在android中应用相机功能，一般有两种：一种是直接调用系统相机，一种自己写的相机。  
我将分别演示两种方式的使用：  
  
第一种：是使用Intent跳转到系统相机,action为:android.media.action.STILL\_IMAGE\_CAMERA  
关键代码：

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. Intent intent = **new** Intent(); //调用照相机
2. intent.setAction("android.media.action.STILL\_IMAGE\_CAMERA");
3. startActivity(intent);

例子:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **import** android.app.Activity;
2. **import** android.content.Intent;
3. **import** android.os.Bundle;
4. **public** **class** CameraTest\_2 **extends** Activity {
5. /\*\* Called when the activity is first created. \*/
6. @Override
7. **public** **void** onCreate(Bundle savedInstanceState) {
8. **super**.onCreate(savedInstanceState);
9. setContentView(R.layout.main);
10. Intent intent = **new** Intent(); //调用照相机
11. intent.setAction("android.media.action.STILL\_IMAGE\_CAMERA");
12. startActivity(intent);
13. }
14. }

想要测试的，可以直接新建一个项目，并且把主activity的代码换成上面的，然后运行，我测试了一下，上面这个代码并不  
需要权限，毕竟只是调用系统自带的程序。  
当然网上还有一些其他相关的调用方法，只要设置对了action,那么系统就会调用系统自带的相机.

第二种:  
（1）首先我们要自己创建一个照相界面，必须考虑用什么控件显示照相机中的预览效果，显然android已经帮我们做好了选择，那就是SurfaceView

控制SurfaceView需要一个surfaceHolder,他是系统提供的一个用来设置SurfaceView的一个对象，可以通过surfaceView.getHolder()这个方法来获得.

Camera提供一个setPreviewDisplay(SurfaceHolder)的方法来连接surfaceHolder,并通过他来控制surfaceView.

而我们则使用android的Camera类提供了startPreview()和stopPreview()来开启和关闭预览.  
关系如下：  
Camera -- -->SurfaceHolder------>SurfaceView.

（2）知道怎么预览了，当然也要知道怎么开启相机.Camera.open()这是个静态方法，如果相机没有别人用着，则会返回一个 相机引用，如果被人用着，则会抛出异常。很奇怪的是，这个方法，不能随便放，如放在构造方法或者onCreate()方法中，都会照成没有预览效果.

（3）  
SurfaceHolder.Callback,这是个holder用来显示surfaceView 数据的接口,他分别必须实现3个方法  
surfaceCreated()这个方法是surface 被创建后调用的  
surfaceChanged()这个方法是当surfaceView发生改变后调用的  
surfaceDestroyed()这个是当surfaceView销毁时调用的.  
surfaceHolde通过addCallBack()方法将响应的接口绑定到他身上.  
surfaceHolder还必须设定一个setType()方法，查看api的时候，发现这个方法已经过时，但是不写，又会报错

（4）  
我用以上知识写了一个MySurfaceView类，他继承于SurfaceView,并在里面实现了照相机的预览功能.这个我觉得最简单的照相机预览代码:  
MySurfaceView.java:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **import** java.io.IOException;
2. **import** android.content.Context;
3. **import** android.graphics.PixelFormat;
4. **import** android.hardware.Camera;
5. **import** android.util.Log;
6. **import** android.view.SurfaceHolder;
7. **import** android.view.SurfaceView;
8. **public** **class** MySurfaceView **extends** SurfaceView **implements** SurfaceHolder.Callback{
9. SurfaceHolder holder;
10. Camera myCamera;
11. **public** MySurfaceView(Context context)
12. {
13. **super**(context);
14. holder = getHolder();//获得surfaceHolder引用
15. holder.addCallback(**this**);
16. holder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);//设置类型
17. }
18. @Override
19. **public** **void** surfaceCreated(SurfaceHolder holder) {
20. // TODO Auto-generated method stub
21. **if**(myCamera == **null**)
22. {
23. myCamera = Camera.open();//开启相机,不能放在构造函数中，不然不会显示画面.
24. **try** {
25. myCamera.setPreviewDisplay(holder);
26. } **catch** (IOException e) {
27. // TODO Auto-generated catch block
28. e.printStackTrace();
29. }
30. }
31. }
32. @Override
33. **public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width,
34. **int** height) {
35. myCamera.startPreview();
36. }
37. @Override
38. **public** **void** surfaceDestroyed(SurfaceHolder holder) {
39. // TODO Auto-generated method stub
40. myCamera.stopPreview();//停止预览
41. myCamera.release();//释放相机资源
42. myCamera = **null**;
43. Log.d("ddd", "4");
44. }
45. }

主activity代码:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **import** android.app.Activity;
2. **import** android.os.Bundle;
3. **import** android.view.View;
4. **import** android.view.View.OnClickListener;
5. **public** **class** CameraTest\_3 **extends** Activity  {
6. /\*\* Called when the activity is first created. \*/
7. MySurfaceView mySurface;
8. @Override
9. **public** **void** onCreate(Bundle savedInstanceState) {
10. **super**.onCreate(savedInstanceState);
11. mySurface = **new** MySurfaceView(**this**);
12. setContentView(mySurface);
13. }
14. }

而且必须给应用添加权限:

**[html]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

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

(5)能够预览了，接下来就是拍照了，拍照用到了一个camera.tackPiture()这个方法,这个方法，有三个参数分别是  
ShutterCallBack shutter,PictureCallBack raw,PictureCallBack jpeg.  
下面是对他们的实现

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **private** ShutterCallback shutter = **new** ShutterCallback() {
3. @Override
4. **public** **void** onShutter() {
5. // TODO Auto-generated method stub
6. Log.d("ddd", "shutter");
7. }
8. };
9. **private** PictureCallback raw = **new** PictureCallback() {
11. @Override
12. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
13. // TODO Auto-generated method stub
14. Log.d("ddd", "raw");
15. }
16. };
17. **private** PictureCallback jpeg = **new** PictureCallback() {
19. @Override
20. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
21. // TODO Auto-generated method stub
22. Log.d("ddd","jpeg");
23. }
24. };

当开始拍照时，会依次调用shutter的onShutter()方法，raw的onPictureTaken方法,jpeg的onPictureTaken方法.  
三个参数的作用是shutter--拍照瞬间调用，raw--获得没有压缩过的图片数据，jpeg---返回jpeg的图片数据  
当你不需要对照片进行处理，可以直接用null代替.  
注意，当调用camera.takePiture方法后，camera关闭了预览，这时需要调用startPreview()来重新开启预览。

我用以上知识，加到上面的那个例子，就形成了下面的代码：  
MySurfaceView.java:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **package** jason.camera;
2. **import** java.io.IOException;
3. **import** android.content.Context;
4. **import** android.graphics.PixelFormat;
5. **import** android.hardware.Camera;
6. **import** android.hardware.Camera.PictureCallback;
7. **import** android.hardware.Camera.ShutterCallback;
8. **import** android.util.Log;
9. **import** android.view.SurfaceHolder;
10. **import** android.view.SurfaceView;
11. **public** **class** MySurfaceView **extends** SurfaceView **implements** SurfaceHolder.Callback{
12. SurfaceHolder holder;
13. Camera myCamera;
14. **private** ShutterCallback shutter = **new** ShutterCallback() {
16. @Override
17. **public** **void** onShutter() {
18. // TODO Auto-generated method stub
19. Log.d("ddd", "shutter");
21. }
22. };
23. **private** PictureCallback raw = **new** PictureCallback() {
25. @Override
26. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
27. // TODO Auto-generated method stub
28. Log.d("ddd", "raw");
30. }
31. };
32. **private** PictureCallback jpeg = **new** PictureCallback() {
34. @Override
35. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
36. // TODO Auto-generated method stub
37. Log.d("ddd","jpeg");
39. }
40. };
41. **public** MySurfaceView(Context context)
42. {
43. **super**(context);
44. holder = getHolder();//获得surfaceHolder引用
45. holder.addCallback(**this**);
46. holder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);//设置类型
47. }
48. **public** **void** tackPicture()
49. {
50. myCamera.takePicture(**null**,**null**,**null**);
51. }
52. **public** **void** voerTack()
53. {
54. myCamera.startPreview();
55. }
56. @Override
57. **public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width,
58. **int** height) {
59. myCamera.startPreview();
60. }
61. @Override
62. **public** **void** surfaceCreated(SurfaceHolder holder) {
63. // TODO Auto-generated method stub
64. **if**(myCamera == **null**)
65. {
66. myCamera = Camera.open();//开启相机,不能放在构造函数中，不然不会显示画面.
67. **try** {
68. myCamera.setPreviewDisplay(holder);
69. } **catch** (IOException e) {
70. // TODO Auto-generated catch block
71. e.printStackTrace();
72. }
73. }
74. }
75. @Override
76. **public** **void** surfaceDestroyed(SurfaceHolder holder) {
77. // TODO Auto-generated method stub
78. myCamera.stopPreview();//停止预览
79. myCamera.release();//释放相机资源
80. myCamera = **null**;
82. }
83. }

主activity:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **import** android.app.Activity;
2. **import** android.os.Bundle;
3. **import** android.view.View;
4. **import** android.view.View.OnClickListener;
5. **public** **class** CameraTest\_3 **extends** Activity **implements** OnClickListener  {
6. /\*\* Called when the activity is first created. \*/
7. MySurfaceView mySurface;
8. **boolean** isClicked = **false**;
9. @Override
10. **public** **void** onCreate(Bundle savedInstanceState) {
11. **super**.onCreate(savedInstanceState);
12. mySurface = **new** MySurfaceView(**this**);
13. setContentView(mySurface);
14. mySurface.setOnClickListener(**this**);
15. }
16. @Override
17. **public** **void** onClick(View v) {
18. // TODO Auto-generated method stub
19. **if**(!isClicked)
20. {
21. mySurface.tackPicture();
22. isClicked = **true**;
23. }**else**
24. {
25. mySurface.voerTack();
26. isClicked = **false**;
28. }
30. }

这样就是实现了拍照的功能，那么怎样要图片保存呢？那么这是就需要在那个参数中的jpeg的  
方法里面进行处理了，那个方法的data参数，就是相片的数据。  
我们通过BitmapFactory.decodeByteArray(data, 0, data.length)来获得图片并通过io处理，将图片保存到想要保存的位置  
下面这段代码，是将照片保存到/sdcard/wjh.jpg;并把一些没有用到的代码全部删掉，剩下一些必须的代码

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **package** java.camera;
2. **import** java.io.BufferedInputStream;
3. **import** java.io.BufferedOutputStream;
4. **import** java.io.File;
5. **import** java.io.FileOutputStream;
6. **import** java.io.IOException;
7. **import** java.io.OutputStream;
8. **import** android.content.Context;
9. **import** android.graphics.Bitmap;
10. **import** android.graphics.BitmapFactory;
11. **import** android.graphics.PixelFormat;
12. **import** android.hardware.Camera;
13. **import** android.hardware.Camera.PictureCallback;
14. **import** android.hardware.Camera.ShutterCallback;
15. **import** android.util.Log;
16. **import** android.view.SurfaceHolder;
17. **import** android.view.SurfaceView;
18. **public** **class** MySurfaceView **extends** SurfaceView **implements** SurfaceHolder.Callback{
19. SurfaceHolder holder;
20. Camera myCamera;
21. **private** PictureCallback jpeg = **new** PictureCallback() {
23. @Override
24. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
25. // TODO Auto-generated method stub
26. **try**
27. {
28. Bitmap bm = BitmapFactory.decodeByteArray(data, 0, data.length);
29. File file = **new** File("/sdcard/wjh.jpg");
30. BufferedOutputStream bos
31. = **new** BufferedOutputStream(**new** FileOutputStream(file));
32. bm.compress(Bitmap.CompressFormat.JPEG,100,bos);
33. bos.flush();
34. bos.close();
35. }**catch**(Exception e)
36. {
37. e.printStackTrace();
38. }
39. }
40. };
41. **public** MySurfaceView(Context context)
42. {
43. **super**(context);
44. holder = getHolder();//获得surfaceHolder引用
45. holder.addCallback(**this**);
46. holder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);//设置类型
47. }
48. **public** **void** tackPicture()
49. {
50. myCamera.takePicture(**null**,**null**,jpeg);
51. }
52. **public** **void** voerTack()
53. {
54. myCamera.startPreview();
55. }
56. @Override
57. **public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width,
58. **int** height) {
59. myCamera.startPreview();
60. }
61. @Override
62. **public** **void** surfaceCreated(SurfaceHolder holder) {
63. // TODO Auto-generated method stub
64. **if**(myCamera == **null**)
65. {
66. myCamera = Camera.open();//开启相机,不能放在构造函数中，不然不会显示画面.
67. **try** {
68. myCamera.setPreviewDisplay(holder);
69. } **catch** (IOException e) {
70. // TODO Auto-generated catch block
71. e.printStackTrace();
72. }
73. }
74. }
75. @Override
76. **public** **void** surfaceDestroyed(SurfaceHolder holder) {
77. // TODO Auto-generated method stub
78. myCamera.stopPreview();//停止预览
79. myCamera.release();//释放相机资源
80. myCamera = **null**;
81. }
82. }

注意，这是必须添加在sd卡上写数据的权限

**[html]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **<uses-permission** android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"**/>**

(7)能够拍照了，这下子要考虑如何让图片更好看了，这显然是专业人士的强项，但是我们在程序上，也可以做一些处理，  
向上面的那些，因为我直接把surfaceView当做整体布局，就可能出现屏幕被拉开了，不是很好看，所以这时，就可以不要把  
surfaceView弄成整体布局，把他弄到一个布局管理器，再设置相关的参数.  
这时需要注意的是有些参数不能随便乱设  
如以下代码:

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. Camera.Parameters parames = myCamera.getParameters();//获得参数对象
2. parames.setPictureFormat(PixelFormat.JPEG);//设置图片格式
3. parames.setPreviewSize(640,480);//这里面的参数只能是几个特定的参数，否则会报错.(176\*144,320\*240,352\*288,480\*360,640\*480)
4. myCamera.setParameters(parames);

还有自动对焦，当然有些手机没有这个功能，自动对焦是通过autoFocus()这个方法调用一个自动对焦的接口，并在里面进行处理。  
注意，这个方法必须在startPreview()和stopPreview()中间。  
AutoFocusCallback是自动对焦的接口，实现它必须实现public void onAutoFocus(boolean success, Camera camera)这个方法，  
所以我们可以将拍照方法放在这里面，然后对焦后再进行拍摄。。效果会好很多。  
注意自动对焦需要添加

**[html]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **<uses-feature** android:name="android.hardware.camera.autofocus" **/>**

下面是主要代码,直接贴出来了

**[java]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **import** java.io.BufferedOutputStream;
2. **import** java.io.File;
3. **import** java.io.FileOutputStream;
4. **import** java.io.IOException;
5. **import** android.app.Activity;
6. **import** android.content.pm.ActivityInfo;
7. **import** android.graphics.Bitmap;
8. **import** android.graphics.BitmapFactory;
9. **import** android.graphics.PixelFormat;
10. **import** android.hardware.Camera;
11. **import** android.hardware.Camera.AutoFocusCallback;
12. **import** android.hardware.Camera.PictureCallback;
13. **import** android.os.Bundle;
14. **import** android.view.SurfaceHolder;
15. **import** android.view.SurfaceView;
16. **import** android.view.View;
17. **import** android.view.Window;
18. **import** android.view.SurfaceHolder.Callback;
19. **import** android.view.View.OnClickListener;
20. **public** **class** CameraTest\_4 **extends** Activity **implements**
21. Callback, OnClickListener, AutoFocusCallback{
22. SurfaceView mySurfaceView;//surfaceView声明
23. SurfaceHolder holder;//surfaceHolder声明
24. Camera myCamera;//相机声明
25. String filePath="/sdcard/wjh.jpg";//照片保存路径
26. **boolean** isClicked = **false**;//是否点击标识
27. //创建jpeg图片回调数据对象
28. PictureCallback jpeg = **new** PictureCallback() {
30. @Override
31. **public** **void** onPictureTaken(**byte**[] data, Camera camera) {
32. // TODO Auto-generated method stub
33. **try**
34. {// 获得图片
35. Bitmap bm = BitmapFactory.decodeByteArray(data, 0, data.length);
36. File file = **new** File(filePath);
37. BufferedOutputStream bos =
38. **new** BufferedOutputStream(**new** FileOutputStream(file));
39. bm.compress(Bitmap.CompressFormat.JPEG, 100, bos);//将图片压缩到流中
40. bos.flush();//输出
41. bos.close();//关闭
42. }**catch**(Exception e)
43. {
44. e.printStackTrace();
45. }
47. }
48. };
49. /\*\* Called when the activity is first created. \*/
50. @Override
51. **public** **void** onCreate(Bundle savedInstanceState) {
52. **super**.onCreate(savedInstanceState);
53. requestWindowFeature(Window.FEATURE\_NO\_TITLE);//无标题
54. //设置拍摄方向
55. **this**.setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);
56. setContentView(R.layout.main);
57. //获得控件
58. mySurfaceView = (SurfaceView)findViewById(R.id.surfaceView1);
59. //获得句柄
60. holder = mySurfaceView.getHolder();
61. //添加回调
62. holder.addCallback(**this**);
63. //设置类型
64. holder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);
65. //设置监听
66. mySurfaceView.setOnClickListener(**this**);
67. }
68. @Override
69. **public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width,
70. **int** height) {
71. // TODO Auto-generated method stub
72. //设置参数并开始预览
73. Camera.Parameters params = myCamera.getParameters();
74. params.setPictureFormat(PixelFormat.JPEG);
75. params.setPreviewSize(640,480);
76. myCamera.setParameters(params);
77. myCamera.startPreview();
79. }
80. @Override
81. **public** **void** surfaceCreated(SurfaceHolder holder) {
82. // TODO Auto-generated method stub
83. //开启相机
84. **if**(myCamera == **null**)
85. {
86. myCamera = Camera.open();
87. **try** {
88. myCamera.setPreviewDisplay(holder);
89. } **catch** (IOException e) {
90. // TODO Auto-generated catch block
91. e.printStackTrace();
92. }
93. }
95. }
96. @Override
97. **public** **void** surfaceDestroyed(SurfaceHolder holder) {
98. // TODO Auto-generated method stub
99. //关闭预览并释放资源
100. myCamera.stopPreview();
101. myCamera.release();
102. myCamera = **null**;
104. }
105. @Override
106. **public** **void** onClick(View v) {
107. // TODO Auto-generated method stub
108. **if**(!isClicked)
109. {
110. myCamera.autoFocus(**this**);//自动对焦
111. isClicked = **true**;
112. }**else**
113. {
114. myCamera.startPreview();//开启预览
115. isClicked = **false**;
116. }
118. }
119. @Override
120. **public** **void** onAutoFocus(**boolean** success, Camera camera) {
121. // TODO Auto-generated method stub
122. **if**(success)
123. {
124. //设置参数,并拍照
125. Camera.Parameters params = myCamera.getParameters();
126. params.setPictureFormat(PixelFormat.JPEG);
127. params.setPreviewSize(640,480);
128. myCamera.setParameters(params);
129. myCamera.takePicture(**null**, **null**, jpeg);
130. }
132. }
133. }

main.xml:

**[html]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **<linearlayout** xmlns:android="http://schemas.android.com/apk/res/android"
2. androidrientation="vertical"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="fill\_parent"
5. **>**
6. **<surfaceview** android:id="@+id/surfaceView1"
7. android:layout\_width="640px"
8. android:layout\_height="480px"
9. android:layout\_gravity="center"**>**

AndroidManifest.xml:

**[html]** [view plaincopyprint?](http://blog.csdn.net/jason0539/article/details/10125017)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
3. package="com.wjh.camera"
4. android:versionCode="1"
5. android:versionName="1.0"**>**
6. **<uses-sdk** android:minSdkVersion="7" **/>**
7. **<uses-permission** android:name="android.permission.CAMERA"**></uses-permission>**
8. **<uses-feature** android:name="android.hardware.camera.autofocus" **/>**
9. **<uses-permission** android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"**/>**
10. **<application** android:icon="@drawable/icon" android:label="@string/app\_name"**>**
11. **<activity** android:name=".CameraTest\_4"
12. android:label="@string/app\_name"**>**
13. **<intent-filter>**
14. **<action** android:name="android.intent.action.MAIN" **/>**
15. **<category** android:name="android.intent.category.LAUNCHER" **/>**
16. **</intent-filter>**
17. **</activity>**
18. **</application>**
19. **</manifest>**

PS:其实自己去实现相机的必要性不大,大部分应用都是直接调用系统的相机,毕竟相机是涉及到一些比较专业的成像知识.