

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. 3341

Трофимов В.О.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью работы является изучение библиотеки Pillow языка python и его практическое применение для решения трёх задач лабораторной работы.

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 (x_0, y_0);

координаты конца (x_1, y_1);

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

Определяются следующие функции для решения каждой из подзадач:

1. `def user_func(image, x0, y0, x1, y1, fill, width):`
2. `def check_coords(image, x0, y0, x1, y1):`
3. `def find_rect_and_recolor(image, old_color, new_color):`

Для решения задачи № 1 была определена функция `def user_func(image, x0, y0, x1, y1, fill, width):`

Данная функция `user_func()` принимает на вход изображение (`image`), координаты начала отрезка (`x0, y0`), координаты конца отрезка (`x1, y1`), цвет отрезка (`fill`) и ширину линии (`width`).

Сначала создается объект `ImageDraw.Draw`, который представляет собой класс для рисования на изображении. Затем с помощью метода `line()` задается рисование отрезка с координатами начала и конца, цветом `fill` и шириной `width`. После наложения отрезка на изображение, функция возвращает `image`.

Для решения задачи № 2 были определены функции `def check_coords(image, x0, y0, x1, y1):` и `set_black_white(image, x0, y0, x1, y1):`

Функция `check_coords(image, x0, y0, x1, y1)` предназначена для проверки корректности координат области на изображении. Она принимает на вход изображение и координаты (`x0, y0`), (`x1, y1`) левого верхнего и правого нижнего углов области.

Объявляются переменные `width, height` значение, которых получается с помощью `image.size`. Далее последовательно проверяются различные условия: координаты должны быть неотрицательными, а также правый нижний угол области должен иметь большую координату `x` и `y`, чем левый верхний угол, и не должен выходить за пределы ширины и высоты изображения. В случае выполнения всех условий возвращается значение `True`, указывающее на корректность координат. В случае несоответствия любому из условий возвращается `None` или `False`.

Функция `set_black_white(image, x0, y0, x1, y1)` предназначена для преобразования указанной области на изображении в черно-белый цвет.

Вызывается функция `check_coords()` для проверки корректности координат области. Если координаты области проверку не прошли, то функция возвращает исходное изображение без изменений. Если координаты области проверку прошли, то дальше объявляется переменная `area`, которая хранит в себе кортеж с координатами области (`x0, y0, x1, y1`). Объявляется переменная `cropped_area`, в которую записывают указанную область на изображении, используя метод `image.crop(area)`. Объявляется переменная `black_white_cropped_area`, которая хранит в себе конвертированную в черно-белый область (`cropped_area.convert("1")`). Исходная область изображения заменяется на полученную черно-белую область с помощью метода `image.paste(black_white_cropped_area, area)`. Функция возвращает изменённое изображение.

Для решения задачи № 3 были определены функции `def find_rect_and_recolor(image, old_color, new_color):` и `def flood_fill(x, y, width, height, pixels, old_color):`

Функция `flood_fill(x, y, width, height, pixels, old_color)` используется для поиска граничных координат прямоугольника. Она работает по принципу "заливки".

Поиск начинается от координат `x, y` и помещает их в стек. Далее извлекает координаты из стека и проверяет их цвет. Если цвет равен `old_color`, то он изменяется на новый цвет и координаты обновляются. После происходит добавление соседних пикселей в стек. Процесс продолжится до тех пор, пока весь прямоугольный регион не будет "залит", тем самым в результате будут известные крайние координаты самого большого прямоугольника заданного цвета. Функция возвращает координаты, представляющие граничный прямоугольник, вычисленные на основе минимальных и максимальных координат найденных пикселей.

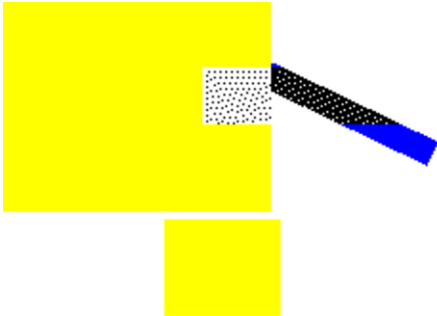
Функция `def find_rect_and_recolor(image, old_color, new_color)` принимает на вход изображение(`image`), цвет прямоугольника, который нужно изменить на новый цвет(`old_color`), новый цвет(`new_color`).

Сначала в функции делается копия входного изображения `image`. С помощью `copy_image.load()` и `image.load()` получаем доступ к пикселям исходного и скопированного изображения соответственно. Затем происходит инициализация начальных значений для координат наибольшего прямоугольника (`max_rectangle_coords`) и его площади (`max_rectangle_area`). Далее осуществляется проход по каждому пикселю изображения, проверяя, если цвет пикселя совпадает с `old_color`, то вызывается функция `flood_fill` для нахождения граничных координат прямоугольника и расчета его площади. Если найденная площадь прямоугольника больше, чем текущая максимальная, то обновляются координаты и площадь максимального прямоугольника. После этого происходит проход и перекраска всех пикселей максимального прямоугольника в новый цвет `new_color`. Функция возвращает изображение с перекрашенным самым большим прямоугольником заданного цвета.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	<pre>img = Image.new('RGB', (300, 200), 'white') user_func(img, 225, 80, 100, 20, 'blue', 15) func(img, 0, 0, 140, 110, 0, None, 1, 'yellow') func(img, 85, 115, 145, 165, 0, None, 1, 'yellow') set_black_white(img, 105, 35, 255, 65) find_rect_and_recolor(img, 'yellow', 'red') img.show()</pre>	

Выводы

Была изучена библиотека Pillow языка python и полученные знания были подкреплены практикой решением трёх задач.

В итоге была разработаны три функции: 1) Функция может нарисовать отрезок на изображении. 2) Функция может преобразовать в чёрно-белый цвет изображение определённую область изображения. 3) Функция находит в изображении самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.py

```
import PIL
import math
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

# Задача 2
def check_coords(image, x0, y0, x1, y1):
    width,height = image.size
    if (x0 >= 0) and (x1 >= 0) and (y0 >= 0) and (y1 >= 0):
        if (x1 > x0) and (y1 > y0) and (x1 < width) and (y1 < height):
            return True

def set_black_white(image, x0, y0, x1, y1):
    if not check_coords(image,x0,y0,x1,y1):
        return image
    area = (x0,y0,x1,y1)
    cropped_area = image.crop(area)
    black_white_cropped_area = cropped_area.convert("1")
    image.paste(black_white_cropped_area,area)
    return image

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    copy_image = image.copy()
    pixels = copy_image.load()
    res_pixels = image.load()
    width,height = image.size

    max_rectangle_coords = (0,0,0,0)
    max_rectangle_area = 0
    for x in range(width):
        for y in range(height):
            if pixels[x,y] == old_color:
                rectangle_coords =
flood_fill(x,y,width,height,pixels,old_color)
                coord_area = (rectangle_coords[2] -
rectangle_coords[0]) * (rectangle_coords[3] - rectangle_coords[1]) # площ
                адь прямоугольника
            if coord_area > max_rectangle_area:
                max_rectangle_coords = rectangle_coords
                max_rectangle_area = coord_area
```

```

        for x in range(max_rectangle_coords[0],max_rectangle_coords[2]):
            for y in range(max_rectangle_coords[1],max_rectangle_coords[3]):
                res_pixels[x,y] = new_color
    return image

def flood_fill(x, y, width, height, pixels, old_color):
    stack = [(x,y)]
    min_coord = [width,height] # минимальные координаты по x и y
    max_coord = [0,0] # максимальные координаты по x и y
    while stack:
        current_x,current_y = stack.pop()
        if 0 <= current_x < width and 0 <= current_y < height:
            if pixels[current_x,current_y] == old_color:
                pixels[current_x,current_y] = (0,0,0,0) # изменяет ц
вет, чтобы не было повторной обработки пикселя
                min_coord[0] = min(min_coord[0],current_x)
                min_coord[1] = min(min_coord[1],current_y)
                max_coord[0] = max(max_coord[0],current_x)
                max_coord[1] = max(max_coord[1],current_y)
                stack.append((current_x,current_y + 1)) #добавляются
координаты "соседних" пикселей и после уже они проходят по циклу
                stack.append((current_x,current_y - 1))
                stack.append((current_x + 1,current_y))
                stack.append((current_x - 1,current_y))
    return (min_coord[0],min_coord[1],max_coord[0] + 1, max_coord[1]
+ 1)

```