**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 |  | Клюкин А.В. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Изучить работу библиотеки Pillow.

## 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 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

Было написано 4 функции

user\_func, которая в зависимости от параметров строила отрезок с помощью метода line

check\_coords, которая проверяла корректность координат и возвращала логическое выражение

set\_black\_white, которая на основе предыдущей функции либо преобразовывает изображение в ЧБ с помощью .convert, либо возвращает начальное изображение

find\_rect\_and\_recolor, которая получает на вход изображение и два цвета. В ней сначала создаются основные переменные и массив. picture = image.load() - получает массив, содержащий данные о пикселях. Затем записываются размеры полученного изображения. Создаются два кортежа, которые будут содержать координаты крайних точек нужного прямоугольника. Начинается цикл в котором перебираются все пиксели. В случае, если пиксель подходит под заданный изменяемый цвет, то создаются две новые переменные координат и приравниваются их к текущим, чтобы не потерять начало фигуры. Далее циклом идет просчет длины и высоты фигуры. Изменяя только одну координату, просматривается условие на выход за пределы координаты и удовлетворению цветового условия. После того, как длина и высота посчитана - можно посчитать площадь, как произведение длины на высоту. Сразу идет сравнение на максимальную площадь. Если это самая большая на момент проверки, то максимум обновляется и записываются координаты начала фигуры и конца в кортежи. После прохождения по всему изображению - перекрашивается прямоугольник по полученным координатам и возвращается изображение.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | user\_func(image, 1,1,100,100,”green”,5) | image | ОК |
|  | set\_black\_white(image, 10, 10, 100, 100) | image | ОК |
|  | find\_rect\_and\_recolor(image, “red”, “green”) | image | ОК |

## Выводы

Была написана программа, использующая основные функции для работы с изображениями с использованием библиотекой Pillow.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Klyukin\_Aleksandr\_lb2.py

import PIL

import numpy as np

import math

from PIL import Image, ImageDraw

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

draw = ImageDraw.Draw(image)

draw.line(((x0, y0), (x1, y1)), fill, width)

return image

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

width, height = image.size

if (x0 < x1 and y0 < y1) and (x0 >= 0 and y0 >= 0) and (x1 <= width and y1 <= height):

return True

else:

return False

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

check = check\_coords(image, x0, y0, x1, y1)

if check is True:

img = image.crop((x0, y0, x1, y1))

img = img.convert('1')

image.paste(img, (x0, y0))

return image

else:

return image

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

picture = image.load()

width, height = image.size

coords\_start = (0, 0)

coords\_end = (0, 0)

max\_area = 0

for x in range(width):

for y in range(height):

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

x\_new = x

y\_new = y

while x\_new < width and picture[x\_new, y] == old\_color:

x\_new += 1

while y\_new < height and picture[x, y\_new] == old\_color:

y\_new += 1

if (x\_new - x) \* (y\_new - y) > max\_area:

max\_area = (x\_new - x) \* (y\_new - y)

coords\_start = (x, y)

coords\_end = (x\_new-1, y\_new-1)

draw = ImageDraw.Draw(image)

draw.rectangle((coords\_start, coords\_end), new\_color)

return image