**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:** В**ведение в архитектуру компьютера**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Пименов П.В. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Изучить принцип работы с библиотекой Pillow, создать программу, выполняющую различные преобразования над изображениями в зависимости от задачи.

## Задание

*Вариант 1*. Задача 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.

Задача 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).

Задача 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).

## Выполнение работы

Задача 1. Для того, чтобы нарисовать треугольник, функция triangle() сперва создает объект класса ImageDraw, с помощью которого можно нарисовать геометрические фигуры на картинке. Для рисования треугольника используется функция polygon(), позволяющая нарисовать многоугольник с введенными параметрами. Функция возвращает измененное изображение.

Задача 2. Для того, чтобы заменить наиболее часто встречаемый цвет, сперва необходимо его определить. Создается словарь colors, ключами которого являются цвета (кортежи чисел), а значениями — количество пикселей этого цвета. С помощью двух вложенных циклов функция перебирает все пиксели и заносит в словарь их цвета. В переменную target с помощью функции max записывается наиболее часто встречаемый цвет. Далее функция создает новое изображение с размерами исходного и второй раз перебирает все пиксели исходного по следующему принципу: если цвет конкретного пикселя является наиболее часто встречаемым, то соответствующий пиксель нового изображения красится в цвет, переданный как аргумент функции (тот цвет, на который надо заменить), в противном случае — соответствующий пиксель красится в цвет исходного пикселя. Функция возвращает новое изображение.

Задача 3. Для того, чтобы создать коллаж, функция сначала создает изображение размера (img\_width \* M, img\_height \* N), где img\_width и img\_height — ширина и высота исходного изображения. Далее, с помощью двойного цикла перебираются «индексы» картинок в коллаже, а сами картинки вставляются по соответствующим индексам с помощью функции paste(). Функция возвращает новое изображение.

Разработанный программный код см. в приложении А.

## Выводы

Были изучены принципы работы с библиотекой Pillow, создана программа, выполняющая простые преобразования над изображениями.

# Приложение А Исходный код программы

Название файла: main.py

from PIL import Image, ImageDraw

# Задача 1

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

drawing = ImageDraw.Draw(img)

if fill\_color is not None:

fill\_color = tuple(fill\_color)

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

return img

# Задача 2

def change\_color(img, color):

colors = {}

width, height = img.size

for i in range(width):

for j in range(height):

pixel = img.getpixel((i, j))

colors.update({pixel: colors.get(pixel, 0) + 1})

target = max(colors, key=colors.get)

changed = Image.new('RGB', img.size)

for x in range(width):

for y in range(height):

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

if pixel == target:

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

else:

changed.putpixel((x, y), pixel)

return changed

# Задача 3

def collage(img, N, M): # M - по X, N - по Y

img\_width = img.size[0]

img\_height = img.size[1]

collage = Image.new('RGB', (img\_width \* M, img\_height \* N))

for i in range(M):

for j in range(N):

collage.paste(img, (img\_width \* i, img\_height \* j))

return collage

# Приложение Б Тестирование

Тестирование фунцкии triangle():

Входные данные:

* img = Image.new(‘RGB’, (300, 300))
* x0 = 50
* y0 = 50
* x1 = 100
* y1 = 50
* x2 = 75
* y2 = 100
* thickness = 5
* color = [0, 255, 0]
* fill\_color = [255, 0, 0]

Выходные данные:

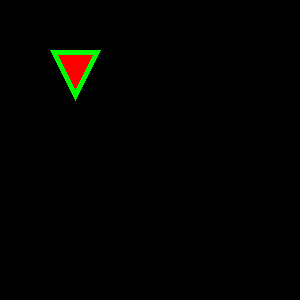


Рисунок 1 — Результат работы функции triangle()

Комментарии:

Функция triangle() работает корректно.

Тестирование функции change\_color():

Входные данные:

* img =

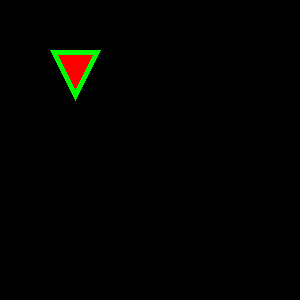


Рисунок 2 — Входные данные для функции change\_color()

* color = [0, 0, 255]

Выходные данные:

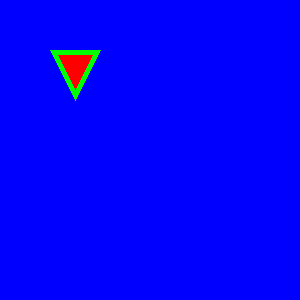


Рисунок 3 — Результат работы функции change\_color()

Комментарии:

Функция change\_color() работает корректно.

Тестирование функции collage()

Входные данные:

* img =

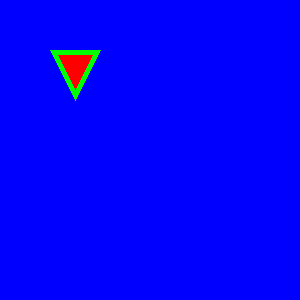
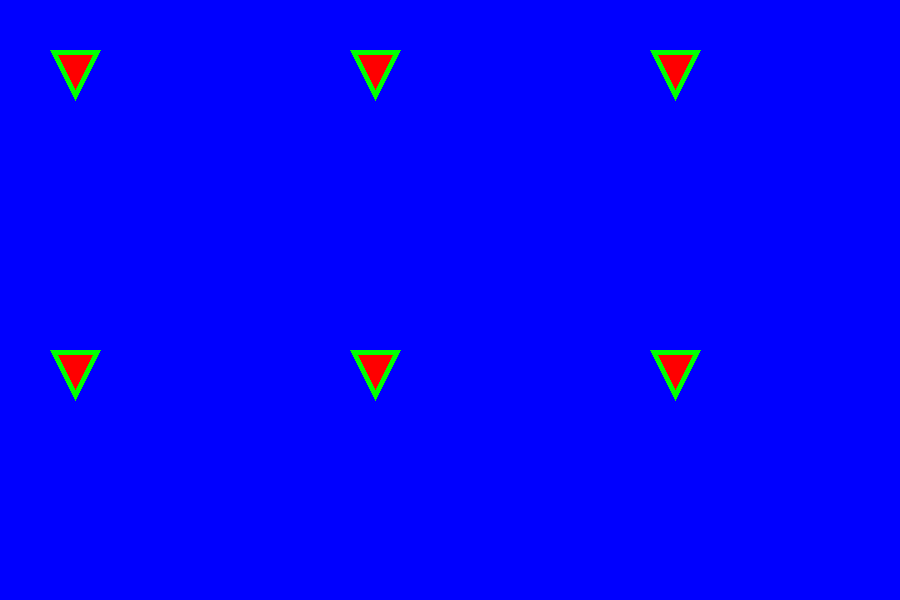


Рисунок 4 — Входные данные для функции collage()

* N = 2
* M = 3

Выходные данные:

Рисунок 5 — Результат работы функции collage()

Комментарии:

Функция collage() работает корректно.