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. 3344 _____ Хангулян С. К.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Целью работы является введение в архитектуру компьютера и изучение и работа с модулем pillow в Python.

Задание

Вариант 3

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 пентаграммы в круге

Необходимо написать функцию solve(), которая рисует на изображении пентаграмму в окружности. Функция solve() принимает на вход:

- Изображение (img)
- координаты центра окружность (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.

Примечание:

Вершины пентаграммы высчитывать по формуле:

phi = (pi/5)*(2*i+3/2)

node_i = (int(x0+r*cos(phi)),int(y0+r*sin(phi)))

x0,y0 - координаты центра окружности, в который вписана пентаграмма

r - радиус окружности

i - номер вершины от 0 до 4

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:

1. 8 самых близких пикселей, если пиксель находится в центре изображения
2. 5 самых близких пикселей, если пиксель находится у стенки
3. 3 самых близких пикселя, если пиксель находится в угле

Функция должна вернуть обработанное изображение, не изменяя исходное.

Средний цвет - берется целая часть от среднего каждой компоненты из rgb. (int(sum(r)/count),int(sum(g)/count),int(sum(b)/count))

Можно реализовывать дополнительные функции.

Выполнение работы

Функция swap. Переменные:

- cp_img – копия изображения;
- img1 – один из квадратов, которые нужно поменять местами;
- img2 – второй квадрат.

Вначале сделана копия изображения cp_img для дальнейшей работы. Затем вырезан и перевернут на 90 градусов по часовой стрелке первый квадрат с помощью функций crop и transpose соответственно. Аналогично со вторым квадратом. Чтобы развернуть квадрат на 90 градусов по часовой стрелке, используется поворот на 270 градусов против часовой стрелки. Затем на копированном изображении на место, где был первый квадрат, накладывается второй, и наоборот. Затем измененное изображение поворачивается на 90 градусов по часовой стрелке путем поворота на 270 градусов против часовой. Функция возвращает измененную копию изображения.

Функция avg_color. Переменные:

- av_img – копия изображения, в которой будут поменяны цвета пикселей;
- pixels_around – список пикселей вокруг (с учетом возможных граней или углов)
- r, g, b – суммы всех оттенков красного, зеленого и голубого пикселей вокруг;
- count – счетчик количества пикселей вокруг;
- begin_x – нижний левый «сосед»;
- begin_y – нижний левый «сосед»;

  *вместе с begin_x образуют левую нижнюю вершину прямоугольника, содержащего пиксели вокруг.
- end_x – верхний правый «сосед»;
- end_y – верхний правый «сосед»;

- *вместе с begin_x образуют правую верхнюю вершину прямоугольника, содержащего пиксели вокруг.

Вначале создается копия изображения, в которой позже будут поменяны цвета пикселей. Затем перебираются все пиксели в области, заданной начальными координатами x0, y0, x1, y1. Создается массив pixels_around, который будет содержать адреса пикселей вокруг, инициализируются суммы r, g, b, счетчик count. Затем с помощью функций max и min определяется, являются ли пиксели крайними или нет. Если значение становится меньше нуля или больше длины или высоты, соответственные пиксели не вносятся в pixels_around. Далее как раз pixels_around и заполняется, а также считается количество элементов этого массива. Затем идет подсчет сумм r, g, b путем перебора всех соответственных цветов пикселей вокруг. Наконец, цвета пикселей заменяются на средние с помощью функции putpixel. Возвращается измененная копия изображения.

Функция pentagram. Переменные:

- color – цвет в виде кортежа;
- drawing – рисунок, на котором будет изображена пентаграмма;
- x1y1 – координаты нижней левой вершины прямоугольника, в который будет вписана окружность;
- x2y2 – координаты верхней правой вершины прямоугольника, в который будет вписана окружность;
- versh – массив, содержащий вершины пятиугольной звезды, вписанной в окружность;
- phi – угол (72 градуса * N);
- node_i – координаты вершины (зависит от phi);

Вначале массив color преобразуется в кортеж для подачи в функцию. Затем определяются вершины прямоугольника x1y1 и x2y2. Затем рисуется окружность, вписанная в этот прямоугольник с помощью функции ellipse. Далее инициализируется массив вершин.

Происходит поиск координат вершин, они заносятся в массив. Затем с помощью цикла и двух условных операторов происходит отрисовка пятиконечной звезды. Возвращается изображение.

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

Функция	Выходные данные	Комментарий
swap	cp_img	Корректно
avg_color	av_img	Корректно
pentagram	img	Корректно

Выводы

Было успешно изучено введение в архитектуру компьютера, освоен модуль pillow в Python, а также реализованы практические задачи, связанные с модулем pillow.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan_Sargis_lb2.py

```
import os
os.environ["OPENBLAS_NUM_THREADS"] = "4"

from PIL import Image, ImageDraw
from math import *

def swap(img, x0, y0, x1, y1, width):
    cp_img = img.copy()
    img1 = cp_img.crop((x0, y0, x0+width, y0+width))
    img1 = img1.transpose(Image.Transpose.ROTATE_270)
    img2 = cp_img.crop((x1, y1, x1+width, y1+width))
    img2 = img2.transpose(Image.Transpose.ROTATE_270)
    cp_img.paste(img1, (x1, y1))
    cp_img.paste(img2, (x0, y0))
    cp_img = cp_img.transpose(Image.Transpose.ROTATE_270)
    return cp_img



def avg_color(img, x0, y0, x1, y1):
    av_img = img.copy()
    for x in range(x0, x1+1):
        for y in range(y0, y1+1):
            pixels_around = []
            r, g, b = 0, 0, 0
            count = 0

            begin_x = max(x-1, 0)
            begin_y = max(y-1, 0)
            end_x = min(img.width-1, x+1)
            end_y = min(img.height-1, y+1)

            for k in range(begin_x, end_x+1):
                for n in range(begin_y, end_y+1):
                    if (k, n) != (x, y):
                        pixels_around.append(img.getpixel((k, n)))
                        count += 1

            for i in pixels_around:
                r += i[0]
                g += i[1]
                b += i[2]

            av_img.putpixel((x,   y),   (int(r/count),   int(g/count),
int(b/count)))

    return av_img



def pentagram(img, x, y, r, thickness, color):
    color = tuple(color)
```

```
    drawing = ImageDraw.Draw(img)
    x1y1 = (x-r, y-r)
    x2y2 = (x+r, y+r)
    drawing.ellipse((x1y1, x2y2), width = thickness, outline = color,)

    versh = []
    for i in range(5):
        phi = (pi/5)*(2*i+3/2)
        node_i = (int(x+r*cos(phi)),int(y+r*sin(phi)))
        versh.append(node_i)

    for j in range(5):
        if j < 3:
            drawing.line((versh[0+j], versh[2+j]), fill = color, width
= thickness, joint = None)
        elif j == 3:
            drawing.line((versh[3], versh[0]), fill = color, width =
thickness, joint = None)
        else:
            drawing.line((versh[4], versh[1]), fill = color, width =
thickness, joint = None)
    return img
```