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

Студентка гр. 3343 _____ Синицкая Д.В.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Научиться работать с функциями библиотеки *Pillow (PIL)*.

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

Функция 1. В решении были использованы: метод *ImageDraw.Draw()* создающий объект для рисования на изображении, список *points*, содержащий координаты вершин треугольника, метод *polygon()* отрисовывающий треугольник.

Функция 2. В решении были использованы: модуль *array()* преобразующий изображение в массив пикселей, функция *reshape(-1, 3)* преобразующая массив в двумерный массив, где каждая строка представляет один пиксель изображения, функция *unique()* находящая все уникальные строки пикселей в массиве и возвращающая их в порядке первого появления, функция *argmax()* возвращающая индекс максимального значения.

Функция 3. В решении были использованы: метод *size()* возвращающий размеры изображения, метод *Image.new()* создающий новый пустой объект изображения, метод *paste()* вставляющий исходное изображение в новое изображение.

Разработанный программный код см. в приложении А.

Выводы

Я приобрела навыки использования функций библиотеки *Pillow (PILL)* языка программирования *python*.

В лабораторной работе было реализованно три функции. Функция *triangle()*, отрисовывающая треугольник на изображении. Функция *change_color()*, заменяющая наиболее часто встречающийся цвет на заданный. Функция *collage()*, создающая коллаж изображений.

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):
    draw = ImageDraw.Draw(img) # объект для рисования на изображении
    points = [(x0, y0), (x1, y1), (x2, y2)] # список, содержащий координаты вершин
    if fill_color is None: # проверка на отстутствие цвета заливки
        korteg=None
    else:
        korteg=tuple(fill_color) # преобразование списка в кортеж
    draw.polygon(points, outline=tuple(color), width=thickness, fill=korteg) # вызов метода для отрисовки треугольника
    return img


# Задача 2
def change_color(img, color):
    pixels = np.array(img) # преобразование в массив пикселей
    r, g, b = color # извлечение значений цветов

    # поиск самого часто встречающегося цвета
    # функция reshape(-1, 3) преобразует массив в двумерный массив, где каждая строка представляет один пиксель изображения
    # функция unique() находит все уникальные строки пикселей в массиве и возвращает их в порядке первого появления
    unique_colors, counts = np.unique(pixels.reshape(-1, 3), axis=0, return_counts=True)
    most_common_color = unique_colors[np.argmax(counts)] # определение самого часто встречающегося цвета

    # замена самого часто встречающегося цвета на заданный цвет
    pixels[np.where((pixels == most_common_color).all(axis=2))] = color

    new_image = Image.fromarray(pixels) # преобразование в обьект изображения
    return new_image

# Задача 3
def collage(img, N, M):
    width, height = img.size # определение ширины и высоты исходного изображения
```

```
            # создание нового пустого изображения, размер которого
позволит создавать повторные размещения
        new_image = Image.new(img.mode, (M * width, N * height))

        for i in range(N):
            for j in range(M):
                # вставка исходного изображения
                new_image.paste(img, (j * width, i * height))

        return new_image
```