**МИНОБРНАУКИ россИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Информатика»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Гусева Е.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Освоение основ архитектуры компьютера и обработки изображений. Работа с модулем Pillow.

## 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'>

1) Рисование треугольника

Необходимо написать функцию triangle(), которая рисует на изображении треугольник

Функция triangle() принимает на вход:

Изображение (img)

Координаты вершин (x0,y0,x1,y1,x2,y2)

Толщину линий (thickness)

Цвет линий (color) - представляет собой список (list) из 3-х целых чисел

Цвет, которым залит (fill\_color - если значение None, значит треугольник не залит) - представляет собой список (list) из 3-х целых чисел

Функция должна вернуть исходное обработанное изображение.

2) Замена наиболее часто встречаемого цвета.

Необходимо написать функцию change\_color(), которая заменяет наиболее часто встречаемый цвет на переданный.

Функция change\_color() принимает на вход:

Изображение (img)

Цвет (color - представляет собой список из трех целых чисел)

Функция должна найти в изображении самый частый цвет и заменить его на переданный, затем вернуть новое изображение (исходное изображение не должно меняться).

3) Коллаж

Необходимо написать функцию collage().

Функция collage() принимает на вход:

Изображение (img)

Количество изображений по "оси" Y (N - натуральное)

Количество изображений по "оси" X (M - натуральное)

Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и вернуть его (новое изображение).

При необходимости можно писать дополнительные функции.

## Выполнение работы

Были импортированы библиотеки *PIL*, *numpy*. Для выполнения первой задачи была написана функция def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill\_color). Создана переменная drawing, в которую записываем возможность редактирования изображения. Далее идет проверка есть ли заливка у фигуры, в зависимости от этого фигура будет залита или нет. Фигура задается с помощью встроенной функции polygon. Функция возвращает измененное изображение.

Для выполнения второй задачи была написана функция def change\_color(img, color). Значения высоты и ширины изображения считываются через img.size, изображение копируется, создается словарь pixs\_colors. Далее с помощью двух циклов, считываются пиксели с изображения, проверяется условие, если цвет пикселя еще не встречался, то в ключи словаря записывается этот цвет и начальное значение 1, иначе увеличивается значение количества раз встреченных чисел. Далее словарь сортируется по величине значения встреченных пикселей одного цвета. В переменную color\_old записывается самый часто встречаемый в изображении цвет. Далее с помощью двойного цикла, в переменную pix\_color считывается значения цвета текущего пикселя, условием проверяется, что если цвет считанного пикселя является самым часто встречаемым, то цвет этого пикселя меняется на тот, что подается на вход функции. Функция возвращает измененное изображение.

Для выполнения третьей задачи была написана функция def collage(img, N, M). Значения высоты и ширины изображения считываются через img.size, по условию коллаж в N раз выше и в M раз шире, чем изначальное изображение. Создается коллаж, размер коллажа задается полученными перед этим collage\_width и collage\_height. Далее циклом в коллаж вставляются все изображния. Функция возвращает коллаж.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | triangle(img.size = (90,51), 74, 3, 59, 11, 65, 44, 6, ), color = [265, 119, 12], None) | img | - |
|  | change\_color(img.size = (90,51), color = [54, 54, 251]) | img | - |
|  | collage(img.size = (90,51), 6, 5) | img | - |

## Выводы

Были освоены основы архитектуры компьютера и обработки изображений, проведена работа с модулем Pillow.

# Приложение А Исходный код программы

Название файла: lb2.py

import PIL

from PIL import Image, ImageDraw

import numpy as np

# Задача 1

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

drawing = ImageDraw.Draw(img)

if fill\_color!=None:

drawing.polygon([(x0, y0), (x1, y1), (x2, y2)], outline=tuple(color), fill=tuple(fill\_color), width=thickness)

else:

drawing.polygon([(x0, y0), (x1, y1), (x2, y2)], outline=tuple(color), width=thickness)

return img

# Задача 2

def change\_color(img, color):

width, height = img.size

new\_image = img.copy()

pixs\_colors={}

for i in range(width):

for j in range(height):

pixs\_color = img.getpixel((i, j))

if pixs\_color in pixs\_colors:

pixs\_colors[pixs\_color] += 1

else:

pixs\_colors.update({pixs\_color: 1})

pixs\_colors = dict(sorted(pixs\_colors.items(), key=lambda item: item[1]))

color\_old = list(pixs\_colors.keys())[-1]

for i in range(width):

for j in range(height):

pix\_color = img.getpixel((i, j))

if pix\_color == color\_old:

new\_image.putpixel((i, j), tuple(color))

return new\_image

# Задача 3

def collage(img, N, M):

width, height = img.size

collage\_width = width \* M

collage\_height = height \* N

collage = Image.new('RGB', (collage\_width, collage\_height))

for i in range(N):

for j in range(M):

x = j \* width

y = i \* height

collage.paste(img, (x, y))

return collage