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 _____ Мавликаев И.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Цель работы – изучить методы библиотеки Pilow, решив 3 подзадачи.

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 полученное изображение.

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.

Подключатся библиотека *numpy* и библиотека *PIL.*

В библиотеке *PIL* подключаются модули *Image, ImageDraw.*

Задача 1.

Функция *triangle()* принимает на вход: изображение (*img*), координаты вершин (*x0,y0,x1,y1,x2,y2*), толщину линий *(thickness*), цвет линий (*color*) – представляет собой список из 3-х целых чисел, цвет, которым залит (*fill_color* – если значение *None*, значит треугольник не залит) – представляет собой список из 3-х целых чисел.

Создаётся объект для рисования *drawing = ImageDraw.Draw(img).*

Проверяется, залит ли треугольник (равен ли *fill_color None*), затем с помощью метода *ImageDraw.polygon(...)* на объекте для рисования drawing рисуется треугольник.

Возвращается исходное обработанное изображение.

Задача 2.

Функция *change_color()* принимает на вход: изображение (*img*), цвет (*color* – представляет собой список из трех целых чисел).

Создаётся новое изображение, такое же как поданное на вход:

img1 = img

Создаются переменные x и y – длины изображения *img* по осям X и Y с помощью функции *size()*. Создаётся словарь *count_pix*. В цикле for проход по объекту, полученному из *img1* с помощью метода *getdata()* (последовательности, каждый элемент которого – массив из трёх элементов –

цвет пикселя). Если в ключах словаря *count_pix* нет данного цвета, он добавляется в словарь со значением 1, иначе по ключу значение увеличивается на 1.

С помощью функции *max* находится наиболее распространённый цвет в изображении (ключ в словаре *count_pix,* по которому хранится самое большое значение) и записывается в переменную max_color.

max_color = max(count_pix, key=count_pix.get)

Создаётся объект для рисования *drawing = ImageDraw.Draw(img1).*

В двойном цикле *for* с помощью метода *getpixel* проверяется каждый пиксель, и если он цвета *max_color*, методом с помощью метода он закрашивается цветом *color*.

drawing.point((i, j), tuple(color))

Функция возвращает изображение *img1*.

Задача 3.

Функция *collage()* принимает на вход: изображение (*img*), количество изображений по "оси" Y (N - натуральное), количество изображений по "оси" X (M – натуральное).

С помощью функции *size()* находятся размеры изображения, которые записываются в переменные *width* и *height*.

Создаётся новое изображение *img1* размера *width*M* на *height*N* пикселей белого цвета *img1 = Image.new("RGB", (width * M, height * N), 'white').* С помощью функции *size()* находятся размеры нового изображения, которые записываются в переменные *width1* и *height1*.

В цикле *for* счётчик *i* увеличивается с 0 до *width1* с шагом *width*. Во внутреннем цикле *for* счётчик *j* увеличивается с 0 до *height1* с шагом *height*. В новое изображение *img1* на координаты *i, j* вставляется старое изображение *img1.paste(img, (i, j)).*

Функция возвращает изображение *img1*.

Выводы.

Были изучены методы библиотеки Pilow. Разработаны три подпрограммы: функция, которая рисует на изображении треугольник, функция, которая заменяет наиболее часто встречаемый цвет на переданный, функция, создающая коллаж изображений. Были использованы методы *ImageDraw.polygon(coordinates, fill, outline, width),* для рисования треугольника, *getpixel(coordinates)* для получения цвета пикселя*, ImageDraw.point(coordinates, color)* для рисования пикселя*, Image.getdata()* для получения объекта последовательности, содержащего значения пикселей, *img1.paste(img, (i, j))* для вставления старого изображения на новое.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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


# Задача 1
def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color):
    drawing = ImageDraw.Draw(img)
    if fill_color == None:
        drawing.polygon([(x0,y0), (x1, y1), (x2,y2)], fill = None, outline = tuple(color), width=thickness)
    else:
        drawing.polygon([(x0,y0), (x1, y1), (x2,y2)], fill = tuple(fill_color), outline = tuple(color), width=thickness)
    return img


# Задача 2
def change_color(img, color):
    img1 = img
    x, y = img.size
    count_pix = dict()
    for i in img1.getdata():
        if i in count_pix:
            count_pix[i] += 1
        else:
            count_pix[i] = 1
    max_color = max(count_pix, key=count_pix.get)
    drawing = ImageDraw.Draw(img1)
    for i in range (x):
        for j in range (y):
            if img1.getpixel((i, j)) == max_color:
                drawing.point((i, j), tuple(color))
    return img1



# Задача 3
def collage(img, N, M):
    width, height = img.size
    img1 = Image.new("RGB", (width * M, height * N), 'white')
    width1, height1 = img1.size
    for i in range (0, width1, width):
      for j in range (0, height1, height):
        img1.paste(img, (i, j))
    return img1
```