Tracking Movement

This lesson describes how to track movement in touch events.

A new `onTouchEvent()` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)) is triggered with an `ACTION_MOVE` (/reference/android/view/MotionEvent.html#ACTION_MOVE) event whenever the current touch contact position, pressure, or size changes. As described in Detecting Common Gestures (detector.html), all of these events are recorded in the `MotionEvent` (/reference/android/view/MotionEvent.html) parameter of `onTouchEvent()` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)).

THIS LESSON TEACHES YOU TO

YOU SHOULD ALSO READ

- Input Events API Guide
- Sensors Overview
- Making the View Interactive
- Design Guide for Gestures
- Design Guide for Touch Feedback

TRY IT OUT

Download the sample

InteractiveChart.zip

Because finger-based touch isn't always the most precise form of interaction, detecting touch events is often based more on movement than on simple contact. To help apps distinguish between movement-based gestures (such as a swipe) and non-movement gestures (such as a single tap), Android includes the notion of "touch slop." Touch slop refers to the distance in pixels a user's touch can wander before the gesture is interpreted as a movement-based gesture. For more discussion of this topic, see Managing Touch Events in a ViewGroup (viewgroup.html#vc).

There are several different ways to track movement in a gesture, depending on the needs of your application. For example:

- The starting and ending position of a pointer (for example, move an on-screen object from point A to point B).
- The direction the pointer is traveling in, as determined by the x and y coordinates.
- History. You can find the size of a gesture's history by calling the `MotionEvent` method `getHistorySize()`. You can then obtain the positions, sizes, time, and pressures of each of the historical events by using the motion event's `getHistorical<Value>` methods. History is useful when rendering a trail of the user's finger, such as for touch drawing. See the `MotionEvent` reference for details.
- The velocity of the pointer as it moves across the touch screen.

Track Velocity

You could have a movement-based gesture that is simply based on the distance and/or direction the pointer traveled. But velocity often is a determining factor in tracking a gesture's characteristics or even deciding whether the gesture occurred. To make velocity calculation easier, Android provides the `VelocityTracker` (/reference/android/view/VelocityTracker.html) class and the `VelocityTrackerCompat` (/reference/android/support/v4/view/VelocityTrackerCompat.html) class in the Support Library (/tools/support-library/index.html). `VelocityTracker` (/reference/android/view/VelocityTracker.html) helps you track the velocity of touch events. This is useful for gestures in which velocity is part of the criteria for the gesture, such as a fling.

Here is a simple example that illustrates the purpose of the methods in the VelocityTracker (/reference/android/view/VelocityTracker.html) API:

```
public class MainActivity extends Activity {
    private static final String DEBUG_TAG = "Velocity";
        ...
    private VelocityTracker mVelocityTracker = null;
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int index = event.getActionIndex();
        int action = event.getActionMasked();
        int pointerId = event.getPointerId(index);

        switch(action) {
            case MotionEvent.ACTION_DOWN:
                if(mVelocityTracker == null) {
                    // Retrieve a new VelocityTracker object to watch the velo
                    mVelocityTracker = VelocityTracker.obtain();
                }
                else {
                    // Reset the velocity tracker back to its initial state.
                    mVelocityTracker.clear();
                }
                // Add a user's movement to the tracker.
                mVelocityTracker.addMovement(event);
                break;
            case MotionEvent.ACTION_MOVE:
                mVelocityTracker.addMovement(event);
                // When you want to determine the velocity, call
                // computeCurrentVelocity(). Then call getXVelocity()
                // and getYVelocity() to retrieve the velocity for each pointe
                mVelocityTracker.computeCurrentVelocity(1000);
                // Log velocity of pixels per second
                // Best practice to use VelocityTrackerCompat where possible.
                Log.d("", "X velocity: " +
                        VelocityTrackerCompat.getXVelocity(mVelocityTracker,
                        pointerId));
                Log.d("", "Y velocity: " +
                        VelocityTrackerCompat.getYVelocity(mVelocityTracker,
                        pointerId));
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                // Return a VelocityTracker object back to be re-used by other
                mVelocityTracker.recycle();
                break;
        }
        return true;
    }
}
```

Note: Note that you should calculate velocity after an ACTION_MOVE (/reference/android/view/MotionEvent.html#ACTION_MOVE) event, not after ACTION_UP (/reference/android

> /view/MotionEvent.html#ACTION_UP). After an ACTION_UP (/reference/android/view/MotionEvent.html#ACTION_UP), the X and Y velocities will be 0.