

Python Development Lab

Project Report

Instructor: Dr. Sandeep Udmale

Team members and Registration numbers:

Name of Student: Shubham Shivaji More

Registration Number: 221080047

Name of Student: Aditya Ramesh Latane

Registration Number: 221080039

Comprehensive Image Editor in Python

Problem Statement:

In today's digital age, image editing has become an integral part of our daily lives. However, the availability of user-friendly and versatile image editing tools is still limited. This project aims to address this gap by developing a comprehensive image editor in Python that caters to both novice and experienced users, providing a range of functionalities from basic editing to advanced image manipulation.

Motivation:

The motivation behind this project is to create an all-encompassing image editing tool that combines ease of use with powerful features. The goal is to empower users to perform various image manipulations without the need for expensive or complex software. This project is driven by a commitment to democratize image editing capabilities and make them accessible to a broader audience.

Methodology:

The project adopts a modular approach, separating image processing functions into a dedicated file (`Func.py`) and the graphical user interface (GUI) into the main script (`main.py`). The Pillow library is utilized for image processing, and Tkinter is employed for GUI development.

Func.py:

This module encapsulates the core image processing functionalities. These include:

- Opening and resizing images.
- Applying selected filters, such as Black and White, Blur, Sharpen, Smooth, and Emboss.
- Rotating and flipping images.
- Cropping images.
- Drawing shapes on images.
- Adjusting brightness and contrast.
- Saving edited images.

main.py:

The main script integrates the Tkinter GUI components and orchestrates the interaction between the user interface and the image processing functions. The GUI includes the following components:

- Buttons:
 - "Add Image": Allows users to load an image into the editor.
 - "Draw": Activates drawing mode, enabling users to draw on the image.
 - "Change Pen Color": Enables users to choose the color for drawing.
 - "Clear": Clears the canvas.
 - "Select Filter": Dropdown for choosing different filters.
 - "Rotate": Rotates the image.
 - "Flip": Flips the image vertically or horizontally.
 - "Crop": Initiates the crop mode.
 - "Brightness": Adjusts the brightness of the image.
 - "Contrast": Adjusts the contrast of the image.
 - "Save Edited Image": Saves the edited image.
- Canvas: The area where the image is displayed and edited.
- Dialogs: Pop-up dialogs for user input, such as rotation angle and filter selection.

Application Flow:

Image Loading:

- User clicks "Add Image" to load an image.
- The image is resized to fit the canvas.

Drawing:

- User clicks "Draw" to enter drawing mode.
- Drawing is facilitated by selecting pen color and size.

Filter Application:

- User selects a filter from the dropdown and clicks "Apply Filter."
- The selected filter is applied to the image.

Image Manipulation:

- User can rotate, flip, or crop the image as desired.
- Adjustments to brightness and contrast can be made.

Saving:

- User clicks "Save Edited Image" to save the final edited image.

Pseudo Code

Func.py

```
def open_and_resize_image(file_path):
    image = Image.open(file_path)
    width, height = int(image.width * 2), int(image.height * 2)
    return image.resize((width, height), image.ANTIALIAS if hasattr(image, 'ANTIALIAS') else Image.BICUBIC)

def apply_selected_filter(image, selected_filter):
    if selected_filter == "Black and White":
        return ImageOps.grayscale(image)
    elif selected_filter == "Blur":
        return image.filter(ImageFilter.BLUR)
    elif selected_filter == "Sharpen":
        return image.filter(ImageFilter.SHARPEN)
    elif selected_filter == "Smooth":
        return image.filter(ImageFilter.SMOOTH)
    elif selected_filter == "Emboss":
        return image.filter(ImageFilter.EMBOSS)
    else:
        return image

def rotate_image(image, angle):
    return image.rotate(angle)

def flip_image(image, direction):
    if direction == "Vertical":
        return ImageOps.flip(image)
    elif direction == "Horizontal":
        return ImageOps.mirror(image)
    else:
        return image

def crop_image(image, x1, y1, x2, y2):
    return image.crop((x1, y1, x2, y2))

def draw_on_image(image, x1, y1, x2, y2, color, size):
    draw = ImageDraw.Draw(image)
    draw.ellipse([x1, y1, x2, y2], fill=color, outline=color)

def save_image(image, file_path):
    if file_path:
        try:
            image.save(file_path)
        except ValueError as e:
            # Handle cases where the file extension is unknown
            if "unknown file extension" in str(e):
                # Provide a default extension like ".png"
                default_save_path = file_path + ".png"
                image.save(default_save_path)
                return default_save_path
            else:
                # If it's another ValueError, raise it
                raise
    else:
        raise ValueError("File path is required")

def adjust_brightness(image, factor):
    enhancer = ImageEnhance.Brightness(image)
    return enhancer.enhance(factor)

def adjust_contrast(image, factor):
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(factor)
```

Main.py

```
def add_image():
    global file_path, original_image, edited_image, canvas_image

    file_path = filedialog.askopenfilename(initialdir="D:/backup dell/PO LAB PROJECT/pictures")
    original_image = open_and_resize_image(file_path)
    edited_image = original_image.copy()

    canvas.config(width=original_image.width, height=original_image.height)
    canvas_image = ImageTk.PhotoImage(original_image)
    canvas.create_image(0, 0, image=canvas_image, anchor="nw")

def change_color():
    global pen_color
    pen_color = colorchooser.askcolor(title="Select Pen Color")[1]

def change_size(size):
    global pen_size
    pen_size = size

def draw(event):
    global edited_image
    x, y = event.x, event.y
    x1, y1 = (x - pen_size), (y - pen_size)
    x2, y2 = (x + pen_size), (y + pen_size)

    draw_on_image(edited_image, x1, y1, x2, y2, pen_color, pen_size)
    display_image(edited_image)

def start_crop(event):
    global cropping, crop_start, crop_rect
    crop_start = (event.x, event.y)
    crop_rect = canvas.create_rectangle(0, 0, 0, 0, outline="red", dash=(2, 2))
    cropping = True
    canvas.bind("<ButtonRelease-1>", end_crop)

def end_crop(event):
    global cropping, crop_start, original_image, canvas_image, crop_rect
    x1, y1 = crop_start
    x2, y2 = (event.x, event.y)

    if x2 >= x1 and y2 >= y1:
        cropped_image = crop_image(original_image, x1, y1, x2, y2)
        display_image(cropped_image)

    cropping = False
    crop_start = (0, 0)
    canvas.delete(crop_rect)

def clear_canvas():
    global original_image, canvas_image
    canvas.delete("all")
    canvas_image = ImageTk.PhotoImage(original_image)
    canvas.create_image(0, 0, image=canvas_image, anchor="nw")

def apply_filter(selected_filter):
    global original_image, canvas_image
    image = open_and_resize_image(file_path)
    filtered_image = apply_selected_filter(image, selected_filter)
    display_image(filtered_image)

def rotate_image_dialog():
    angle = simpledialog.askinteger("Rotate Image", "Enter rotation angle (in degrees):", parent=root, minvalue=0, maxvalue=360)
    if angle is not None:
        image = open_and_resize_image(file_path)
        rotated_image = rotate_image(image, angle)
        display_image(rotated_image)
```

```
def flip_image_dialog():
    direction = simpledialog.askstring("Flip Image", "Enter direction (Vertical/Horizontal):", parent=root)
    if direction and direction.lower() in ["vertical", "horizontal"]:
        image = open_and_resize_image(file_path)
        flipped_image = flip_image(image, direction.capitalize())
        display_image(flipped_image)

def crop_image_dialog():
    global cropping, drawing
    cropping = True
    drawing = False
    canvas.bind("<ButtonPress-1>", start_crop)

def brightness_adjustment_dialog():
    factor = simpledialog.askfloat("Brightness Adjustment", "Enter brightness factor (0.1 - 2.0):", parent=root, minvalue=0.1, maxvalue=2.0)
    if factor is not None:
        image = open_and_resize_image(file_path)
        adjusted_image = adjust_brightness(image, factor)
        display_image(adjusted_image)

def contrast_adjustment_dialog():
    factor = simpledialog.askfloat("Contrast Adjustment", "Enter contrast factor (0.1 - 2.0):", parent=root, minvalue=0.1, maxvalue=2.0)
    if factor is not None:
        image = open_and_resize_image(file_path)
        adjusted_image = adjust_contrast(image, factor)
        display_image(adjusted_image)

def save_image_dialog():
    save_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png"), ("All files", "*.*")])
    if save_path:
        save_image(original_image, save_path)

def display_image(image):
    global canvas_image
    canvas_image = ImageTk.PhotoImage(image)
    canvas.config(width=canvas_image.width(), height=canvas_image.height())
    canvas.create_image(0, 0, image=canvas_image, anchor="nw")

def draw_image_dialog():
    global cropping, drawing
    cropping = False
    drawing = True
    canvas.bind("<B1-Motion>", draw)
```


Output:

window:

Add Image

Draw

Change Pen Color

☐ Small ☐ Medium ☐ Large

Clear

Select Filter

▼

Rotate

Flip

Crop

Brightness

Contrast

Save Edited Image



Discussion

The project successfully combines the functionality of image processing with an intuitive graphical interface. However, areas for improvement include enhancing the UI for better user experience and potentially incorporating more advanced image processing techniques. This project lays the foundation for future developments in creating a comprehensive and user-friendly image editing tool in Python.

*Grateful for your exceptional guidance
and teaching, Dr. Sandeep Udmale Sir*