UMS
UNIVERSITI MALAYSIA SABAH

MCG
Mathematics Computer Graphics
UNIVERSITI MALAYSIA SABAH

SV30703

DIGITAL IMAGE PROCESSING

ASSIGNMENT 1

LECTURER: PROF. DR. ABDULLAH BIN BADE

No.	Name	Matrix
1	MOHAMAD ADLI ZILIKHRAM BIN SAHARUDIN	BS22110469

Table of Contents

No.	Topics	Page
1	Introduction	3
2	Application Explanation: 2.1 Flow diagram 2.2 System Architecture 2.3 Algorithm	 4 5 - 7 8 - 10
3	Strength and Uniqueness	11 - 12
4	Sample Outputs	13 - 16

1.0 Introduction

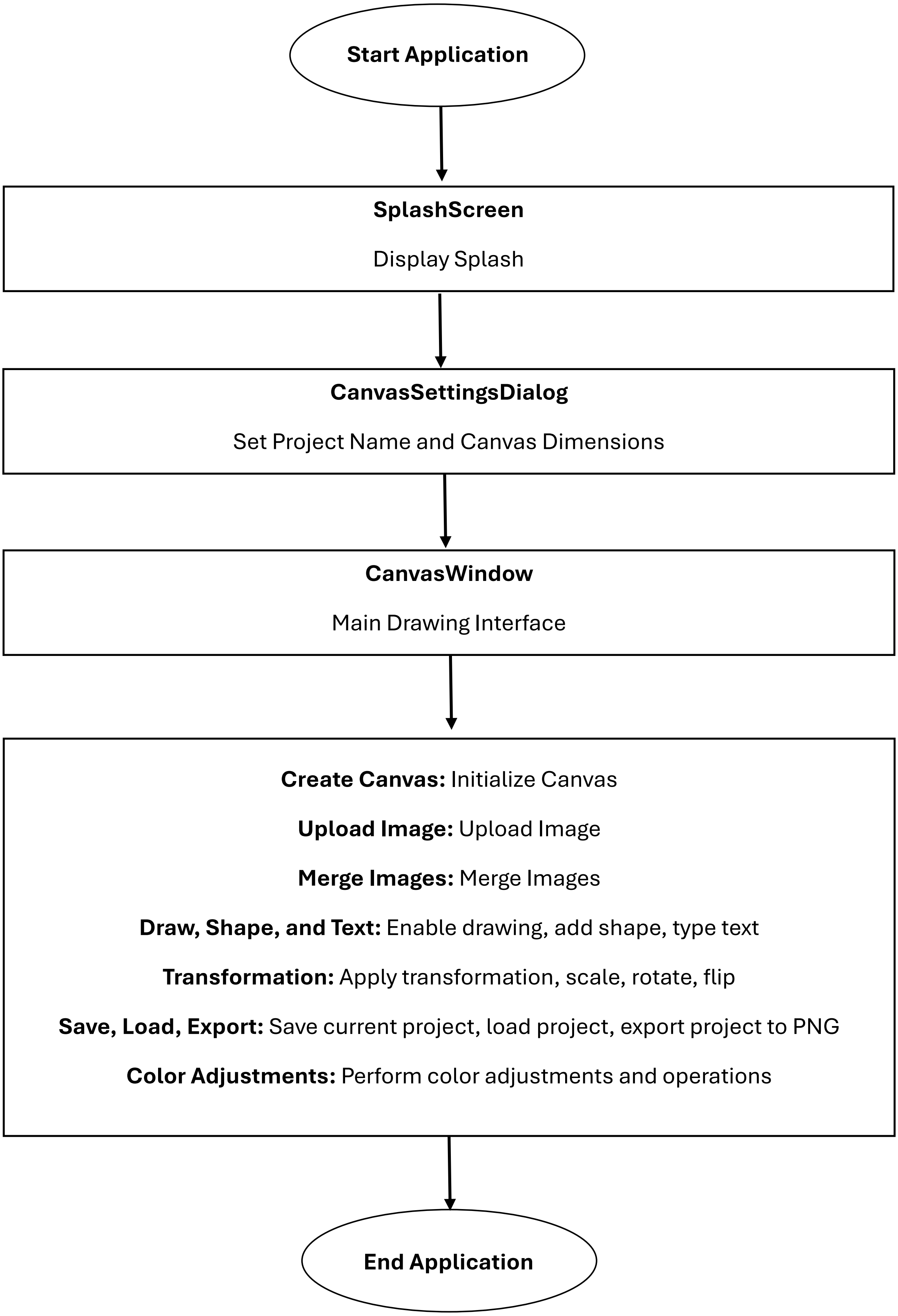
The software project outlined in the provided Python script introduces "Fairy Painting," an advanced graphical editing application leveraging the PyQt5 framework. This application is designed to provide users with a dynamic and interactive environment for creating, editing, and managing digital artwork. Featuring a wide array of functionalities, the program integrates basic and advanced tools for image manipulation, drawing, text editing, and shape creation, catering to diverse user requirements ranging from casual use to professional digital design.

The code employs a modular architecture to ensure scalability and maintainability. Core features include image uploading, editing, merging, and transformation capabilities such as scaling, rotation, flipping, and color adjustment. The integration of OpenCV allows for advanced image processing operations like gamma correction, color space conversion, and histogram analysis. Additionally, the application enhances user experience with a splash screen, customizable canvas settings, and a thumbnail panel for quick access to uploaded and merged images.

The project highlights an emphasis on user interface design and usability. By utilizing PyQt5 widgets such as sliders, buttons, dropdown menus, and dialog boxes, the application facilitates intuitive navigation and operation. Features like real-time feedback during drawing, a mini-canvas for overview, and pixel-level precision adjustments make the application versatile and user-friendly. This report aims to document the functionality, implementation details, and practical applications of the "Fairy Painting" application.

2.0 Application Explanation

2.1 Flow Diagram


Start Application
SplashScreen
Display Splash
CanvasSettingsDialog
Set Project Name and Canvas Dimensions
CanvasWindow
Main Drawing Interface
Create Canvas: Initialize Canvas
Upload Image: Upload Image
Merge Images: Merge Images
Draw, Shape, and Text: Enable drawing, add shape, type text
Transformation: Apply transformation, scale, rotate, flip
Save, Load, Export: Save current project, load project, export project to PNG
Color Adjustments: Perform color adjustments and operations
End Application

2.2 System Architecture

1. Presentation Layer (UI/UX)

- **Objective:** Manages the graphical user interface and user interactions.
- **Components:**
  - **CanvasSettingsDialog:**
    - Allows the user to specify canvas dimensions and project name before starting.
  - **CanvasWindow:**
    - Main interface for drawing and editing, consisting of:
      - Canvas Area: Displays the drawable canvas.
      - Toolbar: Provides tools for drawing, zooming, cropping, flipping, scaling, and color adjustments.
      - Thumbnails Panel: Displays a list of uploaded/edited images for selection.
  - **MergeDialog:**
    - Allows selection of images to merge with options for orientation (horizontal or vertical).
  - **SplashScreen:**
    - Displays an introductory splash image at startup.

2. Logic Layer (Application Logic)

- **Objective:** Handles application workflows and features based on user actions.
- **Key Functionalities:**
  - **Canvas Interaction:**
    - Drawing and text input using various tools (Brush, Pen, Marker, etc.).
    - Shape creation (Rectangle, Circle, Line, Triangle, Square).
    - Transformations (Scaling, Rotation, Cropping, Flipping).
  - **Image Operations:**
    - Upload, save, and load images.
    - Merging multiple images.
    - Exporting the canvas as a PNG file.
  - **Color Adjustments:**
    - Pixel color change, gamma correction, and color model conversions (RGB, HSV, Grayscale, etc.).
  - **Histograms and Analysis:**
    - Generate and display RGB channel histograms.

    - **Mini Canvas:**
        - Provides an overview of the canvas in a smaller dialog.

3. Data Management Layer

- **Objective:** Handles data storage, retrieval, and file operations.
- **Components:**
    - **File Operations:**
        - Save and load canvas states and objects using pickle.
        - Export final outputs as PNG files.
    - **Objects Management:**
        - Tracks all graphical objects (images, shapes, text) on the canvas with attributes such as position, size, and transformations.

4. Event Handling Layer

- **Objective:** Manages interaction-driven events.
- **Components:**
    - **Mouse events for:**
        - Drawing, shape creation, and cropping.
        - Selecting and dragging objects.
        - Applying pixel-level color changes.
    - **Keyboard events for:**
        - Text input and special controls (e.g., Enter to finalize text).

5. Error Handling Layer

- **Objective:** Manages errors and ensures system robustness.
- **Mechanisms:**
    - Validation for invalid image operations (e.g., unsupported formats or empty selections).
    - Graceful degradation for missing resources (e.g., splash screen image fallback).
    - User prompts for missing or incorrect inputs (e.g., empty project name, unselected objects).

6. Advanced Features Layer (Optional Enhancements)

- **Color Model Conversions:**
    - Switch between RGB, HSV, Grayscale, Lab (CIE), HLS, and YCrCb.
- **Custom Gamma Correction:**
    - Applies user-defined gamma adjustments for image enhancement.
- **Real-time Histograms:**
    - Displays RGB intensity distributions dynamically for selected objects.

2.3 Algorithm

1. **Splash Screen Algorithm**
   Displays a splash screen for a specified duration using QTimer.

2. **Canvas Initialization Algorithm**
   Creates a blank canvas using QPixmap and fills it with a specified background color (Qt.white).

3. **Save Canvas Algorithm**
   Converts QPixmap objects and associated metadata into serializable data and saves it using Python's pickle module.

4. **Load Canvas Algorithm**
   Deserializes data from a file and reconstructs the canvas, including persistent elements and image objects.

5. **Canvas Redraw Algorithm**
   Iterates through all elements (shapes, drawings, text, objects) and redraws them on the canvas using a QPainter.

6. **Merge Images Algorithm**
   Horizontally or vertically concatenates selected images using OpenCV functions (cv2.hconcat, cv2.vconcat).

7. **Image Scaling Algorithm**
   Adjusts the size of selected objects using QPixmap.scaled, based on a scaling factor derived from slider input.

8. **Zoom Canvas Algorithm**
   Scales the entire canvas, including all elements and objects, maintaining their relative positions and proportions.

9. **Translation Algorithm**
   Updates the position of selected objects based on slider or input values and moves their associated QRect.

10. **Rotation Algorithm**

    Rotates selected objects around their center using a transformation matrix via QTransform.

11. **Drawing Algorithm**

    Tracks mouse movement and creates continuous paths or segments on the canvas based on the selected drawing tool (e.g., Pen, Eraser).

12. **Shape Creation Algorithm**

    Draws predefined shapes (Circle, Rectangle, Square, Line, Triangle) between the start and end points of a mouse drag event.

13. **Flood-Fill Algorithm (Pixel Color Change)**

    Applies a flood-fill to a connected group of pixels starting from a clicked position, changing their color using a stack-based iterative approach.

14. **Crop Algorithm**

    Defines a rectangular cropping region and applies it to the selected object, updating its dimensions and content.

15. **Flip Image Algorithm**

    Flips an image horizontally or vertically using QTransform.scale.

16. **Gamma Correction Algorithm**

    Adjusts the gamma value of an image using OpenCV's lookup table (cv2.LUT) to modify brightness and contrast.

17. **Color Conversion Algorithm**

    Converts an image's color space (e.g., RGB, HSV, GRAY) using OpenCV's cv2.cvtColor.

18. **Negative Image Algorithm**

    Inverts the pixel values of an image to create a negative effect.

19. **Bitwise Operation Algorithm**

    Performs bitwise operations (AND, OR, XOR) between the image and a derived grayscale version using OpenCV functions.

20. **Mini Canvas Preview Algorithm**

    Scales down the current canvas and displays it in a separate dialog for an overview.

21. **Histogram Display Algorithm**

    Computes and displays the histograms of RGB channels using OpenCV's cv2.calcHist and Matplotlib.

22. **Text Input and Style Algorithm**

    Handles text input and styling (bold, italic, underline) on the canvas, rendering the text using QFont and QPainter.

23. **Thumbnail Management Algorithm**

    Creates and updates thumbnails for each image object and allows selecting objects by clicking thumbnails.

24. **File Export Algorithm**

    Saves the canvas content as an image file (e.g., PNG) using QPixmap.save.

25. **Object Highlighting Algorithm**

    Draws a red rectangle around the selected object to indicate its selection.

3.0 Strength and Uniqueness

- **Dynamic Canvas Creation:**
  The canvas size and settings are dynamically configured through a pre-launch dialog, offering a customizable workspace for users.

- **Advanced Merging Features:**
  The ability to merge multiple images with orientation options (side-by-side or stacked) is a standout feature not commonly found in standard applications.

- **Mini Canvas for Overview:**
  A mini canvas dialog provides an overview of the entire workspace, a feature useful for large-scale projects.

- **Flexible Color Transformations:**
  Support for various color modes (e.g., HSV, CIE, HLS) and advanced transformations like gamma correction highlights its versatility.

- **Histograms and Bitwise Operations:**
  Integrated histogram analysis and bitwise operations make the application suitable for detailed image processing and analysis.

- **Integration with OpenCV and PyQt5:**
  The code demonstrates a seamless integration of OpenCV's image manipulation capabilities with PyQt5's GUI functionalities.

- **Detailed Object Management:**
  Each object on the canvas is tracked with properties like position, rotation, and scale, enabling intricate control and edits.

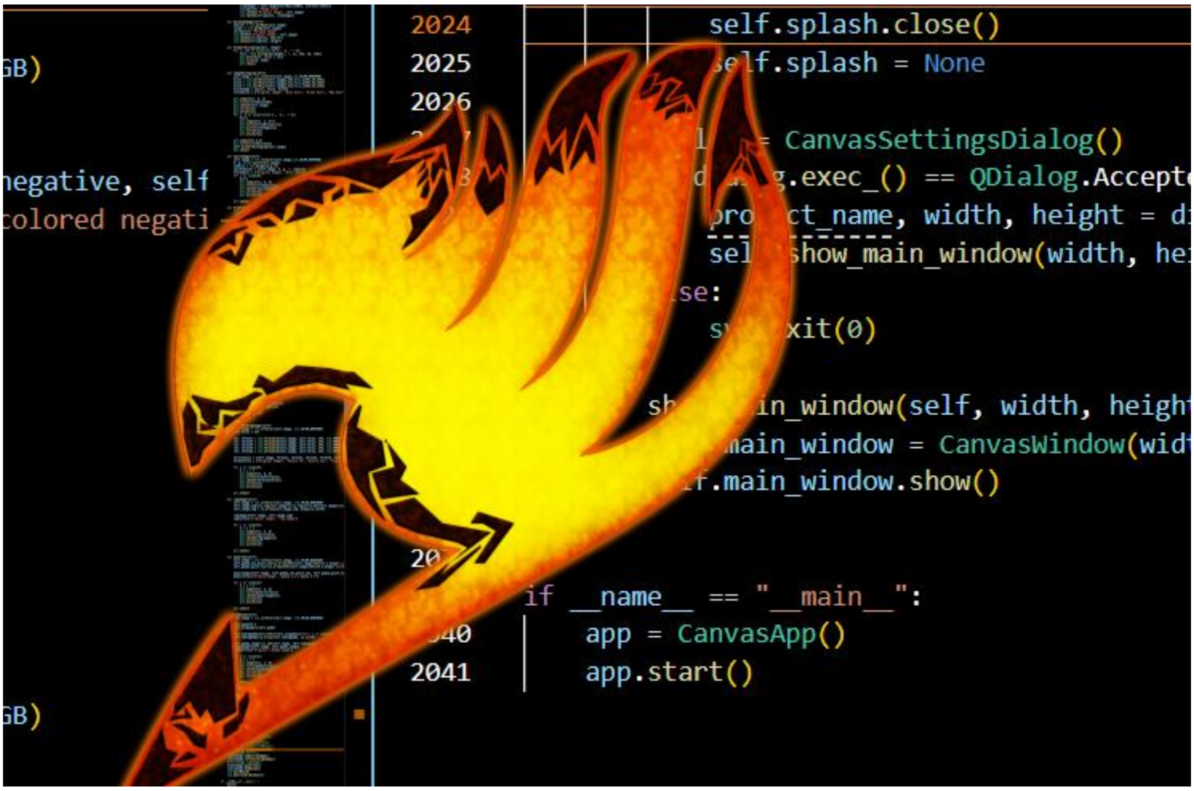
- **Customizable User Experience:**
  Checkable buttons and interactive sliders enhance user control, while feedback mechanisms like tooltips and color previews improve usability.

- **Multiple Tool Modes:**
  Inclusion of tools like brushes, markers, highlighters, and pencils, alongside text formatting options (bold, italic, underline), caters to diverse use cases.

- **Focus on Accessibility:**
  Thoughtful UI design, with visibility toggles for controls and logical grouping of functions, ensures accessibility for users of varying skill levels.

4.0 Sample Outputs

SplashArt



```
GB)

negative, self
colored negati

GB)
```

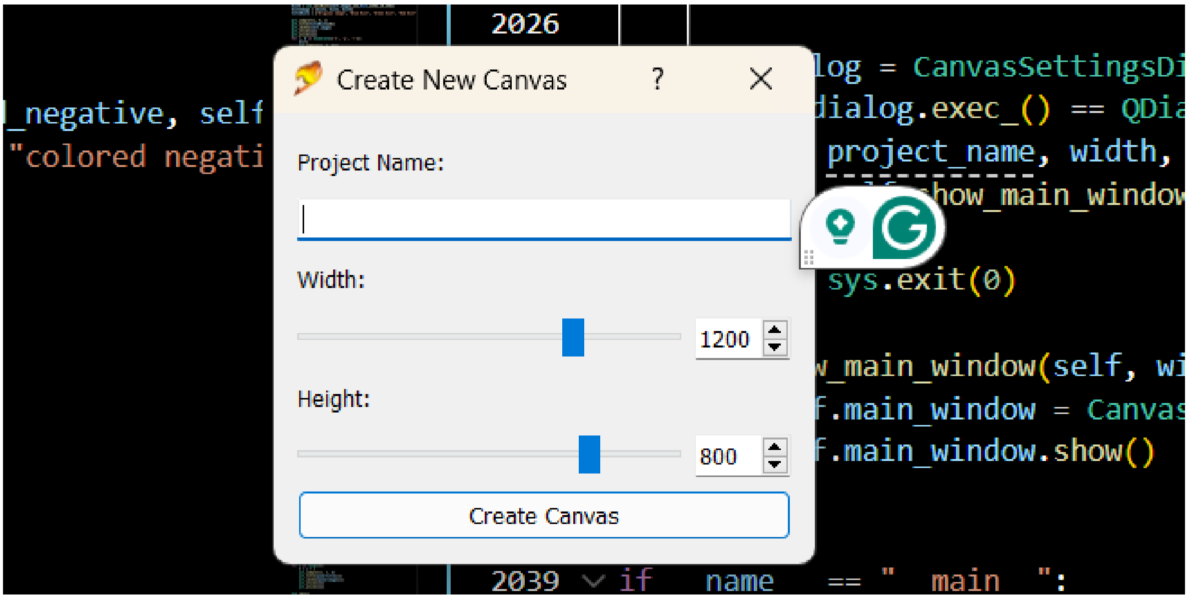
```
        self.splash.close()
        self.splash = None

        l    = CanvasSettingsDialog()
        d     g.exec_() == QDialog.Accepte
            pro  ct_name, width, height = di
            sel  show_main_window(width, he
        se:
            s    xit(0)

    sh      in_window(self, width, height
        main_window = CanvasWindow(widt
        f.main_window.show()

if __name__ == "__main__":
    app = CanvasApp()
    app.start()
```

Canvas Initialization



Create New Canvas

Project Name:

Width: 1200

Height: 800

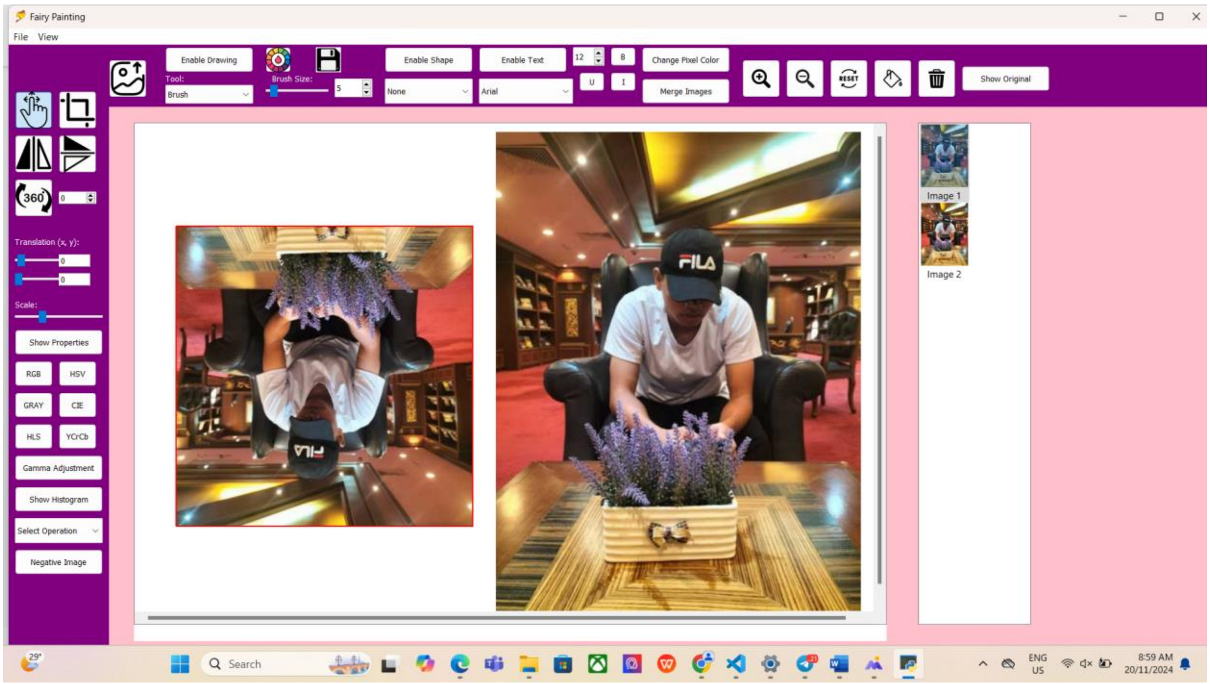
```
_negative, self
"colored negati

log = CanvasSettingsDi
dialog.exec_() == QDia
    project_name, width,
    self.show_main_window
    sys.exit(0)

w_main_window(self, wi
f.main_window = Canvas
f.main_window.show()

if  name  == "  main  ":
```

Image Upload and Manipulation



Drawing, Shape, Texting Tools

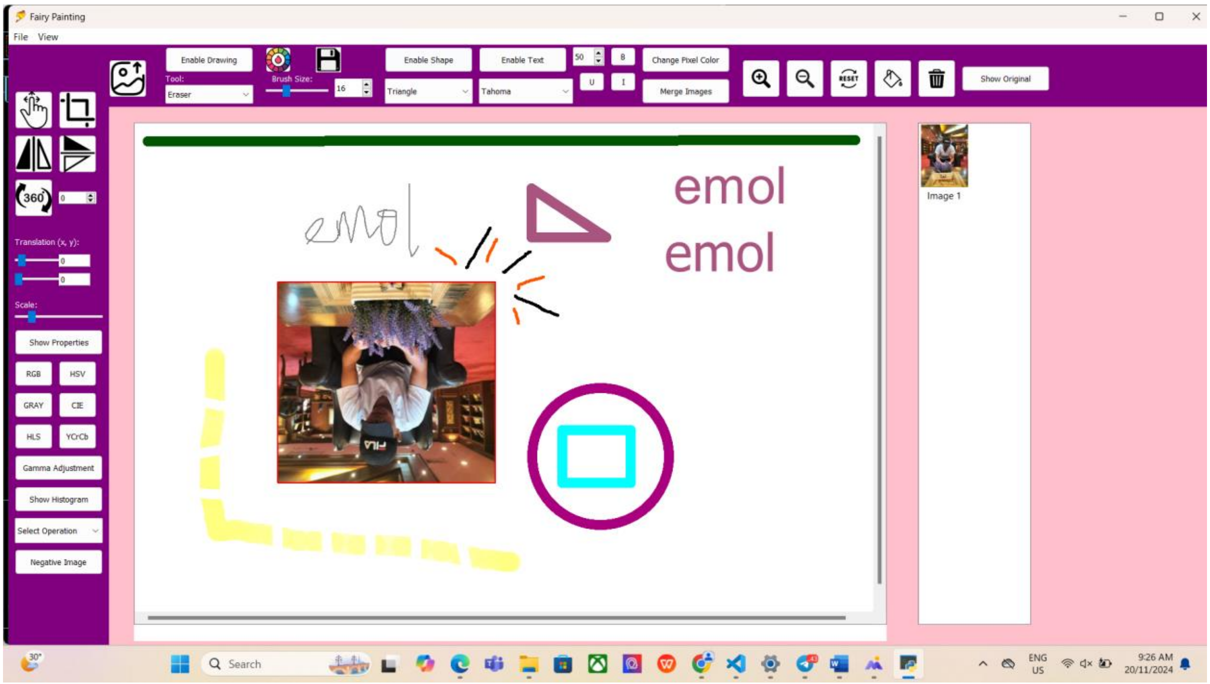
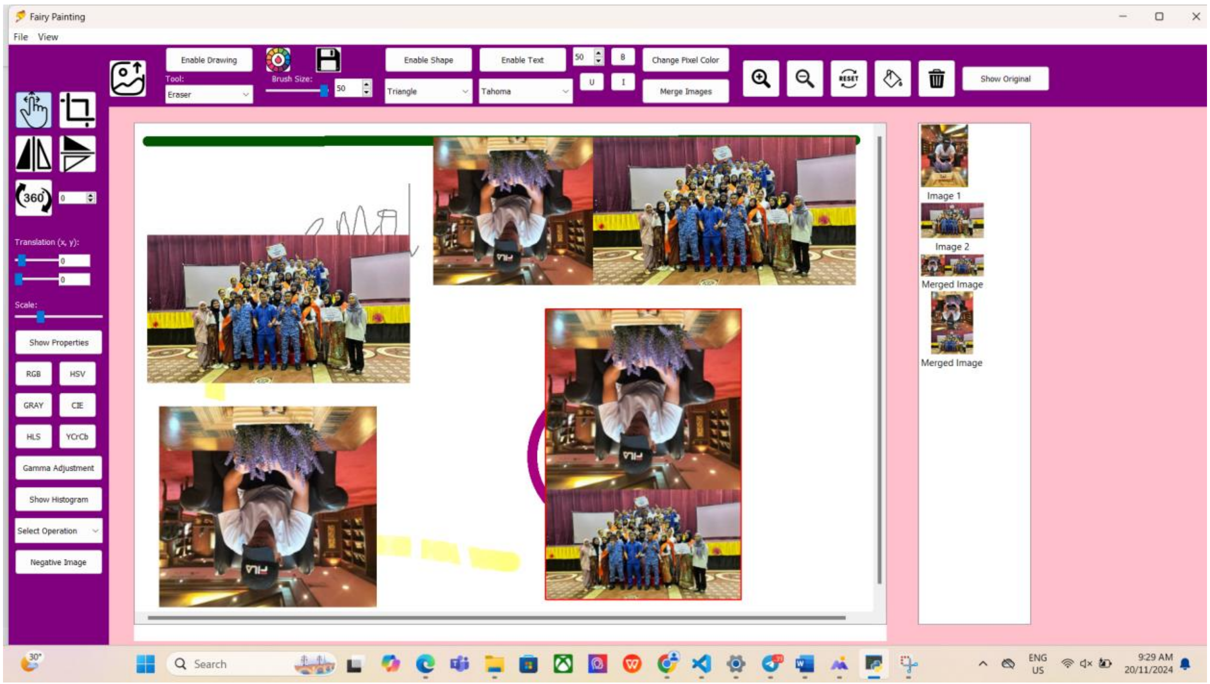


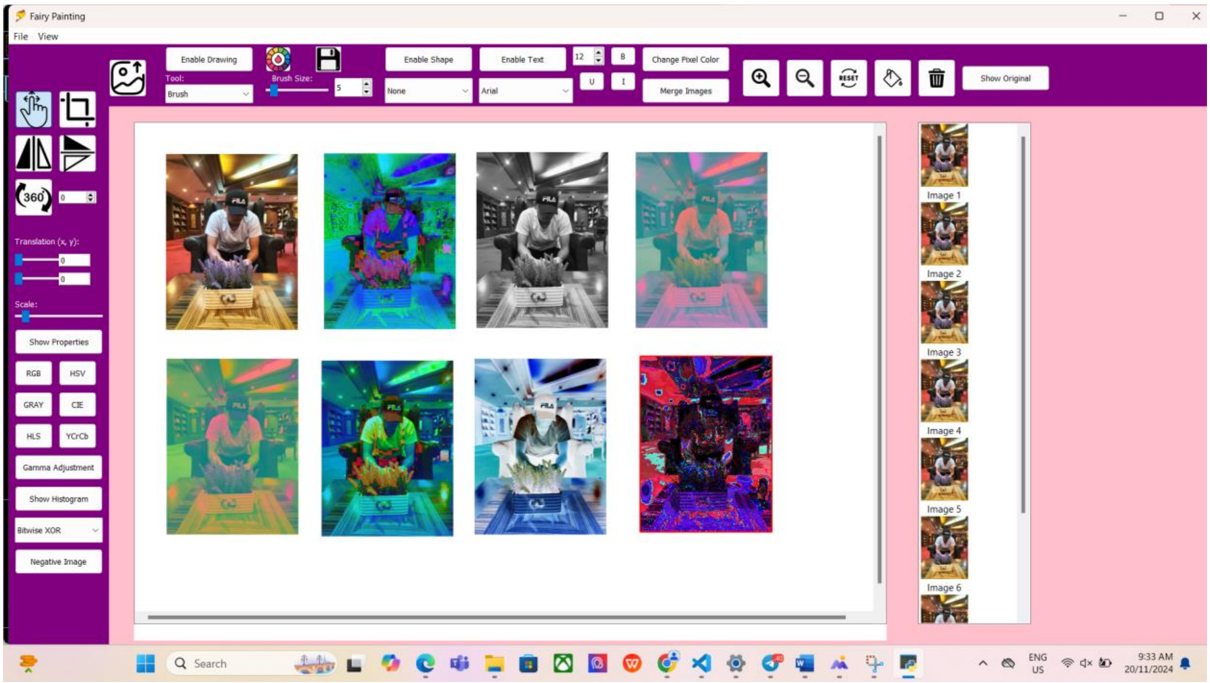
Image Merging



Tool: Eraser

Brush Size: 50

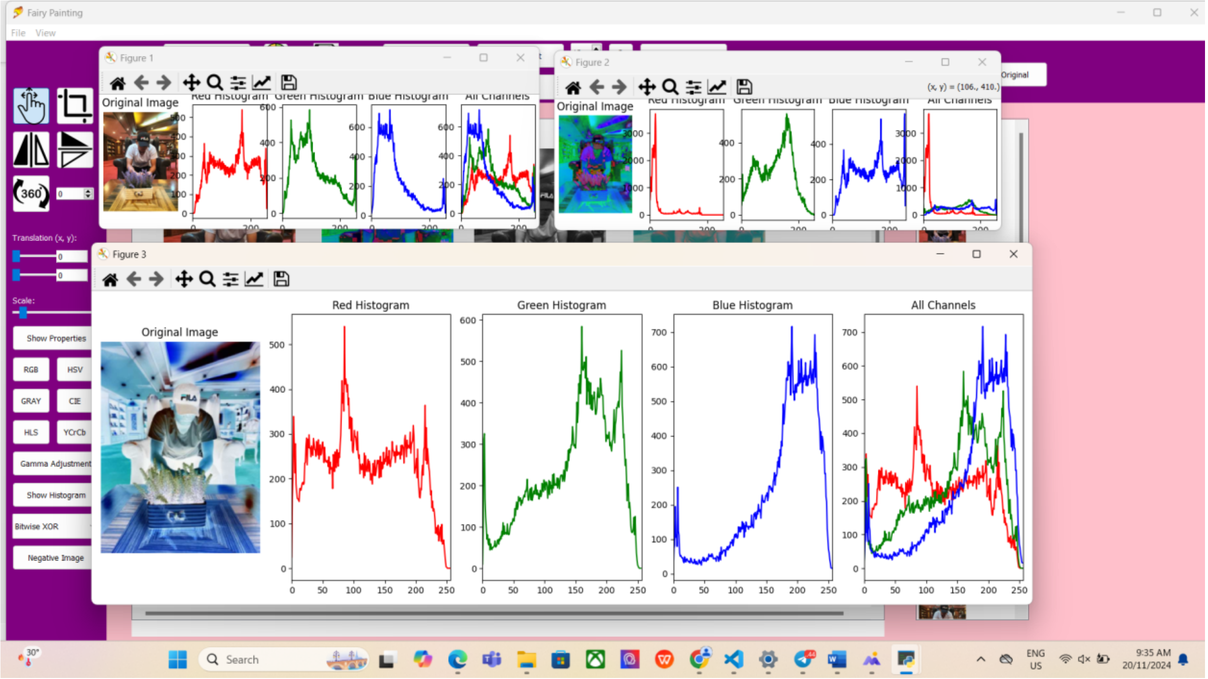
Color Conversion



Tool: Brush

Brush Size: 5

Histogram


Fairy Painting
File
View
Figure 1
Original Image
Red Histogram
Green Histogram
Blue Histogram
All Channels
Figure 2
(x, y) = (106., 410.)
Original
Translation (x, y):
Scale:
Show Properties
RGB
HSV
GRAY
CIE
HLS
YCrCb
Gamma Adjustment
Show Histogram
Bitwise XOR
Negative Image
Figure 3
Original Image
Red Histogram
0
100
200
300
400
500
0
50
100
150
200
250
Green Histogram
0
100
200
300
400
500
600
Blue Histogram
0
100
200
300
400
500
600
700
All Channels
0
100
200
300
400
500
600
700
Search
ENG
US
9:35 AM
20/11/2024
30°