Matthew Staehely

CSC 143 Offenback

1. During my design phase, I drew a diagram of what should be viewing, what should be controlling, and my model, and how I wanted the three of them to communicate. Late in the project I realized I had made a mistake with regards to my controller/model communication and the design stage made this very easy to fix.
2. In my particular program, when an end-user moves a shape, first the controller’s mouse listener asks the viewer where the mouse is on its pane. With this information it, then controller then asks the model if there is a point under the cursor and if the answer is yes, the controller lets the model know that the user wants to drag a shape. With the go-ahead from the model, the controller then feeds a series of x-y coordinates in to the model who continuously updates the shape’s location in its own list, and then lets the graphic viewer know that things have changed, and the panel needs a repaint. When the controller hears the user release the mouse button, it tells the model (which tracks that event), and then the model tells the viewer to now update the JTextArea as well.
3. First I hooked up my two viewers and populated the frame with an arbitrary list of shapes. The original viewer was the terminal, so this made it easy to read and check my results. Once that was done and I got the controller working, I tested by creating, moving, and selecting shapes and took note of what wasn’t working the way I wanted it to. Debugging was done one issue at a time, as I have learned from past mistakes not to try to fix several bugs at once if it can be avoided. I got a lot of null pointer exceptions, mostly due to bad syntax. To my knowledge, the code is to spec.
4. Features I added: A JTextArea and buttons to delete shapes and change their color. I didn’t have enough time to add an actual color picker, as it took me a long time to figure out why my mouse motion listener wasn’t working, and then hooking up the JTextArea was also a non-trivial process. I designed the extra buttons using the same process as the other buttons, and their action commands were used by the model in a fashion similar to a switch statement. For the JTextArea, I wanted its output to have a particular format, and it took a while to first figure out how to do that, and then how to force it to scroll as it output information. I ended up putting that in the class which was originally used as a viewer for the terminal, a functionality which I removed as it was redundant.
5. There was a lot of trial and error in this project, and much reading of the API and stackoverflow. I learned a good deal about how to use swing components, and graphical output ended up being not nearly as daunting as I had thought (though still complicated). It was definitely worth the effort! Even if I don’t end up working in a graphical context often, there was a lot to be learned about coupling and cohesion from this project- it got easy to blur lines between each of the three aspects of MVC (especially MC and MV).