Cell0907

A bit of everything

Android OpenCV ODE Linux Blogger C/C++ Java Qt Inventions AI/Robotics

THURSDAY, JANUARY 16, 2014

Android Camera capture without display/user interface preview

Here I am going to capture the scene with the Android camera without showing in the screen what the camera is seeing but something completely unrelated to the captured image itself. This has several uses. In my case, I plan to do image processing on the pics but actually display something completely unrelated.

If you want to show what the camera is seeing, it is something standard and you can find more documentation out there. I provide some links all the way down.

Avoiding to show it, which somebody would think it should be straightforward, seems to be unsupported in Android. Some say that it may be because of privacy concerns (you want to see that the camera is on by seeing the display on). I kind of doubt it, because, as I'll show it can actually be done easilly. Just that is not documented.

I found so far two approaches, which basically should yield a third one that I still need to research:

1. Directly, using the Android camera API. I found the solution here which pointed also here (the same).
2. Using OpenCV (I'll show that on a different post).
3. One would think that if OpenCV in Android can do it, there may be an Android API way using the same trick that OpenCV is using. Got to research that...

So, anyhow, let's look at the first method, directly, using ***Android camera API***:

The top level steps are:

1. Check if there is a camera
2. If there is, find the id of the one you want (front or back...). Check the findFrontFacingCamera routine in the code below.
3. Open it (camera.open API call). See safeCameraOpen.
4. Create a fake SurfaceView and set it for the camera. Usually this is used for the preview of the camera (again, see all the way below for the typical use), but here we just trick the camera as we never actually display it. This, with #5, are the key differences respect to displaying the preview...
5. When we want to take a picture, triggered on any way (somebody presses a button or touches the screen or, in my case, with a timer), call camera.takePicture. The arguments of the method are callback functions that happen along the process of taking the picture. From the documentation:

   Triggers an asynchronous image capture. The camera service will initiate a series of callbacks to the application as the image capture progresses. The shutter callback occurs after the image is captured. This can be used to trigger a sound to let the user know that image has been captured. The raw callback occurs when the raw image data is available (NOTE: the data will be null if there is no raw image callback buffer available or the raw image callback buffer is not large enough to hold the raw image). The postview callback occurs when a scaled, fully processed postview image is available (NOTE: not all hardware supports this). The jpeg callback occurs when the compressed image is available. If the application does not need a particular callback, a null can be passed instead of a callback method.
   This method is only valid when preview is active (after startPreview()). Preview will be stopped after the image is taken; callers must call startPreview() again if they want to re-start preview or take more pictures. This should not be called between start() and stop().
   After calling this method, you must not call startPreview() or take another picture until the JPEG callback has returned.

7. The shutter callback is mostly used to trigger a shutter sound. The raw callback would be the ideal thing to use in my case, for image processing. Nevertheless, many posts warn about the poor documentation and manufacturer specific issues, so, I go, for the time being, for the safe/next one, which uses jpeg encoding, unfortunately wasting precious processor bandwidth to encode/decode the image and loss of quality (although I don't think the second is so critical to me).
8. Parallel to that, we have an ImageView that we use to display whatever we want related or not to the camera.
9. So, that would be it. Notice that in my code I trigger the capture with a timer done with an AsyncTask. After a certain time, the task sends a message that will make the handle call takePicture, setting the callback to get a jpeg from the camera. When done, sends another message back and then the handle presents that picture in the display, re-stating the whole process. Notice that:
   1. I present the picture because I want (just to show is working), but I didn't have to. I can choose to present whatever I want in ImageView. That was the whole point...
   2. In my case, actually I want to take pictures as fast as possible, so, the timer is actually irrelevant. I could just delete it and call takePicture directly, but I leave it there to explain how I would do it with delay in the middle...

MainActivity.java

```
package com.example.camera1;
import java.io.IOException;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
```

SEARCH

INDEX

Matlab
Electronics
Android
Linux
AI/Robotics
Blogger
C/C++
Java
OpenCV
Qt
Inventions
Healthcare
Thoughts
Cars
Others
IT

CONTACT FORM

Name

Email *

Message *

Send

FAVOURITES

TED conference
Khan's academy
Inventor spot blog
Eva's

```
import android.hardware.Camera.CameraInfo;
import android.util.Log;
import android.view.SurfaceView;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends Activity {
    public static final int DONE=1;
    public static final int NEXT=2;
    public static final int PERIOD=1;
    private Camera camera;
    private int cameraId = 0;
    private ImageView display;
    private Timer timer;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     display=(ImageView)findViewById(R.id.imageView1);
     // do we have a camera?
     if (!getPackageManager()
       .hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
     Toast.makeText(this, "No camera on this device", Toast.LENGTH_LONG)
       .show();
     } else {
      cameraId = findFrontFacingCamera();
      if (cameraId < 0) {
       Toast.makeText(this, "No front facing camera found.",
         Toast.LENGTH_LONG).show();
      } else {
          safeCameraOpen(cameraId);
      }
     }
     // THIS IS JUST A FAKE SURFACE TO TRICK THE CAMERA PREVIEW
     // http://stackoverflow.com/questions/17859777/how-to-take-pictures-in-android-
     // application-without-the-user-interface
     SurfaceView view = new SurfaceView(this);
     try {
            camera.setPreviewDisplay(view.getHolder());
       } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
       }
     camera.startPreview();
     Camera.Parameters params = camera.getParameters();
     params.setJpegQuality(100);
     camera.setParameters(params);
     // We need something to trigger periodically the capture of a
     // picture to be processed
     timer=new Timer(getApplicationContext(),threadHandler);
     timer.execute();
     }
    ////////////////////////////////////thread Handler/////////////////////////////////////////
    private Handler threadHandler = new Handler() {
        public void handleMessage(android.os.Message msg) {
            switch(msg.what){
            case DONE:
                // Trigger camera callback to take pic
                camera.takePicture(null, null, mCall);
                break;
            case NEXT:
                timer=new Timer(getApplicationContext(),threadHandler);
                timer.execute();
                break;
            }
            }
       };
    Camera.PictureCallback mCall = new Camera.PictureCallback() {
        public void onPictureTaken(byte[] data, Camera camera) {
          //decode the data obtained by the camera into a Bitmap
            //display.setImageBitmap(photo);
            Bitmap bitmapPicture
            = BitmapFactory.decodeByteArray(data, 0, data.length);
            display.setImageBitmap(bitmapPicture);
            Message.obtain(threadHandler, MainActivity.NEXT, "").sendToTarget();
            //Log.v("MyActivity","Length: "+data.length);
        }
    };
    private int findFrontFacingCamera() {
        int cameraId = -1;
        // Search for the front facing camera
        int numberOfCameras = Camera.getNumberOfCameras();
        for (int i = 0; i < numberOfCameras; i++) {
            CameraInfo info = new CameraInfo();
            Camera.getCameraInfo(i, info);
            if (info.facing == CameraInfo.CAMERA_FACING_FRONT) {
                Log.v("MyActivity", "Camera found");
          cameraId = i;
          break;
         }
```

```
        }
        return cameraId;
       }
    @Override
    protected void onPause() {
        if (timer!=null){
            timer.cancel(true);
        }
      releaseCamera();
      super.onPause();
     }
    // I think Android Documentation recommends doing this in a separate
    // task to avoid blocking main UI
    private boolean safeCameraOpen(int id) {
      boolean qOpened = false;
      try {
        releaseCamera();
        camera = Camera.open(id);
        qOpened = (camera != null);
      } catch (Exception e) {
        Log.e(getString(R.string.app_name), "failed to open Camera");
        e.printStackTrace();
      }
      return qOpened;
    }
    private void releaseCamera() {
      if (camera != null) {
          camera.stopPreview();
        camera.release();
        camera = null;
      }
    }
}
```

Timer.java

```
package com.example.camera1;
import android.content.Context;
import android.os.Handler;
import android.os.Message;
import android.os.AsyncTask;
public class Timer extends AsyncTask<Void, Void, Void> {
  Context mContext;
    private Handler threadHandler;
  public Timer(Context context,Handler threadHandler) {
    super();
    this.threadHandler=threadHandler;
    mContext = context;
     }
  @Override
    protected Void doInBackground(Void...params) {
      try {
            Thread.sleep(MainActivity.PERIOD);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
      Message.obtain(threadHandler, MainActivity.DONE, "").sendToTarget();
       return null;
  }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.camera1"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="9"
    android:targetSdkVersion="17" />
     <uses-permission android:name="android.permission.CAMERA"/>
  <application
    android:label="@string/app_name"
    android:icon="@drawable/ic_launcher"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
    android:allowBackup="false">
    <activity android:name="MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape"
        android:configChanges="keyboardHidden|orientation">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
```

```
</manifest>
```

And the layout activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" >
  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:src="@drawable/ic_launcher" />
</RelativeLayout>
```

Cheers!

PS.: Check out this video on the Android 3D display I build based on this stuff and the details here.
PS1.: Please, click here to see an index of other posts on Android.
PS2.: Other camera examples:
http://www.vogella.com/articles/AndroidCamera/article.html
http://android-er.blogspot.com.es/2010/12/implement-takepicture-function-of.html
http://android.codota.com/scenarios/518915fdda0a68487f6039c2/android.hardware.Camera?tag=out_2013_05_05_07_19_34&fullSource=1

Posted by cell0907 at 8:35 PM

Labels: android

11 comments:

Anonymous January 19, 2014 at 10:21 AM

Thank you for the post. Some questions:

1. What phone and OS version are you using to testing this?
2. Does it work when the screen is OFF?
3. Have you tested it when running the picture taking from a background service?

My testing with Nexus devices and KitKat 4.4.2 (with a similar) approach show that the fake surface view doesn't work when launching from a service, that the screen flickers when trying to draw a transparent activity, and that the picture can't be taken when the screen is OFF.

Have you tried this?

Reply

Replies

cell0907 January 19, 2014 at 12:52 PM

Hi, got an HTC One, Android 4.3. I have not tried from service or with the screen off. For the 2nd, if you want I can try it, but how do you make the screen off? With this http://stackoverflow.com/questions/6756768/turn-off-screen-on-android/6757206#6757206 or this method http://stackoverflow.com/questions/7198336/android-turn-screen-off?

By the way, does it work in normal app, like the code above?

Anonymous January 19, 2014 at 6:26 PM

Well, if you take this code and put it in a background service, then you can have it take a picture based on a certain event, or by using a timer/alarm. Consider the case of an alarm -- you would just turn the screen off yourself and have the alarm set to fire in 1 minute.

This used to work in the past, but now it doesn't. I take it that Google has disabled this capability.

cell0907 January 25, 2014 at 2:01 PM

Sorry but didn't have time to dig on this... Nevertheless, there is a method that works through OpenCV. Yeah, I know, that may sound like too much detour, but you can check an example I put here https://www.blogger.com/blogger.g?blogID=503320906205597654 7#allposts. Quick summary, basically, when you get a frame, opencv calls public Mat onCameraFrame(CvCameraViewFrame inputFrame) and what you return is displayed... (see point #2 in the link) Obviously, if OpenCV can do it, I guess that it can always be done (ie., not disable). But I don't know though, the inner workings. I think it works with a native routine, but didn't have the time to investigate yet...

Unknown February 13, 2014 at 9:47 PM

Thanks. Can you repost the link to the OpenCV post you mentioned above? Your link doesn't work for me -- it appears to be linked to your username rather than the public-facing blog.

cell0907 February 13, 2014 at 10:03 PM

WOPS!! You right! Try http://cell0907.blogspot.com/2014/01/detecting-faces-in-android-with-opencv.html Actually, I have updated also the original post with this link (option #2) all the way to the top.

Reply

Anonymous April 18, 2014 at 11:38 PM

hi,

I tried to use this code but i am getting

java.lang.NullPointerException
at com.example.clickimage.MainActivity$1.handleMessage

at this line
camera.takePicture(null, null, mCall);

Can you help me out ?

Reply

Replies

Anonymous January 2, 2015 at 2:41 PM

same

Reply

Anonymous October 11, 2014 at 6:11 AM

Does this code work for anyone?

Reply

Anonymous January 15, 2016 at 9:48 AM

it's doesn't work!!!!

Reply

Unknown December 4, 2016 at 11:28 AM

this code is working but it showing camera surface

Reply

Enter your comment...

Comment as: 小草 (Google) **Sign out**

Publish **Preview** Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

PAGES

Blog

About this blog