CS371m - Mobile Computing

Gestures
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, move, lift

---

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

**Action**
Press, wait, lift
Common Gestures

- **Drag**: Rearranges data within a view, or moves data into a container (e.g., folders on Home Screen).
- **Double touch**: Zooms into content. Also used as a secondary gesture for text selection.
- **Pinch open**: Zooms into content.

**Action**
- Long press, move, lift
- Two touches in quick succession
- 2-finger press, move outwards, lift

http://developer.android.com/design/patterns/gestures.html
Common Gestures

• 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 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
- in tic tac toe tutorial you only use location of event (x and y)
MotionEvent

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

```java
public final float getSize (int pointerIndex)
```

Returns a scaled value of the approximate size for the given pointer index (use `getPointerId(int)` to find the pointer identifier for this index). This represents some approximation of the area of the screen being pressed; the actual value in pixels corresponding to the touch is normalized with the device specific range of values and scaled to a value between 0 and 1. The value of size can be used to determine **fat touch** events.

**Parameters**

- `pointerIndex` Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()` - 1.
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
Handling Common Gestures

• create a GestureDetector.OnGestureListener (several gestures)
• or a GestureDetector.DoubleTapListener (more gestures) and register it with the GestureDetector
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>onDown(MotionEvent e)</code></td>
<td>Notified when a tap occurs with the down <code>MotionEvent</code> that triggered it.</td>
</tr>
<tr>
<td><code>onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)</code></td>
<td>Notified of a fling event when it occurs with the initial on down <code>MotionEvent</code> and the matching up <code>MotionEvent</code>.</td>
</tr>
<tr>
<td><code>onLongPress(MotionEvent e)</code></td>
<td>Notified when a long press occurs with the initial on down <code>MotionEvent</code> that triggered it.</td>
</tr>
<tr>
<td><code>onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)</code></td>
<td>Notified when a scroll occurs with the initial on down <code>MotionEvent</code> and the current move <code>MotionEvent</code>.</td>
</tr>
<tr>
<td><code>onShowPress(MotionEvent e)</code></td>
<td>The user has performed a down <code>MotionEvent</code> and not performed a move or up yet.</td>
</tr>
<tr>
<td><code>onSingleTapUp(MotionEvent e)</code></td>
<td>Notified when a tap occurs with the up <code>MotionEvent</code> that triggered it.</td>
</tr>
</tbody>
</table>
### Summary

#### Public Methods

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract boolean</td>
<td><strong>onDoubleTap</strong> (MotionEvent e)</td>
<td>Notified when a double-tap occurs.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td><strong>onDoubleTapEvent</strong> (MotionEvent e)</td>
<td>Notified when an event within a double-tap gesture occurs, including the down, move, and up events.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td><strong>onSingleTapConfirmed</strong> (MotionEvent e)</td>
<td>Notified when a single-tap occurs.</td>
</tr>
</tbody>
</table>
Adapter Classes

• OOP Pattern
• Create a class that implements methods of interface with minimal (or no) functionality
• Standard Java Example
• Interfaces for `MouseListener`(5), `MouseWheelListener`(1), and `MouseMotionListener`(3)
• `MouseAdapter` class implements all three interfaces with empty methods
• extend MouseAdapter and add functionality for events you care about.
GestureDetector.SimpleOnGestureListener

- Implements all methods of GestureDetector.OnGestureListener and GestureDetector.DoubleTapListener
- Does nothing but return false for all the methods
- Extend this class and add more meaningful behavior
Simple Gesture Demo

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

```java
@Override
public boolean onTouchEvent(MotionEvent event) {
    gestureDetect.onTouchEvent(event);
    return true;
}
```
Callback Methods for OnGestureListener

```java
@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

```java
@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float deltaX, float deltaY) {
    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

```java
@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;
}
```
Multi Touch Gestures

• Multiple fingers (pointers) touch screen at same time
• Still handled via MotionEvents
• each pointer (finger) has a MotionEvent
• track via index (in array of MotionEvents) or ID
• MotionEvent object sent to onTouch contains number of "pointers" involved
Displaying Multitouch data

- static methods from `MotionEventCompat` class

```java
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getPointerCount() > 1) {
        gestureType.setText("MULTI TOUCH\nEVENT");
        int action = MotionEventCompat.getActionMasked(event);
        gestureType.append("\n" + actionToString(action));
        int index = MotionEventCompat.getActionIndex(event);
        gestureType.append("\nPointer index: "+ index);
    } else
        gestureDetect.onTouchEvent(event);

    return true;
}
```
Scale Gestures

- ScaleGestureDetector class from Api level 8 (API 2.2)
- pinch to zoom in or out
  - out -> scale up
  - in -> scale down
Create class that implements `ScaleGestureDetector.OnScaleGestureListener`

OR create class that extends `ScaleGestureDetector.SimpleOnScaleGestureListener`:
- adapter class
  - implements methods from `OnScaleGestureListener` with dummy methods
  - override only the methods you care about

Create a `ScaleGestureDetector` with listener

pass Motion events from onTouch
Scaling Example

- listener updates overall scale factor
- shows current scale factor in TextView

```java
// from http://developer.android.com/training/gestures/scale.html
private class MyScaleListener
    extends ScaleGestureDetector.SimpleOnScaleGestureListener {
    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        // Log.d("GESTURE DEMO", "Scale factor: " + detector.getScaleFactor());
        scaleFactor *= detector.getScaleFactor();
        // Log.d("GESTURE DEMO", "Scale factor calculated: " + scaleFactor);
        scaleFactor = Math.max(0.001f, Math.min(scaleFactor, 10.0f));
        // Log.d("GESTURE DEMO", "Scale factor clamped: " + scaleFactor);
        double scaleFactorDisplay = ((int) (scaleFactor * 1000)) / 1000.0f;
        scaleFactorTV.setText("SCALE FACTOR: " + scaleFactorDisplay);
        return true;
    }
}
```
Scale Example

LAST GESTURE:

MULTI TOUCH EVENT
Pointer Up
Pointer index: 1

SCALE FACTOR: 0.336
Drag Gestures

• Similar to handling Scale gestures

• Implement View.OnDragListener
  – one method, onDrag(View v, DragEvent de)

• Drag event phases:
  – start
  – continuing
  – dropped
  – ended
COMPLEX GESTURES
Complex Gestures

• Non standard gestures require lots of code to recognize
• Android 1.6 introduced new APIs to store, load, draw, and recognize gestures
• Gesture Builder app on emulator
  – emulator must include virtual SD card
  – allows creating set of gestures for your application
  – limited success with jelly bean emulators
  – App on class GitHub repo
Complex Gestures

• Each gesture associated with name
• Limited to single pointer
• Multiple gestures can have same name
  – variations on same gesture, better chance of recognizing
• Move gestures from emulator to application res/raw folder
Gesture Data File

- DDMS file explorer
Complex Gestures

• Recognizing gestures via a GestureOverlayView
• simple drawing board on top of view that shows and records user gestures
• When gesture complete GestureLibrary queried to see if gesture is recognized
• Predictions of entered gesture and those in the library
Animal Sounds App

Draw M to Moo
Draw P to Oink
Draw C to Meow
## Predictions

<table>
<thead>
<tr>
<th>AnimalSounds</th>
<th>prediction score: 5.020522997579021, name: Oink2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 11.698475110815773, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.4253241939996129, name: Oink3</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.708742452226205, name: Oink</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.7788133409813087, name: Oink</td>
</tr>
<tr>
<td>Choreographer</td>
<td>Skipped 30 frames! The application may be doing</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.5979739128902553, name: Moo2</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.1312601585038455, name: Moo</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.733056893468628, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 0.7404827760194891, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.0095559070264957, name: Moo2</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.408645869375701, name: Moo2</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 2.048106505538496, name: Oink3</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 3.078060118728627, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 2.932816689691991, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.792527999275177, name: Meow</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.8169176605869966, name: Oink3</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 0.7143366373124087, name: Moo</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 1.5232821190754195, name: Oink</td>
</tr>
<tr>
<td>Choreographer</td>
<td>Skipped 32 frames! The application may be doing</td>
</tr>
<tr>
<td>AnimalSounds</td>
<td>prediction score: 0.7857167276876791, name: Moo</td>
</tr>
</tbody>
</table>
onCreate

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mLLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures);
    if (!mLibrary.load()) {
        finish();
    }

    GestureOverlayView gestures = (GestureOverlayView) findViewById(R.id.gestures);
    gestures.addOnGesturePerformedListener(mGestureListener);

    createSoundPool();
}
@Override
public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
    // from http://android-developers.blogspot.com/2009/10/gestures-on-android-

    ArrayList<Prediction> predictions = mLibrary.recognize(gesture);

    // We want at least one prediction
    if (predictions.size() > 0) {
        Prediction prediction = predictions.get(0);

        Log.d(TAG, "prediction score: "+ prediction.score + ", name: " + prediction.name);

        // We want at least some confidence in the result
        if (prediction.score > 3.0) {
            String name = prediction.name;
            if (name.contains("Moo"))
                mSounds.play(mSoundIDMap.get("Moo"), 1, 1, 1, 0, 1);
            else if (name.contains("Oink"))
                mSounds.play(mSoundIDMap.get("Oink"), 1, 1, 1, 0, 1);
            else if (name.contains("Meow"))
                mSounds.play(mSoundIDMap.get("Meow"), 1, 1, 1, 0, 1);
        }
    }