LE/EECS 4443: Mobile User Interfaces (LAB)

Week 5: Gestures with Android Studio

S.Toyonaga[1]

[1]Lassonde School of Engineering
York University

February 9, 2024

Table of Contents

1 Introduction

2 Gestures

3 Conclusion

Introduction

By the end of this tutorial, you will be able to...

1. Use Simple and Complex Gestures in your Android Application
2. Implement your Own Gesture Detectors using MotionEvent
3. Play sounds from Android Studio

Gestures: Introduction

Remark

A **touch gesture** occurs when a user places **one or more fingers** on the touch screen and your application interprets this pattern of touches as a **gesture**. There are two phases to gesture detection:

1. Gathering touch event data
2. Interpreting the data to determine whether it meets the criteria for the gestures your app supports.

Source: https://developer.android.com/develop/ui/views/touch-and-input/gestures/detector

Components of Touch Events

1. View.OnTouchListener
2. MotionEvent Objects & States

Detecting Gestures

1. GestureDetector Class (Extends $\cdots$)
   - **(Single-Class)** GestureDetector.SimpleOnGestureListener
   - **(Multi-Class)** GestureDetector.SimpleOnScaleGestureListener
2. GestureDetector Interface (Implements $\cdots$)
   - GestureDetector.OnGestureListener
   - GestureDetector.OnDoubleTapListener
3. **onTouch** $\wedge$ **MotionEvent** $\rightarrow$ DIY / Customization!

Connecting GestureDetector to Views

1. Create a nested Java class that extends the GestureDetector class
2. Override & Implement the required methods
3. Initialize your GestureDetector
4. Implement **onTouch** on your desired View
5. In **onTouch**, return gesture_detector_object.onTouchEvent(motionEvent)

Demo: TouchEvents

GestureDetector & Views

Remark

Whether you use *GestureDetector.OnGestureListener* or *GestureDetector.SimpleOnGestureListener*, it's a best practice to **implement an onDown() method that returns true**. This is because all gestures begin with an onDown() message. If you return false from onDown(), as GestureDetector.SimpleOnGestureListener does by default, the system assumes you want to ignore the rest of the gesture, and the other methods of GestureDetector.OnGestureListener aren't called. This might cause unexpected problems in your app. Only return false from onDown() if you truly want to ignore an entire gesture.

Source: https://developer.android.com/develop/ui/views/touch-and-input/gestures/detector

Conclusion

Remark

Thank you for your time!
Do you have any questions? :)