**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. 3344 |  | Коняева М.В. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является освоение функций библиотеки Pillow(PIL), которая позволяет работать с изображениями.

## Задание

Вариант 1. 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'>*

Задача 1. Необходимо написать функцию *triangle()*, которая рисует на изображении треугольник.

Задача 2. Необходимо написать функцию *change\_color()*, которая заменяет наиболее часто встречаемый цвет на переданный. Функция должна найти в изображении самый частый цвет и заменить его на переданный, затем вернуть новое изображение (исходное изображение не должно меняться).

Задача 3. Необходимо написать функцию *collage()*. Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и вернуть его (новое изображение).

## Выполнение работы

Подключается библиотека PIL для дальнейшей работы с изображениями.

Первая функция *triangle()*, которая принимает на вход изображение, координаты вершин треугольника, толщину линии, цвет линии, цвет, которым залит треугольник. Для того чтобы появилась возможность рисовать фигуры на изображении надо перейти к специальному объекту, вызвав *ImageDraw.Draw(img)*. Для удобства запишем шесть чисел координат в список кортежей пар. Рисуем треугольник с помощью функции *drawing.polygon()* в зависимости от параметра *fill\_color*. Функция возвращает полученное изображение.

Вторая функция *change\_color()* принимает на вход изображение и цвет. Записываем в переменную *w* и *h*, высоту и ширину полученного изображения. Создадим словарь, где ключ - это цвет, который есть на изображении, а его значение - количество пикселей с таким цветом. Для этого проходимся по каждому пикселю с помощью двух циклов *for*. Получаем цвет пикселя с помощью функции *img.getpixel()*, если такого цвета нет в ключах, то добавляем ключ и увеличиваем счетчик на один. С помощью функции *max()* находим максимальное значение и его ключ. Меняем часто встречающийся цвет на нужный с помощью функции *image.putpixel().*

Третья функция *def collage()* получает на вход изображение, количество изображений по "оси" Y (N), количество изображений по "оси" X (M). Узнаем размеры коллажа, умножив исходную высоту и ширину на N, M. С помощью двойного цикла *for* вставляем исходное изображение на нужные координаты коллажа. Возвращаем полученный коллаж.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | triangle (img, 10, 10, 40, 40, 100, 100, 3, (255,0,0), (0,255,0) | img | Данные обработаны корректно |
|  | change\_color(img, (255,0,0) | img | Данные обработаны корректно |
|  | collage(img, 3, 3) | img | Данные обработаны корректно |

## Выводы

Были изучены основные функции библиотеки PIL, которые позволяют работать с изображениями и его параметрами. Были использованы такие функции, как *img.polygon()*, которая позволяет рисовать геометрические фигуры, *img.putpixel()* заменяет пиксель изображения и другие. Также были написаны собственные функции, которые позволяют рисовать треугольник по заданным параметрам, заменять нужный цвет и создавать коллаж.

# Приложение А Исходный код программы

Название файла: lb2.py

import PIL

from PIL import Image, ImageDraw

# Задача 1

def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill\_color):

drawing = ImageDraw.Draw(img)

coordinates = [(x0,y0),(x1,y1),(x2,y2)]

if fill\_color is None:

drawing.polygon(coordinates, outline=tuple(color), width=thickness)

else:

drawing.polygon(coordinates, fill=tuple(fill\_color),outline= tuple(color), width=thickness)

return img

# Задача 2

def change\_color(img, color):

image = img

arr = {}

width, height = img.size

for x in range(width):

for y in range(height):

pixel\_color = img.getpixel((x, y))

if pixel\_color not in arr.keys():

arr.update({pixel\_color:0})

else:

arr[pixel\_color]+=1

m = max(arr.values())

keys = [key for key, value in arr.items() if value == m]

for x in range(width):

for y in range(height):

if (image.getpixel((x, y)) == keys[0]):

image.putpixel((x, y), tuple(color))

return image

# Задача 3

def collage(img, N, M):

w = img.size[0] \* M

h = img.size[1] \* N

collage = Image.new("RGB",(w,h))

for x in range(M):

for y in range(N):

collage.paste(img,(img.size[0]\*x,img.size[1]\*y))

return collage