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. 3344 _____ Охрименко Д.И.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Освоние работы с библиотекой Pillow: создание изображения, копирование, вставка, изменение цвета, рисование многогранников.

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

После импорта модуля Image и ImageDraw из библиотеки PIL и numpy создаются три функцию, выполняющие по отдельности исходные задачи.

Функция triangle(args) должна построить на полотне пикселей img фигуру треугольника, что осуществляется с помощью функции polygon() из класса ImageDraw. В зависимости от входных данных изменяется и аргументы для функции, которые прежде были преобразованы в тип tuple.

В функции change_color(args) создаётся словарь, в котором накапливается количество цветов каждого пикселя подаваемого изображения img. Создание массива пикселей осуществляется с помощью двойного массива numpy.array. После нахождение необходимого цвета снова проходим двойным циклом по массиву и заменяем данный цвет на поданный. Благодаря функции fromarray() из класса Image преобразуем массив пикселей обратно в изображение.

Collage(img, N, M) — функция, создающая изображение расширенного размера img.size[0] * M на img.size[1] * N из дублированной исходной картинки img. Для реализации использовалась функция paste() экземляра img класса Image.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	triangle(Image.new('RGB', (300, 300),"black"), 50, 50, 50, 250, 250, 150, (1,2,3), (1,3,37))	img	Верный вывод обработанного изображения
2.	change_color(Image.new('RGB',(300, 300),"black") , (123,123,123))	img	Верный вывод обработанного изображения
3.	collage(Image.new('RGB', (300,300),"black"),1,6)	img	Верный вывод обработанного изображения

Выводы

Была изучена библиотека Pillow. В разработанной программе использованы несколько функций из класса Image и ImageDraw. Изображения подвергались разному роду обработке и были возвращены функциями. Были созданы изображения треугольников разной формы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Okhrimenko_Denis_lb2.py

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


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


def change_color(img, color):
    color = tuple(color)
    d = {}
    pixels = np.array(img)
    for i in pixels:
        for j in i:
            pixel = tuple(j)
            if pixel not in d:
                d[pixel] = 1
            else:
                d[pixel] += 1
    find = max(d.values())
    for key in d:
        if d[key] == find:
            most_relevant_color = key
    for i in range(len(pixels)):
        for j in range(len(pixels[i])):
            if tuple(pixels[i][j]) == most_relevant_color:
                pixels[i][j] = color
    img = Image.fromarray(pixels)
    return img


# Задача 3
def collage(img, N, M):
    image = img
```

```
img = img.resize(size=(img.size[0] * M, img.size[1] * N))
for i in range(N):
    for j in range(M):
        img.paste(image, (image.size[0] * j, image.size[1] * i))
return img
```