

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

Заметаев М.А.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Изучить библиотеку Pillow. Научиться применять её для решения практических задач.

Задание.

Вариант 3

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

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 solve() 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.

Выполнение работы.

Подключена библиотека numpy. Из библиотеке PIL импортированы модули Image, ImageDraw.

Функция pentagram.

Создается объект dr. С помощью метода ellipse создается эллипс по двум координатам, вершинам прямоугольника в который вписан эллипс. Далее по формуле с помощью цикла for вычисляются пять вершин звезды и с помощью метода line соединяются нужные вершины. Затем функция возвращается измененное изображение

Функция swap.

С помощью метода copy() создается копия исходного изображения. Затем с помощью метода crop обрезаются части(part1, part2) нового изображения и поворачиваются на 90 градусов по часовой стрелке. С помощью метода paste в изображения вставляются part1, part2 в обратные места. После изображение поворачивается на 90 градусов с помощью метода rotate и возвращается измененное новое изображение, не меняя старого.

Функция avg_color.

Создается копия изображения - nimg. В переменную sz с помощью метода size записывается длина и ширина изображения. С помощью двойного цикла рассматриваются все возможные пиксели изображения. Для каждого из пикселей создается массив для значений сумм цветов. Создается массив координат pix_near для всех возможных стоящих рядом пикселей, в том числе выходящие за пределы изображения. Далее с помощью дополнительной функции filtr_gran проверяется входят ли эти пиксели в заданное изображение, в противном случае они удаляются из массива. С помощью цикла for для каждого из входящих в изображение соседних для заданного пикселей узнается их цвет в палитре RGB и суммируется в массив sum. С помощью этого массива вычисляется нужный средний цвет для заданного пикселя и с помощью метода putpixel цвет пикселя меняется на него. Функция возвращается измененное изображение.

Вывод

Были изучены основные методы библиотеки pillow и применены для решения практических задач.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

название файла: main_lb2.py

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

def pentagram(img, x, y, r, thickness, color):
    dr = ImageDraw.Draw(img, "RGB")
    x1 = x - r
    y1 = y - r
    x2 = x + r
    y2 = y + r
    dr.ellipse(((x1, y1), (x2, y2)), fill = None, width =
thickness, outline=tuple(color))
    versh = []
    for i in range(5):
        phi = (numpy.pi / 5) * (2 * i + 3 / 2)
        node_i = (int(x + r * numpy.cos(phi)), int(y + r *
numpy.sin(phi)))
        versh.append((node_i))
    versh=tuple(versh)
    for i in range(5):
        dr.line((versh[i],versh[(i + 2) % 5]),fill =
tuple(color),width = thickness)
    return img

def swap(img, x0,y0,x1,y1,width):
    nimg = img.copy()
    part1 = nimg.crop((x0, y0, x0 + width, y0 +
width)).rotate(-90)
    part2 = nimg.crop((x1, y1, x1 + width, y1 +
width)).rotate(-90)
    nimg.paste(part1, (x1, y1))
    nimg.paste(part2, (x0, y0))
    nimg = nimg.rotate(-90)
    return nimg

def avg_color(img, x0,y0,x1,y1):
    nimg = img.copy()
    sz = img.size
    for x in range(x0, x1 + 1):
        for y in range(y0, y1 + 1):
            sum = [0]*3
            pix_near= [(x-1,y-1), (x,y+1), (x,y-1), (x+1,y),
(x+1,y-1), (x+1,y+1), (x-1,y+1), (x-1,y)]
            count = len(pix_near)
            for k in range(count):
                if not filtr_gran(pix_near[count-k-1],sz):
                    pix_near.pop(count-k-1)
            for n in pix_near:
                pixel = img.getpixel(n)
                for m in range(3):
                    sum[m] += pixel[m]
            count = len(pix_near)
            ncolor = (int(sum[0] / count), int(sum[1] / count),
int(sum[2] / count))
```

```
        nimg.putpixel((x, y), ncolor)
    return nimg
```

```
def filtr_gran(coord,sz):
    return (coord[0] >= 0) and (coord[0] < sz[0]) and (coord[1] >=
0) and (coord[1] < sz[1])
```