МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Информатика»

Студент гр. 3344 _____ Гусева Е.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Освоение основ архитектуры компьютера и обработки изображений. Работа с модулем 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'>

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

Были импортированы библиотеки *PIL*, *numpy*. Для выполнения первой задачи была написана функция def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color). Создана переменная drawing, в которую записываем возможность редактирования изображения. Далее идет проверка есть ли заливка у фигуры, в зависимости от этого фигура будет залита или нет. Фигура задается с помощью встроенной функции polygon. Функция возвращает измененное изображение.

Для выполнения второй задачи была написана функция def change_color(img, color). Значения высоты и ширины изображения считываются через img.size, изображение копируется, создается словарь pixs_colors. Далее с помощью двух циклов, считываются пиксели с изображения, проверяется условие, если цвет пикселя еще не встречался, то в ключи словаря записывается этот цвет и начальное значение 1, иначе увеличивается значение количества раз встреченных чисел. Далее словарь сортируется по величине значения встреченных пикселей одного цвета. В переменную color_old записывается самый часто встречаемый в изображении цвет. Далее с помощью двойного цикла, в переменную pix_color считывается значения цвета текущего пикселя, условием проверяется, что если цвет считанного пикселя является самым часто встречаемым, то цвет этого пикселя меняется на тот, что подается на вход функции. Функция возвращает измененное изображение.

Для выполнения третьей задачи была написана функция def collage(img, N, M). Значения высоты и ширины изображения считываются через img.size, по условию коллаж в N раз выше и в M раз шире, чем изначальное изображение. Создается коллаж, размер коллажа задается полученными перед этим collage_width и collage_height. Далее циклом в коллаж вставляются все изображния. Функция возвращает коллаж.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	triangle(img.size = (90,51), 74, 3, 59, 11, 65, 44, 6,), color = [265, 119, 12], None)	img	-
2.	change_color(img.size = (90,51), color = [54, 54, 251])	img	-
3.	collage(img.size = (90,51), 6, 5)	img	-

Выводы

Были освоены основы архитектуры компьютера и обработки изображений, проведена работа с модулем Pillow.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.py

```
import PIL
from PIL import Image, ImageDraw

import numpy as np

# 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)],
outline=tuple(color), fill=tuple(fill_color), width=thickness)
    else:
                    drawing.polygon([(x0,  y0),  (x1,  y1),  (x2,  y2)],
outline=tuple(color), width=thickness)
    return img


# Задача 2
def change_color(img, color):
    width, height = img.size
    new_image = img.copy()

    pixs_colors={}

    for i in range(width):
        for j in range(height):
            pixs_color = img.getpixel((i, j))
            if pixs_color in pixs_colors:
                pixs_colors[pixs_color] += 1
            else:
                pixs_colors.update({pixs_color: 1})

       pixs_colors  =  dict(sorted(pixs_colors.items(),  key=lambda  item:
item[1]))
    color_old = list(pixs_colors.keys())[-1]
    for i in range(width):
        for j in range(height):
            pix_color = img.getpixel((i, j))
            if pix_color == color_old:
                new_image.putpixel((i, j), tuple(color))
    return new_image




# Задача 3
def collage(img, N, M):
    width, height = img.size
```

```
collage_width = width * M
collage_height = height * N

collage = Image.new('RGB', (collage_width, collage_height))

for i in range(N):
    for j in range(M):
        x = j * width
        y = i * height

        collage.paste(img, (x, y))


return collage
```