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. 3344 _____ Жаворонок Д.Н.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Освоение обработки изображений на языке Python.

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.

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

Подсказка: Округляйте все вычисляемые вами значения (кроме значений углов) до целых чисел.

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 в нечетных полосах (счет с нуля). 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.

Выполнение работы

Из библиотеки *PIL* были импортированы *Image, ImageDraw, ImageOps*. Из библиотеки *numpy* были импортированы *pi, sin, cos, ceil*.

def pentagram(img, x0, y0, x1, y1, thickness, color) – функция, принимающая на вход *img – <class 'PIL.Image.Image'>*, в которой хранится первоначальное изображение, координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность – *(x0,y0,x1,y1)*. Используя *ImageDraw.Draw(img)* создается *drawer* для рисования на переданном изображении. Используя *drawer.ellipse(((x0, y0), (x1, y1)), outline=color, width=thickness)* был отрисован круг на *img*. Далее были рассчитаны координаты вершин пентаграммы и записаны в лист *vertices*. После в цикле с помощью *drawer.line((vertices[i], vertices[j], vertices[k]), color, thickness)* были нарисованы линии, соединяющие эти вершины. Функция возвращает измененное изображение img.

def invert(img, N, vertical) – функция, принимающая на вход *img – <class 'PIL.Image.Image'>*, ширину полос в пикселах (*N*), признак того, вертикальные или горизонтальные полосы(*vertical* - если *True*, то вертикальные). Используя *segment_count = int(ceil((width if vertical else height) / N))* рассчитывается полное количество сегментов размера *N* (последний сегмент может быть меньше). Используя *x1, x2 = N*i, clamp(N*(i+1), width)* в случае, если *vertical == True,* и *y1, y2 = N*i, clamp(N*(i+1), height)* в обратном, были рассчитаны координаты линий, в плоскости, в который они могут выйти за границы допустимых значений, чтобы этого избежать, была реализована функция *def clamp(n, bound)*, которая зажимает значения в допустимых пределах. Неизменяемые координаты были равны *0, height* или *0, width* в зависимости от положения. С помощью *img.crop((x1,y1,x2,y2))* были получены вырезки из изображения, которые затем были инвертированы используя *ImageOps.invert()*, после чего вставлены обратно на свое место через *img.paste()*. Функция возвращает измененное изображение img.

def mix(img, rules) – функция, принимающая на вход *img* – *<class 'PIL.Image.Image'>*, словарь с описанием того, какие части на какие менять (*rules*). Используя *side, _ = [x//3 for x in img.size]* рассчитывается длина стороны каждого сегмента. Используя *coords = (side*j, side*i, side*(j+1), side*(i+1))*, были рассчитаны координаты сегментов под номерами от 0 до 8 включительно. С помощью *crops.append((coords, img.crop(coords)))* были получены вырезки из изображения, которые затем были сохранены вместе с их координатами в *crops*, после чего, при проходе по *rules* были перемешаны с помощью *img.paste(final_crop, orig_coords),* где *_, final_crop = crops[final]*, а *orig_coords, _ = crops[orig]*. Функция возвращает измененное 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.	pentagram(Image.new("RGB", (300, 300), "black"), 0, 0, 300, 300, 4, [144, 166, 16])	Правильное изображение	-
2.	invert(pentagram(Image.new("RGB", (300, 300), "black"), 0, 0, 300, 300, 4, [144, 166, 16]), 40, False)	Правильное изображение	-
3.	mix(invert(pentagram(Image.new("RGB", (300, 300), "black"), 0, 0, 300, 300, 4, [144, 166, 16]), 40, False), {0: 4, 1: 1, 2: 2, 3: 1, 4: 7, 5: 4, 6: 2, 7: 8, 8: 7})	Правильное изображение	-

Выводы

Была освоена обработка изображений на языке Python с использованием модуля *Pillow*. Были получены навыки работы с пакетом, изучена работа с функциями рисования геометрических фигур.

Приложение А

Исходный код программы

Название файла: lb2.py

```
from numpy import cos, sin, pi, ceil
from PIL import Image, ImageDraw, ImageOps

def pentagram(img, x0, y0, x1, y1, thickness, color):
    color = tuple(color)
    drawer = ImageDraw.Draw(img)
            drawer.ellipse(((x0,   y0),   (x1,   y1)),   outline=color,
width=thickness)

    r = abs(x1 - x0) // 2
    x = x1 - (abs(x1 - x0) // 2)
    y = y1 - (abs(y1 - y0) // 2)

    vertices = []
    for i in [0, 2, 4, 1, 3]:
        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):
        j = (i+1)%5
        k = (i+2)%5
           drawer.line((vertices[i], vertices[j], vertices[k]), color,
thickness)

    return img

def clamp(n, bound):
    if n < 0: return 0
    if n > bound: return bound
    return n

def invert(img, N, vertical):
    width, height = img.size

    segment_count = int(ceil((width if vertical else height) / N))

    for i in range(1, segment_count, 2):
        if vertical:
            x1, x2 = N*i, clamp(N*(i+1), width)
            y1, y2 = 0, height
        else:
            x1, x2 = 0, width
            y1, y2 = N*i, clamp(N*(i+1), height)

        img.paste(ImageOps.invert(img.crop((x1,y1,x2,y2))), (x1, y1))

    return img

def mix(img, rules):
```

```
side, _ = [x//3 for x in img.size]

crops = []
for i in range(3):
    for j in range(3):
        coords = (side*j, side*i, side*(j+1), side*(i+1))
        crops.append((coords, img.crop(coords)))

for orig, final in rules.items():
    _, final_crop = crops[final]
    orig_coords, _ = crops[orig]
    img.paste(final_crop, orig_coords)

return img
```