Andrew Schoolnick
Interactive Media Development

Processing Paint Application

Navigation

- There are a total of 9 different buttons that the user can interact with
  - *Paint Brush*
    - Contains stroke size attributes
  - *Eraser*
    - Contains stroke size attributes
  - ■ *Square Tool*
    - Contains shape size attributes
  - ● *Circle Tool*
    - Contains shape size attributes
  - T *Text Tool*
    - Contains font size attributes
  - *Color Picker Tool*
    - Contains Color Picker wheel
  - > *Save*
    - By pressing the save button a save prompt is opened up that supports several different file types for output such as png, jpg, gif, and tiff
  - > *Load*
    - By pressing the load button a prompt will open up asking the user to select a file to load.
  - > *Clear*
    - The background will be cleared and the image will reset to its original state
- Dragging the mouse across the screen will cause the currently selected tool to pain across the screen
- Pressing down on the mouse will stamp the current tool on the screen
- When the Text Tool is selected the user can press on an area of the screen to stamp the text box at and can begin typing their message

Design

- As a New Median that likes to focus on web design more than anything else, a flat UI is elegant enough to catch the eye, but at the same time simple enough to use. One of the main goals of the program was to keep it as clean as possible which meant hiding certain menus, adding hovering features to both the mouse and objects being hovered over, and adding a scaled along the sides for precision measurements within the program to align certain objects. The only tool needed is the mouse so anyone can easily access the program without fumbling around looking for what keys to press.

Functionality

- Design aside the content under the hood is what really matters. There are six separate tools that the user can choose from to draw something on the screen. Choosing each of the tools is accessible through the buttons in the top part of the menu, the mouse will hover over the tool and will swap the current tool properties relative to the tool being used and the tool's box will highlight. For each tool there is a simple slider used to adjust the size of the tool, and a color wheel for choosing the color to be used. By hiding some of the tools and keeping it simplistic the application resembles a responsive design layout that can work at just about any resolution. As far as the tools are concerned, the paint brush is a simple line being drawn from the previous mouse position to the current mouse position when pressed and the eraser works in the exact same way however the fill color is set to the canvas background. The rectangle tool and ellipse tool work in pretty much the same way that they can be stamped, moved along the screen and have their sizes be changed, but the text tool allows the user to select a location on the screen and as soon as they have selected the location they can begin typing away.

  As for the saving functionality, a window will pop up allowing the user to choose a filename and location to save to, but no fret if the file extension is input incorrectly because the program handles those types of errors. For example, if the user accidentally saves the file as ".pn" and not ".png" it will automatically adjust the filename and if an error does occur it will catch it and just default to, "image.png." Loading works pretty much the same way, except the user just selects the file to load and it will replace the image currently being displayed on the canvas to the one that was loaded in.

What Could Have Been Done Better

- There are several things I wish I could have had the opportunity to add to the program to increase usability, but due to Processing's system of saving the display's images it made things a lot more difficult to add some more UI like I wanted to.
- One of the features I had in a previous version was a set of lines originating from the mouse that extended to the scale on the side of the screen so the user could precisely see where they were on the screen. I managed to get the system to work via saving an image to a temporary folder then refreshing the background so that the lines wouldn't overwrite the image behind it, however once the user would drag the mouse the method that I used for making the lines would crash the program. However, the upside is that I am glad I gave up on the method because as I was using it I realized each time I'd save the background the quality of the image would distort and just wasn't very pleasant to the eye.
- A scaling rectangle and ellipse was another idea I originally had that I was going to implement into the system using the same system as stated above. After struggling to get it to work, I turned to a friend whom in which shockingly managed to get the system to work, after asking for his assistance I managed to create my own system for doing what he had, but I was unable to replicate it using my method and had the same issue as I had with the lines from the mouse

where the program would crash on my end or get extremely slow after dragging the mouse so it was an idea I personally had to scrap and did an alternative approach to adding shapes to the screen.

- If I had more time I would have loved to add more sliders as well as filters and a selection tool for the shapes that I managed to get working back when we were doing C#

Conclusion

- All in all I am proud of how the paint program turned out, there are some things that I wish I could add and probably will continue to build upon for my portfolio after it has been submitted to the dropbox, such as adding the filtering and selection tools and possibly changing the program's color palette to fit the design of my portfolio, but other than that it came out well and there's a lot of method that I could easily reuse in future programs if I decided to do so.