Image Editor

Welcome to my Image Editor!

This Java-based program provides various image editing tasks.

The main features of this Image Editor include:

1. GrayScale Conversion: Convert an image to grayscale, removing color information.
2. 90-Degree Clockwise Rotation: Rotate an image 90 degrees clockwise.
3. Horizontal Flip: Flip an image horizontally.
4. Vertical Flip: Flip an image vertically.
5. Brightness Adjustment: Adjust the brightness of an image.
6. Gaussian Blur: Blurring the image
7. 90-Degree Anti Clockwise Rotation: Rotate an image 90 degrees clockwise.

1. IMPORT STATEMENTS

The code begins by importing necessary classes from the Java AWT and ImageIO libraries to work with images and user input.

2. UTILITIES CLASS

The Utilities class provides utility methods for reading and saving images.

static BufferedImage readImage(String path) throws IOException: Reads an image from the specified file path using ImageIO and returns it as a BufferedImage.

static void saveImage(BufferedImage inp, String path, String format) throws IOException: Saves a given BufferedImage to a file path in the specified image format.

3. EDITOR CLASS

This class contains several static methods that perform different image editing operations:

greyScale(BufferedImage inp): Converts the input image to grayscale.

hozFlip(BufferedImage inp): Flips the input image horizontally.

verFlip(BufferedImage inp): Flips the input image vertically.

rotate90CW(BufferedImage inp): This method takes an input image and creates a new image where the content is rotated 90 degrees clockwise. It accomplishes this by swapping the x and y coordinates of each pixel in the input image while copying them to the output image. Additionally, the result is further flipped horizontally (hozFlip(out)) to ensure that the rotation is in the clockwise direction.

Rotate90ACW(BufferedImage inp): This method takes an input image and rotates it 90 degrees anticlockwise by swapping the x and y coordinates of each pixel, resulting in a rotated output image. This code is a common technique for simple image rotation operations.

changeBrightness(BufferedImage inp): It applies the brightness adjustment factor to each color channel:

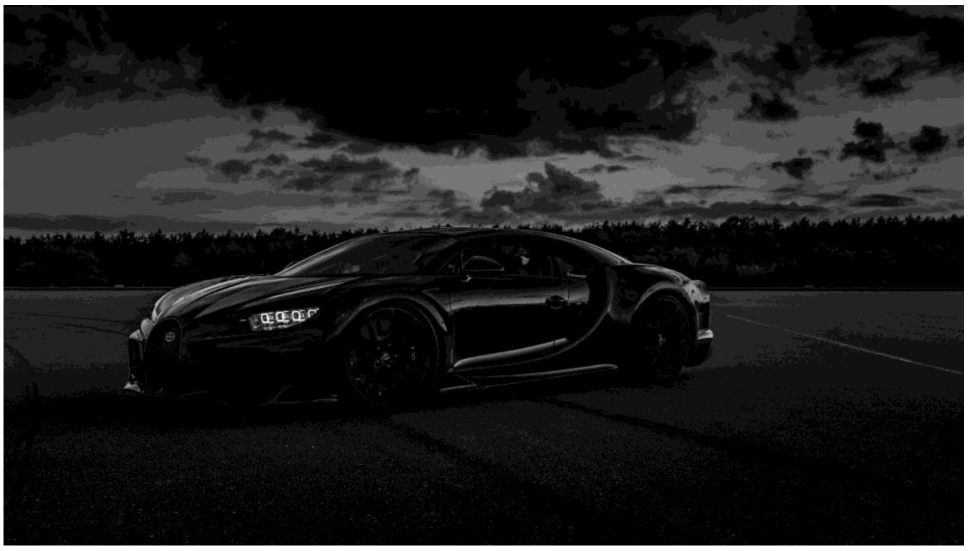
For each color channel (red, green, and blue), it calculates a new value (nRed, nGreen, and nBlue) based on the original value and the provided factor. The factor can be positive (to increase brightness) or negative (to decrease brightness).The calculated values are clamped to the range [0, 255] using Math.min and Math.max to ensure they stay within the valid color channel range. This prevents values from going below 0 (black) or above 255 (white). After processing all the pixels in the input image, the changeBrightness method returns the modified BufferedImage named out, which represents the input image with its brightness adjusted based on the provided factor.

4. MAIN CLASS

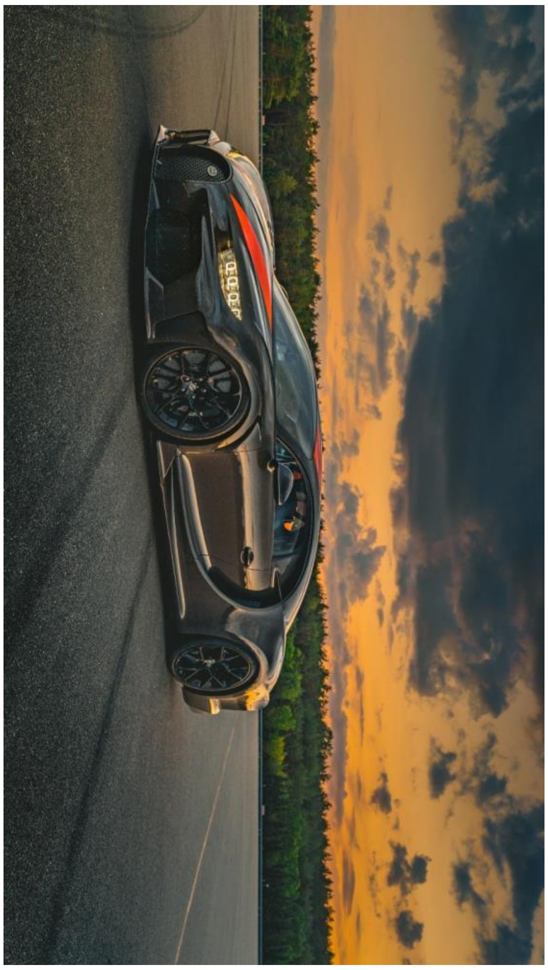
The program starts by displaying a welcome message and prompting the user to enter the file path of the input image. Users can choose from several image processing operations, such as grayscale conversion, brightness adjustment, rotation, flipping, and Gaussian blur. The user's choice is processed using a switch statement, and the corresponding operation is applied to the input image. If the operation is successful, the program prompts the user to specify the file path and format for saving the edited image. The edited image is saved using the Utilities.saveImage method. If any errors occur during image processing or saving, error messages are displayed.

Here are the images I tried all these functions on:

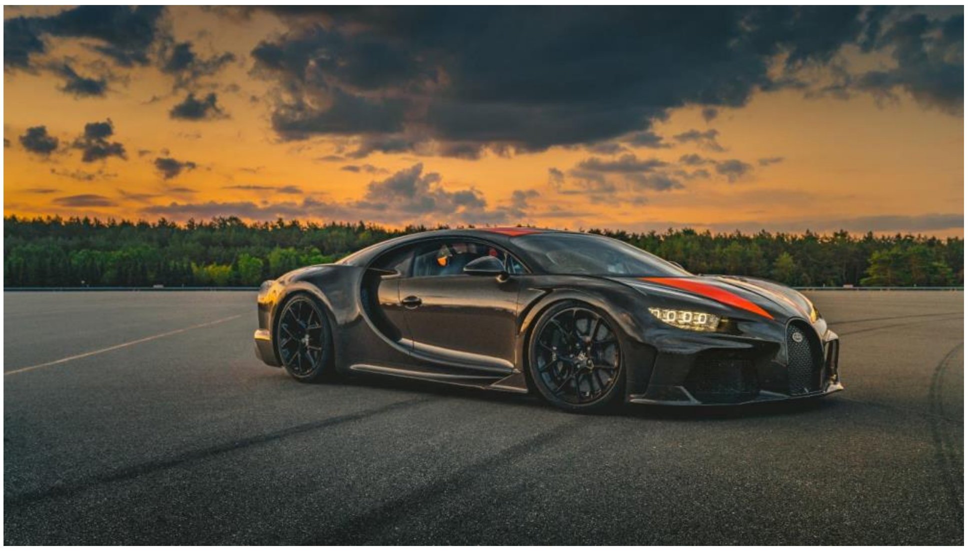
1. GrayScale Conversion



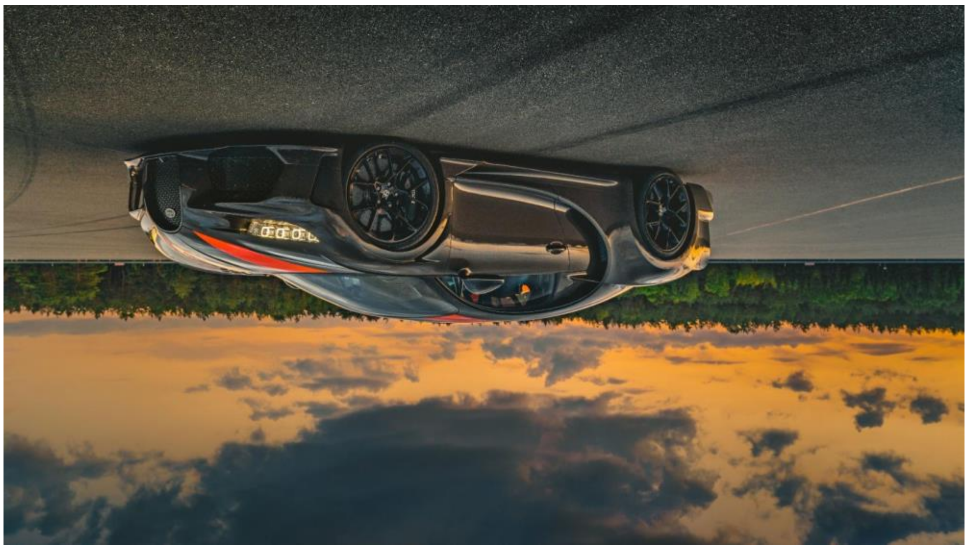
2. 90-Degree Clockwise Rotation



3. Horizontal Flip

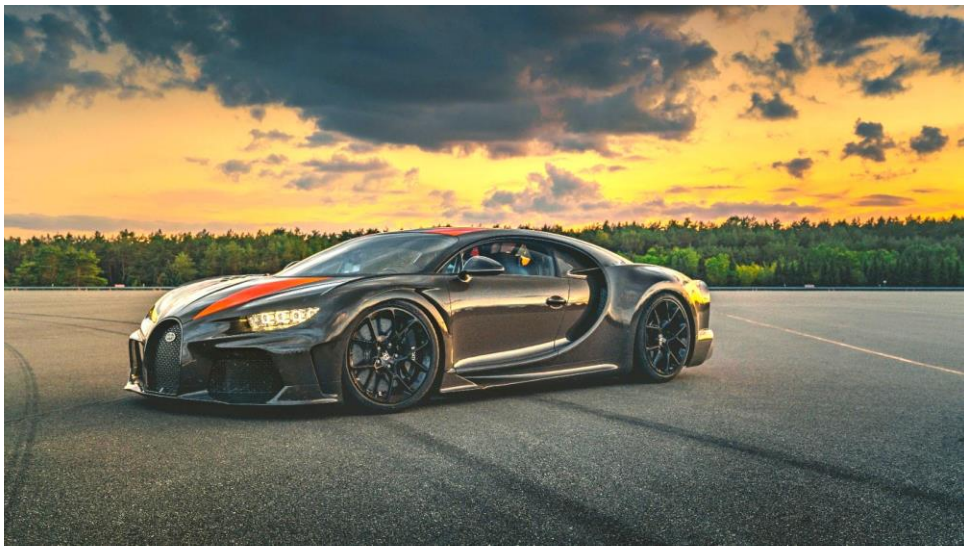


4. Vertical Flip



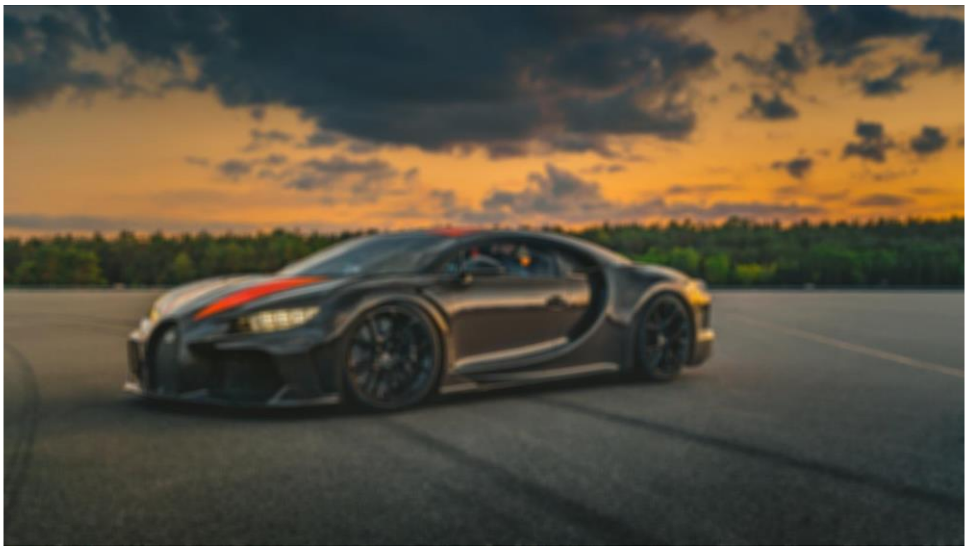
5. Brightness Adjustment

INCREASE BRIGHTNESS:



DECREASE BRIGHTNESS:

6. Gaussian Blur



7. 90-Degree Anti Clockwise Rotation