GRAPHICS AND MULTIMEDIA

• The only modern technology that can compete with mobile phones for ubiquity is the portable digital Media player. As a result, the multimedia capabilities of portable devices are a significant consideration for many consumers.

• Android‘s open platform- and provider-agnostic philosophy ensures that it offers a multimedia library capable of playing and recording a wide range of media formats, both locally and streamed.

• Android exposes this library to your applications, providing comprehensive multimedia functionality including recording and playback of audio, video, and stillimage media stored locally, within an application, or streamed over a data connection. At the time of print, Android supported the following multimedia formats:

a) JPEG

b) PNG

c) OGG

d) Mpeg 4

e) 3GP

f) MP3

g) Bitmap

PLAYING MEDIA RESOURCES

Multimedia playback in Android is handled by the MediaPlayer class. You can play back media stored as application resources, local files, or from a network URI. To play a media resource, create a new Media Player instance, and assign it a media source to play using the setDataSource method. Before you can start playback, you need to call prepare, as shown in the following code snippet:

```
StringMEDIA_FILE_PATH= Settings.System.DEFAULT_RINGTONE_URI.toString();

MediaPlayer mpFile = new MediaPlayer();

try {

mpFile.setDataSource(MEDIA_FILE_PATH);

mpFile.prepare();

mpFile.start();

}

catch (IllegalArgumentException e)
```

{} catch (IllegalStateException e) {}

catch (IOException e) {}

Alternatively, the static create methods work as shortcuts, accepting media resources as a parameter and preparing them for playback, as shown in the following example,

which plays back an application resource:

MediaPlayer mpRes = MediaPlayer.create(context, R.raw.my_sound);

Note that if you use a create method to generate your MediaPlayer object, prepare is called for you. Once a Media Player is prepared, call start as shown below to begin playback of the

associated media resource.

mpRes.start();

mpFile.start();

The Android Emulator simulates audio playback using the audio output of your development platform.

The Media Player includes stop, pause, and seek methods to control playback, as well as methods to find the duration, position, and image size of the associated media. To loop or repeat playback, use the setLooping method. When playing video resources, getFrame will take a screen grab of video media at the specified frame and return a bitmap resource.

Once you‘ve finished with the Media Player, be sure to call release to free the associated

resources, as shown below:

mpRes.release();

mpFile.release();

Since Android only supports a limited number of simultaneous Media Player objects, not

releasing them can cause runtime exceptions.

Using the Camera

The popularity of digital cameras (particularly within phone handsets) has caused their prices to drop just as their size has shrunk dramatically. It‘s now becoming difficult to even find a mobile phone without a camera, and Android devices are unlikely to be exceptions.

To access the camera hardware, you need to add the CAMERA permission to your application manifest, as shown here:

<uses-permission android:name=”android.permission.CAMERA”/>

This grants access to the Camera Service. The Camera class lets you adjust camera settings, take pictures, and manipulate streaming camera previews. To access the Camera Service, use the static open method on the Camera class. When your application has finished with the camera, remember to relinquish your hold on the Service by calling release following the simple use pattern shown in the code snippet below:

```
Camera camera =
Camera.open(); [ ... Do things with the camera ... ]
camera.release();
```