Image-Editor

A JavaFX Application Project

GitHub Repository:
https://github.com/arthurW1935/Image-Editor

Created by:
Sourashis Sarkar

Table of Contents

Introduction

Overview

The JavaFX Image Editor is a Java application designed for editing and manipulating images. It provides a user-friendly graphical interface for performing various image editing operations.

Project Purpose

This project aims to simplify user image editing tasks by offering an easy-to-use interface that allows them to grayscale, flip, rotate, blur, and adjust brightness.

Features

Image Editing: Grayscale, flip, rotate, blur, and adjust brightness.
User-Friendly Interface: Intuitive GUI for effortless image editing.
Cross-Platform: Works on Windows, macOS, and Linux.

Project Structure

Application Class: Entry point of the application.
ImageEditor Class: Contains methods for image manipulation.
Resources: Stores application resources like icons and stylesheets.

Getting Started

Prerequisites

Before using the JavaFX Image Editor, make sure you have the following installed:

- Java Development Kit (JDK) 8 or later
- JavaFX SDK (included with JDK 8 and 11, or separate installation required for newer versions)

Installation

Clone the project repository to your local machine.
Build and run the application using the provided instructions.

Running the Application

Launch the JavaFX Image Editor using the provided instructions in the Installation section.

User Guide

Loading an Image

- Click the "File" menu and select "Open."
- Choose an image file (e.g., PNG, JPG) to open and edit.

Saving an Edited Image

- Click the "File" menu and select "Save."
- Choose a location to save the edited image.

Image Editing Features

- Grayscale
  Click the "GrayScale" button to convert the image to grayscale.

- Horizontal Flip
  Click the "Horizontal Flip" button to flip the image horizontally.

- Vertical Flip
  Click the "Vertical Flip" button to flip the image vertically.

- Rotate Clockwise
  Click the "Rotate Clockwise" button to rotate the image 90 degrees clockwise.

- Rotate Anticlockwise
  Click the "Rotate Anticlockwise" button to rotate the image 90 degrees anticlockwise.

- Adjusting Brightness
  Use the "Brightness" slider to adjust the image brightness. Click the "Apply Brightness" button to apply the brightness adjustment.

- Applying Pixel Blur
  Use the "Pixel Blurr" slider to adjust the pixel blur intensity. Click the "Apply Pixel Blurr" button to apply the pixel blur effect.

Exiting the Application

Close the application window to exit. Any unsaved changes will be lost.

User Interface:

The user interface was built using JavaFX, a cross-platform module for building Java Applications.

- BorderPane was used as the root layout.
- The menubar was made to be set on top.
- The menubar consists of a File Menu, which on click shows an option pane to Open And Save Images.
- The toolbox was made to be set on the left.
- Toolbox has the option of GrayScale, Flip Horizontal, Flip Vertical, Rotate Clockwise, Rotate Anticlockwise, Pixel Blur, and Brightness.
- For Pixel Blur and Brightness, there is a slider provided.
- The image is shown in the center.
- The minimum height and width of the app are set to be 800 and 1400.

Basic Program Workflow

The program is started when we run the main function of the Application class.

The app window gets opened. When we open an image file, the file gets stored in the Resources folder as output.jpg. The output.jpg image is displayed all the time, and any edits in the image are applied to the output image. When the user saves the image, the output image is copied to the required destination. For the brightness and pixel blur, the user needs to adjust the slider according to the requirement and then press the apply button.

The user can apply any function in any order. All the algorithms will be performed on the output.jpg image file, and it will be rewritten.

Supported Operations

1. GrayScale

   The grayscale button allows the user to change the image into a grayscale image.

   Methods:

   `Application.grayscale() method:`
   This function gets called when the grayscale button is executed. It loads the image and passes it to the grayscale method from the ImageEditor class.

   `ImageEditor.grayscale() method:`
   Parameters: BufferedImage object

   This method takes a BufferedImage object, loops through it, scans the RGB values of every pixel, averages it, and assigns the average value in every of the Red, Green, and Blue values in the new image, and returns the image.

   Old pixel: **`(R,G,B)`**
   Average (avg): **`(R+G+B)/3`**
   New pixel: **`(avg, avg, avg)`**

2. Horizontal Flip

   The Horizontal Flip button helps the user to flip the image horizontally (mirror image along the y-axis)

   Methods:

   `Application.horizontalFlip() method:`
   This method gets called when the Horizontal Flip button is executed. It loads the image and passes it to flipHorizontally method from the ImageEditor class.

   `ImageEditor.flipHorizontally() method:`
   Parameters: BufferedImage object

   This method takes a BufferedImage object, loops through every row of the image matrix, inverses the row, and returns the image. It achieves the inversion by swapping the jth and (n-j-1)th element of the row.

   Simple 2D-array implementation:

```
swap(arr[i][j], arr[i][no_of_columns-j-1])
```

3. Vertical Flip

The Vertical Flip button helps the user to flip the image vertically (mirror image along the x-axis)

Methods:

`Application.verticalFlip() method:`

This method gets called when the Horizontal Flip button is executed. It loads the image and passes it to flipVertically method from the ImageEditor class.

`ImageEditor.flipVertically() method:`

Parameters: BufferedImage object

This method takes a BufferedImage object, loops through every column of the image matrix, inverses the column, and returns the image. It achieves the inversion by swapping the ith and (n-i-1)th element of each column.

Simple 2D-array implementation:

```
swap(arr[i][j], arr[no_of_rows-i-1][j])
```

4. Rotate Clockwise

The Rotate Clockwise button helps the user to rotate the image clockwise by 90 degrees.

Methods:

`Application.rotateClockwise() method:`
This method gets called when the Rotate Clockwise button is executed. It loads the image and passes it to rotateClockwise method from the ImageEditor class.

`ImageEditor.rotateClockwise() method:`
Parameters: BufferedImage object

This method takes a BufferedImage object, loops through the image, swaps the pixels according to the formula, and returns the image.

Formula (derived from observation)
A simple 2D-Array Implementation

```
newImage[j][i] = oldImage[height-i-1][j]
```

5. Rotate Anticlockwise

The Rotate Anticlockwise button helps the user to rotate the image anticlockwise by 90 degrees.

Methods:

`Application.rotateAnticlockwise() method:`
This method gets called when the Rotate Anticlockwise button is executed. It loads the image and passes it to rotateAnticlockwise method from the ImageEditor class.

`ImageEditor.rotateCounterClockwise() method:`
Parameters: BufferedImage object

This method takes a BufferedImage object, loops through the image, swaps the pixels according to the formula, and returns the image.

Formula (derived from observation)
A simple 2D-Array Implementation
`newImage[j][i] = oldImage[i][j]`

6. Brightness

This method allows the user to increase or decrease the brightness of an image.

Methods:

`Application.brightness() method:`

This method gets called when the Apply Brightness button is executed. It takes the value from the slider, loads the image and passes it to brightness method from the ImageEditor class.

`ImageEditor.brightness() method:`

Parameters: BufferedImage object, Integer value [-100,100]

This method takes a BufferedImage object, loops through the pixels, and increases the RGB values of each of the pixels using the value according to a formula.
Formula (derived from observation)
OldPixel: **`(R,G,B)`**
newR = **`R(1+value/100)`**
newG = **`G(1+value/100)`**
newB = **`B(1+value/100)`**
NewPixel = **`(newR, newG, newB)`**

Utility Methods

1. **`Application.setImage() method:`**
   Parameters: File object

   This method is used to set the dimensions of the image that needs to be displayed. If the dimension of the image is greater than the threshold, it reduces the size maintaining the aspect ratio, so as to display the image in the Application properly. Here, the threshold height is set to 700 and the threshold weight is set to 1000.

2. **`Application.showImage() method:`**

   This method is called after every edit, to make the changes visible. It opens the output.jpg file, passes it to setImage method to resize it accordingly, and then displays the Image in the Application.

3. **`Application.openFile() method:`**

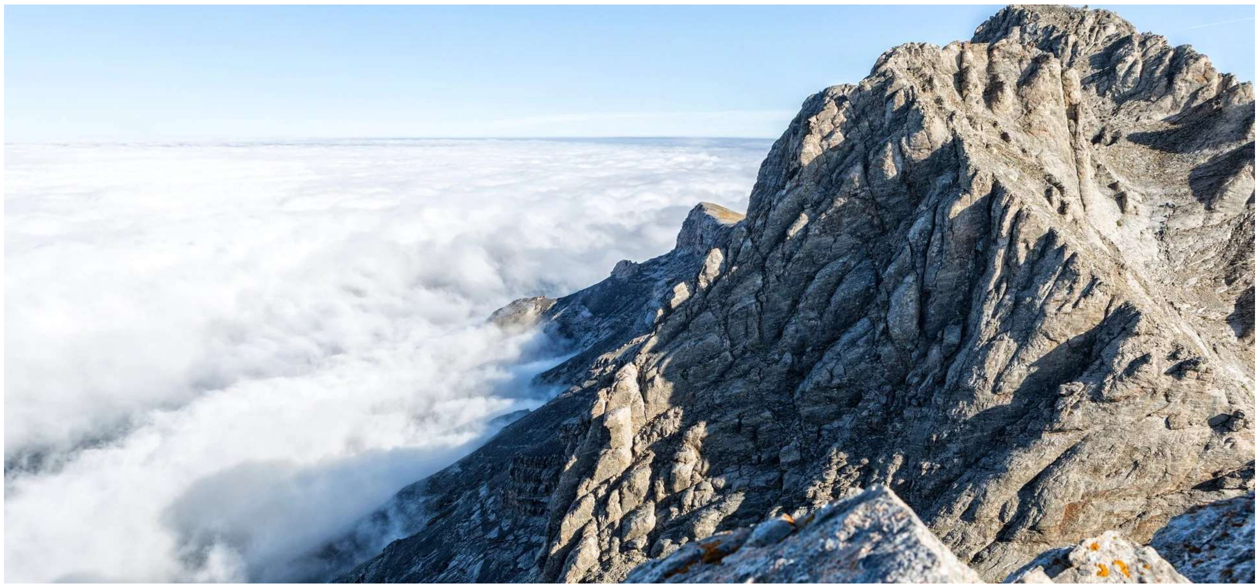
   This method is invoked when a user wants to open a new Image to edit. It opens the File Chooser Dialog Box, for the user to navigate and choose the file of their choice. After the user has selected the file, the file gets copied to output.jpg. Then the showImage method is invoked to display the image.

4. **`Application.saveFile() method:`**

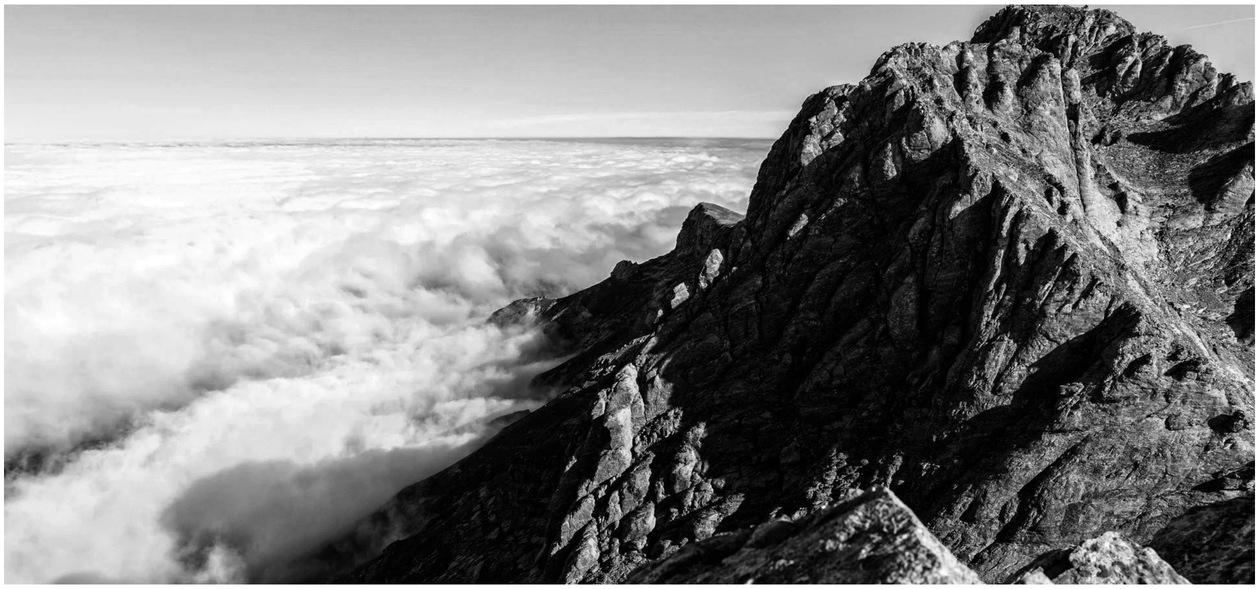
   This method is invoked when a user wants to save the edited Image. It opens the File Chooser Dialog Box, for the user to navigate to the folder and name the file of their choice. After the user has named the file, the output.jpg file gets copied to the file of their choice.

Example Images

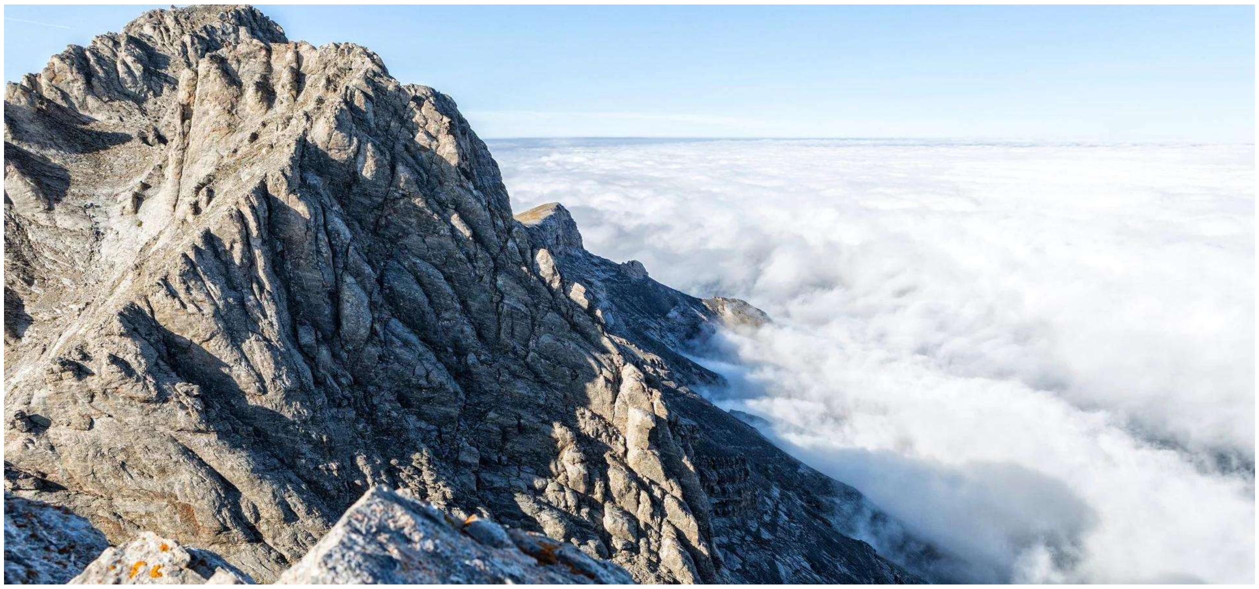
Original Image:



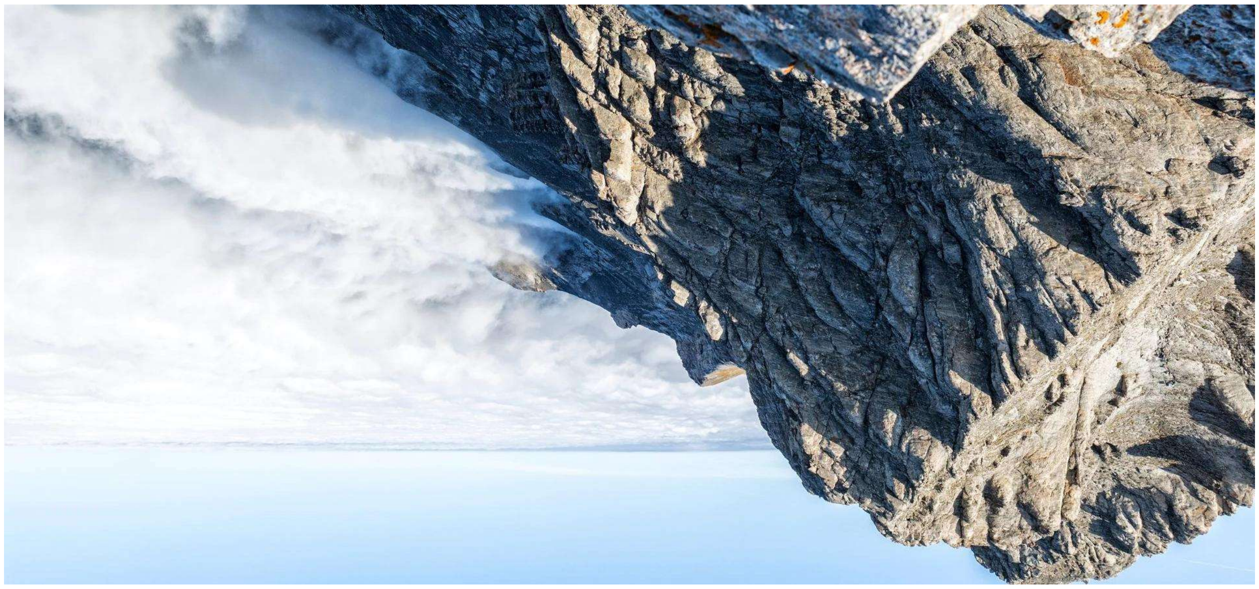
GrayScale Image:



Horizontal Flip:

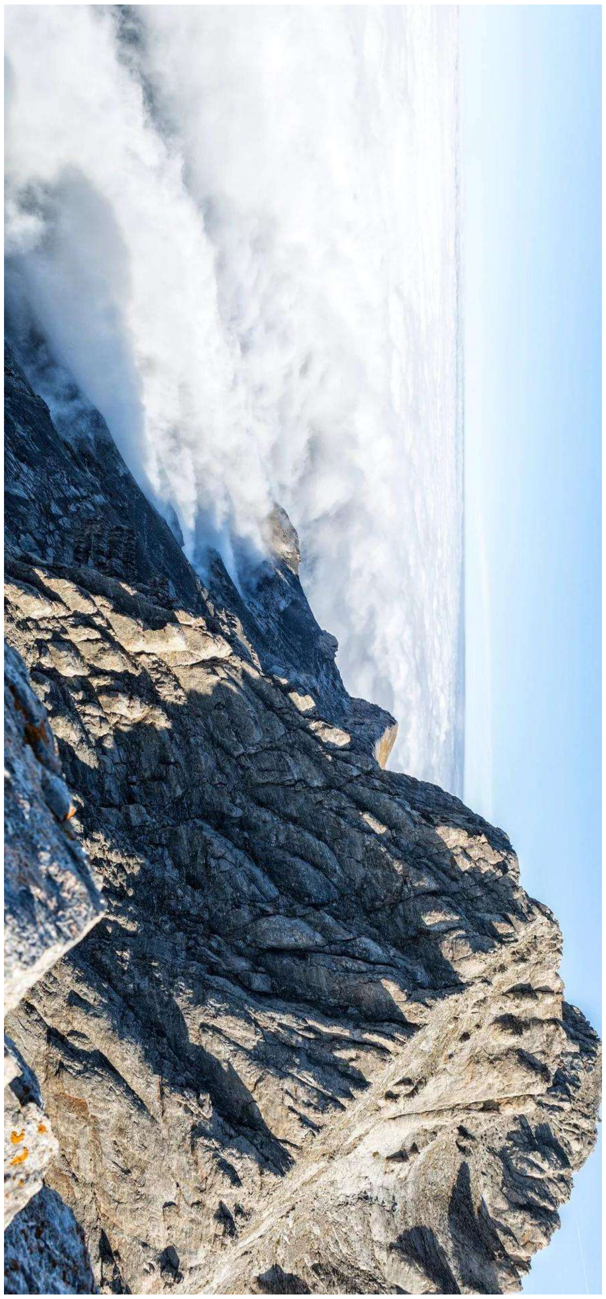


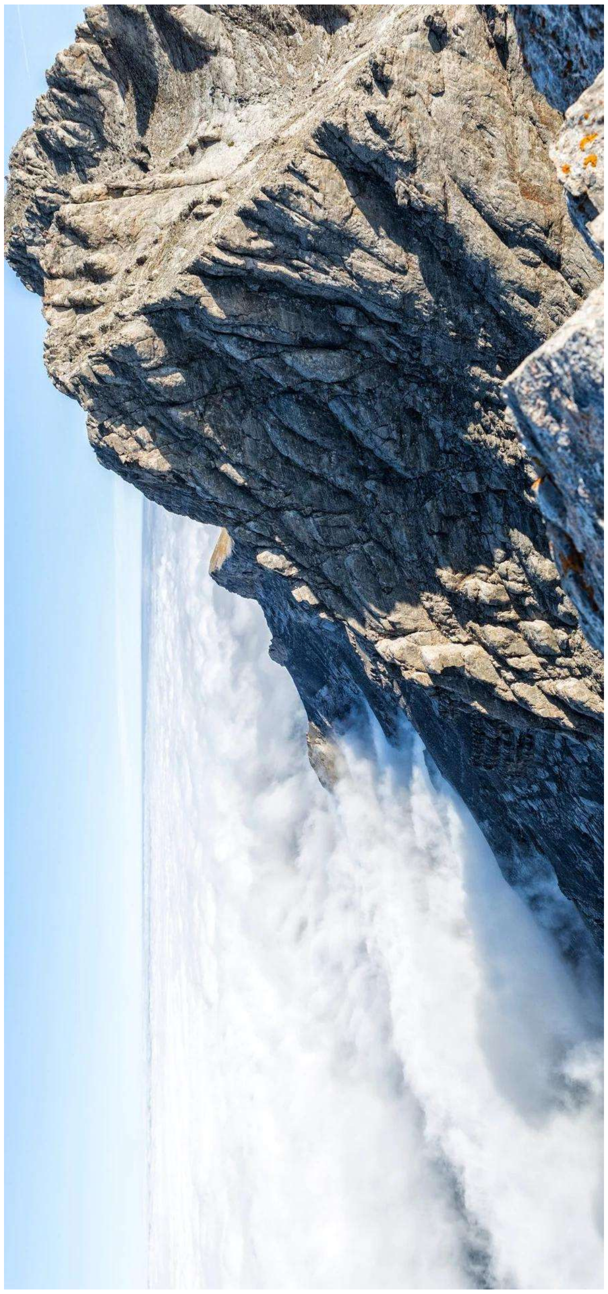
Vertical Flip:



Clockwise:

Anticlockwise:





Brightness:

Pixel Blur: