André Gras
CS5001
180028855

GUI Project Report

I used Swing and extended a JPanel class to be my main class. In the main I create an original frame and set the panel inside the frame, so I can layout using borderlayout for the buttons and gridlayout for my panel. In the constructor of the panel, I calculate the Mandelbrot set. I do this because I create a new panel for the superimposed image upon viewing and therefore calculate a new set with which to view the zoom panel. The zoom panel has a button that allows a user to accept the change. The frame is disposed when the user clicks on the exit button in the top right, which allows for cancelling upon viewing the estimate. The undo and redo buttons go forward and backwards using a hashmap that stores the original set twice and modifies a copy to “keep” the zoom, while also keeping a zoom copy. Zoom is done by taking the mouse X and Y upon press and then upon release to create a rectangular area for zooming. The minimum and maximum real and imaginary values are scaled by dividing the X and Y mouse values from the frame size and then multiplying by the range (max-min). These values are kept consistent with the panning as panning is allowed using the arrow keys. One must focus on the graph by clicking on it first before panning, as the focus will be first on the panning scale. One can also change the maximum iterations at the button prior to zooming or panning. One can undo and redo the most recent zoom; however, pan is not controlled by undo and redo buttons. I decided this because one can undo a pan by simply pressing the opposite directional arrow key, which is more intuitive than clicking a large button. I also implemented the color based on a stackoverflow response on how to implement dynamic color in Swing.

The overall design of my program is poor, as it does not allow for proper modulation and controlling of events. It does not make proper use of various classes to interact with one another and relies on the ever-unreliable global variables, which created many implementation issues from poor memory management (if I had written this in C++, it probably would not have worked). The undo function brings the zoom back to the original set and the redo function can bring the state back to the latest zoom from whichever undo, as I believe this creates a better flow for manipulating the Mandelbrot set.