Chronographer - Location History Visualization

Scott Todd*

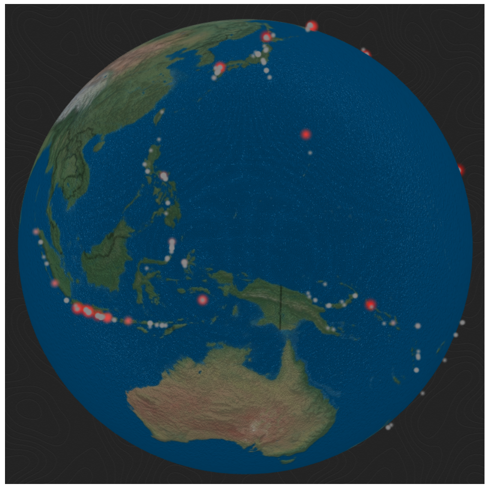

Figure 1: *Visualizing volcanic eruptions.*

Abstract

I created a geographic location history visualization tool titled Chronographer that explores virtual reality interaction as a data analysis aid. I used three.js [thr], the device orientation API [Moz], and the Google Cardboard [Goo a] viewer to construct this visualization, which is hosted at `http://scotttodd.github.io/Chronographer/`. I visualized location history data exported from my Google account using Google Takeout [Goo b] as well as volcanic eruption data from NOAA [NOA]. I went through several design iterations and concluded that this style of visualization is interesting but not currently effective as a tool for learning from geographic data sets. I conducted informal user testing partway through the project but did not perform a rigorous user study.

Keywords: Data Visualization, Geographic Data, Virtual Reality

1 Introduction

Geographic location history data sets combine spatial information which demands context with chronological information which benefits greatly from interaction or animation. Existing map-based tools present geographic data in an understandable format but they each rely on a projection that introduces an extra level of processing in order to relate the data to the physical world. Virtual reality has been growing in popularity and technical maturity in recent years and offers the opportunity to connect people to data in ways not possible before. I wanted to test the utility of virtual reality in helping people understand these data sets.

The representation of the time axis is also important for data analysis. Several techniques have been researched in the past which handle user control over the time axis differently.

Virtual reality devices such as the Oculus Rift and Sony's Project Morpheus are currently in active development. Screen, positional tracking, and rendering technologies have advanced to the point where immersive virtual reality is nearly within reach for consumer-grade projects. The Google Cardboard takes this a step further and leverages current generation smart phones, which are packed with sensors and high resolution screens already, to deliver an affordable virtual reality solution that hobbyists and consumers can easily use.

Virtual reality for web browsers is a recent extension of these new technologies, so support and stability are both limited. The device orientation API is experimental and has several shortcomings, but it may eventually be replaced with official browser support through a technology currently being named WebVR.

1.1 Related Work

1.2 Technical References

"What is Spatial History" by Richard White [White 2010] details efforts to chronicle history through data visualization, making sense of large data sets through visual media, rather than through words alone as traditional history textbooks do. He argues that spatial relations are established through movement - movement of people, goods, and information. He discusses "relational space", where locations are closer at different points during a day due to traffic and other relative, modern factors. He also concludes that visualizations are a means of doing research and generating questions that may have otherwise gone unasked, not a method to communicate findings discovered through other means.

The DimpVis system [Kondo and Collins 2014] explores a novel method of interacting with data points on two-dimensional plots

*email: todds@rpi.edu
Project Repository: https://github.com/ScottTodd/Chronographer

with time as a third axis. Where time is normally controlled by a slider, it allows for users to drag points along hint paths to move through time in either direction.

1.3 Further Inspiration

GitHub user @theopolisme created a website that lets you visualize your Google Location History using an interactive heatmap [@theopolisme]. In his visualization, data points from all time values are compressed onto a single map which features zoom and pan controls.

This blog post [Wheatley] argues that good data visualization tools should consider human visual perception and give insight into the unknown, not the already-understood. It explains how DARPA used the Oculus Rift to visualize three-dimensional network simulations and how engineers at Caltech used virtual worlds such as Second Life and the Unity3D game engine as data visualization platforms.

2 Development

2.1 Data

The data returned through Google Takeout is in JSON form with a simple array of [latitude, longitude, timestamp] triples.

2.2 Initial Ideas

I originally planned to build a two dimensional map interface that would blend a set of data points across time ranges together. At this point, I was inspired by @theopolisme's Location History Visualizer and had recently completed another map-based visualization using d3.js [D3.]. Because of my familiarity with d3.js and three.js, I hoped to combine the powerful WebGL rendering of three.js with the convenient map projection and rendering of d3.js to visualize my own location history. I had also been searching for a reason to experiment with the Google Cardboard, so I worked that into my plans.

My basic visual design centered around displaying data points above a map and using additive or alpha blending to show general regions where points were focused. As the visualization time changed, data points would fade in and out. I used a traditional slider to control the visualization time, which could be set to play automatically or could be manually controlled via mouse or touch. This initial design can be seen in Figure 2.

I used a particle system to represent data points, with each data point having a single particle in the particle system to represent it. These particles would change their display properties based on the difference between the visualization time and their data point's time through computations in a set of custom vertex and fragment shaders (see Algorithm 1).

2.3 First Extensions

After producing a working 2D implementation of my basic idea, I experimented with other projections (Figures 3 and 4), eventually aiming for a spherical projection that would have potential as an effective virtual reality view.

Up until this point, I had been combining d3.js and three.js and used an awkward bridge between them to reliably communicate the projection and viewport information needed to render both a map and a particle system on the same screen (see Algorithm 2). I realized that if I switched entirely to a spherical projection, I could render the map using three.js rather than with d3.js, removing the need to operate through that additional level of abstraction.

Algorithm 1: Data Point Rendering

Data: $visualizationTime$, $minTime$, $maxTime$, $highlightPercent$
foreach *particle* **do**
  Data: $particleTime$, $latitude$, $longitude$
  1. Position in object space based on $latitude$ and $longitude$
  2. Compute $visPercent$ as the percentage that $visualizationTime$ is from $minTime$ to $maxTime$
  3. Compute $particlePercent$ as the percentage that $particleTIme$ is from $minTime$ to $maxTime$
  4. Compute $percentDifference$ and scale by $highlightPercent$
  5. Interpolate HSV color, size, and alpha using this scaled percentage
end

2.4 Final Details

3 Feedback

4 Results

Results.

5 Limitations and Future Work

Limitations and future work.

6 Conclusions

Conclusions.

7 Acknowledgments

Acknowledgments.

References

D3: Data-driven documents. `http://d3js.org/`.

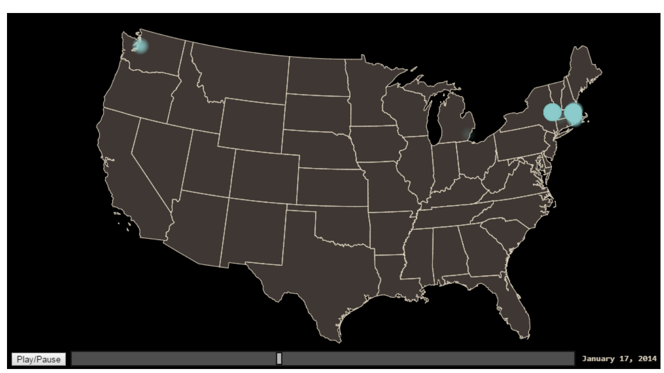

Figure 2: *Initial map of the USA with data point particles. Note the slider at the bottom of the screen.*

Figure 3: *Mercator projection with data point particles.*

Figure 4: *Orthographic projection with data point particles.*

Algorithm 2: Initial Data Point Positioning

At load time : **foreach** *data point* **do**
    Create a particle in the particle system, storing latitude and longitude
end
At run time : **foreach** *frame* **do**
    foreach *particle* **do**
        1. Convert from latitude/longitude to screen-space [x, y] using d3.geo.projection
        2. Convert from screen-space [x, y] to three.js world-space [x, y, z] using THREE.Projector.unprojectVector
        3. Update [x, y, z] vertex position for this particle
    end
    Render all particles using a custom vertex and fragment shader
end

Cardboard: DIY VR for all. `https://developers.google.com/cardboard/`.

Google takeout. `https://google.com/takeout`.

KONDO, B., AND COLLINS, C. 2014. Dimpvis: Exploring time-varying information visualizations by direct manipulation. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints, 1.

Detecting device orientation. `https://developer.mozilla.org/en-US/docs/Web/API/Detecting_device_orientation`.

Volcano data and information. `http://www.ngdc.noaa.gov/hazard/volcano.shtml`.

@THEOPOLISME. location-history-visualizer. `http://theopolis.me/location-history-visualizer/`.

three.js: A javascript 3d library. `http://threejs.org/`.

WHEATLEY, M. Virtual reality brings big data visualization to life. `http://siliconangle.com/blog/2014/05/29/virtual-reality-brings-big-data-visualization-to-lif`

WHITE, R. 2010. What is spatial history? Stanford University, The Spatial History Project.