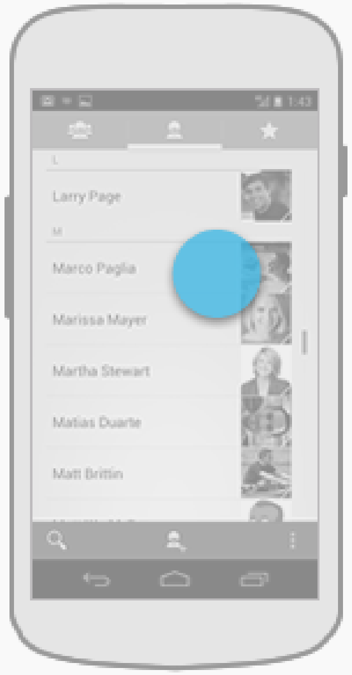
Mobile Application Development

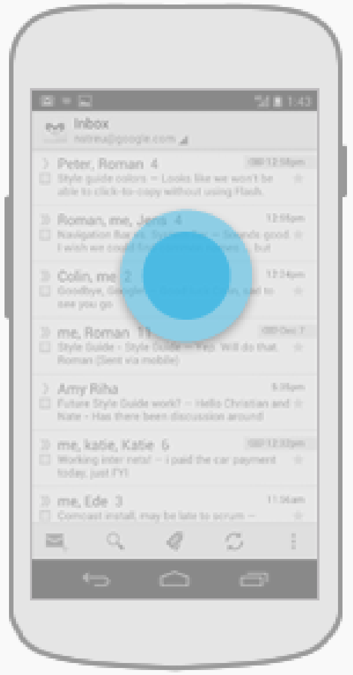
Gestures

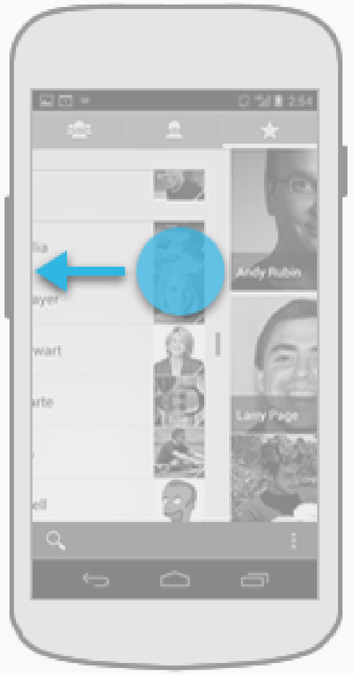
Detect common gestures

- A "touch gesture" occurs when a user places one or more fingers on the touch screen, and your application interprets that pattern of touches as a particular gesture. There are correspondingly two phases to gesture detection:
  - Gather data about touch events.
  - Interpret the data to see if it meets the criteria for any of the gestures your app supports.

Common Gestures







Touch

Triggers the default functionality for a given item.

Action
Press, lift

Long press

Enters data selection mode. Allows you to select one or more items in a view and act upon the data using a contextual action bar. Avoid using long press for showing contextual menus.

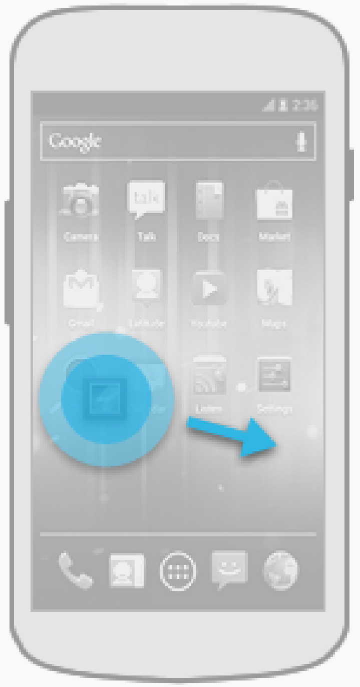
Action
Press, wait, lift

Swipe Or Scroll

Scrolls overflowing content, or navigates between views in the same hierarchy.

Action
Press, move, lift

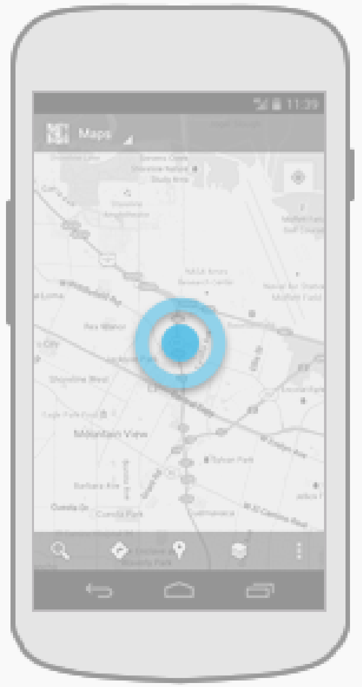
Common Gestures



Drag

Rearranges data within a view, or moves data into a container (e.g. folders on Home Screen).

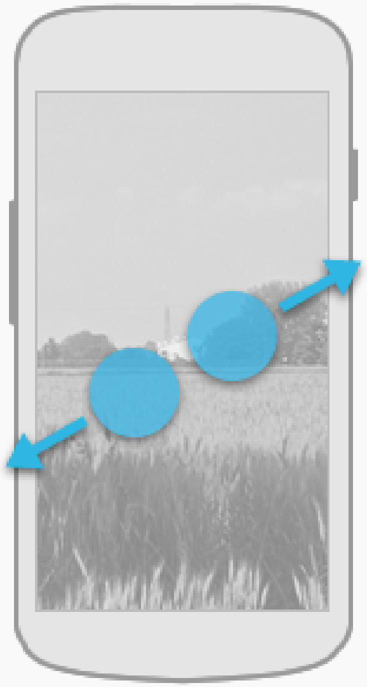
Action
Long press, move, lift



Double touch

Zooms into content. Also used as a secondary gesture for text selection.

Action
Two touches in quick succession

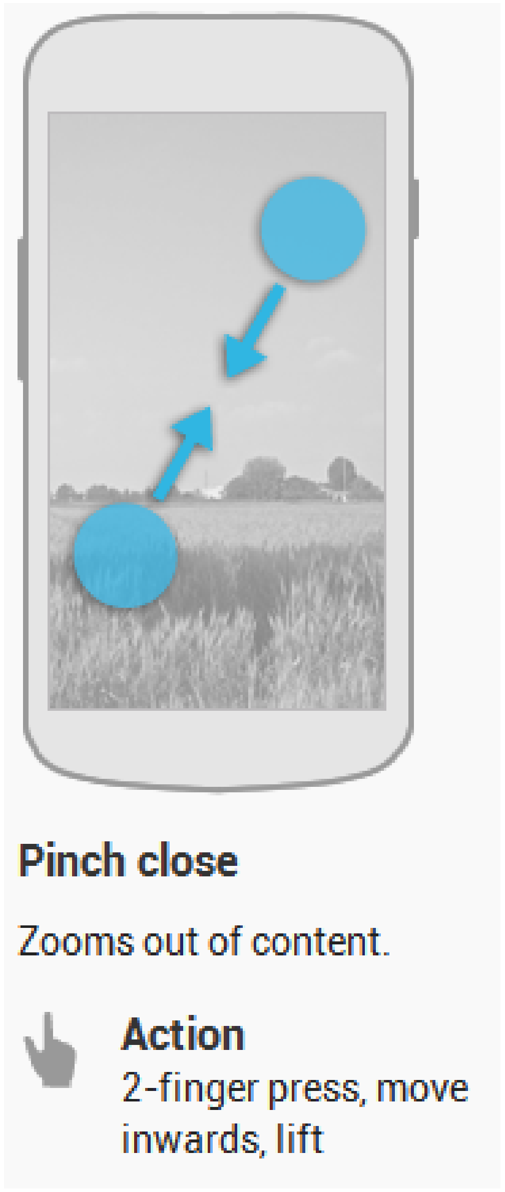


Pinch open

Zooms into content.

Action
2-finger press, move outwards, lift

Common Gestures



Pinch close

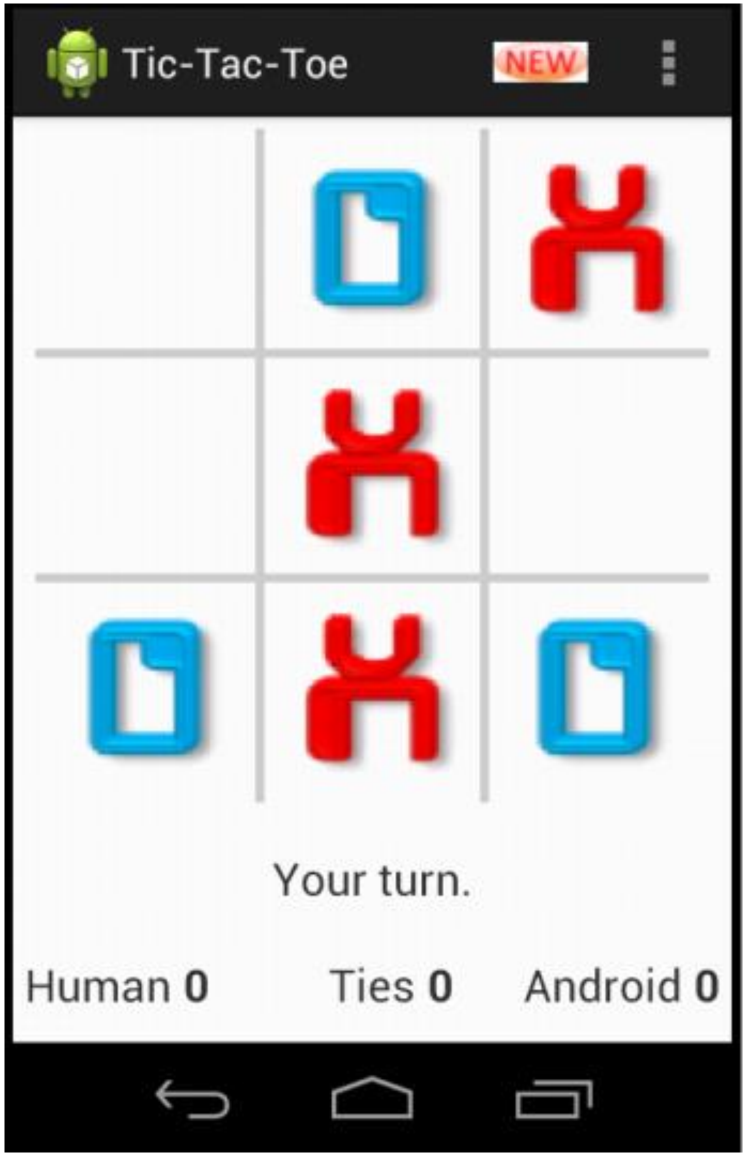
Zooms out of content.

Action
2-finger press, move inwards, lift

- Fling or flick gesture: similar to swipe or drag
- scroll/swipe/drag
  - user presses then moves finger in steady motion before lifting finger
- fling or flick
  - user presses then moves finger in an accelerating motion before lifting

Dealing With Gestures

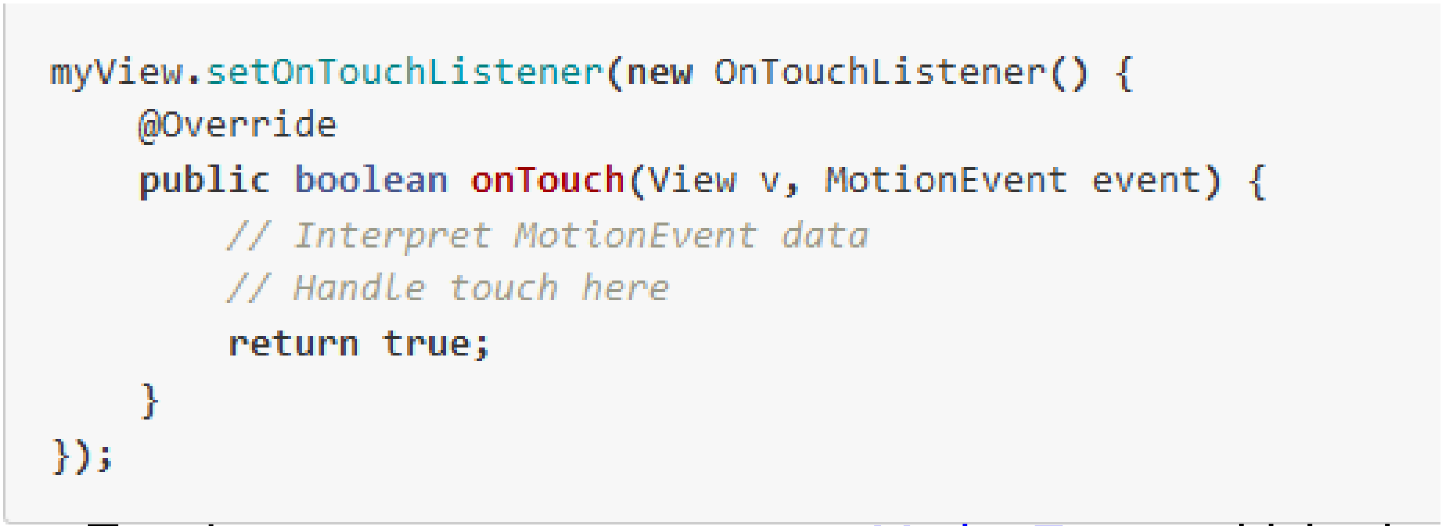
- To handle simple touch events create View.OnTouchListener for view
- Example from tic-tac-toe tutorial, screen press leads to player moving if it is their turn and they touch an open square



onTouchEvent

- passed a MotionEvent object with a **large** amount of data
- The gesture starts when the user first touches the screen, continues as the system tracks the position of the user's finger(s), and ends by capturing the final event of the user's fingers leaving the screen. Throughout this interaction, the MotionEvent delivered to onTouchEvent() provides the details of every interaction.

onTouchEvent

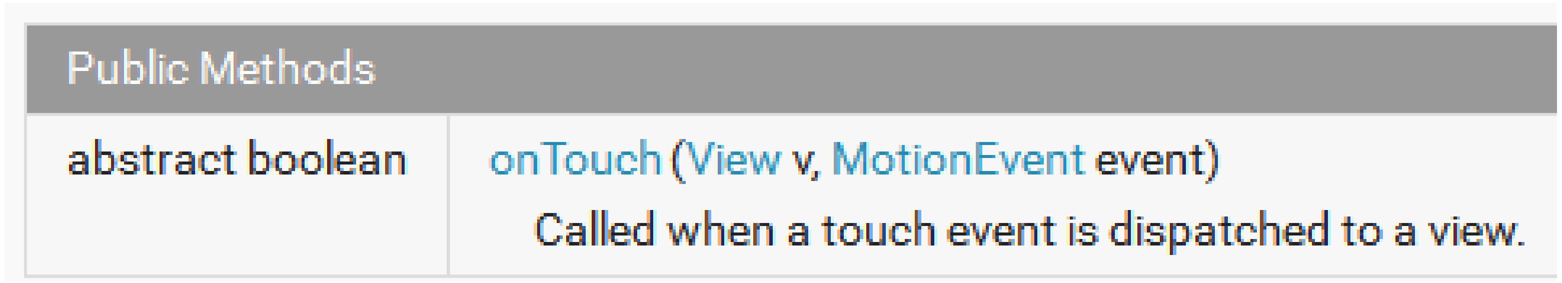


```
myView.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Interpret MotionEvent data
        // Handle touch here
        return true;
    }
});
```

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:

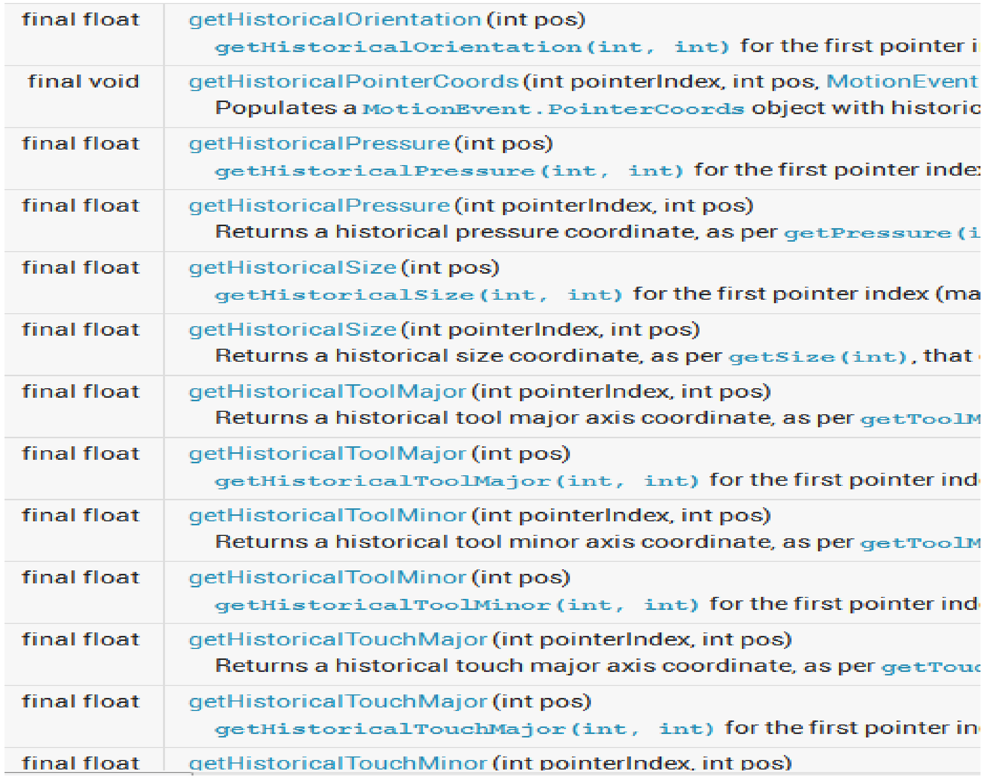
- **getAction()** - Returns an integer constant such as MotionEvent.ACTION_DOWN, MotionEvent.ACTION_MOVE, and MotionEvent.ACTION_UP
- **getX()** - Returns the x coordinate of the touch event
- **getY()** - Returns the y coordinate of the touch event

MotionEvent



Public Methods	
abstract boolean	onTouch(View v, MotionEvent event) Called when a touch event is dispatched to a view.

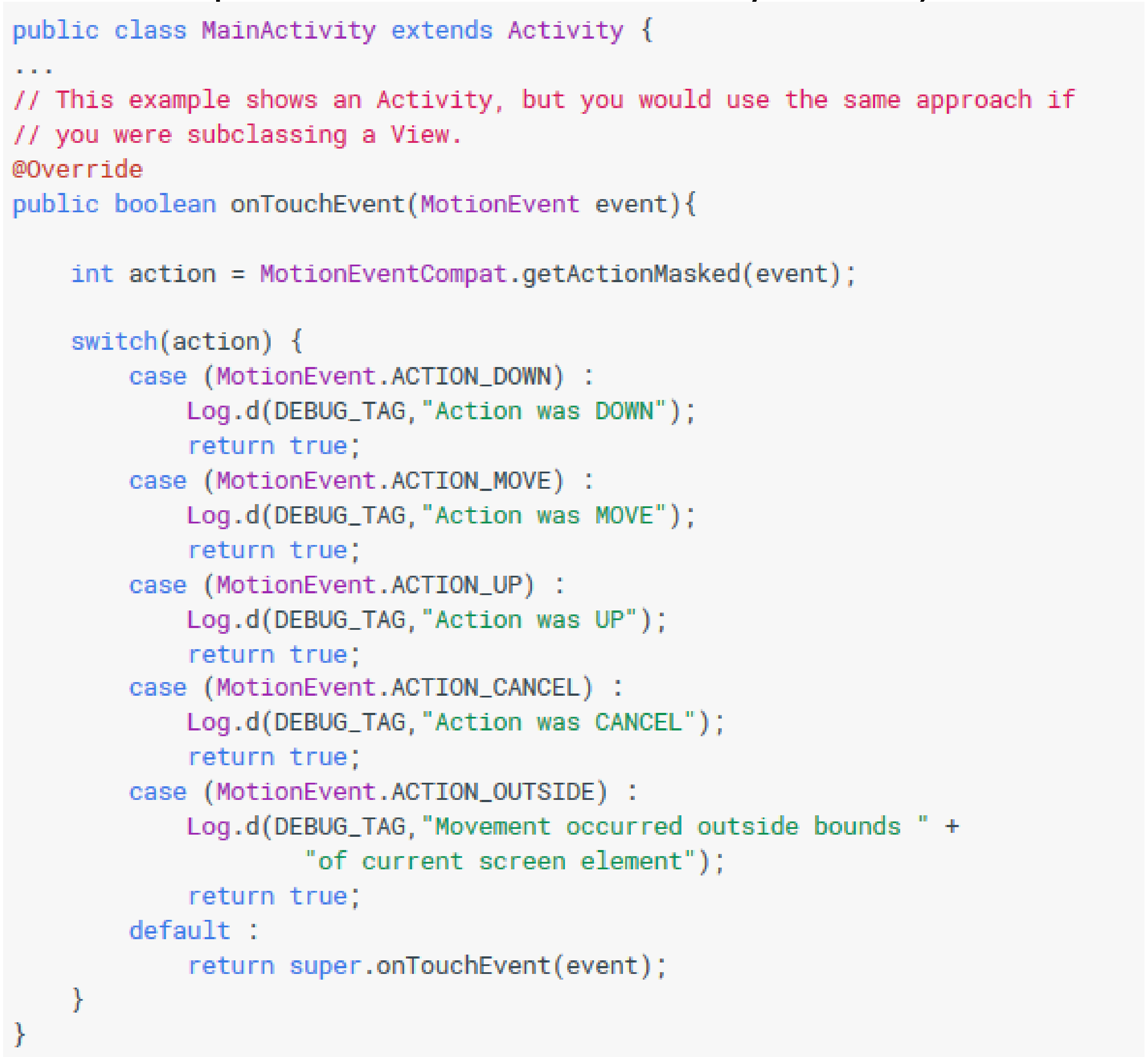
- Example of the astonishing amount of data packed into the **motionEvent** object



final float	getHistoricalOrientation(int pos) getHistoricalOrientation(int, int) for the first pointer i
final void	getHistoricalPointerCoords(int pointerIndex, int pos, MotionEvent Populates a MotionEvent.PointerCoords object with historic
final float	getHistoricalPressure(int pos) getHistoricalPressure(int, int) for the first pointer inde:
final float	getHistoricalPressure(int pointerIndex, int pos) Returns a historical pressure coordinate, as per getPressure(i
final float	getHistoricalSize(int pos) getHistoricalSize(int, int) for the first pointer index (ma
final float	getHistoricalSize(int pointerIndex, int pos) Returns a historical size coordinate, as per getSize(int), that
final float	getHistoricalToolMajor(int pointerIndex, int pos) Returns a historical tool major axis coordinate, as per getToolM
final float	getHistoricalToolMajor(int pos) getHistoricalToolMajor(int, int) for the first pointer ind
final float	getHistoricalToolMinor(int pointerIndex, int pos) Returns a historical tool minor axis coordinate, as per getToolM
final float	getHistoricalToolMinor(int pos) getHistoricalToolMinor(int, int) for the first pointer ind
final float	getHistoricalTouchMajor(int pointerIndex, int pos) Returns a historical touch major axis coordinate, as per getTouc
final float	getHistoricalTouchMajor(int pos) getHistoricalTouchMajor(int, int) for the first pointer in
final float	getHistoricalTouchMinor(int pointerIndex, int pos)

onTouchEvent

To intercept touch events in an Activity or View, override the onTouchEvent()



```
public class MainActivity extends Activity {
...
// This example shows an Activity, but you would use the same approach if
// you were subclassing a View.
@Override
public boolean onTouchEvent(MotionEvent event){

    int action = MotionEventCompat.getActionMasked(event);

    switch(action) {
        case (MotionEvent.ACTION_DOWN) :
            Log.d(DEBUG_TAG,"Action was DOWN");
            return true;
        case (MotionEvent.ACTION_MOVE) :
            Log.d(DEBUG_TAG,"Action was MOVE");
            return true;
        case (MotionEvent.ACTION_UP) :
            Log.d(DEBUG_TAG,"Action was UP");
            return true;
        case (MotionEvent.ACTION_CANCEL) :
            Log.d(DEBUG_TAG,"Action was CANCEL");
            return true;
        case (MotionEvent.ACTION_OUTSIDE) :
            Log.d(DEBUG_TAG,"Movement occurred outside bounds " +
                    "of current screen element");
            return true;
        default :
            return super.onTouchEvent(event);
    }
}
```

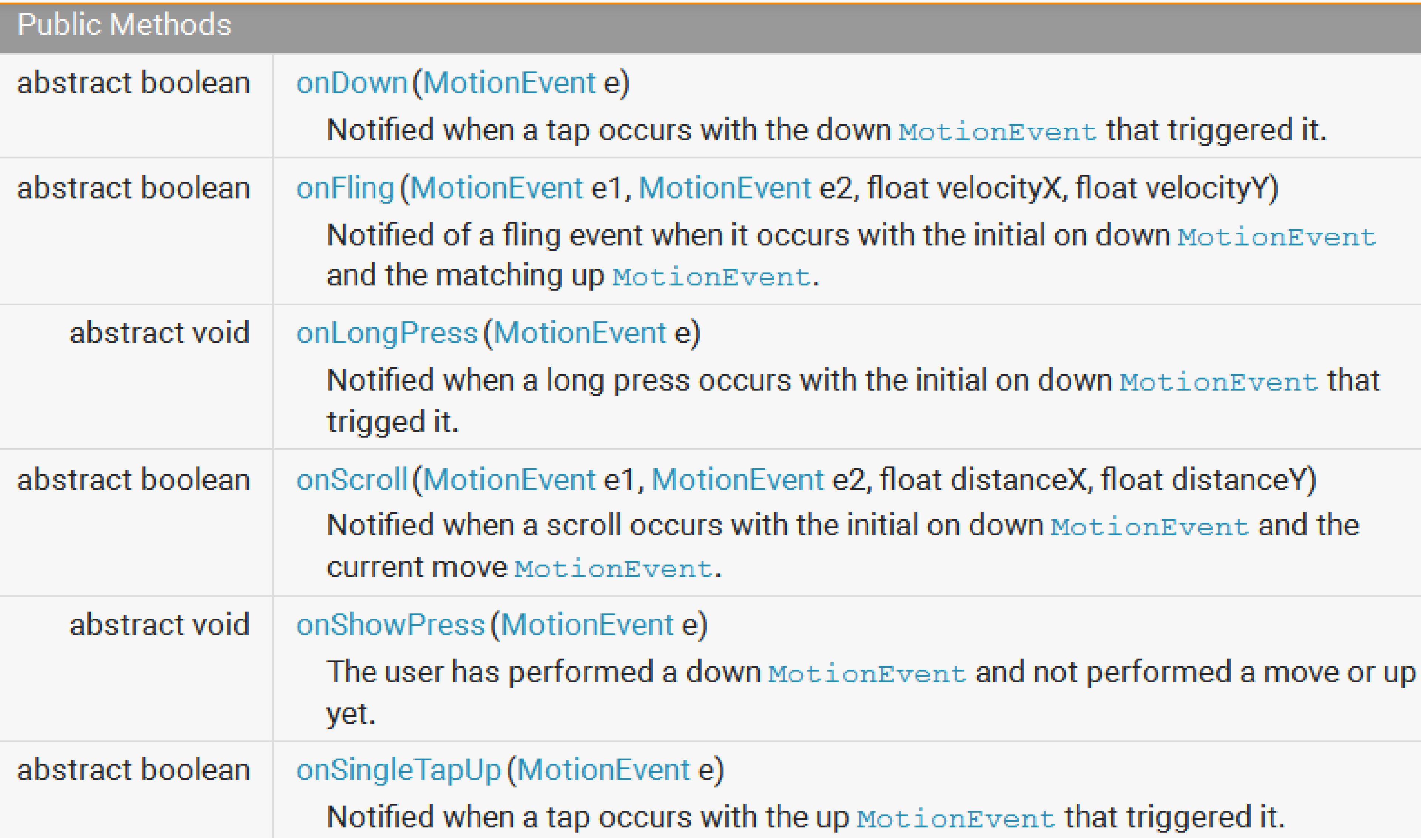
Other View Listeners

- View also has ability to listen for long clicks and drags
- In addition to View.OnTouchListener
- View.OnLongClickListener
- View.OnDragListener

Handling Common Gestures

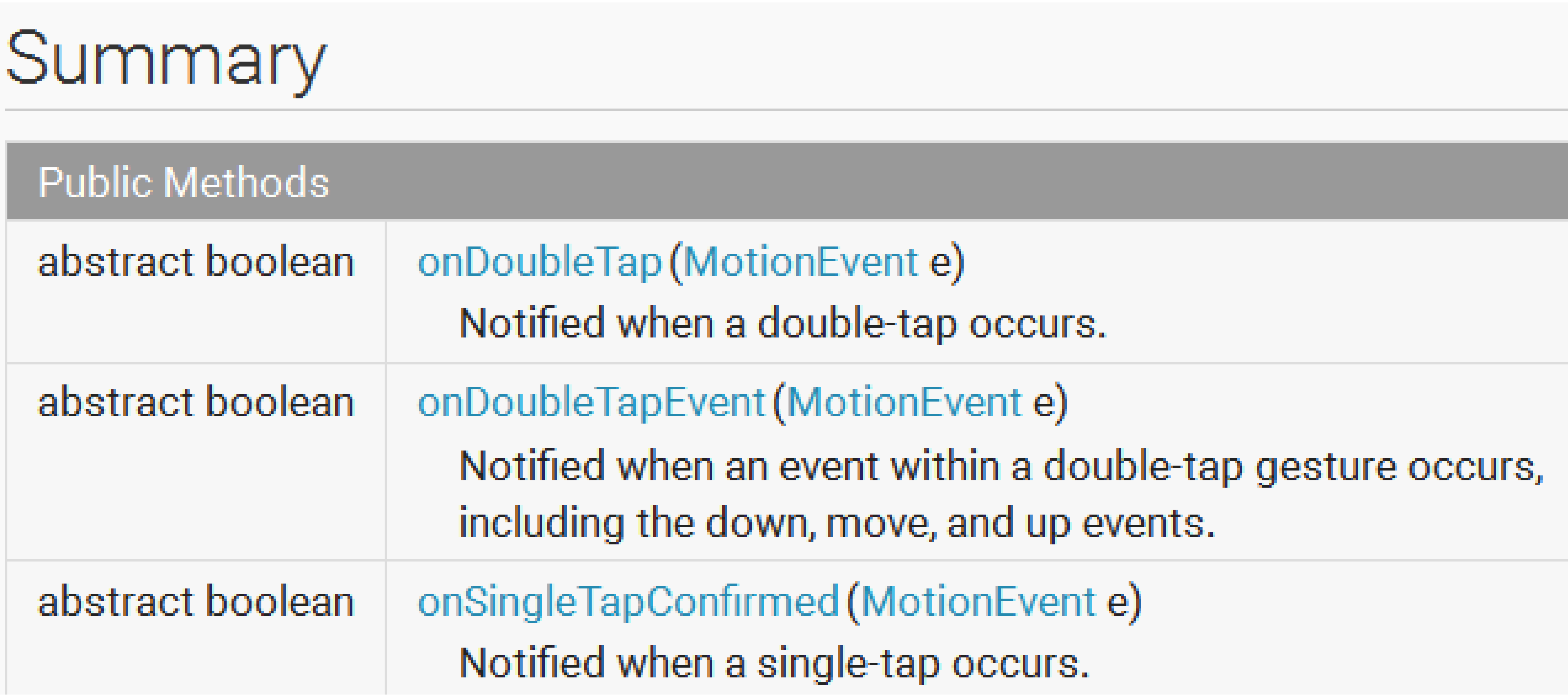
- Instead of trying to decode gestures from the MotionEvent passed to the on touch method
- Use the GestureDetector class
- Add a GestureDetector object to View
- override View.onTouchEvent method to pass MotionEvent on to the GestureDetector.onTouchEvent method
- create a GestureDetector.OnGestureListener or a GestureDetector.DoubleTapListener and register it with the GesturerDetector

GestureDetector.OnGestureListener



Public Methods	
abstract boolean	onDown(MotionEvent e) Notified when a tap occurs with the down `MotionEvent` that triggered it.
abstract boolean	onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) Notified of a fling event when it occurs with the initial on down `MotionEvent` and the matching up `MotionEvent`.
abstract void	onLongPress(MotionEvent e) Notified when a long press occurs with the initial on down `MotionEvent` that trigged it.
abstract boolean	onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) Notified when a scroll occurs with the initial on down `MotionEvent` and the current move `MotionEvent`.
abstract void	onShowPress(MotionEvent e) The user has performed a down `MotionEvent` and not performed a move or up yet.
abstract boolean	onSingleTapUp(MotionEvent e) Notified when a tap occurs with the up `MotionEvent` that triggered it.

GestureDetector.DoubleTapListener

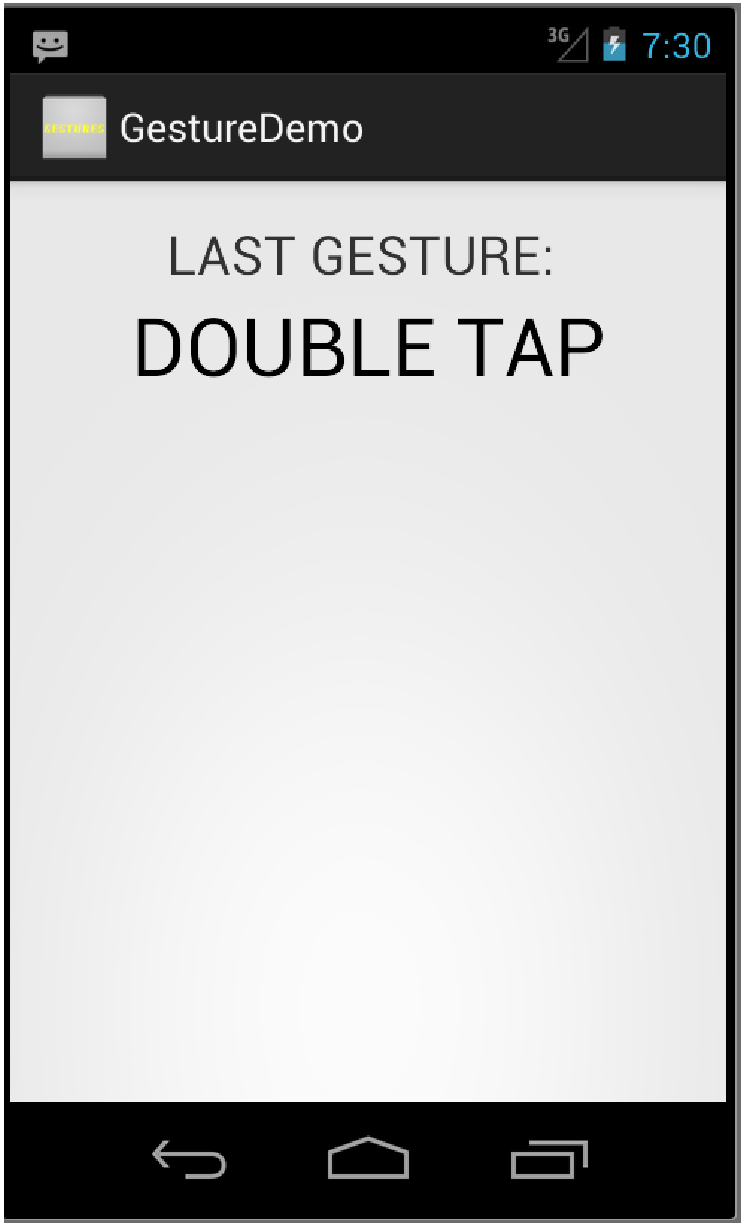


Summary

Public Methods	
abstract boolean	onDoubleTap(MotionEvent e) Notified when a double-tap occurs.
abstract boolean	onDoubleTapEvent(MotionEvent e) Notified when an event within a double-tap gesture occurs, including the down, move, and up events.
abstract boolean	onSingleTapConfirmed(MotionEvent e) Notified when a single-tap occurs.

Simple Gesture Demo

- App that listens for simple gestures
- update lower TextView in call back methods

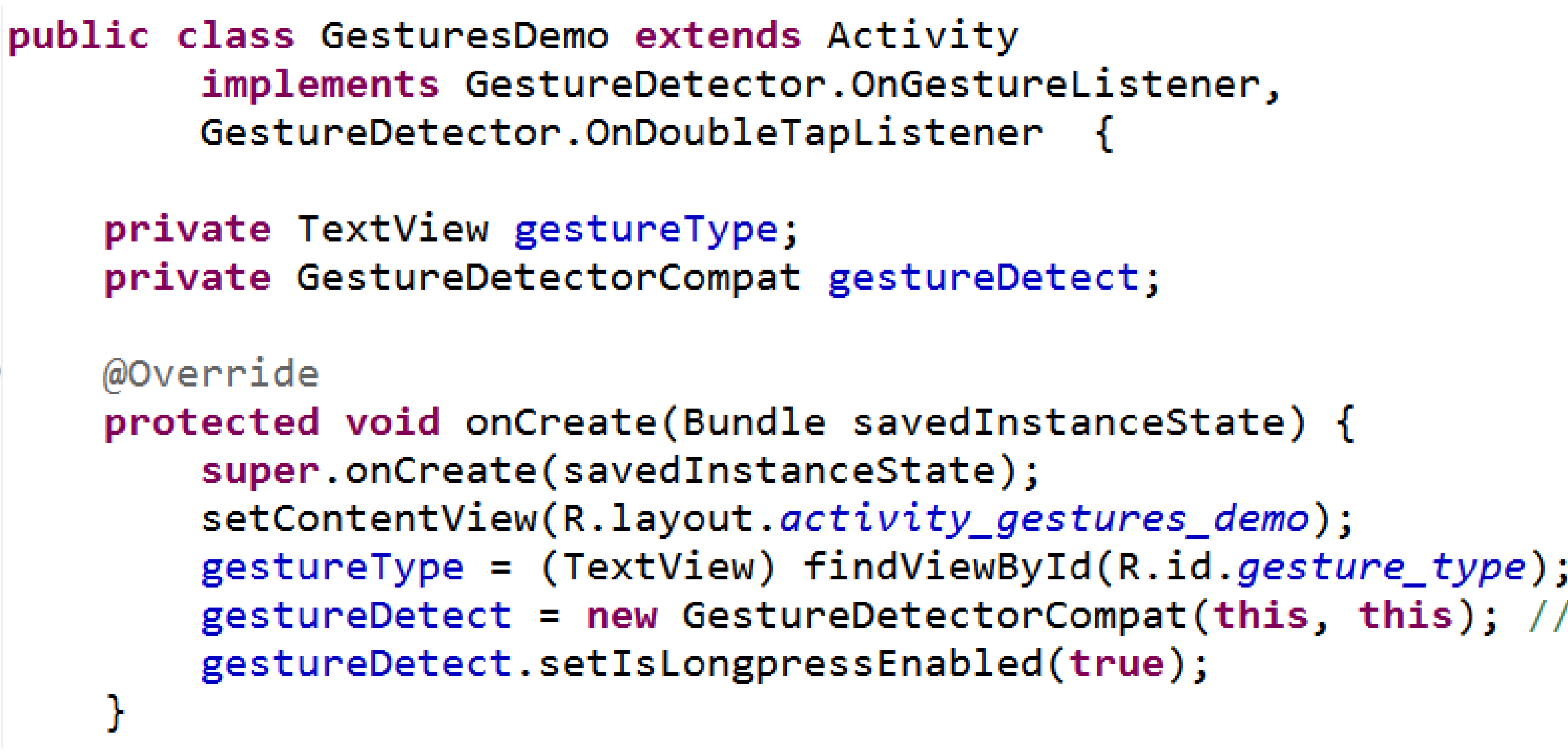

7:30

GestureDemo

LAST GESTURE:

DOUBLE TAP

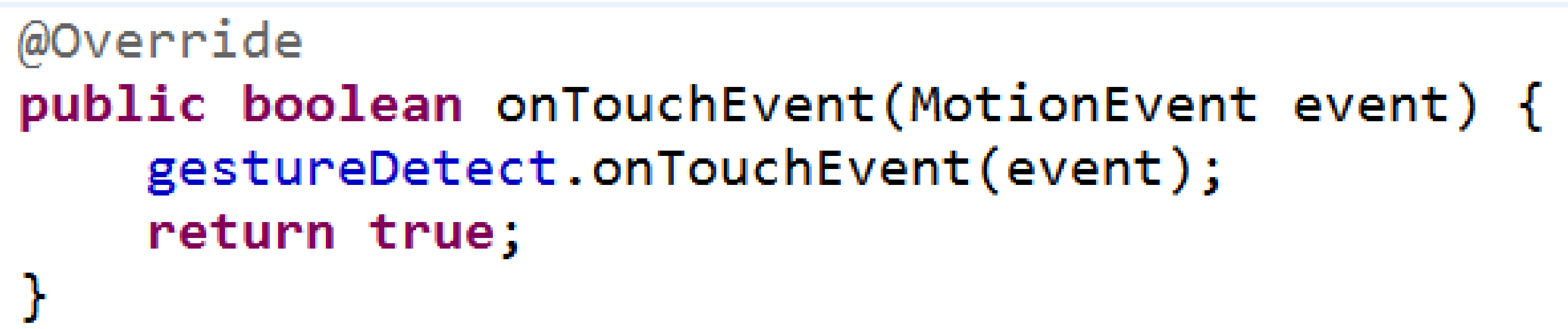
Gesture Demo



```
public class GesturesDemo extends Activity
        implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener  {

    private TextView gestureType;
    private GestureDetectorCompat gestureDetect;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gestures_demo);
        gestureType = (TextView) findViewById(R.id.gesture_type);
        gestureDetect = new GestureDetectorCompat(this, this); //
        gestureDetect.setIsLongpressEnabled(true);
    }
```

Gesture Demo

- Simply pass event on to the GestureDetectorCompat object
  - it will call back methods



```
@Override
public boolean onTouchEvent(MotionEvent event) {
    gestureDetect.onTouchEvent(event);
    return true;
}
```

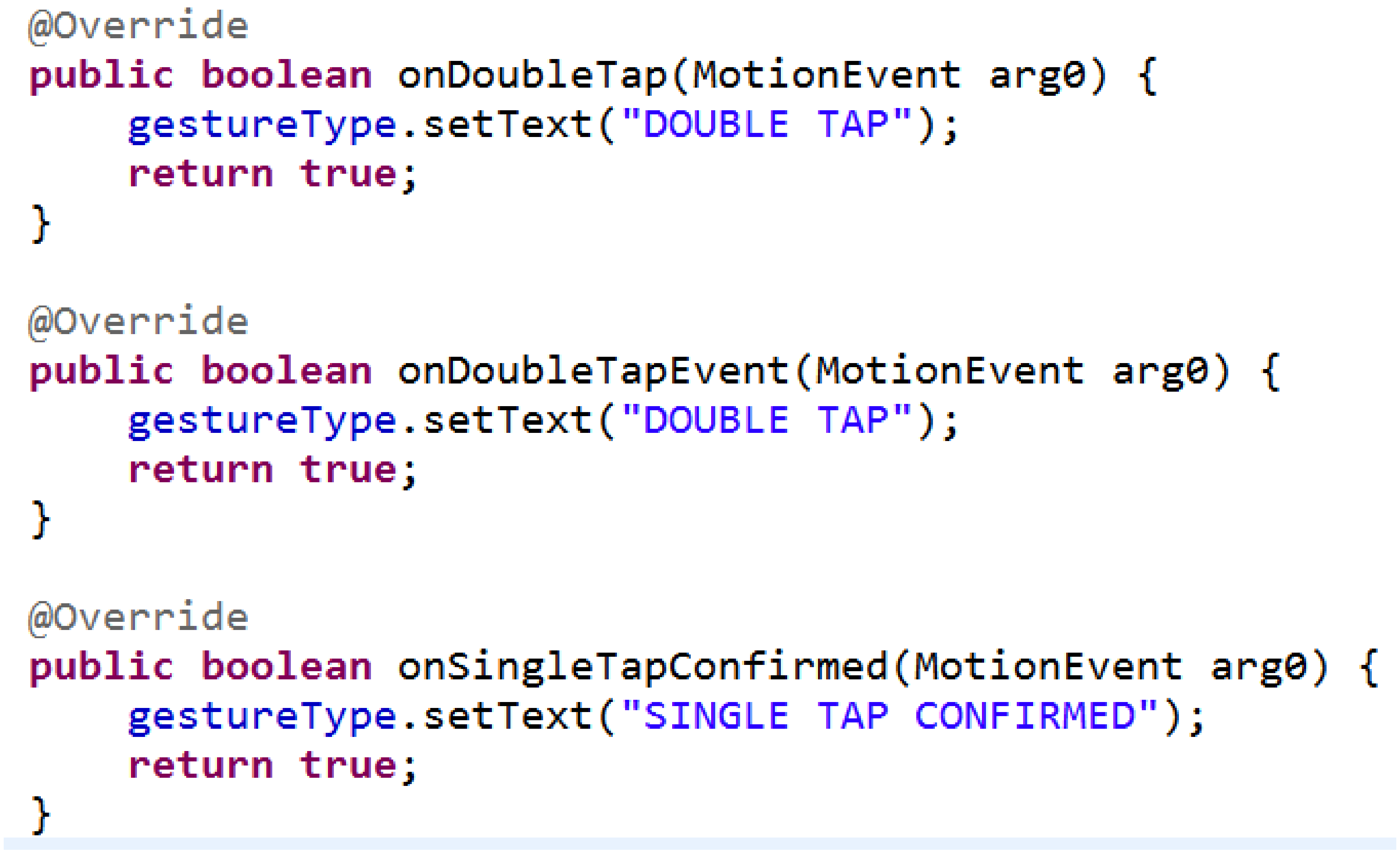
Callback Methods for OnGestureListener

```
@Override
public boolean onDown(MotionEvent e) {
    gestureType.setText("DOWN");
    return true;
}

@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
        float velocityY) {
    gestureType.setText("FLING");
    return true;
}

@Override
public void onLongPress(MotionEvent e) {
    gestureType.setText("LONG PRESS");

}
```

Callback Methods for OnGestureListener

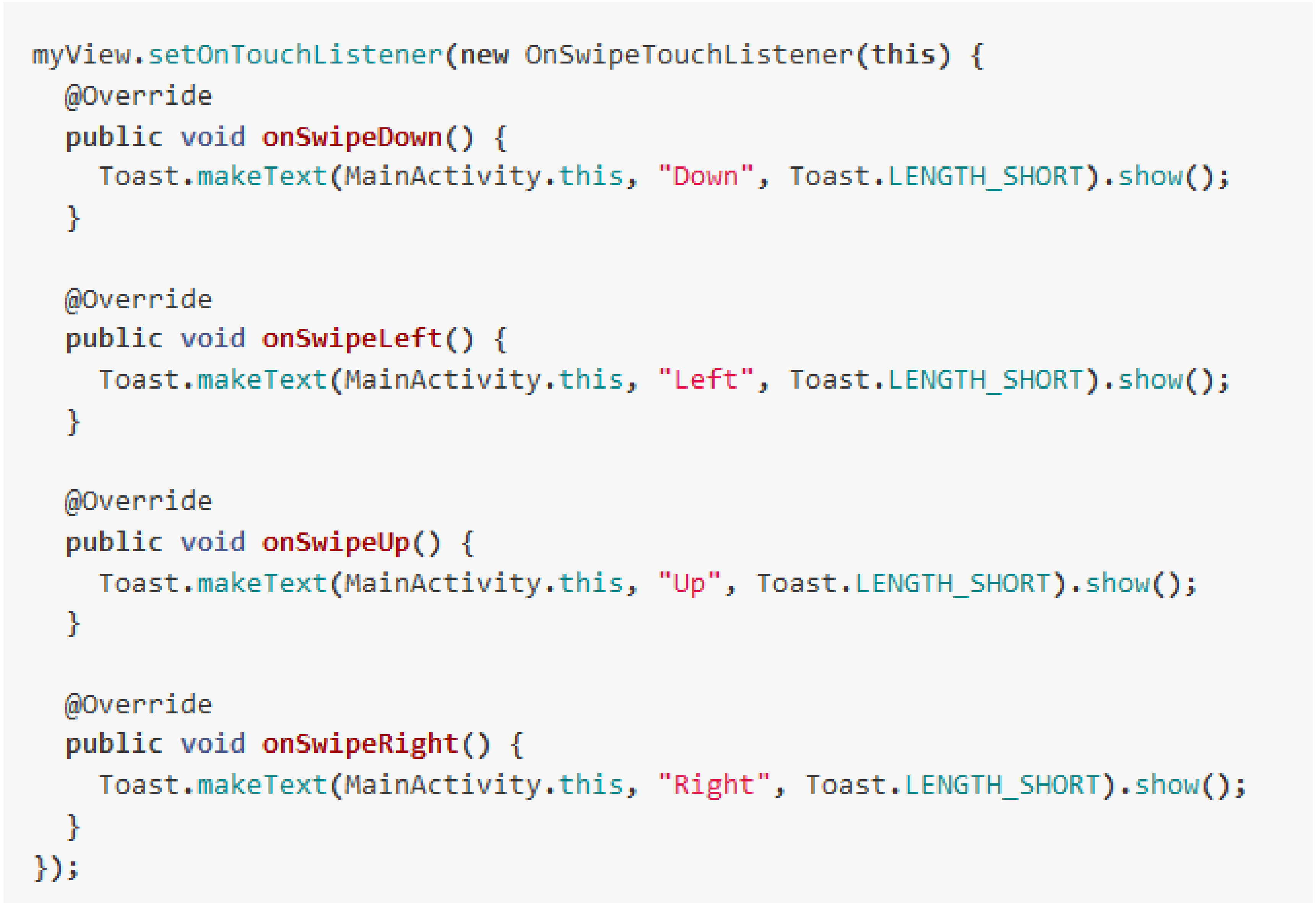
```
@Override
public boolean onScroll (MotionEvent e1, MotionEvent e2,
        float distanceX, float distanceY) {
        gestureType.setText("SCROLL");
        return true;
}

@Override
public void onShowPress(MotionEvent e) {
    gestureType.setText("SHOW PRESS");
}

@Override
public boolean onSingleTapUp(MotionEvent e) {
    gestureType.setText("SINGLE TAP UP");
    return true;
}
```

Callback Methods for DoubleTapListener



```
@Override
public boolean onDoubleTap(MotionEvent arg0) {
    gestureType.setText("DOUBLE TAP");
    return true;
}

@Override
public boolean onDoubleTapEvent(MotionEvent arg0) {
    gestureType.setText("DOUBLE TAP");
    return true;
}

@Override
public boolean onSingleTapConfirmed(MotionEvent arg0) {
    gestureType.setText("SINGLE TAP CONFIRMED");
    return true;
}
```

Swipe Gesture Detection

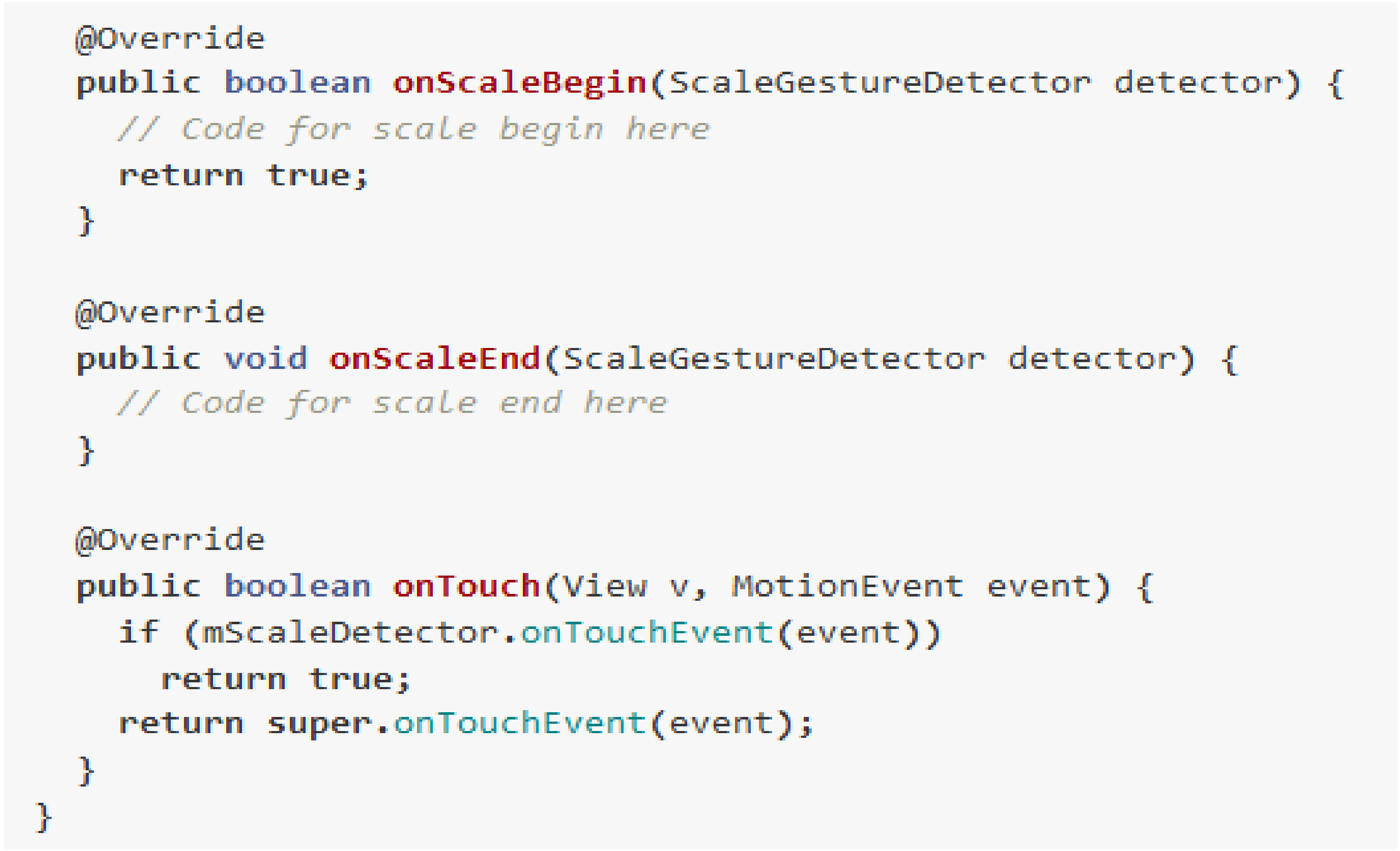


```java
myView.setOnTouchListener(new OnSwipeTouchListener(this) {
  @Override
  public void onSwipeDown() {
    Toast.makeText(MainActivity.this, "Down", Toast.LENGTH_SHORT).show();
  }

  @Override
  public void onSwipeLeft() {
    Toast.makeText(MainActivity.this, "Left", Toast.LENGTH_SHORT).show();
  }

  @Override
  public void onSwipeUp() {
    Toast.makeText(MainActivity.this, "Up", Toast.LENGTH_SHORT).show();
  }

  @Override
  public void onSwipeRight() {
    Toast.makeText(MainActivity.this, "Right", Toast.LENGTH_SHORT).show();
  }
});
```

Pinch to Zoom

```
public class ScaleableTextView extends TextView
        implements OnTouchListener, OnScaleGestureListener {

  ScaleGestureDetector mScaleDetector =
      new ScaleGestureDetector(getContext(), this);

  public ScaleableTextView(Context context, AttributeSet attrs) {
    super(context, attrs);
  }

  @Override
  public boolean onScale(ScaleGestureDetector detector) {
    // Code for scale here
    return true;
  }
```

Pinch to Zoom

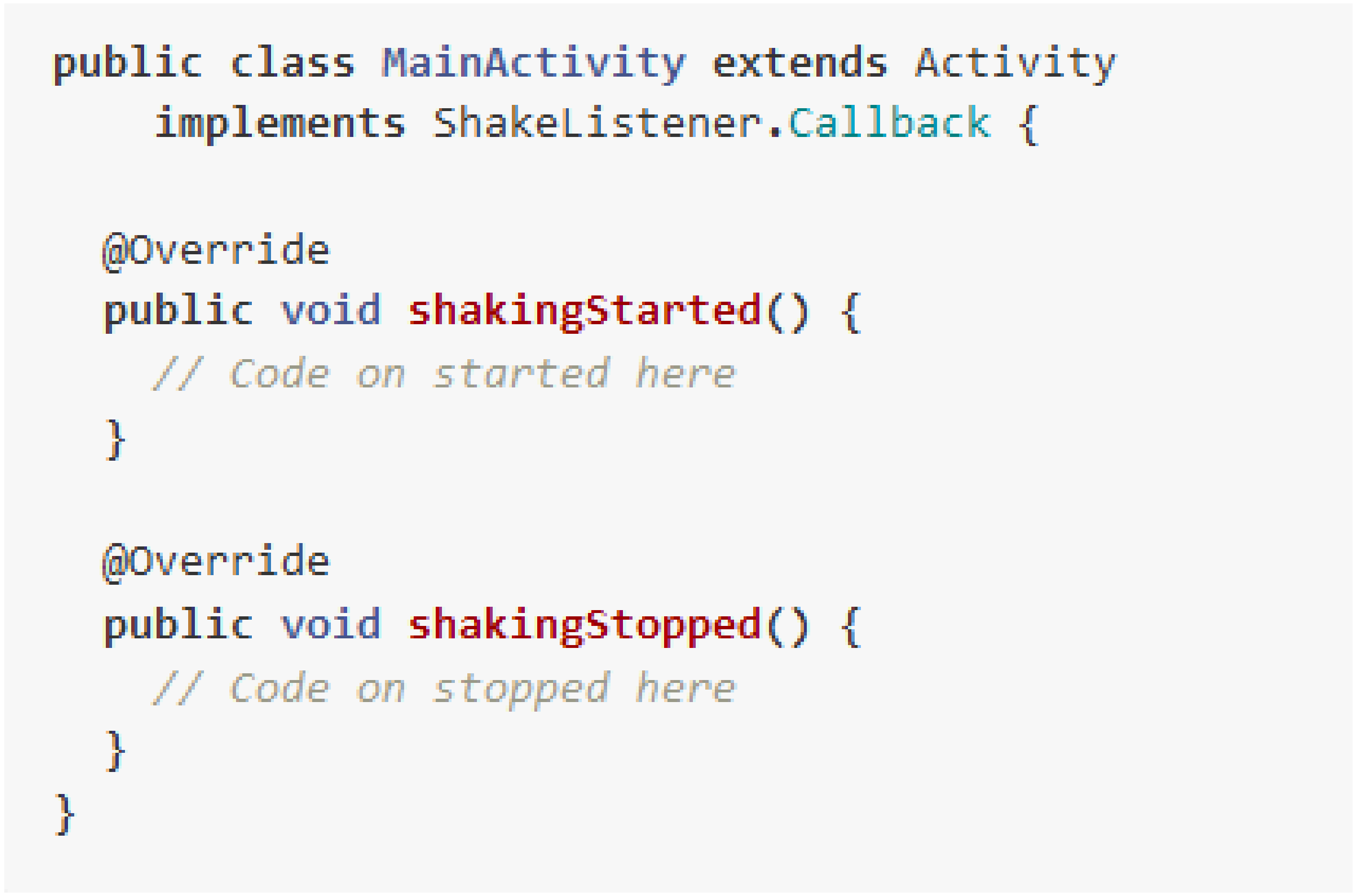


```
  @Override
  public boolean onScaleBegin(ScaleGestureDetector detector) {
    // Code for scale begin here
    return true;
  }

  @Override
  public void onScaleEnd(ScaleGestureDetector detector) {
    // Code for scale end here
  }

  @Override
  public boolean onTouch(View v, MotionEvent event) {
    if (mScaleDetector.onTouchEvent(event))
      return true;
    return super.onTouchEvent(event);
  }
}
```

https://www.sitepoint.com/android-gestures-and-touch-mechanics/

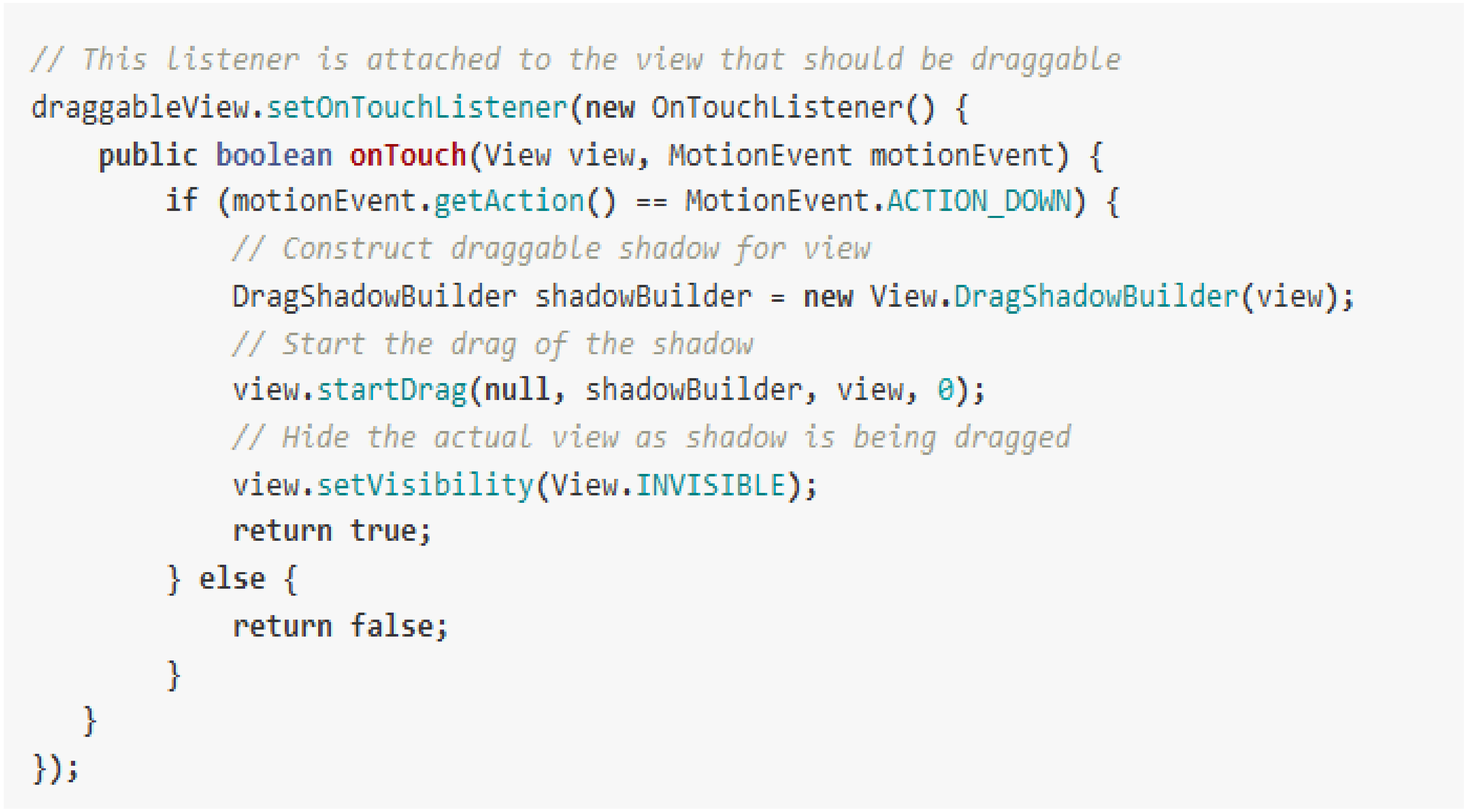
See **Pinch Gesture** example part

Shake Detection



```
public class MainActivity extends Activity
    implements ShakeListener.Callback {

  @Override
  public void shakingStarted() {
    // Code on started here
  }

  @Override
  public void shakingStopped() {
    // Code on stopped here
  }
}
```

Dragging and Dropping



```java
// This listener is attached to the view that should be draggable
draggableView.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
            // Construct draggable shadow for view
            DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(view);
            // Start the drag of the shadow
            view.startDrag(null, shadowBuilder, view, 0);
            // Hide the actual view as shadow is being dragged
            view.setVisibility(View.INVISIBLE);
            return true;
        } else {
            return false;
        }
    }
});
```

Reference

- https://developer.android.com/training/gestures/detector

Read the full page carefully

- https://guides.codepath.com/android/gestures-and-touch-events#overview