Criterion C: Development

Basic GUI setup

It was determined that using two JPanels on one JFrame would be easiest for replicating the GUI mockup that was created. I initially believed that several JPanels were needed, but further investigation narrowed it down to two. One JPanel for the graph, and the other for the rest of the graphical items. After that, I began researching different layout managers in java in order to find which one I should use for each JPanel. Further investigation however showed that having no layout manager would be easiest and allow me to create a custom layout which would allow me to best match the GUI mockup. With all that figured out, it was relatively simple to create the two JPanels and complete the GUI setup. At this point, the application had no functionality other than showing up on the screen.

Calculation methods

In the application, we need to calculate various properties of a projectile such as range and maximum height. Furthermore, to best solve the solution criteria, the application needs to calculate the various properties of a projectile from several different combinations of user input. For example, the user could input initial velocity, the angle, and the height and solve for the rest, or they could input the angle, the time of flight and the range to solve for the rest. This was solved by writing a series of computations based on several different combinations of input. It was then arranged in a large if-else block which determined which combination was inputed by the user.

Data collection from user

The next step would be to take information about the projectile from the user in order calculate and return the answers to the user. After some research, it was decided that JTextField's would best solve this process. JTextField's would allow the user to input information, and that information could be received by the JTextField and be processed. Six JTextField's were added, each representing the six variables that deal with projectile motion. A JButton was also added, with the thought that the user would press this button after they input their information into the text fields, so the application would know when to take the data from the text fields and compute it. Further logic for calculation was also added, as the JTextField's hold strings, and we need doubles to do math. Also, the text fields were left empty as the user could input various combinations of known variables, making it necessary to add additional calculation logic catching the NumberFormatException and assigning 0 values to the variables that were not inputed.

Displaying results

After the user's input is calculated, the application needs to display these results. I considered sending the results to the JTextField's and displaying the results through them or generating another section of text on the GUI that displayed the results calculated from the user's input. The second option was chosen, as it would make the application less confusing and increase the clarity of what is going on to the user. I created more code to display the results on the screen.

Creating and displaying the graph of the projectile's motion

In order to complete the success criteria, the application also needed to create and display a graph of the projectile's motion. As a graph of the projectile's motion would lie in the first quadrant, this represented a problem as I could not figure out how to mathematically draw a graph with the origin of the JPanel residing in the top left corner. With research, I discovered that I could move the origin of the JPanel by translating it and then changing the scale. With that problem solved, I began to figure out how to draw the graph. The solution I came up with was to calculate and fill an integer array full of points that represented the projectiles position. After that, I would draw, using Graphics2D, circles at each point. This would result in a series of circles that would form a line or a path of the projectile's motion, solving this problem.