LA TROBE
UNIVERSITY
All kinds of clever

CSE2MAD

Mobile Application Development
Lecture 7 Part 2

What are Gestures?

- anytime a user touches the screen and does a pre-defined motion that the system understands

- touch events: position, pressure, size, addition of another finger, etc.

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

https://www.youtube.com/watch?v=SrEU72RjIhg

Common Gestures

- Touch gesture : when a user places one or more fingers on the touch screen, and the app interprets that pattern of touches as a particular gesture
- Two phases to gesture detection:
  1. Gathering data about touch events.

  2. Interpreting the data to see if it meets the criteria for any of the gestures your app supports.

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

Common Gestures
- Capturing touch events

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

This snippet uses getActionMasked() to extract the action the user performed from the event parameter.

This is the kind of processing you would have to do for a custom gesture. However, if your app uses common gestures such as double tap, long press, fling, and so on, you can take advantage of the GestureDetector class.

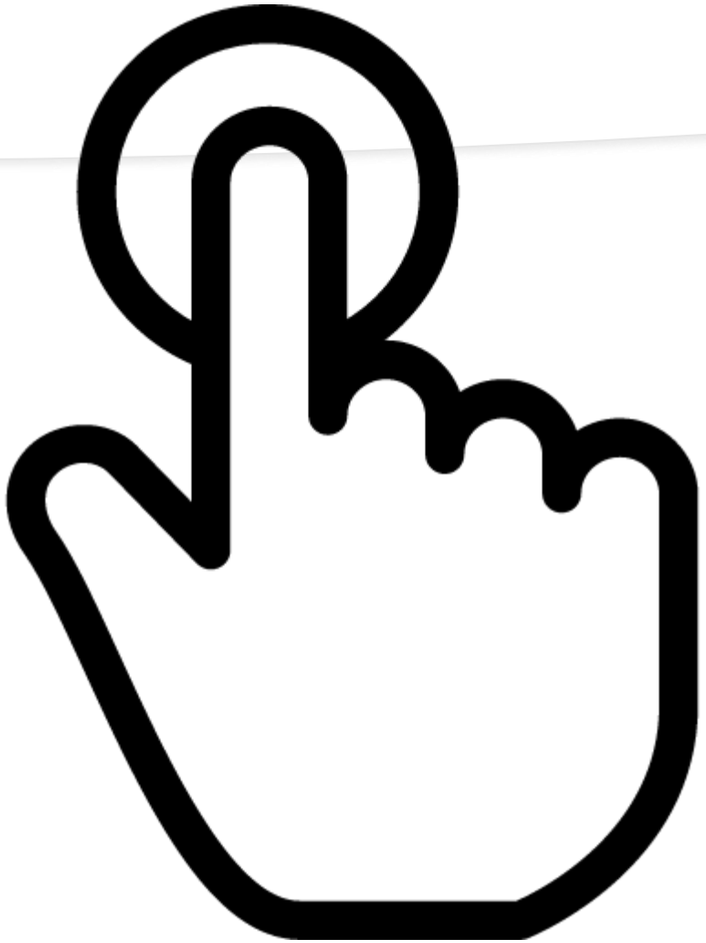
```
@Override
public boolean onTouchEvent(MotionEvent event){

    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);
    }
```

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

Common Gestures
- Capturing touch events

Android provides the GestureDetector class for detecting common gestures. Some of the gestures it supports include onDown(), onLongPress(), onFling(), and so on. You can use GestureDetector in conjunction with the onTouchEvent.

https://developer.android.com/reference/android/view/GestureDetector



https://icon-library.net/icon/select-icon-png-7.html

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

Common Gestures - Detecting All Supported Gestures

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

    private static final String DEBUG_TAG = "Gestures";
    private GestureDetectorCompat mDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDetector = new GestureDetectorCompat(this,this);
        // Set the gesture detector as the double tap listener.
        mDetector.setOnDoubleTapListener(this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        if (this.mDetector.onTouchEvent(event)) {
            return true;
        }
        return super.onTouchEvent(event);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        Log.d(DEBUG_TAG,"onDown: " + event.toString());
        return true;
    }
.......
```

https://developer.android.com/training/gestures/detector#detect-all-supported-gestures

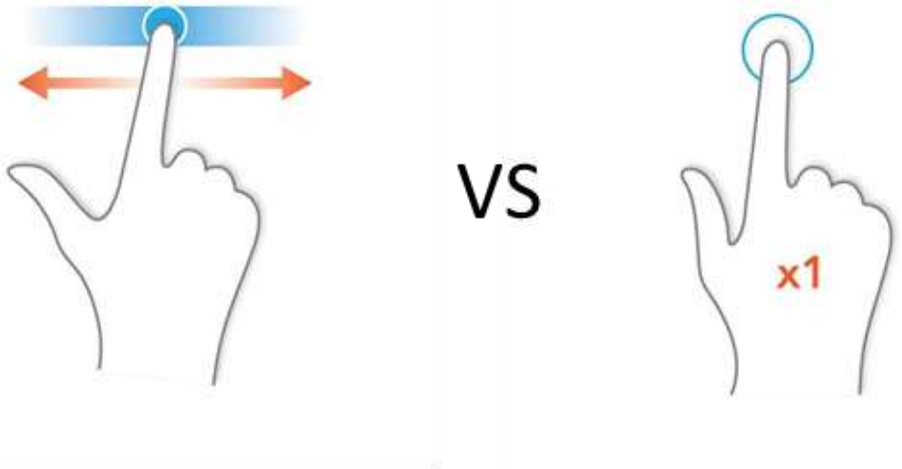
Tracking Movement

onTouchEvent(MotionEvent event)
- ACTION_MOVE

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        int action = event.getAction();

        if(action == MotionEvent.ACTION_MOVE){
            Log.d("MAD","ACTION_MOVE");
        }

        return super.onTouchEvent(event);
    }
}
```

https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE

Touch Slop



Question: Is it a swipe or a tap??

To distinguish between movement-based gestures (eg: swipe) and non- movement gestures (eg: single tap)

Touch slop is the distance in pixels a user's touch can wander before the gesture is considered a movement-based gesture

https://developer.android.com/training/gestures/movement

Different ways to track movement in a Gesture

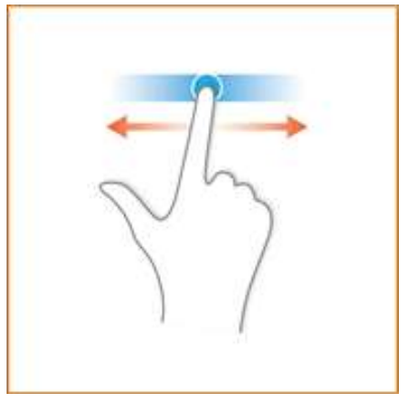
- The starting and ending position of a pointer (for example, move an on-screen object from point A to point B).
- The direction the pointer is traveling in, as determined by the x and y coordinates.
- History. You can find the size of a gesture's history by calling the MotionEvent method getHistorySize(). 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.
- The velocity of the pointer as it moves across the touch screen.

https://developer.android.com/training/gestures/movement

Track Velocity

The rate of speed of the touch event/s can give us clues about the gesture, or if a gesture as occurred

Use VelocityTracker & VelocityTrackerCompat



Horizontal swipe
X velocity: 49.81073
Y velocity: 0.018065836

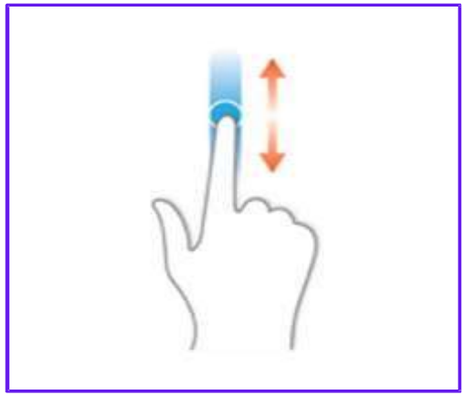
X velocity: 72.73459
Y velocity: -0.029314164

X velocity: 61.3898 Y velocity: -0.0034114174

Vertical swipe X velocity: -0.0026554407
Y velocity: 147.47752

X velocity: -6.275961E-4 Y velocity: 123.17507

X velocity: 2.2685115E- 4
Y velocity: 107.60759



https://developer.android.com/training/gestures/movement

Try this code on your mobile device.
See how the X & Y velocity gives distinct characteristics when you do things like:
slow vertical swipe vs fast vertical swipe
vertical vs horizontal swipe
etc…

```
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
                    // velocity of a motion.
                    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);
                mVelocityTracker.computeCurrentVelocity(1000);
                // Log velocity of pixels per second
                Log.d("", "X velocity: " + mVelocityTracker.getXVelocity(pointerId));
                Log.d("", "Y velocity: " + mVelocityTracker.getYVelocity(pointerId));
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                mVelocityTracker.recycle(); // Return a VelocityTracker object back to be re-
used by others.
                break;
        }
        return true;
    }
}
```

https://developer.android.com/training/gestures/movement

getAction() vs getActionMasked()?

- getAction() returns a pointer id and an event (i.e., up, down, move) information.
- getActionMasked() returns just an event (i.e., up, down, move) information. Other info is masked out
- Eg:
- getAction() returns 0x0105
- getActionMasked() will return 0x0005, which is 0x0105 && ACTION_MASK.
- The value of ACTION_MASK is 0xFF. It masks the following actions.
- ACTION_DOWN 0, UP 1, MOVE 2
- ACTION_POINTER_DOWN 5, UP 6
- The value of ACTION_POINTER_ID_MASK is 0xFF00. It masked the pointer ID from following deprecated constants.
- ACTION_POINTER_1_DOWN 0x0005
- ACTION_POINTER_2_DOWN 0x0105

Pointer (e.g which finger)

https://developer.android.com/training/gestures/movement

Android 10 Gesture Navigation

Gesture navigation

To go Home, swipe up from the bottom of the screen. To switch apps, swipe up from the bottom, hold, then release. To go back, swipe from either the left or right edge.

Android 10

2-button navigation

To switch apps, swipe up on the Home button. To see all apps, swipe up again. To go back, tap the back button.

Android Pie

3-button navigation

Go back, Home and switch apps with buttons at the bottom of your screen.

< Android Pie

https://developer.android.com/training/gestures/gesturenav

https://youtu.be/Ljtz7T8R_Hk