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 Тищенко А. М.

Преподаватель Шевская Н. В.

Санкт-Петербург
2022

Цель работы.

Изучить архитектуру компьютера, а также модуль Pillow.

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'>

Задача 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.

Задача 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.

Задача 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.

Задача 1: переменные: *drawing* объект для отрисовки треугольника на изображении

Программа рисует треугольник в соответствии с условием.

Задача 2: переменные: *arr* – копия изображения в виде массива, *i, j* для хождения по массиву, *colors* для хранения всех цветов и количеств соответствующих пикселей, *px_color* цвет конкретного пикселя, *orig_color* – цвет, который нужно заменить.

Программа проходиться по всем пикселям в изображении, и считает количество пикселей одного цвета, после заменяет их и возвращает измененное изображение

Задача 3: переменные: *arr* – копия изображения в виде массива, *i* для хождения по массиву, *arr_buf* массив для создания коллажа

Программа добавляет к массиву из изображения, пока не получит коллаж, затем возвращает его.

Разработанный программный код см. в приложении А.

Выводы.

Были изучены архитектура компьютера, а также модуль Pillow. Разработаны функции, рисующая треугольник, заменяющая самый часто встречаемый цвет, и создающая коллаж. Первая с помощью *ImageDraw* рисовала треугольник. Вторая считала и заменяла цвета. Третья, объеденяя массивы, создавала коллаж.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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


# Задача 1
def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color)
-> Image:
    drawing = ImageDraw.Draw(img)
    if fill_color is not None:
        fill_color = tuple(fill_color)
    color = tuple(color)
    drawing.polygon(((x0, y0), (x1, y1), (x2, y2)), fill_color, color,
thickness)
    return img


# Задача 2
def change_color(img: Image, color) -> Image:
    arr = np.asarray(img).copy()
    colors = {}
    color = tuple(color)
    for i in range(len(arr)):
        for j in range(len(arr[0])):
            px_color = tuple(arr[i][j])
            if px_color in colors:
                colors[px_color] += 1
            else:
                colors[px_color] = 1
    orig_color = max(colors, key=colors.get)
    for i in range(len(arr)):
        for j in range(len(arr[0])):
            px_color = tuple(arr[i][j])
            if px_color == orig_color:
                arr[i][j] = np.array(color)
    return Image.fromarray(arr)


# Задача 3
def collage(img: Image, N, M) -> Image:
    arr = np.asarray(img).copy()
    arr_buf = arr.copy()
    for i in range(M-1):
        arr = np.hstack((arr, arr_buf))
    arr_buf = arr.copy()
    for i in range(N-1):
        arr = np.vstack((arr, arr_buf))
    return Image.fromarray(arr)
```