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. 3342 _____ Попадюк И.В.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Целью работы является освоение работы с функциями в языке python и обучение работе с Pillow для обработки изображений.

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

В ходе выполнения работы были реализованы следующие три функции:
Первая функция swap принимает изображение, координаты и ширину некоторых участков изображения. Затем функция копирует исходное изображение, вырезает нужные участки, поворачивает эти участки на 90 градусов по часовой стрелке и меняет местами. После этого функция поворачивает все изображение на 90 градусов по часовой стрелке и возвращает результат.
Вторая функция avg_color принимает изображение и границы участка для обработки. Функция делает копию исходного изображения, проверяет расположение каждого пикселя и находит цвет ближайших пикселей, заменяет цвет пикселя на средний цвет окружающих его пикселей. После полной обработки изображения функция возвращает новое изображение с замененными пикселями.
Третья функция pentagram принимает изображение, координаты центра окружности, ее радиус, желаемые толщину линий и цвет. Функция рисует круг, после вычисляет координаты вершин звезды и отрисовывает ее.

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 = 100; y1 = 100; width = 100;	корректные	Функция работает корректно
2.	x0 = 2 ; y0 = 10; x1 = 100; y1 = 60;	корректные	Функция работает корректно
3.	x = 10, y = 10, r = 5, thickness = 2, color = (0, 255, 0)	корректные	Функция работает корректно

Выводы

Были изучены методы работы с функциями в языке python и работа с библиотекой Pillow.

Разработаны функции, которые можно использовать для обработки изображений.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

def swap(img, x0,y0,x1,y1,width):
    result = img.copy()
    zone1 = (x0, y0, x0+width, y0+width)
    zone2 = (x1, y1, x1+width, y1+width)
    new_zone1 = result.crop(zone1).rotate(angle=-90)
    new_zone2 = result.crop(zone2).rotate(angle=-90)
    result.paste(new_zone1, (x1, y1))
    result.paste(new_zone2, (x0, y0))
    result = result.rotate(angle=-90)
    return result


def avg_color(img, x0,y0,x1,y1):
    result = img.copy()
    width = img.size[0]
    height = img.size[1]
    for x in range(x0, x1+1):
        for y in range(y0, y1+1):
            red=[]
            green=[]
            blue=[]
            if x > 0:
                colors = img.getpixel((x-1, y))
                red.append(colors[0])
                green.append(colors[1])
                blue.append(colors[2])
                if y < (height - 1):
                    colors = img.getpixel((x - 1, y + 1))
                    red.append(colors[0])
                    green.append(colors[1])
                    blue.append(colors[2])
                if y > 0:
                    colors = img.getpixel((x - 1, y - 1))
                    red.append(colors[0])
                    green.append(colors[1])
                    blue.append(colors[2])
            if y > 0:
                colors = img.getpixel((x, y-1))
                red.append(colors[0])
                green.append(colors[1])
                blue.append(colors[2])
            if x < (width - 1):
                colors = img.getpixel((x+1, y))
                red.append(colors[0])
                green.append(colors[1])
                blue.append(colors[2])
```

```
                    if y < (height - 1):
                        colors = img.getpixel((x + 1, y + 1))
                        red.append(colors[0])
                        green.append(colors[1])
                        blue.append(colors[2])
                    if y > 0:
                        colors = img.getpixel((x + 1, y - 1))
                        red.append(colors[0])
                        green.append(colors[1])
                        blue.append(colors[2])
                if y < (height - 1):
                    colors = img.getpixel((x, y + 1))
                    red.append(colors[0])
                    green.append(colors[1])
                    blue.append(colors[2])
                new_red = int(sum(red)/len(red))
                new_blue = int(sum(blue)/len(blue))
                new_green = int(sum(green)/len(green))
                rgb = (new_red, new_green, new_blue)
                result.putpixel((x, y), rgb)
        return result


    def pentagram(img, x, y, r, thickness, color):
        drawing = ImageDraw.Draw(img)
        drawing.ellipse((x-r,    y-r,    x+r,    y+r),    fill=None,
outline=tuple(color), width=thickness)
        vertexes = []
        for i in range(5):
            phi = (math.pi/5)*(2*i+3/2)
            vertex = (int(x+r*math.cos(phi)),int(y+r*math.sin(phi)))
            vertexes.append(vertex)
        for i in range(5):
            drawing.line(xy=(vertexes[i],          vertexes[(i+2)%5]),
fill=tuple(color), width=thickness, joint=None)
        return img
```