

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Информатика»
Тема: Введение в архитектуру компьютера

Студентка гр. 2381

Потапова Д.М.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить модуль Pillow языка Python и использовать полученные знания для выполнения заданий.

Задание.

Вариант 1. Предстоит решить 3 подзадачи, используя библиотеку Pillow (PIL). Для реализации требуемых функций студент должен использовать `pymru` 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 полученное изображение.

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).

Выполнение работы.

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

Функция принимает на вход исходное изображения, вершины треугольника, толщину линий, их цвет и цвет заливки (цвета представлены списками). Функция рисует на изображении треугольник. Для решения задачи используется метода *ImageDraw.Draw(img)*. Если аргумент исходной функции *fill_color* равен *None*, то с помощью метода *ImageDraw.polygon* на исходное изображение наносится треугольник с соответствующими параметрами (координаты треугольника в виде кортежа пар, цвета заливки и линий в виде кортежей) без заливки, иначе – с заливкой, цвет которой соответствует *fill_color*. Функция возвращает измененное изображение.

2. Функция *change_color(img, color)*

Функция принимает на вход исходное изображение и цвет, представленный списком. Функция заменяет наиболее часто встречаемый в изображении цвет и заменяет его на новый переданный в виде аргумента *color*. Создается копия изображения *img1* и словарь *color_count*, где ключи – цвета, значения – количество пикселей этих цветов. С помощью метода

img.load() (возвращает объект, с помощью которого можно получить доступ к отдельным пикселям изображения) и циклов *for* осуществляется перебор всех пикселей (и их цветов) и заполнение словаря. В переменную *max_count_color* записывается наиболее часто встречающийся цвет. Осуществляется повторный перебор всех пикселей, проверка на их соответствие найденному цвету и замена на новый цвет при нахождении совпадений. Функция возвращает новое измененное изображение.

3. Функция *collage(image, N, M)*

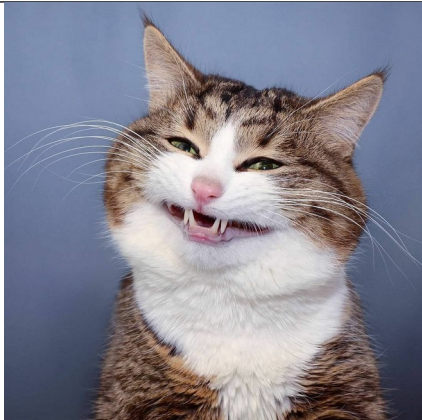
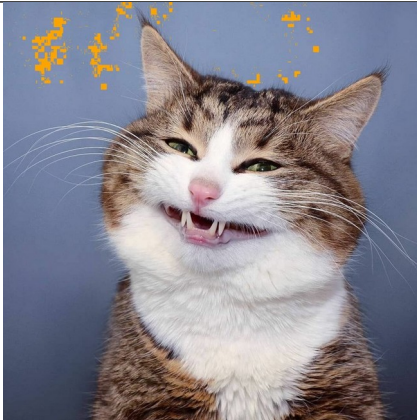




Функция принимает на вход исходное изображение и два целых числа *M* и *N*. Функция создает коллаж из этого изображения размером *N*M*, где *N* – количество изображений по высоте, *M* – по ширине. С помощью метода *Image.new()* создается новое изображение высотой в *N* высот исходного и шириной в *M* широт соответственно и заливается черным цветом. С помощью метода *img.size* создаются переменные широт и высот исходного и нового изображений. Циклами *for* осуществляется проход по пикселям нового изображения, где шаг соответствует высоте и ширине исходного. С помощью метода *img.paste()* исходное изображение вставляется в новое, образуя коллаж. Функция возвращает новое изображение.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии

1.			Функция 2 change_color(img , [255, 165, 0])
2.			Функция 1 triangle(img, 300, 200, 200, 400, 400, 400, 8, [0, 0, 0], [255, 165, 0])
3.			Функция 3 collage(img, 2, 2)

Выводы.

Был изучен модуль Pillow. Полученные знания использованы на практике для выполнения заданий.

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):
    drawing = ImageDraw.Draw(img)
    if fill_color is None:
        drawing.polygon(((x0, y0), (x1, y1), (x2, y2)), fill=None,
outline=tuple(color), width=thickness)
    else:
        drawing.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):
    color_count = {}
    img1 = img.copy()
    pixels = img1.load()
    for i in range(img1.size[0]):
        for j in range(img1.size[1]):
            if pixels[i, j] not in color_count:
                color_count[pixels[i, j]] = 1
            else:
                color_count[pixels[i, j]] += 1
    max_count_color = max(color_count, key=color_count.get)
    for i in range(img1.size[0]):
        for j in range(img1.size[1]):
            if pixels[i, j] == max_count_color:
                pixels[i, j] = tuple(color)
    return img1

# Задача 3
def collage(img, N, M):
    width_img = img.size[0]
    height_img = img.size[1]
    img_new = Image.new("RGB", (width_img * M, height_img * N), (0,
0, 0))
    new_width = img_new.size[0]
    new_height = img_new.size[1]
    for i in range(0, new_width, width_img):
        for j in range(0, new_height, height_img):
            img_new.paste(img, (i, j))
    return img_new
```