МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Информатика»
Тема: Введение в архитектуру компьютера

Студент гр. 3343 _____ Коршков А.А.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2023

Цель работы

Научится работать с модулем Pillow (PIL), а также с функциями numpy, выполнять различные графические преобразования над изображениями.

Задание

Предстоит решить 3 подзадачи, используя библиотеку Pillow (PIL). Для реализации требуемых функций студент должен использовать numpy и PIL. Аргумент image в функциях подразумевает объект типа <class 'PIL.Image.Image'>

1) Рисование треугольника

Необходимо написать функцию triangle(), которая рисует на изображении треугольник

Функция triangle() принимает на вход:

Изображение (img)

Координаты вершин (x0,y0,x1,y1,x2,y2)

Толщину линий (thickness)

Цвет линий (color) - представляет собой список (list) из 3-х целых чисел

Цвет, которым залит (fill_color - если значение None, значит треугольник не залит) - представляет собой список (list) из 3-х целых чисел

Функция должна вернуть исходное обработанное изображение.

2) Замена наиболее часто встречаемого цвета.

Необходимо написать функцию change_color(), которая заменяет наиболее часто встречаемый цвет на переданный.

Функция change_color() принимает на вход:

Изображение (img)

Цвет (color - представляет собой список из трех целых чисел)

Функция должна найти в изображении самый частый цвет и заменить его на переданный, затем вернуть новое изображение (исходное изображение не должно меняться).

3) Коллаж

Необходимо написать функцию collage().

Функция collage() принимает на вход:

Изображение (img)

Количество изображений по "оси" Y (N - натуральное)

Количество изображений по "оси" X (M - натуральное)

Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и вернуть его (новое изображение).

При необходимости можно писать дополнительные функции.

Выполнение работы

В функции triangle необходимо с помощью модуля ImageDraw создать объект для рисования drawing. У него вызывается функция polygon, которая позволяет на основе заданных точек построить какую-то фигуру. Т.к. передаётся в функцию три точки, эта функция создаёт треугольник на изображении. Делаем проверку, что fill_color не None, чтобы нарисовать треугольник с заливкой или без, затем возвращаем изменённое изображение.

Функция change_color получает размеры изображения с помощью атрибута size, затем в переменную pixels_array записывается массив из цветов для каждого пикселя. С помощью функции unique можно найти уникальные цвета, а также кол-во их на изображении и записать в отдельную переменную counts. С помощью np.where(counts == np.max(counts)) можно найти индекс цвета, который часто встречается на изображении. Для более удобной замены цвета воспользуемся функцией tolist и tuple, чтобы преобразовать значение цвета в кортеж. Затем воспользуемся copy(), чтобы создать копию изображения. После чего необходимо с помощью вложенных циклов проверить, что цвет пикселя самый встречаемый и заменить его на заданный в функции цвет. Затем возвращаем изменённое изображение.

В функции collage создаёт чистое полотно, где длина равна x*M (x – длина исходного изображения, M – количество изображений по горизонтали), а высота y * N (y – высота исходного изображения, N – количество изображений по вертикали). Затем с помощью вложенных циклов вставляем исходное изображение на полотно, передвигая его начальную координату на расстояние x и y соответственно.

Выводы

Были изучены различные способы преобразования изображения, написаны функции с использованием библиотек numpy и pillow. Также применены конструкции if-else, for, copy.

Программа может получать на вход исходное изображать и возвращать изменённое, в зависимости от выбранной функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
from PIL import Image, ImageDraw


def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color,
fill_color):
    drawing = ImageDraw.Draw(img)
    if fill_color:
        drawing.polygon([(x0,   y0),   (x1,   y1),   (x2,   y2)],
fill=tuple(fill_color),
                        width=thickness, outline=tuple(color))
    else:
        drawing.polygon([(x0, y0), (x1, y1), (x2, y2)],
                        width=thickness, outline=tuple(color))
    return img


def change_color(img, color):
    x, y = img.size
   pixels_array = np.array([img.getpixel((i, j)) for i in range(x)
for j in range(y)])
    pixels_array,   counts   =   np.unique(pixels_array,   axis=0,
return_counts=True)
    freq_color = tuple(pixels_array[np.where(counts ==
                                np.max(counts))][0].tolist())
    img_new = img.copy()
    pixels = img_new.load()
    for i in range(x):
        for j in range(y):
            if pixels[i, j] == freq_color:
                pixels[i, j] = tuple(color)
    return img_new


def collage(img, N, M):
    x, y = img.size
    img_new = Image.new("RGB", (x * M, y * N))
    for i in range(N):
        for j in range(M):
            img_new.paste(img, (x * j, y * i))
    return img_new
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Исходные изображения:

Рисунок 1 – Изображение для функции triangle

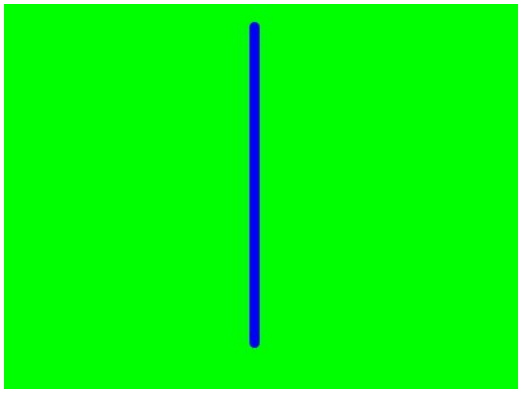

Рисунок 2 – Изображение для функции change_color

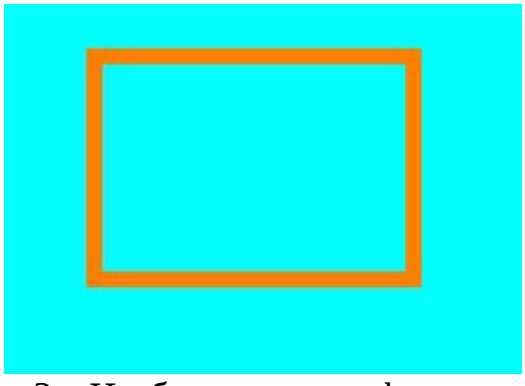

Рисунок 3 – Изображение для функции collage

Параметры функций:
1) Для рисунка 1 функция
triangle(img, 10, 14, 200, 75, 100, 100, 2, (234, 56, 67), (33, 64, 21))
2) Для рисунка 2 функция
change_color(img, (56, 43, 12))
3)Для рисунка 3 функция

collage(img, 2, 3)

Результат:

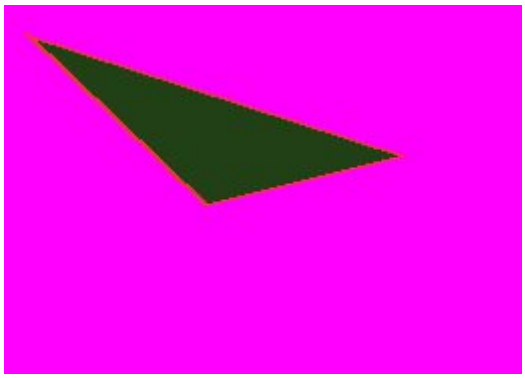

Рисунок 1 – Результат функции triangle

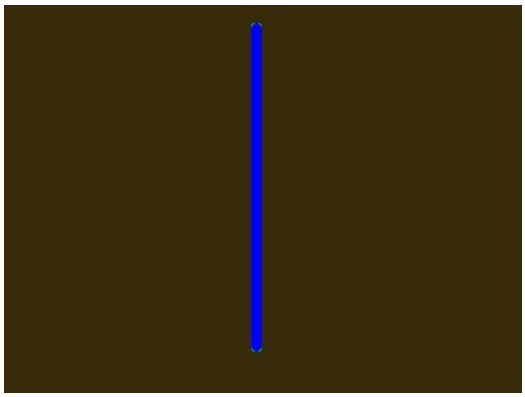

Рисунок 2 – Результат функции change_color

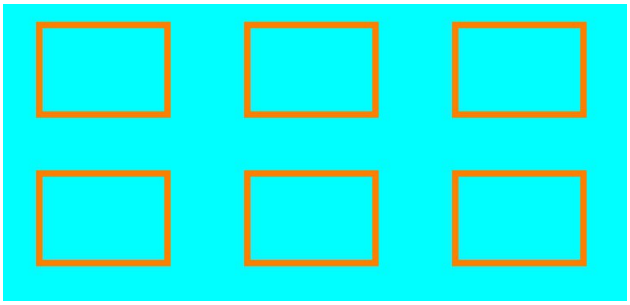

Рисунок 3 – Результат функции collage