

Sample Code:

<https://github.com/devunwired/custom-touch-examples>

Mastering the Android Touch System

Dave Smith
@devunwired

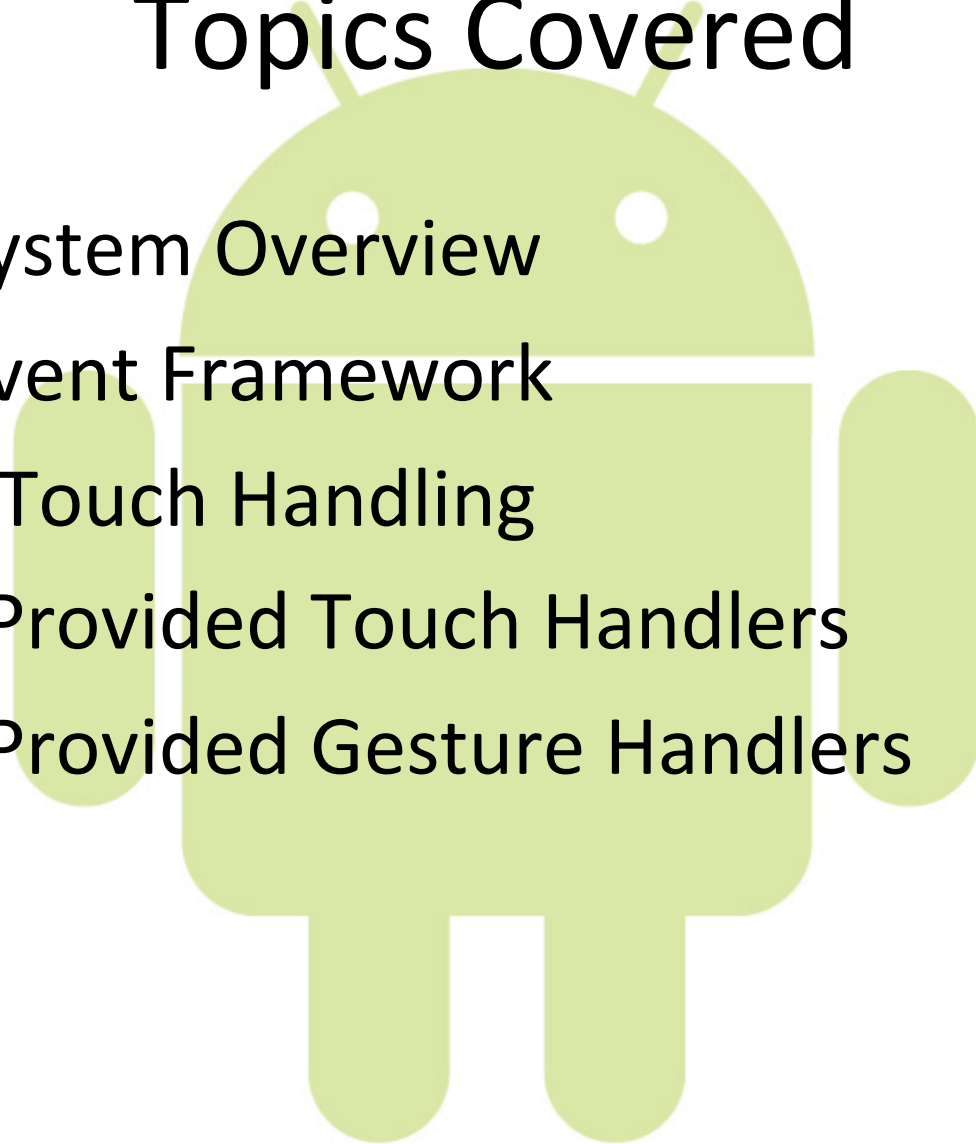
Who Is This Guy?

- Android developer since 2009
 - ROM customization for Embedded applications
- Recovering Spark Chaser
 - Embedded M2M Monitoring systems
 - P2P Radio Links
- Co-Author of Android Recipes from Apress



Topics Covered

- Touch System Overview
- Touch Event Framework
- Custom Touch Handling
- System Provided Touch Handlers
- System Provided Gesture Handlers



How Android Handles Touches



- Each user touch event is wrapped up as a MotionEvent
- Describes user's current action
 - ACTION_DOWN
 - ACTION_UP
 - ACTION_MOVE
 - ACTION_POINTER_DOWN
 - ACTION_POINTER_UP
 - ACTION_CANCEL
- Event metadata included
 - Touch location
 - Number of pointers (fingers)
 - Event time
- A “gesture” is defined as beginning with ACTION_DOWN and ending with ACTION_UP.

How Android Handles Touches



- Events start at the Activity with `dispatchTouchEvent()`
- Events flow top down through views
 - Parents (ViewGroups) dispatch events to their children
 - Can intercept events at any time
- Events flow down the chain (and back up) until consumed
 - Views must declare interest by consuming `ACTION_DOWN`
 - Further events not delivered for efficiency
- Any unconsumed events end at the Activity with `onTouchEvent()`
- Optional External `OnTouchListener` can intercept touches on any `View/ViewGroup`

How Android Handles Touches



- `Activity.dispatchTouchEvent()`
 - Always first to be called
 - Sends event to root view attached to Window
 - `onTouchEvent()`
 - Called if no views consume the event
 - Always last to be called
- `View.dispatchTouchEvent()`
 - Sends event to listener first, if exists
 - `View.OnTouchListener.onTouch()`
 - If not consumed, processes the touch itself
 - `View.onTouchEvent()`

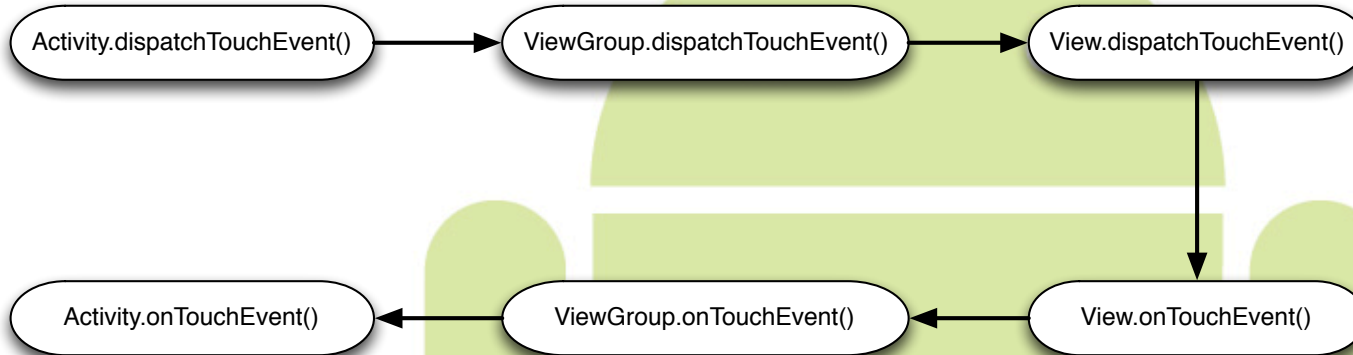
How Android Handles Touches



- `ViewGroup.dispatchTouchEvent()`
 - `onInterceptTouchEvent()`
 - Check if it should supersede children
 - Passes `ACTION_CANCEL` to active child
 - Return true once consumes all subsequent events
 - For each child view, in reverse order they were added
 - If touch is relevant (inside view), `child.dispatchTouchEvent()`
 - If not handled by previous, dispatch to next view
 - If no children handle event, listener gets a chance
 - `OnTouchListener.onTouch()`
 - If no listener, or not handled
 - `onTouchEvent()`
- Intercepted events jump over child step

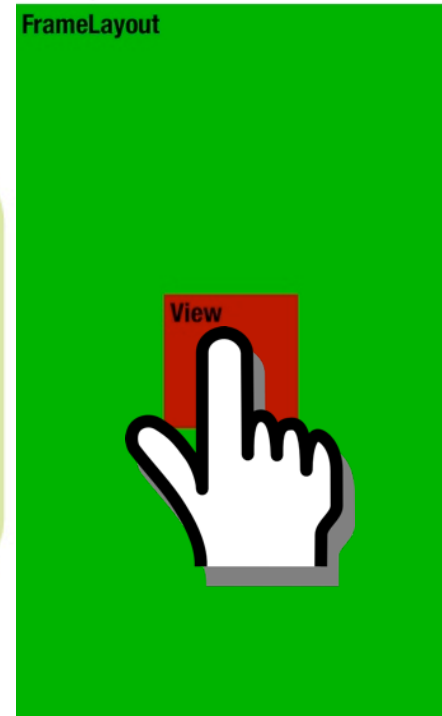
Ignorant View Example

DOWN:

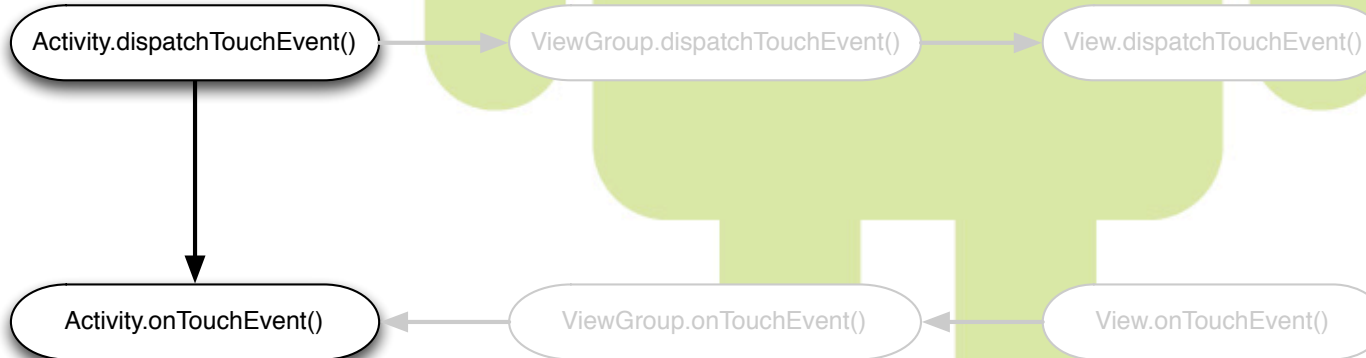


Activity

FrameLayout

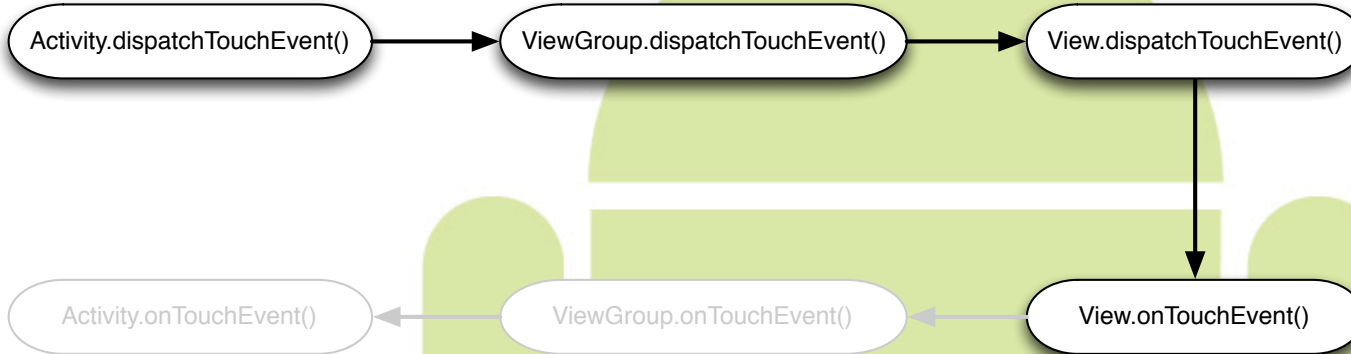


MOVE/UP:

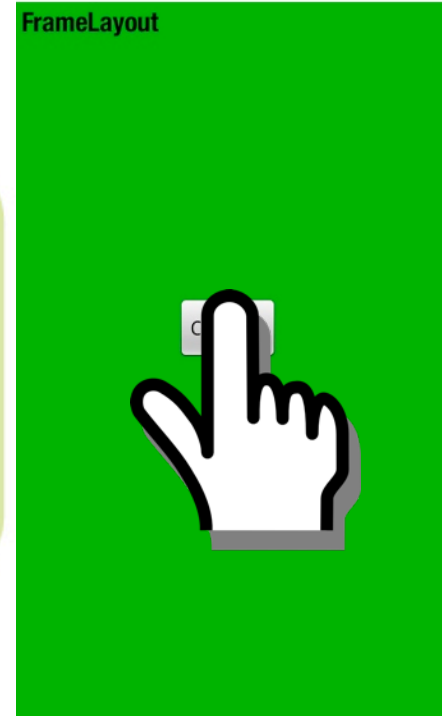


Interested View Example

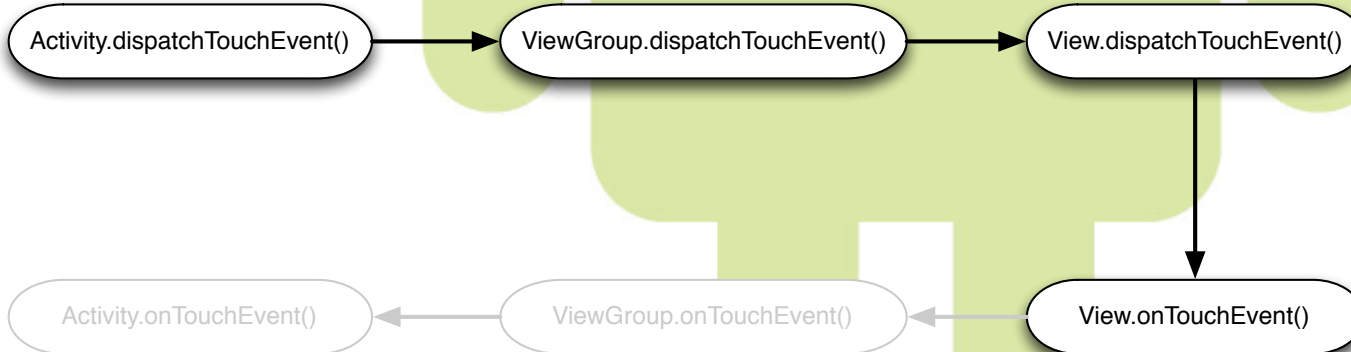
DOWN:



Activity
FrameLayout

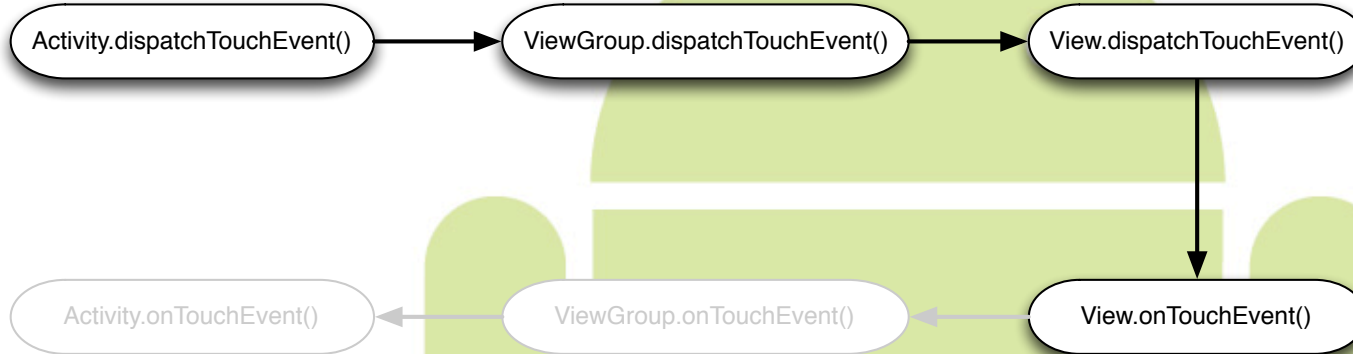


MOVE/UP:

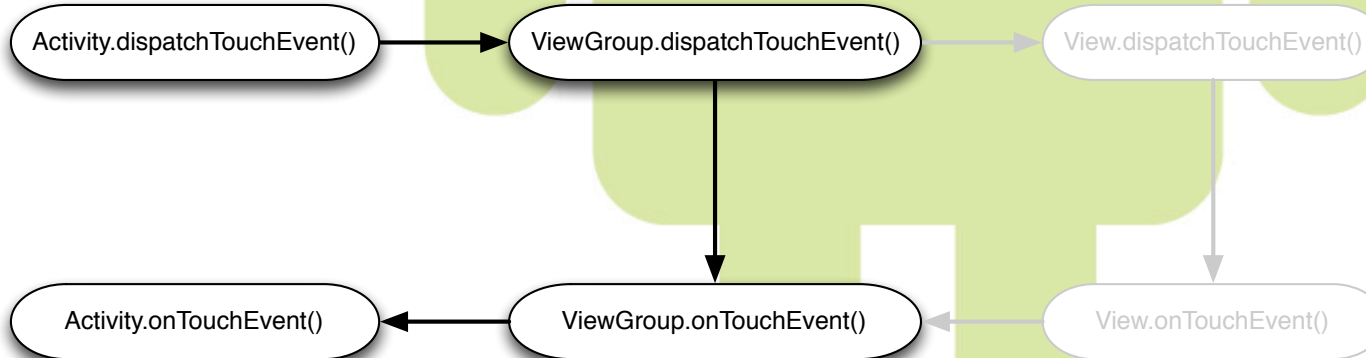


Intercept Example

DOWN:

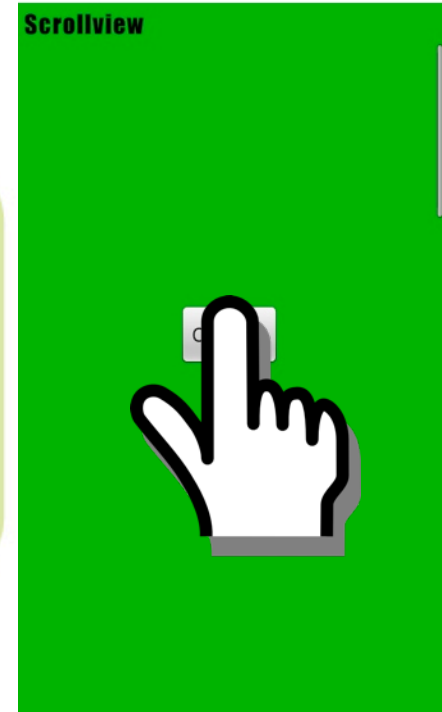


MOVE/UP:



CANCEL!

Activity
ScrollView



Custom Touch Handling



- Handling touch events
 - Subclass to override `onTouchEvent()`
 - Provide an `OnTouchListener`
- Consuming events
 - Return true with `ACTION_DOWN` to show interest
 - Even if you aren't interested in `ACTION_DOWN`, return true
 - For other events, returning true simply stops further processing
- Useful constants available in `ViewConfiguration`
 - `getScaledTouchSlop()`
 - Distance move events might vary before they should be considered a drag
 - `getScaledMinimumFlingVelocity()`
 - Speed at which the system considers a drag to be a fling
 - `getLongPressTimeout()`
 - Time the system waits to consider an event a long-press
 - Display values scaled for each device's density

Custom Touch Handling



- Forwarding touch events
 - Call target's `dispatchTouchEvent()`
 - Avoid calling target's `onTouchEvent()` directly
- Stealing touch events (`ViewGroup`)
 - Subclass to override `onInterceptTouchEvent()`
 - Return true when you want to take over
 - All subsequent events for the current gesture will come to your `onTouchEvent()` directly
 - `onInterceptTouchEvent()` will no longer be called for each event (one-shot redirect)
 - Any current target will receive `ACTION_CANCEL`

Custom Touch Handling Warnings



- Call through to super whenever possible
 - View.onTouchEvent() does a LOT of state management (pressed, checked, etc.) that you will lose if you capture every touch
- Protect ACTION_MOVE with slop checks
 - Fingers are fat and twitchy
- Always Handle ACTION_CANCEL
 - Container views with action (like scrolling) will steal events and you will likely need to reset state
 - Remember after CANCEL, you will get nothing else
- Don't intercept events until you're ready to take them all.
 - Intercept cannot be reversed until the next gesture.

Multi-Touch Handling



- `MotionEvent.getPointerCount()`
 - How many pointers are currently on the screen?
- Use the `ACTION_POINTER_DOWN` and `ACTION_POINTER_UP` events to detect secondary pointers
 - `MotionEvent.getActionMasked()`
 - `MotionEvent.getActionIndex()`
- Use `MotionEvent` methods that take a pointer index parameter to get data for a specific pointer
 - Methods with no parameter always return data for the FIRST pointer

Batching



- For efficiency, ACTION_MOVE events can be batched together in a single MotionEvent
- Latest (current) event is always returned by standard methods
 - getX(), getY(), getEventTime()
- Event occurring between this ACTION_MOVE and the last are found with historical methods
 - getHistoricalX(), getHistoricalY(), getHistoricalEventTime()
 - getHistoricalSize() returns number of batched events
- Can reconstruct all events as they occurred in time for maximum precision

System Touch Handlers



- Don't jump right to custom touch handling if you don't have to...
- OnClickListener
- OnLongClickListener
- onTouchListener
 - Monitor individual MotionEvent without a subclass
 - Can consume touches from a listener
 - Can pre-empt view's handling
- OnScrollListener / View.onScrollChanged()
 - View with existing scroll functionality has scrolled

System Touch Handlers



- For more complex touch interaction
- GestureDetector
 - `onDown()`, `onSingleTapUp()`, `onDoubleTap()`
 - `onLongPress()`
 - `onScroll()` (user dragging finger)
 - `onFling()` (user released drag with velocity)
- ScaleGestureDetector
 - `onScaleBegin()`, `onScale()`, `onScaleEnd()`
- Handled via `OnTouchListener` or `onTouchEvent()`
- Disadvantages
 - Consume UP events and exposes no interface for CANCEL events
 - May require added touch handling if these cases need special handling (e.g. resetting a View's appearance)

Touch Delegate



- Specialized object to assist in forwarding touches from a parent view to its child
- Allows for the touch area of a specific view to be different than its actual bounds
- Called in `onTouchEvent()` of attached View
 - Events have to make it that far without being consumed by a child or listener
- TouchDelegate is designed to be set on the PARENT and passed the CHILD view that touches should be forwarded to, i.e.

```
ViewGroup parent;  
View child;  
Rect touchArea;  
parent.setTouchDelegate( new TouchDelegate(touchArea, child) );
```



Once Again...

- Dave Smith
- Twitter: @devunwired
- Blog: <http://wiresareobsolete.com>
- Samples:
 - <https://github.com/devunwired/custom-touch-examples>