

Damon_tong 专栏 专注技术。。。。

目录视图

摘要视图

RSS 订阅

个人资料



Damon_tong

访问： 244357次

积分： 2874

等级：

排名： 第7039名

原创： 63篇 转载： 25篇

译文： 0篇 评论： 57条

[【免费公开课】解码皮肤美化算法](#) [学院周年礼-顶尖课程钜惠呈现](#) [当讲师？爱学习？投票攒课吧](#) [CSDN 2015博客之星评选结果公布](#)

Android SurfaceView 详解(一)

标签： [android](#) [ui](#) [thread](#) [游戏](#) [框架](#) [图形](#)

2012-05-25 13:47

5560人阅读

[评论\(1\)](#)[收藏](#)[举报](#)分类： [Android \(48\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

最近有接触到SurfaceView，各种不熟悉，在看完各路大神的Blog

不少。做个小结，与大家共同进步



一、先说一下：View类和SurfaceView类

(1.)View类：

文章搜索

文章分类

[Android](#) (49)[java](#) (21)[随笔](#) (6)[Flex](#) (1)[ssh](#) (0)[jQuery](#) (3)[SQL/Oracle](#) (0)[指尖上的Android系列之项目实战](#) (8)[指尖上的Android系列之源码学习](#) (1)[Solr](#) (1)

文章存档

[2014年01月](#) (1)[2013年05月](#) (1)[2013年04月](#) (3)[2012年10月](#) (1)[2012年08月](#) (4)

展开

View 类是Android 的一个超类，每个**View**都有一个用于绘画的画布，这个画布可以进行任意的扩展。

当需要绘制复杂的图像或者对程序的执行效率要求比较高的时候，**View**并不能满足需求。**View**是Widget框架下的一个直接拖拉的控件产物。例如：当需要双缓冲来显示，直接访问画布canvas，这些都导致了我们需要比**View**更加强大的**SurfaceView**。

View：显示视图，内置画布，提供图形绘制函数，触屏时间、按键事件函数等，必须在UI主线程内更新画面，速度较慢。

(2.)SurfaceView类：(API)

SurfaceView 是**View**的继承类，这个视图里内嵌了一个专门用于绘制的**Surface**。可以控制这个**Surface**的格式和尺寸。**SurfaceView**控制这个**Surface**的绘制位置。

SurfaceView：基于view视图进行拓展的视图类，更适合2D游戏的开发；是view的子类，类上使用双缓冲机制，在新的线程中跟新画面所以新界面速度比view快。

Surface是纵深排序的，这说明它总在自己所在的窗口的后面。**SurfaceView** 提供了一个可见区域，只有在这个可见区域内的**surface**部分内容才可见，可见区域外部部分不可见。**surface**的排版显示受到视图层级关系的影响，它的兄弟节点会在顶端显示。这意味者**surface**的内容会被它的兄弟视图遮挡，这一特性可以用来放置遮盖物

(overlays) (例如：文本和按钮等控件)。但是，当**surface**上面有透明控件时，它的每次变化都会引起框架重新计算它和顶层控件的透明效果，这会影响性能。可以通过**SurfaceHolder**接口访问这个**surface**，**getHolder()**方法可以得到这个接口。

Surfaceview变的可见时，**surface**被创建，**surfaceView**隐藏前，**surface**被毁灭。这样可以节省资源。**surface**创建：**surfaceCreated(SurfaceHolder)**和**surfaceDestroyed(SurfaceHolder)**。

SurfaceView的核心提供了两个线程：UI线程和渲染线程。应该注意的是：

a.所有的**SurfaceView**和**SurfaceHolder.Callback**的方法都应该在UI线程里调用，一般来说就是应用程序的主线程。渲染线程所要访问的各种变量应该做同步处理。

b.由于**surface**可能被销毁，它只在**SurfaceHolder.Callback.surfaceCreated()**和

SurfaceHolder.Callback.surfaceDestroyed()之间有效，所以要确保渲染线程访问的是合法有效地**surface**。

阅读排行

- [onTouchEvent方法的使用](#) (43164)
- [Android中的Adapter 详解](#) (21158)
- [在Struts2中jsp前台传值到](#) (11217)
- [JAVA中HashMap和Hasht](#) (9849)
- [onSaveInstanceState和c](#) (7884)
- [Android SurfaceView 详解](#) (6769)
- [Android中的Adapter 详解](#) (6402)
- [Android中使用assets下载](#) (6071)
- [面向对象程序设计与面向](#) (5633)
- [Android SurfaceView 详解](#) (5560)

评论排行

- [onTouchEvent方法的使用](#) (10)
- [指尖上的Android之实战篇](#) (4)
- [Android SurfaceView 详解](#) (4)
- [礼拜天的遐想](#) (3)
- [面向对象程序设计与面向](#) (3)
- [指尖上的Android之实战篇](#) (3)
- [码农+码农=码农？](#) (3)
- [Android中的Adapter 详解](#) (3)
- [JAVA中HashMap和Hasht](#) (2)
- [Android中的Adapter 详解](#) (2)

二、SurfaceView类 和View类的区别

SurfaceView 和View的最本质的区别在于，surfaceView是在一个在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，触摸等消息。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，但是这也会有另外一个问题，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，这样就会有点复杂了。

View：必须在UI的主线程中更新画面，用于被动更新画面。

surfaceView：UI线程和子线程中都可以。在一个新启动的线程中重新绘制画面，主动更新画面。

所以在游戏的应用上，根据游戏的特点，一般分为两类：

a. 被动更新画面的。比如棋类，这种用view就好。因为画面的跟新依赖于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Touch需要的时间比较长些，不会产生

影响。

b.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不停地重绘人的转台，避免阻塞mian UI Thread。所以显然view 不适合，需要surfaceView来控制。

上一篇 [在Canvas中利用Path绘制基本图形](#)

下一篇 [Android SurfaceView 详解\(二\)](#)

顶 踩