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 _____ Калиберов Н.И.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Научиться работать с модулем Pillow (PIL), выполнять различные графические преобразования изображений.

Задание

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.

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(is_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

Подключаем модули PIL, math, numpy, для корректной работы программы. Функция pentagram(img, x0, y0, x1, y1, thickness, color) Для начала рассчитываем радиус окружности r и координаты её центра center_x, center_y. Далее преобразовываем list(color) в tuple(color), так как Pillow принимает цвет как tuple. Находим координаты вершин. С помощью методов ImageDraw на изображении рисуется пентаграмма. Функция invert(img, N, vertical) Для начала создаются переменные vercol и horcol - количество вертикальных и горизонтальных столбцов. Далее находим количество полос в зависимости от того вертикальные они, или горизонтальные. В цикле при помощи ImageOps.invert инвертируем цвета всех нечётных полос. Функция mix(img, rules) Для начала создаём массив пикселей numpy. Далее при помощи циклов создаём список с частями изображения. В следующем цикле части изображения собираются в нужном порядке по правилам rules. Далее собираем матрицу из частей при помощи vstack и hstack. Преобразуем полученную матрицу пикселей в изображение при помощи fromarray.

Выводы

Был успешно изучен модуль Pillow. При помощи модуля была созданна программа, которая выполняет поставленные задачи.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

def pentagram(img, x0, y0, x1, y1, thickness, color):
    r = (x1-x0)//2
    center_x = x0+(x1-x0)//2
    center_y = y0+(y1-y0)//2
    coordinates = []
    color = tuple(color)
    for i in range(1, 6):
        coordinate_x                                                    =
int(center_x+r*math.cos((math.pi/5)*(2*i+3/2)))
        coordinate_y                                                    =
int(center_y+r*math.sin((math.pi/5)*(2*i+3/2)))
        coordinat = (coordinate_x, coordinate_y)
        coordinates.append(coordinat)
    drawing = ImageDraw.Draw(img)
    drawing.line(((coordinates*2)[::2] + [coordinates[0]]), color,
thickness)
    drawing.ellipse((x0, y0, x1, y1), None, color, thickness)
    return img

def invert(img, N, vertical):
    vercol = img.size[0]
    horcol = img.size[1]
    if vertical:
        colums = math.ceil(vercol/N)
    else:
        colums = math.ceil(horcol/N)
    for i in range(1, colums, 2):
        if vertical:
            pol = (N*i, 0, min(N*(i+1), vercol), horcol)
        else:
            pol = (0, N*i, vercol, min(N*(i+1), horcol))
        inv_pol = ImageOps.invert(img.crop(pol))
        if vertical:
            img.paste(inv_pol, (N*i, 0))
        else:
            img.paste(inv_pol, (0, N*i))
    return img

def mix(img, rules):
    pix = np.array(img)
    width, height = img.size
    x, y = width//3, height//3
    box_l = [[] for i in range(9)]
    box = []
    for i in range(3):
        for j in range(3):
            box.append(pix[x*i:x*(i+1), y*j: y*(j+1)])
```

```
        for i in rules:
            box_l[i] = box[rules[i]]
        image     =     np.vstack([np.hstack((box_l[i],     box_l[i+1],
box_l[i+2])) for i in range(0, 7, 3)])
        img = Image.fromarray(image)
        return img
```