Text Mining

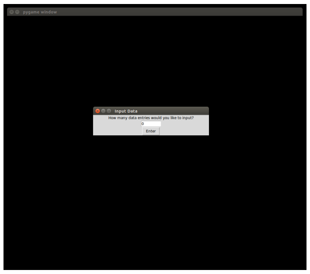
Carl Moser and Serena Chen

March 2015

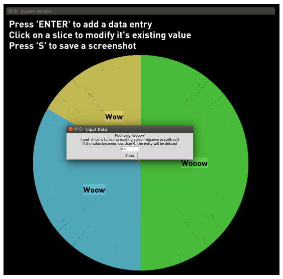
1 Project Overview

The goal of our project was to make a GUI that allows people to create charts. We decided to specifically focus on pie charts. Users can input values and the program automatically and dynamically generates the pie chart.

2 Results

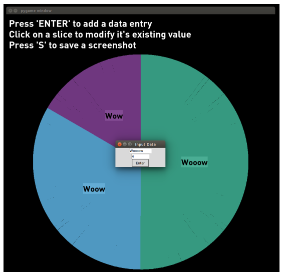


How many data entries would you like to input?



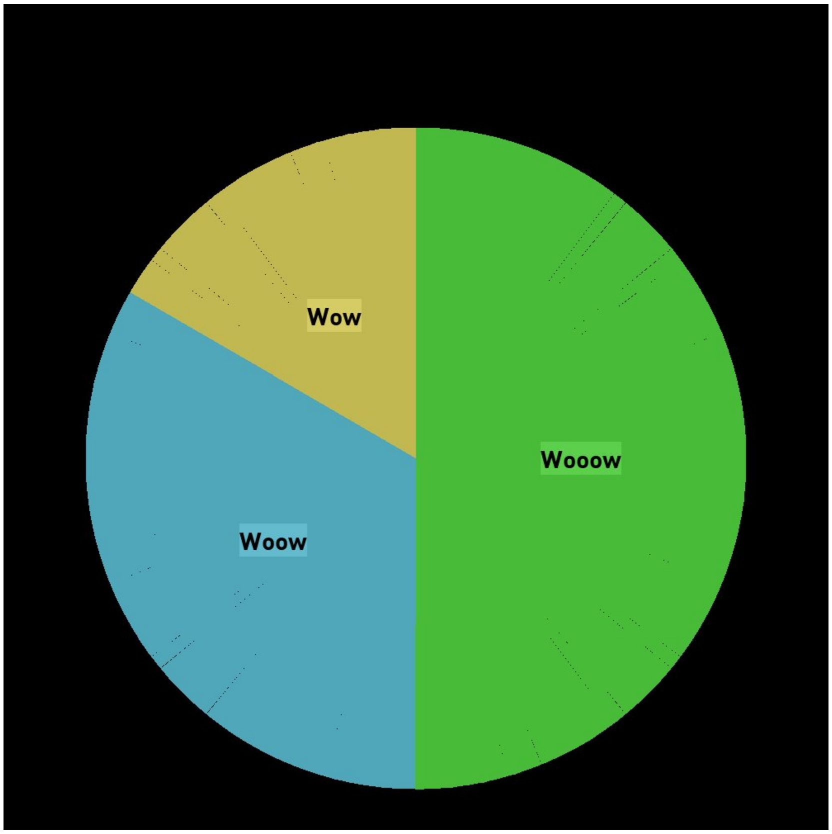
Press 'ENTER' to add a data entry
Click on a slice to modify it's existing value
Press 'S' to save a screenshot

Modifying 'Wooow'
Input amount to add to existing value (negative to subtract)
If the value becomes less than 0, the entry will be deleted



Press 'ENTER' to add a data entry
Click on a slice to modify it's existing value
Press 'S' to save a screenshot

We ended up with a GUI with dialog boxes that guide the user through the chart making process. The method for inputting data is quite simple: when the program starts, the user is prompted for the initial amounts of data points they would like to add, and then prompted for those data points. The graph readjusts every time the user adds a value to the graph. There is also input checking; for example, if the user enters a value that is not a number, they are prompted with an error and allowed to retry. Likewise, if the user inputs a label that is already in the graph, the user will be prompted with an error message as well. In the main pygame window, the user can press Enter to add another entry, 'S' to save a screenshot (without the words at the top), 'V' to toggle whether the raw values are shown, and the user can click on a slice to modify it's value. If the user modifies the value to be 0 or negative, the entry is deleted from the graph.



3 Implementation

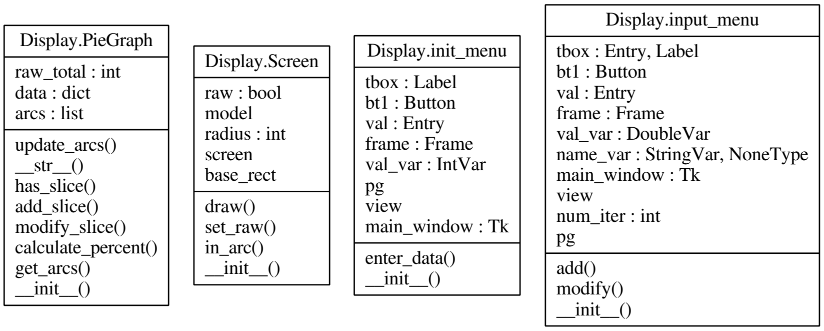


Figure 2: UML diagram for our project

Our project consists of four classes and a main method. Class Pie Graph handles storing the chart data and calculating the proportions of the slices.

Screen is the main window, and handles displaying the pie graph, along with other information needed on the screen. Classes init_menu and input_menu are tk dialog boxes that facilitate the data entry side of user input. The data for the pie graph is stored in a dictionary, and to pass data between PieGraph and Screen, a dictionary was used, where the keys were the names of the attributes and the values were the values of those attributes.

One of the biggest challenges was reconciling the way pygame handles angles for arcs (they start from the top and go clockwise) and the way angles normally work for cosine and sin. It took a lot of testing and normalizing to match up the angles, so that when a user clicked a space, the program would correctly guess which arc it clicked in based on the position of the mouse with respect to the center of the graph. A design challenge we faced was the drawing the arcs for the graph. Pygame's draw.arc method could draw filled in arcs, but the lines were all aliased because of pygame's method of drawing, and drawing multiple arcs did not help. We could've either left the graphics as they were, or we could manually draw lines radiating out from the center, which would take a lot of compute time. We ended up drawing a line radially outward for each step of .00005 radians as well as drawing pygame's draw.arc method. The graphics still aren't perfect, but smaller step size would require far longer compute time, and this way we the graphics aren't terrible.

4 Reflection

We gave ourselves a lot of freedom in this project to pursue the features that we wanted to add. We also sat down at the beginning of the project and made sure that we knew what we wanted to go into the code and the general structure of the code. That being said, we could do a better job of organizing the code and making methods for initializing classes and drawing specific things. We could work to reduce the long lists of statements required to make gui things work, as well as reduce the weird cases that we covered through multiple if statements (we should've used inheritance here) and optimizing our code.

We were also very lackluster in communication–we were often late to our meetings (doing other homework, but we should plan better). We also did not communicate very well with each other what we were going to add to the code. In this case, it worked out since we were working on different portions of the code at different blocks of time. However, in the future in different projects, this could easily cause a mess of confusion about the code.