

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.

Студентка гр. 2384

Преподаватель

Валеева А.А.
Шевская
Н.В.

Санкт-Петербург
2022

Цель работы.

Целью данной лабораторной работы является изучение библиотеки Pillow и написания программы с применением Pillow и numpy.

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

Основные теоретические положения.

Для написания программы была использована библиотека `Pillow`.

Выполнение работы.

Функция `def triangle(img, x1, y1, x2, y2, x3, y3, thickness, color, fill_color)` получает контекст изображения при помощи функции `Draw()`, которая принимает изображение. Если аргумент `fill_color` был получен (`TRUE`), то треугольник будет закрашен переданным в функцию цветом. Фигура рисуется при помощи метода `polygon()`, который принимает список кортежей (координат точек), а также дополнительные параметры, такие как толщина линии (`thickness`), цвет линии(`color`) и цвет для заполнения (`fill_color`, если передан в функцию). В конце функции возвращается изменённое изображение `img2`.

Функция `def change_color(img, color=None)` получает на вход изображение и цвет который нужно заменить. Создаем новое изображение (`img2`), которому передаем тот же режим работы с изображением (`mode`)и размер (`size`), как и у переданного изображения, цвет не важен (`None`). Создаем словарь цветов `colors` и переменную `color`, содержащую кортеж из значений красного, зеленого и синего цветов из переданного в функцию цвета `color`. Далее с помощью вложенного цикла двумерно перебираем пиксели изображения, если встречаемого цвета еще нет в ключах, то добавляем, если же есть – то увеличиваем значение на 1. Переменной `orig_color` присваиваем максимальное значение из словаря `colors`. Далее снова

вложенными циклами перебираем пиксели изображения, если цвет *orig_color*, то меняем его на *color*. В конце функции возвращаем полученное изображение *img2*.

Функция *def collage(img, N, M)* получает на вход изображение и количество повторений по осям. Создадим переменные *x* — размер изображения *img* по оси *X*, *y* — размер изображения *img* по оси *Y*. При помощи метода *crop* вырезаем исходное изображение(*arg*). Далее создаём новое изображение следующего размера: *M*x, N*y*. Далее мы *N*M* раз должны сделать вставку *arg* на наше новое изображение *img* следующим образом - мы сдвигаемся на *x* и проверяем, не ушли ли за границы изображения (если ушли, значит нужно сдвинуться по оси *Y*).

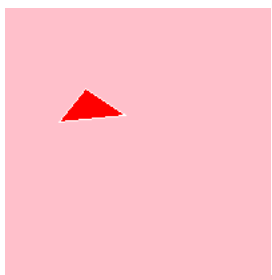
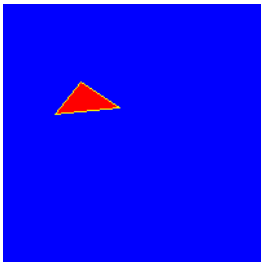
Возвращаем изменённое изображение *img*.

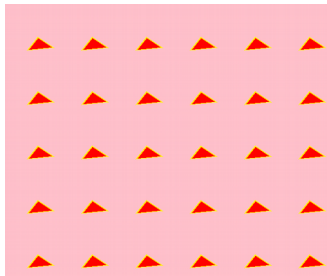
Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<code>triangle(Image.new("RGB", (200, 200), "pink"), 60, 60, 40, 85, 90, 80, 1, [255, 255, 255], [255, 0, 0]).show()</code>		Проверка функции <i>triangle</i> .
2.	<code>change_color(triangle(Image.new("RGB", (69, 20), "black"), 10, 10, 40, 15, 50, 10, 1, [255, 255, 0], [255, 0, 0]), (0, 0, 255)).save("test2.png")</code>		Проверка функции <i>change_color</i> .

3.	<code>collage(change_color(triangle(Image.new("RGB", (69, 20), "black"), 10, 10, 40, 15, 50, 10, 1, [255, 255, 0], [255, 0, 0]), [0, 255, 30]), 3, 2).save("test3.png")</code>		Проверка функции collage.
----	---	--	---------------------------

Выводы.

Была изучена библиотека Pillow и основы архитектуры компьютера, а также была написана программа с использованием модуля collection, которая умеет рисовать треугольники, а также менять цвет изображения и делать коллаж.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла:

Valeeva_Alina_lb2.py

```
import PIL
from PIL import Image, ImageDraw

def triangle(img, x1, y1, x2, y2, x3, y3, thickness, color, fill_color):
    points = [(x1, y1), (x2, y2), (x3, y3)]
    draw = ImageDraw.Draw(img)
    if fill_color:
        draw.polygon(points, fill=tuple(fill_color), width=thickness,
outline=tuple(color))
    else:
        draw.polygon(points, width=thickness, outline=tuple(color))
    return img

def change_color(img, color):
    img2 = Image.new(img.mode, img.size, None)
    img2.paste(img, (0,0))
    colors = {}
    color = tuple(color)
    for i in range(img2.width):
        for j in range(img2.height):
            if img2.getpixel((i, j)) in colors:
                colors[img2.getpixel((i, j))] += 1
            else:
                colors[img2.getpixel((i, j))] = 1
    orig_color = max(colors, key = colors.get)
```

```

for i in range(img2.width):
    for j in range(img2.height):
        if img2.getpixel((i, j)) == orig_color:
            img2.putpixel((i, j), color)
return img2

```

```

def collage(img, N, M):
    x, y = img.size
    arg = img.crop( (0, 0, x, y) )
    img = Image.new( "RGB", (x * M, y * N) )
    x1 = 0
    y1 = 0
    flag = 1
    for i in range(N * M):
        if flag > M:
            flag = 1
            y1 = y1 + y
            x1 = 0
        img.paste(arg, (x1, y1))
        x1 = x1 + x
        flag += 1
    return img

```