Camera

Introduction

- In order to access the camera hardware we first must acquire the required permission.

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

The Camera Activity

- The camera activity has the following intent filter specified in its manifest file.

```
<intent-filter>
<action android:name="android.media.action.IMAGE_CAPTURE" />
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

The Camera Activity

- We can start the camera activity using an intent instantiated accordingly.

```
Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
startActivity(i);
```

Part 1/3

The Camera Activity

```
package com.abelski.samples;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.widget.ImageView;

public class MyCameraActivity extends Activity
{
    final static int REQUEST_CODE = 0;
    ImageView image;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Intent i = new Intent(
            android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(i, REQUEST_CODE);
    }
```

The Camera Activity

```
    protected void onActivityResult(
        int requestCode,
        int resultCode,
        Intent intent)
    {
        super.onActivityResult(requestCode, resultCode, intent);
        if (resultCode == RESULT_OK)
        {
            Bundle extras = intent.getExtras();
            Bitmap data = (Bitmap) extras.get("data");
            image = (ImageView) findViewById(R.id.img);
            image.setImageBitmap(data);
        }
    }
}
```

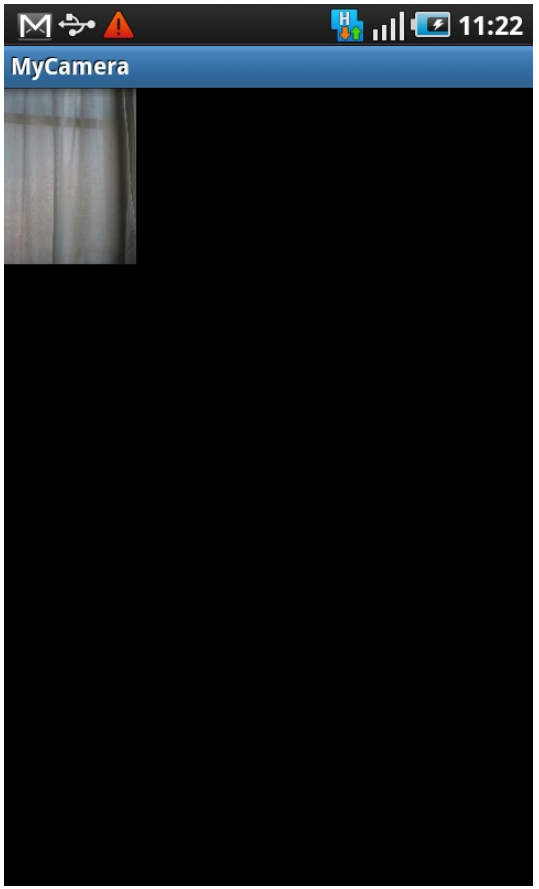
The Camera Activity

```
<?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" >

<ImageView android:id="@+id/img" android:layout_width="wrap_content"
android:layout_height="wrap_content">
</ImageView>

</LinearLayout>
```

The Camera Activity



The Camera Activity

- The camera activity does not return the full-size image back to the calling activity. Doing so requires lots of memory. The camera activity returns a small thumbnail image.
- We can overcome this behavior by passing over the `android.provider.MediaStore.EXTRA_OUTPUT` constant together with the Uri object that refers the file we want to be created on the SD card to hold the image data.

The Camera Activity

```
package com.abelski.samples;

import java.io.File;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.Display;
import android.widget.ImageView;

public class MyCameraActivity extends Activity
{
    final static int REQUEST_CODE = 0;
    ImageView image;
    String path;
```

Part 2/3

Part 3/3

The Camera Activity

```
public void onCreate(Bundle savedInstanceState)
{
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     path = Environment.getExternalStorageDirectory().getAbsolutePath()
          +"/mypix.jpg";
     Uri uri = Uri.fromFile(new File(path));
     Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
     i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, uri);
     startActivityForResult(i, REQUEST_CODE);
}

protected void onActivityResult(
     int requestCode,
     int resultCode,
     Intent intent)
{
     super.onActivityResult(requestCode, resultCode, intent);
     if (resultCode == RESULT_OK)
     {
          //getting the image view widget
          image = (ImageView) findViewById(R.id.img);
          //the android device display
          Display currentDisplay = getWindowManager().getDefaultDisplay();
          int displayWidth = currentDisplay.getWidth();
          int displayHeight = currentDisplay.getHeight();
```

The Camera Activity

```
                //getting image size
                BitmapFactory.Options options = new BitmapFactory.Options();
                options.inJustDecodeBounds = true;
                Bitmap data = BitmapFactory.decodeFile(path,options);
                int imageHeight = options.outHeight;
                int imageWidth = options.outWidth;
                //scaling the image (if needed)
                if(imageHeight>displayHeight || imageWidth>displayWidth)
                {
                     if(imageHeight/displayHeight>imageWidth/displayWidth)
                     {
                          options.inSampleSize = imageHeight/displayHeight;
                     }
                     else
                     {
                          options.inSampleSize = imageWidth / displayWidth;
                     }
                }
                options.inJustDecodeBounds = false;
                data = BitmapFactory.decodeFile(path,options);
                image.setImageBitmap(data);
           }
      }
}
```

The Camera Activity

The `Camera` Class

- The `Camera` class lets you adjust camera settings, take pictures and manipulate streaming camera previews.

```
...
Camera myCam = Camera.Open();
...
```

Auto Focus

- We can easily cancel or enable the auto focus feature. When enabling the auto focus we should pass over a reference for a `Camera.AutoFocusCallBack` object.

```
...
myCam.cancelAutoFocus();
...
Camera.AutoFocusCallBack ob = new MyAutuFocusCallBack();
myCam.autoFocus(ob);
...
```

Auto Focus

- We should define a new class that implements the `Camera.AutoFocusCallBack` interface. This interface includes one method only.
- This interface includes the `onAutoFocus()` method that will be called when the camera is focused .

Auto Focus

```
class MyAutoFocusCallback implements Camera.AutoFocusCallBack
{
    public void onAutoFocus(boolean success, Camera camera)
    {
        //sounds a short blip to indicate the camera is focused
    }
}
```

Picture Parameters

- Calling the `getParameters()` method on our `Camera` object will return a `Camera.Parameters` object on which we can call various methods in order to get more information about our camera.

```
...
Camera.Parameters params = myCam.getParameters();
...
String flashState = params.getFlashMode();
String focusState = params.getFocusMode();
int jpgQuality = params.getJpegQuality();
...
```

Picture Parameters

- Calling the `setParameters()` method on our `Camera` object we can set a new customized version of the `Camera.Parameters` object we received when calling the `getParameters()` method.

```
...
Camera.Parameters params = myCam.getParameters();
...
params.setColorEffect(...);
params.setFlashMode(...);
...
myCam.setParameters(params);
...
```

Scene Modes

- We can easily set and get the scene mode using the following two methods:

```
...
params.setSceneMode(SCENE_MODE_SUNSET);
String sceneMode = params.getSceneMode();
...
```

- Once the scene mode was set we can easily set the new updated `Camera.Parameters` object into our `Camera` object.

```
...
myCam.setParameters(params);
...
```

Scene Modes

- The available scene modes include the following constants that were declared within the `Camera.Parameters` class:

SCENE_MODE_ACTION

SCENE_MODE_AUTO

SCENE_MODE_BEACH

SCENE_MODE_CANDLELIGHT

SCENE_MODE_FIREWORKS

SCENE_MODE_LANDSCAPE

SCENE_MODE_NIGHT

SCENE_MODE_NIGHT_PORTRAIT

SCENE_MODE_PARTY

Scene Modes

SCENE_MODE_PORTRAIT

SCENE_MODE_SNOW

SCENE_MODE_SPORTS

SCENE_MODE_STEADYPHOTO

SCENE_MODE_SUNSET

SCENE_MODE_THEATRE