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. 3344 _____ Сербиновский Ю.М.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Научиться работать с библиотеками в языке программирования Python.

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

Задача 1: создана функции user_func, она рисует линию на изображении и возвращает измененное изображение.

Задача 2: созданы функции set_black_white() и check_coords. Первая функция сначала вырезает кусок изображения с помощью image.crop() и сохраняет его в img_cut, затем конвертирует его (с помощью img_cut.convert()) и вставляет в начальное изображение. Функция возращает измененное изображение. check_coords() проверяет переданные в функцию координаты на корректность.

Задача 3: создана функция find_rect_and_recolor(), которая сначала динамически ищет наибольшую прямоугольную матрицу в массиве цветов изображения, а затем в цикле поэлементно заменяет цвета найденной матрицы. Функция возвращает измененное изображение.

В функции find_rect_and_recolor() использованы переменные, функции и методы: img_height и img_width (высота и ширина изначального изображения), img_data — массив с цветами начального изображения, полученный с помощью list(image.getdata()), img_matrix хранит в себе те же значения что и img_data, но это матрица размера (h, w).

Динамический алгоритм: x1, x2, y1, y2 — координаты самой большой прямоугольной матрицы, area — площадь самого большого прямоугольника, она важна для нахождения координат, row_count — счет строк в цикле, heigth — массив который хранит в себе высоты столбцов, с прохождением по каждой строке значения массива либо зануляются, либо увеличиваются на один, это сделано для отсеивания неподходящих прямоугольников, массив stack сохраняет последние индексы массива height. Если текущий элемент height оказывается меньше элемента height по индексу последнего элемента stack, то удаляется последний элемент stack, одновременно перезаписывая h и w (высоту и ширину потенциального прямоугольника максимальной длины), делаем это (в цикле while) пока не встетим height[stack[-1]] меньший height[i]. Если произведение h и w оказывается больше area, мы перезаписываем координаты и area. Данные

махинации позволяют формировать прямоугольники и найти из них самый большой.

Благодаря полученным координатам пробегаемся в цикле по img_matrix, заменяя цвета. Далее мы переписываем значения из img_matrix в img_data. Помящаем img_data в изображение (image.putdata(img_data)) и возвращаем image.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	find_rect_and_recolor(Image.new('RGB', (200, 200), (255,0,0)), (255, 0, 0), (0,0,255))	image
2.	set_black_white(Image.new('RGB', (200, 200), (255,0,0)), 100, 50, 200, 100)	image
3.	user_func(Image.new('RGB', (200, 200), (255,0,0)), 100, 20, 40, 100, 'blue', 10)	image

Выводы

В ходе лабораторной были изучены некоторые библеотеки языка Python, в особенности Pillow.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
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=fill, width=width)
    #image.show()
    return image

# Задача 2
def check_coords(image, x0, y0, x1, y1):
    if x0 == abs(x0) and x1 == abs(x1) and y0 == abs(y0) and y1 == abs(y1):
        img_size = image.size
        if x0 <= img_size[0] and x1 <= img_size[0] and y0 <= img_size[1]
and y1 <= img_size[1]:
            if x0 < x1 and y0 < y1:
                return True
    else:
        return False

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

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    img_width = image.width
    img_height = image.height
    img_data = list(image.getdata())
    img_matrix = [[] for i in range(img_height)]
    k = 0
    for i in range(img_height):
```

```
        for j in range(img_width):
            img_matrix[i].append(img_data[k])
            k += 1

    x1 = y1 = x2 = y2 = 0
    height = [0] * (img_width + 1)
    area = 0
    row_count = -1
    for row in img_matrix:
        row_count += 1
        for i in range(img_width):
            height[i] = height[i] + 1 if row[i] == old_color else 0
        stack = [-1]
        for i in range(img_width + 1):
            while height[i] < height[stack[-1]]:
                h = height[stack.pop()]
                w = i - 1 - stack[-1]
                if area < h * w:
                    x2 = i - 1
                    y2 = row_count
                    x1 = stack[-1] + 1
                    y1 = row_count - h + 1
                    area = max(area, h * w)
            stack.append(i)

    for i in range(y1, y2+1):
        for j in range(x1, x2 + 1):
            img_matrix[i][j] = new_color
    k = 0
    for i in range(0, img_height):
        for j in range(0, img_width):
            img_data[k] = img_matrix[i][j]
            k += 1
    image.putdata(img_data)
    #image.show()
    return image

#find_rect_and_recolor(Image.new('RGB', (200, 200), (255,0,0)), (255, 0,
0), (0,0,255))
```

```
#user_func(Image.new('RGB', (200, 200), (255,0,0)), 100, 20, 40, 100, 'blue', 10)
#set_black_white(Image.new('RGB', (200, 200), (255,0,0)), 100, 50, 200, 100)
```