CSE512 A3

Michael Beswetherick, Brian Walker

February 10, 2014

1 Final Visualization Description:

For our project we decided to explore the data available on data.seattle.gov. Our visualization allows for a user to filter 911 calls by category and display where those incidents took place on a map of Seattle. The user can also filter by date and time of day. In addition, users can select an individual data point to see information about that particular incident; including time of incident, date, type and id. Since there can be many different types of calls in a single category, a custom search field was added to allow a user to search by keyword.

2 Differences Between Final and Storyboard:

The data set we chose to work with was the Seattle Police Department 911 Incident Response data set. The data is updated every four hours and thus provides an up-to-date look at crime, accidents, and other incidents around the city. We thought it would be interesting to visually display this on a map that would allow users to display the data based on date and time, type of activity and by district or neighborhood. Furthermore, we wanted to provide functionality to click on an individual data point to further display information about that particular incident. We thought about different ways we could display the data as well. We thought it would be great to have a heat map of the selected incident around the Seattle area. This would allow for a quick glance comparison if individual data points were not necessary.

Our final interactive visualization allows for all of the above with the exception of selecting based on a neighborhood and the heat map. We were able to find GeoJson data from Zillow that gave us the geometry for the neighborhoods around Seattle. However, the problem was that we decided to use Google Maps to display individual data points with D3. The svg elements are placed in an overlay on the map. This overlay is sandwiched in between other layers for the map. Thus that layer cannot receive mouse events. We overcame this with data points by putting invisible Google Maps markers on all the same locations as the svg elements. This makes it appear that the svg elements are clickable, when they actually are not. This would make tracking mouse coordinates to select a particular neighborhood very difficult.

The other option was to have a drop down that would allow the user to select a neighborhood. The problem with this is that the API only allows us to bring in data based on square coordinate blocks. The neighborhoods are obviously not square and thus quite a bit of additional logic would need to be added to filter anything outside of the boundaries. We decided to hold off on implementing the neighborhood selection to work on other aspects in case we ran out of time. We did run out of time and thus we were unable to implement it.

The heat map didnt make it into the final visualization because we simply ran out of time. There was a considerable amount of logic to get this to work properly and we were too busy fixing bugs and polishing other aspects of the visualization.

3 Development Process:

The development process took considerably longer than anticipated. The first challenge was finding a data set that actually was interesting. We initially wanted to explore vaccination data against the diseases they are designed to protect against. However, the data for Washington State is sparse and only in .pdf files; making the process of consuming the data a nightmare. We also looked at census and government data but finally decided on the data.seattle.gov Police 911 Incident data set. Once we decided what data set to use we worked on getting a storyboard put together so we could visualize how we wanted to lay out everything and what features we wanted to implement.

The obvious choice of how to explore our data was to filter on types of 911 calls. There were roughly fifty different 911 call types which is way too many to have check boxes for the user to select. We thought about using a drop down menu but it seemed like too many options for the user. Instead, we decided to bucket the different 911 call types into categories. We took what we considered to be the most interesting categories and used them as filters. We also decided to use a custom search box to allow for a user to search for a specific type of call if they so wish. That way if you only waned to search for homicides then you could do so without also seeing all the other violent crimes.

We also wanted to allow users to specify a date range to filter with. We added two text fields to allow the user to enter a start and end date. We also added a range slider to allow users to pick the starting time and ending time range. This option is not as useful unless the date range is only a single day. However, it provides more options for the user to easily explore the data.

The only other way to explore the data set is to use the sector information. However, I could not find any information about what sector maps to what area of Seattle. We wanted to allow for filtering on neighborhood but this proved to difficult to map from latitude/longitude to neighborhood boundaries. We thought the sector information in the data set might map to neighborhoods but the information could not be found.

The next step in our process was deciding on how we could implement all these features. We knew that we wanted to display it on a map and thus deciding how to do that took some time. We would have preferred to use GeoJson data to map Seattle but we could not find this data. The next obvious option was to use Google Maps. The API for Google Maps made it easy to use an overlay to plot our svg data points on the map and provided easy API calls to convert latitude and longitude coordinates to pixel locations on the map. We decided to use Google Maps for these reasons. It wasnt until later that we discovered that using d3 with Google Maps presented its own problems. We had to come up with a way to simulate a mouse over event using Google Maps markers instead of the actual d3 event listeners. This is because Google Maps inserts the overlay layer underneath a top layer that allows the user to manipulate the map.

After we decided which features to include and how we could implement them we got a rough design up and running. We then spent time fixing UI issues, fixing bugs and polishing the UI. Both Michael and myself worked on all aspects of the project. However, I worked primarily on connecting up the back-end logic and Michael worked on the UI elements and the front-end development. That way we could work simultaneously without stepping on each others toes.

The project as a whole took us around 40 hours or so to complete. Longer, if you consider the time it took to find a data set that we wanted to use. The portions that took us the longest were getting the different frameworks and libraries to work together. We used D3, Google Maps, SODA 2, and JQuery to implement all of our features. Getting D3 and Google Maps working together was the hardest hurdle to overcome. We had to come up with some creative solutions to get the functionality that we wanted. It also took a considerable amount of time getting the API for SODA 2 to work properly. Socrata recently updated their API and their website. There are many aspects of the API documentation that is missing or incomplete. It took us at least several hours to just get a basic query working.

4 Running The Code:

The code is entirely web based and all the frameworks needed are included in the GitHub repository. All that should be needed to launch the website is to open index.html in the code directory. Also, it is posted on a UW server at `http://homes.cs.washington.edu/~bdwalker`.