МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Введение в архитектуру компьютера

Студентка гр. 3342 _____ Антипина В.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Получить базовые знания об архитектуре компьютера, изучить модуль Pillow.

Задание

Вариант 1

Предстоит решить 3 подзадачи, используя библиотеку Pillow (PIL). Для реализации требуемых функций студент должен использовать numpy и PILL. 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.

Основные теоретические положения

В библиотеку PIL встроены различные вспомогательные классы (модули):

Image - основной класс для хранения изображения и работы с ним:

- Вывод изображения на экран Image.show()

  Для вывода изображения на экран используется метод Image.show().

- Создание изображения Image.new(...)

Для создания изображения используется метод Image.new(mode, size, color), который принимает на вход: mode - режим работы с изображением, size - это кортеж из двух элементов (width, height), где первый width - ширина изображения (в пикселях), второй height - высота изображения (в пикселях), color -- цвет изображения. Так как в работе используются RGB-изображения, то для представления цвета используются три компоненты: R (red, красный), G (green, зеленый) и B (blue, голубой). Значение каждой компоненты изменяется от 0 до 255. color может быть представлен в двух вариантах: кортеж из трех целых чисел или строка. Например, кортеж (255, 0, 0) обозначает красный цвет, строкой этот же цвет можно передать в функцию как ‘red’.

- Размеры изображения Image.size

Как было обозначено выше, у изображения при создании можно указать размер - кортеж из двух элементов: ширины и высоты. Так, у ранее созданного изображения можно узнать размер, используя поле Image.size, которое возвращает кортеж (width, height).

- Отрисовка одного изображения на другое - Image.paste(other_image, coordinates)

Часто бывает так, что надо скомбинировать 2 изображения, расположив одно на другом. Для этого может помочь специальный метод Image.paste(other_image, coordinates), который принимает на вход 2 аргумента:

other_image - другое изображение класса Image, coordinates - кортеж из двух координат x, y первого изображения.

- Отражение изображения Image.transpose(direction)

В классе Image доступно несколько действий, которые можно совершать над изображениями. Например, можно преобразовать изображение, повернув его или отразив по одной из центральных осей (вертикальной или горизонтальной). Для такого преобразования понадобится Image.transpose(method), который принимает аргумент method - метод преобразования.

Отражение изображения относительно центральной вертикальной оси - Image.Transpose.FLIP_LEFT_RIGHT, а относительно центральной горизонтальной оси изображения - Image.Transpose.FLIP_TOP_BOTTOM.

- Обрезка изображения Image.crop(box)

Если нужно выделить прямоугольную часть изображения, которая задана двумя координатами: левый верхний угол (x1, y1) и правый нижний угол (x2, y2), используется метод Image.crop(box), который принимает на вход аргумент box - кортеж из 4-х значений: (x1, y1, x2, y2). Данный метод возвращает вырезанный прямоугольник.

- Конвертация изображения Image.convert(mode)

Позволяет конвертировать изображения из RGB-формата в другие, например, черно-белый (greyscale) и наоборот. Метод возвращает измененную копию изображения.

Рисование на изображении с помощью ImageDraw:

Для того, чтобы нарисовать что-то на существующем изображении, потребуется воспользоваться классом ImageDraw.

- Объект для рисования ImageDraw.Draw(source_image)

Для того, чтобы появилась возможность рисовать фигуры на изображении, надо перейти к специальному объекту, вызвав

ImageDraw.Draw(source_image), где source_image - Image, на котором мы хотим рисовать.

- Рисование прямоугольника ImageDraw.rectangle(...)

Например, чтобы нарисовать прямоугольник на изображении, у объекта ImageDraw нужно вызвать соответствующий метод ImageDraw.rectangle(coordinates, fill=None, outline=None, width=1), который принимает несколько аргументов: coordinates - координаты прямоугольника (левый верхний и правый нижний угол) - кортеж из 4-х элементов (x1, y1, x2, y2), или кортеж пар ((x1, y1), (x2, y2)), fill - цвет заливки, outline - цвет контура, width - ширина линий контура (по умолчанию равна 1).

- Рисование круга ImageDraw.ellipse(...)

Окружность, круг и эллипс можно нарисовать с помощью метода ImageDraw.ellipse(frame_coordinates, fill=None, outline=None, width=1), который принимает на вход аргументы: frame_coordinates -- координаты ((x1, y1), (x2, y2)) левого верхнего и правого нижнего углов прямоугольника, в который вписан эллипс, fill, o utline, width -- такие же, как и в ImageDraw.rectagle(..)

- Рисование линии ImageDraw.line(...)

Для того, чтобы нарисовать прямую или ломаную линию, можно воспользоваться методом ImageDraw.line(coordinates, fill=None, width=0, joint=None), который принимает на вход аргументы: coordinates - кортеж из пар координат ((x1, y1), (x2, y2), ...), которые нужно соединить, fill, width - такие же, как и в ImageDraw.rectagle(..), joint -- тип соединения ('curve' если нужно закругленного соединения, None -- если не нужно закругленное соединение).

- Рисование полигона ImageDraw.polygon(...)

Если фигура задается большим числом отрезков, нежели чем прямоугольник, то на помощь придет метод ImageDraw.polygon(coordinates, fill=None, outline=None, width=1), который принимает на вход аргументы:

coordinates - координаты точек, которые необходимо соединить линиями; обычно задается как кортеж кортежей-пар ((x1, y1), (x2, y2), ...), fill, outline, width -- такие же, как и в ImageDraw.rectagle(..)

Выполнение работы

Была написана функция triangle, которая принимает на вход изображение, целые числа - координаты вершин треугольника x0, y0, x1, y1, x2, y2, толщину линий, цвет линий и цвет заливки, если последняя есть, и возвращает изменённое изображение. Так как цвет заливки и цвет линий подаются функции в виде списка, для того, чтобы можно было использовать метод ImageDraw.polygon(), необходимо преобразовать списки в кортежи. Это было реализовано с помощью функции tuple() (кортежи были помещены в переменные use_color для цвета линий и to_fill для заливки фигуры). Так как аргумент fill_color может содержать «None», а не список значения цвета, была использована конструкция if-else, в которой переменной to_fill присваивался преобразованный в кортеж список, если fill_color не «None», или значение fill_color в противном случае. С помощью метода Draw была получена возможность «рисовать» на полученном на вход изображении. Для того, чтобы построить треугольник, был использован метод polygon(), на вход которому был подан список координат точек, которые нужно соединить, цвет заливки фигуры или указание на то, что её нет, цвет линий и их толщина. Функция возвращает исходное изображение, к которому был применён метод Draw.

Также была написана функция change_color(), которая получает на вход всего два аргумента: изображение и цвет, в который нужно «перекрасить» пиксели, окрашенный в исходном изображении в наиболее часто встречающийся цвет. Чтобы функция не вносила изменений в исходное изображение, с помощью метода Image.copy() была создана его копия, помещенная в переменную img2, к которой после был применён метод Draw(). С помощью метода Image.load() был получен доступ к пикселям исходного изображения. Преобразованный в кортеж список color был помещён в переменную replacement_color. Был создан пустой словарь hwmch, в который будут помещены все используемые цвета и количество пикселей этого цвета.

С помощью двух вложенных циклов осуществляется добавление цветов и подсчёт их количества в изображении. Для определения значения пикселя был использован метод getpixel(). Если цвет, полученный таким образом, уже есть в словаре, значение, хранящееся по этому ключу, увеличивается на единицу. В противном случае создаётся новая пара «ключ — значение», где ключ — это полученный цвет, а значение — 1. С помощью функции max и метода velues() в переменную maxx было записано максимальное значение из получившегося словаря. Затем в цикле for был описан поиск соответствущего этому значению ключа. Далее в двух вложенных циклах реализовано сравнение значения каждого пикселя img2 и наиболее часто встречающегося цвета. Если они совпадали, с помощью метода point() пиксель менял свой цвет на поданный на вход. Функция возвращает img2.

Была реализована функция collage(), которой на вход подаётся изображение и целые значения N, M — количество изображений по вертикали и диагонали в коллаже. С помощью метода np.copy() была создана копия исходного изображения в переменную img_copied типа nparray. Combined было присвоено значение img_copied. В цикле for от 1 до M с помощью метода np.hstack() к combined дописывались справа изображения M-1 раз (так как одно изображение из M необходимых уже есть). К combined добавляется img_copied, потому что последняя не изменяется в процессе выполнения функции, то есть так мы «дублируем» именно исходное изображение, а не изменяющийся массив. Далее в переменную temp записывается полученная матрица, чтобы теперь она была единицей копирования (так добавляется строка целиком). В цикле for от 0 до N-1 с помощью метода np.vstack() к матрице combined дописываются строки temp. Далее с помощью метода Image.fromarray() тип combined меняется на Image. Функция возвращает background, в которую была записана преобразованная матрица combined.

Тестирование

Программа прошла все тесты на e.moevm.info. Результаты представлены на рис.1

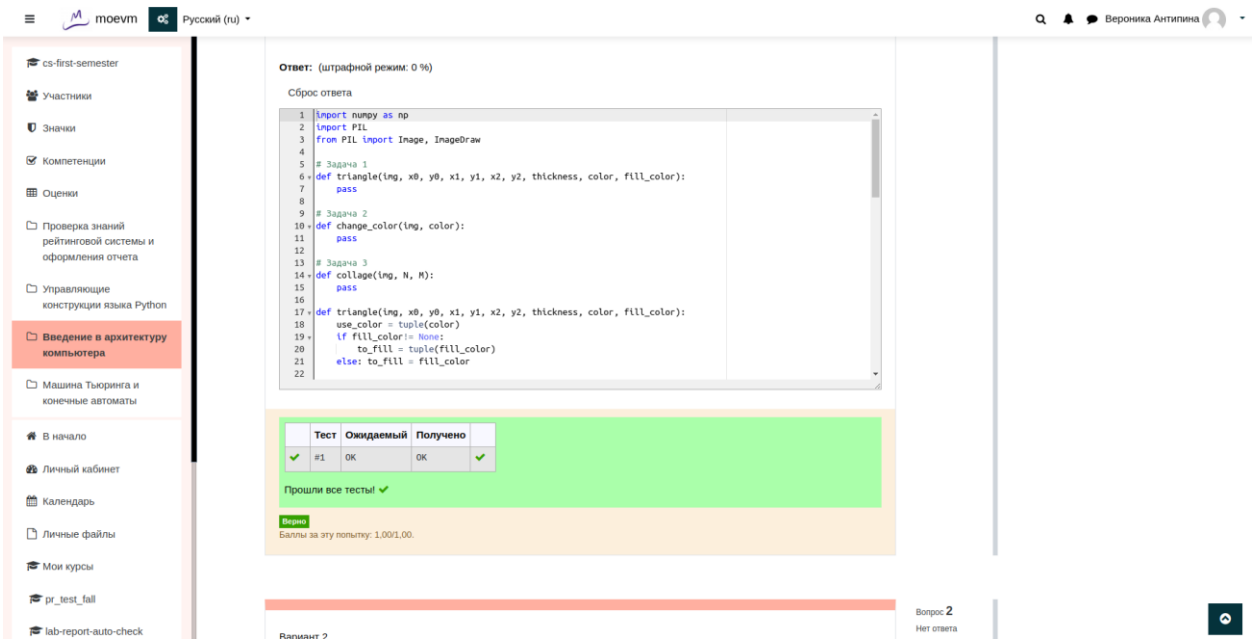

Рисунок 1 — Результаты тестирования на e.moevm.info

Выводы

Были получены базовые знания об архитектуре компьютера, изучен модуль Pillow.

Были написаны функции, которые строят треугольник на изображении, заменяют наиболее часто встречающийся цвет в изображении на заданный, создают коллаж из повторений поданного на вход изображения. В работе были использованы основные методы numpy и Pillow.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Antipina_Veronika_lb2.py

```
import numpy as np
import PIL
from PIL import Image, ImageDraw

def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color,
fill_color):
    use_color = tuple(color)
    if fill_color!= None:
        to_fill = tuple(fill_color)
    else: to_fill = fill_color

    draw = ImageDraw.Draw(img)

    draw.polygon([x0,y0,x1,y1,x2,y2],to_fill,use_color,thickness)

    return img

def change_color(img, color):
    img2 = img.copy()
    draw = ImageDraw.Draw(img2)
    pixels = img.load()
    replacement_color = tuple(color)

    hwmch = {}

    for i in range(img.size[0]):
        for j in range(img.size[1]):
            current_color = img.getpixel((i,j))
            if current_color in hwmch:
                hwmch[current_color]+=1
            else:
                hwmch[current_color]=1

    maxx = max(hwmch.values())
    for keys in hwmch:
        if hwmch[keys]==maxx:most_used_color = keys

    for x in range(img.size[0]):
        for y in range(img.size[1]):
            if img2.getpixel((x,y))==most_used_color:
                draw.point((x,y),fill = replacement_color)

    return img2

def collage(img, N, M):
    img_copied = np.copy(img)
    combined = img_copied
    for i in range(1,M):
        combined = np.hstack((combined,img_copied))
```

```
temp = combined
for j in range(N-1):
    combined = np.vstack((combined,temp))

background = Image.fromarray(combined)

return background
```