G12-2024 - Feature #1123

Support # 1098 (In Progress): Coding Thread

Feature # 1112 (Resolved): GUI for the Code

Example Code Needs to fixed

10/24/2024 01:37 PM - Haolun DING

Status:	Resolved	**Start date:**	10/24/2024
Priority:	High	**Due date:**	10/31/2024
Assignee:	Andifallih Noor MALELA	**% Done:**	100%
Category:		**Estimated time:**	2.00 hours
Target version:		**Spent time:**	6.50 hours

Description

```
class ImageProcessingApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Processing Application")

        # Initialize variables
        self.left_image_path = ""
        self.right_image_path = ""
        self.left_params = {}
        self.right_params = {}
        self.processed_images = {}

        # Create notebook (tabs)
        self.notebook = ttk.Notebook(root)
        self.notebook.pack(expand=True, fill='both')

        # Create frames for each tab
        self.tab_left = ttk.Frame(self.notebook)
        self.tab_right = ttk.Frame(self.notebook)
        self.tab_settings = ttk.Frame(self.notebook)
        self.tab_preview = ttk.Frame(self.notebook)

        self.notebook.add(self.tab_left, text='Left Image')
        self.notebook.add(self.tab_right, text='Right Image')
        self.notebook.add(self.tab_settings, text='Settings')
        self.notebook.add(self.tab_preview, text='Preview')

        # Setup each tab
        self.setup_left_tab()
        self.setup_right_tab()
        self.setup_settings_tab()
        self.setup_preview_tab()

        # Process and Save Button
        self.process_button = tk.Button(root, text="Process and Save Images", command=self.process
_and_save)
        self.process_button.pack(pady=10)

        # Progress Bar
        self.progress_bar = ttk.Progressbar(root, orient="horizontal", length=400, mode="determina
te")
        self.progress_bar.pack(pady=10)

    def setup_left_tab(self):
        frame = self.tab_left

        # Select Image Button
        btn_select = tk.Button(frame, text="Select Left Image", command=self.select_left_image)
        btn_select.pack(pady=5)
```

```
        # Display selected image path
        self.left_image_label = tk.Label(frame, text="No image selected", wraplength=300, anchor="
w", justify="left")
        self.left_image_label.pack(padx=10, pady=5)

        # Parameters for left image
        params_frame = tk.LabelFrame(frame, text="Left Image Parameters", padx=10, pady=10)
        params_frame.pack(padx=10, pady=10, fill="x")

        # Projected Image Width
        tk.Label(params_frame, text="Projected Image Width:").grid(row=0, column=0, sticky="e")
        self.left_projected_image_width = tk.Entry(params_frame)
        self.left_projected_image_width.insert(0, "800")
        self.left_projected_image_width.grid(row=0, column=1, pady=2, sticky="w")

        # Projected Overlap Width
        tk.Label(params_frame, text="Projected Overlap Width:").grid(row=1, column=0, sticky="e")
        self.left_projected_overlap_width = tk.Entry(params_frame)
        self.left_projected_overlap_width.insert(0, "100")
        self.left_projected_overlap_width.grid(row=1, column=1, pady=2, sticky="w")

        # Gamma
        tk.Label(params_frame, text="Gamma:").grid(row=2, column=0, sticky="e")
        self.left_gamma = tk.Entry(params_frame)
        self.left_gamma.insert(0, "1.0")
        self.left_gamma.grid(row=2, column=1, pady=2, sticky="w")

        # Image Side
        tk.Label(params_frame, text="Image Side:").grid(row=3, column=0, sticky="e")
        self.left_image_side = tk.IntVar(value=0)  # Default to left
        tk.Radiobutton(params_frame, text="Left", variable=self.left_image_side, value=0).grid(row
=3, column=1, sticky="w")
        tk.Radiobutton(params_frame, text="Right", variable=self.left_image_side, value=1).grid(ro
w=3, column=1, padx=60, sticky="w")

        # Transparency Factor
        tk.Label(params_frame, text="Transparency Factor:").grid(row=4, column=0, sticky="e")
        self.left_transparency_factor = tk.Entry(params_frame)
        self.left_transparency_factor.insert(0, "1.0")
        self.left_transparency_factor.grid(row=4, column=1, pady=2, sticky="w")

    def setup_right_tab(self):
        frame = self.tab_right

        # Select Image Button
        btn_select = tk.Button(frame, text="Select Right Image", command=self.select_right_image)
        btn_select.pack(pady=5)

        # Display selected image path
        self.right_image_label = tk.Label(frame, text="No image selected", wraplength=300, anchor=
"w", justify="left")
        self.right_image_label.pack(padx=10, pady=5)

        # Parameters for right image
        params_frame = tk.LabelFrame(frame, text="Right Image Parameters", padx=10, pady=10)
        params_frame.pack(padx=10, pady=10, fill="x")

        # Projected Image Width
        tk.Label(params_frame, text="Projected Image Width:").grid(row=0, column=0, sticky="e")
        self.right_projected_image_width = tk.Entry(params_frame)
        self.right_projected_image_width.insert(0, "800")
        self.right_projected_image_width.grid(row=0, column=1, pady=2, sticky="w")

        # Projected Overlap Width
        tk.Label(params_frame, text="Projected Overlap Width:").grid(row=1, column=0, sticky="e")
        self.right_projected_overlap_width = tk.Entry(params_frame)
```

```
        self.right_projected_overlap_width.insert(0, "100")
        self.right_projected_overlap_width.grid(row=1, column=1, pady=2, sticky="w")

        # Gamma
        tk.Label(params_frame, text="Gamma:").grid(row=2, column=0, sticky="e")
        self.right_gamma = tk.Entry(params_frame)
        self.right_gamma.insert(0, "1.0")
        self.right_gamma.grid(row=2, column=1, pady=2, sticky="w")

        # Image Side
        tk.Label(params_frame, text="Image Side:").grid(row=3, column=0, sticky="e")
        self.right_image_side = tk.IntVar(value=1)  # Default to right
        tk.Radiobutton(params_frame, text="Left", variable=self.right_image_side, value=0).grid(ro
w=3, column=1, sticky="w")
        tk.Radiobutton(params_frame, text="Right", variable=self.right_image_side, value=1).grid(r
ow=3, column=1, padx=60, sticky="w")

        # Transparency Factor
        tk.Label(params_frame, text="Transparency Factor:").grid(row=4, column=0, sticky="e")
        self.right_transparency_factor = tk.Entry(params_frame)
        self.right_transparency_factor.insert(0, "1.0")
        self.right_transparency_factor.grid(row=4, column=1, pady=2, sticky="w")

    def setup_settings_tab(self):
        frame = self.tab_settings

        # Configurations can be saved or loaded here
        save_config_btn = tk.Button(frame, text="Save Configuration", command=self.save_configurat
ion)
        save_config_btn.pack(pady=10)

        load_config_btn = tk.Button(frame, text="Load Configuration", command=self.load_configurat
ion)
        load_config_btn.pack(pady=10)

    def setup_preview_tab(self):
        frame = self.tab_preview

        # Labels to display images
        self.original_image_label = tk.Label(frame, text="Original Image")
        self.original_image_label.pack(side="left", padx=10, pady=10)

        self.processed_image_label = tk.Label(frame, text="Processed Image")
        self.processed_image_label.pack(side="right", padx=10, pady=10)

    def select_left_image(self):
        path = filedialog.askopenfilename(title="Select Left Image",
                                          filetypes=[("Image files", "*.png *.jpg *.jpeg *.bmp")])
        if path:
            self.left_image_path = path
            self.left_image_label.config(text=os.path.basename(path))
            logging.info(f"Selected left image: {path}")

    def select_right_image(self):
        path = filedialog.askopenfilename(title="Select Right Image",
                                          filetypes=[("Image files", "*.png *.jpg *.jpeg *.bmp")])
        if path:
            self.right_image_path = path
            self.right_image_label.config(text=os.path.basename(path))
            logging.info(f"Selected right image: {path}")

    def save_configuration(self):
        config = ConfigReader()

        # Left Image Parameters
        if self.left_image_path:
            config.setParameters({
```

```
                'image_name': self.left_image_path,
                'projected_image_width': self.left_projected_image_width.get(),
                'projected_overlap_width': self.left_projected_overlap_width.get(),
                'gamma': self.left_gamma.get(),
                'image_side': self.left_image_side.get(),
                'transparency_factor': self.left_transparency_factor.get()
            })

        # Right Image Parameters
        if self.right_image_path:
            config.setParameters({
                'image_name': self.right_image_path,
                'projected_image_width': self.right_projected_image_width.get(),
                'projected_overlap_width': self.right_projected_overlap_width.get(),
                'gamma': self.right_gamma.get(),
                'image_side': self.right_image_side.get(),
                'transparency_factor': self.right_transparency_factor.get()
            })

        # Save to file
        save_path = filedialog.asksaveasfilename(title="Save Configuration",
                                                 defaultextension=".ini",
                                                 filetypes=[("INI files", "*.ini")])
        if save_path:
            config.save_config(save_path)
            messagebox.showinfo("Success", f"Configuration saved to {save_path}")
            logging.info(f"Configuration saved to {save_path}")

    def load_configuration(self):
        load_path = filedialog.askopenfilename(title="Load Configuration",
                                               filetypes=[("INI files", "*.ini")])
        if load_path and os.path.exists(load_path):
            config = ConfigReader(load_path)

            # Load left image parameters
            self.left_image_path = config.getImageName()
            if self.left_image_path and os.path.exists(self.left_image_path):
                self.left_image_label.config(text=os.path.basename(self.left_image_path))
            self.left_projected_image_width.delete(0, tk.END)
            self.left_projected_image_width.insert(0, config.getProjectedImageWidth())
            self.left_projected_overlap_width.delete(0, tk.END)
            self.left_projected_overlap_width.insert(0, config.getProjectedOverlapWidth())
            self.left_gamma.delete(0, tk.END)
            self.left_gamma.insert(0, config.getGamma())
            self.left_image_side.set(config.getImageSide())
            self.left_transparency_factor.delete(0, tk.END)
            self.left_transparency_factor.insert(0, config.getTransparencyFactor())

            # Load right image parameters
            self.right_image_path = config.getImageName()
            if self.right_image_path and os.path.exists(self.right_image_path):
                self.right_image_label.config(text=os.path.basename(self.right_image_path))
            self.right_projected_image_width.delete(0, tk.END)
            self.right_projected_image_width.insert(0, config.getProjectedImageWidth())
            self.right_projected_overlap_width.delete(0, tk.END)
            self.right_projected_overlap_width.insert(0, config.getProjectedOverlapWidth())
            self.right_gamma.delete(0, tk.END)
            self.right_gamma.insert(0, config.getGamma())
            self.right_image_side.set(config.getImageSide())
            self.right_transparency_factor.delete(0, tk.END)
            self.right_transparency_factor.insert(0, config.getTransparencyFactor())

            messagebox.showinfo("Success", f"Configuration loaded from {load_path}")
            logging.info(f"Configuration loaded from {load_path}")
        else:
            messagebox.showerror("Error", "Failed to load configuration.")
```

```
    def process_and_save(self):
        # Validate inputs
        if not self.left_image_path or not self.right_image_path:
            messagebox.showerror("Error", "Please select both left and right images.")
            return

        # Collect parameters for left image
        left_params = {
            'image_name': self.left_image_path,
            'projected_image_width': int(self.left_projected_image_width.get()),
            'projected_overlap_width': int(self.left_projected_overlap_width.get()),
            'gamma': float(self.left_gamma.get()),
            'image_side': self.left_image_side.get(),
            'transparency_factor': float(self.left_transparency_factor.get())
        }

        # Collect parameters for right image
        right_params = {
            'image_name': self.right_image_path,
            'projected_image_width': int(self.right_projected_image_width.get()),
            'projected_overlap_width': int(self.right_projected_overlap_width.get()),
            'gamma': float(self.right_gamma.get()),
            'image_side': self.right_image_side.get(),
            'transparency_factor': float(self.right_transparency_factor.get())
        }

        # Initialize main display
        main_display = MainDisplay()

        # Process left image
        self.progress_bar["value"] = 0
        self.root.update_idletasks()
        left_processed, left_name = process_image(self.left_image_path, left_params, main_display)
        if left_processed is not None and left_name is not None:
            self.processed_images[left_name] = left_processed
            save_image(left_processed, left_name)
            self.progress_bar["value"] += 50
            self.root.update_idletasks()

        # Process right image
        right_processed, right_name = process_image(self.right_image_path, right_params, main_disp
lay)
        if right_processed is not None and right_name is not None:
            self.processed_images[right_name] = right_processed
            save_image(right_processed, right_name)
            self.progress_bar["value"] += 50
            self.root.update_idletasks()

        # Display images
        self.display_preview()

        # Finalize progress
        self.progress_bar["value"] = 100
        self.root.update_idletasks()
        messagebox.showinfo("Processing Complete", "Images processed and saved successfully.")

    def display_preview(self):
        # Display the first processed image as preview
        if self.processed_images:
            # For simplicity, display the first image
            name, img = next(iter(self.processed_images.items()))
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            pil_img = Image.fromarray(img_rgb)
            pil_img = pil_img.resize((400, 300), Image.ANTIALIAS)
            img_tk = ImageTk.PhotoImage(pil_img)
            self.processed_image_label.config(image=img_tk)
            self.processed_image_label.image = img_tk
```

```
            # Load and display original image
            original_img = cv2.imread(self.left_image_path if name == 'left' else self.right_image
_path, cv2.IMREAD_COLOR)
            if original_img is not None:
                original_rgb = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
                pil_original = Image.fromarray(original_rgb)
                pil_original = pil_original.resize((400, 300), Image.ANTIALIAS)
                original_tk = ImageTk.PhotoImage(pil_original)
                self.original_image_label.config(image=original_tk)
                self.original_image_label.image = original_tk

    def display_image(self, processed_images):
        # This method can be expanded to display multiple images if needed
        pass
```

History

#1 - 10/24/2024 03:25 PM - MYAT Ma De May Phuu Ngon

- Assignee set to Andifallih Noor MALELA

This is the early version of the UI code, needs to be fixed to be applied into current program (alpha_blending_v2.py)

#2 - 10/24/2024 03:25 PM - MYAT Ma De May Phuu Ngon

- Due date set to 10/31/2024

- Estimated time set to 2.00 h

#3 - 10/30/2024 11:10 PM - Andifallih Noor MALELA

- Status changed from New to In Progress

- % Done changed from 0 to 50

```
def process_image(image_path, params, main_display):
    """
    Processes an image based on the provided parameters.

    Args:
        image_path (str): Path to the image file.
        params (dict): Parameters for image processing.
        main_display (MainDisplay): Instance of MainDisplay for image operations.

    Returns:
        tuple: Processed image and its corresponding name.
    """

    # Retrieve parameters from params dictionary
    mask_width = params.get('projected_overlap_width')
    image_width = params.get('projected_image_width')
    gamma = params.get('gamma')
    image_side = params.get('image_side')
    transparency_factor = params.get('transparency_factor')

    # Determine image side name
    if image_side == 0:
        image_name = 'left'
    elif image_side == 1:
        image_name = 'right'
    else:
        logging.error(f"Invalid ImageSide value. Use 0 for left image, 1 for right image.")
        return None, None

    # Load image
    image = main_display.readImage(image_path)

    if image is None:
        logging.error(f"Image loading failed for {image_path}. Skipping...")
        return None, None

    # Initialize result image
```

```
    result_image = main_display.setImage(image).copy()
    cv2.imwrite("initial_result_image.png", result_image)  # Save initial image

    # Initialize MaskCreator
    mask_creator = MaskCreator(image, transparency_factor)

    # Apply image modifications
    mask_creator.create_mask(image_side, mask_width, image_width)
    mask_creator.gammaCorrection(gamma)
    mask_creator.result_image = result_image
    mask_creator.alpha_blending(image_side)

    # Save final result for inspection
    cv2.imwrite("final_result_image.png", mask_creator.result_image)

    return mask_creator.result_image, image_name
```

#4 - 10/31/2024 02:12 PM - Andifallih Noor MALELA

- File G12_Code.zip added

- % Done changed from 50 to 100

Latest code version

#5 - 10/31/2024 03:31 PM - Andifallih Noor MALELA

- File G12_Code.zip added

- Status changed from In Progress to Resolved

Uniformed the method names

#6 - 10/31/2024 03:42 PM - Andifallih Noor MALELA

- File alpha_blending.py added

Final code with uniformed method names

Files

G12_Code.zip	4.81 MB	10/31/2024	Andifallih Noor MALELA
G12_Code.zip	4.81 MB	10/31/2024	Andifallih Noor MALELA
alpha_blending.py	23.8 KB	10/31/2024	Andifallih Noor MALELA