**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 |  | Лихацкий В. Р. |
| 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

Для выполнения первого задания во избежание большой вложенности в блоках кода был создан ряд вспомогательных функций: avg(\*args) – находит целочисленное среднее арифметическое аргументов; radius\_vector(center, dist, angle) – возвращает конец вектора, выходящего из точки center под углом angle, длиной dist; true\_polygon(n, center, radius, rotation=0) – возвращает вершины правильного n-угольника, с центром в точке center, вписанного в окружность радиусом r и повернутого на угол rotation в радианах. В функции pentogram находим центр нашей пентограммы как среднее арифметическое x0, x1 и y0, y1. Радиус пентограммы находится как половина разности x1 и x0. С помощью функции true\_polygon(5, center, radius, 3\*math.pi/10) получаем вершины правильного пятиугольника, повернутого на нужный угол. Для получения пентограммы соединяем отрезками вершины пятиугольника через одну, затем рисуем эллипс по заданным координатам.

В функции invert для удобства поворачиваем изображение если требуются вертикальные полосы. Задаем переменную offset=N – отступ перед следующей полосой которую требуется инвертировать. В цикле while увеличиваем offset на 2N чтобы инвертировать только нечетные полосы. Координаты текущей полосы равны (0, offset) и (ширина изображения, offset + N). С помощью функции crop копируем полосу, инвертируем ее цвет и вставляем с помощью paste. Остается только развернуть изображение обратно, если полосы были вертикальными.

В функции mix сначала создаем копию img result, затем получаем координаты частей изображения. С помощью цикла for rule in rules.items() и дальнейшей деструктуризации списка ключ-значение получаем какие части нужно заменить. Заменяем, копируя часть из оригинального img и вставляя в result и возвращаем result.

## Выводы

Были изучены различные способы взаимодействия с изображениями используя библиотеку Pillow. Для чистоты кода был использован прием декомпозиции функций. Во избежание необходимости в дополнительных проверках и костылях в задании 2 был применен прием предварительного преобразования изображения.

# Приложение А Исходный код программы

Название файла: main.py

import PIL

import math

from PIL import Image, ImageDraw, ImageOps

def avg(\*args):

return sum(args) // len(args)

def radius\_vector(center, dist, angle):

x, y = center

v = (

int(x + dist\*math.cos(angle)),

int(y + dist\*math.sin(angle))

)

return v

def true\_polygon(n, center, radius, rotation=0):

points = []

for i in range(n):

angle = 2\*i\*math.pi/n

point = radius\_vector(center, radius, angle + rotation)

points.append(point)

return points

def pentagram(img: Image.Image, x0, y0, x1, y1, thickness, color):

color = tuple(color)

radius = (x1 - x0) // 2

center = (avg(x1, x0), avg(y1, y0))

pentagon\_ = true\_polygon(5, center, radius, 3\*math.pi/10)

draw = ImageDraw.Draw(img)

draw.ellipse((x0, y0, x1, y1), None, color, thickness)

for i in range(5):

start = pentagon\_[i]

end = pentagon\_[(i + 2) % len(pentagon\_)]

draw.line((start, end), color, thickness, 'curve')

return img

def invert(img: Image.Image, N, is\_vertical):

img = img.rotate(-90 \* is\_vertical, Image.Resampling.NEAREST, True)

offset = N

while offset <= img.size[1]:

box = (0, offset, img.size[0], offset + N)

area = img.crop(box)

area = ImageOps.invert(area)

img.paste(area, box)

offset += 2\*N

img = img.rotate(90 \* is\_vertical, Image.Resampling.NEAREST, True)

return img

def mix(img: Image.Image, rules: "dict[int, int]"):

result = img.copy()

parts = []

w = img.size[0] // 3

h = img.size[1] // 3

for i in range(3):

for j in range(3):

parts.append((j \* w, i \* w, (j + 1) \* w, (i + 1) \* w,))

for rule in rules.items():

to\_, from\_ = rule

result.paste(img.crop(parts[from\_]), parts[to\_])

return result