Roxanne Low

Com 110 Fall 2015

Professor Chung

Final Project – ImageShop

**Problem Specification and Design**

1. Project Statement

* My final project is called ImageShop, which will perform image processing. It takes in user’s image and allow user to apply filter of their choice and stickers. Eventually, user can save the image that they have edited with a filename of their choice. This program does a very similar process as ‘Instagram’ and ‘Snapchat’.

1. 1: Project Components   
   (Classes & Methods including parameters and return types)

* In order to make everything neat and tidy, instead of having the functions of the various pages in my main module, they have been place into a class that can be called on in my main module. The main components of this project include:
* ImageShop.py,
* pageclass.py (pages),
* processorclass.py,
* buttonclass.py
* graphics.py
* processorclass.py

contains a class called ImgProcessor to carry out all the image editing processes. In the processor class, the Image and ImageFilter module from PIL is imported because that is where method is called upon. Similarly, graphics is imported because some of the method performs drawing gif images in graphical window.

To initiate the class, the graphical window and the name of the file that user wishes to process are passed in as parameter. All the instance variables(IV) are listed down in the constructor method although not all of them get assigned something. Thus, methods have to be called on in particular order, which can be seen in pageclass.py. For example, when the method applyFilter is called, the filter type and the user’s new name choice are passed in. User’s name choice is then assigned to (IV) self.namechoice. Once this IV is assigned, when the method savegif() is called, there is no parameter required because self.namechoice can be used to indicate the user’s new filename.

For methods such as setupsticker() and transformsticker(), the parameter that is passed in is the user’s choice. The methods will take in the choice and open up the images from the folder and assign it to the IV self.userstickerjpg which is easily accessible when method such as pastesticker() is called. The only parameters needed are the x and y position of where the user wants the sticker to be on the image.

* pageclass.py

Seeing that the code to perform image processing is called on in Page class, the processorclass is imported into pages class. On top of that, buttonclass is imported because the methods in this class, when setting up the interface of this program will need to create buttons to get user’s click.

Before the start of the class, few methods were written to create Text objects (from graphics) with specific features. This way, instead of editing the text individually in the pages method, the parameters passed in will indicate the specifications of the Text. The methods will set the size, font, style and color accordingly and draw it in the window (a parameter passed into the method).

When setting up the Page class, the graphical window is passed in as a parameter so that it can be assigned to an IV called self.gwin. By doing this, when calling upon the methods in the class, the method will not need to take in a graphical window as a parameter instead they just need to use the IV self.gwin. Moreover, another IV is created called self.imgEdit but it takes on ‘None’ initially. Only when the user’s input of the image filename is saved, will this ImgProcessor object be assigned to self.imgEdit. Then, self.imgEdit can be called easily called between methods in Pages. The methods within this class do not need any parameters passed in because the parameter is already first passed in when the Pages object is called (pages=Pages(win)).

* buttonclass.py

The original Button class was used. However, there is a small improvements made to the class whereby the label of the button can take specific font size. The size of the font is passed in as a parameter.

* graphics.py

The original graphics module is used with minor improvements made. The minimum and maximum boundaries of the font size for the Text object was changed to allow smaller or bigger font sizes. Also, additional font types are added into the option so that the when the texts with features are created, they can be created with a customize font type.

* ImageShop.py

This module contains only the essential main functions, which are calling upon the various pages from the page class module. A while loop is implemented to allow user to go back and forth between the two pages (inputPage() and ProcessPage()). This module imports graphics and pageclass. It also creates the graphical window that will be passed into the Pages object.

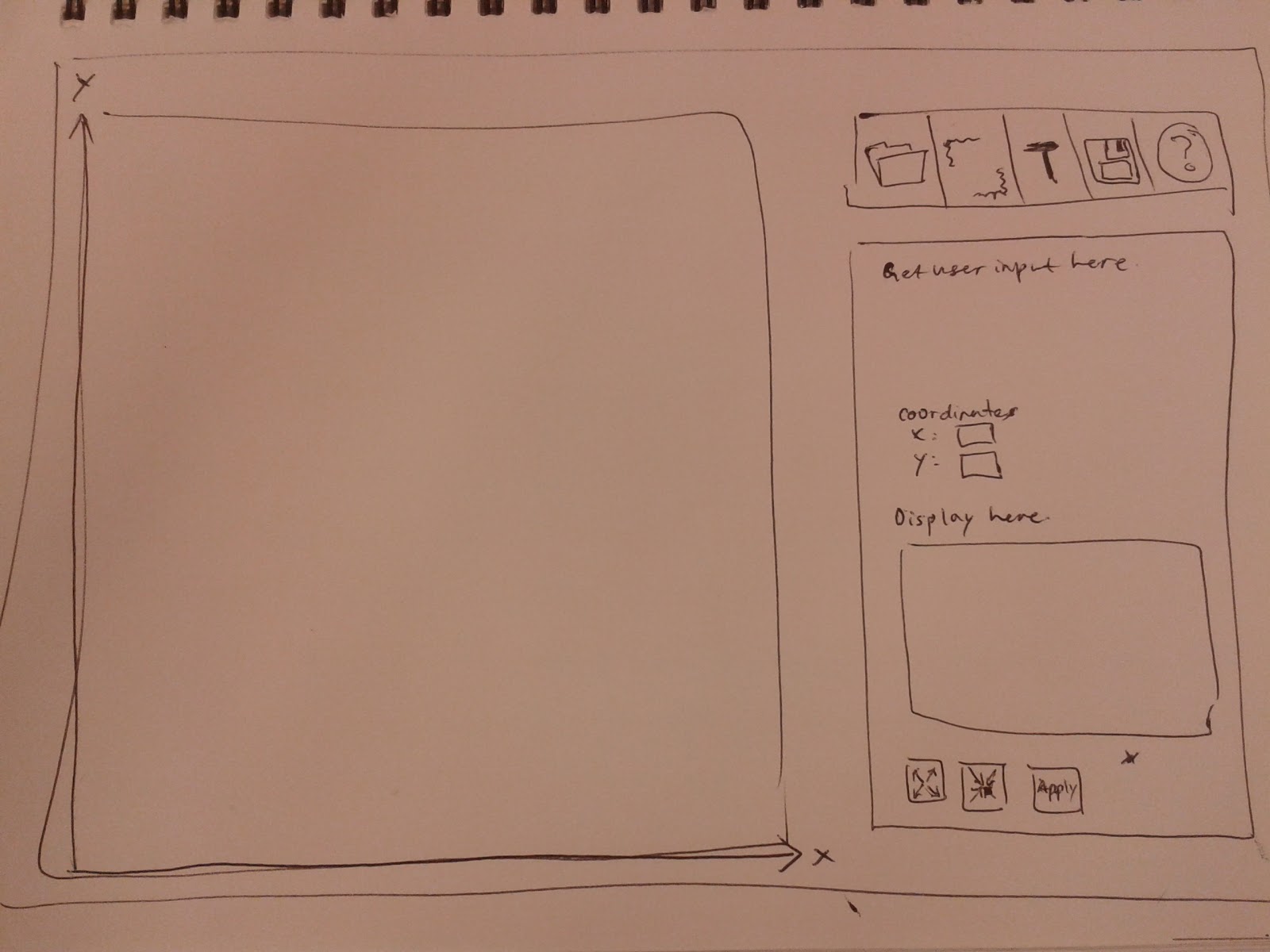
2: Improvements made from original design (due to limitations)  
\*Please refer to google doc for detailed initial proposal of project. Link provided below.   
(<https://docs.google.com/a/conncoll.edu/document/d/1o5IIBKYRGrDEU7vWEdFUyiQrM5oRIbN0lIm4xunVdR8/edit?usp=sharing>)

* Applying filter function was made separated from the page where user can put on stickers on image
* Instead of having a buttons to allow user to change the size of the stickers, program takes in user’s indication of sticker size. Then, a method is called to transform the size of the sticker.
* Remove the method of inserting text because I am unable to save the edited image without going through pixel by pixel.

1. Testing and Modification

* Lots of testing was done to make sure that the program ran flawlessly, checking for the right inputs, moving back and forth between pages, checking the right image size, and coordinates for stickers.
* Modification1: to have more stickers saved within the program so that user has more stickers to choose from
* Modification2: allow user to insert tests and save it on the image as new edit
* Modification3: allow user to go back one step (when applying sticker) rather than having to refresh and start all over again. At the moment, user have to make sure that they type in the coordinates properly, otherwise, one mistake will require them to refresh and start all over again.

\*\*\* To run this program, Python Image Library needs to be installed into python. I’m not too sure how exactly I got PIL installed in my python. It was Tim, the TA who helped me with it. If I remembered correctly, he made me go onto python shell and typed something like ‘install…’ then it ran and installed the PIL in my python. There is no separate module to download like the one we use (graphics.py).



*Initial plan for the interface of the program.*