Android：SurfaceView 的使用（附代码模板）

涤生_Woo 关注
1 2017.05.23 15:48:32 字数 1,013 阅读 3,739

前言

摘自《Android群英传》

> Android提供了View进行绘图处理，View可以满足大部分的绘图需求，但在某些时候也会心有余而力不足。我们知道，View通过刷新来重绘视图，Android 系统通过发出VSYNC信号来进行屏幕的重绘，刷新的时间间隔为16ms。如果在16ms内View完成了你所需要执行的所有操作，那么用户在视觉上就不会产生卡顿的感觉；而如果执行的操作逻辑太多，特别是需要频繁刷新的界面上，例如游戏界面，就会不断阻塞主线程，从而导致画面卡顿。很多情况下，在自定义View的log中会看到如下的警告：
>
> `“Skipped 47 frames! The application may be doing too much work on its main thread.”`
>
> 为了避免这一问题的产生，Android系统提供了SurfaceView组件。

View 和 SurfaceView 的区别

- View 主要适用于主动更新的情况下，而 SurfaceView 主要适用于被动更新，例如频繁地刷新。
- View 在主线程中对画面进行刷新，而 SurfaceView 通常会通过一个子线程来进行页面的刷新。
- View 在绘图时没有使用双缓冲机制，而 SurfaceView 在底层实现机制中就已经实现了双缓冲机制。
  总结就是，如果你的自定义View需要频繁刷新，或者刷新时数据处理量比较大，那么你就可以考虑使用 SurfaceView 来取代 View 了。

SurfaceView 的使用

SurfaceView 的使用虽然比 View 复杂，**但是 SurfaceView 在使用时，有一套使用的模板代码，大部分的 SurfaceView 绘图操作都可以套用这样的模板代码来进行编写。**因此 SurfaceView 的使用会更加简单。

创建一个 SurfaceView 的模板

1、 创建 SurfaceView

创建自定义的 MySurfaceView 继承自 SurfaceView ，并实现两个接口——SurfaceHolder.Callback, Runnable，同时实现其接口方法，如下：

```
1  public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback, Runn
2
3      @Override
4      public void surfaceCreated(SurfaceHolder holder) {
5      }
6
7      @Override
8      public void surfaceChanged(SurfaceHolder holder, int format, int width, int height
9      }
10
11     @Override
12     public void surfaceDestroyed(SurfaceHolder holder) {
```

```
    }

    @Override
    public void run() {
    }
}
```

2、初始化 SurfaceView

在自定义 SurfaceView 的构造方法中，需要对 SurfaceView 进行初始化。通常需要定义以下三个成员变量，如下：

```
private SurfaceHolder mHolder;
//用于绘图的canvas
private Canvas mCanvas;
//子线程标志位
private boolean mIsDrawing;
```

初始化方法就是初始化一个 SurfaceHolder 对象并注册 SurfaceHolder 的回调方法，如下：

```
mHolder = getHolder();
mHolder.addCallback(this);
```

另外两个成员变量——Canvas 和标志位。使用 Canvas 来进行绘图；使用标志位来控制之前提到的用于绘制的子线程。

3、使用 SurfaceView

通过 SurfaceHolder 对象的 lockCanvas() 方法就可以获得当前的 Canvas 绘图对象。接下来就可以与在 View 中进行的绘制操作一样进行绘制了。**这里需要注意，获取到的 Canvas 对象还是继续上次的 Canvas 对象，而不是一个新的 Canvas 对象。**因此，之前的绘图操作都会被保留。如果需要擦除，则可以在绘制前，通过 drawColor() 方法来进行清屏操作。

绘制时，充分利用 SurfaceView 的三个回调方法，在 surfaceCreated() 方法里开启子线程进行绘制，而子线程使用一个 while (mIsDrawing) {} 的循环来不停地进行绘制，而在绘制的具体逻辑中，通过 lockCanvas() 方法来获取 Canvas 对象来绘制，并通过 unlockCanvasAndPost(mCanvas) 方法对画布内容进行提交。整个 SurfaceView 模板代码如下：

```
import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

/**
 * Created by Deeson on 2017/5/23.
 */
public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback, Runn

    private SurfaceHolder mHolder;
    //用于绘图的canvas
    private Canvas mCanvas;
    //子线程标志位
    private boolean mIsDrawing;

    public MySurfaceView (Context context) {
        super(context);
        init();
    }

    public MySurfaceView (Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public MySurfaceView (Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
```

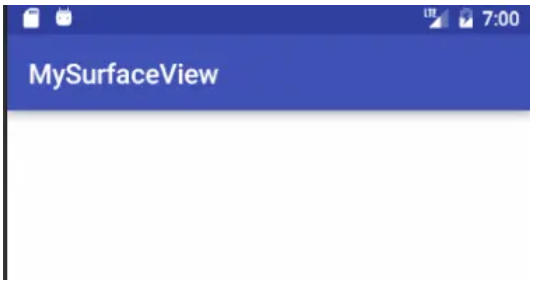
```
    }

    private void init() {
        mHolder = getHolder();
        mHolder.addCallback(this);
        setFocusable(true);
        setFocusableInTouchMode(true);
        this.setKeepScreenOn(true);
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        mIsDrawing = true;
        new Thread(this).start();
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        mIsDrawing = false;
    }

    @Override
    public void run() {
        while (mIsDrawing) {
            draw();
        }
    }

    private void draw() {
        try {
            mCanvas = mHolder.lockCanvas();
            //draw something
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (null != mCanvas) {
                mHolder.unlockCanvasAndPost(mCanvas);
            }
        }
    }
}
```

以上代码基本可以满足大部分 SurfaceView 的绘图需求，**唯一需要注意的是在绘制方法中，将 `mHolder.unlockCanvasAndPost(mCanvas);` 方法放到 finally 代码块中，保证每次都能将内容提交。**

最后

关于 SurfaceView 的实例演练，可以看看我的这篇文章 Android：贝塞尔曲线原理分析

按照惯例，需要送上 demo 下载，如下 gif 所示：



mySurfaceView.gif