codepath / **android_guides** Public

Code Issues 155 Pull requests 3 Actions Projects **Wiki**

Gestures and Touch Events

Jump to bottom

Nathan Esquenazi edited this page on 7 Nov 2016 · 41 revisions

Overview

Gesture recognition and handling touch events is an important part of developing user interactions. Handling standard events such as clicks, long clicks, key presses, etc are very basic and handled in other guides. This guide is focused on handling other more specialized gestures such as:

- Swiping in a direction
- Double tapping for zooming
- Pinch to zoom in or out
- Dragging and dropping
- Effects while scrolling a list

You can see a visual guide of common gestures on the gestures design patterns guide. See the new Material Design information about the touch mechanics behind gestures too.

Usage

Handling Touches

At the heart of all gestures is the onTouchListener and the `onTouch` method which has access to MotionEvent data. Every view has an `onTouchListener` which can be specified:

```
myView.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Interpret MotionEvent data
        // Handle touch here
        return true;
    }
});
```

Each `onTouch` event has access to the MotionEvent which describe movements in terms of an **action code** and a **set of axis values**. The action code specifies the state change that occurred such as a pointer going down or up. The axis values describe the position and other movement properties:

- `getAction()` - Returns an integer constant such as `MotionEvent.ACTION_DOWN` , `MotionEvent.ACTION_MOVE` , and `MotionEvent.ACTION_UP`
- `getX()` - Returns the x coordinate of the touch event

- `getY()` - Returns the y coordinate of the touch event

Note that every touch event can be **propagated through the entire affected view hierarchy**. Not only can the touched view respond to the event but every layout that contains the view has an opportunity as well. Refer to the understanding touch events section for a detailed overview.

Handling Multi Touch Events

Note that `getAction()` normally includes information about both the action as well as the pointer index. In single-touch events, there is only one pointer (set to 0), so no bitmap mask is needed. In multiple touch events (i.e pinch open or pinch close), however, there are multiple fingers involved and a non-zero pointer index may be included when calling `getAction()`. As a result, there are other methods that should be used to determine the touch event:

- `getActionMasked()` - extract the action event without the pointer index
- `getActionIndex()` - extract the pointer index used

The events associated with other pointers usually start with `MotionEvent.ACTION_POINTER` such as `MotionEvent.ACTION_POINTER_DOWN` and `MotionEvent.ACTION_POINTER_UP`. The `getPointerCount()` on the MotionEvent can be used to determine how many pointers are active in this touch sequence.

Gesture Detectors

Within an `onTouch` event, we can then use a GestureDetector to understand gestures based on a series of motion events. Gestures are often used for user interactions within an app. Let's take a look at how to implement common gestures.

For easy gesture detection using a third-party library, check out the popular Sensey library which greatly simplifies the process of attaching multiple gestures to your views.

Double Tapping

You can enable double tap events for any view within your activity using the OnDoubleTapListener. First, copy the code for `OnDoubleTapListener` into your application and then you can apply the listener with:

```
myView.setOnTouchListener(new OnDoubleTapListener(this) {
  @Override
  public void onDoubleTap(MotionEvent e) {
    Toast.makeText(MainActivity.this, "Double Tap", Toast.LENGTH_SHORT).show();
  }
```