Visliu :

Responsive Visualization of Points in Space: Sampling, Clustering, Partitioning

Table of Contents

4. Challenges
5. Applications

Evaluation

Future Work

Bibliography

.

Interactive visualization

“I tell you and you forget. I show you and you remember. I involve you and you understand.”
Confucius, 500 BC

“A picture is worth a thousand words. An interface is worth a thousand pictures.”
Ben Shneiderman, 2003

1. Introduction

Humans have a total of 5 senses through which they interact with the world : sound, sight, touch, smell and taste. Of these, smell and taste are pretty limited in the sense that we have not yet found any efficient way to communicate using them. Touch is limited by the requirement of tangible objects. Sound and Sight are known to be most advanced and are a major part of most of our daily activities. Through language we have even connected them to activate the same areas of the brain using subvocalization. But these two are still quite different. Sight has multiple orders of magnitude higher ability to process signals than sound and unsurprisingly has a larger portion of brain dedicated to it than any other single task.

We like to define Interactive Visualization as the field of study which aims to use sight as a more efficient means for interacting with and understanding the world around us. A key feature of a good interactive visualization system is that it tries to use cognitive intuition to convey information.

In this paper we survey some of the major developments in interactive visualization till now and then combine some of these ideas to build a web system for interactive visualization of points in space. We also use some ideas of our own that enable a responsive interface even with sufficiently large dataset sizes and unreliable networks.

2. Present state of things

Fuelled by continuous developments in hardware technologies and an exponential increase in data collected, Interactive Visualization technologies have developed a great deal in the past few decades. Although a lot of effort and time has been spent on developing better visualization technologies over the past 2 decades, there is still a long way to go.

Here we are going to look at few of the technologies and systems that have recently been developed or come into the spotlight for Interactive Visualization on the web.

Base technologies for rendering on the browser:

a. Canvas (HTML5) : Pretty recent; Natively supported by most browsers, but limited by number of elements that can be rendered at a time.
b. SVG : More than a decade old; Natively supported by most browsers but limited by number of elements that can be rendered at a time. This limit is lower than that of HTML5.
c. WebGL : Most recent; Variable and limited support from different browsers. Brings the power of the client GPU to the browser. Limited by the number of devices having a GPU and no standard browser support. Chrome and Firefox have a whitelisted list of GPUs that they allow webgl to run on.

Libraries/Services for Interactive visualization on the web:

a. Data-Driven Documents (D3) (2011):
    i. Javascript library for DOM manipulation on client side.
    ii. Uses SVG at the core.
    iii. Requires learning HTML, javascript, D3's API and SVG.
    iv. Biggest contribution are the tools that make the connection between data and graphics easy.

b. Google Charts (2010):
    i. Google's own visualization tool that is nicely integrated with BigTable and other Google cloud services.
    ii. Lets users easily create a chart from some data and embed it in a webpage.

c. Highcharts (2009):
    i. Javascript library for visualizations and animations
    ii. Offers an easy way to create charts and embed them in your web page.
    iii. Based on HTML5.

d. Three.js (2010 , stable :2015):
    i. Javascript library for 3D visualizations using webgl
    ii. Abstracts the presently messy interface to webgl
    iii. When it works(browser and client need to support webgl), provides abilities to render and interact with a large number of objects at the same time.
    iv. User experience varies depending on the GPU.

e. Bokeh (2013, still in beta):
    i. Python Interactive visualization library that targets presentation on browser.
    ii. Coding is totally in python
    iii. Is trying to get graphics in the style of D3.js but uses canvas instead of SGV, so can render more objects at a time.

f. Datawrapper (2012):

i. Javascript library for creating interactive embeddable charts.
ii. Visualization is created on the server but it provides a set of interactive abilities.
iii. Which run on client side through its library.

g. Flot (2007):
   i. A plotting library for jQuery that uses the HTML5 canvas.
   ii. Simple and good-looking interactive features, but very limited in functionality.

3. Challenges

With widespread interest in visualization at present, there are tons of new systems and libraries for visualization but all of them face these challenges and most of them refuse or fail to address them properly.

a. Data size: With web visualization, this is the biggest bottleneck right now. The browser even with native canvas support cannot render or manipulate even medium sized datasets. Some systems overcome this by doing aggregation on the server and updating client with a static image. Others need you to take care of your aggregation yourself before rendering your charts. These ways are either hacky or put extra work on the user that should be the responsibility of the visualization system.

b. Network: For libraries that work with a remote server, all data transfer happens in a go and slow or unreliable network leads to bad user-experience.

c. Computation power: Most libraries are client side and are limited by the computation capability of the client in providing responsive interaction on bigger than trivial data sizes or provide a very limited set of operations.

Pad and children

1. Multiscale Zooming Interfaces:

The ideas behind multiscale zooming interfaces can he traced back to Sutherland's Sketchpad system developed by him in 1963 but it was not until much later that systems started using zooming as a primary feature of interactive visualization. "Pad" ,developed by Ken Perlin and David Fox at NYU was a big milestone in realizing the possibilities of zooming interfaces. Pad was followed by multiple systems that took forward the ideas of semantic zooming and filter lenses to build next level systems that have since evolved to become the conventions for data visualization and analysis in their respective fields.

Let us first go over what we mean by zooming in the context of visualizations and then we will briefly look at few of the applications that came out from it.

Let us consider a bunch of 2D objects (marks, icons, etc) displayed on the screen of a monitor. Now there is no actual depth in this setup but the effect of enlarging or shrinking the size of objects with a corresponding increase in inter-object distances on pressing a button or rolling a wheel gives us the illusion of depth. It works very well and uniformly for everyone because it has a direct correspondence with what we experience in daily life. Because of how our vision works, we move closer to an object to see it in more detail and move away to take in the whole view. Since moving closer enlarges an object in our view and moving away shrinks it, on a stationary 2D display, just the act of enlarging or shrinking something is perceived in the same way by the brain. Just the enlarging and shrinking of an object without any changes in its representation is called geometric zooming. Pad introduced semantic zooming which more closely resembles the actual world. With semantic zooming, each object has set magnifications at which its representation changes. When you zoom in, more details are shown while zooming out hides the least important details. A smooth transition or movement of objects on the screen while zooming or panning is very important here as that eases the task of keeping track of objects of interest between two representations.

2. Applications :

In this section we will go over a few significant areas where the concepts introduced by Pad were initially applied and techniques that came out to complement/as a result of the Pad.

a. **Macroscope**: (Henry Lieberman, Powers of Ten Thousand) A nice technique that complements zoom and pan by keeping the context infront of the user at all times. Henry points out that multiple zooming and panning operations might cause the viewer to lose the context from his memory. Macroscope consists of combining multiple layers of information on a single display, using translucency,

focus and other image processing techniques to visually combine layers while retaining the integrity of the individual components.

b. **Portal Lenses in Pad++**:
   //description with examples of usage in present day software

c. **Network Collaboration** :
   //How do they work; benefits over traditional collaborations

d. **Device user-interface**(Benjamin B. Bederson, Larry Stead, James D. Hollan): //Pad++ as an alternate windowing system; benefits over traditional windowing systems; examples of usage in modern OSs

e. **Web-browser** (Benjamin B. Bederson, James D. Hollan): //Pad++ as an alternate web browser; benefits including easily surfing through session history and moving back and forth; examples of usage in Firefox

f. **Interactive Legends**: (M. Eduard Tudoreanu and Delbert Hart)

g. **Multiscale Editing**: (Muse: A Multiscale Editor, George W. Furnas, Xiaolong Zhang) //Also add examples of present day software such as Prezi , etc.

h. **Techniques for Time-Oriented data**(Ragnar Bade and Stefan Schlechtweg, Silvia Miksch): //Visualizing scales, high-frequency data, interacting with time; Example usage in Pulmonary Embolism Care

VISLIU

“Visual Information-Seeking Mantra :Overview first, zoom and filter, then details-on-demand.”
[Shneiderman, 1996]

1. Motivation

After extensive survey of web visualization systems out there, we found that there are a lot of things in which they fall short. We have created a system which uses a server-client architecture to enable responsive visualization of points in space for the client using a combination of clustering, partitioning and sampling along with the learnings from Multiscale Zoomable Interfaces to produce a nice user-experience that lets users analyze and perform basic operations on their data.

2. User-view

The system consists of a single page. The user first chooses a data file from the drop down menu or uploads his own file. The input file consists of 2 tab separated floating points per line. On clicking “Load Data” button, the top representation of the points gets displayed. There is a slider on the top right corner to switch between ‘pan and select modes (both are done by dragging the mouse across a screen area). ‘Select’ mode allows you to select one or multiple points which can then be operated on using the options on the left. How to use zoom and pan for data exploration, perform operations and change visualization configuration will be explained next:

a. Zoom and Pan

Much of our effort has gone into making zooming and panning work in a way that is intuitive for the user.

When a data-set it loaded by the user onto the canvas, the server calculates a top-view of the file by clustering points in areas with point-density greater than a threshold density. The client displays clusters as larger spheres on the canvas. The user can use scrolling on a mouse or touchpad to zoom in and out of the view at whichever point they desire. There is an option of zoom sensitivity on the left side of the canvas which lets user select/change the rate at which the view zooms in or out. This is pretty helpful as a user might want to zoom in pretty slowly for dense data-sets or when he is trying to find patterns in the data-set and fast when he knows the cluster/area of the canvas he wants to zoom in to. The zoom sensitivity varies from 10% to 100% which indicates the percentage by which the distances between points scale up or down with a single step of mouse wheel/touchpad.

A user can click and drag the mouse over the canvas to do panning in a zoomed state. Another feature that helps the user to keep track of where they are with respect to the bigger picture is the grey colored dynamic axis at the top and left side of the canvas. This is super-imposed on top of a static axis which ranges from the respective

minimum and maximum values of x and y in the dataset. The grey shading shrinks and enlarges with the zoom level and moves left/right or top/bottom depending on the viewpoint in the zoomed level. We have found this quite helpful while exploring and operating on data-sets.

b. Operations

The system has multiple operations that can be performed on selected subsets of data. These are: translation, recoloring, point deletion and point addition. The first step to performing an operation is to switch to Select mode by click on the Pan/Select switch on the top right of the page. Once it is in select mode, a user can use click and drag on the canvas to select an arbitrary subset of points. A translucent blue area is shown during mouse drag to indicate the points that are going to get selected if user leaves the mouse button at that time. Multiple click and drag add to the selected point set. As soon as a subset is selected, the operations listed below the Load Data button which were disabled before get activated and can be used:

1. Translate : Consists of two fields, x and y. The selected points get shifted in the present view by these values on clicking the Translate button.
2. Change Color: Click on the change color textbox displays a full color pallet. Clicking anywhere on the pallet changes the color of the selected points to that.
3. Delete Points: Removes the selected points from the dataset.
4. Add Points: Adds these points to the visualization.

c. Configuration

Due to the variety to possible point distributions in 2D space, there is no single optimal configuration to display them all. Also, there are user-preferences on point sizes, cluster sizes, etc. To address these, we have made some global server configurations available to the user to view/adjust according to his needs/preference. Here is the list of modifiable configurations and how they change the visualization:

//Trim and put server prop object properties here

3. Implementation

a. Technology Stack

i. **javascript** :

Javascript is the language of the web. Though it has been ridden with problems and controversies for the major part of the time it has been around, in the last few years it has gained a pretty big following. At present, with the improvement of javascript interpreters, it runs pretty fast as well as support is available in the form of packages and libraries which provide one functionality or the other.

We use javascript both for the front-end as well as the back-end. This provides us the ability to use the same code for data operations on both the client and the server for automatic load distribution.

ii. **canvas**:

The canvas element is a part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It updates a built-in bitmap and does not have a built-in scene graph as SVG does. Therefore, Canvas is considered as a lower level API on which something like SVG can be built. This also makes Canvas more powerful in terms of number of objects it can create and manipulate simultaneously compared with SVG.

We use Canvas for our system.

iii. **nodejs**:

Node.js is an open source, cross-platform runtime environment for server-side and networking applications. It provides an event-driven architecture and a non-blocking I/O API that optimizes an application's throughput and scalability. Node.js uses the Google V8 JavaScript engine to execute code, and a large percentage of the basic modules are written in JavaScript.

We use nodejs for two reasons. Firstly it lets us use javascript on the server-side, secondly, the event-driven architecture and the non-blocking I/O API suits our systems quite well.

iv. **Processing.js:**

Processing is a programming language designed to write visualizations, images, and interactive content. It has been used for a long time by researchers and practitioners alike to create interactive visualizations for years. Processing.js is a port of this language to the web. It basically uses canvas and javascript in the background, but provides an API identical to the original Processing language.

We felt this API to be preferable for designing and manipulating our visualization. We do not however use the event handling API of this library as that was not performing as expected in our experiments.

v. **socket.io:**

This is a javascript framework that enables real-time bidirectional event-based communication. It uses websockets internally which are radically more efficient than AJAX for client server communication. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.

Socket.io contributes in a major way in making our system functional by making constant server-client communication faster and easy to implement and manage.

vi. **Other technologies**

We use Bootstrap for giving better look to buttons and text fields. Jade templating language is used to create the dynamic webpage on the backend. Heroku's cloud hosting service is used to host the app for public access.

b. Zoom and Pan

c. Operations

d. View updates

e. Data processing

f. Server configuration

4. Challenges

5. Applications

Evaluation

Future Work