Final Project.

The design process on this was to determine all the pieces that were going to be visible to the user. This includes the plot area, the color/style selection radio buttons, the x and y coordinate boxes, the plot button, and the reset button. It made the most sense to have the plot take up the largest amount of space since it is going to need to be visible to see where the points are. Then it made sense to have the plot button relatively close to the x and y edit boxes. The buttons were going to take a fair amount of space, so by the side made the most sense.

The coding process was a little more in depth. The final approach that was determined was to start by getting all the pieces on a figure and laid out properly. This was a large trail and error by setting positions at an educated guess and pulling the figure up to determine if the position was going to be appropriate. It was a dancing act, if you adjusted one part too much, then another part would need readjusted. Once that was all together and looking like something that would be functional, then it came down to determining the remaining functions. I had started with the reset button, so there would be an easy feel-good part of the coding to keep the motivation and have a small easy win. I started by hard coding in some plotting points so the plot would start with points on it. Then, pressed the reset button and observed what had happened. After the first run, I had realized that I only had reset the plot, but not the x and y edit boxes. After a quick tweak, the plot was reset too completely blank.

After that code was complete, it was time to determine the best way to get the line mode set. The small obstacle was the way I had first tried to get it to work was by running two separate codes to get, for example, ‘rx’. It didn’t take long until I had realized that it was easier to run an if statement inside another if statement to produce the ‘rx’ with one simple function. It was very lengthy, but it is functional. The first time I had tried this, I had forgotten to use the statement gui.r1.Value, and had only used gui.r1 so it was never going to work properly. When a button is selected, the value is a logical 1. When the button is not selected, it produces a 0. This makes evaluating and using it in an if statement possible. Now it is possible to compare it to a 1 (gui.r1.value == 1). With that, when it is one, it will go into that if loop and evaluate the second part. If it is a 0 or not true, then it moves to the next step of the current if statement.

Now that we had a visible figure, and the color/shape function is working, all that was left is the actual plotting function itself. This one took the longest because I was unaware of how to pull the information out of the edit box after the user had entered information. After a long time of digging and research, I finally found out that the edit box did in fact produce a string value. Once that was known, it was easy enough to pull the string out of the box, turn it into numbers, and make it something to be able to plot with. There was going to be issues if the x points were not equal to the y points. To take care of this, it was easy enough to tell the function that if the length of x is the same as the length of y, then go ahead and plot. But if they are not equal, don’t crash, rather tell the user that there is something wrong with the points and let them adjust and try again.

After all that coding was complete, it was time to attack and try to make it crash with user inputs that might be entered. This was a series of different length points, using commas, spaces, square brackets, etc., and using different line modifications and resetting. After all those tests were complete, it was determined that the code is working for what it was designed for.

The major difficulties I had during this project was that I was unaware of how to pull data from edit boxes and knowing the proper flow the functions should have to interact with each other properly. At first, I had tried to use the .Value option like I had used in the button selection portion of this coding. That was never going to work. I found out there was a property of the edit box that held the string of values within it. From there it was just a matter of turning that string into numbers so that the plot function could work. Than to figure out the flow of the functions, at first, I was going to have the color selection have its own callback, but I found out, when ever you change the button, it would still update if it was written the way I had written it. I used the button selection function from the plot now function to determine the line style and color. It does appear to be working properly at this time.

Using GUI’s is a very realistic part of engineering because this is another tool to be able to clearly communicate what an engineer might design. It is one thing to be able to tell somebody all the specifications, but its another to show them. This could also be useful with the proper coding, to help optimize designs. If you were to hard code a function such as dimensional lumber needed for a span on roof/floor. Than a user can simply type in parts such as weight load, total span, type of material and let the program determine what size of dimensional lumber would be needed. This is still doable through the standard framing chart, but it would be much faster if there was a program all set up and ready to use to determine these requirements.

This project was not approved by an LA, but it was the default option supplied with the instructions for this homework.