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 _____ Гончаров С.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Используя библиотеки Pillow и numpy выполнить поставленные в лабораторной работе задачи.

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 3 задач, были написаны 3 функции. В них использовались библиотеки PIL (для обработки картинок) и numpy (для преобразования изображения в массив).

Функция *user_func* принимает 6 аргументов: изображение (image), координаты (*x0, y0, x1, y1*), цвет отрезка (fill), её толщину (width). Внутри функции user_func вызывается функция ImageDraw с методом Draw. Используем метод line чтобы отрисовать отрезок по заданным координатам.

Функция *check_coords* принимает 5 аргументов: изображение (image), координаты (*x0, y0, x1, y1*). Используя метод (*.size) получаем значения ширины и высоты полученного изображения. С помощью условных конструкций проверяем полученные координаты.

Функция *set_black_white* принимает 5 аргументов: изображение (image), координаты (*x0, y0, x1, y1*). В функции используется *check_coords*, если функция возвращает *False*, то изображение возвращается в исходном виде. Из картинки вырезается обрабатываемая область с помощью метода *crop*, затем используя метод *convert* со значением ‘1’ чтобы получить черно-белое изображение. Это изображение вставляется в изначальное место исходной картинки, а затем возвращается из функции.

Функция *find_rect_and_recolor* принимает 3 аргумента: изображение (image), старый цвет (old_color), новый цвет (new_color). Изображение преобразуется в двумерный числовой массив, оператор array[i][j] = int(array[i][j] == list(old_color)) заменяет цвет пикселя на 1, если он равен старому цвету, и на 0, если он равен старому. Далее, оператор array[i][j] += array[i-1][j] прибавляет к элементу значение из предыдущей строки в том же столбце. По каждой строке производится поиск наибольшей возможной площади для прямоугольника, сохраняя временные данные в *rectangle*, а промежуточный результат в *max_rectanle*, *coordinates*. Если цикл не нашел прямоугольник заданного цвета, то возвращается исходное изображение. Иначе изображение преобразуется *narray* в переменную *array*, затем используя

эти координаты заменяются элементы старого цвета на новый по выделенной области. Массив декодируется обратно в картинку, используя метод (Image.fromarray(array)) и возвращает полученное обработанное изображение.

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: рисованиее отрезка, преобразованиее 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

def user_func(image, x0, y0, x1, y1, fill, width):
    drawing = ImageDraw.Draw(image)
    coordinates = (x0, y0, x1, y1)
    drawing.line(coordinates, fill, width)
    return image

def check_coords(image, x0, y0, x1, y1):
    width, height = image.size
    return (width >= x1) and (x1 > x0) and (x0 >= 0) and (height >= y1)
and (y1 > y0) and (y0 >= 0)

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

def find_rect_and_recolor(image, old_color, new_color):
    array = np.array(image).tolist()
    for i in range(len(array)):
        for j in range(len(array[i])):
            array[i][j] = int(array[i][j] == list(old_color))
    array = np.array(array)

    for i in range(1, len(array)):
        for j in range(len(array[i])):
            if array[i][j] == 0:
                array[i][j] = 0
            else:
                array[i][j] += array[i - 1][j]

    max_rectangle = 0
    coordinates = (0, 0, 0, 0)
    for i in range(len(array)):
        rectangle = 0
        for k in set(array[i]):
            for j in range(len(array[i])):
                if k <= array[i][j]:
                    rectangle += k

                if j == len(array[i]) - 1 or array[i][j + 1] < k:
                    if max_rectangle < rectangle:
                        max_rectangle = rectangle
                        coordinates = (j - rectangle // k + 1, i - k +
1, j, i)
                    rectangle = 0

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

```
        return image

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