Mattia Raffaele Ricciardelli & Federico Santavicca

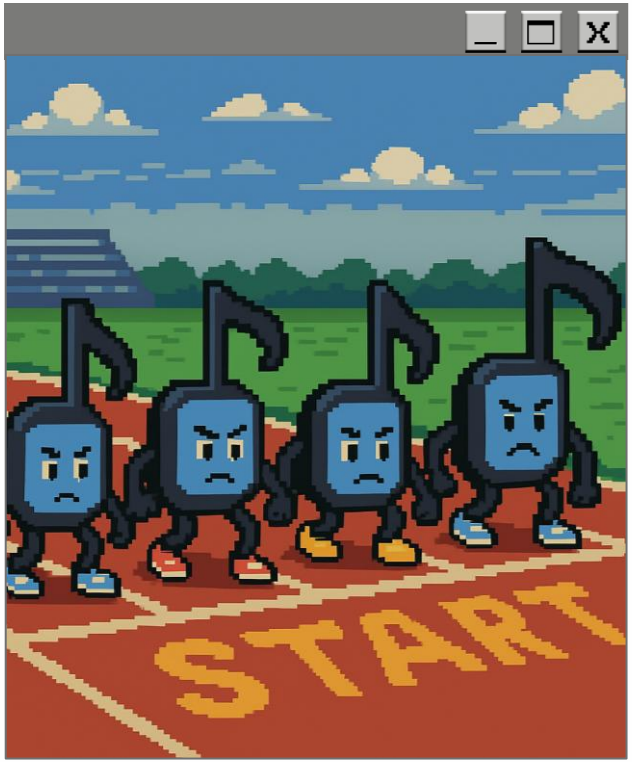
How-To: Music & Sounds on Android Studio

Playback and recording of audio using Android APIs

Table of contents

01 SoundPool

Short audio playing directly loaded on RAM



What it is:

- A class from android.media for playing **short sound clips** efficiently.
- Loads samples into **RAM** and plays them instantly with very low latency.
- Ideal for games, soundboards, and UI feedback (clicks, alerts, etc.).

Key Features:

- Support multiple simultaneous sounds
- Uses AudioAttributes to define usage and content type
- Sounds are preloaded -> minimal delay
- Automatically manages stream priority

Implementation

```
2
3   // 1. Create AudioAttributes
4   val attributes = AudioAttributes.Builder()
5       .setUsage(AudioAttributes.USAGE_GAME)
6       .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
7       .build()
8
9   // 2. Create SoundPool
10  val soundPool = SoundPool.Builder()
11      .setAudioAttributes(attributes)
12      .setMaxStreams(8)
13      .build()
14
15  // 3. Load short sounds (in res/raw)
16  val clickId = soundPool.load(context, R.raw.click, 1)
17
18  // 4. Play the sound
19  soundPool.play(clickId, 1f, 1f, 1, 0, 1f)
20
```

02 MediaPlayer

Audio/Video files player

What it is:

- A high-level class for playing **longer audio or video files**.
- Streams and decodes data gradually, not all at once.
- Suitable for music, podcasts, or audio from URLs.

Key Features:

- Supports **local and remote sources**
- Handles buffering, pausing, and resuming automatically.
- Works with many formats: MP3, WAV, AAC, etc
- Can play from **res/raw**, **file paths**, or **URLs**.

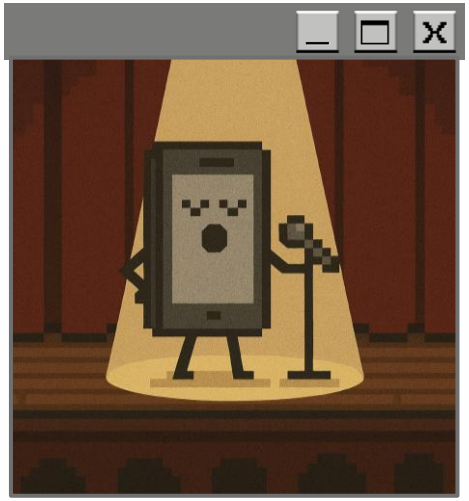
SoundPool vs MediaPlayer

Feature	**SoundPool**	**MediaPlayer**
Purpose	Short effects	Long tracks
Memory	Loads fully into RAM	Streams from storage/network
Latency	Very low (instant)	Higher (prepare time)
Typical Use	Games, UI clicks	Music, podcasts
Simultaneous Sounds	Yes (multi-stream)	Usually one per player
API Focus	Speed	Control & playback features

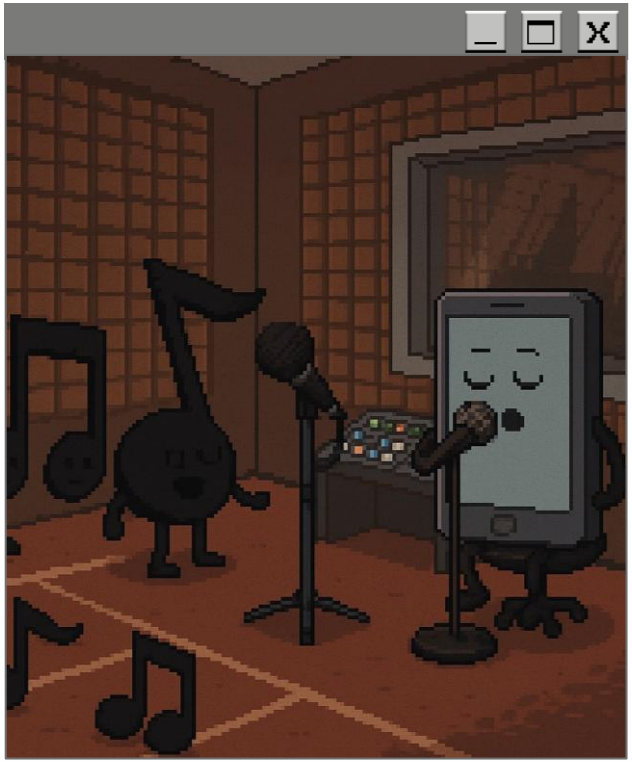
Implementation

```
2
3   // From app resource (res/raw)
4   val mediaPlayer = MediaPlayer.create(context, R.raw.music)
5   mediaPlayer.start()
6
7   // Pause and resume
8   mediaPlayer.pause()
9   mediaPlayer.start()
0
1   // Stop and release when done
2   mediaPlayer.stop()
3   mediaPlayer.release()
4
```

03 MediaRecorder



Audio Recording Library



What it is:

- A high-level API to **record audio (or video)** from the microphone.
- Encodes and saves directly to a file (.3gp, .m4a, .mp4)
- Manages input, encoding, and writing automatically

Key Features:

- Voice notes, interviews, or voice message apps

Implementation

```
val outputFile = "${context.externalCacheDir?.absolutePath}/recording.3gp"

val recorder = MediaRecorder().apply {
    setAudioSource(MediaRecorder.AudioSource.MIC)
    setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
    setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
    setOutputFile(outputFile)
}

recorder.prepare()          // Request microphone permission
recorder.start()            // ActivityCompat.requestPermissions(
// ... recording ...        //     this,
recorder.stop()             //     arrayOf(Manifest.permission.RECORD_AUDIO),
recorder.release()          //     200
                            // )

```

04 AudioRecorder

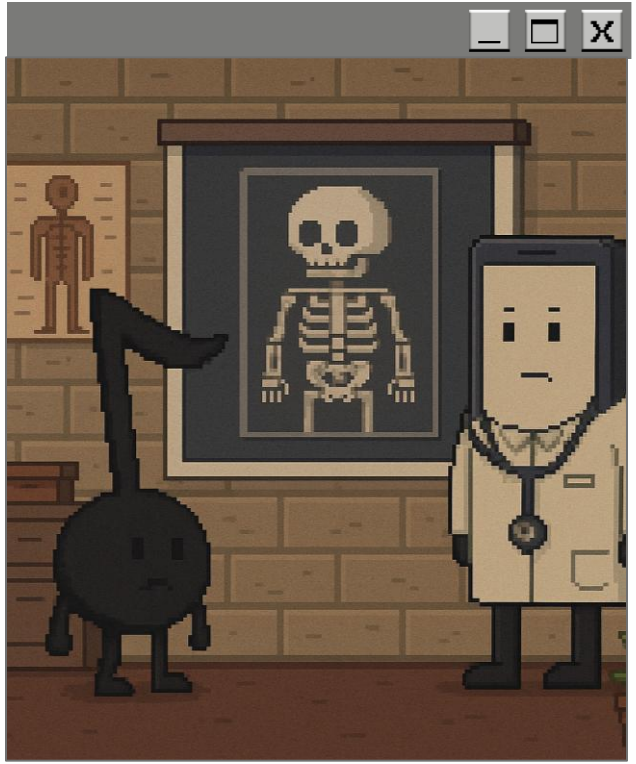
Audio Recorder for a more technical use

What it is:

- A **low-level API** for capturing **raw PCM audio samples**
- Unlike MediaRecorder, it gives **real-time access** to the audio buffer.
- Used for signal analysis, visualizers, filters, or ML models.

Key Features:

- Spectrum analyzers or VU meters.
- Real-time sound processing and feature extraction.
- Scientific or research recording.



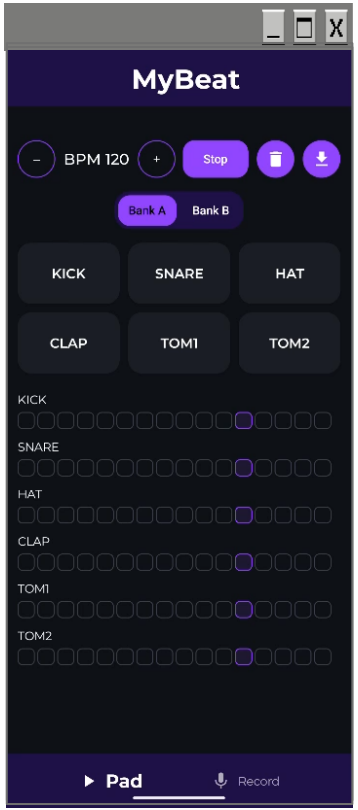
05 Demo App

Beat making and vocal recording app

What it is:

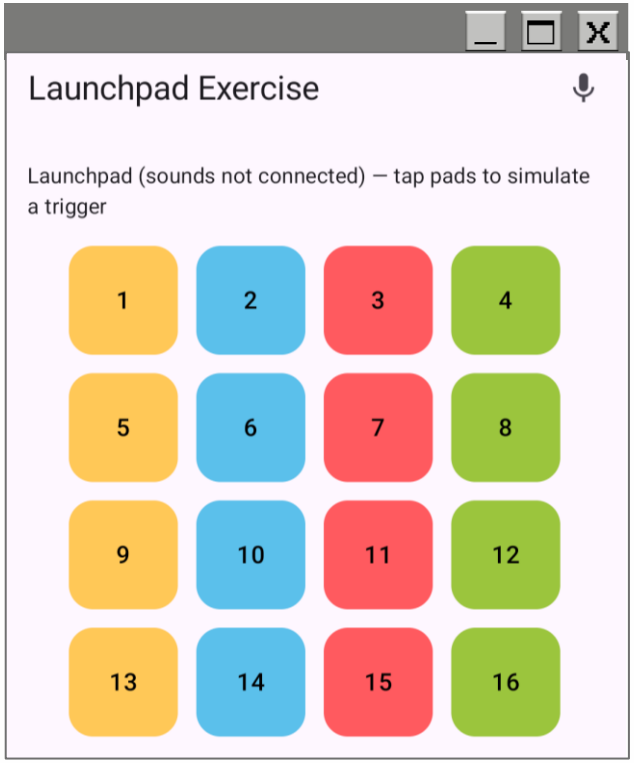
An app that allows you to:

- **Create beats** using pre-sampled sounds in a **launchpad-style**
- **Play, pause, and save** your composed beats in a **local directory**
- **Record voice or audio** over the beats produced within the app
- **Mix recorded beats and vocals** to create and save a **complete song**



06 Exercise

Complete the missing parts to create your own launchpad and voice recorder



Tasks for Students:

- Connect samples to pads (SoundPool/Media Player)
- Implement Recording in Recorder (Don't forget permission request)

Extra Points (Optional):

- Overdub a backing beat while recording
- Show and Play saved recordings

Files to edit:

- SoundEngine.kt
- Recorder.kt
- LaunchpadScreen.kt

THANK YOU :)