Recording Multimedia

Multimedia recording is handled by the aptly named MediaRecorder class. To record audio or video, create a new Media Recorder object, as shown in the following code snippet:

```
MediaRecorder mediaRecorder = new MediaRecorder();
```

Before you can record any media in Android, your application needs the RECORD_AUDIO and / or RECORD_VIDEO permissions. Add uses-permission tags for each of them, as appropriate, in your application manifest.

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

> **The ability to record video has been restricted for the version 1.0 release of Android; however, Audio recording is still available.**

The Media Recorder can be used to confi gure the video and audio sources (generally the camera and microphone), output format, video size and frame rate, and the video and audio encoders to use.

The following code snippet shows how to confi gure a Media Recorder to record audio from the microphone using the default format and encoder:

The emulator supports recording of audio using the microphone device attached to your development platform.

```
// Set the audio source.
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
// Set the output format.
mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
// Set the audio encoders to use.
mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
```

Once you've defi ned your input source and output format, assign a file to store the recorded media using the setOutputFile method as shown below:

```
mediaRecorder.setOutputFile("myoutputfile.mp4");
```

> **The** setOutputFile **method must be called before** prepare **and after** setOutputFormat **or it will throw an Illegal State Exception.**

To begin recording, call prepare followed by the start method, as shown below:

```
mediaRecorder.prepare();
mediaRecorder.start();
```

When you're fi nished, call stop to end the playback, followed by release to free the Media Recorder resources:

```
mediaRecorder.stop();
mediaRecorder.release();
```

When recording video, it's generally considered good practice to display a preview of the recorded video in real time. Using the setPreviewDisplay *method, you can assign a* Surface *to display the video preview.*

As with any other resource, media fi les created by your application will be unavailable to others. As a result, it's good practice to use the Media Store Content Provider to assign metadata, select a fi le location, and publish the recorded media to share recordings with other applications.

To do that, after recording new media create a new ContentValues object to add a new record to the Media Store. The metadata you specify here can include the details including the title, time stamp, and geocoding information for your new media fi le, as shown in the code snippet below:

```
ContentValues content = new ContentValues(3);
content.put(Audio.AudioColumns.TITLE, “TheSoundandtheFury”);
content.put(Audio.AudioColumns.DATE_ADDED,
System.currentTimeMillis() / 1000);
content.put(Audio.Media.MIME_TYPE, “audio/amr”);
```

You must also specify the absolute path of the media fi le being added:

```
content.put(MediaStore.Audio.Media.DATA,
“myoutputfile.mp4”);
```

Get access to the application’s ContentResolver, and use it to insert this new row into the Media Store as shown in the following code snippet:

```
ContentResolver resolver = getContentResolver();
Uri uri = resolver.insert(Audio.Media.EXTERNAL_CONTENT_URI, content);
```

Once the media fi le has been inserted into the media store you should announce it’s availability using a broadcast Intent as shown below:

```
sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, uri));
```