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();
```

Controlling Camera Settings

The current camera settings are available as a Camera.Parameters object. Call the getParameters method on the Camera to access the current parameters.

You can use the set* methods on the returned Parameters to modify the settings. To apply changes, call setParameters, passing in the modifi ed values as shown below:

```
Camera.Parameters parameters = camera.getParameters();
parameters.setPictureFormat(PixelFormat.JPEG);
camera.setParameters(parameters);
```

The Camera Parameters can be used to specify the image and preview size, image format, and preview frame rate.

Using the Camera Preview

Access to the camera's streaming video means that you can incorporate live video into your applications. Some of the most exciting early Android applications have used this functionality as the basis for augmenting reality.

The camera preview can be displayed in real time onto a Surface, as shown in the code snippet below:

```
camera.setPreviewDisplay(mySurface);
camera.startPreview();
[ … ]
camera.stopPreview();
```

You'll learn more about Surfaces in the following chapter, although Android includes an excellent example of using a SurfaceView to display the camera preview in real time. This example is available in the graphics/CameraPreview project in the SDK API demos.

You can also assign a PreviewCallback to be fi red for each preview frame, allowing you to manipulate or display each preview frame individually. Call the setPreviewCallback method on the Camera object, passing in a new PreviewCallback implementation overriding the onPreviewFrame method as shown here:

```
camera.setPreviewCallback(new PreviewCallback() {
public void onPreviewFrame(byte[] _data, Camera _camera) {
// TODO Do something with the preview image.
}
});
```

Taking a Picture

Take a picture by calling takePicture on a Camera object, passing in a ShutterCallback and PictureCallback implementations for the RAW and JPEG-encoded images. Each picture callback will receive a byte array representing the image in the appropriate format, while the shutter callback is triggered immediately after the shutter is closed.

```
private void takePicture() {
camera.takePicture(shutterCallback, rawCallback, jpegCallback);
}
ShutterCallback shutterCallback = new ShutterCallback() {

public void onShutter() {
// TODO Do something when the shutter closes.
}
};
PictureCallback rawCallback = new PictureCallback() {
public void onPictureTaken(byte[] _data, Camera _camera) {
// TODO Do something with the image RAW data.
}
};
PictureCallback jpegCallback = new PictureCallback() {
public void onPictureTaken(byte[] _data, Camera _camera) {
// TODO Do something with the image JPEG data.
}
};
```