

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. 3381

Салдин Д.Е.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы.

Научиться использовать библиотеку Pillow для работы с графическими данными, получить практические знания во время разработки программы обработки изображений.

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 $N \times M$ раз. (N раз по высоте, M раз по ширине) и вернуть его (новое изображение).

При необходимости можно писать дополнительные функции.

Выполнение работы.

Подключенные библиотеки:

`Pillow` — для работы с изображениями

Функции:

`triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill_color)` — рисует треугольник на изображении

`change_color(img, color)` — заменяет наиболее часто встречаемый цвет

`collage(img, N, M)` — создаёт коллаж полученного изображения

Переменные:

`drawing` — хранит объект `ImageDraw`, позволяющий рисовать на изображении

`all_color_info` — хранит информацию о том, сколько раз каждый цвет встречается

`color_repeat` — хранит количество каждого цвета

`max_repeat` — хранит сколько раз встретился наиболее часто встречающийся цвет

`ind` — хранит индекс наиболее часто встречающегося цвета

`most_rep_color` — наиболее часто встречающийся цвет

`red_img` — копия изображения, которая будет отредактирована


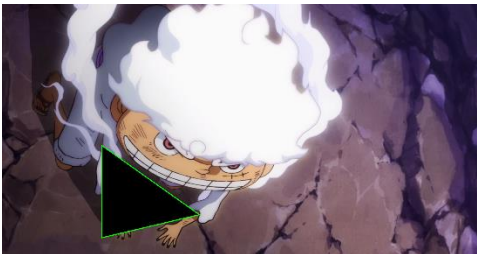




`collage` — изображение размера N на M , на котором будет коллаж

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.			Вызов функции triangle(img, 400, 600, 800, 900, 400, 1000, 5, [0, 255, 0], [0, 0, 0])
2.			Вызов функции change_color(img , [132, 145, 99])
3.			Вызов функции collage(img, 3, 4)

Выводы.

Были изучены методы работы с библиотекой Pillow, в частности модули Image, ImageDraw, были созданы три функции для обработки графических данных, а именно рисунок треугольника, замена цвета, создание коллажа.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.py

```
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 != None:
        drawing.polygon((x0, y0, x1, y1, x2, y2), tuple(fill_color),
tuple(color), thickness)
    else: drawing.polygon((x0, y0, x1, y1, x2, y2), fill_color,
tuple(color), thickness)
    return img

# Задача 2

def change_color(img, color):
    all_color_info = img.getcolors(maxcolors = img.size[0] *
img.size[1])
    color_repeat = [all_color_info[i][0] for i in
range(len(all_color_info))]
    max_repeat = max(color_repeat)
    ind = color_repeat.index(max_repeat)
    most_rep_color = all_color_info[ind][1]
    red_img = img.copy()
    for x in range(img.size[0]):
        for y in range(img.size[1]):
            if red_img.getpixel((x,y)) == most_rep_color:
                red_img.putpixel((x, y), tuple(color))
    return red_img

# Задача 3

def collage(img, N, M):
    collage = Image.new(img.mode, (img.size[0]*M, img.size[1]*N),
"black")
```

```
for y in range(0, img.size[1]*N, img.size[1]):  
    for x in range(0, img.size[0]*M, img.size[0]):  
        collage.paste(img, (x, y))  
return collage
```