**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. 3341 |  | Яковлева А.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Изучение библиотеки *PILLOW(PIL),* умение работать с объектами типа *<class 'PIL.Image.Image'>.*

## 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.

1. 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.

1. 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

Функции:

* *user\_func(image, x0, y0, x1, y1, fill, width)* принимает на вход изображение, координаты начала и конца отрезка, цвет отрезка и его толщину, возвращает изображение, на котором с помощью *ImageDraw* и метода *line* нарисован отрезок с заданными координатами.
* *check\_coords(image, x0, y0, x1, y1)* принимает на вход изображение, координаты левого верхнего и правого нижнего углов области, возвращает *True*, если координаты неотрицательные и не превышают размеров изображения, иначе возвращает *False*.
* *set\_black\_white(image, x0, y0, x1, y1)* принимает на вход изображение, координаты левого верхнего и правого нижнего углов области, и, если координаты корректны, с помощью метода *crop()* вырезается прямоугольник, который конвертируется из RGB-формата в чёрно-белый методом *convert(),* и методом *paste()* полученное изображение комбинируется с основным. Возвращает изображение *image*.
* *find\_rect\_and\_recolor(image, old\_color, new\_color)* принимает на вход изображение, цвет прямоугольника, который нужно найти и цвет, в который его нужно перекрасить. С помощью функции *load()* получает массив пикселей копии изображения, максимальной площади *max\_s* присваивает ноль, проходит по пикселям копии и, когда находит пиксель старого цвета, записываем кортеж из координат левого верхнего и правого нижнего угла прямоугольника, найденных с помощью функции *find\_rect,* если площадь найденного прямоугольника больше *s\_max, s\_max* присваивает найденную площадь,записывает координаты в *coords\_max.* После обхода всех пикселей меняет цвет найденного прямоугольника и возвращает измененное изображение.
* *find\_rect(x, y, width, height, pixdata, old\_color)*. Значениям координат левого верхнего угла x0 и y0 присваивает width и height, нижнего правого x2, y2 приравнивает к нулю, в массив с текущими проверяемыми пикселями записывает переданные значения координат x и y. Пока массив не пустой, берёт из него последние координаты, удаляет их и работает с ними дальше: если они находятся в пределах изображения и цвет данного пикселя равен старому цвету, то координатам левого верхнего угла присваиваются минимальные найденные координаты, координатам правого нижнего – максимальные, в массив с текущими проверяемыми координатами добавляет все смежные координаты для следующей проверки и цвет проверенного пикселя делаем черным, чтобы избежать множественной проверки, возвращает кортеж с координатами левого верхнего угла и правого нижнего.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | user\_func(Image.new("RGB", (50, 50), (255, 0, 255)), 10, 20, 30, 40, 1, 2) |  |  |
|  | set\_black\_white(Image.new("RGB", (100, 100), (0, 255, 255)), 10, 20, 110, 80) |  | Координаты превышают размеры изображения |
|  | set\_black\_white(Image.new("RGB", (100, 100), (0, 255, 255)), 10, 20, 70, 80) |  |  |
|  | find\_rect\_and\_recolor(image, (100, 0, 100), (255, 250, 200)) |  |  |

## Выводы

Была изучена библиотека *PILLOW(PIL),* для каждой задачи были реализованы функции, работающие с объектами типа *<class 'PIL.Image.Image'>.*

# Приложение А Исходный код программы

Название файла: main.py

import PIL

import numpy as np

from PIL import Image, ImageDraw

# Задача 1

def user\_func(image, x0, y0, x1, y1, fill, width):

ImageDraw.Draw(image).line(((x0,y0),(x1,y1)), fill, width)

return image

# Задача 2

def check\_coords(image, x0, y0, x1, y1):

if 0 <= x0 < x1 <= image.size[0] and 0 <= y0 < y1 <= image.size[1]:

return True

return False

def set\_black\_white(image, x0, y0, x1, y1):

if check\_coords(image, x0, y0, x1, y1):

image.paste(image.crop((x0, y0, x1, y1)).convert("1"), (x0, y0))

return image

# Задача 3

def find\_rect\_and\_recolor(image, old\_color, new\_color):

pixdata = image.copy().load()

width, height, max\_s = image.size[0], image.size[1], 0

for x in range(width):

for y in range(height):

if pixdata[x, y] == old\_color:

coords = find\_rect(x, y, width, height, pixdata, old\_color)

s = (coords[2] - coords[0] + 1) \* (coords[3] - coords[1] + 1)

if s > max\_s:

max\_s = s

coords\_max = coords

res\_img = image.load()

for x in range(coords\_max[0], coords\_max[2] + 1):

for y in range(coords\_max[1], coords\_max[3] + 1):

res\_img[x, y] = new\_color

return image

def find\_rect(x, y, width, height, pixdata, old\_color):

x0, y0 = width, height

x2, y2 = 0, 0

curr = [(x, y)]

while len(curr)>0:

x1, y1 = curr.pop()

if (0 <= x1 < width and 0 <= y1 < height and pixdata[x1, y1] == old\_color):

x0 = min(x0, x1)

y0 = min(y0, y1)

x2 = max(x2, x1)

y2 = max(y2, y1)

curr +=[(x1-1,y1),(x1+1,y1),(x1,y1-1),(x1,y1+1)]

pixdata[x1, y1] = (0, 0, 0)

return (x0, y0, x2, y2)