MONASH University

Information Technology

FIT5183: Mobile and Distributed Computing Systems (MDCS)

Lecture 8A
Working with Touch Input & Sensors on Android

Associate Professor Cheng-Siong Lee

SEU-Monash Univesity Joint Graduate School (Suzhou)

Vincent.cs.lee@monash.edu

Overview

- Reviewing approaches to handle touch input on Android
  - Single touch input
  - Gestures (such as swipes)
  - Multi-touch gestures
- Working with sensors such as the Accelerometer and Gyroscope to get data about the device's environment.
  - Movement of the device
  - Orientation of the device

Working with Touch Input

Mobile Touch Interactions

- The iPhone pioneered the use of multi-touch interactions for mobile devices.
  - Many of Apple's developments can be seen as the defacto approach to interacting with touch-enabled mobile devices.
  - Very rare now to find a modern smartphone that does not support a touch interface.

- Both Android and iOS natively support multi-touch interactions.
  - We can work with touch inputs individually to establish multitouch interactions (such as pinch to zoom).
  - Important to note that **not all Android devices use touch screens** (hence why we use onClick listeners).

Android: MotionEvent

- Event which reports movement based upon the input method used (such as mouse, pen or touch).
- MotionEvent objects are created at the point of the interaction.
  - E.g. A finger has touched the screen.
- MotionEvents are present in all touch interactions and gestures performed in Android.
  - We can work with these objects to perform additional actions beyond a simple tap.
  - Example: When your finger moves across the screen, we could draw a line from start to end.

MotionEvent: Actions

- MotionEvent describe the interaction that took place:
  - Action code describes the state of the interaction.
  - Axis values describes the movement and position properties.
- getAction() allows us to retrieve the type of action performed:
  - **MotionEvent.ACTION_DOWN**
    Returned when the user has placed their finger on the screen.
  - **MotionEvent.ACTION_MOVE**
    Returned when user has moved their finger away from starting position.
  - **MotionEvent.ACTION_UP**
    Returned when user has removed their finger.
- There are many possible actions available via MotionEvent.
  - Review the API for the full list.

Using MotionEvents

- We can work with MotionEvent objects through two approaches:
- An **Activity** which implements onTouchEvent(MotionEvent):

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    Log.d("TouchEvent", event.toString());
    return true;
}
```

- A **view** via setOnTouchListener(OnTouchListener):

```
view.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event)
    {
        Log.d("TouchEvent", event.toString());
        return true;
    }
}
```

Gestures

- Gestures are single or multiple touch points that represent an action that can be performed.
  - Enable mapping of complex interactions that perform specific tasks.
  - Can be challenging for applications to interpret complex gestures.
  - Example: Using pinch gesture to zoom in or out of a view.
- Some examples of gestures:
  - Drag
  - Double touch
  - Long press
  - Pinch (Open, Close)
  - Swipe
  - Touch

Using Gestures in Android

- The **GestureDetector** class is used for detecting common gestures in Android applications.
  - Supports common gestures such as onLongPress(), onScroll() and onFling().
  - GestureDetetectorCompat is used instead when supporting older Android devices using the support library.
- One of two possible gesture listeners need to be used to work with gestures:
  - **GestureDetector.OnGestureListener**()
    Forces you to handle all supported gestures.
  - **GestureDetector.SimpleOnGestureListener**()
    Enables you to handle specific gestures.

Example: Android Gestures

- First we build a custom GestureListener to support the specific gestures we wish to employ in our application.

```
public class GestureListener extends SimpleOnGestureListener
{
    @Override
    public boolean onDoubleTapEvent(MotionEvent e) {
        Log.d("GestureListener", "Double Tap");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent downEvent,
    MotionEvent upEvent, float velocityX, float velocityY) {
    Log.d("GestureListener", "Fling");
    return true;
  }
}
```

Example: Android Gestures (cont.)

- We then apply an instance of GestureListener object to an instance of GestureDetector in an Activity.

```
private static GestureDetector gDetector;

protected void onCreate(Bundle savedInstanceState) {
    // Init GestureDetector with our listener
    gDetector = new GestureDetector(this,
    new GestureListener(this));
}
@Override
public boolean onTouchEvent(MotionEvent event) {
    // Send GestureDetector any touch events to process
    gDetector.onTouchEvent(event);
    return true;
}
```

Working with Sensors

Mobile Sensors

- Many mobile smartphones and tablets support a number of different sensors to return data from the environment.
- Mobile sensors can be grouped into three categories:
  - **Motion Sensors**
    Sensors measure the acceleration and rotational forces being applied to the device.
    Example: **Accelerometers** and **Gyroscopes**
  - **Environment Sensors**
    Sensors measure environmental variables such as temperature and illumination.
    Example: **Photometers** and **Thermometers**
  - **Position Sensors**:
    Sensors measure the physical state of the device.
    Example: **Orientation** sensor

Common Sensors for Mobiles

- In particular, there are some sensors that are often used in applications:
- **Accelerometer**
  - Sensor that provides data on movement and gravity.
  - Can also provide details of orientation such as X, Y, Z coords.
- **Gyroscope**
  - Helps calculate the orientation and rotation of a device.
  - Often used to determine the orientation of the device and change the interface layout accordingly.
- Near Field Communication (**NFC**)
  - Enables communication between devices via touching them together and communicating by radio.
  - Starting to appear in many mobile devices.

Android: Sensor Framework

- We use the Android Sensor Framework to work with sensors that are available on the device.
- Android platform supports a large number of sensor types.
  - The vast array of devices has made many different types of unique sensors available.
- Examples of such sensors:
  - Ambient humidity
  - Ambient temperature
  - Magnetic Field
  - Pressure
  - Proximity
  - …

Source:
http://developer.android.com/guide/topics/sensors/sensors_overview.html

Android: Sensor Framework

- Sensor framework provides the following classes and interfaces:
  - **SensorManager**
    Class enables access to an instance of the sensor service. You can access methods which can access and find sensors for a particular device and register/unregister SensorEvent listeners.
  - **Sensor**
    Class enables access to an instance of a particular sensor.
  - **SensorEvent**
    Provides information for a particular sensor that created the event.
  - **SensorEventListener**
    Provides an interface for callback methods and receive SensorEvents when variables change.

Android: Accelerometer

- An example of using the Accelerometer to get X, Y, Z coordinates.

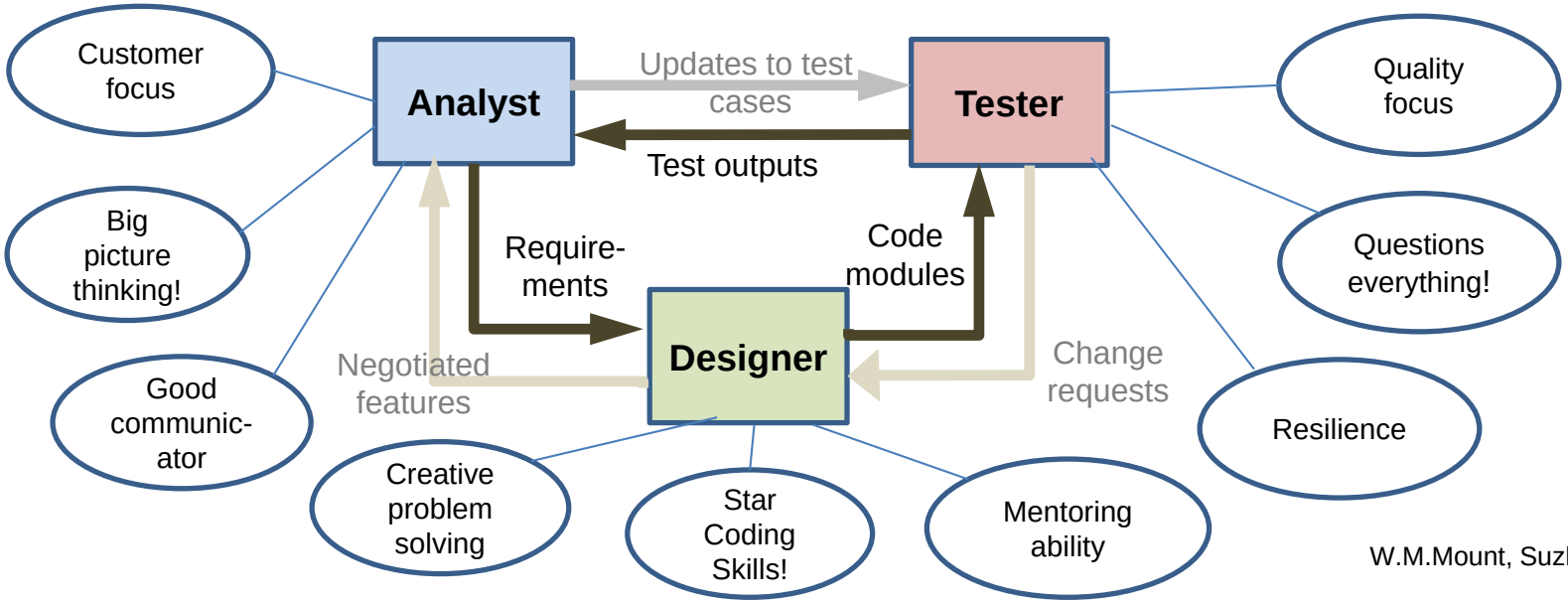
```
private SensorManager mSensorManager;
private Sensor mAccelerometer;

mSensorManager =
(SensorManager)getSystemService(Context.SENSOR_SERVICE);
mAccelerometer =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETE
R);
…
public void onSensorChanged(SensorEvent event) {
    //Send GestureDetector any touch events to process”
    Log.d(”Accelerometer", “X: ” + event.values[0]);
    Log.d(”Accelerometer", “Y: ” + event.values[1]);
    Log.d(”Accelerometer", “Z: ” + event.values[2]);
    return true;
}
```

References

- Lecture Notes of FIT3027/FIT4039: Sensors and Location Awareness
- Sensors Overview: http://developer.android.com/guide/topics/sensors/sensors_overview.html

Let's review the Assignments.. again!

- Group Assignment 1 and 2:
  - Compare to a classic 'V' project lifecycle model - **with feedback loops!**
  - Everyone must help to develop the App! (Author's names in comments)



- Individual Research Presentations (Assignment 3)
  - Prepare individual presentation slides on your part of contribution in Phase I & II assignments. Do an individual oral presentation (5 minutes per person).