**FINAL PROJECT REPORT**

BY COLE GAINES

TOPIC APPROVAL: DEFAULT TOPIC

**INTRODUCTION**

For my final project, I created a graphing application (the default project suggested). This application uses user inputs (x and y position edit boxes, graphing color radio buttons, line/marker style radio buttons, plot title edit box, x-axis title edit box, y-axis title edit box, reset graph and input button, graph button, and quit button) to create a graph based on said inputs.

**INITIAL IDEAS AND/OR THOUGHTS**

I had many ideas for a final project, and prior to completing the current version tried to code many of them. My first idea was to create a game meant to resemble “ping-pong,” where the user had control over a block and hit a moving projectile (a ball) back and forth between their own paddle and another. I would have included user inputs such as paddle color, ball color, possibly ball speed, etc. However, this idea soon became too difficult for my current skill level. My main difficulties with this idea involved (1) how to allow the user to move the paddle and (2) how to make the ball move based on time (I thought that a loop could maybe work, that continues until the user quits or is beat and that increases the x and or y position by + or - 1, depending on the direction). Following the “ping-pong” game idea, I still wanted to create a project about tennis. I planned to create a web-like interface that would allow users to search for and scroll through ATP rankings. This could involve user functions such as displaying ATP points or ATP ranking (one or the other), choosing different timelines of ATP rankings, etc. However, I, again, was unsure of how to complete this project. My primary problems with this project were (1) I was unsure of how to display the photos of each ATP player; (2) I was unsure of how to create a scroll through many tens of people. The final iteration of my coding process was the default project, which already provides most of the needed ideas to implement into the code; I followed the instructions to create user inputs for line style, graphing color, x and y edit boxes, title and axis title edit boxes, a graphing button, and a rest button. On top of those, I added a QUIT push button (mostly for aesthetics). I also wanted/tried to add a function to choose between plotting numbers/points and plotting functions. However, I ran out of time to do this.

**CODING PROCESS**

The coding process went smoothly; there was rarely a point during the process where I actually felt ‘stuck.’ First, I set up all of the ui controls; One error that I noticed in my coding initially, though, was that I forgot to normalize the units on every ui control. Because of this, when I wanted to change the dimensions of the figure, those ui displays whose units were not normalized became out of place, making the whole process more difficult. After creating all of the gui ui controls, I moved onto pulling actual numbers out of the x- and y-position edit boxes. Initially, I was unsure of how to complete this; my first thought (and eventual solution) involved regular expressions. However, I didn’t know how to pull numbers/doubles out of a regular expression, I only knew how to deal with a string. I tried to think of other ideas, but regular expressions seemed the most obvious to me. Therefore, in the xPositionInput and yPositionInput input callback functions, I pull the inputted string from their respective gui and assign it to a local variable. Here, (1) I check whether there are any characters other than “0-9”, “,”, “-” or a “.”; if there are characters other than those mentioned, then a message box will appear displaying an error to the user. Next, I use regular expressions to replace any character other than “0-9” or a “.” or “-” with a space. I also use regular expressions to replace any “space+” with a “space“. I then use regular expressions to check the very end and very beginning of the string for spaces and, if they are present, delete them. Finally, I split the string into cell arrays when there is a character that is not “0-9” or “.” or “-”. Next, I convert the string into a double and store the numbers in a global variable. I then moved onto graphing the coordinates that were imputed; the graphing function was made as a callback function of the “Graph” push button. In this function, I ensure that the two global variables storing the coordinates (one stores y and one stores x) have the same number of elements; if they don't, then the app displays an error. Else, the function runs plot(gui.xCoordinates, gui.yCoordinates);.

Initially, I had a lot of trouble implementing the colorAndLine function. I had 3-4 iterations of this function before arriving at my finalized solution. One attempt at a solution involved a switch expression, where the cases were the various choices, and depending on which choice was ‘true’ the line/marker style or color was based on that ‘true’ statement. However, I didn’t feel comfortable with switch expressions, and I ultimately didn’t have the knowledge or skill to make it work properly. My next iteration was simply a series of if and elseif statements to be evaluated in a non-callback function which would then be called on after the “Graph” button was pressed. However, I never was able to determine the error of this method; nevertheless, it didn’t work and wasn’t particularly elegant or concise in design. My final iteration was to create two functions, LineSelect and ColorSelect, which were callback functions of their respective buttonGroups. In each function, I evaluated the truth values of their respective choices and would assign the global plot a line/marker style based on those value statements. Each function was then called on in the graphing function (when the “Graph” push button was pressed).

Next, I created the yAxisName, titleName, xAxisName, resetButtonPush, and closeFigure functions, all of which were short, simple, and very easy to implement (they were all callback functions of their respective gui ui controls).

**DIFFICULTIES**

I’ve mentioned a few difficulties already, but I will list out the challenges I had in the coding process here.

1. I forgot to normalize the units on some of the gui ui controls; this made placement of the controls much more difficult and made alterations in the size of the figure unfeasible without more editing.
2. I had trouble implementing the line style and graphing color. At first, I simply assigned the characteristics to a global variable based on the truth values of their respective buttonGroups; I then tried to implement that variable within the actual plot call (i.e. plot(x,y,globalvariableColorAndLine). I learned that this is not a proper or functioning manner of establishing the color or line style. I then decided to establish these characteristics outside of the plot call by assigning the plot as a gui.
3. I had trouble converting the strings imputed into the x- and y-coordinate edit boxes to numbers in an array; at first, I simply tried to split the string, which split all of the numbers into a cell array (though some spaces were still present on the ends). However, when I tried to convert these to a matrix of numbers, it would clump all the numbers together to form one big number. To fix this, I used (1) regular expressions to make sure all spaces and all commas were deleted prior to dealing with the rest of the string (numbers and spaces between the strings). Then, I split the string by its spaces so that only numbers (still as strings) were put into the cell array. I then use the str2double function to convert the strings into numbers. I assigned these values to gui.xCoordinates (or gui.yCoordinates) and plotted the graph as plot(gui.xCoordinates,gui.yCoordinates).

**GUI**

Following is a list of my ideas for GUI:

1. My first thought for gui is game-development; as I aforementioned, I had intended to create a table tennis-isk game, but found it too difficult. If I were to pursue projects in the future, my first thought would involve games.
2. Next, often being used for engineering purposes, I believe that an app which allows for calculations of physics problems could be very helpful; for example, dynamics of rotational motion can be lengthy calculations. Obviously it would be difficult to manage all possible problem types in a single application, but creating an application that maybe (1) creates a figure that helps visualize the problem or (2) computes the answer to the problem could be helpful.
3. Another application could be to graph tension/strength of some material under stress and when that material is likely to break based on its inputted characteristics. Along with this could be a cost diagram based on the material type and the characteristics of the material (dimensions, etc.). This could be helpful in visualizing the safest and yet most cost effective way of solving a problem.

**CONCLUSION**

My final project is a graphing application that takes in various user inputs and outputs a graph based on those inputs. The project went fairly smoothly, but there were some difficulties including (1) I forgot to include some key information in my gui’s that would have made the process smoother, (2) I had trouble implementing line style and color into the plot call, and (3) I had trouble extrapolating individual numbers from the string imputed into y and x-coordinate edit boxes.