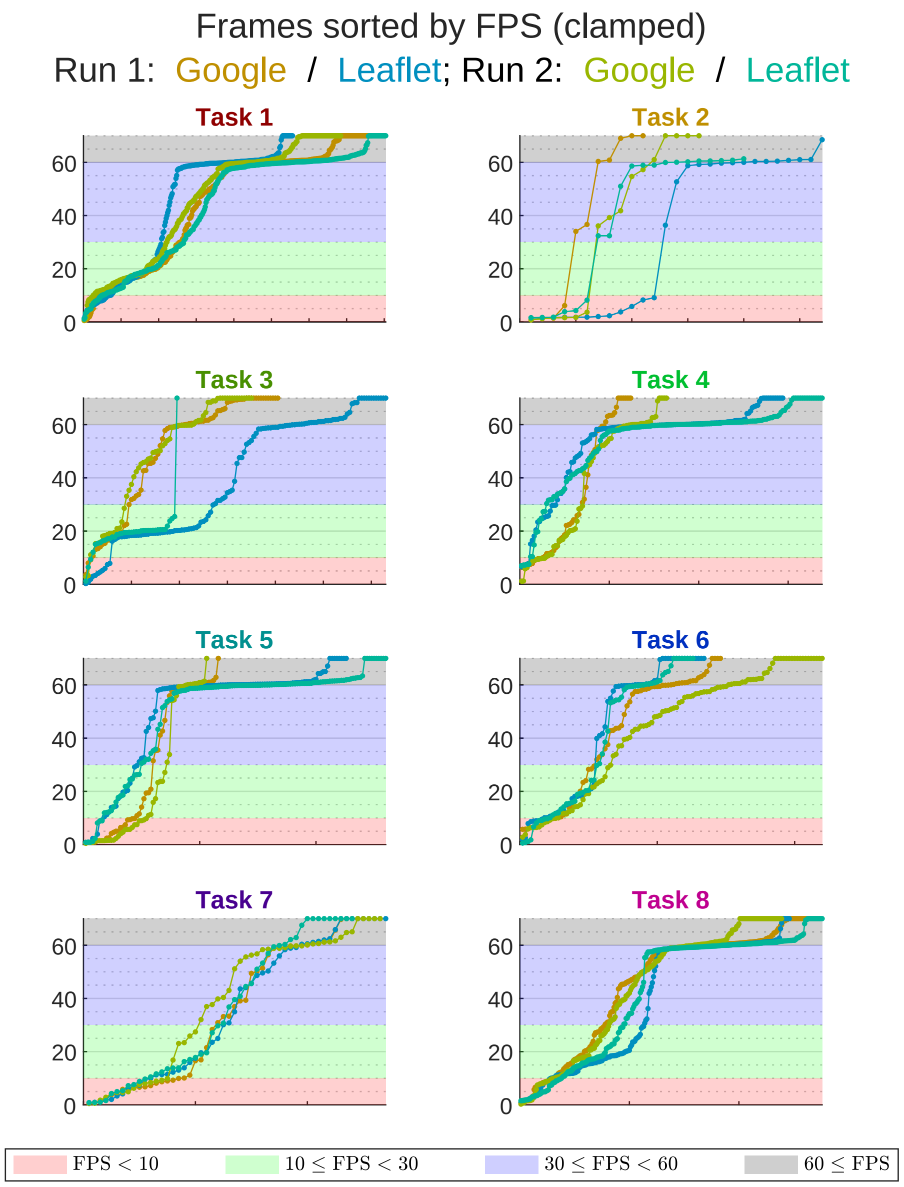


Figure B.7: FPS per task in ascending order, clamped to the range $[0, 70]$. This visualizes the distribution over the different zones of FPS more clearly.

# List of Figures

# List of Tables

# List of Algorithms

# Literature

[Abf] Abflussvorhersagen der Hydrographie Österreichs, bmlrt.gv.at. https://www.bmlrt.gv.at/wasser/schutz_vor_naturgefahren/hochwasserprognose/hw_prognose_at.html. Last accessed 2021.03.07.

[Ari] Ari Lerner. How To Write a Google Maps React Component. https://www.newline.co/fullstack-react/articles/how-to-write-a-google-maps-react-component/. Last accessed 2021.03.13.

[Bas] Basemap.at. https://basemap.at/. Last accessed 2021.03.13.

[Bay] Bayerisches Landesamt für Umwelt. Naturgefahren - Umweltatlas. https://www.umweltatlas.bayern.de/mapapps/resources/apps/umweltatlas/index.html?lang=de. Last accessed 2022.12.08.

[BL] Bundesministerium für Landwirtschaft, Regionen und Tourismus, Stubenring 1, 1012 Wien, Österreich and Land, Forst- und Wasserwirtschaftliches Rechenzentrum (LFRZ), Hintere Zollamtsstraße 4, 1030 Wien. eHORA - Natural Hazard Overview & Risk Assessment Austria. https://hora.gv.at/. Last accessed 2021.03.07.

[CEA07] CEA. Reducing the social and economic impact of climate change and natural catastrophes–insurance solutions and public-private partnerships. 2007.

[Cen20] Centre for Research on the Epidemiology of Disasters - CRED. Natural disasters 2019: Now is the time to not give up. 2020. https://cred.be/sites/default/files/adsr_2019.pdf. Last accessed 2021.03.07.

[Chr] Chrome Developers. Analyze Runtime Performance - Chrome. https://developer.chrome.com/docs/devtools/evaluate-performance/. Last accessed 2022.09.08.

[CKS$^+$15] D. Cornel, A. Konev, B. Sadransky, Z. Horváth, E. Gröller, and J. Waser. Visualization of Object-Centered Vulnerability to Possible Flood Hazards. *Computer Graphics Forum*, 34(3):331–340, June 2015. http://doi.wiley.com/10.1111/cgf.12645.

[CKS+16] D. Cornel, A. Konev, B. Sadransky, Z. Horváth, A. Brambilla, I. Viola, and J. Waser. Composite Flow Maps. *Computer Graphics Forum*, 35(3):461–470, June 2016. http://doi.wiley.com/10.1111/cgf.12922.

[Com] Composite Flow Maps. http://visdom.at/media/videos/mp4/composite_flow_maps.mp4. Last accessed 2021.03.07.

[Cor20] D. Cornel. *Interactive Visualization of Simulation Data for Geospatial Decision Support*. PhD thesis, TU Wien, 2020.

[Dip] Dipl.-Ing. Günter Humer GmbH. FFRM Riskmap. https://ffrm.hangwasser.at/. Last accessed 2022.09.05.

[Dir07] Directive 2007/60/EC of the European Parliament and of the Council of 23 October 2007 on the assessment and management of flood risks (Text with EEA relevance). https://eur-lex.europa.eu/eli/dir/2007/60/oj, October 2007. Last accessed 2021.03.07.

[dvA09] H. de Moel, J. van Alphen, and J. C. J. H. Aerts. Flood maps in Europe – methods, availability and use. *Natural Hazards and Earth System Sciences*, 9(2):289–301, March 2009. https://nhess.copernicus.org/articles/9/289/2009/.

[Ene] Energie Steiermark. Niederösterreich Atlas. https://atlas.noe.gv.at/webgisatlas/(S(pax0xifbgo1kieiosekl0zpz))/init.aspx?karte=atlas_hochwasser&cms=atlas_wasser. Last accessed 2022.09.05.

[Fed18] Federal Ministry of Agriculture, Regions and Tourism - Kommunikation und Service (Abteilung Präs. 5). Floods Directive (2007/60/EC). https://www.bmlrt.gv.at/english/water/eu-international-affairs/floods-directive-2007-60-ec.html, July 2018. Last accessed 2021.03.07.

[Gooa] Google Maps. https://developers.google.com/maps/documentation/javascript/overview. Last accessed 2021.03.13.

[Goob] Google-maps-react npm package. https://www.npmjs.com/package/google-maps-react. Last accessed 2021.03.13.

[HW09] M. Hagemeier-Klose and K. Wagner. Evaluation of flood hazard maps in print and web mapping services as information tools in flood risk communication. *Natural Hazards and Earth System Sciences*, 9(2):563–574, April 2009. https://nhess.copernicus.org/articles/9/563/2009/.

[Int14] Intergovernmental Panel on Climate Change. *Climate Change 2014 Impacts, Adaptation, and Vulnerability Part A: Global and Sectoral Aspects Working*

*Group II Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.* Cambridge University Press, New York, NY, 2014.

[KNK$^+$12] D. Komori, S. Nakamura, M. Kiguchi, A. Nishijima, D. Yamazaki, S. Suzuki, A. Kawasaki, K. Oki, and T. Oki. Characteristics of the 2011 chao phraya river flood in central thailand. *Hydrological Research Letters*, 6:41–46, 2012.

[KSD$^+$19] K. Krösl, H. Steinlechner, J. Donabauer, D. Cornel, and J. Waser. Master of disaster: Virtual-Reality Response Training in Disaster Management. *Proceedings of VRCAI 2019*, 2019. `https://www.vrvis.at/publications/PB-VRVis-2019-040`.

[Map] Mapbox. Mapbox gl js. `https://www.mapbox.com/mapbox-gljs`. Last accessed 2022.12.08.

[Mata] Material-UI. `https://material-ui.com/`. Last accessed 2021.03.13.

[Matb] Material UI. React Pagination component - Material UI. `https://mui.com/material-ui/react-pagination/#buttons`. Last accessed 2022.09.08.

[Matc] Material UI. React Table component - Material UI. `https://mui.com/material-ui/react-table/#custom-pagination-options`. Last accessed 2022.09.08.

[mdn] mdn web docs. Frame rate - Firefox Developer Tools | MDN. `https://developer.mozilla.org/en-US/docs/Glossary/FPS`. Last accessed 2022.09.08.

[Npm] Npm. `https://www.npmjs.com/`. Last accessed 2021.03.13.

[Ope] OpenLayers. OpenLayers. (https://openlayers.org/). Last accessed 2022.12.08.

[Pea03] L. Pearce. Disaster Management and Community Planning, and Public Participation: How to Achieve Sustainable Hazard Mitigation. *Natural Hazards*, 28(2/3):211–228, 2003. `http://link.springer.com/10.1023/A:1022917721797`.

[Reaa] React. `https://reactjs.org/`. Last accessed 2021.03.13.

[Reab] React Leaflet. `https://react-leaflet.js.org/`. Last accessed 2021.03.13.

[Reac] React Leaflet npm package. `https://www.npmjs.com/package/react-leaflet`. Last accessed 2021.03.13.

[Read] React Measure. `https://github.com/souporserious/react-measure`. Last accessed 2021.03.13.

[Red] Redux. https://redux.js.org/. Last accessed 2021.03.13.

[RH16] A. Reithofer and G. Humer. Flash Flood Risk Map Upper Austria - Evaluierung Des Schadensrisikos Durch Starkregenereignisse Anhand Eines Erweiterten 2D-Strömungsmodells. In *AGIT - Journal Für Angewandte Geoinformatik*, volume 2-2016 of *AGIT - Journal Für Angewandte Geoinformatik*, pages 406–411, 2016-01-01, 2016. http://gispoint.de/fileadmin/user_upload/paper_gis_open/AGIT_2016/537622055.pdf.

[Rot15] R. E. Roth. Interactivity and cartography: A contemporary perspective on user interface and user experience design from geospatial professionals. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 50(2):94–115, 2015.

[SMPF00] R. Sugumaran, J. Meyer, T. Prato, and C. Fulcher. Web-based decision support tool for floodplain management using high-resolution DEM. *Photogrammetric engineering and remote sensing*, 66(10):1261–1265, 2000.

[Sta] Stack Overflow. How to get the FPS in chrome devtools - Stack Overflow. https://stackoverflow.com/a/48081289. Last accessed 2022.09.08.

[Typ] Typescript. https://www.typescriptlang.org/. Last accessed 2021.03.13.

[Vis] Visdom Szenarien simulieren und visualisieren. http://visdom.at/media/pdf/visdom_flyer_dt.pdf. Last accessed 2021.03.07.

[Vla] Vladimir Agafonkin. Leaflet. https://github.com/Leaflet/Leaflet/blob/master/LICENSE. Last accessed 2021.03.13.

[VRV] VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH. FLOODVISOR. http://visdom.at/projects/floodvisor/. Last accessed 2021.03.07.

[W3C] W3C. Web Design and Applications - W3C. https://www.w3.org/standards/webdesign/. Last accessed 2022.09.08.

[Was11] J. Waser. *Visual Steering to Support Decision Making in Visdom.* PhD thesis, Technische Universität Wien, Vienna, May 2011. http://diglib.eg.org/handle/10.2312/8274.

[WFR$^+$10] J. Waser, R. Fuchs, H. Ribičič, B. Schindler, G. Blöschl, and E. Gröller. World Lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, November 2010. http://ieeexplore.ieee.org/document/5613487/.

[WKC18] Jürgen Waser, Artem Konev, and Daniel Cornel. On-the-fly Decision Support in Flood Management. *GIM International*, (November/December):22–25, 2018. Last accessed 2021.03.07.

[WRF$^{+}$11] J. Waser, H. Ribicic, R. Fuchs, C. Hirsch, B. Schindler, G. Bloschl, and M. Eduard Groller. Nodes on Ropes: A Comprehensive Data and Control Flow for Steering Ensemble Simulations. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1872–1881, December 2011. `http://ieeexplore.ieee.org/document/6064950/`.