# 咕咚首页滑动卡顿分析

先说结论：由于手指抬起UP事件的时候，有做耗时操作，导致VSYNC消息阻塞，整体UI就丢帧。

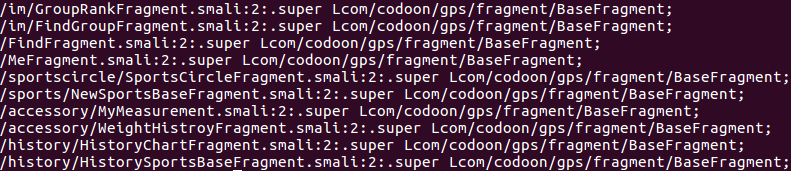
接着一步一步讲一下我的分析过程。

## 一、反编译并添加基本的Log



没找到任何类继承这个类，那么这个就是运动板块所使用的Fragment了。

这个类继承自com/codoon/gps/fragment/BaseFragment。



这些类也继承了BaseFragment。那么来看看咕咚的BaseFragment做了些什么？有点好奇。

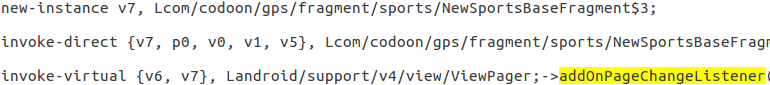
打开smali文件，此类继承自原生的Fragment并实现了View$OnTouchListener接口，用来辅助处理View的Touch事件。包括自动打开关闭VelocityTracker，监测速度，如果检测到向左或者向右滑动，则发出广播，并通知监听器mOnSlideListener。

继续分析NewSportsBaseFragment，主要关注以下几点：

1. UI的布局、切换等
2. 左右滑动效果怎么实现的，分析为什么会出现近乎跳变式滑动

首先是onCreateView，这里面有加载之前分析的资源文件0x7f0304a1(sports\_pre\_start\_layout.xml)。并且创建了一个mMainBGDrawable: GradientDrawable，这个就是用来在滑动时显示渐变效果的。

这里创建了一个内部类实例，实现了OnPageChangeListener。里面做了页面滑动的相关动画操作。



NewSportsBaseFragment$3.smali里面实现了三个接口方法：

.method public onPageScrollStateChanged(I)V

.method public onPageScrolled(IFI)V

.method public onPageSelected(I)V

在代码中里可以看到：

new-instance v2, Lcom/codoon/gps/fragment/sports/RotatePageTransformer;

invoke-virtual {v0, v1, v2}, Landroid/support/v4/view/ViewPager;->setPageTransformer(ZLandroid/

通过设置PageTransFormer来实现ViewPager翻页效果。

ViewPager在回调transformPage的时候，应该是线性的，不应该是

但实际上，除了ViewPager滑动卡顿，“开始”按钮的动画一样会卡顿。

会不是是插值器计算有问题呢？或者速度计算有问题？ViewPager有什么接口可以造成这种效果呢？

因为PageTransFormer和OnPageChangeListener同时出现同样的问题。

仔细看了一下，**上一个View会直接消失，然后下一个View以动画出来**。

那是不是onPageSelected有调用上一个View的setVisibility？

但是滑动指示器也是跳变的。

从布局来看，ViewPager之上没有有其他自定义View用来拦截Touch事件。而咕咚自己定义的ControlableViewPager只是限制了本身能不能滑动。所以按理说ViewPager本身的滑动效果是支持平滑翻页的。但为什么会出现跳变，直觉上可能是在onPageSelected没处理好，或者PageTransFormer有问题。因为点击在“运动”和“计步”之间切换，也会调用到onPageSelected，这种情况是跳变过去的，可能这一块的逻辑在Fling的时候并不适用。

从动画的关键环节.method public transformPage(Landroid/view/View;F)V入手比较好。先看ViewPager通知PageTransFormer的消息有没有跳变。

原来的代码是返回到goto\_0，代码里面有三处都是goto :goto\_0

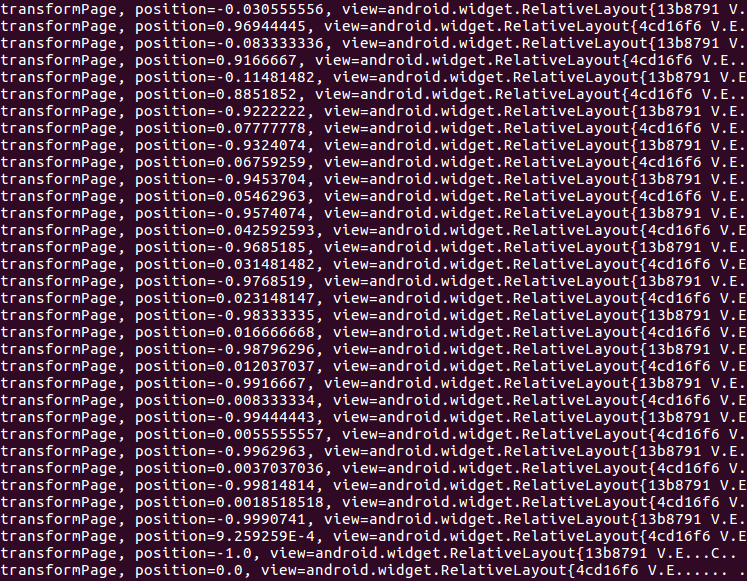


所以要加日志的话，那就将goto\_0改为我要加的日志，然后再goto: goto\_1。goto\_1就是原来的返回。

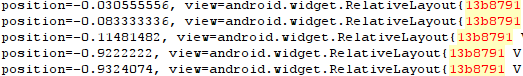


添加的内容也就是Log.d("Rotate", "transformPage, position=" + position + ", view=" + view)。

然后回编译，签名，重新安装。测试一下，滑动的时候，日志有了。



从日志来看，确实在滑动的时候，position从-0.11跳变到了-0.92.



接下来，就要找出来，为什么ViewPager传出来的值会跳变了。

ViewPager通知mPageTransformer的位置在：

protected void onPageScrolled(int position, float offset, int offsetPixels)

if (mPageTransformer != null) {

final int scrollX = getScrollX();

final int childCount = getChildCount();

for (int i = 0; i < childCount; i++) {

final View child = getChildAt(i);

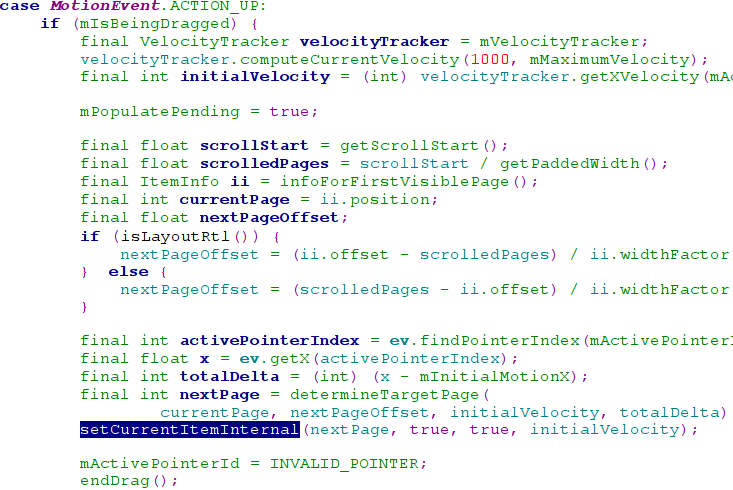
final LayoutParams lp = (LayoutParams) child.getLayoutParams();

if (lp.isDecor) continue;

final float transformPos = (float) (child.getLeft() - scrollX) / getClientWidth();

mPageTransformer.**transformPage**(child, transformPos); //依次通知每个子View的变化

按照理论上说，



UP事件后，会调用setCurrentItemInternal来调转到指定Page。

void setCurrentItemInternal(int item, boolean smoothScroll, boolean always, int velocity) {

前三个参数都没什么好看的。速度的影响最大。

接着会调用scrollToItem(item, smoothScroll, velocity, dispatchSelected);

这里有两个关键的地方。

**destX** = (int) (width \* Math.max(mFirstOffset,

Math.min(curInfo.offset, mLastOffset))

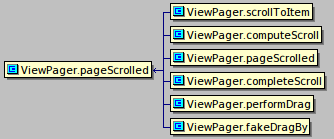
if (smoothScroll) {

**smoothScrollTo**(destX, 0, velocity);

smoothScrollTo()方法通过与mScroller和View绘制中的computeScroll()方法配合完成平滑滑动效果。

mScroller.startScroll(sx, sy, dx, dy, duration);

到时候在框架中加日志这里是重点观察的位置。

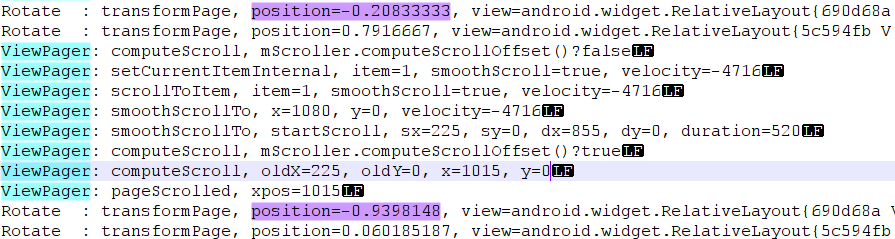


pageScrolled有这么多地方调用，但滑动然后抬起手指的时候，

## 二、在反编译的ViewPager中添加日志

接下来，就是在ViewPager中加日志了。因为咕咚使用了V4包里面的ViewPager。

所以先在V4包的ViewPager中添加相关日志，然后编译出jar包。然后反编译出smali，替换咕咚中相应的文件后回编译成apk。



可以看到就是在这两次之间，出现的跳变。通过Scroller计算出来的第一个x就是1015，太大了。

为什么Scroller有问题呢？仔细看了一下从startScroll到computeScrollOffset中间花了209ms。刷新太慢了，而这个默认的插值器又是属于初始速度很快的插值器，所以对于滑动总时长才520ms，209ms不更新，跳变就很明显。

那为什么有209ms没有刷新呢？ViewPager原生在computeScroll的时候都是会调用

ViewCompat.postInvalidateOnAnimation(this)

这样UI可以持续更新。

## 三、修改框架加Log

所以，最后分析出来，就是在UP事件的时候，做了耗时操作，导致UI阻塞了200ms左右，造成卡顿。

UP事件由ViewPager拦截并在UI线程分发，那是谁做了耗时操作呢？从之前分析来看，有两个类做了ViewPager的事件监听。一个是PageTransFormer，另一个是NewSportsBaseFragment内的内部成员OnPageChangeListener。而PageTransFormer只有一个回调transformPage，不太可能会知道UP事件。只有OnPageChangeListener会知道UP事件。

UP事件的时候，OnPageChangeListener会有两个回调：

1. onPageScrollStateChanged，UP的时候通知进入SCROLL\_STATE\_SETTLING滑动中的状态。
2. onPageSelected，UP的时候通知即将进入哪一页。

## 四、修改卡顿问题

能否通过IdleHandler解决UI分发的问题？应该也不好，总会阻塞UI。看能否另开线程。

运动圈的滑动同样有卡顿问题，考虑为同类问题。

ControlableViewPager只是限制了本身能不能滑动。这个在第一次的引导登录页面有用到。

等翻到登录页面之后，就禁用了滑动。