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. 3343 _____ Отмахов Д. В.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Изучить принципы работы с модулем Pillow (PIL) языка Python. Реализовать программу, обрабатывающую изображение.

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:

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

Выполнение работы

Функция pentagram(img, x, y, r, thickness, color) принимает на вход изображение (img), координаты центра окружности (x, y), радиус окружности (r), толщину линий и окружности (thickness), цвет линий и окружности (color). Преобразовывает изображение следующим образом: рисует на нем пентаграмму в окружности. Функция возвращает преобразованное изображение (img).

Функция swap(img, x0, y0, x1, y1, width) принимает на вход квадратное изображение (img), координаты левого верхнего угла первого квадратного участка (x0, y0), координаты левого верхнего угла второго квадратного участка (x1, y1) и длину стороны квадратных участков (width). Функция создает копию изображения (new_img) и преобразовывает ее следующим образом: меняет местами переданные участки изображений, поворачивает каждый из них на 90 градусов по часовой стрелке, а затем поворачивает все изображение на 90 градусов по часовой стрелке. Функция возвращает преобразованное изображение (new_img).

Функция avg_color(img, x0, y0, x1, y1) принимает на вход изображение (img), координаты левого верхнего угла области (x0,y0), координаты правого нижнего угла области (x1,y1). Функция создает копию изображения (new_img) и преобразовывает ее следующим образом: заменяет цвет каждого пикселя в заданной области на средний цвет пикселей вокруг. Функция возвращает преобразованное изображение (new_img).

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№	Входные данные	Выходные данные	Комментарий
1.	img.size = (300, 300); x = 44; y = 99; r = 15; thickness = 3; color = [38, 12, 84]	img	Функция pentagram().
2.	img.size = (300, 300); x0 = 0; y = 0; x1 = 150; y1 = 150; width = 150	new_img	Функция swap().
3.	img.size = (300, 300); x0 = 13; y0 = 93; x1 = 235; y1 = 180	new_img	Функция avg_color().

Выводы

Были изучены принципы работы с модулем Pillow (PIL) языка Python. Была реализована программа, обрабатывающая изображение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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


def pentagram(img, x, y, r, thickness, color):
    drawing = ImageDraw.Draw(img)
    vertices = []
    for i in range(5):
        phi = (pi / 5) * (2 * i + 3 / 2)
        node_i = (int(x + r * cos(phi)), int(y + r * sin(phi)))
        vertices.append(node_i)
    for i in range(5):
        node1 = vertices[i]
        node2 = vertices[(i + 2) % 5]
        drawing.line([node1, node2], tuple(color), thickness)
    drawing.ellipse([(x - r, y - r), (x + r, y + r)], None, tuple(color),
thickness)
    return img


def swap(img, x0, y0, x1, y1, width):
    new_img = img.copy()
    cropped_img1 = img.crop((x0, y0, x0 + width, y0 + width))
    cropped_img2 = img.crop((x1, y1, x1 + width, y1 + width))
    new_img.paste(cropped_img1.transpose(Image.Transpose.ROTATE_270),
(x1, y1))
    new_img.paste(cropped_img2.transpose(Image.Transpose.ROTATE_270),
(x0, y0))
    new_img = new_img.transpose(Image.Transpose.ROTATE_270)
    return new_img


def avg_color(img, x0, y0, x1, y1):
    new_img = img.copy()
    for x in range(x0, x1 + 1):
        for y in range(y0, y1 + 1):
            neighbors = [img.getpixel((x - 1, y)), img.getpixel((x - 1,
y - 1)),
                         img.getpixel((x, y - 1)), img.getpixel((x + 1,
y - 1)),
                         img.getpixel((x + 1, y)), img.getpixel((x + 1,
y + 1)),
                         img.getpixel((x, y + 1)), img.getpixel((x - 1,
y + 1))]
            new_color = (int(sum(neighbor[0] for neighbor in neighbors)
/ len(neighbors)),
                         int(sum(neighbor[1] for neighbor in neighbors)
/ len(neighbors)),
                         int(sum(neighbor[2] for neighbor in neighbors)
/ len(neighbors)))
```

```
            new_img.putpixel((x, y), new_color)
    return new_img
```