Chapter 11 Playing Video

11.1 Introduction

We have discussed how to play audio in Chapter 9 using the class *MediaPlayer*. This class can also play video clips. In fact, the Android multimedia framework supports a wide variety of common media types to allow users to play audio or video from various sources, such as media files or data arriving from a network, and one can easily integrate audio, video and images into an application.

The class *MediaPlayer* is the main API for playing audio and video, while the class *AudioManager* manages audio sources and outputs on a device. However, unlike audio, which can be played in the background, we need to create a surface to play a video..

Like playing audio, to play video over a network, we need to request network access by adding the **uses-permission** statement in the manifest file *AndroidManifest.xml*:

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

While playing a video, if we do not want to keep the screen from dimming or the application process from sleeping, we also need to add the wake-lock statement in the manifest file:

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

11.2 A Simple Example

We present here a simple example of playing video in Android. We call the project and application *VideoDemo1*, and the package *video.videodemo1*. We shall write two Java files, *MainActivity.java* and *VideoViewActivity.java* to play a video, and two layout files, *activity_main.xml* and *videomain_main.xml* to define the screen layout. We also create the directory *res/raw* and put the video file to be played in the directory.

The following is the code of *MainActivity*. Its main task is to set up a button, which listens to any clicking event. When the button is clicked, a new *Intent* of the class *VideoViewActivity* is created, and a new activity of this *Intent* is started:

Program Listing 11-1 *MainActivity.java* of Project *VideoDemo1*

```
package video.videodemo1;

import android.view.View.OnClickListener;
import android.content.Intent;
import android.widget.Button;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends Activity
{
 Button button;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    // Get the layout from activity_main.xml
    setContentView(R.layout.activity_main);

    // Locate the button in activity_main.xml
    button = (Button) findViewById(R.id.streamButton);

    // Listen to button clicks
    button.setOnClickListener(new OnClickListener() {
     public void onClick(View arg0) {
      // Start New Activity.class
      Intent intent=new Intent(MainActivity.this,VideoViewActivity.class);
      startActivity ( intent );
     }
   });
  }
 }
-------------------------------------------------------------------------
```

When the activity *VideoViewActivity* is started, the class creates a *ProgressDialog* object, which is a dialog that shows a progress indicator with an optional text message or view, and progress range 0..10000. It then creates a *MediaController* with layout defined in *res/layout/videoview_main.xml* to play the specified video source:

Program Listing 11-2 *VideoViewActivity.java* of Project *VideoDemo1*

```
// VideoViewActivity.java
package video.videodemo1;

import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaPlayer;
import video.videodemo1.R;
import android.widget.*;
import android.app.*;
import android.net.Uri;
import android.util.Log;
import android.os.Bundle;

public class VideoViewActivity extends Activity
{
 ProgressDialog progressDialog;
 VideoView videoView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  // Get the layout from video_main.xml
  setContentView(R.layout.videoview_main);
  // Find your VideoView in video_main.xml layout
  videoView = (VideoView) findViewById(R.id.videoView);
  // Create a progressbar
  progressDialog = new ProgressDialog(VideoViewActivity.this);
  // Set progressbar title
  progressDialog.setTitle("Video Streaming");
```

```
   // Set progressbar message
   progressDialog.setMessage("Loading...");
   progressDialog.setIndeterminate ( false );
   progressDialog.setCancelable ( false );
   // Show progressbar
   progressDialog.show();

   try {
    // Create MediaController
    MediaController mediaController = new
                            MediaController(VideoViewActivity.this);
    mediaController.setAnchorView(videoView);
    //Note: omit extension here though the real filename has it
    Uri video=Uri.parse("android.resource://video.videodemo1/raw/android");
    videoView.setMediaController ( mediaController );
    videoView.setVideoURI(video);
   } catch (Exception e) {
    Log.e("Error", e.getMessage());
    e.printStackTrace();
   }
   videoView.requestFocus();
   videoView.setOnPreparedListener(new OnPreparedListener() {
    // Close the progress bar and play the video
    public void onPrepared(MediaPlayer mp) {
     progressDialog.dismiss();
     videoView.start();
    }
   });
  }
 }
-----------------------------------------------------------------------
```

In the example, we have hard-coded the video source, which is the 3gp video file *res/raw/android.3gp*. Note that we do not include the file extension ".3gp" in the specification. Even though the full file name is *android.3gp*, we just refer to it as *android*:

```
    Uri video = Uri.parse ("android.resource://video.videodemo1/raw/android");
```

Here the video file is in a local directory. If we want to play a video from a web site, we have to specify the full file name full URL path like the following example:

```
   String videoURL = "http://www.forejune.com/android/videos/android.3gp";
   Uri video = Uri.parse ( videoURL );
```

The main layout file *activity_main.xml* has nothing special. It simply defines a clicking button, similar to some of our previous examples:

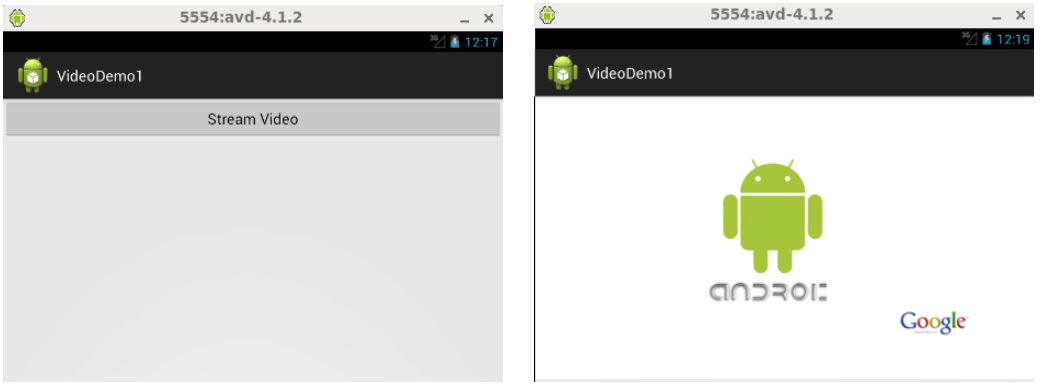
```
 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <Button
     android:id="@+id/streamButton"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/button" />
 </LinearLayout>
```

The other layout file, *videoview_main.xml* is just as simple. It defines where on the screen the video should be presented:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
  android:orientation="vertical" >
  <VideoView
       android:id="@+id/videoView"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
</LinearLayout>
```

When we run this application, it will first present a UI consisting of a button for the user to click on as shown in Figure 11-1(a). When the user clicks on the button, the video is loaded and played. Figure 11-1(b) shows a frame of the video. When the video is finished, the screen is still of the *VideoView*. We can press the **ESC** key to exit the screen, returning to the parent activity, which shows the clicking button of Figure 11-1(a). Clicking on the button will play the video again.



(a) (b)

Figure 11-1 UI and Output of *VideoDemo1*