Android Hardware Sensors

android.hardware.Sensor

Dan Lapp, Matt Hills

android.hardware.Sensor

- Versatile library for detecting motion, environmental changes, position
- Several possible types of hardware sensors
- Available sensors are device dependent

Types of Sensors

- Motion Sensors
  - Measure acceleration and rotational forces along 3 axis
  - ex Accelerometer, Gyroscope, Rotation
- Environmental Sensors
  - Measure air temperature, air pressure, illumination, humidity
  - Barometers, photometers, thermometers
- Position Sensors
  - Measures physical position of device
  - Orientation sensor, magnetometer
- Not all devices have all sensors
  - ex Most mobile devices have an accelerometer and magnetometer, but few have barometers or thermometers

Detecting Sensors

- Always verify sensors before using them
  - If Your app relies on a specific sensor, add it to the manifest

```
<uses-feature android:name="android.hardware.sensor.accelerometer" android:required="true" />
```

  - If your app uses a sensor, but can run without it

```
private SensorManager mSensorManager;
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE) != null){
        // Success! There's a pressure sensor.
}
else {
        // Failure! No pressure sensor.
}
```

Sensor Availability by Platform

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a[1]	n/a[1]
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes[2]	Yes[2]	Yes[2]	Yes
TYPE_PRESSURE	Yes	Yes	n/a[1]	n/a[1]
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes[2]	Yes	Yes	Yes

Motion Sensors

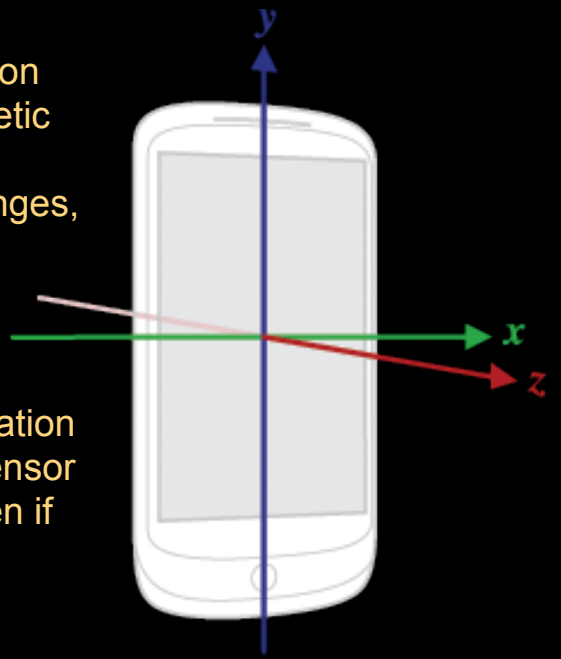
- Used to monitor tilt, shake, rotation or swing
- Can be used for user input or the reflection of the physical environment
- Each motion sensor returns an array of sensor values for each `SensorEvent`
  - `SensorEvent.values[0]` is the value along the x axis
  - `SensorEvent.values[1]` is the value along the y axis
  - `SensorEvent.values[2]` is the value along the z axis
- These 3 values represent the forces exerted across each individual axis

Types of Motion Sensors

- Accelerometer
  - Acceleration force including gravity in m/s^2
  - Used for motion detection (shake, tilt, etc.)
- Gravity
  - Force of gravity in m/s^2
  - Used for motion detection such as shake
- Gyroscope
  - Rate of rotation in rad/s
  - Used for rotation detection (spin, turn, etc.)
- Linear Acceleration
  - Acceleration force excluding gravity in m/s^2
  - Used for monitoring acceleration along a single axis
- Rotation Vector
  - Rotation vector component along each axis
  - Used for detecting current rotation of device

Coordinate System (CS)

- Uses standard 3-axis coordinate system
- The CS is defined relative to the device's default screen orientation
- Used by acceleration, gravity, gyroscope, linear accel, geomagnetic field sensors
- Axes are not swapped when the device's screen orientation changes, the CS never changes as the devices moves (same as OpenGL)
- Apps must not assume that the natural orientation is portrait, as some are landscape
- Application matches sensor data to the on-screen display
  - Use the `getRotation()` method to determine screen rotation
  - Use the `remapCoordinateSystem()` method to map sensor coordinates to screen coordinates. You need to do this even if your manifest specifies portrait-only display.



Sensor Object

- instance is initialized with a constant type, which represents the type of sensor
- `getMaximumRange()`,the maximum range of the sensor
- `getMinDelay()`, minimum delay allowed between 2 events in microsecond or 0 if the sensor only returns a value when the data is measuring changes
- `getName()`, returns the name of the sensor
- `getPower()`, returns the power in mA used by the sensor
- `getResolution()`, resolution of the sensor in sensor's unit of measure
- `getType()`, generic type of the sensor
- `getVendor()`, the sensor's manufacturer
- `getVersion()`, version of the sensor's module

Sensor Manager

- Lets you access the device's sensors
- Get the instance of this class by calling

```
SensorManager mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
```

- Enable or disable sensors in sensor manager
  - Make sure to disable sensors when you are not using them
  - Android does not automatically disable sensors even when the screen turns off

```
mSensorManager.cancelTriggerSensor(TriggerEventListener listener, Sensor sensor)
```

SensorEvent

An event for `Sensor` that contains the following properties

- `int accuracy`: the accuracy of the event
  - if accuracy is off, recalibration with the environment may be required (see SensorManager accuracy constants)
- `Sensor sensor`: the sensor that generated the event
- `long timestamp`: time the event occurred in ns
- `float values[]`: the data pertaining to the sensor
  - The values depend on the sensor used
  - For motion, it is usually the value for the x, y and z axis

SensorEventListener

- Receives notifications from SensorManager when sensor values change
- This is an interface that will be implemented by an `Activity` using sensors
- There are two abstract methods that can be implemented
  - `void onAccuracyChanged(Sensor sensor, int accuracy)`
  - `void onSensorChanged(SensorEvent event)`

- Don’t save the instance of event past the lifecycle of these methods, it may be reused and can change

```
class SensorActivity extends Activity implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        //enable sensors every time activity resumes
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onPause() {
        //disable sensors when not using them
        super.onPause();
        mSensorManager.unregisterListener(this);
    }
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        //do something when sensor accuracy changes
        //int accuracy is the new accuracy of the sensor
    }
    public void onSensorChanged(SensorEvent event) {
        //do something when sensor changes
    }
}
```

Accelerometer vs Gyroscope

- Accelerometer
  - measures acceleration relative to free-falling frame of reference
  - acceleration force of x,y,z axis
- Gyroscope
  - used to measure or maintain orientation of a device
  - rate of rotation around x,y,z axis

Accelerometer vs Gyroscope Cont

- From our testing, accelerometer gives a constant reading of the tilting of the phone, the more you tilt it the greater the value, while the gyroscope measures the instant movement value of the phone. So when you tilt it with force the value will be bigger
- Slowly moving the device with the accelerometer will increase the values, slowly moving with the gyroscope will have steady values

Best Practices

- Unregister sensor listeners when activity pauses
- Don’t test on emulator as they cannot emulate sensors
- Don’t block `onSensorChanged()`, try to do as little work as possible in this method as it gets called often
- Avoid deprecated methods and sensor types
- Verify the sensors exist on the device before you use them, device manufacturers are not required to provide any sensors
- Choose sensor delays carefully. Allowing the system to send extra data that isn’t needed wastes resources and battery