

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多 ▼](#)

[您还未登录！](#) [登录](#) [注册](#)

technical

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

Android中的Surface和SurfaceView

博客分类：

- [android](#)

[androidsurfacesurfaceViesurfaceHolder](#)

Android中的Surface和SurfaceView

一、什么是Surface

简单的说Surface对应了一块屏幕缓冲区，每个window对应一个Surface，任何View都要画在Surface的Canvas上（后面有原因解释）。传统的view共享一块屏幕缓冲区，所有的绘制必须在UI线程中进行。

在SDK的文档中，对Surface的描述是这样的：“Handle onto a raw buffer that is being managed by the screen compositor”，翻译成中文就是“由屏幕显示内容合成器(screen compositor)所管理的原始缓冲区的句柄”，这句话包括下面两个意思：

1、通过Surface（因为Surface是句柄）就可以获得原生缓冲器以及其中的内容。就像在C++语言中，可以通过一个文件的句柄，就可以获得文件的内容一样。

2、原始缓冲区（a raw buffer）是用于保存当前窗口的像素数据的。

引申地，可以认为Android中的Surface就是一个用来画图形（graphics）或图像（image）的地方。

根据Java方面的常规知识，我们知道通常画图是在一个Canvas对象上面进行的，由此，可以推知一个Surface对象中应该包含有一个Canvas（画布）对象。因此，在前面提及的两个意思的基础上，可以再加上一条：

3、Surface中有一个Canvas成员，专门用于画图的。

由以上的概括，我们总结如下：Surface中的Canvas成员，是专门用于供程序员画图的场所，就像黑板一样；其中的原始缓冲区是用来保存数据的地方；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、原始缓冲区以及其它方面的内容。

Surface是用来管理数据的。（句柄）

二、什么是SurfaceView

说SurfaceView是一个View也许不够严谨，然而从定义中`public class SurfaceView extends View{.....}`显示SurfaceView确实是派生自View，但是SurfaceView却有自己的Surface，请看SurfaceView的源码：

```
1.  if (mWindow == null) {  
2.      mWindow = new MyWindow(this);  
3.      mLayout.type = mWindowType;  
4.      mLayout.gravity = Gravity.LEFT|Gravity.TOP;  
5.      mSession.addWithoutInputChannel(mWindow, mWindow.mSeq, mLayout,  
6.      mVisible ? VISIBLE : GONE, mContentInsets);  
7.  }
```

很明显，每个SurfaceView创建的时候都会创建一个MyWindow，`new MyWindow(this)`中的this正是SurfaceView自身，因此将SurfaceView和window绑定在一起，由第一部分我们知道，一个window对应一个Surface，因此SurfaceView也就内嵌了一个自己的Surface，可以认为SurfaceView是用来控制Surface中View的位置和尺寸的。

SurfaceView就是展示Surface中数据的地方，同时可以认为SurfaceView是用来控制Surface中View的位置和尺寸的。

大家都知道，传统View及其派生类的更新只能在UI线程，然而UI线程还同时处理其他交互逻辑，这就无法保证View更新的速度和帧率了，而SurfaceView可以用独立的线程进行绘制，因此可以提供更高的帧率，例如游戏，摄像头取景等场景就比较适合SurfaceView来实现。

三、什么是SurfaceHolder

SurfaceHolder是一个接口，其作用就像一个关于Surface的监听器，提供访问和控制SurfaceView内嵌的Surface 相关的方法。它通过三个回调方法，让我们可以感知到Surface的创建、销毁或者改变。

在SurfaceView中有一个方法getHolder，可以很方便地获得SurfaceView内嵌的Surface所对应的监听器接口SurfaceHolder。

除下面将要提到的SurfaceHolder.Callback外，SurfaceHolder还提供了很多重要的方法，其中最重要的就是：

- 1、abstract void addCallback(SurfaceHolder.Callback callback)：为SurfaceHolder添加一个SurfaceHolder.Callback回调接口。
- 2、abstract Canvas lockCanvas()：获取一个Canvas对象，并锁定之。所得到的Canvas对象，其实就是Surface中一个成员。
- 3、abstract Canvas lockCanvas(Rect dirty)：同上。但只锁定dirty所指定的矩形区域，因此效率更高。
- 4、abstract void unlockCanvasAndPost(Canvas canvas)：当修改Surface中的数据完成后，释放同步锁，并提交改变，然后将新的数据进行展示，同时Surface中相关数据会被丢失。

2、3、4中的同步锁机制的目的，就是为了在绘制的过程中，Surface中的数据不会被改变。**lockCanvas是为了防止同一时刻多个线程对同一canvas写入。**

总结：从设计模式的高度来看，Surface、SurfaceView和SurfaceHolder实质上就是广为人知的MVC，即Model-View-Controller。Model就是模型的意思，或者说是数据模型，或者更简单地说就是数据，也就是这里的Surface；View即视图，代表用户交互界面，也就是这里的SurfaceView；SurfaceHolder很明显可以理解为MVC中的Controller（控制器）。

四、什么是SurfaceHolder.Callback

SurfaceHolder.Callback主要是当底层的Surface被创建、销毁或者改变时提供回调通知，由于绘制必须在Surface被创建后才能进行，因此SurfaceHolder.Callback中的surfaceCreated 和surfaceDestroyed 就成了绘图处理代码的边界。

SurfaceHolder.Callback中定义了三个接口方法：

1、abstract void [surfaceChanged](#)([SurfaceHolder](#) holder, int format, int width, int height)：当surface发生任何结构性的变化时（格式或者大小），该方法就会被立即调用。

2、abstract void [surfaceCreated](#)([SurfaceHolder](#) holder)：当surface对象创建后，该方法就会被立即调用。

3、abstract void [surfaceDestroyed](#)([SurfaceHolder](#) holder)：当surface对象在将要销毁前，该方法会被立即调用。

五、实例演示

下面，我们通过一个非常简单例子来实际感受一下，请留意代码中的注释：

1、在Eclipse中创建一个Android Project项目TestSurfaceView，并选择生成缺省的Activity TestSurfaceViewActivity

2、创建一个绘制线程如下：

```
1. import android.graphics.Canvas;
2. import android.graphics.Color;
3. import android.graphics.Paint;
4. import android.graphics.Rect;
5. import android.view.SurfaceHolder;
6.
7. // 绘制线程
8. public class MyThread extends Thread
9. {
10.     private SurfaceHolder holder;
11.     private boolean run;
12.
13.     public MyThread(SurfaceHolder holder)
14.     {
15.         this.holder = holder;
16.         run = true;
17.     }
18.
19.     @Override
20.     public void run()
21.     {
22.         int counter = 0;
23.         Canvas canvas = null;
24.         while(run)
25.         {
26.             // 具体绘制工作
27.             try
28.             {
```

```
29. // 获取Canvas对象, 并锁定之
30. canvas= holder.lockCanvas();
31.
32. // 设定Canvas对象的背景颜色
33. canvas.drawColor(Color.WHITE);
34.
35. // 创建画笔
36. Paint p = new Paint();
37. // 设置画笔颜色
38. p.setColor(Color.BLACK);
39. // 设置文字大小
40. p.setTextSize(30);
41.
42. // 创建一个Rect对象rect
43. Rect rect = new Rect(100, 50, 380, 330);
44. // 在canvas上绘制rect
45. canvas.drawRect(rect,p);
46. // 在canvas上显示时间
47. canvas.drawText("Interval = " + (counter++) + " seconds.", 100, 410, p);
48. Thread.sleep(1000);
49. }
50. catch(Exception e)
51. {
52.     e.printStackTrace();
53. }
54. finally
55. {
56.     if(canvas != null)
57.     {
58.         // 解除锁定, 并提交修改内容
59.         holder.unlockCanvasAndPost(canvas);
60.     }
61. }
62. }
63. }
64.
65. public boolean isRun()
```

```
66.     {
67.         return run;
68.     }
69.
70.     public void setRun(boolean run)
71.     {
72.         this.run = run;
73.     }
74. }
```

3、自定义一个SurfaceView类如下：

```
1. import android.content.Context;
2. import android.view.SurfaceHolder;
3. import android.view.SurfaceView;
4.
5. public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback
6. {
7.     private SurfaceHolder holder;
8.     private MyThread myThread;
9.
10.    public MySurfaceView(Context context)
11.    {
12.        super(context);
13.
14.        // 通过SurfaceView获得SurfaceHolder对象
15.        holder = getHolder();
16.
17.        // 为holder添加回调结构SurfaceHolder.Callback
18.        holder.addCallback(this);
19.    }
```

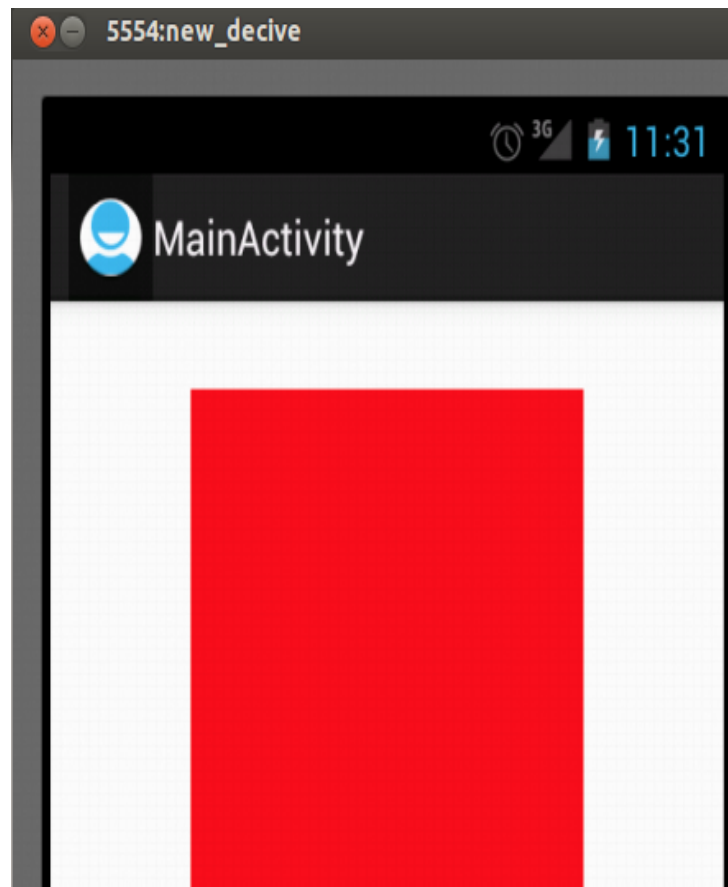


```
20.         // 创建一个绘制线程，将holder对象作为参数传入，这样在绘制线程中就可以获得holder
21.         // 对象，进而在绘制线程中可以通过holder对象获得Canvas对象，并在Canvas上进行绘制
22.         myThread = new MyThread(holder);
23.     }
24.
25.     // 实现SurfaceHolder.Callback接口中的三个方法，都是在主线程中调用，而不是在绘制线程中调用的
26.     @Override
27.     public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
28.     {
29.     }
30.
31.     @Override
32.     public void surfaceCreated(SurfaceHolder holder)
33.     {
34.         // 启动线程。当这个方法调用时，说明Surface已经有效了
35.         myThread.setRun(true);
36.         myThread.start();
37.     }
38.
39.     @Override
40.     public void surfaceDestroyed(SurfaceHolder holder)
41.     {
42.         // 结束线程。当这个方法调用时，说明Surface即将要被销毁了
43.         myThread.setRun(false);
44.     }
45. }
```

4、修改TestSurfaceViewActivity.java代码，使之如下：

1. **import** android.app.Activity;
2. **import** android.os.Bundle;

```
3.  
4. public class TestSurfaceViewActivity extends Activity  
5. {  
6.     @Override  
7.     public void onCreate(Bundle savedInstanceState)  
8.     {  
9.         super.onCreate(savedInstanceState);  
10.        //setContentView(R.layout.main);  
11.        setContentView(new MySurfaceView(this));  
12.    }  
13. }
```





运行结果：

android view, surfaceview, glsurfaceview的区别：

答：SurfaceView是从View基类中派生出来的显示类，直接子类有GLSurfaceView和VideoView，可以看出GL和视频播放以及Camera摄像头一般均使用SurfaceView

SurfaceView和View最本质的区别在于，surfaceView是在一个新起的单独线程中可以重新绘制画面而View必须在UI的主线程中更新画面。



那么在UI的主线程中更新画面 可能会引发问题，比如你更新画面的时间过长，那么你的主UI线程会被你正在画的函数阻塞。那么将无法响应按键，触屏等消息。

当使用surfaceView 由于是在新的线程中更新画面所以不会阻塞你的UI主线程。但这也带来了另外一个问题，就是事件同步。比如你触屏了一下，你需要surfaceView中thread处理，一般就需要有一个event queue的设计来保存touch event，这会稍稍复杂一点，因为涉及到线程同步。

所以基于以上，根据游戏特点，一般分成两类。

1)被动更新画面的。比如棋类，这种用view就好了。因为画面的更新是依赖于 onTouch 来更新，可以直接使用 invalidate。因为这种情况下，这一次Touch和下一次的Touch需要的时间比较长些，不会产生影响。

2)主动更新。比如一个人在一直跑动。这就需要有一个单独的thread不停的重绘人的状态，避免阻塞main UI thread。所以显然view不合适，需要surfaceView来控制。

分享到:  

[android软键盘的用法总结](#) | [Android Dialog使用](#)

- 2013-10-31 14:05
- 浏览 8544
- [评论\(6\)](#)
- 分类:[移动开发](#)
- [相关推荐](#)

评论

6 楼 [hnraysir](#) 2015-07-02

赞，必须要顶顶顶！

5 楼 [asdf_2012](#) 2014-09-15

必须顶，好文章。简洁


4 楼 [libing1991](#) 2014-05-26

讲的很详细，对于surface，surfaceview,surfaceholder三个类的概念变清晰了，特别是MVC结构，谢谢！！

3 楼 [参考人物](#) 2014-04-09



2 楼 [wxf04125](#) 2014-03-28

特地登陆来定一下，谢谢！

1 楼 [fengzhonghun102](#) 2013-11-07

好文章啊！必须要顶顶顶！

发表评论



[您还没有登录,请您登录后再发表评论](#)



miaowei

- 浏览: 137517 次
- 性别:
- 来自: 北京
- 我现在离线

最近访客

[更多访客>>](#)



[tingchan](#)



[lxuyang210](#)



[Assistne](#)



[tcxdawn](#)

文章分类

- [全部博客 \(189\)](#)
- [spring \(4\)](#)
- [jFreeChart \(1\)](#)
- [oracle \(3\)](#)
- [tomcat \(1\)](#)
- [mysql \(2\)](#)
- [jbpm \(2\)](#)
- [jQuery \(4\)](#)
- [财经 \(3\)](#)
- [php \(1\)](#)
- [linux \(1\)](#)
- [ExtJs \(2\)](#)
- [android \(90\)](#)
- [生活 \(5\)](#)
- [Json \(2\)](#)
- [html \(4\)](#)
- [java \(24\)](#)
- [other \(15\)](#)
- [struts 2 \(3\)](#)
- [hibernate \(3\)](#)
- [Lucene \(2\)](#)
- [Thread \(1\)](#)
- [javaScript \(2\)](#)
- [weblogic \(1\)](#)
- [T-SQL \(3\)](#)
- [Ant \(1\)](#)
- [memcached \(6\)](#)
- [apache \(1\)](#)
- [android 事件处理 \(0\)](#)
- [git \(1\)](#)
- [设计模式 \(1\)](#)

社区版块

- [我的资讯 \(0\)](#)

- [我的论坛](#) (0)
- [我的问答](#) (1)

存档分类

- [2015-08](#) (1)
- [2015-07](#) (1)
- [2015-06](#) (1)
- [更多存档...](#)

最新评论

- [hnraysir](#): 赞，必须要顶顶顶！
[Android中的Surface和SurfaceView](#)
- [hety163](#): 如果你设置的是一个viewgroup的ontouch，想判断不 ...
[一个view如何同时响应onTouch和onClick事件](#)
- [yue_670176656](#): [flash=200,200][[/flash]][img][ur ...
[九宫格手势密码案例](#)
- [asdf_2012](#): 必须顶，好文章。简洁
[Android中的Surface和SurfaceView](#)
- [zxw13651485](#): 自定义键盘输入，这个demo不错，就是不知道能否在所有主流手机 ...
[android软键盘的用法总结](#)

声明：ITeye文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2016 ITeye.com. All rights reserved. [京ICP证110151号 京公网安备110105010620]

12