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. 2381 _____ Соколов С.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2022

Цель работы.

Изучить библиотеку *Pillow*, получить практические навыки использования библиотеки для работы с графическими данными.

Задание.

Вариант 4.

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'>

Задача 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.

Задача 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. 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*.

Задача 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.

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 с 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_biggest_rect* принимает 2 аргумента: изображение, цвет искомого прямоугольника. 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*. После нахождения координат максимального прямоугольника заданного цвета возвращается переменная *coordinates.*

Функция *find_rect_and_recolor* принимает 3 аргумента: изображение, старый цвет, новый цвет. Вызывается метод *find_biggest_rect*, результат которого записывается *cords*, если 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.

Результаты тестирования см. в приложении Б.

Выводы.

Была изучена библиотека Pillow, получены практические навыки использования библиотеки для работы с графическими данными.

Разработана программа, включающая функции: рисования отрезка, преобразования 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=fill, width=width)
    return image


# Задача 2
def check_coords(image, x0, y0, x1, y1):
    height, width = image.height, image.width

    if not (0 <= x0 < width and 0 <= y0 < height):
        return False
    if not (0 < x1 < width and 0 < y1 < height):
        return False

    if not (x0 < x1 and y0 < y1):
        return False

    return True


def set_black_white(image, x0, y0, x1, y1):
    if not check_coords(image, x0, y0, x1, y1):
        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_biggest_rect(image, color):
    # to bin matrix
    arr = np.array(image).tolist()
    for i in range(len(arr)):
        for j in range(len(arr[i])):
            arr[i][j] = 1 if arr[i][j] == list(color) else 0
    arr = np.array(arr)

    # count max height
    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]

    # find max area
```

```
    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
                        # x0, y0, x1, y1
                        coordinates = (j - area // k + 1, i - k + 1, j, i)
                    area = 0

    return coordinates


def find_rect_and_recolor(image, old_color, new_color):
    coords = find_biggest_rect(image, old_color)

    if coords == (0, 0, 0, 0):
        return image

    arr = np.array(image)
    arr[coords[1]:coords[3] + 1, coords[0]:coords[2] + 1, :3] = list(new_color)
    image = Image.fromarray(arr)

    return image
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

На рисунке Б.1 изображено исходное изображение, которое подаётся на вход функции замены самого большого прямоугольника заданного цвета.

На рисунках Б.2-Б.8 представлены результаты работы функции замены цвета.

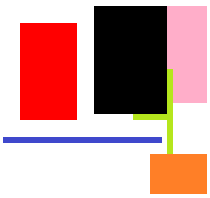

Рисунок Б.1 – Исходное изображение

Рисунок Б.2

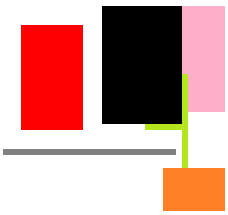

Рисунок Б.3

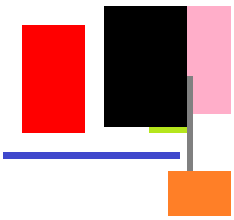

Рисунок Б.4

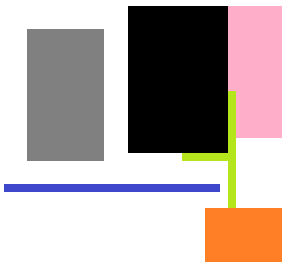

Рисунок Б.5

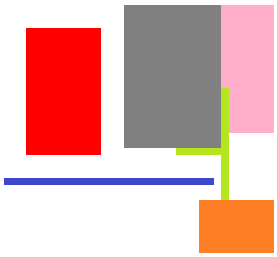

Рисунок Б.6

Рисунок Б.7

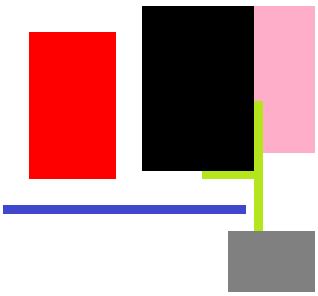

Рисунок Б.8

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Изображение Б.1, белый цвет	Изображение Б.2	Замена белого цвета; проверка поиска наибольшей области, проверка области, касающейся левой и нижней стороны.
2.	Изображение Б.1, синий цвет	Изображение Б.3	Замена синего цвета; проверка замены прямоугольника в 1px высотой.
3.	Изображение Б.1, лаймовый цвет	Изображение Б.4	Замена лаймового цвета; проверка замены прямоугольника в 1px шириной.
4.	Изображение Б.1, красный цвет	Изображение Б.5	Замена красного; обычный прямоугольник в центре.
5.	Изображение Б.1, черный цвет	Изображение Б.6	Замена черного; проверка поиска области, касающейся верхней границы.
6.	Изображение Б.1, розовый цвет	Изображение Б.7	Замена розового; проверка поиска области, касающейся верхней и правой границы.
7.	Изображение Б.1, оранжевый цвет	Изображение Б.8	Замена оранжевого; проверка поиска области, касающейся нижней и правой границы.