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. 2381 _____ Долотов Н.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2022

Цель работы.

Изучить библиотеку Pillow для работы с графическими данными, получить практические навыки использования данной библиотеки для работы с изображениями.

Задание.

Вариант 1

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.

Для решения поставленных задач были написаны 3 функции с использованием библиотеки Pillow для работы с изображениями.

Функция *triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color)*

Функция рисует на изображении треугольник по заданным входным параметрам. Функция принимает на вход изображение (*img*), координаты вершин треугольника (*x0,y0,x1,y1,x2,y2*), толщину линий (*thickness*), цвет линий (*color*), цвет заливки (*fill_color*). Условие *if()* проверяет, нужно ли заливать треугольник цветом, и если нужно, то преобразует список из 3 чисел в кортеж. Функция *ImageDraw.Draw()* создаёт объект *ImageDraw*, на котором можно рисовать различные 2D фигуры. Метод *.polygon()* рисует на заданном изображении многоугольник с заданными входными параметрами: координаты точек, цвет и толщина линий, цвет заливки. Функция возвращает изображение с нарисованным треугольником.

Функция *change_color(img, color)*

Функция принимает на вход изображение (*img*) и заменяет наиболее часто встречаемый цвет на переданный (*color*). *.size* – метод, который возвращает

кортеж типа (ширина изображения, высота изображения). С помощью цикла *for* в цикле *for* осуществляется подсчёт пикселей различных цветов в изображении: цвет каждого оригинального пикселя – ключ в словаре *pixel_colors*, значение – количество пикселей данного цвета в изображении. *the_most_common_color_val_key* – ключ наиболее часто встречаемого цвета. Далее с помощью цикла *for* в цикле *for* производится проверка каждого цвета пикселя на совпадение с наиболее часто встречаемым цветом и последующая замена на переданный в функцию цвет. Функция возвращает изменённое изображения, где наиболее часто встречаемый пиксель цвета заменён на другой цвет.

Функция *collage(img, N, M)*

Функция создаёт коллаж изображений (*NxM*); принимает на вход изображение (*img*), количество изображений по "оси" Y и X (*N* и *M*). *new_img* – пустое изображение требуемого для коллажа размера (*img_width * M, img_height * N*). Цикл *for* в цикле *for* добавляет в список *coordinates* кортеж координат левого верхнего угла изображения. Далее цикл *for* с помощью метода *.paste()* вставляет исходное изображение (*img*) в новое (*new_img*) *M*N* раз, используя значения из списка *coordinates*. Функция возвращает коллаж изображений.

Разработанный код см. в приложении А.

Выводы.

Были изучены методы и функции из библиотеки Pillow для работы с изображениями и получены практические навыки работы с графическими данными.

Разработана программа, содержащая 3 функции: функция triangle(), рисующая треугольник на изображении, функция change_color(), заменяющая наиболее часто встречаемый в изображении цвет на другой цвет и функция collage(), создающая коллаж определённого размера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw


def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color):
    if fill_color is not None:
        fill_color = tuple(fill_color)
    drawing = ImageDraw.Draw(img)
    drawing.polygon((x0, y0, x1, y1, x2, y2), fill=fill_color,
outline=tuple(color), width=thickness)
    return img


def change_color(img, color):
    pixel_colors = {}
    width, height = img.size
    pixels = img.load()
    for i in range(width):
        for j in range(height):
            pixel_color = str(pixels[i, j])
            if pixel_color not in pixel_colors:
                pixel_colors[pixel_color] = 1
            else:
                pixel_colors[pixel_color] += 1
    the_most_common_color_val_key = max(pixel_colors,
key=pixel_colors.get)
    for i in range(width):
        for j in range(height):
            if str(pixels[i, j]) == the_most_common_color_val_key:
                pixels[i, j] = tuple(color)
    return img


def collage(img, N, M):
    img_width, img_height = img.size
    new_img_width, new_img_height = img_width * M, img_height * N
    new_img = Image.new("RGB", (new_img_width, new_img_height), (0, 0, 0))
    coordinates = []
    for i in range(M):
        for j in range(N):
            coordinates.append((i * img_width, j * img_height))
    for k in range(M * N):
        new_img.paste(img, coordinates[k])
    return new_img
```