

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

Комосский Е.А.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить Pillow и использовать полученные знания для выполнения заданий.

Задание.

Вариант 1.

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.

Подключаем библиотеку `numpy` с псевдонимом `np`, подмодули `Image` и `ImageDraw` из библиотеки `Pillow`.

Функция `triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color)`.

Функция рисует на изображении треугольник.

Функция получает на вход исходное изображение, координаты вершин треугольника, толщину линий и их цвет, а также цвет заливки. Если последний равен `None`, то треугольник не залит цветом. и точек в виде кортежей. Далее с помощью `draw.polygon` функция наносит изображение треугольника на исходное изображение и возвращает его.

Функция `change_color(img, color)`.

Функция находит наиболее часто встречаемый цвет и заменяет его на переданный.

Функция получает на вход исходное изображение и цвет, на который нужно заменить. Создаётся словарь `count` в который будет вноситься цвет пикселя и количество таких цветов в изображении. С помощью `getdata()` получаем цвета всех пикселей и проходимся циклом по ним, записывая количество в словарь. Записываем наиболее часто встречаемый цвет в `max_color`. В `width, height` передаём ширину и высоту изображения. Проходимся циклом по каждому пикселю и меняем его цвет на заданный, если он совпадает со значением `max_color`.

Функция возвращает изменённое изображение.

Функция `collage(img, N, M)`.

Функция создаёт коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и возвращает его (новое изображение).

Функция получается на вход исходное изображение и значения N и M. Создаёт `img1` размером N высот исходного изображения и M широт с белой заливкой. С помощью цикла проходим по новому изображению и вставляем `img1`.

Функция возвращает новое изображение `img1`, не изменяя исходное.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Была изучена библиотека Pillow. Знания были применены на практике для решения задач.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb2.py

```
import numpy as np
from PIL import Image, ImageDraw
# Задача 1
def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color):
    draw = ImageDraw.Draw(img)
    if fill_color is None:
        draw.polygon([(x0,y0), (x1, y1), (x2,y2)], fill = None, outline =
tuple(color), width=thickness)
    else: draw.polygon([(x0,y0), (x1, y1), (x2,y2)], fill =
tuple(fill_color), outline = tuple(color), width=thickness)
    return img

# Задача 2
def change_color(img, color):
    count= {}
    for i in img.getdata():
        if i not in count:
            count[i] = 0
        count[i] += 1
    max_color = max(count, key=count.get)
    arr = np.array(img)
    arr[arr == max_color] = color
    img = Image.fromarray(arr)
    return img

# Задача 3
def collage(img, N, M):
    img1 = Image.new("RGB", (img.size[0]*M, img.size[1]*N), 'white')
    width, height = img1.size
    for x in range(0, width, img.size[0]):
        for y in range(0, height, img.size[1]):
            img1.paste(img, (x, y))
    return img1
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.			<code>triangle(img, 300, 200, 200, 400, 400, 400, 2, [0, 0, 0], None)</code>
2.			<code>change_color(img, [0, 0, 0])</code>
3.			<code>collage(img, 3, 5)</code>