**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 |  | Бубякина Ю.В. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Получение навыков обработки изображений с помощью языка Python.

## Задание

Вариант 4

Предстоит решить 3 подзадачи, используя библиотеку Pillow (PIL). Для реализации требуемых функций студент должен использо-вать numpy и PIL. Аргумент image в функциях подразумевает объект ти-па <class 'PIL.Image.Image'>

1) Рисование отрезка. Отрезок определяется:

• координатами начала

• координатами конца

• цветом

• толщиной.

Необходимо реализовать функцию user\_func(), рисующую на картинке отре-зок

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 ле-вого 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 об-ласть изображения в черно-белый (используйте для конвертации пара-метр '1'). В этой функции должна вызываться функция проверки, и, ес-ли область некорректна, то должно быть возвращено исходное изоб-ражение без изменений. Примечание: поскольку черно-белый формат изображения (greyscale) является самостоятельным форматом, а не ва-риацией 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**

Была импортирована библиотека PIL.

Была реализована функция user\_func(image, x0, y0, x1, y1, fill, width), которая принимает на вход изображение(image), координаты начала отрезка(x0,y0), координаты конца отрезка(x1,y1), цвет(fill) и толщину(width). Для возможности рисовать на изображении в переменную drawing была записана функция ImageDraw.Draw(image). Далее к переменной drawing был применен метод drawing.line(((x0, y0, x1, y1)), fill, width), который рисует на изображении линию с заданными координатами, цветом и толщиной. Функция возвращает обработанное изображение.

В рамках 2 задачи были реализованы две функции: check\_coords() и set\_black\_white(). Функция check\_coords(image, x0, y0, x1, y1) принимает на вход изображение и координаты области, которую необходимо преобразовать в Ч/Б. Функция проверяет корректность заданных координат и возвращает True в случае успеха. Функция set\_black\_white(image, x0, y0, x1, y1) принимает на вход изображение и координаты области, которую необходимо преобразовать в Ч/Б. В случае, если вывод функции check\_coords() оказался False, функция set\_black\_white() вернет исходное изображение. Если же проверяющая функция вывела True, то в переменную cut передается необходимая, вырезанная из исходного изображения, область. Далее в переменную cut\_final с помощью метода cut.convert(‘1’) передается вырезанная ранее область в формате Ч/Б. Последним шагом является image.paste(cut\_final, (x0, y0, x1, y1)), который вставляет в исходное изображение преобразованную область на место старой по тем же координатам. Функция возвращает обработанное изображение.

Была реализована функция find\_rect\_and\_recolor(image, old\_color, new\_color), принимающая на вход изображение(image), цвет нужного прямоугольника(old\_color) и цвет, на который его необходимо заменить(new\_color) в RGB формате. Чтобы работать с изображением функция использует метод image.load() в переменной pixel\_obj. В переменные width и height заносятся размеры изображения с помощью image.size. В переменных max\_area и max\_coords будут храниться максимальная площадь прямоугольника и координаты левой верхней и правой нижней точек этого прямоугольника. Далее с помощью цикла for перебирается каждый пиксель и при совпадении его цвета с old\_color с помощью цикла while начинается перебор пикселей, пока они нужного цвета, и формирование координат найденного прямоугольника. Когда найденный прямоугольник полностью «просканирован» в переменную area записывается его площадь по координатам x1 и y1. Далее эта площадь проверяется на максимальную и если на данный момент она наибольшая из найденных, то она записывается в переменную max\_area, а в переменную max\_coords записываются координаты левой верхней и правой нижней точек этого прямоугольника. Последний шаг – это двойной цикл for, который перебирает каждый пиксель на исходном изображении и при совпадении очередного пикселя с областью максимального прямоугольника с помощью метода image.putpixel((x, y), new\_color) он перекрашивается в нужный цвет. Функция возвращает обработанное изображение.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | user\_func(Image.new(“RGB”, (251,251), “red”), 2,2,78,78,’yellow’, 5) | image | - |
|  | check\_coords(Image.new(“RGB”, (300,300), “lime”), 21,21,24,24) | True | - |
|  | set\_black\_white(Image.new(“RGB”, (312,312), “red”), 34,34,123,123) | image | - |
|  | find\_rect\_and\_recolor(Image.new(“RGB”, (450,450), “black”), (255,255,255), (123,23,90)) | image | - |

## Выводы

Освоены методы работы с библиотекой Pillow. Получены навыки работы с изображениями, рисования линий, смены формата изображения и изменение отдельных объектов изображения.

# Приложение А Исходный код программы

Название файла: Bubyakina\_Yuliya\_lb2.py

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):

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

return True

return False

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

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

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

cut\_final = cut.convert('1')

image.paste(cut\_final,(x0,y0,x1,y1))

return image

return image

# Задача 3

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

old\_color=tuple(old\_color)

new\_color = tuple(new\_color)

pixel\_obj = image.load()

width,height = image.size

max\_area=0

max\_coords=()

for x in range(width):

for y in range(height):

if pixel\_obj[x,y]==old\_color:

area=1

x1=1

y1=1

while x1+x<width and pixel\_obj[x1+x,y]==old\_color:

x1+=1

while y1+y<height and pixel\_obj[x,y1+y]==old\_color:

y1+=1

area=y1\*x1

if area>max\_area:

max\_area=area

max\_coords=(x,y,x+x1-1,y+y1-1)

for x in range (max\_coords[0],max\_coords[2]+1):

for y in range(max\_coords[1], max\_coords[3] + 1):

image.putpixel((x,y),new\_color)

return image