EyeTrackerJS

Data Visualization Final Project, March 3, 2017

Chris Chagnon, Jeffrey Estrada, Tiffany Vo

Overview and Motivation of EyeTrackerJS

In the User Experience Community Eye-tracking has become a ubiquitous and well utilized commodity for tracking user interactions with a website. It tells a story of what the user does from beginning to end and defines trends on what the user looks for in a website. Unfortunately consuming, searching, querying, and sharing this data is difficult and, often times, restricted to proprietary (and expensive) software platforms.

The main motivation for this project is to explore innovative and novel ways of interacting with eye tracking data on the web, to expose its value to a wider market. It will look at ways to recreate the story from the given data and perform analysis for users to come up with their own hypothesis from the data. It will also be important to study the feasibility of using the web and if there are limitations in performance or scalability when working with these data sets.

Research Questions

The primary research questions we are attempting to answer are:

1. Can JavaScript (particularly D3) support large data sets for eye-tracking data, and render without taxing the browser or engine.

2. How can interaction with Eye tracking data be improved through the use of a real-time interpreted language such as D3 in JavaScript.

Research Questions (continued)

Throughout the project we had to remember our focus and make sure that the we were answering our research questions without allowing for major scope creep.

Question 1 was quickly answered because javaScript had almost no performance impacts from our large data sets. However, we had to modify this question from “Can it do..” to “How much of it”. We did hit barriers in the polling frequency (e.g. Javascript cannot call faster than 4ms

Research Questions (continued)

Question 2 was the most susceptible to scope creep as finalizing one feature meant we could see the potential for other features. It was easy to try and do too much, too fast, and too soon. We had to combat this by making a list of features we would like to do, but really focusing on refining the features we *did* have.

Objective	Benefit
To Create a tool / platform that allows for upload and interaction with eye-tracking data in an easy to use (and share) manner.	Wider-access to eye tracking data outside of proprietary software and formats.
To allow for customization of views and renderings.	There is a lot to be gained by allowing the users to customize their views, apply filters, and change rendering methods. In doing so, new information may be able to be derived.

Data Source and Information

We obtained data using Tobii eye trackers in the WPI UXDM lab. We record ourselves looking at an image and saving the data. The data itself includes time/index data as well as x, y, and z positional data and other information about the eye, subject, and image.

As we predicted, there was a lot of data from looking at the image for a minute. There were extra variables that were not necessary to our visuals. We narrowed down the numerous data columns; only data pertaining to our visuals remained.

Time Stamp

RecordingTimestamp	LocalTimeStamp

Gaze Point Data

GazePointIndex	GazePointX	GazePointY

Fixation Data

FixationIndex	SaccadeIndex	GazeEventType	GazeEventDuration	FixationPointX	FixationPointY

Data Clean Up

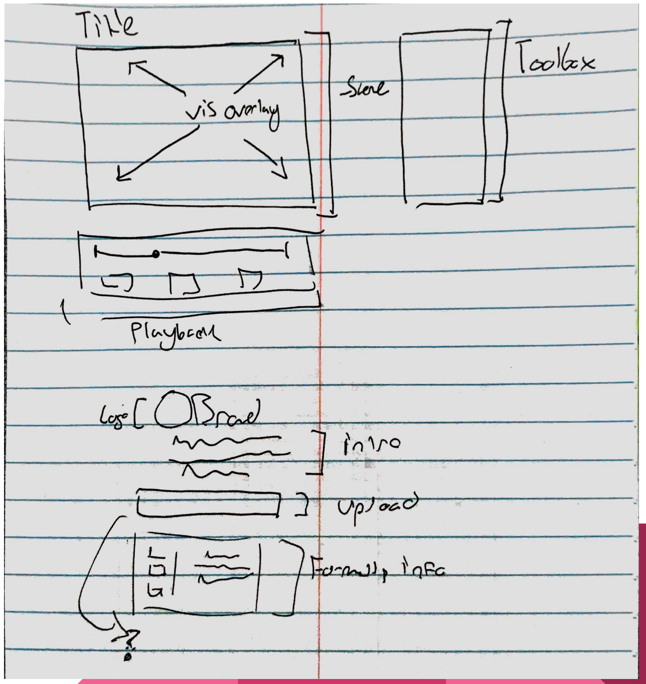
There were some rows that were blank for fixation data because the EyeTracker program computed the Saccade Index based on gaze duration in a certain time period and area of the image. However, those rows will still have gaze point data because there is a point for every recorded time stamp. There are also some rows that are blank because the EyeTracker could not detect the eyes on the screen. We did not manually delete any rows of data. However, in our code, we checked data going in to see if was equal to zero or NaN before continuing on with analysis of the data for the visualizations.

F	G	H	I	J	K
GazeEventDuration	FixationPointX	FixationPointY	GazePointIndex	GazePointX	GazePointY
551	532	207	116	537	208
551	532	207	117	542	191
551	532	207	118	558	176
44			119	562	29
70	546	13	120	567	29
70	546	13	121		
34			122	552	136
34			123	561	148
33			124	566	64
67	554	87	125	548	67
67	554	87	126	560	107
34			127	556	164
66	557	211	128	556	196
66	557	211	129	557	224
99			130	538	140
99			131	465	194
99			132	515	343
142	507	353	133	510	349
142	507	353	134	504	348

Design of Site

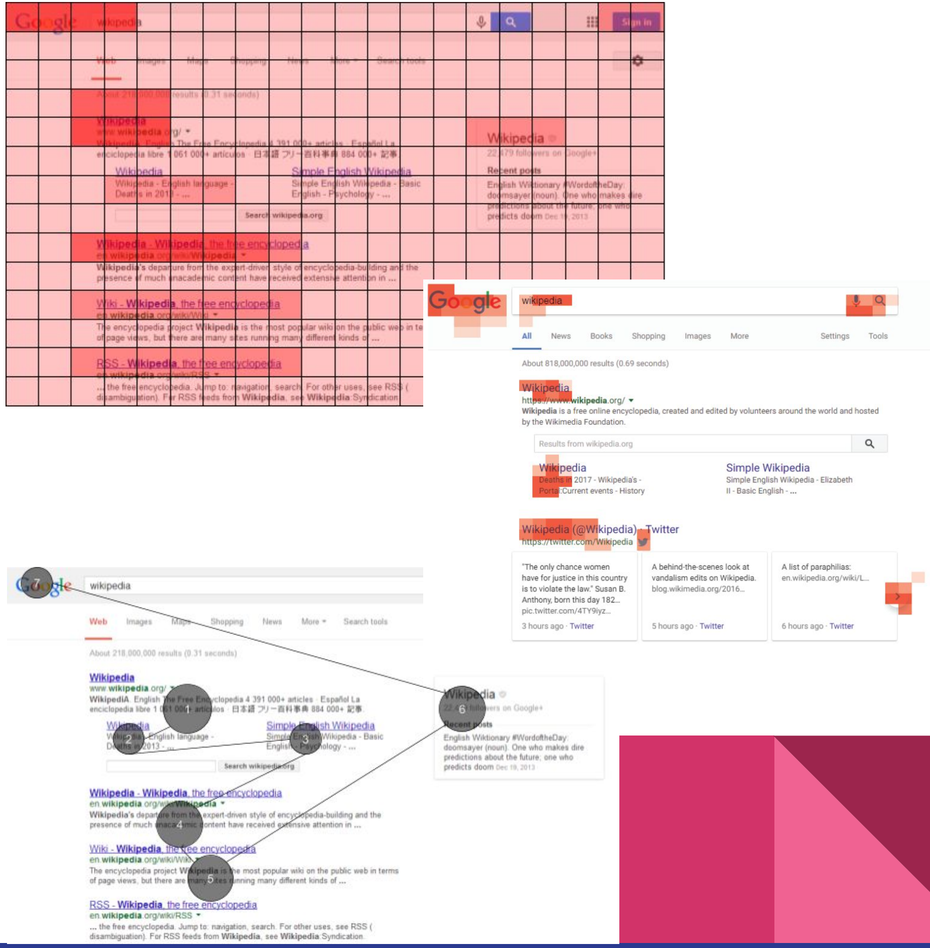
Wireframing session:

- ❏ Trying to figure out layout and functions. Why are things placed where they are?
- ❏ Do we need backend in PhP? Benefits? Advantages?
- ❏ Notes:
  - ❏ How will our site deal with responsive content?
  - ❏ If stage is finite and data refers to pixels, we need to handle scaling intelligently.
  - ❏ For larger scenes (websites) we cannot scale the image and data elegantly. We will need to dynamically change domain/range based on scene AND viewport.



Creating Visuals

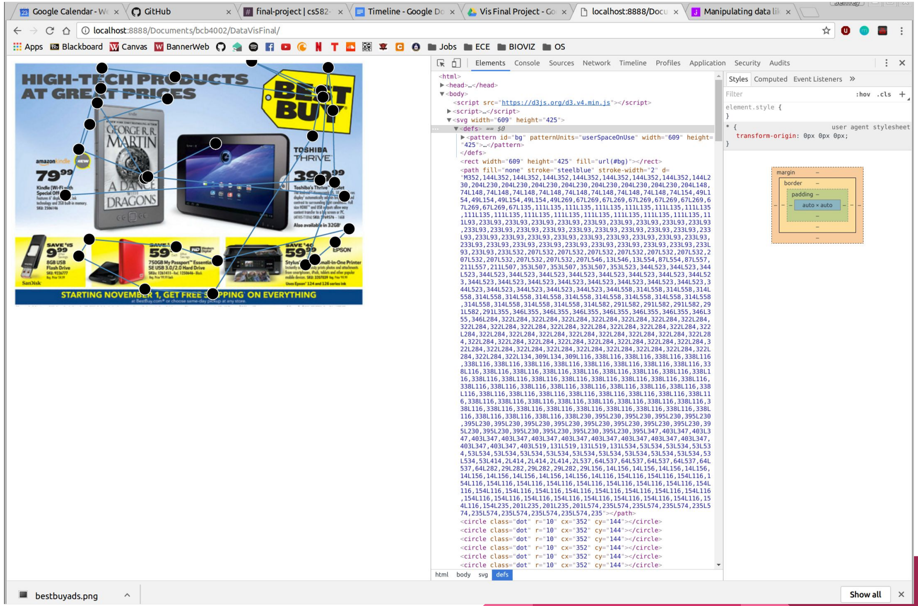
We needed to create some basic visualizations that will show our eye tracking data. This includes a heat map to show where the concentration of our users is going. A gaze plot in order to show the path of a user's eyesight on whichever image they are given. Then with these basic visuals, we can create a robust system that will show progression through time, and explore how a user interacts with a given visual. Shown on the left are our preliminary sketches, examples, and ideas for the visuals.



Creating Visuals: Gaze Plot

Iteration 1:

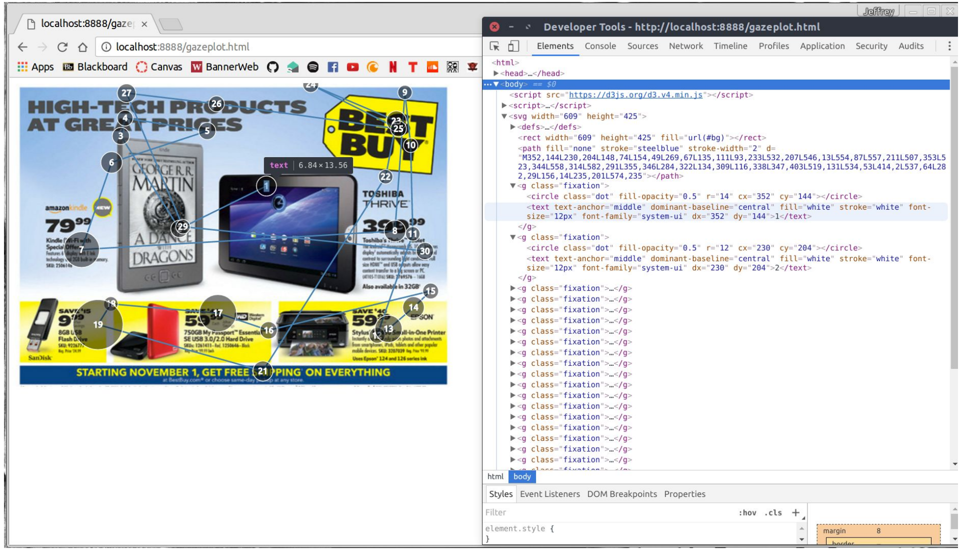
- ❑ Removed null points
- ❑ Still have duplicate, unnecessary points
- ❑ Need to add sequence numbers to circles
- ❑ Change size of circles depending on frequency



Creating Visuals: Gaze Plot

Iteration 2

- This plot now removes repeats in data, and for each node in the network, the information is placed in a <g> so that for future use it will become easy to modify
- Each node size is linked with the fixation duration
- Path is now a lot simpler, because we don’t have repeats of data points.



Creating Visuals: Heatmaps

Iteration 0

While first beginning, the heatmap started off as a scatterplot of squares just to view where at the data goes and figuring out how to create heatmaps in d3. That visualization did not work at all; the data did not make sense. It is all clustered on the right hand side of the image. Where the X and Y values scaled wrong? It turns out the wrong data column was used (ADCSpx rather than MCSpx, which changes how the image correlates to the pixels). However, it did help figure out how to pull in the data.

Creating Visuals: Heatmaps

Iteration 1

The program takes in the data and finds out which box (from 0 to 9 scaled to the pixels of the image) based on their coordinates. An array of arrays holds the number of times the eye looked into that area, incrementing by 1 for every data point in the pixel range. We mapped the data's X and Y coordinates based on the multidimensional array index and the Z axis color value based on the value at that array point. The top image is a 10 resolution heatmap and the bottom is a 20 resolution heatmap. At this point, we are able to make a heatmap, however the data still does not match the appropriate boxes.

Creating Visuals: Heatmaps

Iteration 2

The fix to the data not lining up from the previous iteration was to switch the X and Y values; they were getting mixed up. We also changed the z axis to be determined by opacity rather than color. That way if there are no gazes in an area, it has no fill rather than a light tint which falsely indicates data for that area of the screen. Finally, sometimes there are overlaps or gaps between the grid tiles which we believe have to do with decimal points in the math section. As before, the top image is 10 resolution and the bottom is 20 resolution.

Creating Interaction

The Scrub Bar

The playback tool was designed with a scrub bar that allows for the user to play the visualization in sequence according to the time index. This allows for selection, scrolling, scrubbing, and finite control of different points in the vis.

Playback

The play/pause button sets the visualization to auto-play through all of the data in sequence without manual input. There are also nuanced features that we implemented such as hovering over the scrub handle pausing the playback, and allowing playback from a given point.

Creating Interaction

The ToolBox

The toolbox is a feature which allows for manipulating data. For this project we focused on the two primary overlay filters for Gaze plots and Heatmaps, but longer term this could be where more advanced controls exist, such as changing playback speed, colors, opacity, and setting the resolution of the heat maps.

Evaluation

For both questions we found that sharing eye-tracking using javaScript is not only feasible, but efficient. Using our dataset we did not hit any true barriers. Even adjusting for size of the heat-maps to very small pixel-width grids did not seem to have too large an impact on the engine.

There are many more features that could be added. If done in a smart way, and modularly, we have the ability to continue development and feature rollout to try different treatments.

One main area that we will need to focus on is dealing with larger resolutions and responsive design of our viewer. It would make sense to be able to record items larger than our example ad, and scale our stage and visualizations to meet this. These considerations are important because not every person using this tool will be doing so from a monitor with the same resolution.