# CMSC436: Programming Handheld Systems

Fall 2017

# 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()

getActionIndex()

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

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

| | Action | IDs |
|---|---|---|
| 1st touch → | ACTION_DOWN | 0 |
| | ACTION_MOVE … | 0 |
| 2nd touch → | ACTION_POINTER_DOWN | 1 |
| | ACTION_MOVE … | 0,1 |
| 1st lift → | ACTION_POINTER_UP | 0 |
| 2nd lift → | ACTION_UP | 1 |

# Multi-touch Handling Example

| | **Action** | **IDs** |
|---|---|---|
| 1st touch → | ACTION_DOWN | 0 |
| | ACTION_MOVE … | 0 |
| 2nd touch → | ACTION_POINTER_DOWN | 1 |
| | ACTION_MOVE … | 0,1 |
| 2nd lift → | ACTION_POINTER_UP | 1 |
| 1st lift → | ACTION_UP | 0 |

# Multi-touch Handling Example

| | Action | ID |
|---|---|---|
| 1st touch → | ACTION_DOWN | 0 |
| 2nd touch → | ACTION_POINTER_DOWN | 1 |
| 3rd touch → | ACTION_POINTER_DOWN | 2 |
| | ACTION_MOVE | 0,1,2 |
| 2nd lift → | ACTION_POINTER_UP | 1 |
| 1st lift → | ACTION_POINTER_UP | 0 |
| 3rd lift → | ACTION_UP | 2 |

# TouchIndicateTouchLocation

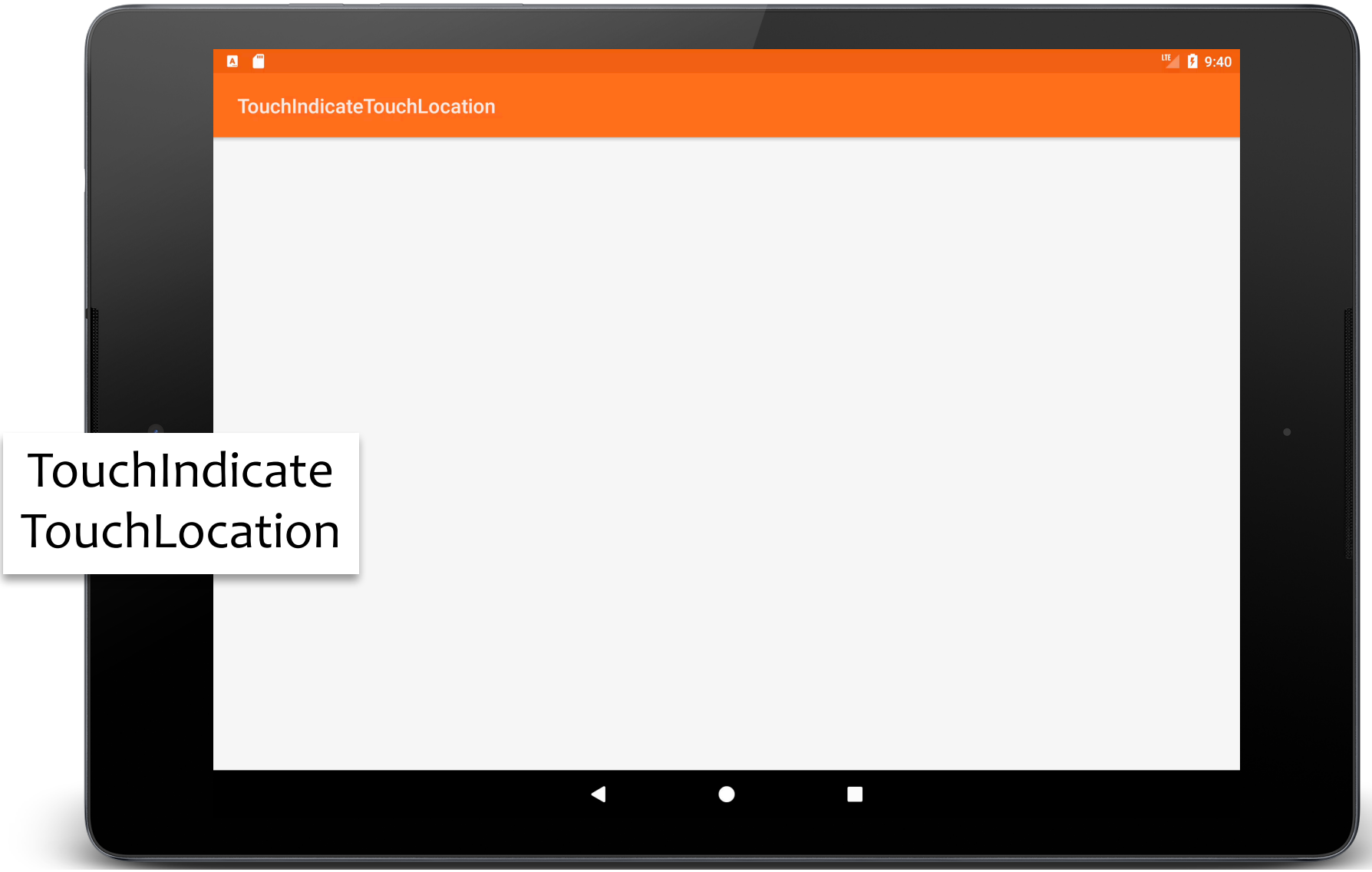
Application draws a circle wherever the users touches the screen

- 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


9:40
TouchIndicateTouchLocation
TouchIndicate
TouchLocation

```
public void onCreate(Bundle savedInstanceState) {
   …
   mFrame = findViewById(android.R.id.content);

   // Initialize pool of View.
   initViews();

   …
}

private void initViews() {
   for (int idx = 0; idx < MAX_TOUCHES; idx++) {
      mInactiveMarkers.add(new MarkerView(this, -1, -1));
   }
}
```

```
public void onCreate(Bundle savedInstanceState) {
    …
    // Create and set on touch listener
    mFrame.setOnTouchListener(new OnTouchListener() {
       public boolean onTouch(View v, MotionEvent event) {
          …
```

```
…
switch (event.getActionMasked()) {
   // Show new MarkerView
   case MotionEvent.ACTION_DOWN:
    case MotionEvent.ACTION_POINTER_DOWN: {
       int pointerIndex = event.getActionIndex();
       int pointerID = event.getPointerId(pointerIndex);
       MarkerView marker = mInactiveMarkers.remove();
       if (null != marker) {
          mActiveMarkers.put(pointerID, marker);
          marker.setXLoc(event.getX(pointerIndex));
          marker.setYLoc(event.getY(pointerIndex));
          updateTouches(mActiveMarkers.size());
          mFrame.addView(marker);
       }
       break;
   }
…
```

```
// Remove one MarkerView
case MotionEvent.ACTION_UP:
 case MotionEvent.ACTION_POINTER_UP: {
    int pointerIndex = event.getActionIndex();
    int pointerID = event.getPointerId(pointerIndex);
    MarkerView marker = mActiveMarkers.remove(pointerID);
    if (null != marker) {
       mInactiveMarkers.add(marker);
       updateTouches(mActiveMarkers.size());
       mFrame.removeView(marker);
    }
    break;
}
  …
```

```
// Move all currently active MarkerViews
   case MotionEvent.ACTION_MOVE: {
      for (int idx = 0; idx < event.getPointerCount(); idx++) {
         int ID = event.getPointerId(idx);
         MarkerView marker = mActiveMarkers.get(ID);
         if (null != marker) {
            // Redraw only if finger has traveled a minimum distance
            if (Math.abs(marker.getXLoc() - event.getX(idx)) > MIN_DXDY
                  || Math.abs(marker.getYLoc() - event.getY(idx)) > MIN_DXDY) {
               // Set new location
               marker.setXLoc(event.getX(idx));
               marker.setYLoc(event.getY(idx));
               // Request re-draw
               marker.invalidate();
               …
return true;
```

```
private class MarkerView extends View {
   …
   public MarkerView(Context context, float x, float y) {
      super(context);
      mX = x;       mY = y;
      mPaint.setStyle(Style.FILL);
      Random rnd = new Random();
      mPaint.setARGB(255, rnd.nextInt(256), rnd.nextInt(256),
            rnd.nextInt(256));
   }
   …
   protected void onDraw(Canvas canvas) {
      canvas.drawCircle(mX, mY, 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

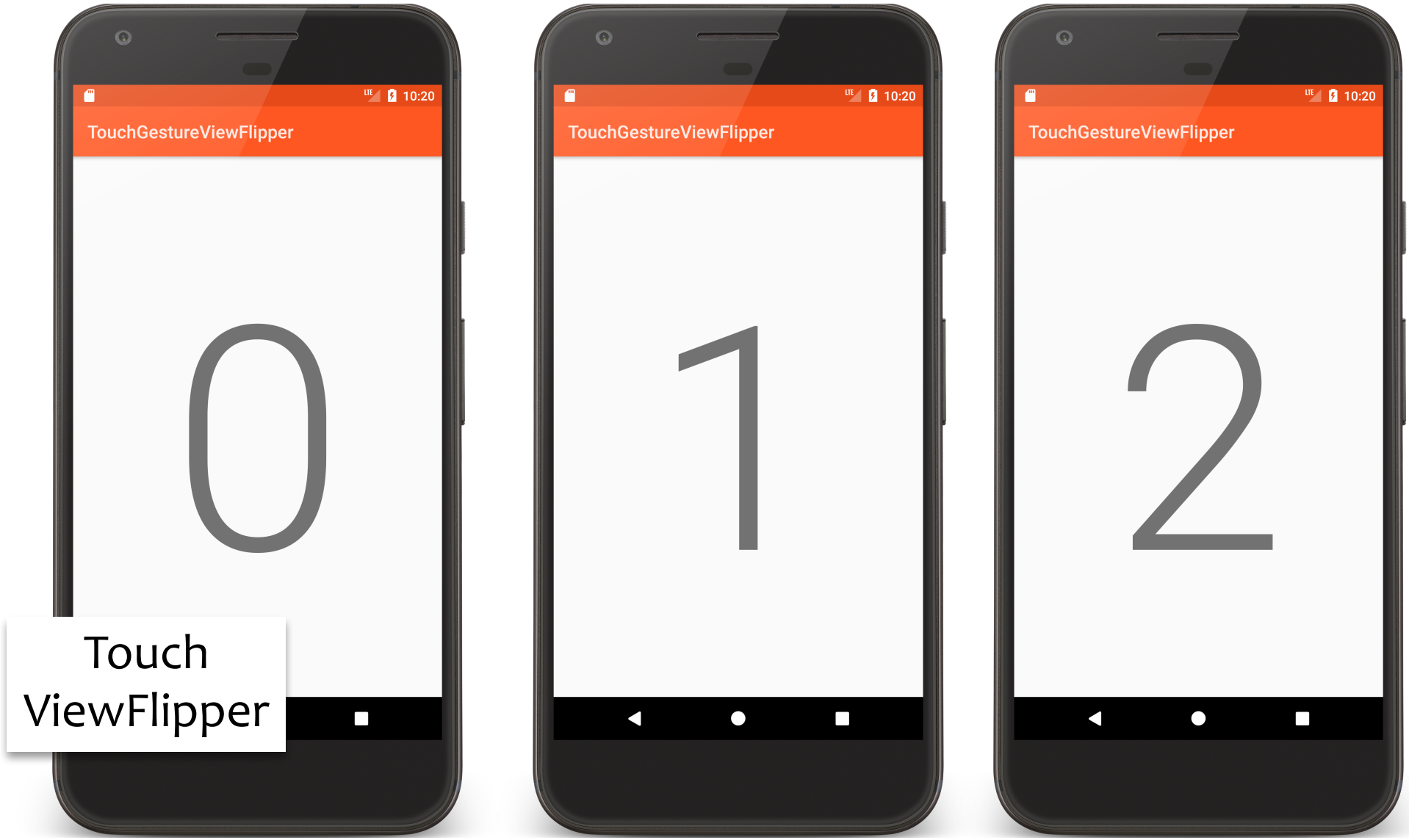
# TouchGestureViewFlipper

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


10:20
TouchGestureViewFlipper
0
10:20
TouchGestureViewFlipper
1
10:20
TouchGestureViewFlipper
2
Touch
ViewFlipper

```
<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>
```

```
<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 void onCreate(Bundle savedInstanceState) {
  …
  mGestureDetector = new GestureDetector(this,
      new GestureDetector.SimpleOnGestureListener() {
        public boolean onFling(MotionEvent e1, MotionEvent e2,
                                                    float velocityX, float velocityY) {
          if (velocityX < -10.0f) {
            mCurrentLayoutState = mCurrentLayoutState == 0 ? 1 : 0;
            switchLayoutStateTo(mCurrentLayoutState);
          }
          return true;
        } }); }

public boolean onTouchEvent(MotionEvent event) {
  return mGestureDetector.onTouchEvent(event);
}
```

```
private void switchLayoutStateTo(int switchTo) {
  mCurrentLayoutState = switchTo;

  mFlipper.setInAnimation(inFromRightAnimation());
  mFlipper.setOutAnimation(outToLeftAnimation());

  mCount++;

  if (switchTo == 0) {
    mTextView1.setText(String.valueOf(mCount));
  } else {
    mTextView2.setText(String.valueOf(mCount));
  }

  mFlipper.showPrevious();
}
```

```
private Animation inFromRightAnimation() {
    Animation inFromRight = new 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.setDuration(500);
    inFromRight.setInterpolator(new LinearInterpolator());
    return inFromRight;
 }
```

```
  private Animation outToLeftAnimation() {
    Animation outToLeft = new 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.setDuration(500);
    outToLeft.setInterpolator(new 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

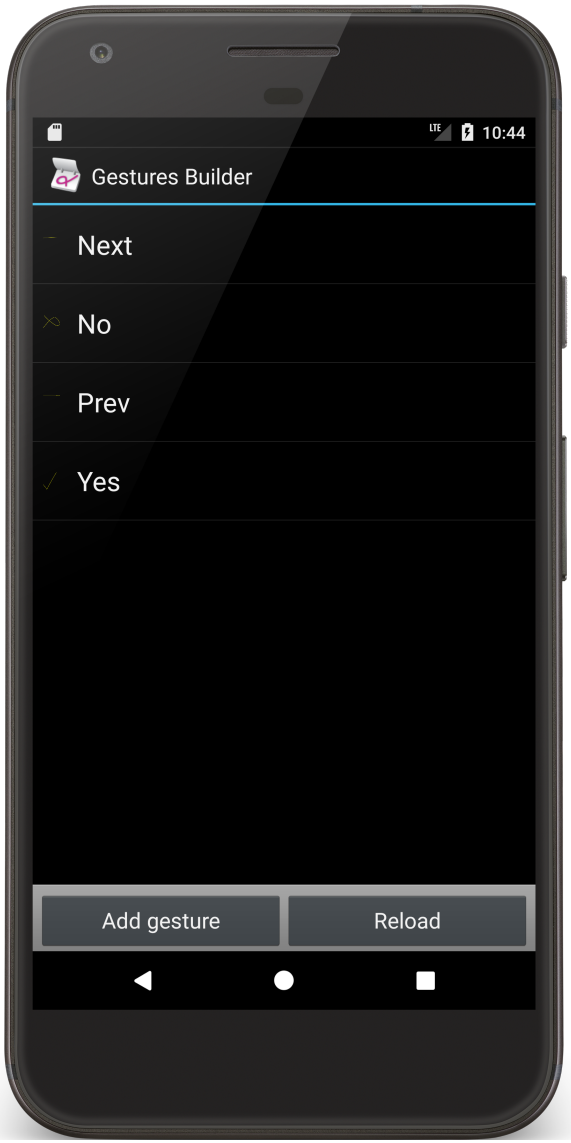
Include a GestureOverlayView in your layout

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


10:44
Gestures Builder
Next
No
Prev
Yes
Add gesture
Reload

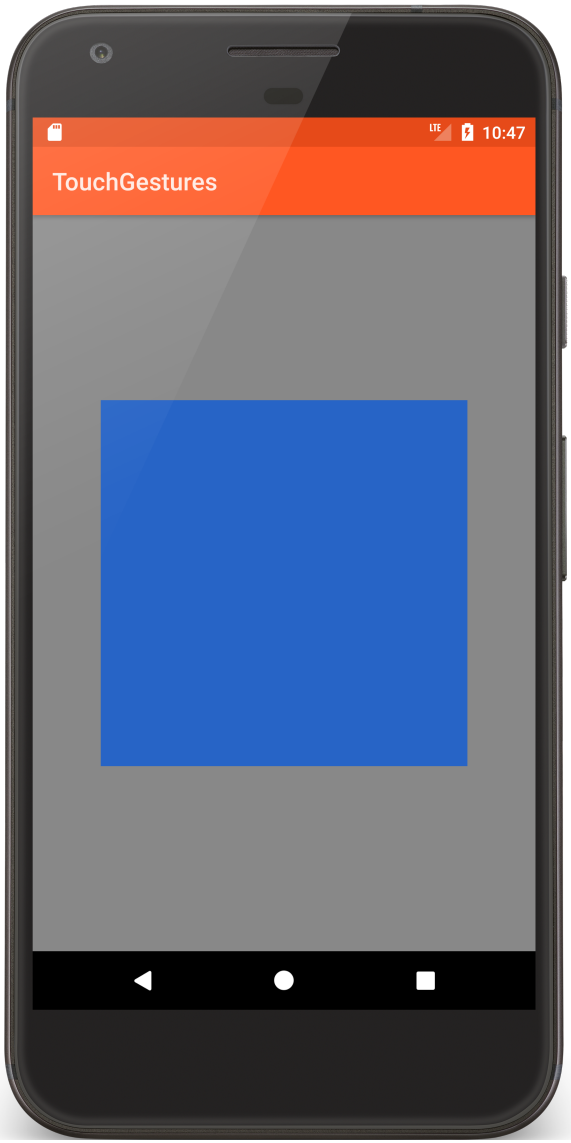
# TouchGestures

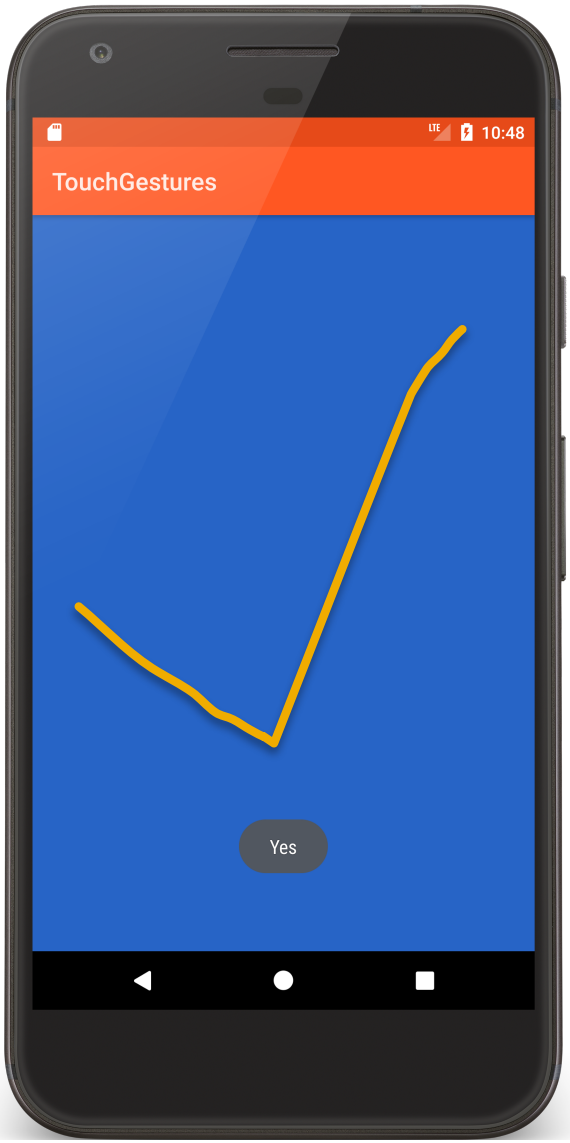
Application displays a small View with a colored background

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

Can make an check or X-like gesture to set or cancel the application’s current background color

Touch Gestures



```
<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" >
      </FrameLayout>
   </android.gesture.GestureOverlayView>
</RelativeLayout>
```

```
public class GesturesActivity extends Activity implements OnGesturePerformedListener {
…
   public void onCreate(Bundle savedInstanceState) {
    …
      // Add gestures file - contains 4 gestures: Prev, Next, Yes, No
      mLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures);
      if (!mLibrary.load()) {
         finish();
      }

      // Make this the target of gesture detection callbacks
      GestureOverlayView gestureView = findViewById(R.id.gestures_overlay);
      gestureView.addOnGesturePerformedListener(this);

   }
```

```
public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {

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

    // Get highest-ranked prediction
    if (predictions.size() > 0) {
        Prediction prediction = predictions.get(0);
    ...
```

```
// Ignore weak predictions
        if (prediction.score > 2.0) {
            switch (prediction.name) {
                case PREV:
                    mBgColor -= 100;
                    mFrame.setBackgroundColor(mBgColor);
                    break;

                case NEXT:
                    mBgColor += 100;
                    mFrame.setBackgroundColor(mBgColor);
                    break;

                case YES:
                    mLayout.setBackgroundColor(mBgColor);
                    break;
                ...
```

```
            case NO:
                    mLayout.setBackgroundColor(mStartBgColor);
                    mFrame.setBackgroundColor(mFirstColor);
                    break;
                }
                Toast.makeText(this, prediction.name, Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "No prediction", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "No prediction", Toast.LENGTH_SHORT).show();
        }
    }
}
```

# Next Time

MultiMedia