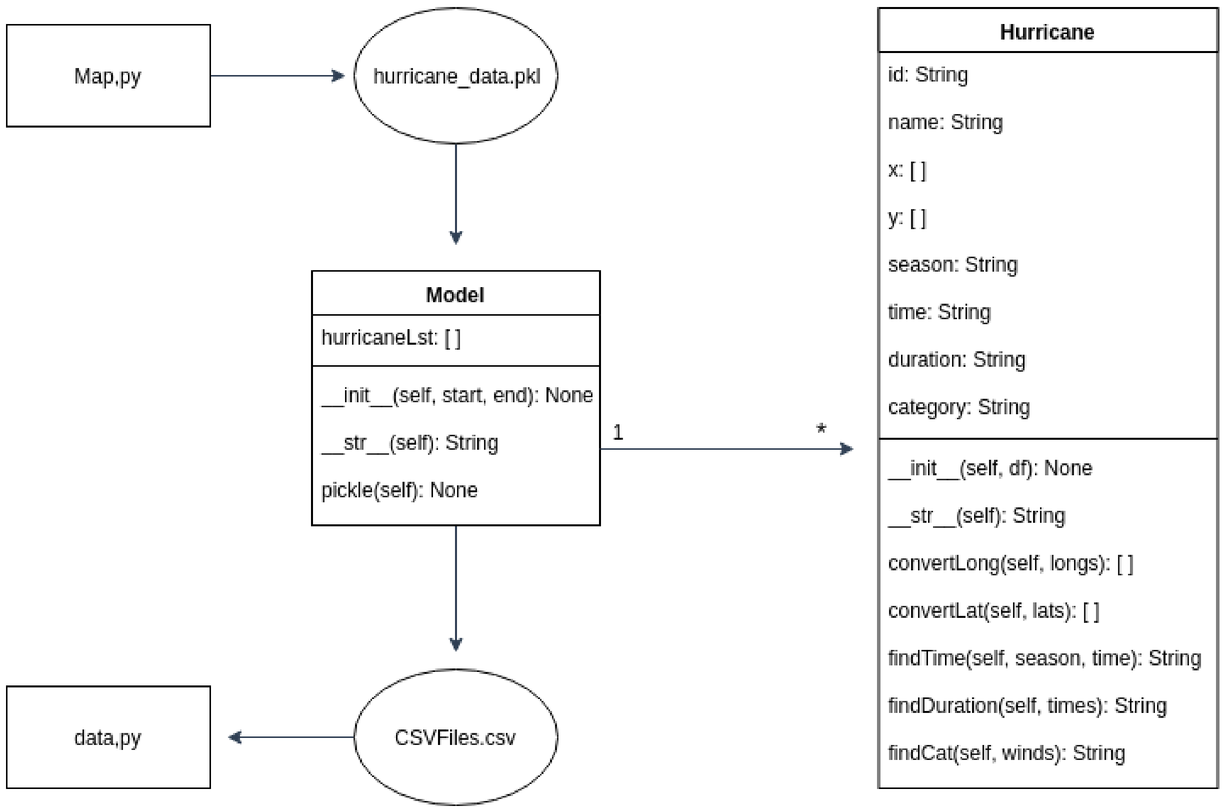
Cat Map Project Write-up and Reflection

Project Overview

Cat Map is an interactive visualization of hurricane data within the NOAA database. The data is obtained by accessing csv files stored in a FTP server. Various hurricane paths are plotted on a map that users can easily zoom into and explore. The points and paths are color coded according to the category of the hurricane. Additional features include a slider for year that determines which hurricanes are plotted and a hover window that is generated for every point in a hurricane path. The hover includes the name, time, duration, and category for a given hurricane.

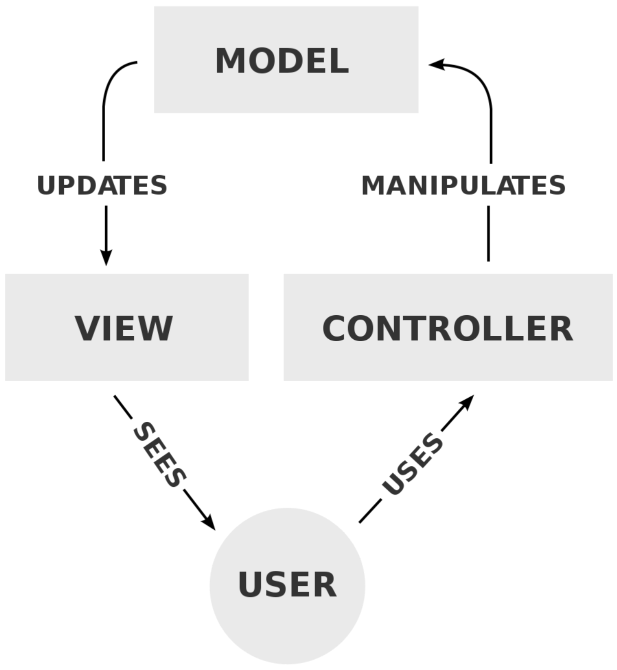
Implementation

The Cat Map utilizes the model-view-user-controller process and object-oriented programming. The model, which is the map, updates based on the changes that the user makes. A majority of the interactive components are managed by the bokeh library. The figure below shows a UML diagram of our program.



The two main classes are Model and Hurricane. A hurricane object represents the relevant data for a single hurricane from the NOAA database. A model object represents the collection of hurricane objects that are plotted. In Model, “CSVFiles.csv”, which is created in “data.py”, is referenced. Model also generates a pickle file, “hurricane_data.pkl”, that is inputted into “Map.py”. Each python script has detailed documentation on specific functions and methods.

A major design decision that we experienced during our project was the method of storing and updating data in bokeh. At first, we just plotted the points and paths of the hurricanes, but when we wanted to add in a hover, we found out that we needed a ColumnDataSource to store each hurricane’s information. After adding in a ColumnDataSource for each hurricane, we were able to get the hover working, but when we tried to implement the slider, we discovered that we needed one ColumnDataSource for all of the data. With that change, we got our MVP working but then experienced more challenges involving rendering and hovering. In a final attempt, we restructured our bokeh code to have a callback function that creates a new ColumnDataSource each time the value of the slider changed. This solution fixed our problem, but now we have to run “Map.py” using the bokeh server instead of just outputting the map to an html file. Even so, our design decisions all made sense and improved the overall functionality of Cat Map.



Results

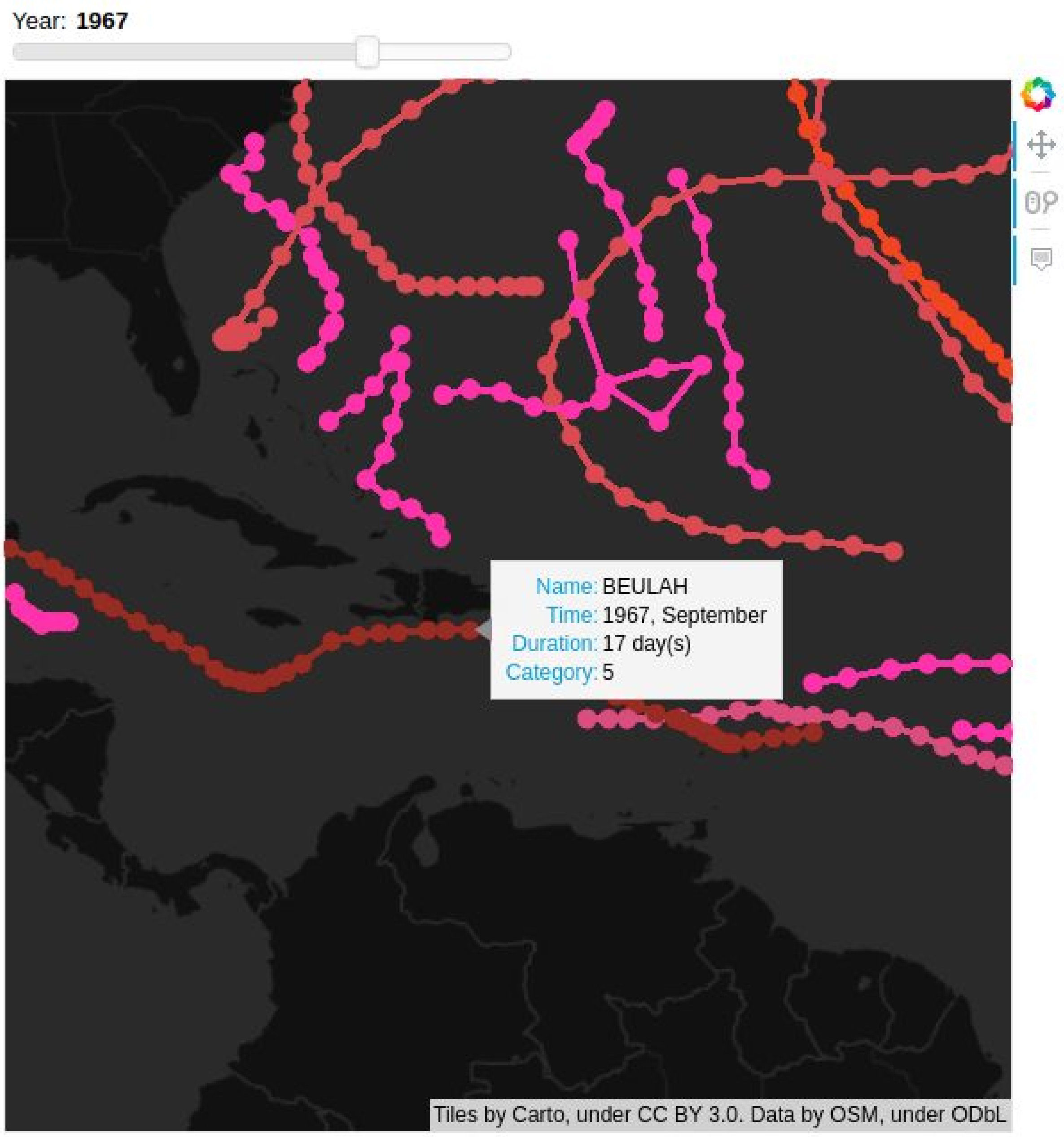


Figure 1: Results from the Cat Map with an emphasis on the hover tool

As seen in Figure 1, the Cat Map displays hurricane path data on a world map. We plotted over a black map tile to make the hurricane data the central focus. The hurricanes themselves are plotted in 7 total colors, according to the intensity of the storm. The colors range from white for no data available to shades of pink for tropical storms and depressions to shades of red for each hurricane category. The storm intensity is also represented in the hover tool under "category". Hover also includes other essential information about each storm such as name, hurricane season, month, and duration. This tool is accessible by hovering over any of the points on a hurricane path.

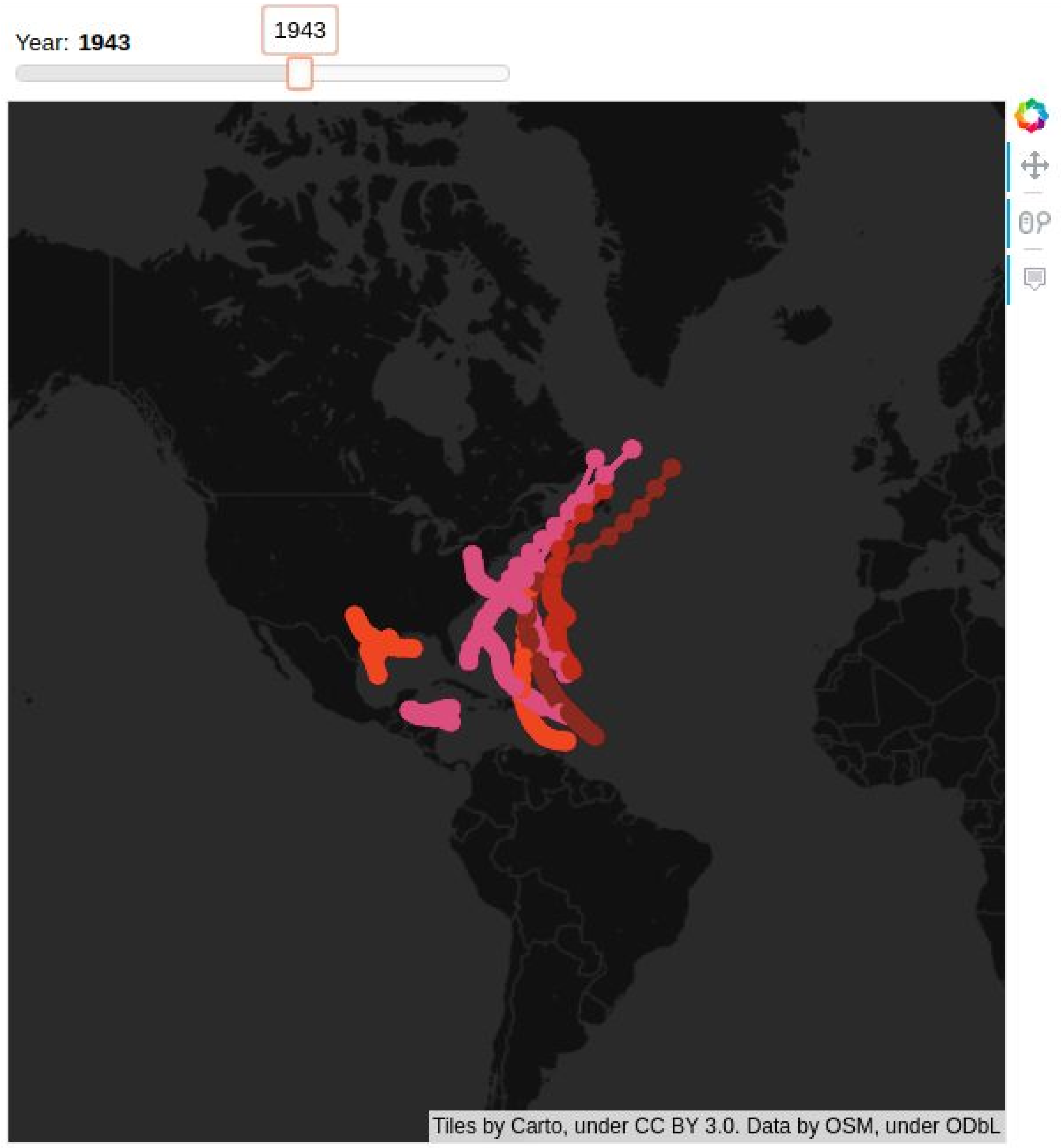


Figure 2: Results from the Cat Map with an emphasis on the slider tool

In addition to the hover tool, we implemented a slider tool seen in Figure 2. This allows us to focus on the hurricane data for each year individually. This is the main feature we sought to add after seeing the NOAA database display hurricane data without regard to time for given areas. We were able to pickle our data, allowing us to process and display data from 1842 to 2017. Thus we can interact with this map via the hover and slider along with the built in pan and scroll tools.

Reflection

Overall, our project went really well. We were able to accomplish most of our goals. We started with the idea of visualizing the NOAA hurricane data in an interactive way and succeeded in creating a visually appealing and interactive map. One reason why we were able to accomplish so much was because of our efficiency. We finished the MVP relatively quickly, which allowed us enough time to expand on some more features. Our design process involved setting up a solid foundation and then constantly improving on it by iterating on the code. We sometimes had to make some big changes from file to file, but overall, the diagram of our program remained relatively consistent.

One thing that could be improved is the efficiency of our code, especially when all 13,000 hurricanes are included in one map. It still takes a while for the rendering. Another improvement would be to find a way to run the map without using the bokeh server.

Overall, the project was slightly out of our scope, but we worked hard and accomplished it. Next time, we would try to start out with a simpler idea and think of cool expansions after getting the basics down. The main takeaways we got from this project were the power of Bokeh and the ability to write code with the help of documentation. Bokeh is a very useful library for data visualization that we are now comfortable to use in future projects. Though, we wished we knew more about how bokeh deals with data before starting on our project, because we spent a lot of time figuring out the best way to get our features to work with our data. Some of that time could have been invested to develop other features.

In terms of our team process, we had good team dynamics and similar views regarding the project. We pair programmed during class and divided the work outside of class. We also met a few times throughout the week to update each other on our progress. We ended up not pair programming as much as we intended to because of time constraints and scheduling conflicts.

Next time we will attempt to adjust our coding strategies to align better with each other. Sometimes, one of us would get super excited and sprint away with coding. We will also split up the work in a better way to emphasize what each individual wants to learn.