Connor Gillis

Profosser Aries

Abstract

The purpose of this Java application is to process digital images. Glitch art is defined as the practice of using digital or analog errors for aesthetic purposes by either corrupting digital data or physically manipulating electronic devices. The overall objective of the application will be to digitally process images to produce glitch art. Users will select an input file, configure their settings, and run the manipulation processes. Then the resulting image will be created in a directory of the user's choice.

Introduction

I created this project in oder to simplify the means by which users could create this type of art, with little or no technical know-how. My research into this type of image manipulation reminded me of my love for design and image editing. These two factors along with my computer science knowledge led me to pursue this project.

System Description

The system created revolves around 4 java class files: Profile, Process, Processes, and Final. The main executable method is found in the Final class. The role of this Final class is to first create a GUI (Graphical User Interface). This GUI accepts multiple user parameters such as an input file, output file, as well as the users' settings. Before creating an instance of the GUI itself, the Final class establishes the elements which must be represented on the GUI. The

elements are as follows: 3 buttons, 12 sliders, and a progress bar. These elements were imported from the Java Swing library.

The main method of the Final class creates various listeners which wait for an action to be performed. The first of these listeners waits for an action to be performed on the first button, selectSourceButton. When the button experiences an event, in this case a click, a file chooser dialog box is opened. The user may then select the image file which they would like to alter. The second button, when clicked, opens another file chooser. One difference between this file chooser and the second is that the second accepts only directories. This is useful because a user, rather than selecting a file to have the image outputted to, they can select a directory location where a completely new file is created.

As previously stated the sliders control user settings. These settings affect various effects. The 6 effects are: greyscale, sepia, negative, red, green, and blue. Each effect uses two sliders. One controls the amount of each effect, another controls its respective size. When mentioning size it is important to understand what the variable size represents; In my project effects are applied upon the image in a looped fashion. In turn, for each loop, 1 cluster of effects are applied. Size and amount variables affect both the cluster length and amount of loops. Another element of each cluster, or group of pixels to be affected, is the cluster origin points. These points are random values created in the range between 0 and the images overall width and height. In summary, each cluster has an x-origin point, a y-origin, point and both an x and y cluster length (determined by size slider). The pixels which fall in the range of the cluster are affected by one of the effects, and as previously stated the amount slider affects how may clusters are created and manipulated per effect.

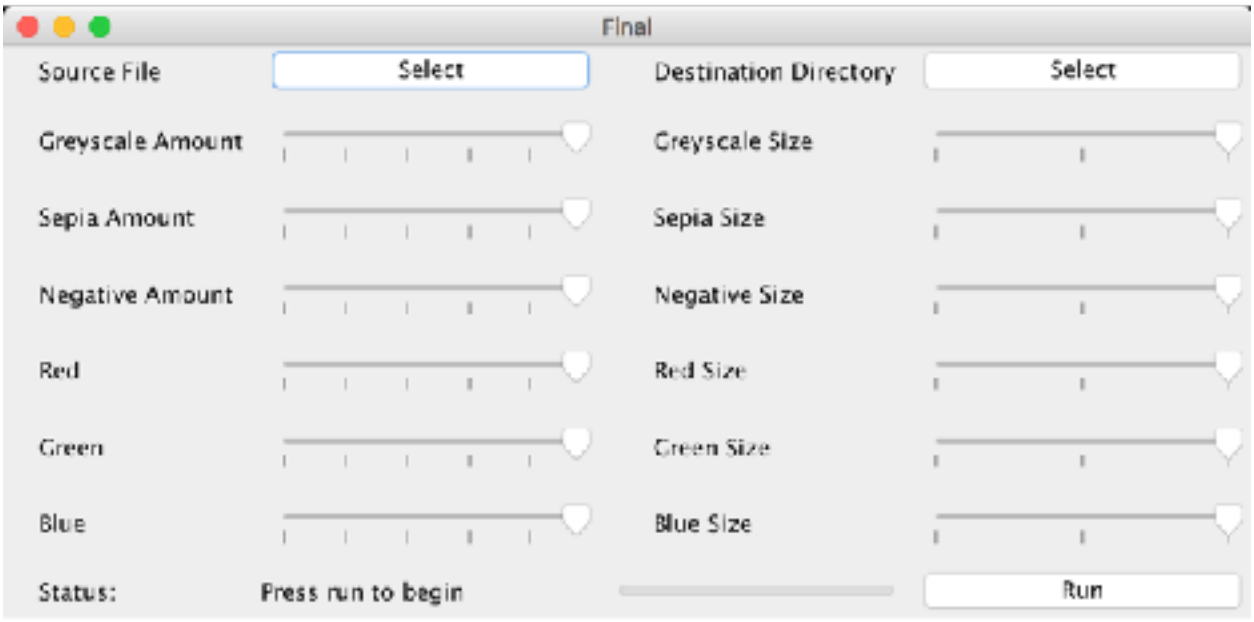
These user parameters are saved as an instance of the Profile object. This profile object contains attributes which describe the users preferences, making them accessible to other portions of the program. Upon running, this Profile object is then passed as an argument to the method Create, part of the Process class. The process class then takes over, performing manipulations across the image object, effect by effect, cluster by cluster. The final output image is then rendered and passed back through FileIO as a user accessible file.

In order to fully illustrate the manipulation of these effects, take the greyscale process for instance. The image input file and output directory are selected by the user in the GUI. Upon run the input image is loaded as a file via FileIO class. Next the BufferedImage class creates an image instance based on this file input. Next the Process class performs the greyscale process. The amount greyscale slider determines the amount of clusters. For each of these clusters the two origin points are created as random numbers between 0 and the value established by the methods getWidth and getHeight via BufferedImage class. The cluster length values are determined by the size slider. If the size is 1, then maximum cluster size is proportional to 1/3 of the image height or width. If the size was 2 the maximum cluster size would be proportional to 1/2 of the image height or width. For each effect the pixels in the cluster area are manipulated. Each effect uses different methods of manipulation but generally alter each pixels RGB values in order to achieve the desired effect.

During testing a case in which a cluster would not run became apparent. The error ArrayCoordinateOutOfBounds was raised when a cluster length added to the origin point fell out of the images bounds. In this case the error was handled by creating new cluster lengths, however in this case they would equal the height or width subtracted by the origin point(s).

For a further explanation of the system, a **sequence diagram** and corresponding text can be found in this directory.

User Manual



1. User selects a source file, this input file should be named ‘input’ and should be in JPG format.
2. User selects a destination directory. This folder is where the resulting image will be stored.
3. User should configure their settings using the sliders.
4. When complete, users should press the Run button
5. When the progress bar is complete users should refer to their selected destination directory to retrieve their result image.
6. Note: Each run, even with the same configurations, may be different as all values are randomized. The configurations only represent a range in which the cluster length values may fall.

References

"How to Read and Write Image File in Java." *DY Classroom*. N.p., n.d. Web. 08 May 2017. <https://www.dyclassroom.com/image-processing-project/how-to-read-and-write-image-file-in-java>.

LIANG, Y. DANIEL. *INTRODUCTION TO JAVA PROGRAMMING*. S.l.: PRENTICE HALL, 2017. Print.