

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. 3341

Кудин А.А.

Преподаватель

Молодцев Д.А.

Санкт-Петербург

2023

Цель работы

Целью работы было изучение и практическое применение библиотеки Pillow(PIL) и numpy. А именно написание 3 подзадач, реализованных в функциях, которые используют эти библиотеки.

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 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.

Примечание:

Вершины пентаграммы высчитывать по формуле:

$$\text{phi} = (\text{pi}/5) * (2*i + 3/2)$$
$$\text{node_i} = (\text{int}(x0 + r * \cos(\text{phi})), \text{int}(y0 + r * \sin(\text{phi})))$$

$x0, y0$ - 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*.

$(\text{int}(\text{sum}(r)/\text{count}), \text{int}(\text{sum}(g)/\text{count}), \text{int}(\text{sum}(b)/\text{count}))$

Выполнение работы

1. Функция `pentagram`:

- Назначение: Рисует пятиконечную звезду (пентаграмму) на изображении.
- Параметры:
 - image: Изображение, на котором будет нарисована пентаграмма.
 - centerX, centerY: Координаты центра пентаграммы.
 - radius: Радиус окружности, в которую вписана пентаграмма.
 - line_thickness: Толщина линий звезды.
 - line_color: Цвет линий.
- Логика: Функция вычисляет вершины звезды, рисует окружность вокруг будущей звезды и затем соединяет вершины линиями.

2. Функция `crop_and_rotate`:

- Назначение: Вырезает часть изображения и вращает её.
- Параметры:
 - image: Исходное изображение.
 - posX, posY: Координаты верхнего левого угла области для вырезки.
 - size: Размер вырезаемой области.
- Логика: Вырезает квадратную область изображения и вращает её на 270 градусов.

3. Функция `swap`:

- Назначение: Меняет местами две области на изображении и вращает результирующее изображение.
- Параметры:
 - original_image: Исходное изображение.
 - posX1, posY1, posX2, posY2: Координаты областей для обмена.
 - size: Размер областей для обмена.
- Логика: Вырезает и вращает две области изображения, обменивает их местами, а затем вращает весь результат.

4. Функция ``calculate_avg_color``:

- Назначение: Вычисляет средний цвет пикселей вокруг заданной точки.
- Параметры:
 - `source_image`: Изображение для анализа.
 - `coordX`, `coordY`: Координаты центрального пикселя.
- Логика: Собирает цвета пикселей вокруг указанной точки и вычисляет их среднее значение.

5. Функция ``avg_color``:

- Назначение: Применяет функцию `calculate_avg_color` к каждому пикселю в заданной области изображения.
- Параметры:
 - `image`: Изображение для обработки.
 - `leftX`, `topY`, `rightX`, `bottomY`: Границы области обработки.
- Логика: Перебирает пиксели в заданной области, вычисляя и устанавливая средний цвет для каждого из них.

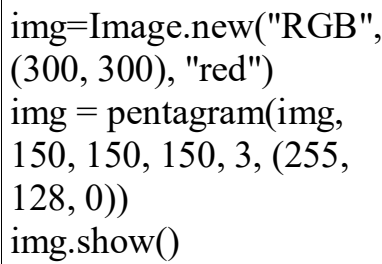




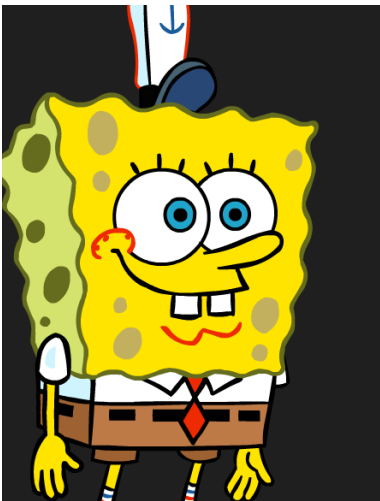
6. Функция ``calculate_vertex``:

- Назначение: Вычисляет координаты вершины пятиугольника или звезды.
- Параметры:
 - `number`: Номер вершины.
 - `centerX`, `centerY`: Координаты центра.
 - `radius`: Радиус.
- Логика: Вычисляет координаты вершины на основе угла и радиуса относительно центра.

Тестирование

Разработанный программный код см. в приложении А.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	 <pre>img=Image.new("RGB", (300, 300), "red") img = pentagram(img, 150, 150, 150, 3, (255, 128, 0)) img.show()</pre>	
2.		
3.		

Выводы

Были изучены библиотеки Pillow(PIL) и numpy в языке Python.

Создана и описана программа для решения 3 задач, которые были описаны.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

def pentagram(image, centerX, centerY, radius, line_thickness,
line_color):
    pentagram_draw = ImageDraw.Draw(image)

    star_vertices = []
    for vertex in range(5):
        angle = (math.pi / 5) * (2 * vertex + 1.5)
        vertex_x = int(centerX + radius * math.cos(angle))
        vertex_y = int(centerY + radius * math.sin(angle))
        star_vertices.append((vertex_x, vertex_y))

    pentagram_draw.ellipse((centerX - radius, centerY - radius,
centerX + radius, centerY + radius), outline=tuple(line_color),
width=line_thickness)

    for vertex in range(5):
        next_vertex = (vertex + 2) % 5
        pentagram_draw.line((star_vertices[vertex],
star_vertices[next_vertex]), fill=tuple(line_color),
width=line_thickness)

    return image

def crop_and_rotate(image, posX, posY, size):
    cropped_image = image.crop((posX, posY, posX + size, posY +
size))
    rotated_image = cropped_image.rotate(270)
    return rotated_image
```

```

def swap(original_image, posX1, posY1, posX2, posY2, size):
    part1 = crop_and_rotate(original_image, posX1, posY1, size)
    part2 = crop_and_rotate(original_image, posX2, posY2, size)
    result_image = original_image.copy()
    result_image.paste(part1, (posX2, posY2))
    result_image.paste(part2, (posX1, posY1))
    return result_image.rotate(270)

def calculate_avg_color(source_image, coordX, coordY):
    image_width, image_height = source_image.size
    pixel_data = source_image.load()
    adjacent_pixels = []
    for deltaX in range(-1, 2):
        for deltaY in range(-1, 2):
            if 0 <= (coordX + deltaX) < image_width and 0 <= (coordY
+ deltaY) < image_height:
                adjacent_pixels.append(pixel_data[coordX + deltaX,
coordY + deltaY])
    adjacent_pixels.remove(pixel_data[coordX, coordY])
    avg_color = [0, 0, 0]
    for color in adjacent_pixels:
        for color_index in range(3):
            avg_color[color_index] += color[color_index]
    avg_color = [int(color_sum / len(adjacent_pixels)) for color_sum
in avg_color]
    return tuple(avg_color)

def avg_color(image, leftX, topY, rightX, bottomY):
    image_copy = image.copy()
    image_pixels = image_copy.load()
    for posX in range(leftX, rightX + 1):
        for posY in range(topY, bottomY + 1):
            image_pixels[posX, posY] = calculate_avg_color(image,
posX, posY)
    return image_copy

def calculate_vertex(number, centerX, centerY, radius):
    angle = (np.pi / 5) * (2 * number + 1.5)

```

```
        vertex = (int(centerX + radius * np.cos(angle)), int(centerY +  
radius * np.sin(angle)))  
    return vertex
```