* [ОБЗОР КУРСА](https://lyceum.yandex.ru/courses/123/groups/568)

[Урок Библиотеки №3](https://lyceum.yandex.ru/courses/123/groups/568/lessons/666)

**Библиотеки Python. Часть № 3 (графика + звук)**

**План урока**

1

[Фильтры](https://lyceum.yandex.ru/courses/123/groups/568/lessons/666/materials/1461#1)

2

[Модуль wave](https://lyceum.yandex.ru/courses/123/groups/568/lessons/666/materials/1461#2)

**Аннотация**

*В этом уроке мы продолжим изучение библиотеки PyPIL и научимся создавать собственные фильтры для растровых изображений. Также мы познакомимся с модулем wave для обработки звука.*

**1. Фильтры**

Когда-то Instagram превратился из заурядной социальной сети в очень популярный феномен именно из-за удачной реализации встроенных фильтров. Фильтры можно было накладывать на фото — фотографии после этого обычно становились красивыми и похожими на профессиональные.

Фильтры очень широко применяются в киноиндустрии. Сравните цветовую гамму молодежных комедий или современных блокбастеров с, например, классическим «Шерлоком Холмсом».

Иначе говоря, фильтры очень востребованы — начиная от самых простых и заканчивая работами с привлечением искусственного интеллекта, например, в проекте [Prisma](http://prisma-ai.com/).

**Фильтр**

Фильтр можно воспринимать как любое преобразование заданного изображения. Чтобы добиться наилучшего эффекта, их можно накладывать последовательно.

В библиотеке PIL реализовано много встроенных фильтров и инструментов (вырезание, изменение размеров, и т. д.). Фактически это такой программируемый мини-Photoshop, но мы попытаемся поработать с фильтрами самостоятельно, чтобы поучиться восприятию цветовой палитры и алгоритмизации.

Для начала попробуем превратить изображение в чёрно-белое.

Чёрно-белое изображение содержит только информацию о яркости, но не о цветах. У таких изображений все три компоненты имеют одинаковое значение, поэтому мы можем просто «размазать» суммарную яркость пикселя поровну по трём компонентам.

**for** i **in** range(x):

**for** j **in** range(y):

r, g, b = pixels[i, j]

bw = (r + g + b) // 3

pixels[i, j] = bw, bw, bw



Можно сказать, что мы «слили» содержимое контейнеров R, G, B в одну ёмкость, а затем разлили обратно — но уже поровну в каждый контейнер. Суммарная яркость пикселя осталась прежней, но информация о цвете не сохранилась. Фотография же стала более «задумчивой».

Попробуем поменять местами зелёный и синий каналы:

**for** i **in** range(x):

**for** j **in** range(y):

r, g, b = pixels[i, j]

pixels[i, j] = r, b, g



Давайте подумаем над тем, как получить негатив. Если на позитиве белое изображение (255), то на негативе должно быть черное (0) и наоборот. То есть для значения х негативом будет 255 — x.

**for** i **in** range(x):

**for** j **in** range(y):

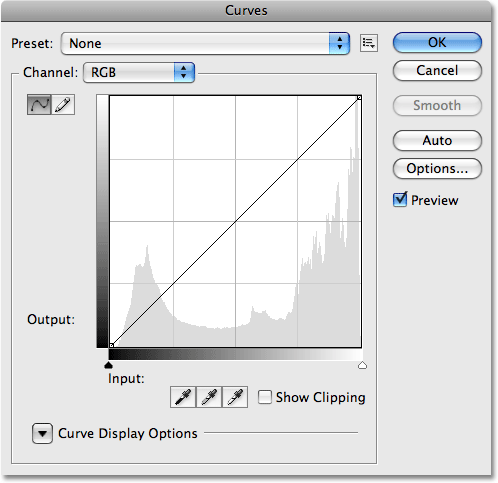
r, g, b = pixels[i, j]

pixels[i, j] = 255 - r, 255 - g, 255 - b



Как видим, на негативе можно рассмотреть некоторые детали, которые не видны на позитиве. Во многих редакторах (включая Photoshop) есть инструмент Кривые (Curves).

Он позволяет задать функцию, меняющую яркость всего пикселя или отдельной компоненты в зависимости от исходной яркости. Изначально эта функция представляет собой прямую y=x.



В Python можно написать функцию, которая работает как инструмент Curves. Например, мы можем высветлить темные участки в изображении, не трогая светлые. Это очень частая операция: например, когда на снимке светлое небо и очень темное здание, потому что фотоаппарат подстроился под яркость неба.

**Высветление**

«Высветлить» означает увеличить значения всех цветовых компонентов на какой-то коэффициент. Важно помнить, что эти значения не могут быть больше 255.

**def** curve(pixel):

r, g, b = pixel

brightness = r + g + b

**if** brightness < 60:

k = 60 / brightness

**return** min(255, int(r \* k \*\* 2)), min(255, int(g \* k \*\* 2)), \  
 min(255, int(b \* k \*\* 2))

**else**:

**return** r, g, b

**for** i **in** range(x):

**for** j **in** range(y):

pixels[i, j] = curve(pixels[i, j])

Результат:



**2. Модуль wave**

Фильтры можно накладывать не только на изображение, но и на звуковые файлы. Для манипуляции с «сырыми», необработанными аудиоданными предназначен модуль wave. В ОС Windows такие данные хранятся в файлах с расширением **.wav**.

**Аудиоданные**

Сырые аудиоданные представляет собой зависимость амплитуды звукового сигнала от времени. Посмотрите на график этой зависимости. Вы часто видели его в фильмах, репортажах журналистов и т. д. Вдоль оси абсцисс откладывается время, вдоль оси ординат — амплитуда (интенсивность, громкость) звукового сигнала.



Этот график строится по точкам, причём вместо пар (t, y) хранятся только значения y, а ось абсцисс задана частотой дискретизации (количеством отсчетов в секунду) — как правило, она составляет 44 100 Гц. Такой частоты достаточно, чтобы человек мог прослушать оцифрованный звук и не заметить его отличия от реального.

**Представление аудиофайла**

Получается, что представление звукового файла для программиста очень простое. Это список целых чисел (положительных и отрицательных) — значения амплитуды звукового сигнала.

В следующем примере мы рассмотрим основные этапы работы со звуковым файлом. Наша программа будет разворачивать звуковой файл в обратную сторону, то есть проигрывать музыкальное произведение задом наперед.

Пример файла можно взять [тут](https://yastatic.net/s3/lyceum/content/resources/in.wav).

**import** wave

**import** struct

source = wave.open("in.wav", mode="rb")

dest = wave.open("out.wav", mode="wb")

dest.setparams(source.getparams())

*# найдем количество фреймов*

frames\_count = source.getnframes()

data = struct.unpack("<" + str(frames\_count) + "h",  
 source.readframes(frames\_count))

*# собственно, основная строка программы - переворот списка*

newdata = data[::-1]

newframes = struct.pack("<" + str(len(newdata)) + "h", \*newdata)

*# записываем содержимое в преобразованный файл.*

dest.writeframes(newframes)

source.close()

dest.close()

**Важно**

Мы открываем два файла: исходный — **source** и формируемый — **dest**. Одно значение амплитуды в терминах библиотеки **wave** называется фреймом.

Наибольшую трудность здесь представляют строки с использованием встроенного модуля struct. Пока договоримся, что функции этого модуля могут на лету распаковывать и запаковывать данные разной природы. Интересно, что палиндромы в аудиофайлах сохраняют свою «палиндромность» (например, «А роза упала на лапу Азора»).

Что ещё можно сделать?

* Если мы уберём, например, каждый второй фрейм, то ускорим воспроизведение вдвое. При этом частота тоже вырастет в два раза («голос Чипа и Дейла»). Кстати, на телевидении и радио часто ускоряют видео и аудио на 5-10%: это незаметно для уха, но позволяет разместить больше рекламы в эфире.
* Если мы увеличим все фреймы в какое-то количество раз, то сделаем произведение громче, а если уменьшим — тише.
* Копируя каждый фрейм 2 раза, мы замедлим воспроизведение и понизим частоту.

[Справка](https://yandex.ru/support/lyceum-students)

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках проекта «Яндекс.Лицей», принадлежат АНО ДПО «ШАД». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «ШАД».

© 2018 – 2020  ООО «[Яндекс](https://yandex.ru/)»

Чаты