Mikhail Kouzminov

CMPS 161 Assignment 1 Technical Document

The Problem Given:

The goal for this assignment was to create a system for predicting and visualizing wind velocity and flow. Uses for this include visualization of theoretically where massless smoke would go, or as part of a system for predicting dust flow. There are many meteorological stations, or met-stations that are used to predict wind around us, and there are several possible ways to display wind.

The approach:

Data Used:

In the case of this assignment, I decided to visualize the flow of wind in the relatively small space of the Bay Area, from Gilroy to San Francisco. Since wind is usually recorded as an average over a period of time, I decided to see on how small, and therefore accurate (since wind is less likely to change), a time period I could gather wind data on. While there are many met-stations in this area, many of them only record the average wind for the month, or for a week. I decided to make the wind data I use be recorded over as small a time period as possible. As such, I somewhat arbitrarily decided to take my data Friday, February 1st, at 12 AM. However, since I was only looking in a small area and on that particular date, I only found 7 stations with data properly recorded, as some stations were out of order, while others only took readings every day.

It is possible for a user to provide their own JSON string to upload information on stations in this area. Simply have a file named "stations.json" in the same folder as the code. It is also fairly easy to change the coordinates of the area displayed by changing the code a little. The data I found and ended up using came from Windfinder.com and was:

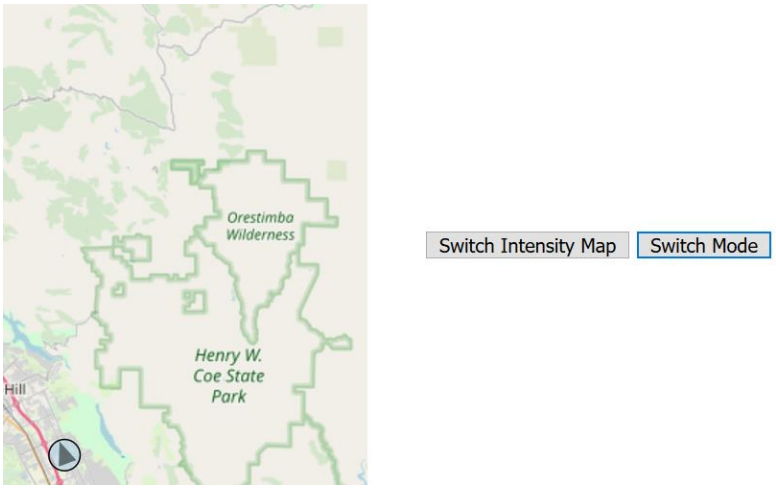
Location Name:	X coordinate	Y coordinate	Angle (in degrees)(taken clockwise from the South)	Wind strength (mph)
Sant Clara County Airport	-121.58	37.08	160	3.45
Los Gatos	-121.95	37.20	178	10.36
Mountain View/Moffet Field	-122.04	37.40	150	5.75
San Jose International Airport	-122.91	37.36	150	4.6
Los Altos Hills	-122.13	37.35	251	2.3
Poverty Ridge	-121.75	37.44	209	5.75
La Honda	-122.25	37.30	243	1.15

Representation/Graphics decisions:

UI design:

I decided to go with a simplistic representation with all the required factors. Namely, there is a simple map and it is possible to turn on a heat map by pushing a button, and to switch between displaying glyphs at stations, at square locations, and streamlines at square locations by pushing another button. One thing to note is that the 2 buttons are on the right side of the screen, which is easier to see than on the bottom (at least on my wide-screen display). The heat map button is labeled as intensity map, since we are displaying wind velocity on a weather map, not temperature, so the name heat map might have gotten confusing in this case. This display might also be improved by

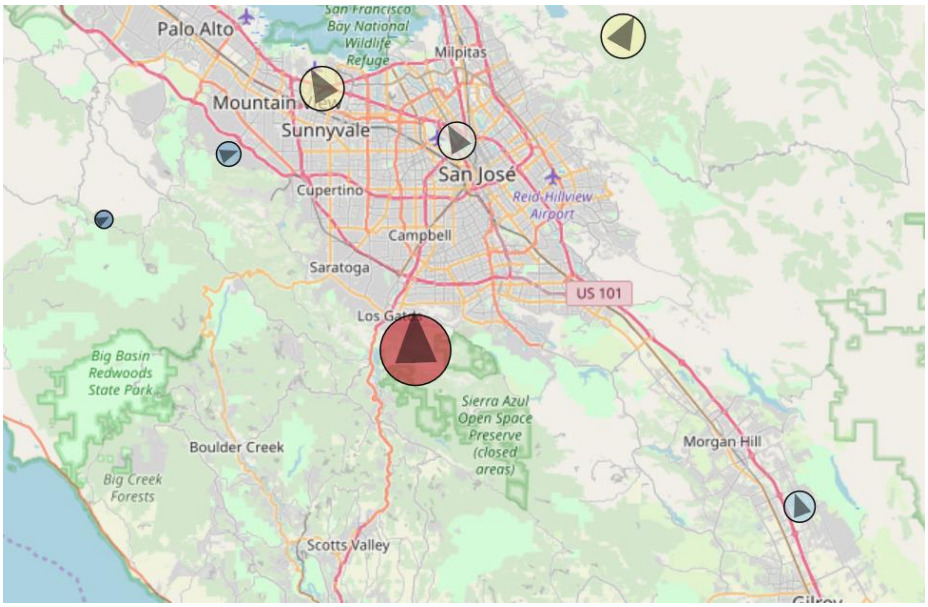
changing the text on the Mode button to be what the current mode is, but I judged the content of the screen to be sufficient to explain this, so I left the text constant.



A display of the basic UI with part of the basic map with markers to the side. Note that buttons are easily visible.

Basic Map with stations displayed:

Since I felt the data might have gotten too crowded if I simply used arrows, and data represented via colored arrows might not be accurately readable, I decided to make my arrow glyphs be arrow markers of various sizes pointing in certain directions, with an underlying colored circle. This both provides a background so that the arrow is clearly visible against a solid backdrop instead of the map, and allows more precise obvious representation, as arrow color scales like a heatmap, even when the heatmap is off. It even allows certain details behind semi-transparent markers to be visible. This representation may however lead to the angle of wind being displayed less precisely than it would be possible with a longer arrow.



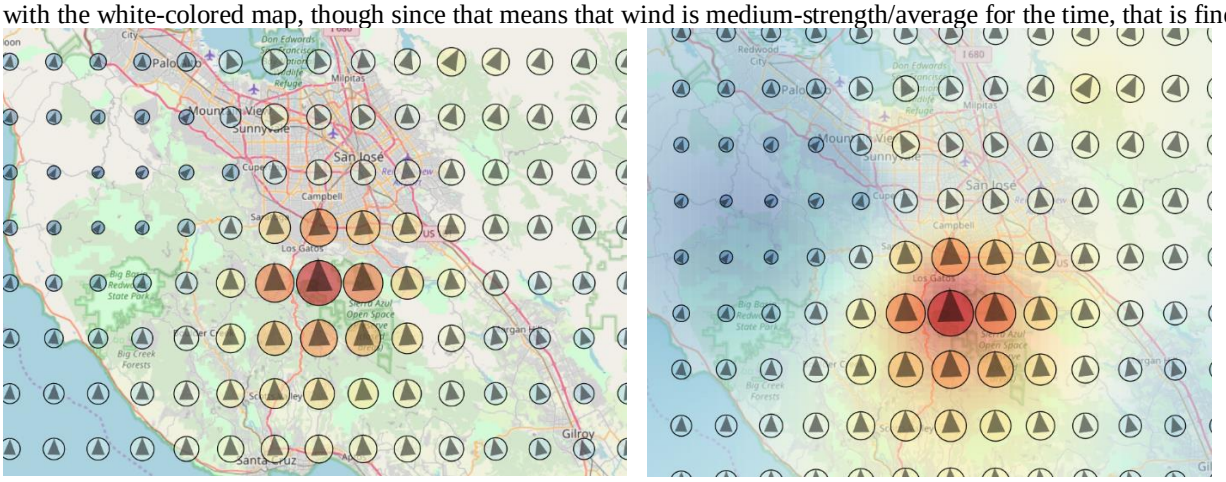
Map with stations represented – note scarcity of available stations, as well as how arrow markers allow wind information to be displayed clearly without hiding information from the original map

Arrow Plot and Heatmap:

Since markers are small and transparent, many more than usual can be placed without losing map detail. My arrow plot uses 200 markers arranged in a 20 * 10 grid pattern. This allows both more information to be displayed, but also, due to markers having gradual color changes, to also work to show heat fairly accurately even when compared to the actual heatmap. This may have changed if I had chosen more points to interpolate data from, leading to more interpolated spots with high activity and therefore greater arrow size, this type of representation seems more precise to my eyes. I also specifically chose to have a minimum size for the glyphs, since the one on the left was too small to see when I first drew it.

One thing to note is that since I didn't have any weather stations close to the Pacific Ocean, and since the strongest source of wind is the Northern wind in the middle of the map, the Shepard's interpolation finds there to be a weak Northern wind over the Pacific, which is unlikely to be true (in reality, it is most likely to be westward).

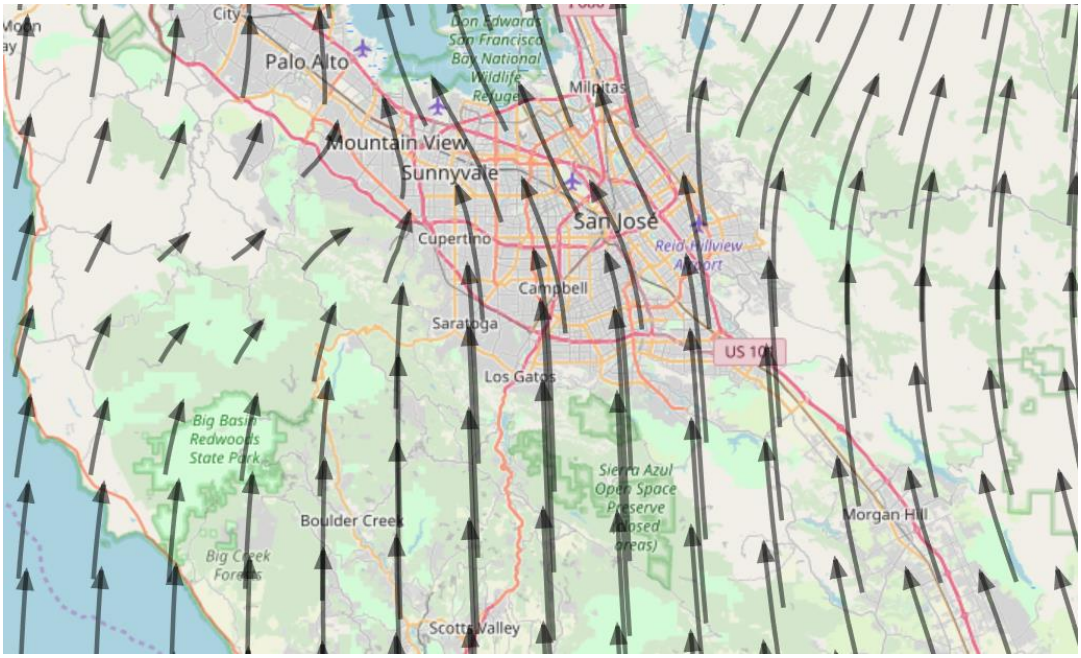
As for the heatmap, while a small resolution such as 100 * 100 does lead to some parts of the map looking slightly pixelated, the map is small enough that it isn't very jarring. In fact, higher resolutions have more space visible between squares and as such can have the effect of looking more obviously pixelated. The Red-Yellow-Blue color scheme also requires less explanation for a heat map than any other, and as such is less likely to require a key. However, having blue coloring on a heatmap over the Pacific Ocean might be confusing, since it is harder to tell whether wind is particularly week. This is however made up for by the inclusion of yellow in the color scheme and that it is at the very least possible to tell that the wind is in general weak. Likewise, yellow also blends in somewhat with the white-colored map, though since that means that wind is medium-strength/average for the time, that is fine.



Images of the Arrow Plot, with and without the Heat Map. Note how due to arrow coloring matching with the Heat Map, the arrows in the image without the Heat Map still allow the user to see its original shape.

Streamlines:

While making streamlines, I experimented with length and step count some, in order to try and find a map that wasn't too busy. Eventually, I decided to use a streamlines with 20 steps, each of time ½ hour. Since the lines are all fairly parallel, using this many lines still allows the original map to be visible, allowing a more thorough understanding of wind direction. While there may be too many lines, and they may be too close together in cases with more dramatic wind change, but that does not happen in this example. The semi-transparency also helps in that it allows data on the map without over-crowding it.



The busiest part of the streamlines map. Note how there is little overall change in streamline slope, and that despite fdensity of arrows, the lack of change allows the map to be understood. It would be a lot harder to read if slope changed more drastically.

Additional Programing choices:

I decided to have the calculations for each part of the map occur in the code right before the portion where the part is created in SVG. Because of this, it may be more confusing than doing all calculations, then all SVG drawing, but I thought this would be best and allow a good grouping by part that code creates, which I thought was less confusing than grouping it by type of code. This way, however, it may be more confusing to change the variables that control various parts of the code, such as # of steps in a streamline path, which is defined right before the calculations for streamlines, and not the top of the code. Likewise, according to some code users some of the math might be a little confusing and run too long in one line, and there may be too many variable names, but I tried to find a good balance in this case of compacting code into a smaller line and having variables that are never displayed but go into calculations.

Mathematically speaking, distance is calculated in each square by translating to geographic coordinates and finding distance in radians with the geoDistance D3 function. This both reduces math used and is more accurate, since it accounts for the slight curvature of the Earth (though this is unlikely to matter). Likewise, the current method of calculating values for every square and keeping them in matrices is also somewhat more memory-extensive. However, it is still not a lot of memory usage, and reduces the number of calculations actually performed, since that data is used more than once.

User Suggestions (Implemented in code after receiving):

I changed the name of the Heat Map button to intensity map, since Heat is confusing (too similar to temperature) on a weather map.

I made the streamlines thinner, after being told my streamlines take up too much of the map space.

Bibliography/ Code Sources Used:

Main Source (from which I took most of my starting code, including the map/tiles system:
http://bl.ocks.org/mbostock/eb0c48375fcdcdc00c54a92724733d0d

Tutorials for markers (Borrowed a decent amount of code):

http://tutorials.jenkov.com/svg/marker-element.html

http://bl.ocks.org/dustinlarimer/5888271

Wikipedia article on Shepard's Interpolation:

https://en.wikipedia.org/wiki/Inverse_distance_weighting

Wikipedia article on RK4/ Euler methods:

https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods