

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. 2381

Кривов С.А.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы

Изучить библиотеку Pillow.

Задание

Вариант 2

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 `pentagram()`, которая рисует на изображении пентаграмму в круге.

Функция `pentagram()` принимает на вход:

Изображение (`img`)

Координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность (`x0,y0,x1,y1`)

Толщину линий и окружности (`thickness`)

Цвет линий и окружности (`color`) - представляет собой список (`list`) из 3-х целых чисел

Функция должна вернуть обработанное изображение.

2) Инвертирование полос

Необходимо реализовать функцию `invert`, которая делит изображение на "полосы" и инвертирует цвет нечетных полос (счёт с нуля).

Функция `invert()` принимает на вход:

Изображение (`img`)

Ширину полос в пикселах (`N`)

Признак того, вертикальные или горизонтальные полосы (`vertical` - если `True`, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной `N` пикселей. И инвертировать цвет нечетных полос.

Последняя полоса может быть меньшей ширины, чем `N`.

3) Поменять местами 9 частей изображения

Необходимо реализовать функцию `mix`, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция `mix()` принимает на вход:

Изображение (`img`)

Словарь с описанием того, какие части на какие менять (`rules`)

Пример словаря `rules`: {0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна вернуть обработанное изображение.

Можно реализовывать дополнительные функции.

Выполнение работы

Импортируем модуль *numpy* и пакеты библиотеки *Pillow*: *Image*, *ImageDraw*, *ImageOps* для работы с изображениями.

Функция *pentagram*

Принимает на вход: изображение (*img*), координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность (*x0,y0,x1,y1*), толщину линий и окружности (*thickness*), цвет линий и окружности (*color*) - представляет собой список (*list*) из 3-х целых чисел.

Возвращает обработанное изображение.

Приводим список *color* к кортежу для работы с функциями пакета *ImageDraw*.

Вычисляем радиус и центр окружности, вписанной в квадрат.

Создаем объект *drawing* для рисования. Рисуем окружность методом *ellipse()*.

Находим вершины пентаграммы *coordinates*: вычисляем вершины пятиугольника только проходим их через одну.

Рисуем ломанную в виде звезды, с помощью метода *line*.

Возвращаем полученное изображение.

Функция *invert*

Принимает на вход: изображение (*img*), ширину полос в пикселях (*N*), признак того, что полосы вертикальные или горизонтальные (*vertical* - если *True*, то вертикальные)

Функция разделяет изображение на вертикальные или горизонтальные полосы шириной *N* пикселей. И инвертирует цвет в нечетных полосах.

Инвертируем цвета копии изображения с помощью *Image.Ops.invert()*. В зависимости от значения *vertical* в цикле *for* копируем в исходное изображение из инвертированного нечетные вертикальные (*vertical == True*) или горизонтальные (*vertical == False*) полосы. Для этого вычисляем значения начала и конца полосы, если конец полосы выходит за границы изображения, считаем его равным ширине/высоте.

Возвращаем полученное изображение.

Функция *mix*

Принимает на вход: изображение (*img*), словарь с описанием того, какие части на какие менять (*rules*).

Функция делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Получаем размеры изображения, находим длину стороны одной части.

Создаем пустое новое изображение с размерами идентичными исходному.

В цикле проходимся по номеру *n* каждому региону. Из словаря *rules* достаем номер *m* другого региона, на который требуется заменить. Копируем регион *m* в пустой регион *n*.









Возвращаем полученное изображение.

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
Функция <i>pentagram</i>			
1.			<i>pentagram(img, 15, 100, 115, 200, 1, (171, 84, 171))</i> OK
Функция <i>invert</i>			
2.			<i>invert(img, 50, False)</i> OK
3.			<i>invert(img, 100, True)</i> OK
Функция <i>pentagram</i>			
4.			<i>mix(img, {0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8})</i> OK

Выводы

Была изучена библиотека Pillow.

Разработаны функция `pentagram`, рисующая пентаграмму в заданном квадрате на изображении, функция `invert`, инвертирующая цвет чередующихся полос заданной ширины, функция `mix`, меняющая части изображения местами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *main.py*

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

def pentagram(img, x0, y0, x1, y1, thickness, color):
    color = tuple(color)
    # # радиус и центр окружности
    r = int(abs(x1 - x0) // 2)
    center_x = int(x1 - (abs(x1 - x0) // 2))
    center_y = int(y1 - (abs(y1 - y0) // 2))
    drawing = ImageDraw.Draw(img)
    drawing.ellipse((x0, y0, x1, y1), fill = None , width= thickness,
outline= color )
    coordinates = []
    for i in range(0, 6):
        phi = (np.pi / 5) * (4 * i + 3 / 2)
        node_i = (int(center_x + r * np.cos(phi)), int(center_y + r *
np.sin(phi)))
        coordinates.append(node_i)
    drawing.line(coordinates, color, thickness)
    return img

def invert(img, N, vertical):
    img_invert = ImageOps.invert(img)
    width, height = img.size
    if vertical:
        for i in range(1, width // N+1, 2):
            x0 = i*N
            if (i+1)*N <= width:
                x1 = (i+1)*N
            else:
                x1 = width
            part = img_invert.crop((x0, 0, x1, height))
            img.paste(part, (x0, 0, x1, height))
    else:
        for i in range(1, height // N+1, 2):
            x0 = i*N
            if (i+1)*N <= height:
                x1 = (i+1)*N
```

```

        else:
            x1 = height
            part = img_invert.crop((0, x0, width, x1))
            img.paste(part, (0, x0, width, x1))
    return img

def mix(img, rules):
    width, height = img.size
    part_side = width // 3
    res = Image.new("RGB", (width, height), (0, 0, 0))
    for n in range(9):
        m = rules[n]
        x = m % 3
        y = m // 3
        part = img.crop((x * part_side, y * part_side, (x + 1) *
part_side, (y + 1) * part_side))
        x = n % 3
        y = n // 3
        res.paste(part, (x * part_side, y * part_side, (x + 1) *
part_side, (y + 1) * part_side))
    return res

```