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 _____ Антонов Н. Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучение библиотеки Pillow (PIL), работа с ней, использование numpy и PIL вместе.

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

Функция *triangle()*:

1) метод *ImageDraw.Draw()* - создаёт объект для рисования изображения.
2) список *points* - содержит координаты вершин треугольника.
3) метод *polygon()* - отрисовывает треугольник.

Функция *change_color()*:

1) модуль *array()* - преобразует изображение в массив пикселей.
2) функция *reshape()* - преобразует массив в двумерный массив, где каждая строка - это один пиксель изображения.
3) функция *unique()* - находит все уникальные пиксели в массиве и возвращает их.
4) функция *argmax()* - возвращает индекс максимального значения.

Функция *collage()*:

1) метод *size()* - возвращает размеры изображения.
2) метод *Image.new()* - создаёт новый пустой объект изображения.
3) метод *paste()* - вставляет исходное изображение в новое изображение.

Разработанный программный код см. в приложении А.

Выводы

Реализованы 3 функции, работающие с библиотекой Pillow (PIL). Программа содержит функции: *triangle()*, *change_color()*, *collage()*. Из библиотеки Pillow были изучены и применены такие вещи как:

1) *ImageDraw.Draw()* рисующий изображение.
2) *polygon()* отвечающий за изображение треугольника.
3) *reshape()* преобразующий массив в двумерный.
4) *unique()* использовавшийся для поиска индивидуальных пикселей.
5) *argmax()* для поиска максимального значения.
6) *Image.new()* для создания объекта изображения.
7) *paste()* для вставки изображения.

Приложение А
Исходный код программы

```
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)
    ver = [(x0, y0), (x1, y1), (x2, y2)]
    if fill_color is None:
        spis = None
    else:
        spis = tuple(fill_color)
    draw.polygon(ver,              outline=tuple(color),
width=thickness, fill=spis)
    return img

# Задача 2
def change_color(img, color):
    r, g, b = color
    pixels = np.array(img)
    unique_colors, counts = np.unique(pixels.reshape(-1,
3), axis=0, return_counts=True)
    most_common_color = unique_colors[np.argmax(counts)]
    pixels[np.where((pixels                              ==
most_common_color).all(axis=2))] = color
    new_image = Image.fromarray(pixels)
    return new_image

# Задача 3
def collage(img, N, M):
    width, h = img.size
    new_image = Image.new(img.mode, (M * width, N * h))
    for i in range(N):
        for j in range(M):
            new_image.paste(img, (j * width, i * h))
    return new_image
```