

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

Газукина Д.Д.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить основные функции библиотеки Pillow (PIL) для выполнения практических задач.

Задание.

Вариант 3

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 пентаграммы в круге

Необходимо написать функцию `solve()`, которая рисует на изображении пентаграмму в окружности.

Функция `solve()` принимает на вход: изображение (`img`), координаты центра окружности (`x,y`), радиус окружности, толщину линий и окружности (`thickness`). Цвет линий и окружности (`color`) - представляет собой список (`list`) из 3-х целых чисел

Функция должна изменить исходное изображение и вернуть его изображение.

2) Поменять местами участки изображения и поворот

Необходимо реализовать функцию `solve`, которая меняет местами два квадратных, одинаковых по размеру, участка изображений и поворачивает эти участки на 90 градусов по часовой стрелке, а затем поворачивает изображение на 90 градусов по часовой стрелке.

Функция `solve()` принимает на вход: квадратное изображение (`img`), координаты левого верхнего угла первого квадратного участка(`x0,y0`), координаты левого верхнего угла второго квадратного участка(`x1,y1`), длину стороны квадратных участков (`width`).

Функция должна сначала поменять местами переданные участки изображений. Затем повернуть каждый участок на 90 градусов по часовой стрелке. Затем повернуть всё изображение на 90 градусов по часовой стрелке.

Функция должна вернуть обработанное изображение, не изменяя исходное.

3) Средний цвет

Необходимо реализовать функцию `solve`, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель). Функция принимает на вход: изображение (`img`), координаты левого верхнего угла области (`x0,y0`), координаты правого нижнего угла области (`x1,y1`). Функция должна заменить цвета каждого пикселя в этой области на средний цвет пикселей вокруг.

Пиксели вокруг:

8 самых близких пикселей, если пиксель находится в центре изображения

5 самых близких пикселей, если пиксель находится у стенки

3 самых близких пикселя, если пиксель находится в углу

Функция должна вернуть обработанное изображение, не изменяя исходное.

Можно реализовывать дополнительные функции.

Выполнение работы.

Подключаются библиотеки `numpy` и `PIL (Pillow)`. Импортируются модули `Image`, `ImageDraw`.

Функция `pentagram(img, x, y, r, thickness, color)`.

Создается объект `drawing`, далее с помощью метода `ellipse` создается окружность по двум координатам — верхнего левого угла и правого нижнего.

В цикле `for` (от 0 до 4 включительно) вычисляется каждая из координат пяти углов пентаграммы (с помощью нахождения углов через синус и

косинус — функции `np.cos()` и `np.sin()` из `numpy`). Находятся начало и конец каждого отрезка (`cords = [(Xstart, Ystart), (Xend, Yend)]`), далее с помощью метода `line` рисуются 5 отрезков звезды. Функция возвращает измененное изображение.

Функция `swap(img, x0, y0, x1, y1, width)`.

Создается копия изображения с помощью метода `copy()`, далее с помощью метода `crop` создаются два изображения `piece1` и `piece2` и поворачиваются на 90 градусов по часовой стрелке методом `rotate(-90)`. Методом `paste` составные части вставляются на координаты друг друга в копию исходного изображения, которое затем тоже поворачивается. Функция возвращает измененную копию изображения.

Функция `avg_color(img, x0, y0, x1, y1)`.

Создается копия изображения `img_res`, выгружаются значения пикселей с помощью метода `load()` и размеры изображения (`img_res.size`).

Вспомогательная функция `end_of_img(xy, shape=img_shape)`: нужна для того, чтобы проверить, не выходят ли индексы, к которым мы обращаемся позднее, за пределы изображения. Возвращает `True`, если индексы существуют (не выходят за пределы «двумерного массива» пикселей), иначе `False`.

В цикле `for` (две переменные, вложенный цикл) перебирается каждый пиксель области, координаты двух углов которой подаются в качестве аргументов функции. Создается кортеж индексов пикселей `pix_ind`, которые находятся рядом с пикселем `[x,y]`. С помощью функции `filter` удаляются все индексы, выходящие за границы изображения. Далее создается кортеж значений цветов, полученных по индексам пикселей с помощью функции `getpixel`. Вычисляются суммы значений R, G, B этих пикселей и находится среднее для каждого из них (`res_color`). Текущему пикселю (значения `x` и `y`) присваивается средний цвет: `arr[x, y] = res_color`.

Возвращается измененная копия исходного изображения.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Были изучены и использованы при решении практических задач основные функции библиотеки Pillow (PIL).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import PIL
import numpy as np
from PIL import Image, ImageDraw

def pentagram(img, x, y, r, thickness, color):
    drawing = ImageDraw.Draw(img)
    xy = [x-r, y-r, x+r, y+r]
    drawing.ellipse(xy, None, tuple(color), thickness)
    for i in range(0,5):
        ugol = (np.pi / 5) * (2 * i + 3 / 2)
        Xstart = int(x + r * np.cos(ugol))
        Ystart = int(y + r * np.sin(ugol))
        end_of_line = (np.pi / 5) * (2 * (i+2) + 3 / 2)
        Xend = int(x + r * np.cos(end_of_line))
        Yend = int(y + r * np.sin(end_of_line))
        cords = [(Xstart, Ystart), (Xend, Yend)]
        drawing.line(tuple(cords), tuple(color), thickness)
    return img

def swap(img, x0, y0, x1, y1, width):
    a = width
    img_res = img.copy()
    piecel = img_res.crop((x0, y0, x0 + a, y0 + a)).rotate(-90)
    piece2 = img_res.crop((x1, y1, x1 + a, y1 + a)).rotate(-90)
    img_res.paste(piecel, (x1, y1))
    img_res.paste(piece2, (x0, y0))
    img_res = img_res.rotate(-90)
    return img_res

def avg_color(img, x0, y0, x1, y1):
    img_res = img.copy()
    arr = img_res.load()
    img_shape = img_res.size

    def end_of_img(xy, shape=img_shape):
        if (xy[0] >= 0) and (xy[0] < shape[0]) and (xy[1] >= 0)
and (xy[1] < shape[1]):
            return True
        else:
            return False

    for x in range(x0, x1+1):
        for y in range(y0, y1+1):

            pix_ind = ( (x-1,y-1), (x,y-1), (x+1,y-1), (x+1,y),
(x+1,y+1), (x,y+1), (x-1,y+1), (x-1,y) )
```

```

        pix_ind = tuple(filter(end_of_img, pix_ind))
        pix = tuple(map(img.getpixel, ((i[0],i[1]) for i in
pix_ind)))







        R, G, B = 0, 0, 0
        for i in range(len(pix)):
            R += pix[i][0]
            G += pix[i][1]
            B += pix[i][2]
        res_color = tuple((R // len(pix), G // len(pix), B //
len(pix)))
        arr[x, y] = res_color
    return img_res

```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	 img	 img	pentagram(img, 300, 300, 200, 10, (200,10,56))
2.	 img	 img_res	swap(img, 100, 100, 300, 300, 200)
3.	 img	 img_res	avg_color(img, 0, 0, 250, 250)