[conowen]大钟的专栏

Android图形系统之Surface、SurfaceView、SurfaceHolder及SurfaceHolder.Callback之间的联系

标签： android 图形 interface callback thread

2012-08-03 17:29 23771人阅读 评论(6) 收藏 举报

分类： Android学习笔记（23）

目录(?) [+]

```
/**********************************************************************************************
 * author：conowen@大钟
 * E-mail：conowen@hotmail.com
 *site:http://menwoo.com/
 *深圳市大望谷科技有限公司
 * http://blog.csdn.net/conowen
 * 注：本文为原创，仅作为学习交流使用，转载请标明作者及出处。
 **********************************************************************************************/
```

1、Surface

Surface

extends Object

implements Parcelable

java.lang.Object

↳ android.view.Surface

Class Overview

Handle onto a raw buffer that is being managed by the screen compositor.

简单翻译：

Surface是原始图像缓冲区（raw buffer）的一个句柄，而原始图像缓冲区是由屏幕图像合成器（screen compositor）

1.1、 就如在C语言编程一样，通过一个文件的句柄，就可以操作文件，获取文件的内容。 同样的，通过Surface就可以获取raw buffer其中的内容。原生缓冲区（raw buffer）存储着当前窗口的像素数据。

1.2、事实上，当得到一个Surface对象时，同时会得到一个Canvas（画布）对象。这一点可以通过查看\frameworks\base\core\java\android\view\Surface.java文件可知道Surface类定义了一个Canvas成员变量

瑞芯微RK (3)
嵌入式相关 (1)
计算机相关 (2)
Android多媒体&流媒体开发 (6)
Android的NDK开发 (5)
所想所感 (1)
所想所感 ios (1)

文章存档

2015年09月 (1)
2014年11月 (4)
2014年04月 (2)
2012年08月 (6)
2012年07月 (4)
展开

阅读排行

Android学习笔记(21)—— (54166)
【整理】Android-Recove (47154)
Android的NDK开发(1)— (35597)
Android学习笔记(13)—— (28006)
Android图形系统之Surfa (23747)
Android学习笔记(12)—— (23704)
【整理】Libav、FFmpeg (22613)
Android的文件系统结构 (22525)
RKAndroidTool工具的各 (21844)
Android的NDK开发(3)— (21414)

评论排行

Android学习笔记(21)—— (76)
关于havlenapetr-FFMpe (64)
Android多媒体开发（3） (51)
Android多媒体开发（4） (46)
Android多媒体开发（5） (41)
Android学习笔记(12)—— (16)
Android学习笔记(20)—— (15)
Android多媒体开发（2） (14)
Android学习笔记(1)—— (9)
Android的NDK开发(2)— (8)

推荐文章

*机器学习与数据挖掘网上资源搜罗——良心推荐
*架构设计：系统间通信（17）——服务治理与Dubbo 中篇（分析）
* 拉开大变革序幕（中）：Docker 场景化尝试
*Android应用开发allowBackup敏感信息泄露的一点反思
*Linux多线程实践（四 ）线程的特定数据
*Android点击Button水波纹效果

最新评论

关于havlenapetr-FFMpeg在Andr
人生一抹儿绿色: 向大神致敬，谢谢分享

```java
//@\frameworks\base\core\java\android\view\Surface.java
// The mSurfaceControl will only be present for Surfaces used by the window
// server or system processes. When this class is parceled we defer to the
// mSurfaceControl to do the parceling. Otherwise we parcel the
// mNativeSurface.
private int mSurfaceControl;
private int mSaveCount;
private Canvas mCanvas;
private int mNativeSurface;
private int mSurfaceGenerationId;
private String mName;
```

1.3、 理解Canvas对象，可以把它当做画布，Canvas的方法大多数是设置画布的大小、形状、画布背景颜色等等，要想在画布上面画画，一般要与Paint对象结合使用，顾名思义，Paint就是画笔的风格，颜料的色彩之类的。

```java
// 创建画笔
Paint paint = new Paint();
paint.setColor(Color.RED);// 设置红色

canvas.drawCircle(60, 20, 10, paint);// 画一个圆
```

1.4、Surface本身的作用类似一个句柄，得到了这个句柄就可以得到其中的Canvas、原生缓冲器以及其它方面的内容。

1.5、Surface实现了Parcelable接口，（implements Parcelable），也就是说Surface对象可以把显示内容的数据写入到 Parcel 中，并且能够从Parcel读回数据。

Parcelable

android.os.Parcelable

▶Known Indirect Subclasses

AbsSavedState,AbsoluteSizeSpan,AccessibilityEvent,AccessibilityNodeInfo,AccessibilityServiceInfo,Account,Acc

and 144 others.

Class Overview

Interface for classes whose instances can be written to and restored from a `Parcel`. Classes implementing the Pa

简单翻译：

实现这个接口的对象可以写入数据到Parcel，同时也可以把数据读出来。

2、SurfaceView

SurfaceView

extends View

java.lang.Object

↳ android.view.View

↳ android.view.SurfaceView

▶Known Direct Subclasses

GLSurfaceView,RSSurfaceView,VideoView

Class Overview

Provides a dedicated drawing surface embedded inside of a view hierarchy. You can control the format of this surf

size; the SurfaceView takes care of placing the surface at the correct location on the screen

The surface is Z ordered so that it is behind the window holding its SurfaceView; the SurfaceView punches a hole

surface to be displayed. The view hierarchy will take care of correctly compositing with the Surface any siblings of
would normally appear on top of it. This can be used to place overlays such as buttons on top of the Surface, thou
can have an impact on performance since a full alpha-blended composite will be performed each time the Surface
Access to the underlying surface is provided via the SurfaceHolder interface, which can be retrieved by calling`get`
The Surface will be created for you while the SurfaceView's window is visible; you should
implement`surfaceCreated(SurfaceHolder)` and`surfaceDestroyed(SurfaceHolder)` to discover when
and destroyed as the window is shown and hidden.
One of the purposes of this class is to provide a surface in which a secondary thread can render into the screen. If
this way, you need to be aware of some threading semantics:

- All SurfaceView and `SurfaceHolder.Callback` methods will be called from the thread running the Surfa
  (typically the main thread of the application). They thus need to correctly synchronize with any state that is a
  drawing thread.
- You must ensure that the drawing thread only touches the underlying Surface while it is valid --
  between`SurfaceHolder.Callback.surfaceCreated()` and`SurfaceHolder.Callback.surfaceD`

简单翻译：

SurfaceView提供了一个专门用于绘制的surface，这个surface内嵌于。你可以控制这个Surface的格式和尺寸，
Surface在屏幕的正确绘制位置。

surface是Z-ordered的（也就是说在xyz坐标系中，按照Z坐标排序的，Z值大的表面覆盖在Z值小的表面的上方
在窗口的后面。surfaceview在显示窗口处为Surface提供了一个可见区域，通过这个区域，才能看到Surface里面的
图层（overlays）在Surface上面，如Button、Textview之类的。但是，需要注意的是，如果Surface上面有全透明的
的每一次变化，这些全透明的控件就会重新渲染，这样的话，就影响性能与显示的效果。

你可以通过SurfaceHolder这个接口去访问Surface，而执行getHolder()方法可以得到SurfaceHolder接口。

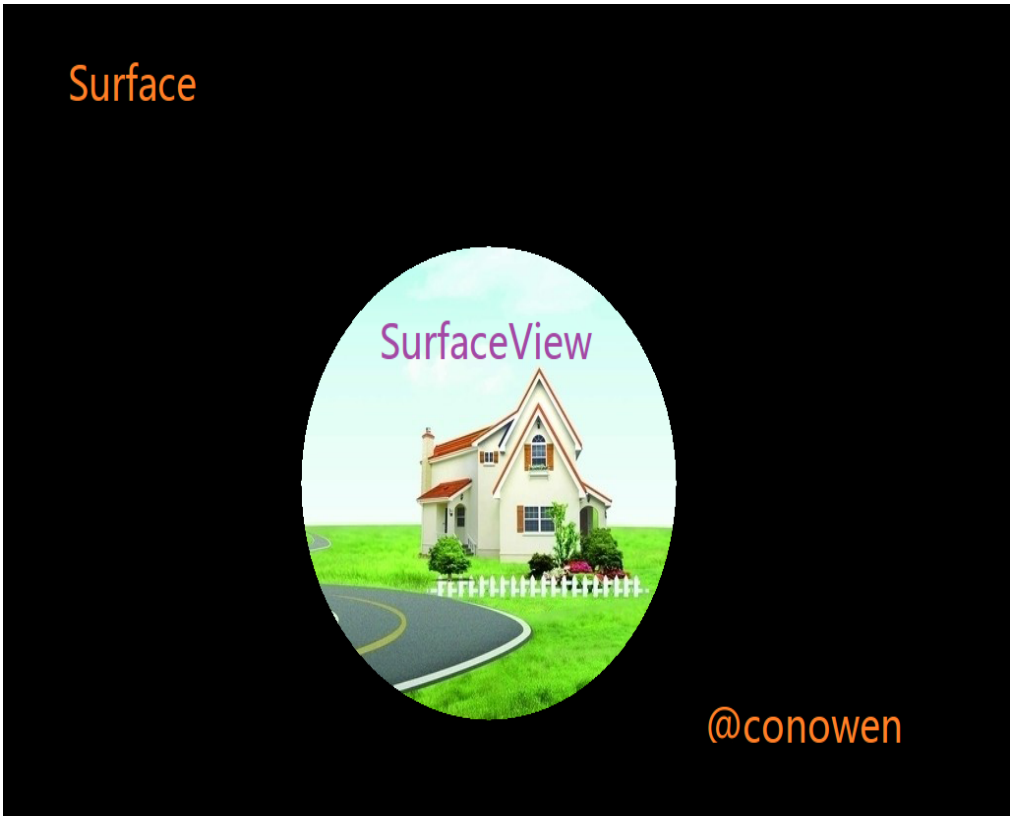
当SurfaceView的窗口可见时，Surface就会被创建，当SurfaceView窗口隐藏时，Surface就会被销毁。当然了
`surfaceCreated(SurfaceHolder)` 和 `surfaceDestroyed(SurfaceHolder)` 这两个方法来验证一下Surfa
销毁。

SurfaceView提供了一个运行在渲染线程的surface，若你要更新屏幕，你需要了解以下线程知识。

- 所有SurfaceView 和 `SurfaceHolder.Callback`的方法都应该在主线程（UI线程）里面调用，应该要确保
  步性。
- 你必须确保只有当Surface有效的时候，（也就是当Surface的生命周期在`SurfaceHolder.Callback.sur`
  `SurfaceHolder.Callback.surfaceDestroyed()`之间）才能让渲染进程访问。

2.1、SurfaceView与Surface的联系

简单来说，SurfaceView与Surface的联系就是，Surface是管理显示内容的数据（implementsParcelable），包括存储于数据的交换。而SurfaceView就是把这些数据显示出来到屏幕上面。
两者联系如图所示：



3、SurfaceHolder

SurfaceHolder

android.view.SurfaceHolder

Class Overview

Abstract interface to someone holding a display surface. Allows you to control the surface size and format, edit the monitor changes to the surface. This interface is typically available through the`SurfaceView` class.

When using this interface from a thread other than the one running its `SurfaceView`, you will want to carefully rea methods`lockCanvas()` and`Callback.surfaceCreated()`.

简单翻译:

SurfaceHolder是控制surface的一个抽象接口，你可以通过SurfaceHolder来控制surface的尺寸和格式，或者修 surface的变化等等，SurfaceHolder是SurfaceView的典型接口。

与直接控制SurfaceView来修改surface不同，使用SurfaceHolder来修改surface时，需要注意`lockCanvas()` `Callback.surfaceCreated()`.这两个方法。

SurfaceHolder控制surface的流程所使用的几个方法。

3.1、abstract void addCallback(SurfaceHolder.Callback callback)

Add a Callback interface for this holder.// 给SurfaceHolder一个回调对象。

3.2、abstract Canvas lockCanvas(Rect dirty)

Just like lockCanvas() but allows specification of a dirty rectangle.

// 锁定画布中的某一个区域，返回的画布对象Canvas（当更新的内容只有一个区域时，同时要追求高效，可 新一部分的区域，而不必更新全部画布区域）

3.3、abstract Canvas lockCanvas()

Start editing the pixels in the surface.// 锁定画布，返回的画布对象Canvas

3.4、abstract void removeCallback(SurfaceHolder.Callback callback)

Removes a previously added Callback interface from this holder.//移除回调对象

3.5、abstract void unlockCanvasAndPost(Canvas canvas)

Finish editing pixels in the surface.// 结束锁定画图，并提交改变。

4、SurfaceHolder.Callback

SurfaceHolder.Callback

android.view.SurfaceHolder.Callback

Known Indirect Subclasses

GLSurfaceView,NativeActivity,RSSurfaceView,SurfaceHolder.Callback2

Class Overview

A client may implement this interface to receive information about changes to the surface. When used with a`SurfaceView`, the Surface being held is only available between calls to`surfaceCreated(SurfaceHolder)` and`surfaceDestroyed(SurfaceHolder)`. The Callback is set with`SurfaceHolder.addCallback` method.

简单翻译：

SurfaceHolder.Callback是监听surface改变的一个接口

4.1、public abstract voidsurfaceChanged(SurfaceHolder holder, int format, int width, int height)

holder The SurfaceHolder whose surface has changed.

format The new PixelFormat of the surface.

width The new width of the surface.

height The new height of the surfa

//surface发生改变时被调用

4.2、public abstract voidsurfaceCreated(SurfaceHolder holder)

Parameters

holder The SurfaceHolder whose surface is being created

//在surface创建时被调用，一般在这个方法里面开启渲染屏幕的线程。

4.3、public abstract voidsurfaceDestroyed(SurfaceHolder holder)

Parameters

holder The SurfaceHolder whose surface is being destroyed.

//销毁时被调用，一般在这个方法里将渲染的线程停止。

附上上述所说几种的联系方法

```java
SurfaceHolder = SurfaceView.getHolder();

Surface = SurfaceHolder.getSurface();

Canvas =SurfaceHolder.LockCanvas(Rect dirty)

Canvas   =Surface.lockCanvas(Rect dirty)
```

5、Demo小程序

共有两个class效果图如下，具体看代码和注释。

进入程序，执行

```java
public void surfaceCreated(SurfaceHolder holder)
```

然后执行

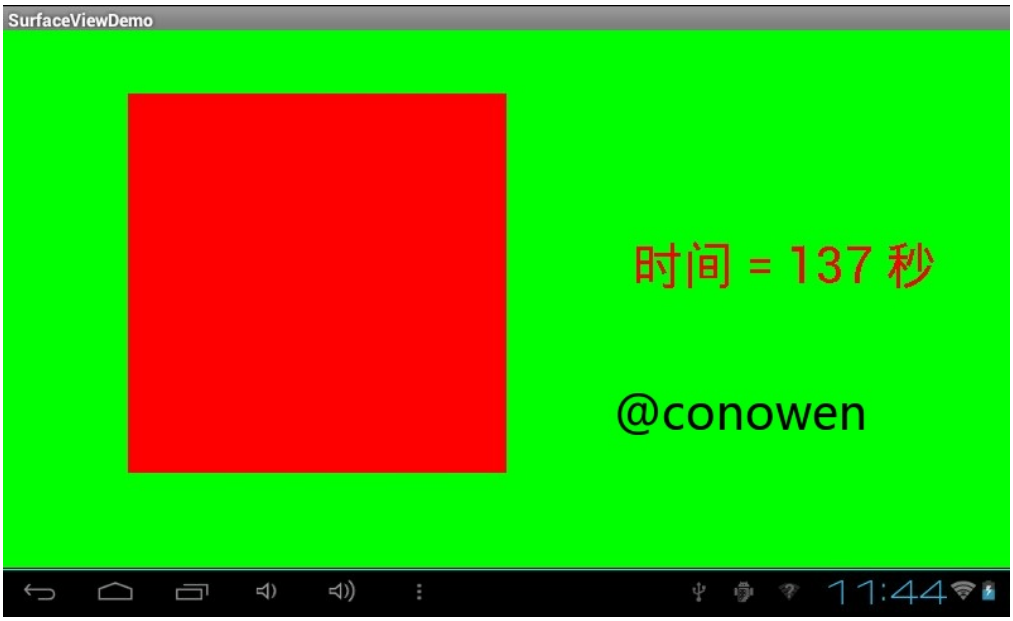
```java
public void surfaceChanged(SurfaceHolder holder, int format, int width,  int height)
```

退出程序调用surfaceDestroyed，释放资源。

```java
public void surfaceDestroyed(SurfaceHolder holder)
```

/***/

效果图：



@MySurfaceView.java

```java
/*
 * author: conowen
 * e-mail: conowen@hotmail.com
 * date  :  2012.8.4
 */
package com.conowen.SurfaceViewDemo;

import android.app.Activity;
import android.os.Bundle;

public class SurfaceViewDemoActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // setContentView(R.layout.main);
        setContentView(new MySurfaceView(this));

    }
}
```

@SurfaceViewDemoActivity.java

[java]

```java
/*
 * author: conowen
 * e-mail: conowen@hotmail.com
 * date  :  2012.8.4
 */
package com.conowen.SurfaceViewDemo;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView implements
        SurfaceHolder.Callback {

    private String TAG = "conowen";

    private SurfaceHolder sfh;
    private boolean ThreadFlag;
    private int counter;
    private Canvas canvas;

    private Thread mThread = new Thread(new Runnable() {

        @Override
        public void run() {
            // TODO Auto-generated method stub
            while (ThreadFlag) {

                // 锁定画布，得到Canvas对象
                canvas = sfh.lockCanvas();

                // 设定Canvas对象的背景颜色
                canvas.drawColor(Color.GREEN);

                // 创建画笔
                Paint p = new Paint();
                // 设置画笔颜色
                p.setColor(Color.RED);
                // 设置文字大小
                p.setTextSize(40);

                // 创建一个Rect对象rect
                // public Rect (int left, int top, int right, int bottom)
                Rect rect = new Rect(100, 50, 400, 350);
                // 在canvas上绘制rect
                canvas.drawRect(rect, p);
                // 在canvas上显示时间
                // public void drawText (String text, float x, float y, Paint
                // paint)
                canvas.drawText("时间 = " + (counter++) + " 秒", 500, 200, p);

                if (canvas != null) {
                    // 解除锁定，并提交修改内容，更新屏幕
                    sfh.unlockCanvasAndPost(canvas);
                }
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }

    });

    public MySurfaceView(Context context) {
        super(context);
        // TODO Auto-generated constructor stub

        // 通过SurfaceView获得SurfaceHolder对象
        sfh = this.getHolder();
```

```
78.
79.         // 为SurfaceHolder添加回调结构SurfaceHolder.Callback
80.         sfh.addCallback(this);
81.
82.     }
83.
84.     @Override
85.     public void surfaceChanged(SurfaceHolder holder, int format, int width,
86.             int height) {
87.         // TODO Auto-generated method stub
88.         Log.i(TAG, "surfaceChanged");
89.
90.     }
91.
92.     @Override
93.     public void surfaceCreated(SurfaceHolder holder) {
94.         // TODO Auto-generated method stub
95.         Log.i(TAG, "surfaceCreated");
96.         counter = 0;
97.         ThreadFlag = true;
98.         mThread.start();
99.
100.    }
101.
102.    @Override
103.    public void surfaceDestroyed(SurfaceHolder holder) {
104.        // TODO Auto-generated method stub
105.        Log.i(TAG, "surfaceDestroyed");
106.        ThreadFlag = false;
107.
108.    }
109.
110. }
```

上一篇 C语言指针操作、内存操作、字符串函数之备忘录（更新中~）

下一篇 Android显示系统之Pixel、Bitmap、Drawable、Canvas、Paint和Matrix之间的联系

我的同类文章

Android学习笔记（23）

- Android学习笔记(5)————SQLite的介绍与相...
- Android学习笔记(11)————Android的私人文...
- Android学习笔记(19)————实现一个记住密码...
- Android学习笔记(6)————Android的线程与进程
- Android学习笔记(18)————利用tts开发一个an...
- Android学习笔记(16)————Android的XML解...

主题推荐 android color surface 图形 rgb

猜你在找

查看评论