

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(ка) гр. 3343

Гельман П.Е.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Изучить функционал библиотеки Pillow языка Python, научиться применять его для обработки изображений внутри программ.

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

x_0 , y_0 - координаты левого верхнего угла, x_1 , y_1 - координаты правого нижнего угла, то x_1 должен быть больше x_0 , а y_1 должен быть больше y_0);

- `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 первой задачи была создана функция *user_func()*, которая принимает на вход изображение, координаты начала отрезка, координаты конца отрезка, цвет отрезка и его толщину. Чтобы нарисовать отрезок воспользуемся методом *line()* класса *Draw*. Созданная функция возвращает изменённое изображение.

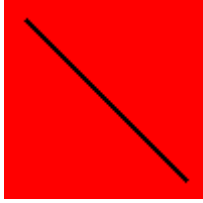
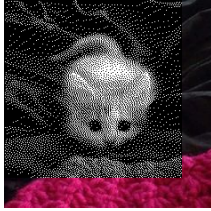

Решение второй задачи реализовано с помощью функций *check_coords()* и *set_black_white()*. Первая функция проверяет координаты заданной области на корректность. Вторая – преобразовывает эту область изображения в черно-белый цвет с помощью метода *convert()*.

Для решения третьей задачи были созданы функции *find_rect_and_recolor()* и *largest_rectangle()*. В функции *largest_rectangle()* подаются на вход изображение, нужный цвет и координаты пикселя. Далее в ней происходит вычисление размеров прямоугольника начиная с определенного пикселя. В функции *find_rect_and_recolor()* ведется поиск максимально большого прямоугольника заданного цвета, берётся каждый пиксель, вызывается функция *largest_rectangle()*, вычисляется площадь и сравнивается с предыдущей *max_rect_area*. После завершения внешнего цикла разукрашивается самый большой прямоугольник выбранным новым цветом *new_color*, используя функцию *ImageDraw.Draw.rectangle()*. В конце возвращается изменённое изображение.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<code>image = Image.new('RGB', (100, 100), color='red')</code>		Выходные данные задачи 1 соответствуют ожиданиям
2.	<code>image=Image.open("C:/Users/pgel6/Desktop/2bb4aea6b0fdc2b6d9adc300cfa7825c.jpg")</code> <code>result_image = set_black_white(image, 0, 0, 200, 200)</code>		Выходные данные задачи 2 соответствуют ожиданиям
3.	<code>image=Image.new("RGB", (100,100), 'black')</code> <code>image1=Image.new("RGB", (20,20), 'purple')</code> <code>image.paste(image1, (80,50))</code>		Выходные данные задачи 3 соответствуют ожиданиям

Выводы

В ходе выполнения лабораторной я научилась пользоваться библиотекой Pillow языка Python. Pillow предоставляет мощные инструменты для загрузки, изменения и сохранения изображений, что делает ее незаменимым инструментом для различных проектов, связанных с обработкой графики.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
# 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):
    width, height = image.size
    if x0 < 0 or y0 < 0 or x1 >= width or y1 >= height or x1
<= x0 or y1 <= y0:
        return False

    return True

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

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    pixels = image.load()
    width, height = image.size
    max_rect = (0, 0, 0, 0)
    max_rect_area = 0
    for x in range(width):
        for y in range(height):
            if pixels[x, y] == old_color:
                rect = largest_rectangle(image, old_color, x,
y)
                rect_area = (rect[2] - rect[0]) * (rect[3] -
rect[1]) # площадь
                if rect_area > max_rect_area:
                    max_rect = rect
                    max_rect_area = rect_area

    draw = ImageDraw.Draw(image)
    draw.rectangle(max_rect, fill=new_color)

    return image

def largest_rectangle(image, color, start_x, start_y):
    pixels = image.load()
    width, height = image.size
    x, y = start_x, start_y
    while x < width and pixels[x, y] == color:
        x += 1
    x -= 1
    while y < height and pixels[x, y] == color:
        y += 1
    y -= 1

    return (start_x, start_y, x, y)
```