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. 3342 _____ Русанов А.И.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Целью работы является освоение работы с функциями в языке python и с библиотеками numpy и Pillow.

Задание

Вариант 3.

Задача 1.

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.

Задача 2.

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.

Задача 3.

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 угле

Функция должна вернуть обработанное изображение, не изменяя исходное.

Выполнение работы

Данная программа написана на языке Python с использованием библиотек numpy и Pillow. Она состоит из 3-функций, которые вызываются сразу на сайте https://e.moevm.info.

Первая функция swap возвращает обработанное изображение, не изменяя исходное. Функция 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.

Вторая функция avg_color. Функция обрабатывает исходное изображение, заменяя цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).

Третья функция pentagram возвращает обработанное изображение. Функция по заданным параметрам рисует на изображении пентаграмму в окружности и возвращает его.

Функции, используемые в этой программе:

- image.new создает новое изображение
- image.copy создает копию изображения
- image.paste вставляет одно изображение в другое
- image.rotate поворачивает изображение
- image.getpixel получает цвет пикселя изображения
- ellipse рисует эллипс на изображении
- line рисует линию между двумя точками на изображении
- image.size возвращает размер изображения

Разработанная программа демонстрирует использование функций библиотек numpy и Pillow и работу функций на языке Python.

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.	x0 = 0 ; y0 = 0; x1 = 150; y1 = 150; width = 150;	корректные	Функция работает корректно
2.	x0 = 0 ; y0 = 0; x1 = 100; y1 = 100;	корректные	Функция работает корректно
3.	x = 100, y = 100, r = 80, thickness = 10, color = (255, 0, 0)	корректные	Функция работает корректно

Выводы

Были изучены правила работы с функциями в языке python и работа с библиотекой numpy.

Разработаны функции, возвращающие решения определенных математических заданий.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw
from numpy import pi, sin, cos


def swap(img, x0, y0, x1, y1, width):
    img_res = img.copy()
    part_1 = img_res.crop((x0, y0, x0 + width, y0 + width)).rotate(-
90)
    part_2 = img_res.crop((x1, y1, x1 + width, y1 + width)).rotate(-
90)
    img_res.paste(part_1, (x1, y1))
    img_res.paste(part_2, (x0, y0))
    img_res = img_res.rotate(-90)
    return img_res


def avg_color(img, x0, y0, x1, y1):
    result_img = img.copy()

    width, height = img.size

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

            r_values = []
            g_values = []
            b_values = []

            if x > 0:
                r, g, b = img.getpixel((x - 1, y))
                r_values.append(r)
                g_values.append(g)
                b_values.append(b)
                if y > 0:
                    r, g, b = img.getpixel((x - 1, y - 1))
                    r_values.append(r)
                    g_values.append(g)
                    b_values.append(b)
                if y < height - 1:
                    r, g, b = img.getpixel((x - 1, y + 1))
                    r_values.append(r)
                    g_values.append(g)
                    b_values.append(b)

            if x < width - 1:
                r, g, b = img.getpixel((x + 1, y))
                r_values.append(r)
                g_values.append(g)
                b_values.append(b)
                if y > 0:
```

```
                    r, g, b = img.getpixel((x + 1, y - 1))
                    r_values.append(r)
                    g_values.append(g)
                    b_values.append(b)
                if y < height - 1:
                    r, g, b = img.getpixel((x + 1, y + 1))
                    r_values.append(r)
                    g_values.append(g)
                    b_values.append(b)

            if y > 0:
                r, g, b = img.getpixel((x, y - 1))
                r_values.append(r)
                g_values.append(g)
                b_values.append(b)

            if y < height - 1:
                r, g, b = img.getpixel((x, y + 1))
                r_values.append(r)
                g_values.append(g)
                b_values.append(b)

            avg_r = int(sum(r_values) / len(r_values))
            avg_g = int(sum(g_values) / len(g_values))
            avg_b = int(sum(b_values) / len(b_values))

            result_img.putpixel((x, y), (avg_r, avg_g, avg_b))

    return result_img


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(5):
        phi = (pi / 5) * (2 * i + 3 / 2)
        x_0 = int(x + r * cos(phi))
        y_0 = int(y + r * sin(phi))
        end_of_line = (pi / 5) * (2 * (i + 2) + 3 / 2)
        x_1 = int(x + r * cos(end_of_line))
        y_1 = int(y + r * sin(end_of_line))
        coordinates = [(x_0, y_0), (x_1, y_1)]
        drawing.line(tuple(coordinates), tuple(color), thickness)
    return img
```