Interactive Visualizations (IVI) Bericht

Luca Gisler

HS2023

Repository:

Data: https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php

Contents

[2. LO1: Performance 2](#_Toc154135115)

[WebGL 2](#_Toc154135116)

[Tiles 2](#_Toc154135117)

# LO1: Performance

Performance in interactive visualizations is vital in the big data era to ensure a positive user experience, provide real-time data analysis, handle complex data effectively, ensure accessibility and scalability, facilitate quick insight discovery, and maintain a competitive edge. In this LO it will be looked at how the performance of Visualizations behaves.

## WebGL

In the context of performance visualizations, WebGL's functionality as a rasterization engine is crucial. It allows for efficient rendering of complex visuals like points, lines, and triangles, essential for detailed and dynamic graphical representations. The use of GPU for processing and the requirement for specific code through vertex and fragment shaders, written in GLSL, enable high-performance visualizations. These shaders play a key role: the vertex shader computes vertex positions for rendering primitives, and the fragment shader assigns colors to each pixel, directly influencing the visual output's quality and performance. Since WebGL’s API primarily focuses on setting up states for these shaders, it's integral in optimizing the performance of visualizations. Efficient use of WebGL in performance visualizations hinges on how well one can manage and execute these shader functions and state setups, ensuring that the necessary data is accessible and processed effectively by the GPU (*WebGL Fundamentals*, o. J.).

The benefit of the GPU over the CPU is that is specifically made for Visual tasks. It contains thousands of kernels, processes are running parallel to each other, more efficiently in in processing lots of small tasks(*CPU vs. GPU*, 2022).

To investigate the performance differences, I created a visualization that shows 500000 random points in a 2D coordinate system. Points in a 2D coordinate system. Once the visualization was rendered with the SVG mode and once with WebGL.

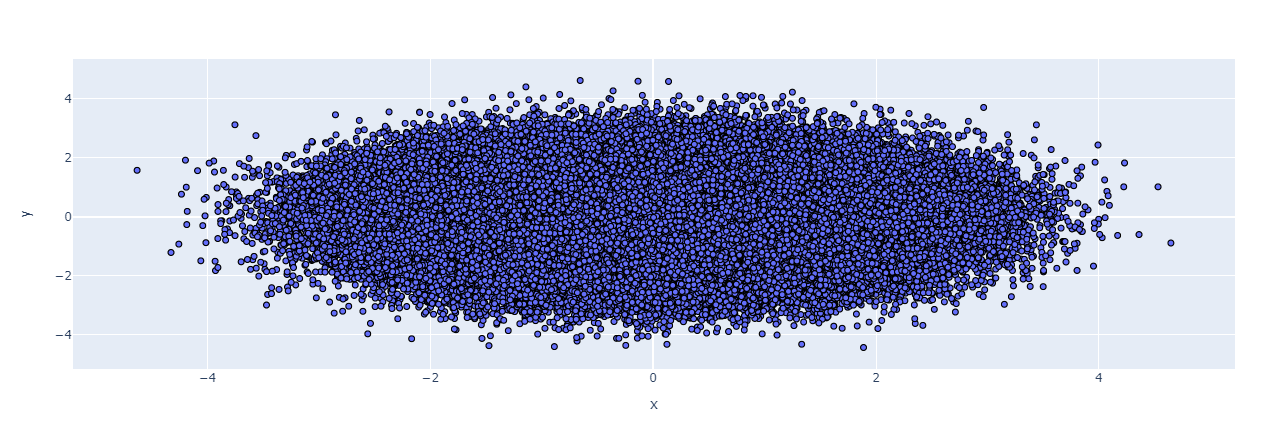


Figure 1: Plot WebGL

The plot created with WebGL was shown much faster than the one created with SVG. The runtime difference was not that huge. For the WebGL [1.31s] and for SVG [1.54s]. The plot created with WebGL was much smother to handle than the SVG. Handling the SVG plot let VScode to crash or stop working. This shows that the WebGL plot is much more resource efficient than SVG.

Ein Bild, das Screenshot, Majorelle Blue, Electric Blue (Farbe) enthält.

Automatisch generierte Beschreibung

Figure 2: Plot SVG

In addition the look of the SVG plot is different from the WebGL. The SVG points are much more detailed and have very clear borderlines.

## Tiling

Tiling can improve the performance of interactive visualization of large amounts of data by dividing the data into smaller dividing the data into smaller tiles and loading and displaying only the required parts. This allows the computing power to be more effectively and reduces the load on the computer. The interactive nature of Tiling allows users to users to navigate quickly and easily between different sections of the data, enabling a faster and more and more responsive experience (Forrest, 2023).

Datashader is a graphical pipeline system for the fast and flexible creation of meaningful visualizations. creation of meaningful visualizations of large datasets. Datashader divides the creation of images into a series of explicit steps that allow calculations to be performed on intermediate representations.(Yang, 2022)

Ein Bild, das Screenshot, Farbigkeit enthält.

Automatisch generierte Beschreibung

Figure 3: Datashader plot

In this example, the plot contains 1.5 million datapoints. The handling of the plot runs very smoothly and without any delays.

A slightly different approach is taken from the framework Deck.gl. JavaScript, typically constrained by its single-threaded event loop model, struggles with tasks that require heavy computation, such as managing large datasets and rendering 3D graphics. To address this limitation, deck.gl employs the WebGL library, which enables asynchronous access to the user's computer GPU. This approach shifts the computationally intensive tasks away from the browser to the GPU, which is more capable of handling such demands. Consequently, deck.gl is able to produce impressive visualizations, efficiently managing and rendering millions of data points with enhanced speed and performance (Muramoto, 2019).

# LO2: Dashboard design principles

Dashboards, as visual tools, use charts, graphs, and various elements to present data effectively. A well-designed dashboard focuses on maximizing visual representation while efficiently managing textual content. This design strategy is crucial because humans can process visual information much faster than text, up to sixty thousand times quicker. Consequently, the layout and design of a dashboard are vital in user experience design, significantly influencing the success of applications and websites.

Good dashboard design is not just about aesthetics; it plays a key role in optimizing the customer journey. By adhering to key design principles, it ensures that users spend more time engaging with the website and effectively extract insights from the data presented. These principles, which primarily focus on the strategic placement and design of dashboard elements, function more as guidelines rather than strict rules. They are not fixed but are crucial in creating an intuitive and informative dashboard that enhances user interaction and data understanding (Mokkup.ai, 2023).

## Schneiderman’s mantra

“Schniederman’s Mantra is an extremely influential organizing principle for the creation of  visualization systems. It goes as follows» (HAMPDATAVISUALIZATION, 2016):

**Overview first**:

**Zoom and Filter:**

**Details on Demand:**