# 1 速度追踪器

## 1.1 VelocityTracker（速度追踪器）

[VelocityTracker](https://developer.android.com/reference/android/view/VelocityTracker.html) 是一个跟踪触摸事件滑动速度的帮助类，用于实现flinging以及其它类似的手势。它的原理是把触摸事件 MotionEvent 对象传递给VelocityTracker的[addMovement(MotionEvent)](https://developer.android.com/reference/android/view/VelocityTracker.html" \l "addMovement(android.view.MotionEvent))方法，然后分析MotionEvent 对象在单位时间类发生的位移来计算速度。你可以使用getXVelocity() 或getXVelocity()获得横向和竖向的速率到速率时，但是使用它们之前请先调用computeCurrentVelocity(int)来初始化速率的单位 。

**例如**：computeCurrentVelocity函数的参数值为1000 。这个表示计算过去1000ms（即1s）内的速度。这个速度其实就是 （(当前的位置）-(之前的位置))/时间.如果得到的值为200就表示这1000ms内，X方向移动了200像素。  
    速度是有正负的，右划就是正的，左划就是负的，上划为负，下划为正。

## 1.2 惯性滑动

## 还有个比较常见的需求，比如我们右划了一下，view往右移动，同时希望在手指抬起来之后，view能够按照惯性继续滑动一段距离然后停止，这是就需要确认 手指抬起的时候的速度，可以在move up的事件时计算

模版写法：

1. **public** **class** TemplateView **extends** View {
2. **private** **static** **final** String DEBUG\_TAG = "TemplateVelocity";
3. **private** **static** **final** **int** V\_CONSTANT=50;
4. **private** **static** **final** **int** UNIT\_MIL\_SECOND=1000;  //单位毫秒
5. **private** VelocityTracker mVelocityTracker = **null**;
6. @Override
7. **public** **boolean** onTouchEvent(MotionEvent event) {
8. **int** index = event.getActionIndex();
9. **int** action = event.getActionMasked();
10. initVelocityTrackerIfNotExists();
11. **switch**(action) {
12. **case** MotionEvent.ACTION\_DOWN:
13. **if**(mVelocityTracker == **null**) {
14. // Retrieve a new VelocityTracker object to watch the velocity of a motion.
15. mVelocityTracker = VelocityTracker.obtain();
16. }
17. **else** {
18. // Reset the velocity tracker back to its initial state.
19. mVelocityTracker.clear();
20. }
21. // Add a user's movement to the tracker.
22. mVelocityTracker.addMovement(event);
23. **break**;
24. **case** MotionEvent.ACTION\_MOVE:
25. mVelocityTracker.addMovement(event);
26. **break**;
27. **case** MotionEvent.ACTION\_UP:
28. **case** MotionEvent.ACTION\_CANCEL:
29. **final** VelocityTracker velocityTracker = mVelocityTracker;
30. velocityTracker.computeCurrentVelocity(UNIT\_MIL\_SECOND);
31. **int** initialVelocity = (**int**) velocityTracker.getYVelocity();
33. **if** ((Math.abs(initialVelocity) > V\_CONSTANT)) {
34. //速度够快，比如翻页，速度是个常数，大约在30-100之间
35. //                    ...
36. } **else**{
37. //                    ...
38. }
39. recycleVelocityTracker();
40. **break**;
41. }
42. **return** **true**;
43. }

46. **private** **void** initVelocityTrackerIfNotExists() {
47. **if** (mVelocityTracker == **null**) {
48. mVelocityTracker = VelocityTracker.obtain();
49. }
50. }
52. **private** **void** recycleVelocityTracker() {
53. **if** (mVelocityTracker != **null**) {
54. mVelocityTracker.recycle();
55. mVelocityTracker = **null**;
56. }
57. }
58. }