**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. 3342 |  | Иванов Д. М. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Изучить библиотеку языка Python для рисования: Pyllow. С ее помощью написать программу, состоящую из трех функций, каждя из которых проводит операции с изображениями.

## Задание

Задача 1.

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.

Задача 2.

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).

Задача 3.

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)

## Выполнение работы

Для работы с изображениями и математических расчетов размеров фигур были использованы библиотеки *numpy* и *Pillow*. Рассмотрим каждую функцию в отдельности.

1. *def pentagram(img, x0, y0, x1, y1, thickness, color)*: Необходимо нарисовать пентаграмм в круге. Для этого сначала рисуется эллипс, затем расчитываются радиус окружности и вершины пентаграмма и в конце вершины соединяются.
2. *def invert(img, N, vertical):* Мы с помощью цикла пробегаем по нечетным полосам и с помощью функции *invert()* мы инвертируем в них цвет.
3. *def mix(img, rules):* Для начала создадим словарь с номером части и ее координатами на изображении. После разделим ее на эти части и сохраним их в массив. Создадим новый массив с сделанными изменениями и получим новое изображение.

Переменные:

* drawing – объект, на котором будет производиться рисование
* r – радиус окружности
* cen\_x, cen\_y – координаты центра окружности
* coords – массив с вершинами пентаграммы
* phi – угол
* node\_i - вершины
* w, h – размеры изображения
* k – номер части
* part, new\_part – часть изображения и ее перевернутый аналог
* dic – словарь с номером части изображения и ее координатами
* parts, result – массивы с изначальными частями и после перестановок
* w, h, size\_x, size\_y – размеры изображения
* img\_0 – часть изображения
* x0, y0, x1, y1 – координаты частей изображения

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | def pentagram(img, 50, 100, 120, 170, 3, [34, 56, 77]): | изображение | Верный вывод |
| 2. | def invert(img, 8, True) | изображение | Верный вывод |
| 3. | def mix(img, {0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}): | изображение | Верный вывод |

## Выводы

Была разработана программа, которая проводит различные операции с изображением. Изучена такая библиотека языка Python, как Pillow.

# Приложение А Исходный код программы

Название файла: main.py

import PIL, numpy

from PIL import Image, ImageDraw, ImageOps

from numpy import pi, cos, sin, ceil

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

drawing = ImageDraw.Draw(img)

drawing.ellipse([(x0, y0), (x1, y1)], width=thickness, outline=tuple(color))

r = (x1 - x0) // 2

cen\_x = ((x1 - x0) // 2) + x0

cen\_y = ((y1 - y0) // 2) + y0

coords = []

for i in range(5):

phi = (pi) \* (2 \* i + 3 / 2) /5

node\_i = (int(cen\_x + r \* cos(phi)), int(cen\_y + r \* sin(phi)))

coords.append(node\_i)

drawing.line([coords[0], coords[2]], fill=tuple(color), width=thickness)

drawing.line([coords[0], coords[3]], fill=tuple(color), width=thickness)

drawing.line([coords[1], coords[3]], fill=tuple(color), width=thickness)

drawing.line([coords[1], coords[4]], fill=tuple(color), width=thickness)

drawing.line([coords[2], coords[4]], fill=tuple(color), width=thickness)

return img

def invert(img, N, vertical):

w, h = img.size

k = 0

if vertical:

for x in range(0, w, N):

if k % 2 == 1:

part = img.crop((x, 0, x + N, h))

inv\_part = ImageOps.invert(part)

img.paste(inv\_part, (x, 0))

k += 1

else:

for y in range(0, h, N):

if k % 2 == 1:

part = img.crop((0, y, w, y + N))

inv\_part = ImageOps.invert(part)

img.paste(inv\_part, (0, y))

k += 1

return img

def mix(img, rules):

dic = {0: (0, 0), 1: (0, 1), 2: (0, 2), 3: (1, 0), 4: (1, 1), 5: (1, 2), 6: (2, 0), 7: (2, 1), 8: (2, 2)}

parts = [[0 for \_ in range(3)] for \_ in range(3)]

result = [[0 for \_ in range(3)] for \_ in range(3)]

w, h = img.size

size\_x = w // 3

size\_y = h // 3

for i in range(3):

for j in range(3):

img\_0 = img.crop((0 + j \* size\_y, 0 + i \* size\_x, 0 + (j + 1) \* size\_y, 0 + (i + 1) \* size\_x))

parts[i][j] = img\_0

result[i][j] = img\_0

for i in rules:

x0, y0 = dic[i]

x1, y1 = dic[rules[i]]

result[x0][y0] = parts[x1][y1]

for i in range(3):

for j in range(3):

img.paste(result[i][j], (0 + j \* size\_y, 0 + i \* size\_x))

return img