Andrew Pang
anypang@ucsc.edu
CMP 161/L - Introduction to Data Visualization
Instructor Alex Pang
2/13/19

Programming Assignment 1
Real Time Weather Map

1. Problem

The purpose of this project is to represent real time wind patterns all across California. Using OpenWeatherMap API and Google Maps API, real time wind pattern can be displayed. Within a 10x10 grid, data measurements from weather stations are represented in different ways such as a heatmap, colored vector field, and streamlines.

2. Approach

2.1. Software Used

1. OpenWeatherMap API.
2. Google Maps API for Javascript.
3. Chroma.js Color Scale Helper

2.2. Map Display

I decided to use the Google Maps API because of the large documentation and support that already exist within the community. Much of this project is done through this API. “google.maps.Map” class is used for the map projection. The “google.maps.LatLng” class is used for geographical coordinates. To draw on the map, I used the “google.maps.Polyline” class. The 10x10 grid overlay, wind arrows, and streamlines are drawn with this class. Furthermore, the Google Maps API also includes a “google.maps.visualization.HeatmapLayer” class, along with an extra parameter for the weighting value of the data point. In my case, the weighting value is the wind speed measured.

2.3. Data

To obtain the weather data, I used the OpenWeatherMap API. The goal was to capture the data for the whole state of California so I used a 10x10 square grid to bound the state. But since California is an odd shape, by consequence of the square grid, I also encapsulated the neighboring state, Nevada, and parts of the California Coastal ocean. This way, we can get a bigger picture of which direction the wind currents are coming from. The main API request string I use, calls the current weather data for several cities within a rectangle zone. The data returned is in JSON format, which will be needed to be converted to Google Maps’ LatLat class for map display. The API decides which weather stations it obtains within that rectangular zone but the density of weather data is specified by the user. Since my goal is to capture the entire state, I needed to obtain enough data to represent the wind pattern accuracy without obtaining too much that noisy/duplicate data will slow down the program.

2.4. Representation

The four representation of data is Heatmap, Station Wind, Arrow Plot, and Streamline. Heatmap takes the data from OpenWeatherMap and the interpolated winds and plots them on the map. The heated

areas represent areas of higher wind speed. The Station Wind displays the data taken from the OpenWeatherMap API call. The length and color of the arrow/wind reflects the wind speed. Color is picked from a range of color hues. A gradient was created with the help of Chroma.js Color Scale Helper. Bluer arrows are slower than red arrows, while purple would be the intermediate color. Arrow Plot arrows represents interpolated winds computed using Shepard's interpolation method at each grip point. The Streamline arrows displays winds computed using Euler integration. Each grid point was used as the initial point then iterated forwards/backwards as each point/coordinates is computed from the previously iterated point.

2.5. Limitations/constraints

A few limitation come from the OpenWeatherMap API. For the free version of the OpenWeatherMap API, I am only allowed no more than 60 calls per minute and weather data can take up to 2 hours to update but, this allows the program to have relatively real time data. Display time for streamlines is considerably noticeable. A constraint is how there is no historical data. It would be nice to see wind patterns over time but as a free user of the OpenWeatherMap, current data is only accessible. Historical data can be accessed at higher tiered payments.

2.6. User Interface and Features

A nice feature of this program is the real time data. Weather station measurements are called from the OpenWeatherMap API everytime the html file is opened/refreshed. No external library or software should be needed to run the file. Different data representations can be chosen by the buttons near the top. One other feature is the terrain overlay map, thanks to the Google Maps API. With the terrain overlay, mountains and valleys are displayed. Together with the streamline view, it is easy to see how mountains affects the wind such as by creating the Santa Ana winds.

3. The Result

A real time wind map with minimum library requirements for lightweight, fast, and shareable program. See html for result pictures.

References:

- *OpenWeatherMap API*
  - https://openweathermap.org/
- *Google Maps API*
  - https://developers.google.com/maps/documentation/javascript/reference/
- *Shepard's method*
  - https://en.wikipedia.org/wiki/Inverse_distance_weighting
- *Chroma.js Color Scale Helper*
  - https://gka.github.io/palettes/#colors=blue,red|steps=12|bez=1|coL=1
- *Euler's Method*
  - https://en.wikipedia.org/wiki/Euler_method
- *Bilinear Interpolation*
  - https://en.wikipedia.org/wiki/Bilinear_interpolation