Класс SoundPool

[SoundPool.Builder](http://developer.alexanderklimov.ru/android/theory/soundpool.php#builder) (API 21)

Основное отличие класса **SoundPool** от класса **MediaPlayer** - поддержка одновременного проигрывания нескольких аудиопотоков, что широко используется в играх.

Обычно **SoundPool** используют для проигрывания маленьких аудиоклипов. **SoundPool** может играть несколько звуков одновременно. Звуки также можно повторять (зациклить). В этом случае для остановки проигрывания нужно явно вызывать метод остановки. Размер файла, который может проиграть **SoundPool** не должен превышать 1 мегабайт.

Поддерживаются файлы практически всех популярных форматов: mp3, ogg, wav, aac, 3gp, flac. После загрузки файлы перекодируются (в 16-bit PCM моно или стерео) для упрощения воспроизведения.

Можно управлять скоростью воспроизведения аудио файла. Скорость 1,0 означает воспроизведение звука в исходном виде, 2,0 - воспроизведение в два раза быстрее, а 0,5 в два раза медленнее. Диапазон скорости воспроизведения составляет от 0,5 до 2,0.

Можно программно задать количество аудио потоков, проигрываемых одновременно. Если максимальное число потоков будет превышено, автоматически остановится поток с самым низкий приоритетом. При наличии нескольких потоков с тем же низким приоритетом, будет выбран старейший из них. В случае, если приоритет нового потока ниже, чем все активные потоки, новый звук не будет проигран. Ограничение максимального числа потоков помогает снизить загрузку процессора.

Конструктор класса SoundPool

SoundPool (int maxStreams, int streamType, int srcQuality)

* **maxStreams** - максимальное количество потоков, который могут воспроизводится одновременно
* **streamType** - тип аудиопотока, это константа из класса **AudioManager**. Здесь чаще всего используется AudioManager.STREAM\_MUSIC.
* **srcQuality** - качество кодирования. Сейчас этот параметр не используется, поэтому всегда будем передавать 0.

В API 21 конструктор считается устаревшим. Смотри ниже.

Методы

Загрузить аудио файлы в **SoundPool** можно с помощью четырёх методов:

int load(AssetFileDescriptor afd, int priority)

Загрузка из папки **assets**. Нужно знать путь к файлу. **AssetManager** - класс, предоставляющий доступ к файлам в директории **assets**. Получить экземпляр класса для данного приложения можно методом **getAssets()**. **AssetFileDescriptor** - файловый дескриптор для файла из директории **assets**.

int load(Context context, int resId, int priority)

Загрузка файла по идентификатору ресурса. Звуковые файлы могут находиться в каталоге **res/raw**, и соответственно идентификатор будет **R.raw.sound**.

int load(String path, int priority)

Загрузка файла из указанного пути.

int load(FileDescriptor fd, long offset, long length, int priority)

Загрузка файла из файловой системы устройства. Этот вариант полезен, если вы храните несколько звуков в одном двоичном файле. Параметр **offset** определяет смещение от начала файла, а **length** определяет длину звука.

В каждом методе мы передаем **priority** - приоритет файла. Методы возвращают идентификатор загруженного звука, который позже используется для воспроизведения.

Загрузка файлов происходит асинхронно, то есть после вызова метода **load()** файл не сразу будет доступен для воспроизведения, так как необходимо время для его загрузки и перекодирования. Для того чтобы отследить окончание загрузки нужно обработать событие **OnLoadCompleteListener**. Для этого устанавливаем обработчик события с помощью метода **void setOnLoadCompleteListener(SoundPool.OnLoadCompleteListener listener)**, где **SoundPool.OnLoadCompleteListener** - это интерфейс с единственным методом **void onLoadComplete(SoundPool soundPool, int sampleId, int status)**, который вызывается когда файл загружен. Здесь **sampleId** - идентификатор загруженного файла (то есть то, что вернул метод **load()**), а **status** - статус загрузки (значение 0 обозначает успешную загрузку).

Метод для проигрывания звука

int play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)

* **soundID** - идентификатор звука (который вернул load())
* **leftVolume** - уровень громкости для левого канала (от 0.0 до 1.0)
* **rightVolume** - уровень громкости для правого канала (от 0.0 до 1.0)
* **priority** - приоритет потока (0 - самый низкий)
* **loop** - количество повторов (0 - без повторов, (-1) - зациклен)
* **rate** - скорость воспроизведения (от 0.5 до 2.0)

Метод возвращает целое число **int**, которое используется в других методах в качестве параметра **streamID**.

Остановка воспроизведения

void stop(int streamID)

Приостановить воспроизведение

void pause(int streamID)

Возобновить приостановленный поток (если поток был не на паузе, то метод не даст никакого результата)

void resume(int streamID)

Установка количество повторов для потока

void setLoop(int streamID, int loop)

Установка приоритета для потока

void setPriority(int streamID, int priority)

Установка скорости для потока

void setRate(int streamID, float rate)

Установка громкости для потока

void setVolume(int streamID, float leftVolume, float rightVolume)

Приостановить все активные потоки

void autoPause()

Возобновить все потоки

void autoResume()

Удаляет звук из **SoundPool**. Метод возвращает **true**, если операция прошла успешно, и **false**, если такого **SoundID** не существует

boolean unload(int soundID)

Удаляет все загруженные звуки из **SoundPool** и освобождает память. После вызова этого метода экземпляр класса **SoundPool** уже нельзя использовать. Метод используется при выходе из программы, чтобы освободить ресурсы.

void release()

Пример

Добавим на форму две кнопки, а также звуковой файл в папку ресурсов **res/raw**. Образцы бесплатных звуков можно взять

Сайты с образцами звуков:

[Royalty Free Sound Effects (FX) Library for Download](http://soungle.com/)

[FindSounds - Search the Web for Sounds](http://www.findsounds.com/) (есть русский раздел)

[Free Sound Effects](http://rpg.hamsterrepublic.com/ohrrpgce/Free_Sound_Effects)

[Freesound.org](http://www.freesound.org/)

[Bfxr. Make sound effects for your games](http://www.bfxr.net/)

package ru.alexanderklimov.SoundPoolDemo;

import ...

public class AudioPlayer extends Activity {

private SoundPool mSoundPool;

private int mSoundId = 1;

private int mStreamId;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mSoundPool = new SoundPool(4, AudioManager.STREAM\_MUSIC, 100);

mSoundPool.load(this, R.raw.cat, 1);

Button playButton = (Button) findViewById(R.id.buttonPlay);

Button pauseButton = (Button) findViewById(R.id.buttonPause);

playButton.setOnClickListener(onPlayButtonClickListener);

pauseButton.setOnClickListener(onPauseButtonClickListener);

}

Button.OnClickListener onPlayButtonClickListener

= new Button.OnClickListener() {

@Override

public void onClick(View v) {

AudioManager audioManager = (AudioManager) getSystemService(Context.AUDIO\_SERVICE);

float curVolume = audioManager.getStreamVolume(AudioManager.STREAM\_MUSIC);

float maxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM\_MUSIC);

float leftVolume = curVolume / maxVolume;

float rightVolume = curVolume / maxVolume;

int priority = 1;

int no\_loop = 0;

float normal\_playback\_rate = 1f;

mStreamId = mSoundPool.play(mSoundId, leftVolume, rightVolume, priority, no\_loop,

normal\_playback\_rate);

Toast.makeText(getApplicationContext(),

"soundPool.play()",

Toast.LENGTH\_LONG).show();

}

};

Button.OnClickListener onPauseButtonClickListener

= new Button.OnClickListener() {

@Override

public void onClick(View v) {

mSoundPool.pause(mStreamId);

Toast.makeText(getApplicationContext(),

"soundPool.pause()",

Toast.LENGTH\_LONG).show();

}

};

}

Кнопку управления громкостью телефона можно настроить для управления громкости звуков:

context.setVolumeControlStream(AudioManager.STREAM\_MUSIC);

SoundPool.Builder (API 21)

В Android 5.0 (API 21) вызов конструктора считается устаревшим. На смену ему пришёл класс **SoundPool.Builder**. Ниже приводится код из одного проекта.

// sSoundPool = new SoundPool(5, AudioManager.STREAM\_MUSIC, 0);

AudioAttributes attributes = new AudioAttributes.Builder()

.setUsage(AudioAttributes.USAGE\_GAME)

.setContentType(AudioAttributes.CONTENT\_TYPE\_SONIFICATION)

.build();

sSoundPool = new SoundPool.Builder()

.setAudioAttributes(attributes)

.build();

# Кто сказал Мяу? - работаем со звуками Му, Мяу, Гав

Напишем программу, которая поможет определить, кто-же сказал Мяу? Меня всегда интересовал данный вопрос.

Подготовим заранее картинки различных животных и вставим их в папку **res/drawable-xhdpi**. Создадим разметку с кнопками:

<?xml version="1.0" encoding="utf-8"?>

<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/gridLayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="center"

android:columnCount="2"

android:rowCount="3"

tools:context=".MainActivity">

<ImageButton

android:id="@+id/imageButtonCow"

android:background="@android:color/transparent"

android:src="@drawable/cow" />

<ImageButton

android:id="@+id/imageButtonChicken"

android:background="@android:color/transparent"

android:src="@drawable/chicken" />

<ImageButton

android:id="@+id/imageButtonCat"

android:background="@android:color/transparent"

android:src="@drawable/cat" />

<ImageButton

android:id="@+id/imageButtonDuck"

android:background="@android:color/transparent"

android:src="@drawable/duck" />

<ImageButton

android:id="@+id/imageButtonSheep"

android:background="@android:color/transparent"

android:src="@drawable/sheep" />

<ImageButton

android:id="@+id/imageButtonDog"

android:background="@android:color/transparent"

android:src="@drawable/dog" />

</GridLayout>

Разметка представляет собой сетку, в которой можно задать число колонок и рядов.

Положим подготовленные аудио-файлы с голосами животных в директорию **assets**. По умолчанию в проекте такой папки нет. Выбираем **File | New | Folder | Assets Folder**. В диалоговом окне оставляем всё без изменений и нажимаем кнопку **Finish**. Файлы, лежащие в этой папке, считайте тоже ресурсами. Но они имеют свои особенности, в частности вы можете создавать свою структуру подпапок.

Переходим к программной части. Нам надо создать объект **SoundPool**, загрузить в него аудио-файлы из папки **assets** методом **load()**.

Зададим максимальное количество одновременно проигрываемых потоков - 3.

При нажатии на кнопку будем проигрывать нужный звук.

package ru.alexanderklimov.saymeow;

import android.annotation.TargetApi;

import android.content.res.AssetFileDescriptor;

import android.content.res.AssetManager;

import android.media.AudioAttributes;

import android.media.AudioManager;

import android.media.SoundPool;

import android.os.Build;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.MotionEvent;

import android.view.View;

import android.widget.ImageButton;

import android.widget.Toast;

import java.io.IOException;

public class MainActivity extends AppCompatActivity {

private SoundPool mSoundPool;

private AssetManager mAssetManager;

private int mCatSound, mChickenSound, mCowSound, mDogSound, mDuckSound, mSheepSound;

private int mStreamID;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// if (android.os.Build.VERSION.SDK\_INT < Build.VERSION\_CODES.LOLLIPOP) {

// // Для устройств до Android 5

// createOldSoundPool();

// } else {

// // Для новых устройств

// createNewSoundPool();

// }

// mAssetManager = getAssets();

//

// // получим идентификаторы

// mCatSound = loadSound("cat.ogg");

// mChickenSound = loadSound("chicken.ogg");

// mCowSound = loadSound("cow.ogg");

// mDogSound = loadSound("dog.ogg");

// mDuckSound = loadSound("duck.ogg");

// mSheepSound = loadSound("sheep.ogg");

ImageButton cowImageButton = (ImageButton) findViewById(R.id.imageButtonCow);

// cowImageButton.setOnClickListener(onClickListener);

ImageButton chickenImageButton = (ImageButton) findViewById(R.id.imageButtonChicken);

chickenImageButton.setOnClickListener(onClickListener);

ImageButton catImageButton = (ImageButton) findViewById(R.id.imageButtonCat);

catImageButton.setOnClickListener(onClickListener);

ImageButton duckImageButton = (ImageButton) findViewById(R.id.imageButtonDuck);

duckImageButton.setOnClickListener(onClickListener);

ImageButton sheepImageButton = (ImageButton) findViewById(R.id.imageButtonSheep);

sheepImageButton.setOnClickListener(onClickListener);

ImageButton dogImageButton = (ImageButton) findViewById(R.id.imageButtonDog);

dogImageButton.setOnClickListener(onClickListener);

cowImageButton.setOnTouchListener(new View.OnTouchListener() {

public boolean onTouch(View v, MotionEvent event) {

int eventAction = event.getAction();

if (eventAction == MotionEvent.ACTION\_UP) {

// Отпускаем палец

if (mStreamID > 0)

mSoundPool.stop(mStreamID);

}

if (eventAction == MotionEvent.ACTION\_DOWN) {

// Нажимаем на кнопку

mStreamID = playSound(mCowSound);

}

if (event.getAction() == MotionEvent.ACTION\_CANCEL) {

mSoundPool.stop(mStreamID);

}

return true;

}

});

}

View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.imageButtonCow:

playSound(mCowSound);

break;

case R.id.imageButtonChicken:

playSound(mChickenSound);

break;

case R.id.imageButtonCat:

playSound(mCatSound);

break;

case R.id.imageButtonDuck:

playSound(mDuckSound);

break;

case R.id.imageButtonSheep:

playSound(mSheepSound);

break;

case R.id.imageButtonDog:

playSound(mDogSound);

break;

}

}

};

@TargetApi(Build.VERSION\_CODES.LOLLIPOP)

private void createNewSoundPool() {

AudioAttributes attributes = new AudioAttributes.Builder()

.setUsage(AudioAttributes.USAGE\_GAME)

.setContentType(AudioAttributes.CONTENT\_TYPE\_SONIFICATION)

.build();

mSoundPool = new SoundPool.Builder()

.setAudioAttributes(attributes)

.build();

}

@SuppressWarnings("deprecation")

private void createOldSoundPool() {

mSoundPool = new SoundPool(3, AudioManager.STREAM\_MUSIC, 0);

}

private int playSound(int sound) {

if (sound > 0) {

mStreamID = mSoundPool.play(sound, 1, 1, 1, 0, 1);

}

return mStreamID;

}

private int loadSound(String fileName) {

AssetFileDescriptor afd;

try {

afd = mAssetManager.openFd(fileName);

} catch (IOException e) {

e.printStackTrace();

Toast.makeText(getApplicationContext(), "Не могу загрузить файл " + fileName,

Toast.LENGTH\_SHORT).show();

return -1;

}

return mSoundPool.load(afd, 1);

}

@Override

protected void onResume() {

super.onResume();

if (android.os.Build.VERSION.SDK\_INT < Build.VERSION\_CODES.LOLLIPOP) {

// Для устройств до Android 5

createOldSoundPool();

} else {

// Для новых устройств

createNewSoundPool();

}

mAssetManager = getAssets();

// получим идентификаторы

mCatSound = loadSound("cat.ogg");

mChickenSound = loadSound("chicken.ogg");

mCowSound = loadSound("cow.ogg");

mDogSound = loadSound("dog.ogg");

mDuckSound = loadSound("duck.ogg");

mSheepSound = loadSound("sheep.ogg");

}

@Override

protected void onPause() {

super.onPause();

mSoundPool.release();

mSoundPool = null;

}

}

При загрузке файлов метод **load()** возвращает идентификатор **soundID**, который сохраняем для дальнейшего использования. Объявим для каждого звука отдельную переменную, если же звуков много лучше завести для этого ассоциативный массив.

Файловый дескриптор **AssetFileDescriptor** для файла из директории **assets** получаем с помощью метода **openFd()**, принимающего в качестве параметра имя файла. Если файл не найден или не может быть открыт, то выводим сообщение и в качестве **soundID**возвращаем -1.

По нажатию кнопки вызываем метод **playSound()**, передавая ему нужный идентификатор звука. В методе проверяем этот идентификатор. Если файл не был найден, то метод **loadSound()** возвращает -1, а если метод **load()** класса **SoundPool** не смог загрузить файл, то **soundID** будет равен 0, поэтому проверяем, что *SoundID > 0*, что означает, что файл был успешно загружен. Если же все хорошо, то вызываем метод **play()**.

В версии Android 5.0 конструктор класса **SoundPool** является устаревшим. В коде использовано условие **if** с проверкой версии системы на устройстве, а также использованы аннотации, чтобы студия не ругалась на устаревший метод. Про аннотации мы поговорим в другой статье, пока воспринимайте их как подсказку-предупреждение при написании кода, чтобы выбрать правильный вариант.

Программа держит загруженные звуки в памяти. Если они вам не нужны, то нужно освободить ресурсы. Я сделал это в методе **onPause()**, соответственно загрузку пришлось перенести в **onResume()**.

Запустим программу и выясним, так кто-же сказал Мяу?



Написано по мотивам статьи [Звуковые эффекты](http://ru-code-android.blogspot.ru/2013/03/blog-post.html).

Один из читателей захотел выводить звук не через щелчок, а нажатие на кнопку. А когда палец открывается от экрана, то звук должен прекращаться. Получился интересный эффект, который мы нашли сообща. Код для кнопки с коровой (предыдущий код лучше убрать):

cowImageButton.setOnTouchListener(new View.OnTouchListener() {

public boolean onTouch(View v, MotionEvent event) {

int eventAction = event.getAction();

if (eventAction == MotionEvent.ACTION\_UP) {

// Отпускаем палец

if (mStreamID > 0)

mSoundPool.stop(mStreamID);

}

if (eventAction == MotionEvent.ACTION\_DOWN) {

// Нажимаем на кнопку

mStreamID = playSound(mCowSound);

}

if (event.getAction() == MotionEvent.ACTION\_CANCEL) {

mSoundPool.stop(mStreamID);

}

return true;

}

});

При воспроизведении звука мы получаем его идентификатор, используемый для остановки воспроизведения.