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 Двиков Д.В

Преподаватель Шевская Н.В.

Санкт-Петербург
2022

Цель работы.

Изучить библиотеку pillow. Применить полученные знания для решения графических задач, представленных в работе

Задание.

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 импортируем Image, ImageDraw

Функция user_func получает на вход изображение, координаты начала(x0, y0), координаты конца(x1, y1), цвет отрезка и её толщину. Рисуется изображение при помощи метода “Image.Draw”. На созданной картинке, рисуется отрезок по таким параметрам как: координаты, цвет отрезка, его толщина

Функция check_coords получает на вход изображение и координаты. Данная функция, при помощи условных операторов, проверяет корректность заданных координат

Функция set_black_white_получает на вход: изображение и координаты. Функция начинается с того, что проверяет корректность координат при помощи функции check_coord 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_rect_and_recolor получает на вход изображение, старый цвет и новый цвет. Создается массив, в котором картинка преобразовывается в двоичную матрицу, где элементы массива равные единице – это искомый цвет, а остальные равны нулю. После чего мы ищем в ней максимальный квадрат из единиц. Каждый столбик превращается в гистограмму.

Функция max_area_histogram принимает на вход гистограммы из функции find_rect_and_recolor и находит максимальную площадь из них.

Выводы:

Была освоена библиотека Pillow. Полученные знания были применены на практике. Были разработаны такие функции как: рисования отрезка, преобразование изображения в черно-белый цвет, нахождения прямоугольника заданного цвета и его перекрашивания в другой цвет,

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

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

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

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

def max_area_histogram(hist):
    stack = []
    max_area = 0
    best_yy = None

    index = 0
    while index < len(hist):
        if (not stack) or (hist[stack[-1]] <= hist[index]):
            stack.append(index)
            index += 1
        else:
            top = stack.pop()
            yy = (stack[-1] + 1, index) if stack else (0, index)
            area = hist[top] * (yy[1] - yy[0])
            if area > max_area:
                max_area = area
                best_yy = yy

    while stack:
        top = stack.pop()
        yy = (stack[-1] + 1, index) if stack else (0, index)
```

```
            area = hist[top] * (yy[1] - yy[0])
            if area > max_area:
                max_area = area
                best_yy = yy

    return max_area, best_yy


def find_rect_and_recolor(image, old_color, new_color):
    if not isinstance(old_color, tuple):
        old_color = PIL.ImageColor.getrgb(old_color)
    width, height = image.size
    max_area = 0
    best_xy = None

    part = numpy.zeros(image.size, dtype=numpy.uint32)
    for i in range(width):
        for j in range(height):
            part[i, j] = int(image.getpixel((i, j)) == old_color)

    hist = numpy.zeros(height, dtype=numpy.uint32)
    for i in range(width):
        hist = numpy.multiply((hist + 1), part[i])
        area, yy = max_area_histogram(hist)
        if yy is not None:
            width = area // (yy[1] - yy[0])
            assert width == min(hist[yy[0] : yy[1]])
        if area > max_area:
            max_area = area
            best_xy = (i + 1 - width, yy[0]), (i, yy[1] - 1)

    if best_xy is not None:
        draw = ImageDraw.Draw(image)
        draw.rectangle(best_xy, fill=new_color, outline=new_color)
    return image
```