**ViewDragHelper**

**更详细可参考：**

**<http://www.cnblogs.com/lqstayreal/p/4500219.html>**

**https://juejin.im/entry/5768e7f1816dfa005475f2d0**

ViewDragHelper是自定义ViewGroup时处理子view拖拽交互的一个帮助类。其中对子View进行的各种拖拽操作的回调，需要实现ViewDragHelper中抽象类Callback中的方法。下面是Callback中回调方法：

|  |  |
| --- | --- |
| tryCaptureView | 传递当前触摸的子View实例，如果当前的子View需要进行拖拽移动，则在此方法中进行判断并返回true |
| clampViewPositionHorizontal | 返回子View横向的拖动范围 |
| clampViewPositionVertical | 返回子View纵向的拖动范围 |
| onEdgeTouched | ACTION\_DOWN或ACTION\_POINTER\_DOWN事件发生时如果触摸到监听的边缘会调用此方法 |
| onEdgeDragStarted | ACTION\_MOVE事件发生时，检测到开始在某些边缘有拖动的手势,并且没有锁定边缘，会调用此方法，只会调用一次 |
| onEdgeLock(int edgeFlags) | 返回true表示锁定edgeFlags对应的边缘，锁定后的那些边缘在ACTION\_MOVE事件发生时，会过滤掉不正常的拖动手势 |
| onViewDragStateChanged | 拖动状态改变时会调用此方法，状态state有STATE\_IDLE、STATE\_DRAGGING、STATE\_SETTLING三种取值。 |
| onViewPositionChanged | 正在被拖动的View或者自动滚动的View的位置改变时会调用此方法。 |
| onViewCaptured | tryCaptureViewForDrag()成功捕获到子View时会调用此方法。 |
| onViewReleased | 拖动View松手时（processTouchEvent()的ACTION\_UP）或被父View拦截事件时（processTouchEvent()的ACTION\_CANCEL）会调用此方法。 |
| getViewHorizontalDragRange | 返回给定的child在相应的方向上可以被拖动的最远距离，默认返回0。 \* ACTION\_DOWN发生时，若触摸点处的child消费了事件，并且想要在某个方向上可以被拖动，就要在对应方法里返回大于0的数。 |
| getViewVerticalDragRange | 同上 |

使用

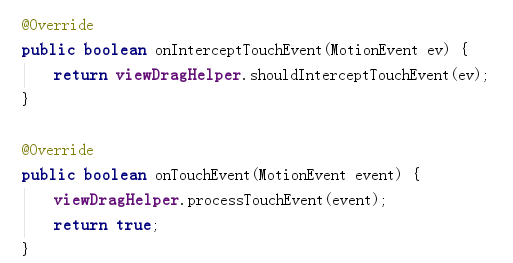
下面展示一个随手指移动的TextView的基础使用：

**第一步：**



首先ViewDragHelper的使用必须在ViewGroup的子类里使用，然后在构造方法里初始化一个实例，ViewDragHelper的初始化过程是由工厂模式生产的。

**第二步：**



初始化完实例后，将容器的事件拦截onInterceptTouchEvent和事件处理onTouchEvent里的事件交给ViewDragHelper实例处理。

**第三步：**

****

在初始化ViewDragHelper的时候我们传入的第三个参数是手势操作的回调接口的实现类，第三步就是主要就是实现这个接口。随手指移动的功能主要是实现三个方法解释见上图，更详细的说明见项目android-sources。

* **方法介绍**

**ViewDragHelper常用方法介绍**

//构造工厂方法，sensitivity用来调节mTouchSlop的值，默认一般传递1即可

sensitivity越大，mTouchSlop越小，对滑动的检测就越敏感，譬如手指move多少才算滑动，否则忽略

public static ViewDragHelper **create(ViewGroup forParent, float sensitivity, Callback cb)**

//设置允许父View的某个边缘可以用来响应托拽

相当于控制了CallBack对象的onEdgeTouched()和onEdgeDragStarted()方法是否被回调

public void **setEdgeTrackingEnabled(int edgeFlags)**

//两个传递MotionEvent的方法

public boolean **shouldInterceptTouchEvent(MotionEvent ev)**

public void **processTouchEvent(MotionEvent ev)**

//主动在父View内捕获指定的子view用于拖曳，会回调tryCaptureView()

public void **captureChildView(View childView, int activePointerId)**

//指定某个View自动滚动到指定的位置，初速度为0，可在任何地方调用

如果这个方法返回true，那么在接下来动画移动的每一帧中都会回调continueSettling(boolean)方法，直到结束

public boolean **smoothSlideViewTo(View child, int finalLeft, int finalTop)**

//以松手前的滑动速度为初值，让捕获到的子View自动滚动到指定位置，只能在Callback的onViewReleased()中 使用；如果这个方法返回true，那么在接下来动画移动的每一帧中都会回调continueSettling(boolean)方法，

直到结束

public boolean **settleCapturedViewAt(int finalLeft, int finalTop)**

//以松手前的滑动速度为初值，让捕获到的子View在指定范围内fling惯性运动，只能在Callback的 onViewReleased()中使用；如果这个方法返回true，那么在接下来动画移动的每一帧中都会回调

continueSettling(boolean)方法，直到结束

public void **flingCapturedView(int minLeft, int minTop, int maxLeft, int maxTop)**

/\*\*

\* 在整个settle状态中,这个方法会返回true，deferCallbacks决定滑动是否Runnable推迟，一般推迟

\* 在调用settleCapturedViewAt()、flingCapturedView()和smoothSlideViewTo()时，

\* 需要实现mParentView的computeScroll()方法，如下：

\* @Override

\* public void computeScroll() {

\* if (mDragHelper.continueSettling(true)) {

\* ViewCompat.postInvalidateOnAnimation(this);

\* }

\* }

\*/

public boolean **continueSettling(boolean deferCallbacks)**

//设置与获取最小速率，一般保持默认

public void **setMinVelocity(float minVel)**

public float **getMinVelocity()**

//获取当前子View所处状态

public int **getViewDragState()**

//返回可触摸反馈区域边缘大小，单位为px

public int **getEdgeSize()**

//返回当前捕获的子View，如果没有则为null

public View **getCapturedView()**

//获取当前拖曳的View的Pointer ID

public int **getActivePointerId()**

//获取最小触发拖曳动作的灵敏度差值，单位为px

public int **getTouchSlop()**

//类似ACTION\_CANCEL事件的触发调运

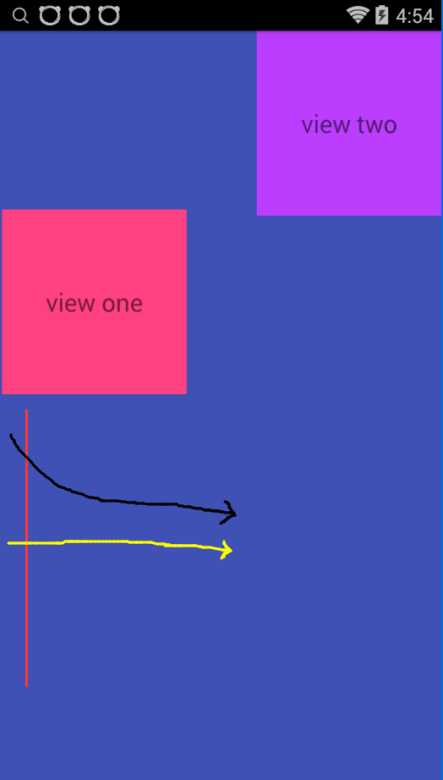
public void **cancel()**

//终止手势，结束动画滚动等，恢复初始STATE\_IDLE状态

public void **abort()**

**侧滑功能**

侧滑功能就是从左侧侧滑出菜单栏的功能，这里可以监听界面的上下左右四个方向侧滑，这里通过图片解释一下onEdgeLock方法：



如上图view one是可以随着左侧侧滑手势而向右移动的View，他能移动说明onEdgeDragStarted方法被调用。图中红线代表侧滑的有效区域，只有在这个区域内向右侧滑才能回调onEdgeDragStarted方法使view one跟随手指移动。

onEdgeLock方法的作用是锁定任意一个或者全部边缘，即可控制onEdgeDragStarted方法的回调时机。如上图假如onEdgeLock方法没有锁定左侧边缘，手指不论按黄线还是黑线的轨迹移动，view one都会随着手指移动而向右侧移动，就是说onEdgeDragStarted方法都会回调。一旦当onEdgeLock方法锁定左侧边缘，手指按照黑线轨迹移动view one就不会向右移动，也就是说onEdgeDragStarted方法不会回调了，此时就好像黑色轨迹是一条错误的滑动手势，这样通过onEdgeLock方法锁定某个边缘就可以过滤掉这个边缘上错误的侧滑手势。