Android音频开发之AudioTrack

未见哥哥 已关注
1 2018.05.17 17:53:17 字数 1,304 阅读 19,120

在前两节中分享了Android音频开发之音频基本概念和Android音频开发之音频采集，本文分享的是如何使用 AudioTrack 来播放 使用AudioRecord 采集后的 PCM 数据。

1. 构造 AudioTrack 实例

```
public AudioTrack(int streamType, int sampleRateInHz, int channelConfig, int audioForm
```

> 在采样 `pcm` 音频数据需要设置对应的采样率，采样精度，采样的通道数和采样的缓冲区大小，如果播放的音频是使用 `AudioRecord` 录制的，那么这些参数配置信息需要和 `AudioRerord` 一致，不然播放就会出现奇怪的问题。

1.1 AudioTrack 播放音频时会有两种方式：

> 音频播放的方式，有两种方式 MODE_STATIC 或者 MODE_STREAM 。

- MODE_STATIC 预先将需要播放的音频数据读取到内存中，然后才开始播放。
- MODE_STREAM 边读边播，不会将数据直接加载到内存

1.2 MODE_STREAM 的方式构建 AudioTrack 实例

```
/**
* 构建 AudioTrack 实例对象
*/
private void createStreamModeAudioTrack() {
    if (audioTrack == null) {
        bufferSize = AudioTrack.getMinBufferSize(44100, AudioFormat.CHANNEL_OUT_STEREO
        audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                44100, AudioFormat.CHANNEL_OUT_STEREO, AudioFormat.ENCODING_PCM_16BIT,
    }
}
```

1.3 MODE_STATIC 的方式构建 AudioTrack 实例

```
/**
* 构建 AudioTrack 实例对象
*/
//file 就是需要播放的音频文件，这里的buffersize就是文件的大小
audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
        44100, AudioFormat.CHANNEL_OUT_STEREO,
        AudioFormat.ENCODING_PCM_16BIT, (int) file.length(), AudioTrack.MODE_STATIC);
```

2. 写入数据

> 不间断通过 `write` 方法的写数据给 AudioTrack .

注意： 对于 MODE_STREAM 写入数据，会阻塞，直到写入的数据都传输给 AudioTrack。
对于 MODE_STATIC 会将数据拷贝到缓冲区中，并在该方法返回后执行 play() 方法播放音频数据。

- int write (byte[] audioData, int offsetInBytes,int sizeInBytes)
- int write (short[] audioData, int offsetInShorts, int sizeInShorts)
- int write (float[] audioData, int offsetInFloats, int sizeInFloats,int writeMode)

该方法的返回值：

- 正确：>=0 该值表示写入的数据量。
- 错误：<0
  - ERROR_INVALID_OPERATION
  - ERROR_BAD_VALUE
  - ERROR_DEAD_OBJECT
  - ERROR

2.1 两种方式写入数据的区别

- MODE_STATIC

在 AudioTrack 创建之处，会初始化一个与其相关联的 buffer 缓冲区，这个缓冲区的大小是在构造方法指定的。这个大小表示 AudioTrack 可以播放多久。对于 `MODE_STATIC` 这种模式下，这个 buffer 的大小就是需要播放的文件或者流的大小。

```
//写入数据大小 array 就是预先将音频数据加载到array数组中
int writeResult = audioTrack.write(array, 0, array.length);
//检查写入的结果，如果是异常情况，则直接需要释放资源
if (writeResult == AudioTrack.ERROR_INVALID_OPERATION || writeResult == AudioTrack.ERR
        || writeResult == AudioTrack.ERROR_DEAD_OBJECT || writeResult == AudioTrack.ER
    //出异常情况
    isPlaying = false;
    release();
    return;
}

```

- MODE_STREAM

使用这种方式是通过将数据写入到缓冲区中，而需要注意写入到这个缓冲区的数据大小，需要确保小于或者等于这个构造 AudioTrack 的缓冲区大小。

AudioTrack 不是 final 类型，也就是说可以使用继承实现自己的功能，但是官方文档表示不建议这样做。

```
//边读边播
byte[] buffer = new byte[bufferSize];
while (fis.available() > 0) {
    int readCount = fis.read(buffer);
    if (readCount == -1) {
        Log.e(TAG, "没有更多数据可以读取了");
        break;
    }
    int writeResult = audioTrack.write(buffer, 0, readCount);
    if (writeResult >= 0) {
        //success
    } else {
        //fail
        //丢掉这一块数据
        continue;
    }
}

```

这个缓冲区大小可以通过 AudioTrack.getMinBufferSize 来获取

```
bufferSize = AudioTrack.getMinBufferSize(44000, AudioFormat.CHANNEL_OUT_STEREO, AudioF
```

3. 状态判断

3.1 AudioTrack 状态判断

> 检测一个已经创建好的 AudioTrack 的状态，确保操作在正确初始化之后进行。当需要进行播放前，校验 AudioTrack 是否处于正确的状态。

```
int getState ()
```

返回值介绍：

- STATE_INITIALIZED 表示 AudioTrack 已经是可以使用了。
- STATE_UNINITIALIZED 表示 AudioTrack 创建时没有成功地初始化。
- STATE_NO_STATIC_DATA 表示当前是使用 MODE_STATIC ，但是还没往缓冲区中写入数据。当接收数据之后会变为 STATE_INITIALIZED 状态。

```
//播放时，状态校验
if (audioTrack.getState() != AudioTrack.STATE_INITIALIZED) {
    Log.e(TAG, "不能播放，当前播放器未处于初始化状态..");
    return;
}
```

3.2 AudioTrack 播放状态

```
int getPlayState()
```

- PLAYSTATE_STOPPED 停止
- PLAYSTATE_PAUSED 暂停
- PLAYSTATE_PLAYING 正在播放

4. 播放 play

- 对于 MODE_STATIC 模式，必须要调用 `write(...)` 相关方法将数据写入到对应的缓冲区中，然后才可以调用 `paly(...)` 方法进行播放操作。

```
//先将所有的数据写入到缓冲区
write(...)
//然后在播放
play(..)
```

- 对于 MODE_STREAM 模式

```
paly(...)

new Thread() {
    public void run() {
        //一系列的 write 操作
        `write(...)`
    }

}.start();
```

5. AudioTrack 状态

5.1 停止播放

> 对于 `MODE_STREAM` 模式，如果单是调用 stop 方法， AudioTrack 会等待缓冲的最后一帧数据播放完毕之后，才会停止，如果需要立即停止，那么就需要调用 `pause` 然后调用 `flush` 这两个方法，那么 AudioTrack 就是丢缓冲区中剩余的数据。

```
void stop ()
```

5.2 暂停

> 暂停播放，但是缓冲区中没有被播放的数据不会被舍弃，调用 play 方法即可接着播放，

```
void pause ()
```

5.3 刷新

> 刷新正在排队播放的音频数据，调用该方法会将写入到缓冲区但没有被播放的音频数据都会被丢弃。如果是非 STREAM 或者没有执行 pasuse 或者 stop 将不会有任何效果。

```
void flush()
```

5.4 释放

> 释放本地 AudioTrack 对象。

```
void release ()
```

示例代码

```
public void stop() {
    if ((audioTrack != null) && (audioTrack.getState() == AudioTrack.STATE_INITIALIZED
        if (audioTrack.getPlayState() != AudioTrack.PLAYSTATE_STOPPED) {
            audioTrack.flush();
            audioTrack.stop();
        }
}
```

6. AudioTrack 和 MediaPlayer 的区别?

- AudioTrack 只能播放 pcm 原始数据，不能播放视频。
- MediaPlayer 可以播放视频和音频。
- AudioTrack 只支持 pcm 原始音频数据。
- MediaPlayer 支持 mp3,wav,aac...

- MediaPlayer 在底层会创建指定的格式的解码器，将音频数据转化为 pcm 然后再交给 pcm 去播放。MediaPlayer底层会创建 AudioTrack，将解码后的数据交给 AudioTrack 播放。

- 每一个音频流对应着一个AudioTrack类的一个实例，
  每个AudioTrack会在创建时注册到 AudioFlinger中，
  由AudioFlinger把所有的AudioTrack进行混合（Mixer），然后输送到AudioHardware中进行播放，目前Android同时最多可以创建32个音频流，也就是说，Mixer最多会同时处理32个AudioTrack的数据流。

7. 参考文档

- https://juejin.im/entry/5a1cda475188254a701ec89b 深入理解Android音频框架AudioTrack到AudioFlinger及Mix过程。
- https://www.ktanx.com/blog/p/2408 Android音频: 如何使用AudioTrack播放一个WAV格式文件?