Touch & Gestures

Programming the Android Platform
Topics

- MotionEvents
- Touch Handling
- Gestures
MotionEvent

- Represents a movement in an input device reading
  - e.g., pen, trackball, mouse, finger
- Records event’s time, action, pressure, location, source, and more.
Some common MotionEvent actions

- ACTION_DOWN
- ACTION_UP
- ACTION_POINTER_DOWN
- ACTION_POINTER_UP
- ACTION_MOVE
Some useful MotionEvent methods

- getPointerCount()
- getActionIndex()
- getActionMasked()
- getX()
- getY()
View.onTouchEvent() delivers MotionEvent to the View being touched
If subclassing, onTouchEvent(), return true if the MotionEvent has been consumed; false otherwise
Handling Touch Events

- View.OnTouchListener defines touch event callback methods
  - onTouch() called when user performs a touch event, such as pressing, releasing or dragging an item on the screen
    - Should return true if it has consumed the event
  - View.setOnTouchListener() registers listener for Touch callbacks
1-touch – each touch is a single event
  - Process ACTION_DOWN, ACTION_MOVE & ACTION_UP independently
2+-touch – multiple touches combined to form a gesture
  - Identify & process combinations of touches such as ACTION_DOWN, followed by ACTION_POINTER_DOWN
  - Some common Gestures can be recognized by GestureDetectors
    - E.g., double tap, fling, confirmed single tap, etc.
## Multi-touch Handling

<table>
<thead>
<tr>
<th>Action</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_DOWN</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_POINTER_DOWN</td>
<td>1</td>
</tr>
<tr>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_POINTER_UP</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_UP</td>
<td>1</td>
</tr>
</tbody>
</table>
Multi-touch Handling (cont.)

<table>
<thead>
<tr>
<th>#1 touch</th>
<th>Action</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACTION_DOWN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>#2 touch</td>
<td>ACTION_POINTER_DOWN</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>#2 lift</td>
<td>ACTION_POINTER_UP</td>
<td>1</td>
</tr>
<tr>
<td>#1 lift</td>
<td>ACTION_UP</td>
<td>0</td>
</tr>
</tbody>
</table>
Multi-touch Handling (cont.)

<table>
<thead>
<tr>
<th>Action</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_DOWN</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_POINTER_DOWN</td>
<td>1</td>
</tr>
<tr>
<td>ACTION_POINTER_DOWN</td>
<td>2</td>
</tr>
<tr>
<td>ACTION_MOVE</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_POINTER_UP</td>
<td>1</td>
</tr>
<tr>
<td>ACTION_POINTER_UP</td>
<td>0</td>
</tr>
<tr>
<td>ACTION_UP</td>
<td>2</td>
</tr>
</tbody>
</table>

#1 touch → ACTION_DOWN 0
#2 touch → ACTION_POINTER_DOWN 1
#3 touch → ACTION_POINTER_DOWN 2
#2 lift → ACTION_POINTER_UP 1
#1 lift → ACTION_POINTER_UP 0
#3 lift → ACTION_UP 2
public class IndicateTouchLocationActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        ...
        final FrameLayout frame = ...
        frame.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                int actionCode = event.getActionMasked();
                for (int i = 0; i < event.getPointerCount(); i++) {
                    switch (actionCode) {
                        case /* interesting motion event */ : {
                            // make a view showing touch location
                            frame.addView(/* new view */);
                        }
                        ...
                    }
                }
            }
        });
    }
}
public class ViewFlipperTestActivity extends Activity {
    private ViewFlipper mFlipper;
    private int mCurrentLayoutState;
    private GestureDetector gd;
    public void onCreate(Bundle savedInstanceState) {
        ...
        mFlipper = (ViewFlipper) findViewById(R.id.view_flipper);
        mCurrentLayoutState = 0;
        ....
    }
}
... 

gd = new GestureDetector(this,
        new GestureDetector.SimpleOnGestureListener()
        {
            public boolean onFling(MotionEvent e1, MotionEvent e2,
                                    float velocityX, float velocityY) {
                if (velocityX < -10.0f) { /* change view */ }
                return true;
            }
        });

public boolean onTouchEvent(MotionEvent event) {
    return gd.onTouchEvent(event);
}
Custom Gestures

- GestureBuilder allows developers to create & save custom gestures
- GestureLibraries supports loading custom gestures & recognizing then at runtime
- GestureOverlayView intercepts user gestures and invokes application code to handle them
GestureBuilder

- Comes bundled with emulator
- Stores gestures to /mnt/sdcard/gestures
- Copy this file to /res/raw directory
GestureOverlayView

- View intercepts gestures
- Invokes OnGesturePerformedListener
public class GesturesActivity extends Activity implements OnGesturePerformedListener {
    private GestureLibrary mLibrary;
    public void onCreate(Bundle savedInstanceState) {
        ...
        mLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures);
        if (!mLibrary.load()) { finish(); }
        GestureOverlayView gestures =
            (GestureOverlayView) findViewById(R.id.gestures);
        gestures.addOnGesturePerformedListener(this);
        ...
    }
    ...
}
public void onGesturePerformed(
        GestureOverlayView overlay, Gesture gesture) {
    ArrayList<Prediction> predictions = mLibrary.recognize(gesture);
    if (predictions.size() > 0) {
        Prediction prediction = predictions.get(0);
        if (prediction.score > 1.0) {
            // Check prediction.name and take appropriate action
        }
    }
}
Source Code Examples

- TouchIndicateTouchLocation
- TouchGestureViewFlipper
- TouchGestures