Recording in Progress

This class is being recorded

Please turn off your video and/or video if you do not wish to be recorded
CMSC436: Programming Handheld Systems
Multi-Touch & Gestures
Today’s Topics

MotionEvents
Touch Handling
Gestures
MotionEvent

Represents a movement in an input device reading

pen, trackball, mouse, finger
MotionEvent

Action Code

State change that occurred

Action Values

Position and movement properties, such as time, source, location, pressure, and more

This lesson focuses on touch events read from a touch screen
MultiTouch

MultiTouch screens emit one movement trace per touch source

Individual touch sources are called pointers
MultiTouch

Each pointer has a unique ID for as long as it is active.

MotionEvents can refer to multiple pointers.

Each pointer has an index within the event, but that index may not be stable over time.
Some MotionEvent actions

ACTION_DOWN
ACTION_POINTER_DOWN
ACTION_POINTER_UP
ACTION_MOVE
ACTION_UP
ACTION_CANCEL
Consistency Guarantees*

For touch events, Android tries to guarantee that touches:

- Go down one at a time
- Move as a group
- Come up one at a time or are cancelled

Applications should be tolerant to inconsistency.
MotionEvent methods

getActionMasked()
getActionButtonIndex()
getPointerId(int pointerIndex)
getPointerCount()
getX(int pointerIndex)
getY(int pointerIndex)
findPointerIndex (int pointerId)
Handling Touch Events on a View

The View being touched receives
View.onTouchEvent(MotionEvent event)
onTouchEvent() should return true if the MotionEvent has been consumed; false otherwise
Handling Touch Events with a Listener

View.OnTouchListener defines touch event callback methods

  boolean onTouch(View v, MotionEvent event)

View.setOnTouchListener() registers listener for Touch callbacks
Handling Touch Events with a Listener

onTouch() called when a touch event, such as pressing, releasing or dragging, occurs

onTouch() called before the event is delivered to the touched View

Should return true if it has consumed the event; false otherwise
Handling Multiple Touch Events

Multiple touches combined to form a more complex gesture

Identify & process combinations of touches,

For example, a double tap

    ACTION_DOWN, ACTION_UP, ACTION_DOWN, ACTION_UP in quick succession
Multi-touch Handling
## Multi-touch Handling Example

<table>
<thead>
<tr>
<th>Action</th>
<th>IDs</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>0,1</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 Example

<table>
<thead>
<tr>
<th>Action</th>
<th>IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st touch ACTION_DOWN</td>
<td>0</td>
</tr>
<tr>
<td>1st touch ACTION_MOVE ...</td>
<td>0</td>
</tr>
<tr>
<td>2nd touch ACTION_POINTER_DOWN</td>
<td>1</td>
</tr>
<tr>
<td>2nd touch ACTION_MOVE ...</td>
<td>0,1</td>
</tr>
<tr>
<td>2nd lift ACTION_POINTER_UP</td>
<td>1</td>
</tr>
<tr>
<td>1st lift ACTION_UP</td>
<td>0</td>
</tr>
</tbody>
</table>
## Multi-touch Handling Example

<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>0,1,2</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>
TouchIndicateTouchLocation

Application draws a circle wherever the user touches the screen

  Circle’s color is randomly selected

Redraws circles as user drags across the screen
TouchIndicateTouchLocation

The size of the circles are proportional to the number of currently active touches
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    mFrame = findViewById(android.R.id.content)

    // Initialize pool of View.
    initViewViews()
    ...
}

private fun initViewViews() {
    for (idx in 0 until MAX_TOUCHES) {
        mInactiveMarkers.add(MarkerView(this, -1f, -1f))
    }
}
// Create and set on touch listener
mFrame.setOnTouchListener(object : OnTouchListener {
    override fun onTouch(v: View, event: MotionEvent): Boolean {
        ...
        when (event.actionMasked) {
            // Show new MarkerView
            MotionEvent.ACTION_DOWN, MotionEvent.ACTION_POINTER_DOWN ->
                val pointerIndex = event.actionIndex
                val pointerID = event.getPointerId(pointerIndex)
                val marker = mInactiveMarkers.remove()
                marker?.apply {
                    mActiveMarkers[pointerID] = this
                    xLoc = event.getX(pointerIndex)
                    yLoc = event.getY(pointerIndex)
                    updateTouches(mActiveMarkers.size)
                    mFrame.addView(this)
                }
        }
    }
})
// Remove one MarkerView
MotionEvent.ACTION_UP, MotionEvent.ACTION_POINTER_UP -> {
    val pointerIndex = event.actionIndex
    val pointerID = event.getPointerId(pointerIndex)
    val marker = mActiveMarkers.remove(pointerID)
    marker?.apply {
        mInactiveMarkers.add(this)
        updateTouches(mActiveMarkers.size)
        mFrame.removeView(this)
    }
}
// Move all currently active MarkerViews
MotionEvent.ACTION_MOVE -> {
  for (idx in 0 until event.pointerCount) {
    val currId = event.getPointerId(idx)
    val marker = mActiveMarkers[currId]
    marker?.apply {
      // Redraw only if finger has traveled a minimum distance
      if (abs(xLoc - event.getX(idx)) > MIN_DXDY ||
          abs(yLoc - event.getY(idx)) > MIN_DXDY) {
        // Set new location
        xLoc = event.getX(idx)
        yLoc = event.getY(idx)
        // Request re-draw
        invalidate()
      }
    }
    ... return true
private inner class MarkerView internal constructor( context: Context, internal var xLoc: Float, internal var yLoc: Float): View(context) {
private var mTouches = 0
private val mPaint = Paint()
init {
    mPaint.style = Style.FILL
    val rnd = Random()
    mPaint.setARGB(255, rnd.nextInt(256), rnd.nextInt(256), rnd.nextInt(256))
}

override fun onDraw(canvas: Canvas) {
    canvas.drawCircle(xLoc, yLoc, (MAX_SIZE / mTouches), mPaint)
}
GestureDetector

A class that recognizes common touch gestures
Some built-in gestures include confirmed single tap, double tap, fling
GestureDetector

Activity creates a GestureDetector which implements GestureDetector.OnGestureListener interface

Activity will receive calls to onTouchEvent() when Activity is touched

onTouchEvent delegates call to GestureDetector.OnGestureListener
TouchGestureViewFlipper

Shows a TextView displaying a number
If the user performs a right to left “fling” gesture,
The TextView will scroll off the screen
A new TextView will scroll in behind it
main.xml

```
<ViewFlipper xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/view_flipper"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <include
        layout="@layout/first"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"/>
    <include
        layout="@layout/second"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"/>
</ViewFlipper>
```
first.xml

<merge xmlns:android="http://schemas.android.com/apk/res/android">
  <TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:gravity="center"
    android:textAppearance="@android:style/TextAppearance.Material.Display4"
    android:textSize="@dimen/text_size"/>
</merge>
public override fun onCreate(savedInstanceState: Bundle?) {
    ...
    mGestureDetector = GestureDetector(this,
        object : GestureDetector.SimpleOnGestureListener() {
            override fun onFling(e1: MotionEvent, e2: MotionEvent,
                velocityX: Float, velocityY: Float): Boolean {
                if (velocityX < -10.0f) {
                    mCurrentLayoutState = if (mCurrentLayoutState == 0)
                        1 else 0
                    switchLayoutStateTo(mCurrentLayoutState)
                } else {
                    return true
                }
            }
        })
    }
}
override fun onTouchEvent(event: MotionEvent): Boolean {
    return mGestureDetector.onTouchEvent(event)
}

private fun switchLayoutStateTo(switchTo: Int) {
    mCurrentLayoutState = switchTo
    mFlipper.inAnimation = inFromRightAnimation()
    mFlipper.outAnimation = outToLeftAnimation()
    mCount++
    if (switchTo == 0) {
        mTextView1.text = mCount.toString()
    } else {
        mTextView2.text = mCount.toString()
    }
    mFlipper.showPrevious()
}
private fun inFromRightAnimation(): Animation {
    val inFromRight = TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, +1.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f
    )
    inFromRight.duration = 500
    inFromRight.interpolator = LinearInterpolator()
    return inFromRight
}
private fun outToLeftAnimation(): Animation {
    val outToLeft = TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, -1.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f
    )
    outToLeft.duration = 500
    outToLeft.interpolator = LinearInterpolator()
    return outToLeft
}
Creating Custom Gestures

The GestureBuilder application lets you create & save custom gestures

Comes bundled with SDK
Creating Custom Gestures

GestureLibraries supports loading custom gestures & then recognizing them at runtime.
Creating Custom Gestures

Include a GestureOverlayView in your layout

The Overlay intercepts user gestures and invokes your application code to handle them
GestureBuilder

Stores gestures to /mnt/sdcard/gestures

Copy this file to /res/raw directory
Application displays a small View with a colored background.

User can swipe left and right to cycle between different candidate background colors.

Can make a “check” or “X-like gesture” to set or cancel the application’s current background color.
main.xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <android.gesture.GestureOverlayView
        android:id="@+id/gestures_overlay"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_centerInParent="true">
        <FrameLayout
            android:id="@+id/frame"
            android:layout_width="@dimen/target_size"
            android:layout_height="300dp"
            android:layout_gravity="center" />
    </android.gesture.GestureOverlayView>
</RelativeLayout>
public override fun onCreate(savedInstanceState: Bundle?) {
    // Add gestures file – contains 4 gestures: Prev, Next, Yes, No
    mLLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures)
    if (mLibrary?.load()) {
        finish()
    }

    // Make this the target of gesture detection callbacks
    val gestureView
        findViewById<GestureOverlayView>(R.id.gestures_overlay)
    gestureView.addOnGesturePerformedListener(this)
}
override fun onGesturePerformed(overlay: GestureOverlayView, gesture: Gesture) {

    // Get gesture predictions
    val predictions = mLibrary.recognize(gesture)

    // Get highest-ranked prediction
    if (predictions.size > 0) {
        val prediction = predictions[0]
    }

    ...
// Ignore weak predictions
if (prediction.score > 2.0) {
    when (prediction.name) {
        PREV -> {
            mBgColor -= 100
            mFrame.setBackgroundColor(mBgColor)
        }
        NEXT -> {
            mBgColor += 100
            mFrame.setBackgroundColor(mBgColor)
        }
        YES -> mLayout.setBackgroundColor(mBgColor)
        NO -> {
            mLayout.setBackgroundColor(mStartBgColor)
            mFrame.setBackgroundColor(mFirstColor)
        }
    }
}
GesturesActivity.kt


Toast.makeText(this, prediction.name, Toast.LENGTH_SHORT).show();
Next Time

MultiMedia
Example Applications

TouchIndicateTouchLocation
TouchGestureViewFlipper
TouchGestures