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. 3341 _____ Моисеева А.Е.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучить библиотеку Pillow, научиться работать с графическим типом данных. Используя усвоенные навыки, решить 3 подзадачи, каждая из которых должна выполнять определённый алгоритм действий и на выходе выдавать изображение.

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

Из библиотеки Pillow подключаются методы Image, ImageDraw. Подключается модуль numpy.

Функция user_func()

Функция принимает на вход изображение - image, координаты отрезка – x0, y0, x1, y1, цвет заливки – fill, толщину - width. Далее вызывается метод ImageDraw для получения объекта рисования. Затем с помощью функции line рисуется отрезок по заданным параметрам.

Функция check_coords()

Функция принимает на вход изображение - image, координаты – x0, y0, x1, y1. Переменные height, width принимают значение высоты и ширины изображения соответственно. В конце с помощью условных операторов if/else проверяется корректность входных координат.

Функция set_black_white()

Функция принимает на вход изображение - image, координаты – x0, y0, x1, y1. От этих же параметров вызывается функция check_coords. Если она возвращает 0 – изображение остаётся в исходном виде и тут же возвращается. Далее с помощью метода crop выбирается фрагмент исходного изображения по заданным координатам. С использованием метода convert выбранный фрагмент изображения преобразуем в чёрно-белый формат. Изменённую часть вставляем на изначальное место и возвращаем результат.

Функция find_rect_and_recolor()

На вход принимается изображение - image, исходный цвет – old_color и цвет после обработки – new_color. Переменная coordinates задаётся с помощью функции find_max_rect от параметров image, old_color. Если она возвращает значение (0, 0, 0, 0), то возвращаем исходное изображение без изменений. Происходит объявление переменной arr с помощью функции numpy.array от исходного изображения. По найденным координатам

элементы старого цвета необходимой области меняют цвет на новый. Массив arr декодируется в картинку image, возвращаемую функцией.

Функция find_max_rect()

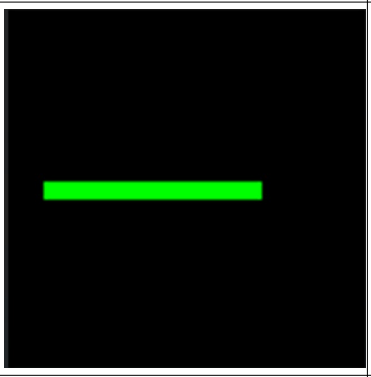
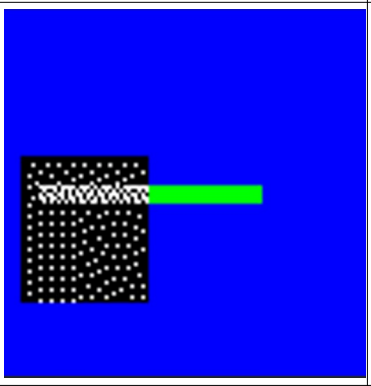
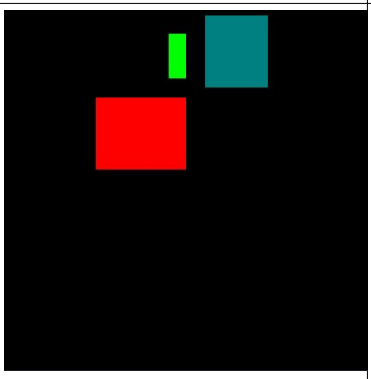
Функция принимает на вход изображение - image, цвет – color. Происходит преобразование в двумерный числовой массив, в котором элементы искомого цвета заменяются на единицы, а остальные на нули. Ненулевые элементы массива принимают значение количества ненулевых элементов над ним. По каждой строке происходит поиск максимально-возможной площади прямоугольника. При этом временные данные сохраняются в space, промежуточный результат в max_space, coordinates. Функция возвращает координаты наибольшего прямоугольника искомого цвета.

Написанный 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.	image = Image.new("RGB", (100, 100), 'black') user_func(image, 10, 50, 70, 50, 'lime', 5)		Создаётся фоновое изображение, на нём с помощью функции user_func рисуется отрезок/
2.	image = Image.new("RGB", (100, 100), 'blue') user_func(image, 10, 50, 70, 50, 'lime', 5) set_black_white(image, 5, 40, 40, 80)		Создаётся фоновое изображение с отрезком, затем через функцию set_black_white фрагмент изображения меняет формат на чёрно-белый.
3.	img_or = Image.new("RGB", (400, 400), 'black') drawing = ImageDraw.Draw(img_or) drawing.rectangle((180, 30, 200, 80), 'lime', 1) drawing.rectangle((100, 100, 200, 180), 'red', 1) drawing.rectangle((220, 10, 290, 90), 'lime', 1) new_img = find_rect_and_recolor(img_or, (0, 255, 0), (0, 128, 128))		Создаётся фоновое изображение с тремя разными по размеру прямоугольниками, два из которых лаймового цвета. Затем с помощью функции find_rect_and_recolor больший из них перекрашивается в бирюзовый цвет.

Выводы

Была изучена библиотека Pillow, необходимая для работы с графическими данными. Разработана программа, выполняющая три подзадачи: рисование отрезка, преобразование фрагмента изображения в чёрно-белый формат, перекрашивание наибольшего прямоугольника определённого цвета в новый цвет.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.py

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

# Задача 1
def user_func(image, x0, y0, x1, y1, fill, width):
    draw = ImageDraw.Draw(image)
    draw.line((x0, y0, x1, y1), fill, width)
    return image

# Задача 2
def check_coords(image, x0, y0, x1, y1):
    height, width = image.height, image.width
    if ((x0 >= x1) or (y0 >= y1)):
        return False
    if ((x0 < 0) or (y0 < 0)):
        return False
    if ((x1 >= width) or (y1 >= height)):
        return False
    return True

def set_black_white(image, x0, y0, x1, y1):
    if check_coords(image, x0, y0, x1, y1) == 0:
        return image
    cropped_image = image.crop((x0, y0, x1, y1))
    cropped_image = cropped_image.convert("1")
    image.paste(cropped_image, (x0, y0))
    return image

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    coordinates = find_max_rect(image, old_color)
    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

def find_max_rect(image, 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(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_space = 0
        coordinates = (0, 0, 0, 0)
        for i in range(len(arr)):
            space = 0
            for k in set(arr[i]):
                for j in range(len(arr[i])):
                    if k <= arr[i][j]:
                        space += k
                    if j == len(arr[i]) - 1 or arr[i][j + 1] < k:
                        if max_space < space:
                            max_space = space
                             coordinates = (j - space // k + 1, i - k
+ 1, j, i)
                        space = 0
        return coordinates
```