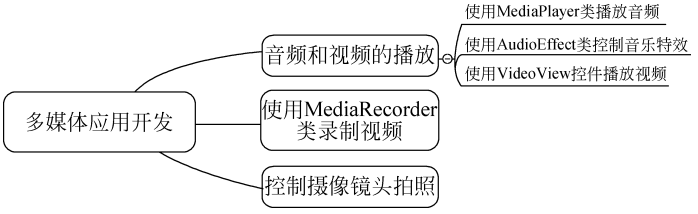
第5章 多媒体应用开发

本章导图



主要内容

- 使用 MediaPlayer 类播放音频。
- 使用 AudioEffect 类控制音乐特效。
- 使用 VideoView 控件播放视频。
- 使用 MediaRecorder 类录制视频。
- 控制摄像头拍照。

难点

- 掌握各种音频视频类和控件的使用方法。
- Android 相机拍照功能的实现。

随着手机硬件的不断提升，手机已经成为人们日常生活中必不可少的设备，设备里面的多媒体资源想必是很多人的兴趣所在。多媒体资源一般包括音频、视频等，Android 系统针对不同的多媒体提供了不同的类进行支持。

Android 提供了常见的音频、视频的编码、解码机制，支持的音频格式有 MP3、WAV 和 3GP 等，支持的视频格式有 MP4 和 3GP 等。接下来，本章将针对多媒体应用中的音频和视频操作进行讲解。

5.1 音频和视频的播放

Android 提供了简单的 API 来播放音频、视频，下面将详细介绍如何使用它们。

5.1.1　使用 MediaPlayer 类播放音频

Android 应用中播放音频文件的功能一般都是通过 MediaPlayer 类实现的，该类提供了全面的方法支持多种格式的音频文件。MediaPlayer 类的常用方法如表 5.1 所示。


观看视频

表 5.1　MediaPlayer 类的常用方法

方　法	说　明
setDataSource()	设置要播放音频文件的位置
prepare()	在开始播放之前调用该方法完成准备工作
start()	开始或继续播放音频
pause()	暂停播放音频
reset()	重置 MediaPlayer 对象
seekTo()	从指定位置开始播放音频
stop()	停止播放音频，调用该方法后 MediaPlayer 对象无法再播放音频
release()	释放与 MediaPlayer 对象相关的资源
isPlaying()	判断当前是否正在播放音频
getDuration()	获取载入的音频文件的时长

接下来通过演示使用 MediaPlayer 类播放音频的过程来了解 MediaPlayer 的使用，具体如下。

1. 实例化 MediaPlayer 类

使用 MediaPlayer 类播放音频时，首先创建一个 MediaPlayer 类的对象，接着调用 setAudioStreamType()方法设置音频类型。示例代码如下：

```
MediaPlayer mediaPlayer = new MediaPlayer ();                    //创建 MediaPlayer 类的对象
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);       //设置音频类型
```

上述代码中的 setAudioStreamType()方法中传递的参数表示音频类型。音频类型有很多种，最常用的音频类型有以下几种。

- AudioManager.STREAM_MUSIC：音乐。
- AudioManager.STREAM_RING：响铃。
- AudioManager.STREAM_ALARM：闹钟。
- AudioManager.STREAM_NOTIFICTION：提示音。

2. 设置数据源

根据音频文件存放的位置不同，将数据源的设置分为三种方式，分别为设置播放应用自带的音频文件、设置播放 SD 卡中的音频文件和设置播放网络音频文件。示例代码如下：

```
//1.设置播放应用自带的音频文件
mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.xxx);
//2.设置播放 SD 卡中的音频文件
mediaPlayer.setDataSource("SD 卡中的音频文件的路径");
//3.设置播放网络音频文件
mediaPlayer.setDataSource("http://www.xxx.mp3");
```

需要注意的是，播放网络中的音频文件时，需要在清单文件中添加访问网络的权限，示例代码如下。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

3. 播放音频文件

一般在调用 start()方法播放音频文件之前，程序会调用 prepare()方法或 prepareAsync()方法将音频文件解析到内存中。prepare()方法为同步操作，一般用于解析较小的文件；prepareAsync()方法为异步操作，一般用于解析较大的文件。

(1) 播放小音频文件。

示例代码如下：

```
mediaPlayer.prepare();
mediaPlayer.start();
```

需要注意的是，使用 create()方法创建 MediaPlayer 对象并设置音频文件时，不能调用 prepare()方法，直接调用 start()方法播放音频文件即可。

(2) 播放大音频文件。

示例代码如下：

```
mediaPlayer.prepareAsync();
mediaPlayer.setOnPreparedListener(new OnPreparedlistener){
    public void onPrepared(MediaPlayer player){
        player.start();
    }
}
```

上述代码中，prepareAsync()方法是子线程中执行的异步操作，不管它是否执行完毕，都不会影响主线程操作。setOnPreparedListener()方法用于设置 MediaPlayer 类的监听器，用于监听音频文件是否解析完成，如果解析完成，则会调用 onPrepared()方法，在该方法内部调用 start()方法播放音频文件。

4. 暂停播放

pause()方法用于暂停播放音频。在暂停播放之前，首先要判断 MediaPlayer 对象是否存在，并且当前是否正在播放音频。示例代码如下：

```
if(mediaPlayer!= null && mediaPlayer.isPlaying()){
    mediaPlayer.pause();
}
```

5. 重新播放

seekTo()方法用于定位播放。该方法用于快退或快进音频播放，方法中传递的参数表示将播放时间定在多少毫秒，如果传递的参数为 0，则表示从头开始播放。示例代码如下：

```
//1.播放状态下进行重播
if(mediaPlayer!= null && mediaPlayer.isPlaying()){
    mediaPlayer.seekTo(0);                    //设置从头开始播放音频
    return;
}
//2.暂停状态下进行重播,要调用 start()方法
```

```
if(mediaPlayer!= null){
    mediaPlayer.seekTo(0);              //设置从头开始播放音频
    mediaPlayer.start();
}
```

6. 停止播放

stop()方法用于停止播放，停止播放之后还要调用 release()方法将 MediaPlayer 对象占用的资源释放并将该对象设置为 null。示例代码如下：

```
if(mediaPlayer!= null && mediaPlayer.isPlaying()){
    mediaPlayer.stop();                    //停止播放
    mediaPlayer.release();                 //释放 MediaPlayer 对象占用的资源
    mediaPlayer = null;
}
```

7. 播放不同来源音频文件的步骤

下面简单归纳一下用 MediaPlayer 类播放不同来源音频文件的步骤。

1）播放应用的资源文件

播放应用的资源文件需要如下两步。

（1）调用 MediaPlayer 类的 create(Context context,int resid)方法加载指定资源文件。

（2）调用 MediaPlayer 类的 start()、pause()、stop()等方法控制播放。

例如如下代码：

```
MediaPlayer mPlayer = MediaPlayer.create(this,R.raw.song);
mPlayer.start();
```

提示：音频资源文件一般放在 Android 应用的/res/raw 目录下。

2）播放应用的原始资源文件

播放应用的原始资源文件按如下步骤执行。

（1）调用 Context 对象的 getAssets()方法获取应用的 AssetManager。

（2）调用 AssetManager 对象的 openFd(String name)方法打开指定的原始资源，该方法返回一个 AssetFileDescriptor 对象。

（3）调用 AssetFileDescriptor 对象的 getFileDescriptor()、getStartOffset()和 getLength()方法来获取音频文件的文件描述符、开始位置、长度等。

（4）创建 MediaPlayer 对象，并调用 MediaPlayer 对象的 setDataSource(FileDescriptor fd,long offset,long length)方法来装载音频资源。

（5）调用 MediaPlayer 对象的 prepare()方法准备音频。

（6）调用 MediaPlayer 对象的 start()、pause()、stop()等方法控制播放。

注意，虽然 MediaPlayer 对象提供了 setDataSource(FileDescriptor fd)方法来装载指定音频资源，但实际使用时这个方法似乎有问题：不管程序调用 openFd(String name)方法时指定打开哪个原始资源，MediaPlayer 将总是播放第一个原始的音频资源。

例如如下代码片段：

```
AssetManager am = getAssets();
//打开指定音乐文件
AssetFileDescriptor afd = am.openFd(music);
MediaPlayer mPlayer = new MediaPlayer();
//使用 Mediaplayer 加载指定的声音文件
mPlayer.setDataSource(afd.getFileDescriptor()
    , afd.getStartOffset()
    , afd.getLength());
//准备声音
mPlayer.prepare();
//播放
mPlayer.start();
```

3）播放外部存储器上的音频文件

播放外部存储器上的音频文件按如下步骤执行。

（1）创建 MediaPlayer 对象，并调用 MediaPlayer 对象的 setDateSource(String path)方法装载指定的音频文件。

（2）调用 MediaPlayer 对象的 prepare()方法准备音频。

（3）调用 MediaPlayer 对象的 start()、pause()、stop()等方法控制播放。

例如如下代码：

```
MediaPlayer mPlayer = new MediaPlayer();
//使用 MediaPlayer 加载指定的声音文件
mPlayer.setDataSource("/mnt/sdcard/mysong.mp3");
//准备声音
mPlayer.prepare();
//播放
mPlayer.start();
```

4）播放来自网络的音频文件

播放来自网络的音频文件有两种方式：①直接使用 MediaPlayer 对象的静态 create(Contextcontext, Uri uri)方法；②调用 MediaPlayer 对象的 setDataSource(Context context, Uri uri)装载指定 Uri 对应的音频文件。

以第二种方式播放来自网络的音频文件的步骤如下。

（1）根据网络上的音频文件所在的位置创建 Uri 对象。

（2）创建 MediaPlayer 对象，并调用 MediaPlayer 对象的 setDateSource(Context context,Uri uri)方法装载 Uri 对应的音频文件。

（3）调用 MediaPlayer 对象的 prepare()方法准备音频。

（4）调用 MediaPlayer 对象的 start()、pause()、stop()等方法控制播放。

例如如下代码片段：

```
Uri uri = Uri.parse("http://www.crazyit.org/abc.mp3");
MediaPlayer mPlayer = new MediaPlayer();
//使用 MediaPlayer 根据 Uri 来加载指定的声音文件
mPlayer.setDataSource(this, uri);
//准备声音
mPlayer.prepare();
```

```
//播放
mPlayer.start();
```

MediaPlayer 除了调用 prepare()方法来准备声音之外，还可以调用 prepareAsync()方法来准备声音。prepareAsync()方法与普通 prepare()方法的区别在于，prepareAsync()方法是异步的，它不会阻塞当前的 UI 线程。

归纳起来，MediaPlayer 的状态图如图 5.1 所示。

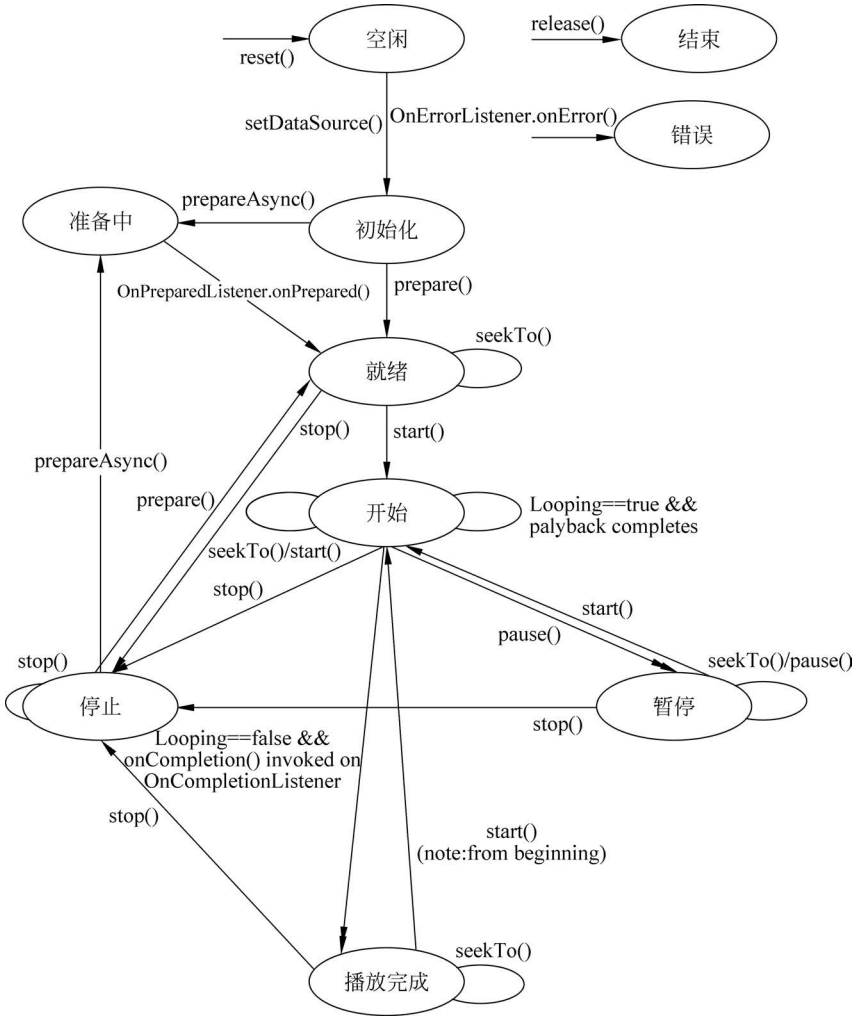


图 5.1　MediaPlayer 的状态图

5.1.2　使用 AudioEffect 类控制音乐特效

Android 可以控制播放音乐时的均衡器、重低音、音场及显示音乐波形等，这些都是靠 AudioEffect 及其子类来完成的，它包含如下常用子类：

- AcousticEchoCanceler：取消回声控制器。
- AutomaticGainControl：自动增益控制器。

- NoiseSppressor：噪声压制控制器。
- BassBoost：重低音控制器。
- Equalizer：均衡控制器。
- PresetReverb：预设音场控制器。
- Visualizer：示波器。

上面的子类中前三个子类的用法很简单，只要调用它们的静态 create()方法创建相应的实例，然后调用它们的 isAvailable()方法判断是否可用，再调用 setEnabled(boolean enabled)方法启用相应效果即可。

1. AcousticEchoCanceler：取消回声控制器

该功能的示意代码如下：

```
//获取取消回声控制器
AcousticEchoCanceler canceler = AcousticEchoCanceler.create(0
    , mPlayer.getAudioSessionId())
if(canceler.isAvailable())
{
    //启用取消回声功能
    canceler.setEnabled(true);
}
```

2. AutomaticGainControl：自动增益控制器

该功能的示意代码如下：

```
//获取自动增益控制器
AutomaticGainControl ctrl = AutomaticGainControl.create(0
    , mPlayer.getAudioSessionId())
if(ctrl.isAvailable())
{
    //启用自动增益控制功能
    ctrl.setEnabled(true);
}
```

3. NoiseSppressor：噪声压制控制器

该功能的示意代码如下：

```
//获取噪声压制控制器
NoiseSuppressor suppressor = NoiseSuppressor.create(0
    , mPlayer.getAudioSessionId())
if(suppressor.isAvailable())
{
    //启用噪声压制功能
    suppressor.setEnabled(true);
}
```

BassBoost、Equalizer、PresetReverb、Visualizer 这 4 个类，都需要调用构造器来创建实例。创建实例时，同样需要传入一个 audioSession 参数，为了启用它们，同样需要调用 AudioEffect 基类的 setEnabled(true)方法。

4. BassBoost：重低音控制器

低音增强，用于增强或放大声音的低频。它与简单的均衡器相当，但仅限于低频范围内的一个频段放大。

获取 BassBoost 对象之后，可调用它的 setStrength(short strength)方法来设置重低音的强度。示例代码如下：

```
BassBoost bassBoost = new BassBoost(0,mediaPlayer.getAudioSessionId());
bassBoost.setEnabled(true);
if (bassBoost.getStrengthSupported()){
     bassBoost.setStrength((short) 100);
}
```

其中，getStrengthSupported()表示是否支持设置强度。如果此方法返回 false，则仅支持一种强度，并且 setStrength() 方法始终舍入到该值。

setStrength()设置效果的当前强度，强度的有效范围是[0, 1000]，0 表示最温和的效果，1000 表示最强的效果。

5. Equalizer：均衡控制器

Equalizer 提供了 getNumberOfPresets()方法获取系统所有预设的音场，并提供了 getPresetName()方法获取预设音场名称。获取 Equalizer 对象之后，可调用它的 getNumberOfBands()方法获取该均器支持的总频率数，再调用 getCenterFreq(short band)方法根据索引来获取频率。当用户想为某个频率的均衡器设置参数值时，可调用 setBandLevel(short band, short level)方法进行设置。示例代码如下：

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.test_cbr/*音频路径*/);
Equalizer equalizer = new Equalizer(0, mediaPlayer.getAudioSessionId());
equalizer.setEnabled(true);
//获取均衡器引擎支持的频段数
short bands = equalizer.getNumberOfBands();
//获取最大和最小增益
final short minEQLevel = equalizer.getBandLevelRange()[0];
final short maxEQLevel = equalizer.getBandLevelRange()[1];
for (short i = 0; i < bands; i++) {
     final short band = i;
     //获取当前频段的中心频率，分别为 60Hz,230Hz,910Hz,3600Hz,14000Hz
     int currentFreq = equalizer.getCenterFreq(band);
     //获取给定均衡器频段的增益
     short level = equalizer.getBandLevel(band);
     Log.d("Equalizer","currentFreq is: " + currentFreq + ", band level is: " + level);
     //为给定的均衡器频带设置增益值
     equalizer.setBandLevel(band,xx);
}
```

在构造函数 Equalizer(int priority, int audioSession)中，参数如下：

int priority：优先级，多个应用可以共享同一 Equalizer 引擎，该参数指出控制优先权，默认为 0。

int audioSession：音频会话 ID，系统范围内唯一，Equalizer 将被附加在拥有相同音频会话 ID 的 MediaPlayer 或 AudioTrack 上生效。

Android 系统预置了一些增益参数，可通过下面代码获取：

```
short presets = equalizer.getNumberOfPresets();
//获取系统预设的增益
for (short i = 0; i < presets; i++) {
    Log.d("presets",equalizer.getPresetName(i));
}
```

结果为 Normal、Classical、Dance、Flat、Folk、Heavy Metal、Hip hop、Jazz、Pop、Rock。

然后通过 equalizer.usePreset()；使用系统预置参数。

销毁时：

```
if (equalizer != null) {
     equalizer.setEnabled(false);
     equalizer.release();
     equalizer = null;
}
```

6. PresetReverb：预设音场控制器

PresetReverb 使用预设混响来配置全局混响，适合于音乐。预置的常见混响场景有：

PresetReverb.PRESET_LARGEHALL：适合整个管弦乐队的大型大厅。

PresetReverb.PRESET_LARGEROOM：适合现场表演的大型房间的混响预设。

获取 PresetReverb 对象之后，可调用它的 setPreset(short preset)方法设置使用预设置的音场。示例代码如下：

```
PresetReverb presetReverb = new PresetReverb(0,mediaPlayer.getAudioSessionId());
presetReverb.setEnabled(true);
presetReverb.setPreset(PresetReverb.PRESET_LARGEROOM);
```

7. Visualizer：示波器

Visualizer 对象并不用于控制音乐播放效果，它只是显示音乐的播放波形。为了实时显示该示波器的数据，需要为该组件设置一个 OnDataCaptureListener 监听器，该监听器将负责更新波形显示组件的界面。

5.1.3 使用 VideoView 控件播放视频

播放视频与播放音频相比，播放视频需要使用视觉控件将影像展示出来。Android 系统中的 VideoView 控件就是播放视频用的，借助它可以完成一个简易的视频播放器。

VideoView 控件提供了一些用于控制视频播放的方法，如表 5.2 所示。

表 5.2 VideoView 控件的常用方法

方　法	说　明
setVideoPath()	设置要播放的视频文件的位置
start()	开始或继续播放视频
pause()	暂停播放视频
resume()	重新开始播放视频

续表

方　法	说　明
seekTo()	从指定位置开始播放视频
isPlaying()	判断当前是否正在播放视频
getDuration()	获取载入的视频文件的时长

接下来讲解如何通过 VideoView 控件播放视频的过程，具体介绍如下。

1. 在布局文件中添加 VideoView 控件

如果想在界面上播放视频，则首先需要在布局文件中放置 1 个 VideoView 控件用于显示视频播放界面。在布局中添加 VideoView 控件的示例代码如下：

```
<VideoView
      android:id="@+id/videoview"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
```

2. 视频的播放

使用 VideoView 控件既可以播放本地存放的视频，也可以播放网络中的视频。示例代码如下：

```
VideoView videoView = (VideoView) findViewById(R.id.videoview);
videoView.setVideoPath("mnt/sdcard/xxx.avi");                    //播放本地视频
videoView.setVideoURI(Uri.parse("http://www.xxx.avi"));          //加载网络视频
videoView.start();                                               //播放视频
```

根据上述代码可知，播放本地视频时需要调用 VideoView 控件的 setVideoPath()方法，将本地视频地址传入该方法中即可。播放网络视频时需要调用 VideoView 控件的 setVideoURI()方法，通过调用 parse()方法将网络视频地址转换为 Uri 并传递到 setVideoURI()方法中。

需要注意的是，播放网络视频时需要在 AndroidManifest.xml 文件的< manifest >标签中添加访问网络的权限。示例代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

3. 为 VideoView 控件添加控制器

使用 VideoView 控件播放视频时，可以通过 setMediaController()方法为它添加一个控制器 MediaController，该控制器中包含媒体播放器（MediaPlayer）中的一些典型按钮，如播放/暂停(Play/Pause)、倒带(Rewind)、快进(Fast Forward)与进度滑动器(Progress Slider)等。VideoView 控件能够绑定媒体播放器，从而使播放状态和控件中显示的图像同步。示例代码如下：

```
MediaController controller = new MediaController(context);
videoView.setMediaController(controller);                        //为 VideoView 控件绑定控制器
```

具体的使用详见本章实例视频播放器的实现。

5.2 使用 MediaRecorder 类录制音频

手机一般都提供了麦克风硬件，而 Android 系统就可以利用该硬件来录制音频了。

为了在 Android 应用中录制音频，Android 提供了 MediaRecorder 类。使用 MediaRecorder 类录制音频的过程很简单，按如下步骤进行即可。

（1）创建 MediaRecorder 对象。

（2）调用 MediaRecorder 对象的 setAudioSource()方法设置声音来源，一般传入 MediaRecorder.AudioSource.MIC 参数指定录制来自麦克风的声音。

（3）调用 MediaRecorder 对象的 setOutputFormat()方法设置所录制的音频文件格式。

（4）调用 MediaRecorder 对象的 setAudioEncoder()、setAudioEncodingBitRate(int bitrate)、setAudioSamplingRate(int samplingRate)方法设置所录制的声音编码格式、编码位率、采样率等，这些参数将可以控制所录制的声音品质，文件大小。一般来说，声音品质越好，声音文件越大。

（5）调用 MediaRecorder 对象的 setOutputFile(String path)方法设置录制的音频文件的保存位置。

（6）调用 MediaRecorder 对象的 prepare()方法准备录制。

（7）调用 MediaRecorder 对象的 start()方法开始录制。

（8）录制完成，调用 MediaRecorder 对象的 stop()方法停止录制，并调用 release()方法释放资源。

注意，上面的步骤中第（3）、（4）步千万不能搞反，否则程序将会抛出 IllegalStateException 异常。

下面的程序示范了如何使用 MediaRecorder 类来录制声音，该程序的界面布局很简单，只提供了两个简单的按钮来控制录音开始、停止，故此处不再给出界面布局文件。程序代码如下：

```
public class RecordSound extends Activity
    implements OnClickListener
{
    //程序中的两个按钮
    ImageButton record , stop;
    //系统的音频文件
    File soundFile ;
    MediaRecorder mRecorder;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取程序界面中的两个按钮
        record = (ImageButton) findViewById(R.id.record);
        stop = (ImageButton) findViewById(R.id.stop);
```

```
        //为两个按钮的点击事件绑定监听器
        record.setOnClickListener(this);
        stop.setOnClickListener(this);
    }
    @Override
    public void onDestroy()
    {
        if (soundFile != null && soundFile.exists())
        {
            //停止录音
            mRecorder.stop();
            //释放资源
            mRecorder.release();
            mRecorder = null;
        }
        super.onDestroy();
    }
    @Override
    public void onClick(View source)
    {
        switch (source.getId())
        {
            //点击 record 按钮
            case R.id.record:
                if (!Environment.getExternalStorageState().equals(
                    android.os.Environment.MEDIA_MOUNTED))
                {
                    Toast.makeText(RecordSound.this
                        , "SD 卡不存在,请插入 SD 卡!"
                        , 5000)
                        .show();
                    return;
                }
                try
                {
                    //创建保存录音的音频文件
                    soundFile = new File(Environment
                        .getExternalStorageDirectory()
                        .getCanonicalFile() + "/sound.amr");
                    mRecorder = new MediaRecorder();
                    //设置录音的声音来源
                    mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                    //设置录制的声音的输出格式(必须在设置声音编码格式之前设置)
                    mRecorder.setOutputFormat(MediaRecorder
                        .OutputFormat.THREE_GPP);
                    //设置声音编码的格式
                    mRecorder.setAudioEncoder(MediaRecorder
                        .AudioEncoder.AMR_NB);
                    mRecorder.setOutputFile(soundFile.getAbsolutePath());
                    mRecorder.prepare();
                    //开始录音
                    mRecorder.start();              //①
                }
```

```
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                    break;
                //点击 stop 按钮
                case R.id.stop:
                    if (soundFile != null && soundFile.exists())
                    {
                        //停止录音
                        mRecorder.stop();          //②
                        //释放资源
                        mRecorder.release();       //③
                        mRecorder = null;
                    }
                    break;
            }
        }
    }
```

上面的程序中大段的粗体字代码用于设置录音的相关参数，例如输出文件的格式、声音来源等。上面的程序中①粗体字代码控制 MediaRecorder 类开始录音；当用户点击 stop 按钮时，程序中②号代码控制 MediaRecorder 类停止录音，③号粗体字代码用于释放资源。

录音完成后将可以看到/mnt/sdcard/目录下生成一个 sound.amr 文件，这就是录制的音频文件。Android 模拟器将会直接使用宿主机上的麦克风，因此如果读者的手机上有麦克风，那么该程序即可正常录制声音。

上面的程序需要使用系统的麦克风进行录音，因此需要向该程序授予录音的权限，也就是在 AndroidManifest.xml 文件中增加如下配置：

```
<!-- 授予该程序录制声音的权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

5.3 控制摄像头拍照

现在的手机一般都会提供相机功能，有些相机的镜头甚至支持 1000 万以上像素，有些甚至支持光学变焦，这些手机已经变成了专业数码相机。为了充分利用手机上的相机功能，Android 应用可以控制拍照和录制视频。

5.3.1 通过 Camera 进行拍照

Android 应用提供了 Camera 来控制拍照，使用 Camera 进行拍照也比较简单，按如下步骤进行即可。

（1）调用 Camera 的 open()方法打开相机。

（2）调用 Camera 的 getParameters()方法获取拍照参数。该方法返回一个 Camera.Parameters 对象。

（3）调用 Camera. Parameters 对象方法设置相机参数。

（4）调用 Camera 的 setParameters()方法，并将 Camera. Parameters 对象作为参数传入，这样即可对相机的拍照参数进行控制。

（5）调用 Camera 的 startPreview()方法开始预览取景，在预览取景之前需要调用 Camera 的 setPreviewDisplay(SurfaceHolder holder)方法设置使用哪个 SurfaceView 来显示取景图片。

（6）调用 Camera 的 takePicture()方法进行拍照。

（7）结束程序时，调用 Camera 的 stopPreview()方法结束取景预览，并调用 release()方法释放资源。

下面的程序示范了使用 Camera 来进行拍照，该程序的界面中只提供了一个 SurfaceView 组件来显示预览取景，十分简单。程序代码如下：

```
public class CaptureImage extends Activity
{
    SurfaceView sView;
    SurfaceHolder surfaceHolder;
    int screenWidth, screenHeight;
    //定义系统所用的照相机
    Camera camera;
    //是否在浏览中
    boolean isPreview = false;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        //设置全屏
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.main);
        WindowManager wm = (WindowManager) getSystemService(
            Context.WINDOW_SERVICE);
        Display display = wm.getDefaultDisplay();
        //获取屏幕的宽和高
        screenWidth = display.getWidth();
        screenHeight = display.getHeight();
        //获取界面中 SurfaceView 组件
        sView = (SurfaceView) findViewById(R.id.sView);
        //获得 SurfaceView 的 SurfaceHolder
        surfaceHolder = sView.getHolder();
        //为 surfaceHolder 添加一个回调监听器
        surfaceHolder.addCallback(new Callback()
        {
            @Override
            public void surfaceChanged(SurfaceHolder holder, int format, int width,
                int height)
            {
            }
            @Override
```

```
            public void surfaceCreated(SurfaceHolder holder)
            {
                //打开摄像头
                initCamera();
            }
            @Override
            public void surfaceDestroyed(SurfaceHolder holder)
            {
                //如果 camera 不为 null ,则释放摄像头
                if (camera != null)
                {
                    if (isPreview)
                        camera.stopPreview();
                    camera.release();
                    camera = null;
                }
            }
        });
        //设置该 SurfaceView 自己不维护缓冲
        surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    private void initCamera()
    {
        if (!isPreview)
        {
            camera = Camera.open();
        }
        if (camera != null && !isPreview)
        {
            try
            {
                Camera.Parameters parameters = camera.getParameters();
                //设置预览照片的大小
                parameters.setPreviewSize(screenWidth, screenHeight);
                //每秒显示 4 帧
                parameters.setPreviewFrameRate(4);
                //设置图片格式
                parameters.setPictureFormat(PixelFormat.JPEG);
                //设置 JPG 照片的质量
                parameters.set("jpeg-quality", 85);
                //设置照片的大小
                parameters.setPictureSize(screenWidth, screenHeight);
                camera.setParameters(parameters);
                //通过 SurfaceView 显示取景画面
                camera.setPreviewDisplay(surfaceHolder);          //①
                //开始预览
                camera.startPreview();                            //②
                //自动对焦
                camera.autoFocus(null);
            }
            catch (Exception e)
            {
```

```
                e.printStackTrace();
            }
            isPreview = true;
        }
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
            //当用户按照相键、中央键时执行拍照
            case KeyEvent.KEYCODE_DPAD_CENTER:
            case KeyEvent.KEYCODE_CAMERA:
                if (camera != null && event.getRepeatCount() == 0)
                {
                    //拍照
                    camera.takePicture(null, null , myjpegCallback);//③
                    return true;
                }
                break;
        }
        return super.onKeyDown(keyCode, event);
    }

    PictureCallback myjpegCallback = new PictureCallback()
    {
        @Override
        public void onPictureTaken(byte[] data, Camera camera)
        {
            //根据拍照所得的数据创建位图
            final Bitmap bm = BitmapFactory.decodeByteArray(data
                , 0, data.length);
            //加载/layout/save.xml 文件对应的布局资源
            View saveDialog = getLayoutInflater().inflate(
                R.layout.save, null);
            final EditText photoName = (EditText) saveDialog
                .findViewById(R.id.phone_name);
            //获取 saveDialog 对话框上的 ImageView 控件
            ImageView show = (ImageView) saveDialog.findViewById(R.id.show);
            //显示刚刚拍得的照片
            show.setImageBitmap(bm);
            //使用对话框显示 saveDialog 控件
            new AlertDialog.Builder(CaptureImage.this)
                .setView(saveDialog)
                .setPositiveButton("保存", new OnClickListener()
                {
                    @Override
                    public void onClick(DialogInterface dialog,
                        int which)
                    {
                        //创建一个位于 SD 卡上的文件
                        File file = new File(Environment.getExternalStorageDirectory()
```

```
                        , photoName.getText().toString() + ".jpg");
                FileOutputStream outStream = null;
                try
                {
                    //打开指定文件对应的输出流
                    outStream = new FileOutputStream(file);
                    //把位图输出到指定文件中
                    bm.compress(CompressFormat.JPEG, 100, outStream);
                    outStream.close();
                }
                catch (IOException e)
                {
                    e.printStackTrace();
                }
            }
        })
        .setNegativeButton("取消", null)
        .show();
    //重新浏览
    camera.stopPreview();
    camera.startPreview();
    isPreview = true;
}
};
}
```

上面的程序中大段粗体字代码用于设置相机的拍照参数，这些参数可以控制图片的品质和图片文件的大小。

上面的程序中①号粗体字代码设置使用指定的 SurfaceView 来显示取景预览图片，程序中②号粗体字代码则用于开始预览取景。当用户按下相机的拍照键或中央键时，程序的③号粗体字代码调用 takePicture()方法进行拍照。

调用 takePicture()方法进行拍照时传入了一个 PictureCallback 对象——当程序获取了拍照所得的图片数据之后，PictureCallback 对象将会被回调，该对象负责保存图片或将其上传到网络。

Android 模拟器不会使用宿主机上的摄像头作为相机镜头，因此取景预览将总是一片空白，这是因为模拟器没有摄像头的缘故。当用户进行拍照时，系统将会使用一张已有的图片作为图片。

用户在对话框中输入图片的名称后，程序将会把拍得的图片保存到 SD 卡上。

运行该程序需要获得相机拍照的权限，因此需要在 AndroidManifest.xml 文件中增加如下代码片段进行授权：

```
<!-- 授予程序使用摄像头的权限 -->
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
```

5.3.2 录制视频短片

MediaRecorder 类除了可用于录制音频之外，还可用于录制视频。使用 MediaRecorder

类录制视频与录制音频的步骤基本相同。只是录制视频时不仅需要采集声音，还需要采集图像。为了让 MediaRecorder 类录制时采集图像，应该在调用 setAudioSource(int audio_source)方法时再调用 setVideoSource(int video_source)方法来设置图像来源。

除此之外，还需在调用 setOutputFormat()方法设置输出文件格式之后进行如下步骤。

(1) 调用 MediaRecorder 对象的 setVideoEncoder()、setVideoEncodingBitRate(int bitRate)、setVideoFrameRate()方法设置所录制的视频的编码格式、编码位率、每秒多少帧等，这些参数将可以控制所录制的视频的品质、文件的大小。一般来说，视频品质越好，视频文件越大。

(2) 调用 MediaRecorder 对象的 setPreviewDisplay(Surface sv)方法设置使用哪个 SurfaceView 控件来显示视频预览。

剩下的代码则与录制音频的代码基本相同。下面的程序示范了如何录制视频，该程序的界面中提供了两个按钮用于控制开始和结束录制；除此之外，程序界面中还提供了一个 SurfaceView 控件来显示视频预览。该程序的界面布局文件如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
       xmlns:android = "http://schemas.android.com/apk/res/android"
       android:orientation = "vertical"
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       android:gravity = "center_horizontal"
       >
< LinearLayout
       android:orientation = "horizontal"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       android:gravity = "center_horizontal"
       >
< ImageButton
       android:id = "@ + id/record"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       android:src = "@drawable/record"
       />
< ImageButton
       android:id = "@ + id/stop"
       android:layout_width = "wrap_content"
       android:layout_height = "wrap_content"
       android:src = "@drawable/stop"
       />
</LinearLayout >
<!-- 显示视频预览的 SurfaceView -->
< SurfaceView
       android:id = "@ + id/sView"
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       />
</LinearLayout >
```

提供了上面所示的界面布局文件之后，接下来就可以在程序中使用 MediaRecorder 类来录制视频了。录制视频与录制音频的步骤基本相似，只是需要额外设置视频的图像来源、视频格式等，除此之外还需要设置使用 SurfaceView 控件显示视频预览。录制视频的程序代码如下：

```
public class RecordVideo extends Activity
    implements OnClickListener
{
    //程序中的两个按钮
    ImageButton record , stop;
    //系统的视频文件
    File videoFile ;
    MediaRecorder mRecorder;
    //显示视频预览的 SurfaceView
    SurfaceView sView;
    //记录是否正在进行录制
    private boolean isRecording = false;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取程序界面中的两个按钮
        record = (ImageButton) findViewById(R.id.record);
        stop = (ImageButton) findViewById(R.id.stop);
        //让 stop 按钮不可用
        stop.setEnabled(false);
        //为两个按钮的单击事件绑定监听器
        record.setOnClickListener(this);
        stop.setOnClickListener(this);
        //获取程序界面中的 SurfaceView 控件
        sView = (SurfaceView) this.findViewById(R.id.sView);
        //下面设置 Surface 不需要自己维护缓冲区
        sView.getHolder().setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        //设置分辨率
        sView.getHolder().setFixedSize(320, 280);
        //设置该组件让屏幕不会自动关闭
        sView.getHolder().setKeepScreenOn(true);
    }

    @Override
    public void onClick(View source)
    {
        switch (source.getId())
        {
            //点击 record 按钮
            case R.id.record:
                if (!Environment.getExternalStorageState().equals(
                    android.os.Environment.MEDIA_MOUNTED))
                {
                    Toast.makeText(RecordVideo.this
```

```
                        , "SD 卡不存在,请插入 SD 卡!"
                        , 5000)
                        .show();
                    return;
                }
                try
                {
                    //创建保存录制视频的视频文件
                    videoFile = new File(Environment
                        .getExternalStorageDirectory()
                        .getCanonicalFile() + "/myvideo.mp4");
                    //创建 MediaPlayer 对象
                    mRecorder = new MediaRecorder();
                    mRecorder.reset();
                    //设置从麦克风采集声音
                    mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                    //设置从摄像头采集图像
mRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
                    //设置视频文件的输出格式(必须在设置声音编码格式、
                    //图像编码格式之前设置)
                    mRecorder.setOutputFormat(MediaRecorder
                        .OutputFormat.MPEG_4);
                    //设置声音编码的格式
                    mRecorder.setAudioEncoder(MediaRecorder
                        .AudioEncoder.DEFAULT);
                    //设置图像编码的格式
                    mRecorder.setVideoEncoder(MediaRecorder
                        .VideoEncoder.MPEG_4_SP);
                    mRecorder.setVideoSize(320, 280);
                    //每秒 4 帧
                    mRecorder.setVideoFrameRate(4);
                    mRecorder.setOutputFile(videoFile.getAbsolutePath());
                    //指定使用 SurfaceView 来预览视频
                    mRecorder.setPreviewDisplay(sView.getHolder().get Surface());    //①
                    mRecorder.prepare();
                    //开始录制
                    mRecorder.start();
                    System.out.println(" --- recording --- ");
                    //让 record 按钮不可用
                    record.setEnabled(false);
                    //让 stop 按钮可用
                    stop.setEnabled(true);
                    isRecording = true;
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }
                break;
            //点击 stop 按钮
            case R.id.stop:
                //如果正在进行录制
                if (isRecording)
```

```
                {
                    //停止录制
                    mRecorder.stop();
                    //释放资源
                    mRecorder.release();
                    mRecorder = null;
                    //让 record 按钮可用
                    record.setEnabled(true);
                    //让 stop 按钮不可用
                    stop.setEnabled(false);
                }
                break;
            }
        }
    }
```

上面的程序中粗体字代码设置了视频所采集的图像来源，以及视频的压缩格式、视频分辨率等属性，程序的①号粗体字代码则用于设置使用 SurfaceView 控件显示指定视频预览。

运行该程序需要使用麦克风录制声音，需要使用摄像头采集图像，这些都需要授予相应的权限。不仅如此，由于该录制视频时视频文件增大得较快，可能需要使用外部存储器，因此需要对应用程序授予相应的权限。也就是需要在 AndroidManifest.xml 文件中增加如下授权配置：

```
<!-- 授予该程序录制声音的权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<!-- 授予该程序使用摄像头的权限 -->
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 授予使用外部存储器的权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

当在模拟器上运行该程序时，由于模拟器上没有摄像头硬件支持，因此程序无法采集视频所需的图像，所以建议在有摄像头硬件支持的真机上运行该程序。

5.4 应用实例：视频播放器

本章前面讲解了如何通过 VideoView 控件播放视频的相关知识，接下来我们以图 5.2 所示的界面为例，讲解如何通过 VideoView 控件实现一个视频播放器的案例，具体步骤如下。

1. 创建程序

创建一个名为 VideoView 的应用程序，包名指定为 cn.itcast.videoview。

2. 导入视频文件

选中 res 文件夹，在该文件夹中创建一个 raw 文件夹，将视频文件 video.mp4 放入 raw 文件夹中，如图 5.3 所示。

3. 放置界面控件

在 activity_main.xml 文件中，放置 1 个 ImageView 控件用于显示播放（暂停）按钮，1 个 VideoView 控件用于显示视频。完整布局代码如下：

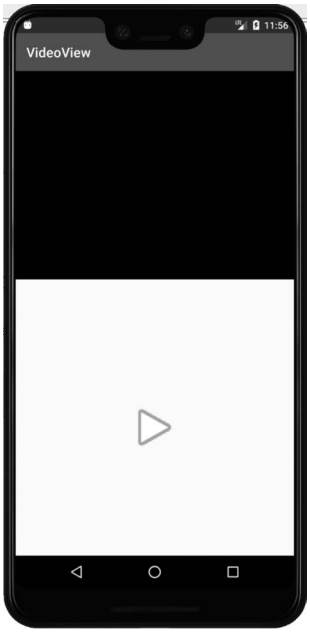


图 5.2　视频播放器界面

图 5.3　raw 文件夹

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      xmlns:tools = "http://schemas.android.com/tools"
      android:layout_width = "match_parent"
      android:layout_height = "match_parent"
      tools:context = ".MainActivity">
      < ImageView
            android:id = "@ + id/bt_play"
            android:layout_width = "80dp"
            android:layout_height = "80dp"
            android:layout_alignParentBottom = "true"
            android:layout_centerHorizontal = "true"
            android:layout_marginBottom = "150dp"
            android:src = "@android:drawable/ic_media_play" />
      < VideoView
            android:id = "@ + id/videoview"
            android:layout_width = "match_parent"
            android:layout_height = "match_parent" />
</RelativeLayout >
```

4. 编写界面交互代码

在 MainActivity 中创建一个 play()方法，在该方法中实现视频播放的功能。具体代码如下：

```
public class MainActivity extends AppCompatActivity implements
            View.OnClickListener {
      private VideoView videoView;
      private MediaController controller;
```

```
    ImageView iv_play;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        videoView = (VideoView) findViewById(R.id.videoview);
        iv_play = (ImageView) findViewById(R.id.bt_play);
        //拼出在资源文件夹下的视频文件路径字符串
        String url = "android.resource://" + getPackageName() + "/" + R.raw.video;
        //字符串解析成 Uri
        Uri uri = Uri.parse(url);
        //设置 VideoView 的播放资源
        videoView.setVideoURI(uri);
        //VideoView 绑定控制器
        controller = new MediaController(this);
        videoView.setMediaController(controller);
        iv_play.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.bt_play:
            play();
            break;
        }
    }
    //播放视频
    private void play() {
        if (videoView != null && videoView.isPlaying()) {
            iv_play.setImageResource(android.R.drawable.ic_media_play);
            videoView.stopPlayback();
            return;
        }
        videoView.start();
        iv_play.setImageResource(android.R.drawable.ic_media_pause);
        videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                iv_play.setImageResource(android.R.drawable.ic_media_play);
            }
        });
    }
}
```

上述代码中，通过 setVideoURI()方法将视频文件的路径加载到 VideoView 控件上，并通过 setMediaController()方法为 VideoView 控件绑定控制器，该控制器可以显示视频的播放、暂停、快进、快退和进度条等按钮。

重写了 onClick()方法，在该方法中调用 play()方法用于播放视频。

创建了一个 play()方法，在该方法中首先通过 isPlaying()方法判断当前视频是否正在播放，如果正在播放，则通过 setImageResource()方法设置播放按钮的图片为 ic_media_play.png，接着调用 stopPlayback()方法停止播放视频，否则，调用 start()方法播放视频，并

设置暂停按钮的图片为 ic_media_pause.png，接着通过 setOnCompletionListener()方法设置 VideoView 控件的监听器，当视频播放完时，会调用该监听器中的 onCompletion()方法，在该方法中将播放按钮的图片设置为 ic_media_play.png。

运行结果如图 5.4 和图 5.5 所示。

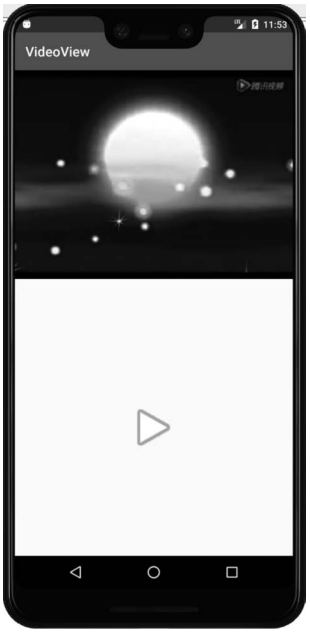


图 5.4　视频播放器运行结果 1

图 5.5　视频播放器运行结果 2

5.5　小结

本章主要讲解了音频、视频的播放过程以及音频的录制过程，同时介绍了使用手机的摄像头进行视频的录制过程。音频和视频都是非常重要的多媒体形式，Android 系统为音频、视频等多媒体的播放、录制提供了强大的支持。通过本章学习，需要重点掌握如何使用 MediaPlayer 类播放音频，如何使用 VideoView 控件播放视频。除此之外，还需要掌握通过 MediaRecorder 类录制音频的方法，以及控制摄像头拍照、录制视频的方法。通过本章知识的学习，希望读者能够开发一些跟音频和视频相关的软件。

习题 5

1. 简述使用 MediaPlayer 播放音频的步骤。

2. 简述 MediaRecorder 录制音频的步骤。

3. 请实现一个简单的音乐播放器，至少包含播放、暂停、上一曲、下一曲按钮以及显示播放时长的进度条。