Text-to-Speech and SoundPool

By Colton Loewen

Table of Contents

[Introduction 1](#__RefHeading___Toc589_2901352904)

[Text-to-Speech (TextToSpeech) 1](#__RefHeading___Toc106_3629927699)

[Getting Started 2](#__RefHeading___Toc110_3629927699)

[Configuration 2](#__RefHeading___Toc692_3726144975)

[Speaking 3](#__RefHeading___Toc694_3726144975)

[SoundPool (SoundPool) 4](#__RefHeading___Toc108_3629927699)

[Alternatives 4](#__RefHeading___Toc591_2901352904)

[Getting Started 4](#__RefHeading___Toc112_3629927699)

[Loading Sounds 4](#__RefHeading___Toc713_2646493646)

[Configuration 5](#__RefHeading___Toc715_2646493646)

[Playing Sounds 5](#__RefHeading___Toc717_2646493646)

[Notes 5](#__RefHeading___Toc593_2901352904)

This tutorial is split into 4 parts. This introduction serves to define the structure and planned objectives. The following 2 sections describe and show how to properly implement the core features of the presented app. And the final section aims to provide additional information that did not directly relate to the core features (structure, dependencies, comments).

This tutorial is aimed at:

* Describing key steps in using features.
* Showing the syntax of a value, method in a class, or fully qualified object.

All code is enclosed in a dashed yellow box (where supported). Excluding this example.

* And options that can be taken to configure the feature.

All the facts presented in this tutorial can be found in the Bibliography at the bottom. For the best possible experience, use the Open Document Text (odt) version with LibreOffice Writer and disable spell checking.

# (TextToSpeech)

Text-to-Speech (TTS) is a technology which allows you to synthesize speech from text. The speech can than be immediately played, or saved to a file. To use TTS, construct the TextToSpeech class. This class should only be constructed when you need it and destructed when you are done with it.

## Getting Started

Before using TTS, you should make sure the user’s device has it installed (TTS is a bulky feature and may not be included on all devices). To do so, start an activity for a result using an intent with the TextToSpeech.Engine.ACTION\_CHECK\_TTS\_DATA action (Text-to-Speech Introduction). Pass the Intent to startActivityForResult(Intent intent, int requestCode) with a requestCode you can check for later (requestCode can not be negative).

If the user has a voice engine installed, they will be prompted to select one. Once the activity is finished, a result will be returned to the activities’ onActivityResult(int requestCode, int resultCode, Intent data) method. To check the result, confirm that the requestCode matches and test the resultCode against a constant provided by TextToSpeech.Engine (passing is CHECK\_VOICE\_DATA\_PASS and failing is CHECK\_VOICE\_DATA\_FAIL).

If the user does not have TTS installed, you can request for it to be installed by sending an intent with the TextToSpeech.Engine.ACTION\_INSTALL\_TTS\_DATA action (using startActivity(Intent intent)). This may fail, so be sure to check if TTS is installed again when the user returns (Text-to-Speech Introduction).

To use TTS, construct the TextToSpeech class (TextToSpeech(Context context, TextToSpeech.OnInitListener listener)). You can optionally include the package name of a TextToSpeech engine as the last argument. The listener’s onInit(int status) method will be called when it is finished loading.

You should always destruct the TextToSpeech instance when you are finished with it. TTS uses a lot of system resources and if you forget to do so, all other apps will be unable to use TTS. To do so, call shutdown() on your instance (Text-to-Speech Introduction).

## Configuration

How words are said can be customized by the user and your app. You should whoever not expect your configurations to always be applied as older versions of Android (Ice Cream and blow) allow users to override an app’s Text-to-Speech settings. You can check if this has occurred by calling areDefaultsEnforced() (TextToSpeech). You can:

* Change the language of speech.

The language can be set by calling setLanguage(Locale loc). The closest matching voice will be used (the language may not be available). You should set this to the language of your activity (Text-to-Speech Introduction).

* Use phonetic spelling to adjust how words are pronounced.

Phonetic spelling is a method of writing words using regular characters based on how they sound vocally (Phonetic Spelling). These spellings are often included in English dictionaries using Latin characters.

* Provide sound effects (earcons) for text using a sound file.

This can be done using the addEarcon(String earcon, String packagename, int resourceId) method. The package name is your application’s package name, which can be found in the manifest (TextToSpeech). Any time the earcon is found in text, the sound file will be played. To distinguish earcons from regular text, you should surround the earcon in brackets (example: **[alarm]**). A SUCCESS or ERROR constant will be returned.

* Provide pronunciations using a sound file.

You can add a pronunciation by calling addSpeech(String text, String packagename, int resourceId). The process is the same as adding an earcon. If the text matches text passed to speak (unlike with an earcon where every word is checked), the sound file will be played.

* Change the rate and pitch of the speech.

The rate (speed) of playback can be set by calling setSpeechRate(float speechRate). A rate of 1 is normal (TextToSpeech). You can set the pitch by calling setPitch(float pitch). A pitch of 1 is also normal.

* Provide audio attributes to control playback.

How sound is played can be controlled by calling setAudioAttributes(AudioAttributes audioAttributes). I describe AudioAttributes in more detail for the SoundPool.

* Change the used engine.

An installed or created engine can be supplied when constructing TextToSpeech to synthesize text. To create a custom engine, create a class derived from the TextToSpeechService (TextToSpeechService).

## Speaking

Text-to-Speech is based on a queue (first-out-last-in flow) which stores actions that the engine will need to perform. When synthesizing speech, a mode is provided to control how the entry will be inserted into this queue. Two modes are provided, QUEUE\_ADD (add to the queue) and QUEUE\_FLUSH (clear queue and add to the queue).

The queue is not manipulated directly, instead speak(CharSequence text, int queueMode, Bundle params, String utteranceId) or synthesizeToFile(CharSequence text, Bundle params, File file, String utteranceId) should be called. The params Bundle is not required (you can pass in null). It is for engine specific parameters (keys must be prefixed with the engine’s name). The utteranceId is a unique identifier for the request.

# SoundPool (SoundPool)

The SoundPool is a technology which allows you to play sounds. In order to play a sound, a stream needs to be created, and the file needs to be loaded. The sound pool automatically creates and reuses streams for you when you create a SoundPool class.

## Alternatives

Alternatives to the SoundPool class exist. They are the MediaPlayer, and AudioTrack. When to use each of these classes depends on the quantity and length of the sounds you plan to play.

* The AudioTrack is best for **singular**, **short** sounds.
* The SoundPool is best for **multiple**, **short** sounds.
* The MediaPlayer is best for **long** sounds.

## Getting Started

In order to create the SoundPool class, you need to use the SoundPool.Builder. You can construct the builder, than optionally set it’s maximum number of streams (setMaxStreams(int maxStreams)), and set audio attributes (setAudioAttributes(AudioAttributes attributes)). Once the builder is properly setup, call build().

If you are supporting Android API 20 or below, the SoundPool.Builder will not be supported and you will have to use the sound pool constructor. The constructor (SoundPool(int maxStreams, int streamType, int srcQuality)), takes a maximum number of streams (how many sounds can play at a time), a stream type (which volume control is used), and a source quality (never used, provide 0). The max streams depends on your use-case. You should allow for your sounds to play, while not using up too many system resources (for example, 2 for playing button sound effects). The stream type is a constant which identifies which volume control is used. AudioManager.STREAM\_MUSIC (AudioManager) works fine in most cases.

Like with the TextToSpeech class, the SoundPool instance should be destructed when you are finished with it. You can do so by calling release(). Other apps will still be able to play sounds if you forget, whoever it still uses a fair amount of system resources.

## Loading Sounds

Android supports a wide range of audio formats (Media Formats), including Wave, MPEG-4, FLAC, and my personal favorite, Ogg (open-source, non-proprietary, compressed format). These sound files can be included as resources with your app in the raw resource folder. If you are unable to make the sounds you want, you can find several from [freesound.org](https://freesound.org/).

To track when a sound loads, you need to call setOnCompleteListener(SoundPool.OnLoadCompleteListener listener) on the SoundPool. You can load a sound by calling load(Context context, int resId, int priority). The context is your context, the resId is your sound resource id (example: R.raw.button\_pressed), and the priority currently has no effect (enter 1 for future compatibility). The method will return an integer which you can use to play the sound (a stream id).

## Configuration

The sound pool can be configured before (using the builder) and after you create it (through its methods):

* You can pass in AudioAttributes.

While constructing the sound pool, you can set AudioAttributes, which describe the audio streams.

* Pass in arguments when playing a sound.

In order to play a sound, you will need to pass in some arguments which describe how to play the sound.

* Change how a sound is playing, while it is playing.

Many of the arguments you pass in to play a sound can later be changed using methods of the sound pool and the returned stream id.

## Playing Sounds

In order to loaded sound, call play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate). You need to pass in the sound id that was returned by load earlier. The leftVolume and rightVolume are the the left and right speaker volumes respectively (1 being the normal volume). The priority is used when finding sounds to recycle (0 being the lowest). If you do not want the sound to loop, pass in 0 for the loop, otherwise pass in a number of times to loop. The rate is the speed to play the sound at (1 being normal). Calling this method returns a streamId, which can be used to change how the sound is playing.

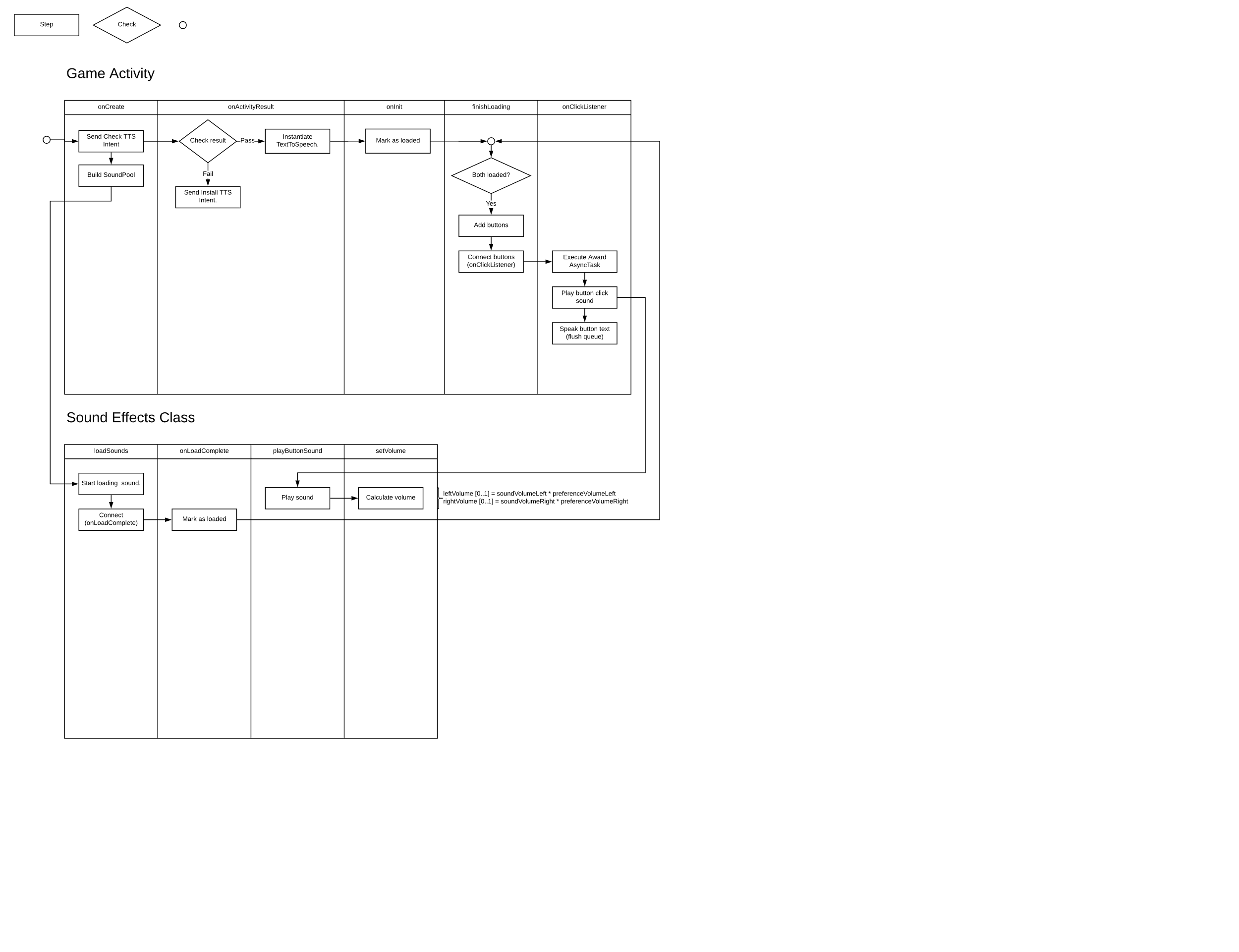
The SoundPool may reclaim a loaded sound’s resources at any time. To determine which sound should be reclaimed, it uses the priority. This behavior can occur at any time, and can not be configured. Fortunately, even if this does occur, you can continue to use the sound id with any of the sound pool’s methods.

# Notes

The app uses 2 activities. The first activity is the game itself (in GameActivity.java). The game activity is the starting point, and connects all of the features provided by the app. In order to properly support the mascot, the com.android.support:support-vector-drawable:27.1.1 dependency was added (along with adding vectorDrawables.useSupportLibrary = true to android.defaultConfig). The core features are all implemented in the game activity. All other classes where designed for support (providing settings, managing points, and playing sounds).

The second activity is for settings related to the game (in SettingsActivity.java). The settings activity is only used as a frame for a preferences fragment. The slider preference comes from the com.github.jayschwa:AndroidSliderPreference:v1.0.1 dependency.

* Some warnings have been suppressed, these issues have been looked into and addressed.
* An OnTouchListener was used in place of an OnClickListener purely to provide pointer coordinates for stereo sound.
* The Points class uses ArrayLists as a First-in-Last-out queue. Java does support queue classes but does not provide adequate methods to view arbitrary entries in them.
* Was this tutorial helpful, please let me know! <https://www.strawpoll.me/16218891>



Bibliography

TextToSpeech: Android Open Source, , 2018, https://developer.android.com/reference/android/speech/tts/TextToSpeech.html

Text-to-Speech Introduction: Android Developers Blog, Introduction to Text-To-Speech in Android, 2009, https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html

Phonetic Spelling: Dictionary.com, What is phonetic spelling?, 2018, https://www.dictionary.com/e/phonetic-spelling/

TextToSpeechService: Android Open Project, , 2018, https://developer.android.com/reference/android/speech/tts/TextToSpeechService

SoundPool: Android Open Project, , 2018,

AudioManager: Android Open Project, , 2018, https://developer.android.com/reference/android/media/AudioManager

Media Formats: Android Open Project, Supported Media Formats, 2018, https://developer.android.com/guide/topics/media/media-formats