21110185\_HoVanHuynhHop\_BtTuan01

HW1) Thu thập (tìm trên mạng hoặc tự chụp ảnh) tối thiểu 10 ảnh phong cảnh

(độ phân giải 566x1080 pixels), sau đó lưu mỗi ảnh thành 3 file theo các định

dạng PNG, BMP, JPG.

HW2) Viết chương trình bằng Python hoặc C++

• Đọc ảnh và hiển thị các ảnh đã tạo ở HW1.

• Tách ảnh RGB thành 3 lớp R, G, B và hiển thị chúng

• Chuyển ảnh RGB thành ảnh đa mức xám (gray scale)

• Read image set and show each image on each window.

• Quay ảnh 100 lần: mỗi lần quay 10 độ, và tạm dừng 0.1 giây, hiển thị

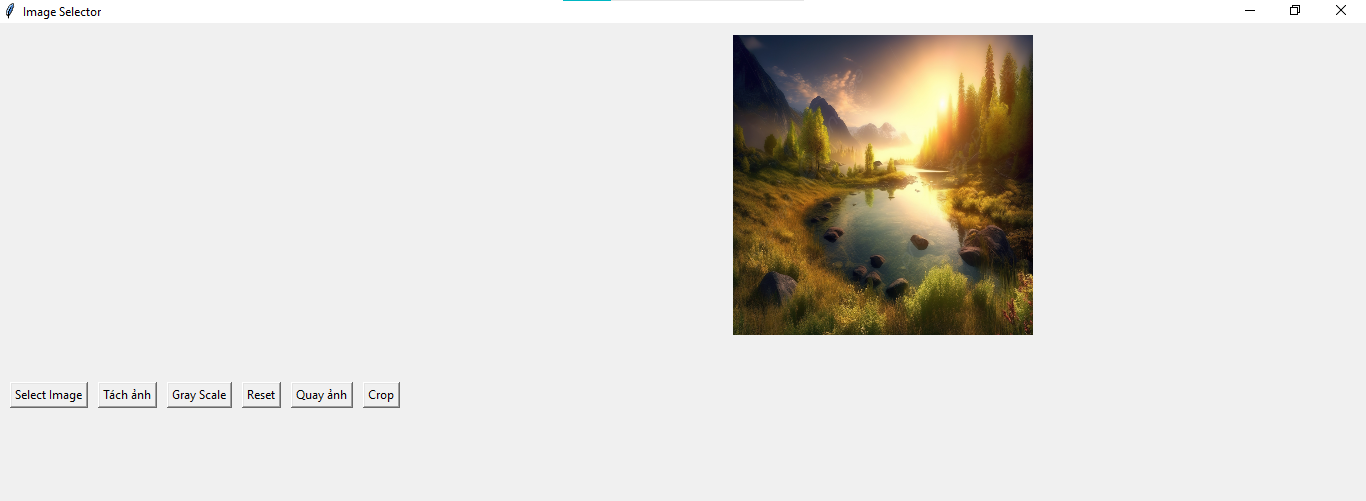
trên cùng cửa sổ.

• Đọc vào 1 ảnh, sau đó crop 1⁄4 ảnh tính từ tâm ảnh và hiển thị ảnh trước

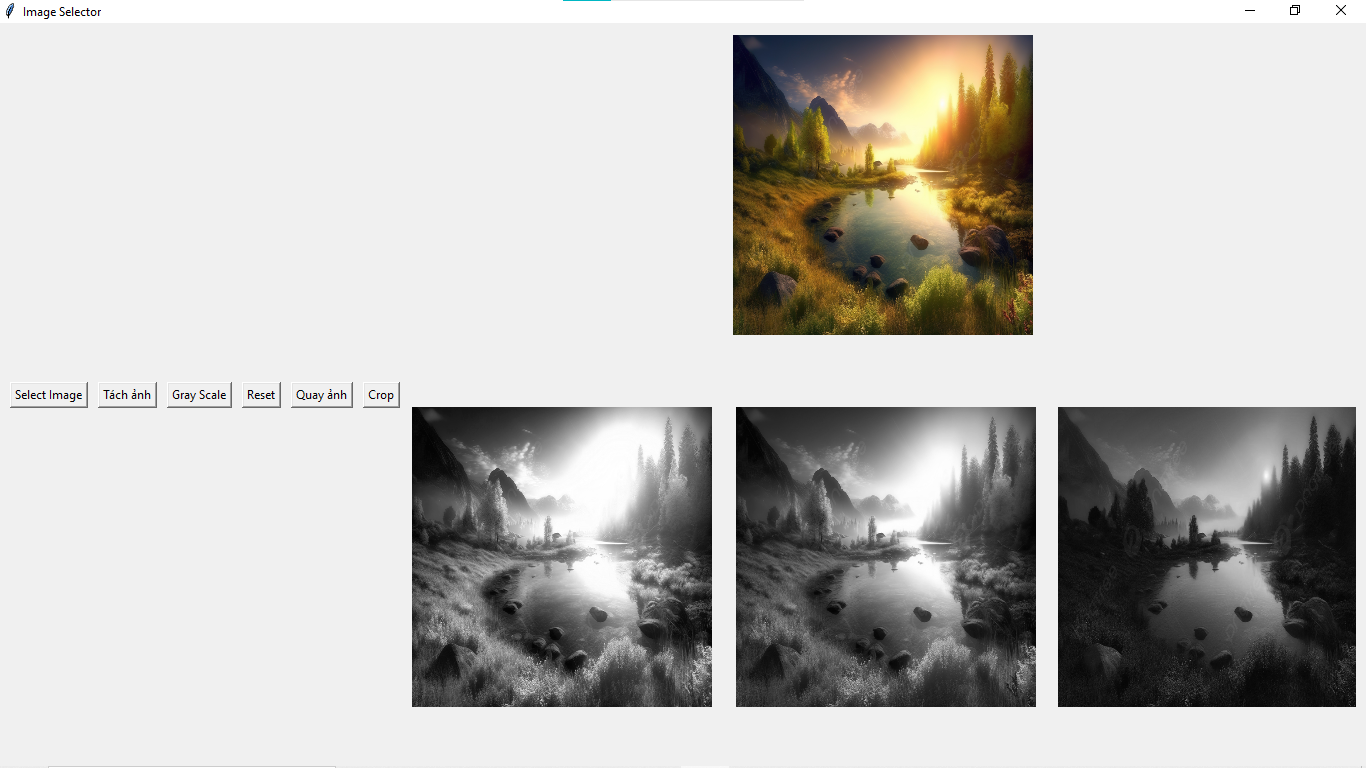
và sau khi crop.

\*Kết quả:

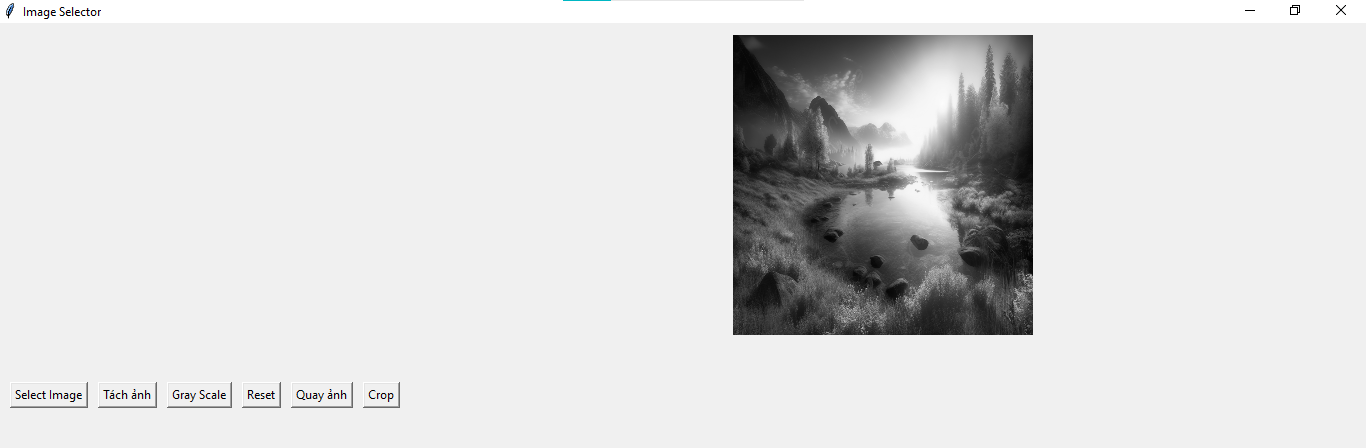
Giao diện:



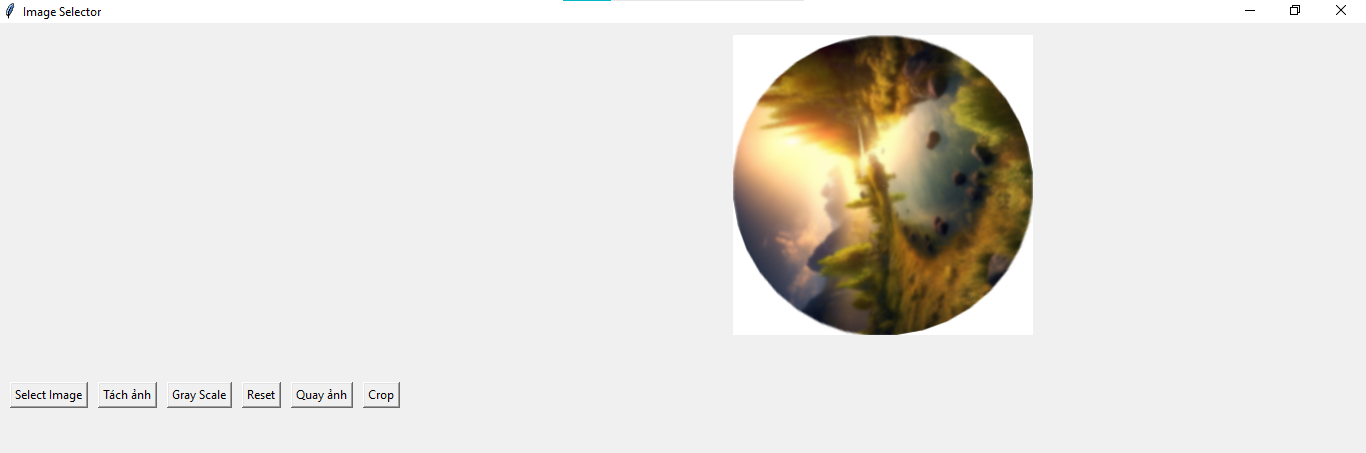
Tách ảnh:



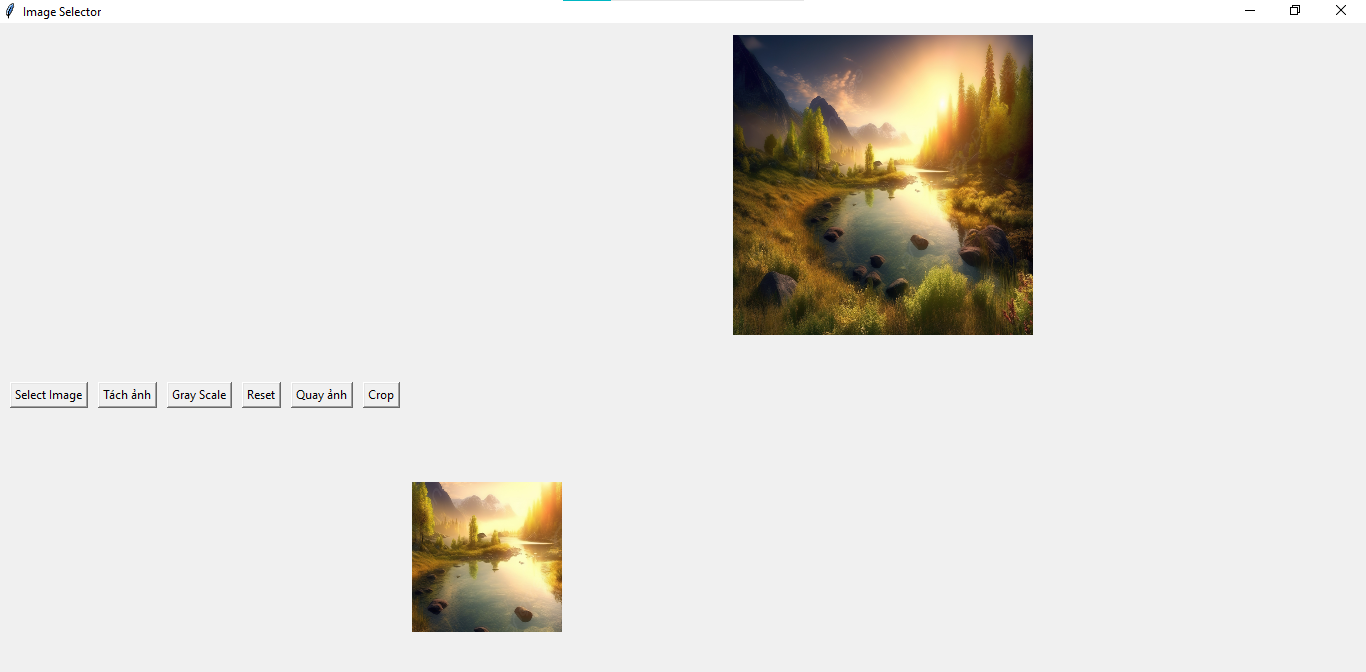
Gray Scale:



Quay ảnh:



Crop ảnh:



\* Code:

import cv2

from PIL import Image, ImageTk

import tkinter as tk

from tkinter import filedialog

import time

import numpy as np

from threading import Thread

class ImageSelectorApp:

    def \_\_init\_\_(self, master):

        self.master = master

        self.master.title("Image Selector")

        # Tạo nút "Select Image" để chọn ảnh

        self.select\_button = tk.Button(master, text="Select Image", command=self.choose\_image)

        self.select\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Tách ảnh" để thực hiện quá trình tách ảnh

        self.split\_button = tk.Button(master, text="Tách ảnh", command=self.split\_image)

        self.split\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Gray Scale" để chuyển đổi ảnh thành ảnh đa mức xám

        self.gray\_button = tk.Button(master, text="Gray Scale", command=self.convert\_to\_gray)

        self.gray\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Reset" để xóa toàn bộ ảnh đã hiển thị

        self.reset\_button = tk.Button(master, text="Reset", command=self.reset\_images)

        self.reset\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Quay ảnh" để thực hiện chức năng quay ảnh

        self.rotate\_button = tk.Button(master, text="Quay ảnh", command=self.rotate\_images)

        self.rotate\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Crop" để cắt ảnh

        self.crop\_button = tk.Button(master, text="Crop", command=self.crop\_image)

        self.crop\_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo label để hiển thị ảnh gốc

        self.original\_image\_label = tk.Label(master)

        self.original\_image\_label.pack(pady=10, padx=10)  # Điều chỉnh giảm đệm

        # Tạo label để hiển thị các lớp R, G, B

        self.r\_label = tk.Label(master)

        self.r\_label.pack(side="left", pady=10, padx=10)

        self.g\_label = tk.Label(master)

        self.g\_label.pack(side="left", pady=10, padx=10)

        self.b\_label = tk.Label(master)

        self.b\_label.pack(side="left", pady=10, padx=10)

        # Biến instance để kiểm tra trạng thái quay ảnh

        self.is\_rotating = False

        # Biến instance để lưu ảnh gốc

        self.original\_image = None

        # Tiến trình quay ảnh

        self.rotation\_process = None

    # Hàm để chọn ảnh từ thư mục

    def choose\_image(self):

        file\_path = filedialog.askopenfilename()

        if file\_path:

            self.display\_original\_image(file\_path)

    # Hàm để hiển thị ảnh gốc

    def display\_original\_image(self, file\_path):

        # Đọc ảnh sử dụng OpenCV

        original\_image = cv2.imread(file\_path)

        original\_image = cv2.cvtColor(original\_image, cv2.COLOR\_BGR2RGB)  # Chuyển đổi sang màu RGB

        # Thiết lập kích thước ảnh cố định (ví dụ: 300x300)

        target\_size = (300, 300)

        original\_image = cv2.resize(original\_image, target\_size)

        # Lưu ảnh gốc vào biến instance

        self.original\_image = original\_image

        # Hiển thị ảnh gốc trong label

        original\_image\_pil = Image.fromarray(original\_image)

        original\_image\_tk = ImageTk.PhotoImage(original\_image\_pil)

        self.original\_image\_label.config(image=original\_image\_tk)

        self.original\_image\_label.image = original\_image\_tk

    # Hàm để tách ảnh thành các lớp R, G, B và hiển thị

    def split\_image(self):

        # Kiểm tra xem đã chọn ảnh chưa

        if self.original\_image is None:

            return

        # Lấy ảnh gốc từ biến instance

        original\_image = self.original\_image

        # Tách thành các lớp R, G, B

        r, g, b = cv2.split(original\_image)

        # Hiển thị các lớp R, G, B trong label

        r\_pil = Image.fromarray(r)

        r\_tk = ImageTk.PhotoImage(r\_pil)

        self.r\_label.config(image=r\_tk)

        self.r\_label.image = r\_tk

        g\_pil = Image.fromarray(g)

        g\_tk = ImageTk.PhotoImage(g\_pil)

        self.g\_label.config(image=g\_tk)

        self.g\_label.image = g\_tk

        b\_pil = Image.fromarray(b)

        b\_tk = ImageTk.PhotoImage(b\_pil)

        self.b\_label.config(image=b\_tk)

        self.b\_label.image = b\_tk

    # Hàm để chuyển ảnh thành ảnh đa mức xám (gray scale) và hiển thị

    def convert\_to\_gray(self):

        # Kiểm tra xem đã chọn ảnh chưa

        if self.original\_image is None:

            return

        # Lấy ảnh gốc từ biến instance

        original\_image = self.original\_image

        # Chuyển ảnh sang ảnh đa mức xám

        gray\_image = cv2.cvtColor(original\_image, cv2.COLOR\_RGB2GRAY)

        # Hiển thị ảnh đa mức xám trong label

        gray\_image\_pil = Image.fromarray(gray\_image)

        gray\_image\_tk = ImageTk.PhotoImage(gray\_image\_pil)

        self.original\_image\_label.config(image=gray\_image\_tk)

        self.original\_image\_label.image = gray\_image\_tk

    # Hàm để xóa toàn bộ ảnh đã hiển thị

    def reset\_images(self):

        # Kiểm tra xem đang trong quá trình quay ảnh hay không

        if self.is\_rotating == False:

            # self.rotation\_process.join()  # Đợi tiến trình quay ảnh kết thúc trước khi tiếp tục

            self.original\_image\_label.config(image=None)

            self.original\_image\_label.image = None

        # Xóa ảnh gốc

        self.original\_image\_label.config(image=None)

        self.original\_image\_label.image = None

        # Xóa các lớp R, G, B

        self.r\_label.config(image=None)

        self.r\_label.image = None

        self.g\_label.config(image=None)

        self.g\_label.image = None

        self.b\_label.config(image=None)

        self.b\_label.image = None

    # Hàm để quay ảnh

    def rotate\_images(self):

        # Kiểm tra xem đã chọn ảnh chưa

        if self.original\_image is None:

            return

        # Kiểm tra xem đang trong quá trình quay ảnh hay không

        if self.rotation\_process and self.rotation\_process.is\_alive():

            return

        # Đặt biến cờ báo hiệu đang trong quá trình quay ảnh

        self.is\_rotating = True

        # Lấy ảnh gốc từ biến instance

        original\_image = self.original\_image.copy()

        # Bắt đầu tiến trình quay ảnh

        self.rotation\_process = Thread(target=self.rotate\_images\_thread, args=(original\_image,))

        self.rotation\_process.start()

    # Hàm chạy trong tiến trình để quay ảnh

    def rotate\_images\_thread(self, original\_image):

        # Thực hiện quay ảnh 100 lần, mỗi lần quay 10 độ và tạm dừng 0.1 giây

        for \_ in range(10):

            original\_image = self.rotate\_image(original\_image, angle=10)

            self.display\_rotated\_image(original\_image)

            self.master.update()

            time.sleep(0.1)

        # Đặt biến cờ trạng thái quay ảnh về False sau khi kết thúc

        self.is\_rotating = False

    # Hàm để quay ảnh một góc nhất định

    def rotate\_image(self, image, angle):

        rows, cols, \_ = image.shape

        M = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)

        rotated\_image = cv2.warpAffine(image, M, (cols, rows), borderMode=cv2.BORDER\_CONSTANT, borderValue=(255, 255, 255))

        return rotated\_image

    # Hàm để hiển thị ảnh sau khi quay

    def display\_rotated\_image(self, image):

        # Hiển thị ảnh gốc trong label

        rotated\_image\_pil = Image.fromarray(image)

        rotated\_image\_tk = ImageTk.PhotoImage(rotated\_image\_pil)

        self.original\_image\_label.config(image=rotated\_image\_tk)

        self.original\_image\_label.image = rotated\_image\_tk

    # Hàm để cắt ảnh

    def crop\_image(self):

        # Kiểm tra xem đã chọn ảnh chưa

        if self.original\_image is None:

            return

        # Lấy ảnh gốc từ biến instance

        original\_image = self.original\_image

        # Lấy kích thước ảnh gốc

        height, width, \_ = original\_image.shape

        # Tính toán kích thước cần cắt (1/4 ảnh tính từ tâm ảnh)

        new\_width = int(width / 2)

        new\_height = int(height / 2)

        # Tính toán vị trí bắt đầu của ảnh cắt

        start\_x = int(width / 4)

        start\_y = int(height / 4)

        # Cắt ảnh

        cropped\_image = original\_image[start\_y:start\_y + new\_height, start\_x:start\_x + new\_width]

        # Hiển thị ảnh trước khi cắt

        original\_image\_pil = Image.fromarray(original\_image)

        original\_image\_tk = ImageTk.PhotoImage(original\_image\_pil)

        self.original\_image\_label.config(image=original\_image\_tk)

        self.original\_image\_label.image = original\_image\_tk

        # Hiển thị ảnh sau khi cắt

        cropped\_image\_pil = Image.fromarray(cropped\_image)

        cropped\_image\_tk = ImageTk.PhotoImage(cropped\_image\_pil)

        self.r\_label.config(image=cropped\_image\_tk)

        self.r\_label.image = cropped\_image\_tk

# Tạo cửa sổ tkinter

root = tk.Tk()

# Tạo đối tượng ứng dụng

app = ImageSelectorApp(root)

# Main loop của cửa sổ tkinter

root.mainloop()