

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

Бойцов В.А.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Ознакомиться с функционалом библиотеки Pillow и решить 3 подзадачи с использованием её возможностей и библиотеки NumPy.

Задание

Вариант 3

1) Рисование пентаграммы в круге

Необходимо написать функцию `pentagram()`, которая рисует на изображении пентаграмму в окружности.

Функция `pentagram()` 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.

2) Поменять местами участки изображения и поворот

Необходимо реализовать функцию `swar()`, 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 `swar()` 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 `avg_color()`, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).

Функция `avg_color()` 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 углу

Функция должна вернуть обработанное изображение, не изменяя исходное.

Основные теоретические положения

Pillow - это форк библиотеки *PIL* (*Python Imaging Library*), предназначенной для работы с изображениями в Python. *Pillow* предоставляет средства для открытия, редактирования и сохранения различных форматов изображений.

Image - это модуль, предоставляющий класс с тем же именем (*Image*), который используется для представления изображений в *Pillow*. Этот класс обеспечивает широкий спектр методов для работы с изображениями, включая открытие изображений из файлов, создание новых изображений и выполнение различных операций редактирования.

ImageDraw - это модуль, предоставляющий класс с тем же именем (*ImageDraw*), который позволяет рисовать на изображении. Этот класс содержит методы для рисования геометрических фигур, текста и других элементов на изображении.

Выполнение работы

Из библиотеки *PIL* импортируются модули *Image* и *ImageDraw*, также подключается библиотека *numpy*.

Далее описываются следующие функции:

def make_img_part(img, x, y, width). Вспомогательная функция для второй подзадачи. Принимает на вход изображение *img*, координаты области *x* и *y* и сторону области *width*. Создаётся переменная *imgResult*, в которую с помощью метода *img.crop()* записывается квадратная область изображения. Далее с помощью метода *imgResult.rotate()* изображение поворачивается по часовой стрелке на 90 градусов (т.е. методу передаётся число 270).

def swap(img, x0, y0, x1, y1, width). Функция для решения второй подзадачи. Её входные и выходные данные, как и входные и выходные данные всех функций для решения подзадач, совпадают с перечисленными в задании. Создаются переменные *img1* и *img2*, в которые функция *make_img_part()* возвращает соответствующие области исходного изображения *img*. Создаётся переменная *imgResult*, в которую копируется изображение *img* с помощью метода *img.copy()*, далее с помощью метода *imgResult.paste()* вставляются области, записанные в переменных *img1* и *img2*, после чего полученное изображение поворачивается на 90 градусов по часовой стрелке и возвращается функцией как *imgResult*.

def get_color(img, x, y). Вспомогательная функция для третьей подзадачи. Принимает на вход изображение *img* и координаты пикселя *x* и *y*. В переменные *width* и *height* записываются размеры изображения *img*, далее в переменную *pixels* с помощью метода *img.load()* записывается информация о всех пикселях, составляющих изображение. Создаётся пустой список *pixelsAround*, в котором будут храниться данные о цвете каждого пикселя вокруг пикселя с переданными координатами. Затем в двух вложенных циклах *for*, итераторы *i* и *j* которых пробегают значения от -1 до 1, проверяется, существуют ли пиксели вокруг пикселя с координатами *x* и *y*, и, если существуют, данные о них записываются в *pixelsAround* (кроме центрального). Затем создаётся список

resultColor, в котором будет храниться средний цвет пикселей. Далее в каждый из трёх компонентов этого списка записывается сумма соответствующей компоненты каждого пикселя из списка *pixelsAround*. Затем для каждой компоненты находится среднее значение (с помощью деления на количество элементов в списке *pixelsAround*). Функция возвращает кортеж списка *resultColor*.

def avg_color(img, x0, y0, x1, y1). Функция для решения третьей подзадачи. Создаётся копия изначального изображения *imgResult*, а также в переменную *pixels* записывается информация о каждом пикселе исходного изображения с помощью *imgResult.load()*. Затем в двух вложенных циклах *for*, итераторы *x* и *y* которых пробегают все значения от *x0* до *x1* и от *y0* до *y1* соответственно, цвет *[x,y]*-го пикселя задаётся функцией *get_color()*. Функция возвращает *imgResult*.

def get_node(num, x, y, r). Вспомогательная функция для первой подзадачи. Принимает на вход номер вершины *num*, координаты центра окружности *x* и *y* и её радиус *r*. Создаётся переменная *phi*, в которую записывается угол отклонения, соответствующий номеру вершины *num*. Далее в переменную *node* записываются координаты вершины пентаграммы. Функция возвращает переменную *node*. В функции для вычисления координат используются методы (*numpy.sin()* и *numpy.cos()*) и константы (число Π) из библиотеки *numpy*.





def pentagram(img, x, y, r, thickness, color). Функция для решения первой подзадачи. Создаётся объект для рисования *drawing* для изображения *img* с помощью метода *ImageDraw.Draw()*. Список *color* приводится к кортежу. С помощью метода *drawing.ellipse()* отрисовывается окружность с центром *(x, y)*, радиусом *r*, толщины *thickness* и цвета *color* (параметр *None* отвечает за заливку – она не нужна). Далее для каждого номера вершины от 0 до 4 в цикле *for* вычисляются координаты вершины *node_1* и *node_2* с помощью функции *get_node()*, причем номер второй вершины на два больше первой – так обеспечивается цикличность отрисовки линий. Затем отрисовывается отрезок между этими вершинами с помощью метода *drawing.line()*. Функция возвращает изображение *img*.

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.	<pre>img=Image.new("RGB", (200, 200), "black") img = pentagram(img, 100, 100, 95, 3, (255, 128, 0))</pre>		Пентаграмма нарисована правильно
2.			Области вырезаны и вставлены верно, повороты правильные
3.			Цвета пикселей подобраны правильно, границы области соблюдены

Выводы

В результате выполнения работы были освоены основные возможности библиотеки Pillow, а также была написана программа, использующая библиотеки Pillow и Numpy и реализующая 3 подзадачи по обработке изображений и выполнению операций с ними.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

def make_img_part(img, x, y, width):
    imgResult = img.crop((x, y, x+width, y+width))
    imgResult = imgResult.rotate(270)
    return imgResult

def swap(img, x0, y0, x1, y1, width):
    img1 = make_img_part(img, x0, y0, width)
    img2 = make_img_part(img, x1, y1, width)
    imgResult = img.copy()
    imgResult.paste(img1, (x1, y1))
    imgResult.paste(img2, (x0, y0))
    imgResult = imgResult.rotate(270)
    return imgResult

def get_color(img, x, y):
    width, height = img.size
    pixels = img.load()
    pixelsAround = []
    for i in range(-1, 2):
        for j in range(-1, 2):
            if (0 <= (x+i) < width and 0 <= (y+j) < height):
                pixelsAround.append(pixels[x+i, y+j])
    pixelsAround.remove(pixels[x, y])
    resultColor = [0, 0, 0]
    for color in pixelsAround:
        for rgb in range(3):
            resultColor[rgb] += color[rgb]
    for rgb in range(3):
        resultColor[rgb] = int(resultColor[rgb] / len(pixelsAround))
    return tuple(resultColor)

def avg_color(img, x0, y0, x1, y1):
    imgResult = img.copy()
    pixels = imgResult.load()
    for x in range(x0, x1+1):
        for y in range(y0, y1+1):
            pixels[x, y] = get_color(img, x, y)
    return imgResult

def get_node(num, x, y, r):
    phi = (np.pi/5) * (2*num + 3/2)
    node = (int(x + r*np.cos(phi)), int(y + r*np.sin(phi)))
    return node

def pentagram(img, x, y, r, thickness, color):
    drawing = ImageDraw.Draw(img)
    color = tuple(color)
```

```
        drawing.ellipse(((x-r, y-r),(x+r, y+r)), None, color,
thickness)
    for i in range(0, 5):
        node_1 = get_node(i, x, y, r)
        node_2 = get_node(i+2, x, y, r)
        drawing.line((node_1, node_2), color, thickness)
    return img
```