**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**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Смирнова Е.С. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## **Цель работы**

Изучить библиотеку *Pillow* и использовать полученные знания на примере практического задания.

## **Задание**

(Вариант 4)

​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 отрезка. Отрезок определяется:

координатами начала

координатами конца

цветом

толщиной.

Необходимо реализовать функцию *user\_func()*, рисующую на картинке отрезок

Функция *user\_func()* принимает на вход:

изображение;

координаты начала *(x0, y0)*;

координаты конца *(x1, y1)*;

цвет;

толщину.

Функция должна вернуть обработанное изображение.

2) Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется:

Координатами левого верхнего угла области;

Координатами правого нижнего угла области;

Алгоритмом, если реализовано несколько алгоритмов преобразования изображения (по желанию студента).

Нужно реализовать 2 функции:

*check\_coords(image, x0, y0, x1, y1)* - проверяет координаты области *(x0, y0, x1, y1)* на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку *x0, y0* - координаты левого верхнего угла, *x1, y1* - координаты правого нижнего угла, то *x1* должен быть больше *x0*, а *y1* должен быть больше *y0*);

*set\_black\_white(image, x0, y0, x1, y1)* - преобразовывает заданную область изображения в черно-белый (используйте для конвертации параметр '1'). В этой функции должна вызываться функция проверки, и, если область некорректна, то должно быть возвращено исходное изображение без изменений. Примечание: поскольку черно-белый формат изображения *(greyscale)* является самостоятельным форматом, а не вариацией *RGB*-формата, для его получения необходимо использовать метод Image.convert.

3) Найти самый большой прямоугольник заданного цвета и перекрасить его в другой цвет. Функционал определяется:

Цветом, прямоугольник которого надо найти

Цветом, в который надо его перекрасить.

Написать функцию *find\_rect\_and\_recolor(image, old\_color, new\_color)*, принимающую на вход изображение и кортежи *rgb*-компонент старого и нового цветов. Она выполняет задачу и возвращает изображение. При необходимости можно писать дополнительные функции.

## **Выполнение работы**

Для решения поставленных задач были написаны 3 функции с использованием подключаемого модулей *numpy* и *Pillow (PIL)*.

Функция *user\_func* принимает 6 аргументов: изображение, координаты *x0, y0, x1, y1*, цвет отрезка, её толщину. В ходе выполнения функции вызывается *ImageDraw* метод *Draw* для получения объекта для рисования. Далее, используя функцию *line* рисуется отрезок.

Функция *check\_coords* принимает 5 аргументов: изображение, координаты *x0, y0, x1, y1*. Используются переменные высоты и ширины изображения. Далее, с помощью условных конструкций проверяется корректность введенных координат.

Функция *set\_black\_white* принимает 5 аргументов: изображение, координаты *x0, y0, x1, y1*. В функции используется метод *check\_coords*, если он вернул *False*, изображение возвращается в исходном виде. Далее, из картинки вырезается обрабатываемая область c помощью метода *crop*, затем используя метод *convert* получаем Ч/Б изображение. Это изображение вставляется в изначальное место исходной картинки, а затем возвращается из функции.

Функция *find\_rect\_and\_recolor* принимает 3 аргумента: изображение, старый цвет, новый цвет. Преобразуется в двумерный числовой массив, где элементы, содержащие искомый цвет, заменяются на единицы, а остальные обнуляются. Далее, в ненулевых элементах записывается число, ненулевых элементов над ним. По каждой строке производится поиск наибольшей возможной площади для прямоугольника, сохраняя временные данные в *area*, а промежуточный результат в *max\_area*, *coordinates*. Eсли он не нашел самого большого прямоугольника заданного цвета, то возвращается исходное изображение. Иначе изображение преобразуется *narray* в переменную *arr*, затем используя эти координаты заменяются элементы старого цвета на новый по выделенной области. Массив декодируется обратно в картинку, переменную *image*, которая возвращается из функции.

Разработанный программный код см. в приложении А.

## **Выводы**

Была изучена библиотека *Pillow*, получены практические навыки использования библиотеки для работы с графическими данными. С помощью полученных знаний были составлены функции для решения практических задач: рисованиe отрезка, преобразованиe в Ч/Б изображение, нахождение самого большого прямоугольника заданного цвета и его перекрашивания в другой цвет..

# **ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.py

from PIL import Image, ImageDraw

import numpy as np

# Задача 1

def user\_func(image, x0, y0, x1, y1, fill, width):

drawing = ImageDraw.Draw(image)

drawing.line(((x0,y0),(x1,y1)), fill, width, joint=None)

return image

# Задача 2

def check\_coords(image, x0, y0, x1, y1):

if (x0>0 and x1>0 and y0>0 and y1>1 and x1>x0 and y1>y0 and image.size[0]>=x1 and image.size[1]>=y1):

return True

else:

return False

def set\_black\_white(image, x0, y0, x1, y1):

if (check\_coords(image, x0, y0, x1, y1) is True):

img = image.crop((x0, y0, x1, y1))

img = img.convert("1")

image.paste(img, (x0, y0))

return image

# Задача 3

def find\_rect\_and\_recolor(image, old\_color, new\_color):

arr = np.array(image).tolist()

for i in range(len(arr)):

for j in range(len(arr[i])):

arr[i][j] = int(arr[i][j] == list(old\_color))

arr = np.array(arr)

for i in range(1, len(arr)):

for j in range(len(arr[i])):

if arr[i][j] == 0:

arr[i][j] = 0

else:

arr[i][j] += arr[i - 1][j]

max\_area = 0

coordinates = (0, 0, 0, 0)

for i in range(len(arr)):

area = 0

for k in set(arr[i]):

for j in range(len(arr[i])):

if k <= arr[i][j]:

area += k

if j == len(arr[i]) - 1 or arr[i][j + 1] < k:

if max\_area < area:

max\_area = area

coordinates = (j - area // k + 1, i - k + 1, j, i)

area = 0

if coordinates == (0, 0, 0, 0):

return image

arr = np.array(image)

arr[coordinates[1]:coordinates[3] + 1, coordinates[0]:coordinates[2] + 1, :3] = list(new\_color)

image = Image.fromarray(arr)

return image