ivi report Simon Luder

Inhaltsverzeichnis

LO1: Performance

Interactive visualizations help users to analyze and explore large amounts of data at a short amount of time by rendering it into an optical format. Visualizing data is therefore important, especially for data scientists but also for less technical affine people because it is easier to understand as raw data. Unfortunately, this interactivity quickly pushes one to performance limits, for example due to limited hardware. These limits must be stretched or, if possible, be avoided. In this chapter am going to explain some concepts and methods which can help to prevent walking in such a performance bottleneck.

Tiling

Tiling or tiled rendering describes a process where a computer graphic is separated into a grid/tiles and every tile is then rendered separately. If a user interacts with this graphic, only the part of the graphic that is in the user's field of view is visualized. A large visualization, which requires a lot of memory, is divided into many smaller parts, which can be loaded much faster due to their size. The advantage is that with this technique visualizations can get previously rendered and save computing power which allow the user to switch between the different levels of detail in real-time.

This is especially useful if you have a huge map where you want to zoom in and explore, just like with google maps. If you would try to render the whole world with the same resolution as you have at the highest zoom level, it would take a very, very long time. Instead google breaks down this map into more smaller tiles which are then placed next to each other. If you zoom into one of these tiles, the layers that lay below and around the actual tile can be pre-rendered, which allows a smooth transition between the tiles and different zoom levels. Depending on the location of the map, google maps has about 22 different zoom levels. At each level that is zoomed in the previous tile is divided into four new tiles. That means on the deepest zoom-level google maps would need 4^16 or around 4.3 trillion single tiles to map the whole world.

Level of Detail

"Level of detail" management is relatively self-explanatory. This term is also closely related to game development, where it describes how detailed the individual objects need to be represented. For example, objects that are in the background don't need to be displayed as precisely as those in the background, because you generally don't recognize these details.

If you transfer this concept to data visualization, then it is about how detailed a graphic has to be for what I would like to represent, especially with 3-dimensional vizualisations, which are rendered with a large number of polygons just like in video games. Of course, the more precise the level of detail, the more polygons must be generated, and the more computing power or time is needed.

This statement can be easily checked by using a simple python script to test out different visualization libraries and compare their performance with an increasing number of datapoints. I did this to compare the three libraries "plotly", "matplotlib" and "seaborn".

Although I did create this example to compare the performance of line plots, it can easily be adapted to 3 dimensional plots.

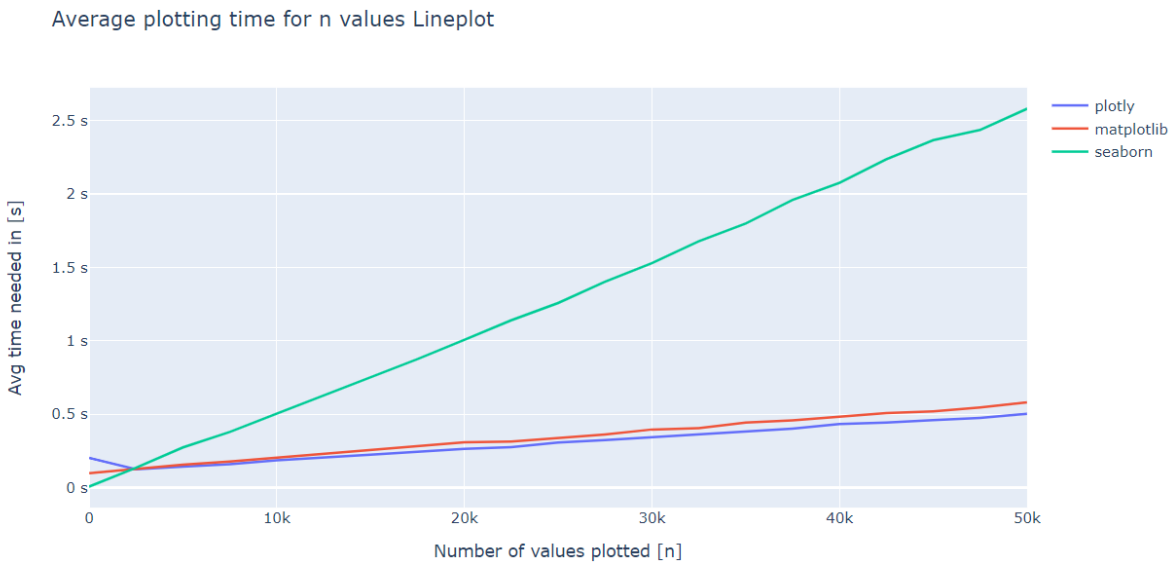

Figure 1: Visualizing time needed with increasing numbers of datapoints

You can clearly see that some libraries are faster than others, but all libraries have a linear relation between the number of datapoints to plot and the plotting time.

Blitting or Blit

Blit stands for block transfer. Blitting is a technique where a block of data is rapidly moved from one part of memory to another which speeds up the process of image rendering by multiple times. This technique is often used in the game industry for two-dimensional games but can also be applied to the visualization of animated plots and visualizations. Matplotlib has for example the function "FuncAnimation()" which allows to select blitting with an additional parameter.

In the notebook Blitting_example.ipynb you will find an example of blitting. There sin-waves get animated. The first time without and the second time with blitting.

In the first example, six sinewaves with different frequencies were generated. After that the matplotlib function "canvas.draw()" was used to create an animation of these frequencies with 100 frames. During that the time and average frames per second were measured and displayed at the end of the animation. The problem is that the "canvas.draw()" function redraws the whole visualization which means that not only the x and y values for the sinewave, but also the whole style and formatting needs to be rendered for every single frame.

The second example uses blitting to render the constant properties of the plot once (in this case the background and style) and then preserves this information. This means that only the sine waves need to be re-rendered for each frame, which makes our animation way faster. This small adjustment alone brings us in this example already more than a tenfold increase in speed.

CPU vs GPU rendering

CPU Central Processing Unit, GPU stands for Graphics Processing Unit. Traditionally all graphics in python are rendered via the CPU. But meanwhile GPU's are considered dramatically faster when it comes to rendering. While CPU's have less cores, these are much more powerful compared to a GPU core, they have much smaller numbers of them. While CPU's can create powerful and high-quality images, they are much slower than GPU's. Because despite the weaker cores GPU's have a much higher numbers of them which makes them ideal for rendering a huge amount of data. Depending on the model a modern GPU is considered to be around 5 to 20 times faster than a CPU makes a lot more efficient which also lowers the costs for computing power.

In python there are already multiple libraries for computing via GPU. I personally have no experience with GPU processing. But I think it is a technology that will be implemented more often in the future and therefore would surely like to try it out for myself.

LO2: Dashboard design principles

A dashboard is a tool that provides a centralized, interactive and visual means of monitoring and analyzing data from one or multiple sources. A good dashboard provides all relevant information, shows its data in a simple but meaningful graphical layout and has an intuitive structure which lets the viewer explore or switch between different points of view.

Wettermonitor

In my first semester there was the challenge "Wettermonitor für Wassersportler". Our task here was to download weather data provided by the Zurich city police via an API at regular intervals and visualize it on a dashboard in a way that is easy to understand and the target group, in our case sailors of a sailing club, can use this information to plan a sailing trip. In addition, we should also try to make a prediction about the further wind development and display it as well.

We have tried out diffent tools to visualize our data. At first we have used chronograf, a software designed by indluxdb for fast visualisation. The problem with chronograf was the limited possibility for personalisation and individual design of the dashboard. We also were not able to make predictions that way. Because of that we the then switched over to python and used the plotly / dash libraries for creating the final dashboard shown below.

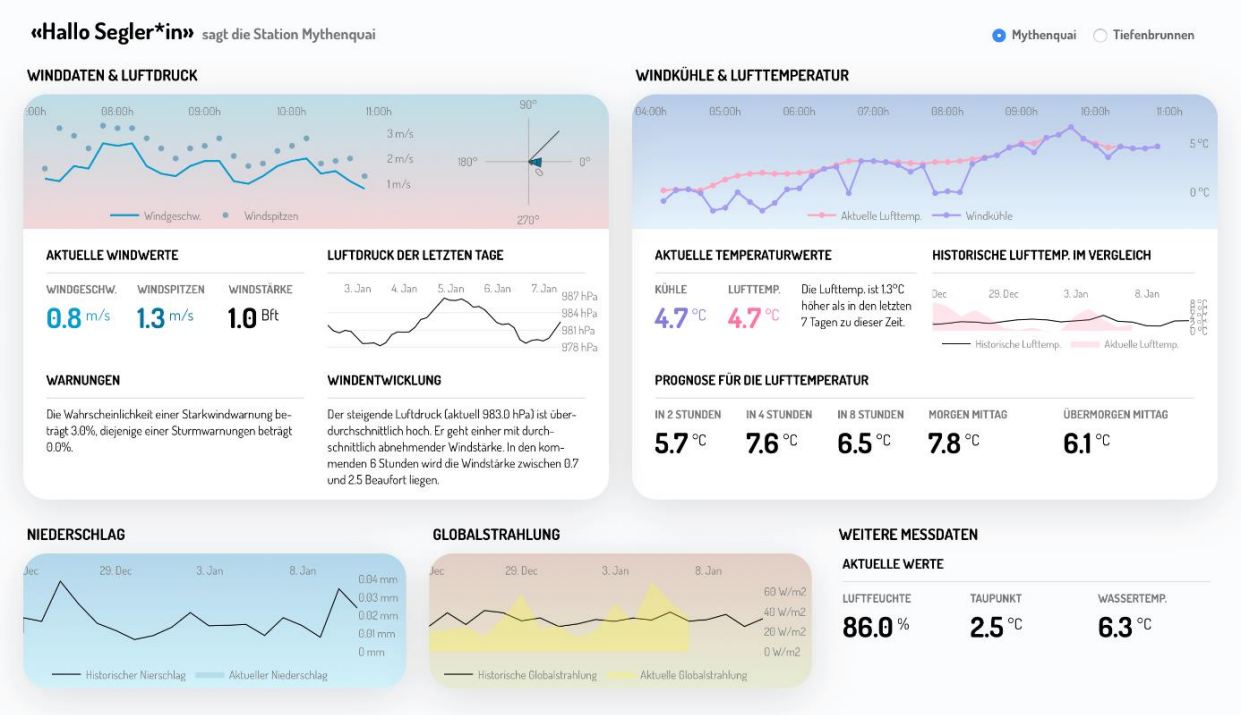

Figure 2: Dashboard from Wettermonitor challenge

According to our surveys, created in advance in the module "pko", the most important information for our target group is wind and its development. Therefore, this information is at the top of our dashboard and gets the most space in our layout. All other values are placed at the bottom. In that way the viewer should be led via Z-pattern trough the dashboard.

Shneiderman's mantra

A dashboard is about letting the user explore data visually. When it comes to displaying a lot of information, it can quickly turn into an information overload. Ben Shneiderman investigated this problem and has developed a guideline on how to provide information to the user, called "Shneiderman's mantra". The idea is to give the viewer an overview first, then zoom and filter and at last, details on demand. At our infoboard for sailors you get a rough overview of the general weather conditions for the current day. The measured values are visually divided into 5 different categories: Wind development and strength, air temperature, precipitation, solar radiation and other measurements. At the top right it is then possible to select the measuring station. You can choose between Mythenquai and Tiefenbrunnen. We built this dashboard with the intention to only give fast a rough overview. It is not designed for exploratory analysis. Therefore, it doesn't provide the possibility to zoom in or get more details from single plots. Technically because of the interactivity of dash and plotly, it would be possible to get more information by clicking on a datapoint within the graphics. In our case, because we wanted to create a simple information dashboard, we have deactivated this interactive feature.

Linking and Brushing

Brushing and linking describes a technique to compare attributes of one or more data points over several visualizations. Brushing means selecting a subset of datapoints with an input

device like a mouse. Thereby selected data points of one visualization are marked e.g. by an own color. It is also possible to hide not selected data. This marking of the selected points is then converted to all other visualizations, which allows a specific optical comparison. This makes it easier to explore only a small subset of data to see for example if there is a correlation between different attributes or to detect potential clusters.

In our dashboard, we mostly show different attributes as time series. One possibility to implement linking and brushing to our dashboard would be to link some of the time series to an interval selection, where you can adjust the time range you want to see. However, we did not implement this feature, because it was not defined in our tasks for the dashboard. But because I wanted to try out this technique myself, I created a short notebook, where parts of mathematical series can be selected by brush-linking.

Notebook: Brushing and Linking example.ipynb

Liked Filters

One way to create interactivity on your dashboard is to allow filtering via pre-defined parameters. You can achieve that by adding custom buttons, sliders or dropdown menus. This can be especially useful on dashboards, where you can quickly be limited in space and don't have room for many individual graphics. Because our dashboard has a lot of information shown at the same time, it can be difficult to find the information you are looking for. You can already filter the data of the weather monitor by station, although the choice with only two stations is relatively limited here. A next step could be to implement a feature, that allows to filter the plots you want by topic.

In the "Vacancy, no vacancy" project we didn't have a dashboard. But since a bit part of our analysis was focused on the four rotating elements (revolving wall, revolving cabinet and two revolving lamps), we added the possibility to filter by these elements via dropdown bar in some of our plots. Unfortunately, the GitHub repo is not public. However, for demonstration purpose, I have copied one of our visualizations that uses linked filters into my ivi repo:
https://github.com/SimonLuder/ivi_report_github/tree/main/Vacancy%20Example

Focus + Context

The Focus + context visualization is a technique used for space-saving and clear display of data. These are particularly common in visualizations which are based on a large amount of information like geographical or graph data e.g. Network data. Another use case would be the use in mobile applications because this technique is also suitable to fit the constraints of small screens and interaction with fingers. It combines focus-displaying, where a small part of the layout is shown enlarged and more detailed while the context-displaying shows the whole picture in lower resolution to provide an overview. The position of the focus is determined by the point-of-interest which is set by the user itself. In addition to the purely optical magnification/enhancement, it is also possible to display additional information in the focus area.

LO3: HCI Basics

Human-Computer Interaction (HCI) is the study of how computer-technology influences human behavior and activities. Examples range from obvious technologies like smartphones or digital dashboards, but also simpler ones like traffic lights. The core concept if HCI based on psychology, computing and ergonomics which are combined to create several acknowledged principles which focus on how to design computer-technology to make it as pleasant and intuitive to use as possible. Or as in ISO 9241–11 it is defined as: “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” In this chapter I will describe some of the fundamental principles which are part of the HCI basics regarding the implementation and design of interactive visualizations and virtual dashboards.

Fitts’s law

Fitts’s law is one of the fundamental concepts for creating an interactive product like a website, an app or in our case a dashboard. It was created by Paul Fitts and describes how quickly people can select a target. The law says that the time required do acquire a target depends on the size and the distance to the target, whereby large targets that are close to interact have the lowest interaction costs and small targets that are far away have the highest interaction costs.

So, said we have a scatterplot with a lot of single small dots representing a datapoint and you want to interact them. Here it would be impractical if it would be necessary hover directly over the selected point. To keep the interaction costs low, a possibility would be to implement that always the nearest datapoint is selected by using the Voronoi method. Plotly itself already delivers such a functionality for its interactive visualizations. It lets you choose between three possible parameters (closest point on x-axis, y-axis and Euclidean distance). The graphic below shows such a scatter plot using the Voronoi method for selecting the nearest point.

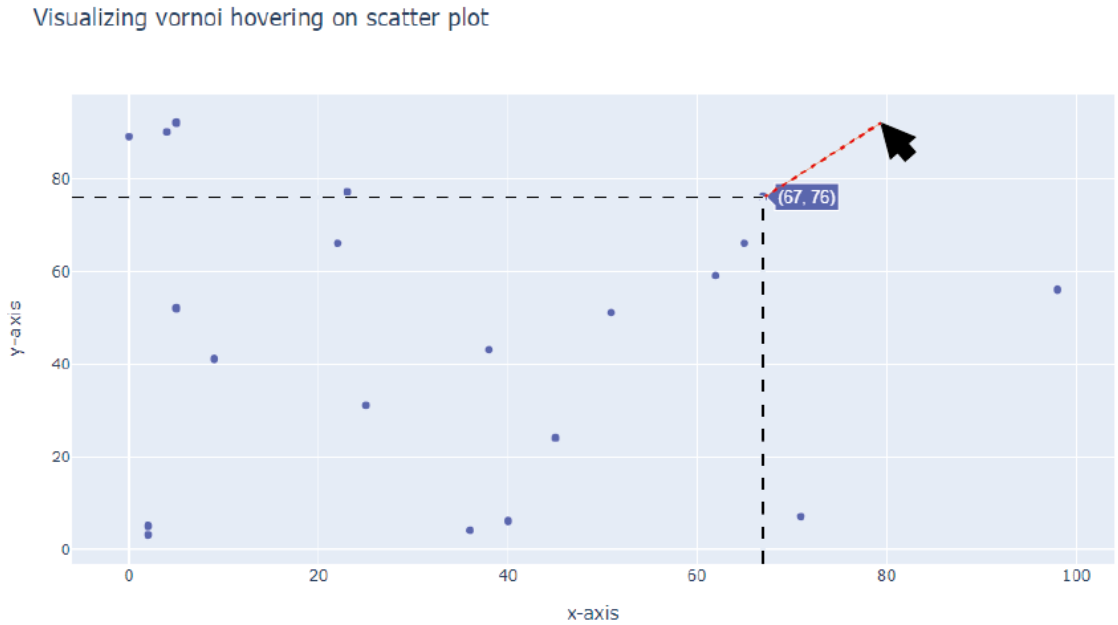

Figure 3 vornio hovering from HCI examples.ipynb

Millers law

Millers law focuses in limits of our capacity for information processing. It describes the fact, that most humans are only able to remember 7 +/- 2 bits of information when conducting a task that requires cognitive abilities. Which means the number of stimuli our brain can process at a time is limited. Transferred to data visualization it therefore often means less is more. Interactive visualizations and animated graphs might look fancy, but if it does not generate more insight than just a simple graph it is basically just visual garbage that distracts the user's attention and possible prevent him from understanding the key message of the visualization. Put it this way: simpler is often better. That also applies to dashboard design.

Webbers law

Webber's law describes the just noticeable difference a stimulus has to change in order to be noticed. This law applies to all stimuli while the minimal threshold varies between different mediums. information can be displayed in various forms like text, maps, graphs, etc. Whereby clear optical separation between individual elements is necessary. A poorly implemented design lacks clearly differentiating between the different stimuli.

The graphic below shows an example of Webber's law. The two graphics on the left (10 points and 100 points) show a scatterplot with the corresponding number of randomly generated points in a range between 0 and 100 for both x and y axis. The graphics on the right show the same visualization with 10 more points added. While the change between 10 points and 20 points is relatively easy observable, the change between 100 and 110 points is more difficult to see. That is because according to Webber's law smaller proportional changes get more difficult to spot.

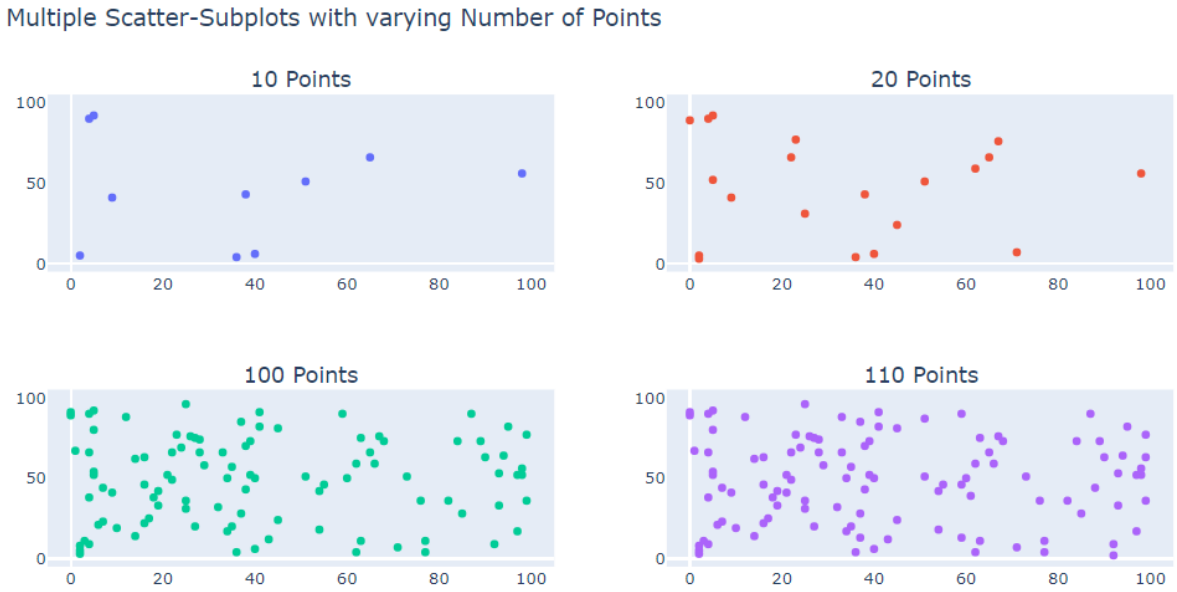

Figure 4 from HCI examples.ipynb

Change Blindness

Change blindness describes the phenomenon that people tend to have problems recognizing even bigger changes when there is an interruption between the changes. A well-known example would be by showing a slightly changed image two times, but with some flickering in-between. The same applies when comparing data visualizations. If we remember our example plot from Webber's law and take only the part where change is clearly visible, it is relatively easy to compare them side by side.

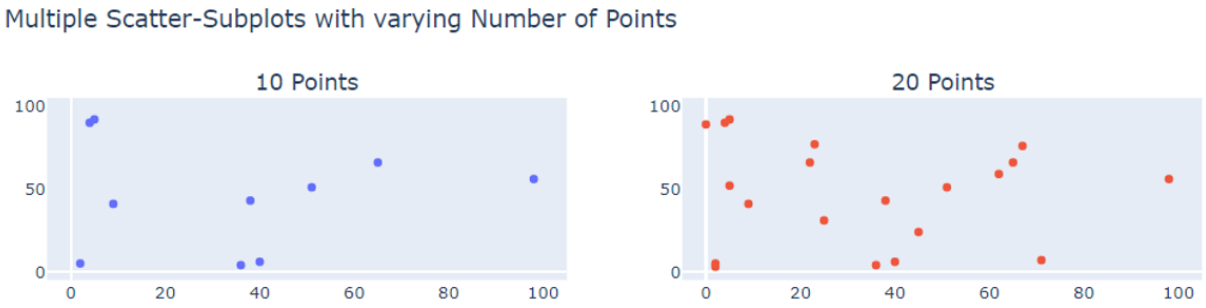

Figure 5 from HCI examples.ipynb

I want you to now cover the "20 Points" chart and only look at the "10 Points" chart. After a while it is time to switch. Cover the "10 Points" chart and try to tell what points on the "20 Points" chart are new. You can see that it is now much more difficult to tell the difference as when having them side by side. This is a very important point to notice, when showing differences or change, it is important to give the viewer the possibility for side by side comparison.

More HCI components to explore

- Hick's law – Examines the relationship between the amount of stimulus given and reaction time.
- Pareto principle – Also called 80/20 rule and describes distribution of expected outcomes in relation to cause.
- Gestalt principle - (Described in gdv report LO2)

LO4: Evaluation

This chapter focusses on the evaluation of interactive visualizations.

In my gdv report, I have already covered some aspects of usability testing. The test product there was the dashboard from the Wettermonitor challenge. The target was to create a usability testing protocol which aims to help us understand the effectivity of our visualizations. The goal now is to create a usability test, which focusses on the evaluation of efficiency, effectiveness and satisfaction while interacting with the product. However, because the monitor does not offer a lot of interactive visualizations, I am going to conduct the usability test on an external website which was created by a company called “The Visual Agency”. It shows the largest existing collection of writings, drawings and other protocols from Leonardo Da Vinci. The link to the website is here: https://codex-atlanticus.it/#/

Setup

The structural design of the website is based on various visual and interactive guidelines. It first offers an overview of the entirety of the individual documents, allows the documents to be filtered according to character traits through various filters and allows individual pieces to be viewed in detail. The interactive elements of this website are:

Overview page:

- Preview of single pages via hovering
- Show pages:
    - “All” or “active pages”.
    - Numerically or chronologically ordered.
- Filter by:
    - Subject & Topic.
    - Page numbers via interval selection.
    - Years via interval selection.

Detailed view page:

- “Recto” (front) and “Verso” (back) vision of the selected page.
- Recommended pages with similar subjects.

So, a simple task on this website might be: Select a page which is created before the year 1500 about human sciences that contains jokes.

I want to conduct the usability test in a moderated format. To better observe the task execution the meeting will be in-person. The size of the test group was set to a typical number of five people. The participants will conduct the test on a Windows laptop with QHD resolution of 2560x1440 pixels. The questions are a mix of exploration and assessment. For quality measures we record task completeness, task accuracy and time needed for each task as well as the ten questions from the “System Usability Scale” with a rating scale of 1 to 5. The measured data is written down in the test protocol. Other possibilities to measure the efficiency would be to use mouse tracking to measure how clear and direct the user’s

movements are. Because the overview page offers a lot of information at once, eye tracking would also be an interesting measure to get information about the user's attention and how much Hicks law (more stimuli = slower reaction time) comes to play.

Ideally to get accurate and significant data, the test participants should be part of our target group. I want to test how well people which visit the website for the first time find their way around. Also, the test participants should be between 18 and 55 years old and have an average computer skill level.

The usability test protocol can be found here:
https://github.com/SimonLuder/ivi_report_github/tree/main/Usability_test

Results

According to the usability test results the participants found on average 6 out of the defined nine interactive features within the first two minutes. Sometimes the test participants had some difficulties finding their way around at the beginning and became more and more confident towards the end of the test. The feature that was found most on average is the page preview. The feature that was found at least is the turn page button, this might also be because they are labeled with "recto" and "verso" which was not expressive for the test users.

Overall, the website offers more interactivity than the users expected at first. Therefore the "how to read" manual with animated information is definitely useful and covers all important information needed to navigate through the webpage.

There might be a small bug in the reset Button. Sometimes the participants saw only 1118 out of 1119 documents in total after clicking the reset button which means there is one page missing. After refreshing the website, they saw the missing document was there again.

Also, when opening the website or applying filters, the animation flow lags sometimes even with a more modern laptop.

The collected data could now be analyzed further and used to improve the usability even more. But overall, I think the website is a great example of how to design an interactive visualization that engages people to do own researches.

External Projects

Github ivi repository: https://github.com/SimonLuder/ivi_report_github

Wettermonitor: https://github.com/fabianjordi/wettermonitor-fuer-wassersportler

Interesting Weblinks

LO1

- why is plotting with Matplotlib so slow? https://stackoverflow.com/questions/8955869/why-is-plotting-with-matplotlib-so-slow

LO2

- Interactive dashboard from plotly example gallery: https://dash-gallery.plotly.host/dash-oil-and-gas/

LO3

- Interaction Design for Data Visualization" talk Sources by Mike McCroklin: https://medium.com/@milr0c/interaction-design-for-data-visualization-talk-sources-f8ccec2ca1a7
- Ranking Visualizations of Correlation Using Weber's Law by Lane Harrison: https://www.cs.tufts.edu/~remco/publications/2014/InfoVis2014-JND.pdf
- Using a d3 voronoi grid to improve a chart's interactive experience: https://www.visualcinnamon.com/2015/07/voronoi.html

LO4

- How to Measure Product Usability with the System Usability Scale (SUS) Score: https://uxplanet.org/how-to-measure-product-usability-with-the-system-usability-scale-sus-score-69f3875b858f