About This Article

This article gives a overview ofthe most common issues in Android.This article shows issues related to Camera, Sensor, SeekBar andContent Providers.

Scope:

This article is intended for Android developers wishing to develop mobile applications. It assumes good knowledge of Android andJava programming language.

Position of camera preview

Android API levels:

7, 8, 9, 10

Issue:

Preview displays in the wrong orientation.

Solution:

Set the screen orientation to landscape. This can be done by adding android:screenOrientation="landscape" to the <activity></activity>tag in your AndroidManifest.xml file. If you wouldprefer portrait orientation, add android:screenOrientation="portrait" to the XML and call android.hardware.Camera's method setDisplayOrientation(90):

```
Camera mCamera = Camera.open();
// ...
mCamera.setDisplayOrientation(90);
```

Additional links:

http://developer.android.com/reference/android/hardware/Camera.html#setDisplayOrientation%28int%29
http://code.google.com/p/android/issues/detail?id=1193
http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/graphics/CameraPreview.html

Issue:

Unable to set the preview to another position on the screen.

Solution:

You can move preview by changing the position of its underlying surface. Typically, preview is displayed on the SurfaceView. So, you should position SurfaceView in (for example) main.xml and call setContentView(R.layout.main) in your Activity.

Additional links:

http://developer.android.com/reference/android/view/SurfaceView.html

Recognition of front camera

Android API levels:

9, 10

Issue:

I don't know how to use front camera

Solution:

Instead of camera.open() use following getFrontFacingCamera() function:

```
Camera getFrontFacingCamera() throws NoSuchElementException {
    Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
    for (int cameraIndex = 0; cameraIndex < Camera.getNumberOfCameras(); cameraIndex++) {
        Camera.getCameraInfo(cameraIndex, cameraInfo);
        if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
            try {
                return Camera.open(cameraIndex);
            }
            catch (RuntimeException e) {
                e.printStackTrace();
            }
        }
    }
    throw new NoSuchElementException("Can't find front camera.");}
```

Samsung Galaxy S with Android API level 7, 8 solution

```
After Camera mCamera = Camera.open() use setFrontCamera(mCamera):

void setFrontCamera(Camera camera) {
    Camera.Parameters parameters = camera.getParameters();
    parameters.set("camera-id", 2);
    // (800, 480) is also supported front camera preview size at Samsung Galaxy S.
    parameters.setPreviewSize(640, 480);
    camera.setParameters(parameters);
}
```

Additional links:

http://developer.android.com/reference/android/hardware/Camera.CameraInfo.html http://stackoverflow.com/questions/4241292/how-to-use-front-facing-camera-on-samsung-galaxy-s

You can find the complete source code referred to above camera problems in the Appendix, at the end of this document.

Incorrect Light Sensor values

Issue:

There are differences between values returned from the Light Sensor in the S and S II devices. There are 4 ranges of returned values which are shown in Table1.

Sr No	Galaxy S	Galaxy S II	Range
1	6	10	(0; 1000)
2	5000	1000	<1000; 9000)
3	9000	10000	<9000; 15000)
4	15000	16000	<15000; …)

Solution:

```
package com.samsung.sensorexample;
```

```
import com.samsung. lightsensorexample.R;
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class SensorExampleActivity extends Activity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor mLightSensor;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mLightSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
        setContentView(R.layout.main);
    }

    protected void onResume() {
        mSensorManager.registerListener(this, mLightSensor, SensorManager.SENSOR_DELAY_UI);
        super.onResume();
    }

    protected void onPause() {
        mSensorManager.unregisterListener(this);
        super.onPause();
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    private enum LightLevel {
        LIGHT_LVL_UNKNOWN, LIGHT_LVL_1, LIGHT_LVL_2, LIGHT_LVL_3, LIGHT_LVL_4
    }

    private LightLevel getLightLevelBySensorValue(float sensorValue) {
        if (sensorValue < 1000.0) {
            return LightLevel.LIGHT_LVL_1;
        }
        if (sensorValue >= 1000.0 && sensorValue < 9000.0) {
            return LightLevel.LIGHT_LVL_2;
        }
        if (sensorValue >= 9000.0 && sensorValue < 15000.0) {
            return LightLevel.LIGHT_LVL_3;
        }
        if (sensorValue >= 15000) {
            return LightLevel.LIGHT_LVL_4;
        }
        return LightLevel.LIGHT_LVL_UNKNOWN;
    }

    public void onSensorChanged(SensorEvent event) {
        final LightLevel lightLevel = getLightLevelBySensorValue(event.values[0]);
        if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
            TextView txt = (TextView) findViewById(R.id.lightSensor);
            txt.setText("Light level: " + lightLevel);
        }
    }

}
```

SeekBar’s stack overflow

Issue:

Changing the progress on the SeekBar inside the onProgressChanged() method causes a stack overflow on some devices.

Description:

OnSeekBarChangeListener has a onProgressChanged() method that is called every time the progress property of the SeekBar is changed. Calling setProgress() inside the onProgressChanged() method will create an infinite loop and cause a stack overflow if there are no prevention mechanisms. On Samsung Galaxy S, Galaxy S II and Galaxy Tab 10.1 with Android 2.3 this loop will be blocked after a couple of iterations

Solution:

If a programmer's intention is to prevent changing the value of progress, he/she should consider using the setEnabled(false) method instead.

SMS/MMS/Contacts/Calendar

Some developers use content://sms/inbox, content://mms-sms/conversations/ or content://calendar/events URIs to retrieve messages, calendars, etc. For instance:

```
Uri inbox = Uri.parse("content://sms/inbox");
Cursor cursor = getContentResolver().query(inbox, null, null, null, null);
while (cursor.moveToNext()) {
    String sms = cursor.getString(2);
}
```
However these are not part of the Android SDK and should not be accessed by android applications. They won't work on some devices and can be removed in future releases.

This link:

http://android-developers.blogspot.com/2010/05/be-careful-with-content-providers.html
describes common bad practices concerning content providers. There's no official way of accessing the afore mentioned content.

SMS

Android documentation describes the android.telephony.SmsManager class that can be used to send SMS messages. To intercept incoming SMS messages you can use BroadcastReceiver by overriding onReceive() method. This link:

http://mobiforge.com/developing/story/sms-messaging-android
provides valuable examples. Notice that android.telephony.gsm.SmsManager is a deprecated class.

MMS

The method mention above won't work for MMS messages and there's no official substitute.

Calendar

It is not supported. There are no official substitutes.

Contacts

Contacts.People class is deprecated and so are "content://contacts/people/" "content://contacts/lookup/". You should use ContactsContract instead.

Summary:

content://sms, content://mms-sms, content://calendar/

It is not supported. There are no official substitutes.

content://contacts/people/

Contacts.People class is deprecated. See ContactsContract
content://contacts/lookup/

content://contacts/lookup/ is deprecated. See ContactsContract
Additional links:

http://android-developers.blogspot.com/2010/05/be-careful-with-content-providers.html http://developer.android.com/reference/android/provider/ContactsContract.htmlhttp://groups.google.com/group/android-developers/browse_thread/thread/0332110ff8ecbc20http://developer.android.com/reference/android/telephony/SmsManager.html http://mobiforge.com/developing/story/sms-messaging-android http://mobdev.olin.edu/mobdevwiki/FrontPage/Tutorials/SMS%20Messaging http://stackoverflow.com/questions/3012287/how-to-read-mms-data-in-android

Appendix

Camera preview
src/com/samsung/CameraPreview/CameraPreviewActivity.java:

```java
package com.samsung.cameraPreview;

import java.io.IOException;
import java.util.NoSuchElementException;

import android.app.Activity;
import android.content.Context;
import android.hardware.Camera;
import android.os.Bundle;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.Window;

public class CameraPreviewActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Hide the window title.
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
    }

}

class Preview extends SurfaceView implements SurfaceHolder.Callback {

    private Camera mCamera;
    private final SurfaceHolder mHolder;
    final static int SUPPORTED_WIDTH = 640;
    final static int SUPPORTED_HEIGHT = 480;

    public Preview(Context context, AttributeSet attributes) {
        super(context, attributes);
        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    /**
     * Works in API level >= 10.
     *
     * @return Front camera handle.
     */
    Camera getFrontFacingCamera() throws NoSuchElementException {
        final Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
        for (int cameraIndex = 0; cameraIndex < Camera.getNumberOfCameras(); cameraIndex++)
{
            Camera.getCameraInfo(cameraIndex, cameraInfo);
            if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
                try {
                    return Camera.open(cameraIndex);
                } catch (final RuntimeException e) {
                    e.printStackTrace();
                }
            }
        }
        throw new NoSuchElementException("Can't find front camera.");
    }

    /**
     * Works in API level >= 7 at Samsung Galaxy S.
     *
     * @param Camera handle.
     */
    void setFrontCamera(Camera camera) {
        final Camera.Parameters parameters = camera.getParameters();
        parameters.set("camera-id", 2);
        try {
            camera.setParameters(parameters);
        } catch (final RuntimeException e) {
            // If we can't set front camera it means that device hasn't got "camera-id".
Maybe it's not Galaxy S.
            e.printStackTrace();
        }
    }

    /**
     * @see android.view.SurfaceHolder.Callback#surfaceChanged(android.view.SurfaceHolder,
int, int, int)
     */
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        // Now that the size is known, set up the camera parameters and begin the preview.
        final Camera.Parameters parameters = mCamera.getParameters();
        parameters.setPreviewSize(SUPPORTED_WIDTH, SUPPORTED_HEIGHT);
        mCamera.setParameters(parameters);
        mCamera.startPreview();
    }

    /**
     * The Surface has been created, acquire the camera and tell it where to draw.
     *
     * @see android.view.SurfaceHolder.Callback#surfaceCreated(android.view.SurfaceHolder)
     */
    public void surfaceCreated(SurfaceHolder holder) {
        if (android.os.Build.VERSION.SDK_INT >= 10) {
            mCamera = getFrontFacingCamera();
            mCamera.setDisplayOrientation(90);
        } else {
            mCamera = Camera.open();
            setFrontCamera(mCamera);
        }
        try {
            mCamera.setPreviewDisplay(holder);
        } catch (final IOException e) {
            mCamera.release();
            mCamera = null;
            e.printStackTrace();
        }
    }

    /**
     * @see
android.view.SurfaceHolder.Callback#surfaceDestroyed(android.view.SurfaceHolder)
     */
    public void surfaceDestroyed(SurfaceHolder holder) {
        // Surface will be destroyed when we return, so stop the preview.
        // Because the CameraDevice object is not a shared resource, it's very
        // important to release it when the activity is paused.
        mCamera.stopPreview();
        mCamera.release();
        mCamera = null;
    }


}
```

res/layout/main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:configChanges="orientation">
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <Button android:layout_height="wrap_content" android:layout_width="match_parent"
android:text="Button" android:id="@+id/button1">
        <com.samsung.cameraPreview.Preview
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:id="@+id/preview" android:layout_marginTop="100px"/>
    </LinearLayout>
</FrameLayout>
```

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      android:versionCode="1"
      android:versionName="1.0" package="com.samsung.cameraPreview">
    <uses-sdk android:minSdkVersion="7"/>

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name="com.samsung.cameraPreview.CameraPreviewActivity"
                  android:label="@string/app_name"
                  android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" android:required="false" />
    <uses-feature android:name="android.hardware.camera.front" android:required="false" />
</manifest>
```

Contents provider

src/com/samsung/contacts/ ContactsDemoActivity.java

```
package com.samsung.contacts;

import android.accounts.Account;
import android.accounts.AccountManager;
import android.app.Activity;
import android.app.alertDialog;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract.Contacts;
import android.provider.ContactsContract.RawContacts;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;

/**
 * This class demonstrates how to select and display contacts
 */
public class ContactsDemoActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final Cursor contacts = getContacts();
        ListView contactListView = setList(contacts);

        contactListView.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView parent, View view, int position, long id)
{

                Cursor contactData = getContactData(contacts, position);
                showContactData(contactData, view);
                contactData.close();
            }
        });
    }

    /**
     * This method will select all contacts
     * of a first registered account. If there are no accounts,
     * all existing contacts will be selected.
     *
     * @return Cursor returned after contacts query.
     *          This cursor is used by other methods,
     *          e.g. OnItemClickListener(), so it remains opened.
     */
    private Cursor getContacts() {
        Uri rawContactUri = RawContacts.CONTENT_URI;
        AccountManager am = AccountManager.get(getApplicationContext());
        Account[] accounts = am.getAccounts();

        // Select only contacts of a first account if there are any accounts
        if(accounts.length > 0) {
            rawContactUri = RawContacts.CONTENT_URI.buildUpon()
                .appendQueryParameter(RawContacts.ACCOUNT_NAME, accounts[0].name)
                .appendQueryParameter(RawContacts.ACCOUNT_TYPE, accounts[0].type)
                .build();
        }

        String[] projection = new String[] {
                RawContacts._ID,
                RawContacts.CONTACT_ID,
                RawContacts.DELETED,
                RawContacts.ACCOUNT_NAME,
                RawContacts.ACCOUNT_TYPE
        };

        return managedQuery(rawContactUri, projection, null, null, null);
    }

    /**
     * Set adapter describing how to display contacts list.
     *
     * @param contacts Cursor returned by getContacts().
     * @return List control reference.
     */
    private ListView setList(Cursor contacts) {
        ListView contactListView = (ListView) findViewById(R.id.contactList);

        String[] fields = new String[] {
                RawContacts.CONTACT_ID,
                RawContacts.DELETED,
                RawContacts.ACCOUNT_NAME,
                RawContacts.ACCOUNT_TYPE
        };

        // SimpleCursorAdapter constructor that is used here is deprecated since API 11.
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
R.layout.contact_list_item, contacts,
                fields, new int[] {R.id.contactId, R.id.contactDeleted,
R.id.contactAccName, R.id.contactAccType});

        contactListView.setAdapter(adapter);

        return contactListView;
    }

    /**
     * This method will select particular contact data.
     *
     * @param contacts Cursor returned by getContacts().
     * @param position Position of contact to be returned.
     * @return Cursor returned after querying for the contact.
     */
    private Cursor getContactData(Cursor contacts, int position) {
        int idIndex = contacts.getColumnIndex(RawContacts.CONTACT_ID);
        contacts.moveToPosition(position);
        int contactId = contacts.getInt(idIndex);
        String[] projection = new String[] {
                Contacts.DISPLAY_NAME,
                Contacts.HAS_PHONE_NUMBER
        };

        return managedQuery(Contacts.CONTENT_URI,   projection, Contacts._ID + "=?",
                new String[] {Integer.toString(contactId)}, null);
    }
    /**
     * Show popup with the selected contact data.
     *
     * @param contactData All data of contact.
     * @param view Current view.
     */
    private void showContactData(Cursor contactData, View view) {
        int nameIndex = contactData.getColumnIndex(Contacts.DISPLAY_NAME);
        int hasPhoneIndex = contactData.getColumnIndex(Contacts.HAS_PHONE_NUMBER);

        String details = "";

        if(contactData.moveToFirst()) {
            while(!contactData.isAfterLast()) {
                String name  = contactData.getString(nameIndex);
                boolean hasPhoneNumber = (contactData.getInt(hasPhoneIndex) == 1);
                details += "Name: " + name + ", HasPhone: " + hasPhoneNumber + "\n";

                contactData.moveToNext();
            }

            alertDialog.Builder builder = new alertDialog.Builder(view.getContext());
            builder.setMessage(details);
            builder.setNeutralButton("OK", null);
            builder.show();
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.samsung.contacts"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="10" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ContactsDemoActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
</manifest>
```

SeekBar

```
package com.samsung.progress;

import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;

public class ProgressBarOverflowActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        SeekBar bar = (SeekBar) findViewById(R.id.seekBar1);

        bar.setProgress(20);
        bar.setEnabled(false); // OK

        bar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
{
                seekBar.setProgress(20); // WRONG!
            }

        });
    }
}
```

ref:http://developer.samsung.com/android/technical-docs/Common-Issues-in-Android