CS251 Proj4

CS251 Project 4 - Integrating Data and View

The main goal of this project was to allow the user to plot interactive data on the axes. This week we brought together data, GUI, DialogBoxes, buttons, and viewing. The result was an interactive visualization application that can read in a data set. The user should(at a minimum) be able to interactively view the plotted data in up to 5 user selected dimensions -- 3 spatial, 1 color, and 1 size.

USER MANUAL:

?1. Select Data File: First off, the user will select a csv data file to be read into the application. They can choose the file by either clicking the plot data button, choosing "File, Open" or pressing "command-O". All of these will call handleOpen which opens a window where the user selects the data file they want to open.

2. Select to Plot Data: Now, once the user has selected a file to read into the application, he or she now has the option to plot the data on the 3-D axes! The interactive application allows the user to do this by simply clicking on the "Plot Data" button on the left panel of the screen. This will immediately take the user to my handleChooseAxes function and open up interactive DialogBoxes. As an **extension(3)** to the project to handle functionality, if the user clicks on "Plot Data" before reading in a csv data file, the application will automatically called handleOpen and make the user choose a data file before continuing.

3. Pick Axes, Color, Size and Shape: Next, once the user has both read in a data file and selected "Plot Data" which called handleChooseAxes, DialogBoxes will pop up. These DialogBoxes will allow the user to interact with the application and ultimately choose a number of dimensions. By default, for the given csv data file the user will be able to choose the data on the X-axis, the data on the Y-axis, the data on the Z-axis, the color of the points, and the size of the points. As an **extension(1),** I added another option to the user selected dimension, shape! So, when the user clicks plot data, 6 DialogBoxes will appear asking them to choose X, Y, Z, Color, Size, and Shape!

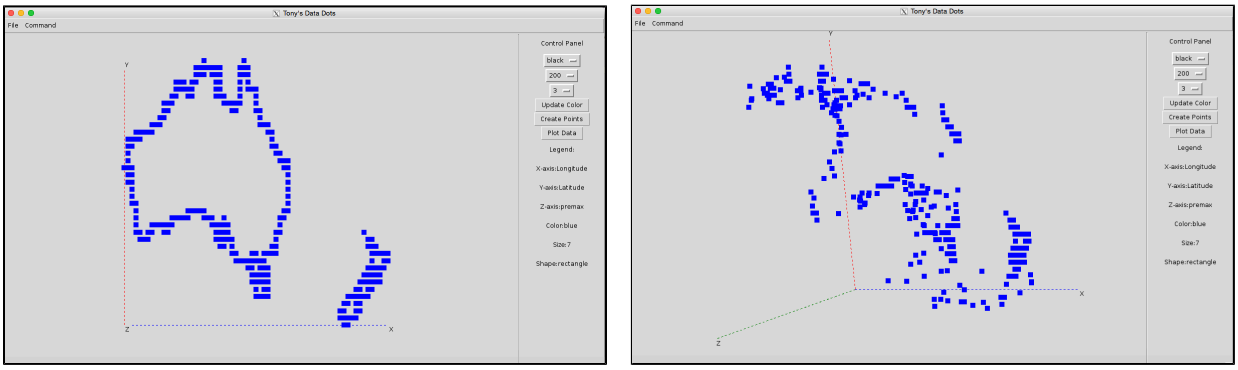
4. Build the Points: Now, the application will actually build the points we want to graph after the user has selected its DialogBox options. Our buildPoints will start by deleting our entire canvas by calling a function I made called totalReset that resets the axes **and** deletes the data points. Next, the data of the selected headers, color, size , and shape data are normalized. This data is then transformed using the vtm and the correct shapes are assigned to the plots.

5. Interact with the Points: Once the points are plotted, the user can easily move the axes **and** the plotted points around the screen simultaneously. The points can be moved, rotated, and scaled just like the axes. To handle this, we have the function from Task 1 of the project called updatePoints.

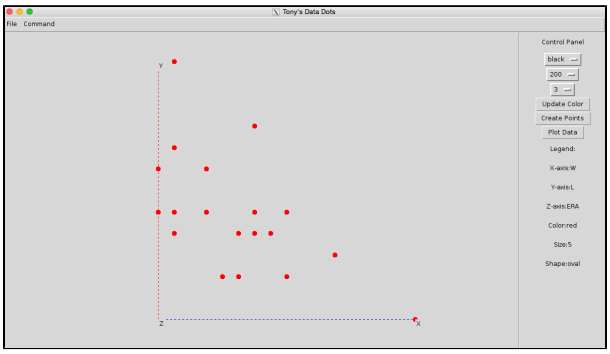
6. Legends!: One of my extensions was to create legends that let the user know what each axis represents and the selections for color, size, and shape.

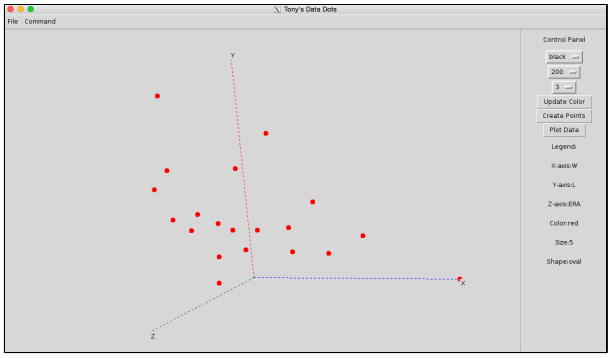
Pictures:

Picture with AustralianCoast.csv data:



Here are pictures with my data MLBPitching.csv from Project2:





Extensions:

Extension 1. Shapes! I simply added another dialog box and gave the user the option to choose the shape of the data points. They can be either rectangular, oval, or arc shaped.

Extension 2. Functionality! This handles the possibility that the user clicks the "Plot Data" button before they have chosen a file to be read and used. If no file is read in yet, it will ask the user to select a file to be read in before moving to handleChooseAxes and plotting data.

Extension 3. Legends! I received help from Julia Saul with this extension to create a Legend on the right side panel of my application display. Here is a snippet of the default code for the legends. The legends update later in my function once parameters have been selected:

```
#----------Project 4, Extension 3----------
        #EXTENSIONN3
        text = tk.StringVar()
        leg = tk.Label( rightcntlframe, textvariable = text, width=20 )
        leg.pack( side=tk.TOP, pady=10 )
        text.set("Legend:")

        #Creates a legend for the X-axis
        self.xaxisLegend = tk.StringVar()
        leg = tk.Label( rightcntlframe, textvariable = self.xaxisLegend, width=20 )
        leg.pack( side=tk.TOP, pady=10 )
        self.xaxisLegend.set("Have not chosen X-axis")

        #Creates a legend for the Y-axis
        self.yaxisLegend = tk.StringVar()
        leg = tk.Label( rightcntlframe, textvariable = self.yaxisLegend, width=20 )
        leg.pack( side=tk.TOP, pady=10 )
        self.yaxisLegend.set("Have not chosen Y-axis")
```

Here is what the Legend looks like with selected values:

Wrap-Up:

What I learned. This week I was able to bring all the elements of my past projects together and create a successful interactive data visualization application. The user is fully capable of choosing data to read in and interacting with its axes, size, shape, and color. Overall, I gained a better understanding of Python, GUI, data visualization, and DialogBoxes.

Who helped me. I received help this week from CS251 classmates Julia Saul and Steven Parrott. Also, I worked alongside Steven Parrott and

Jay Moore.