CS475/575

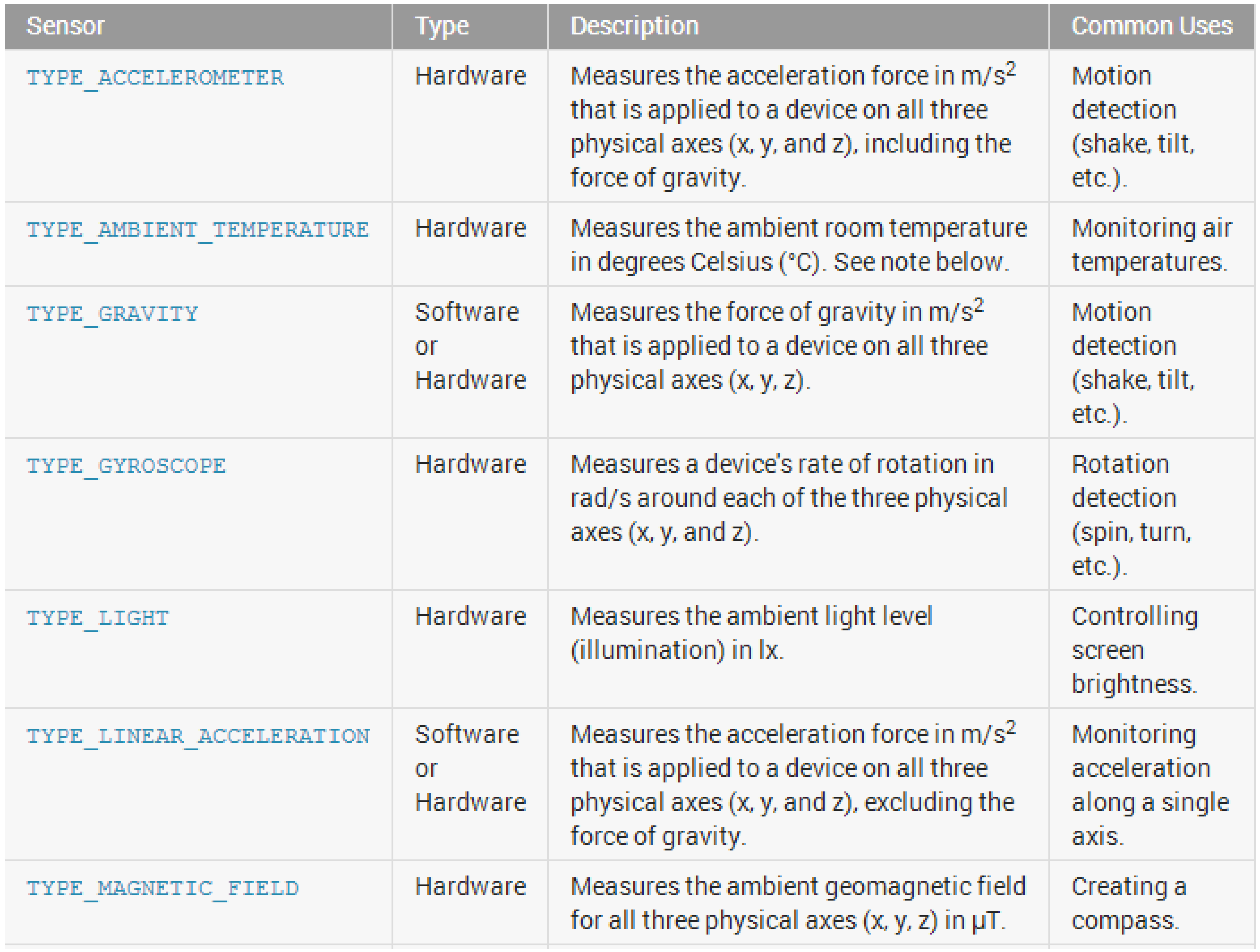
Sensing and Sensors

Content adapted from
http://www.cs.utexas.edu/~scottm/cs378/schedule.htm
Used with permission

Sensors

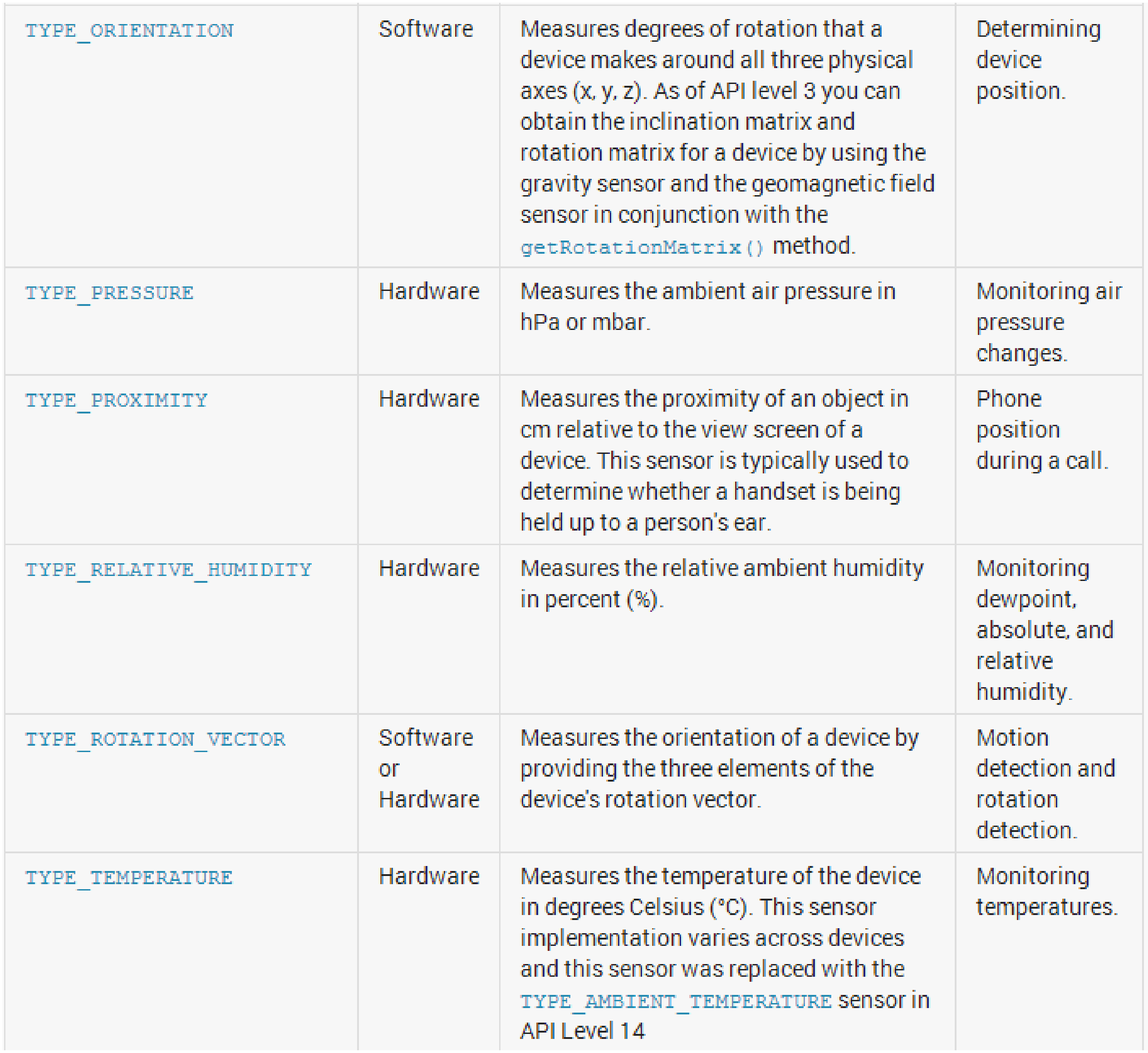
- Deliver raw data to applications. Measure and monitor
  - motion
  - orientation (aka position)
  - environmental conditions

Kinds



Sensor	Type	Description	Common Uses
`TYPE_ACCELEROMETER`	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
`TYPE_AMBIENT_TEMPERATURE`	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
`TYPE_GRAVITY`	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
`TYPE_GYROSCOPE`	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
`TYPE_LIGHT`	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
`TYPE_LINEAR_ACCELERATION`	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
`TYPE_MAGNETIC_FIELD`	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.

Kinds



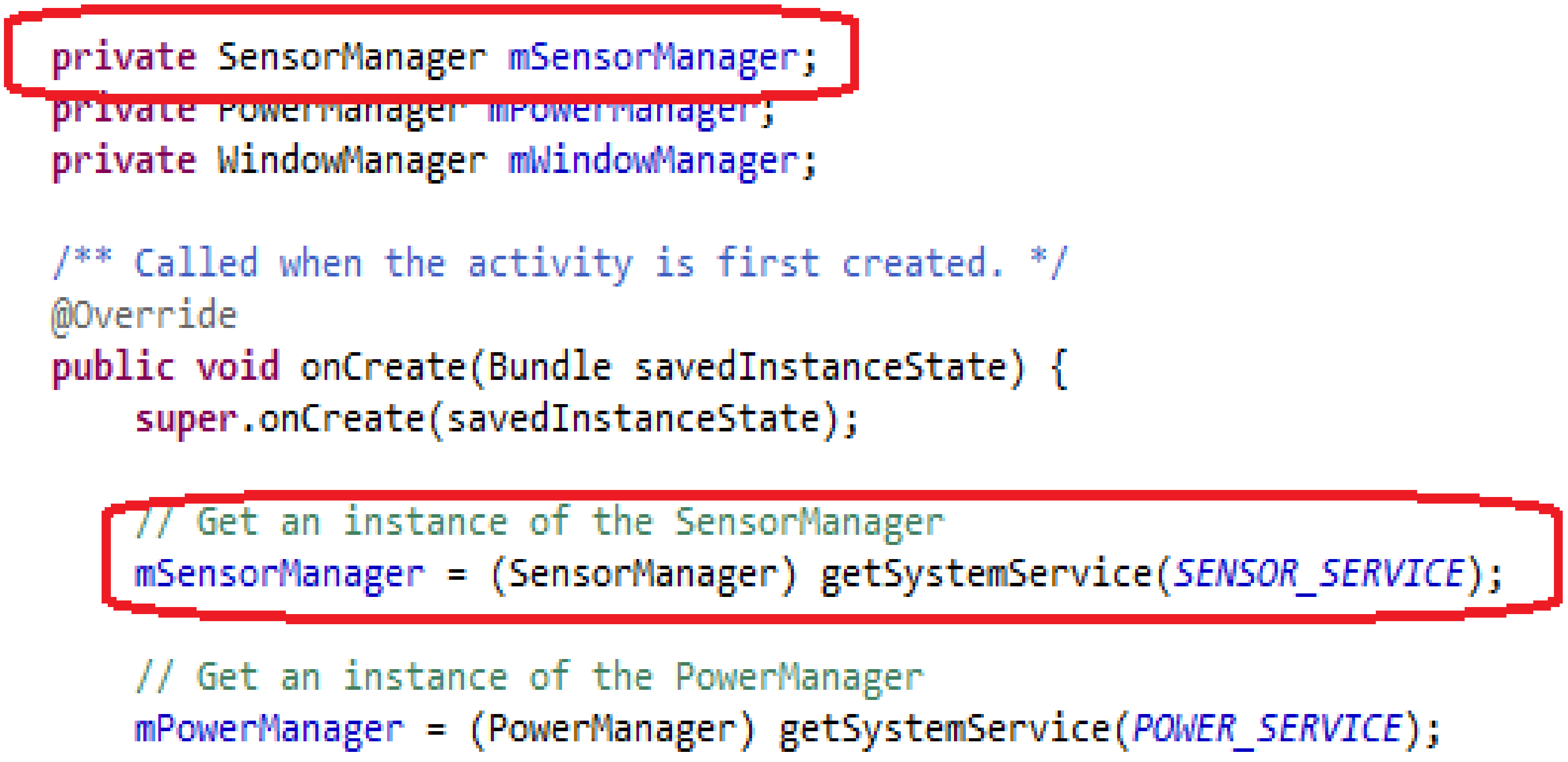
TYPE_ORIENTATION	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the `getRotationMatrix()` method.	Determining device position.
TYPE_PRESSURE	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with the `TYPE_AMBIENT_TEMPERATURE` sensor in API Level 14	Monitoring temperatures.

Enumerating Sensors

- Obtain the *SensorManager* object
- Enumerate sensors via getSensorList(...)

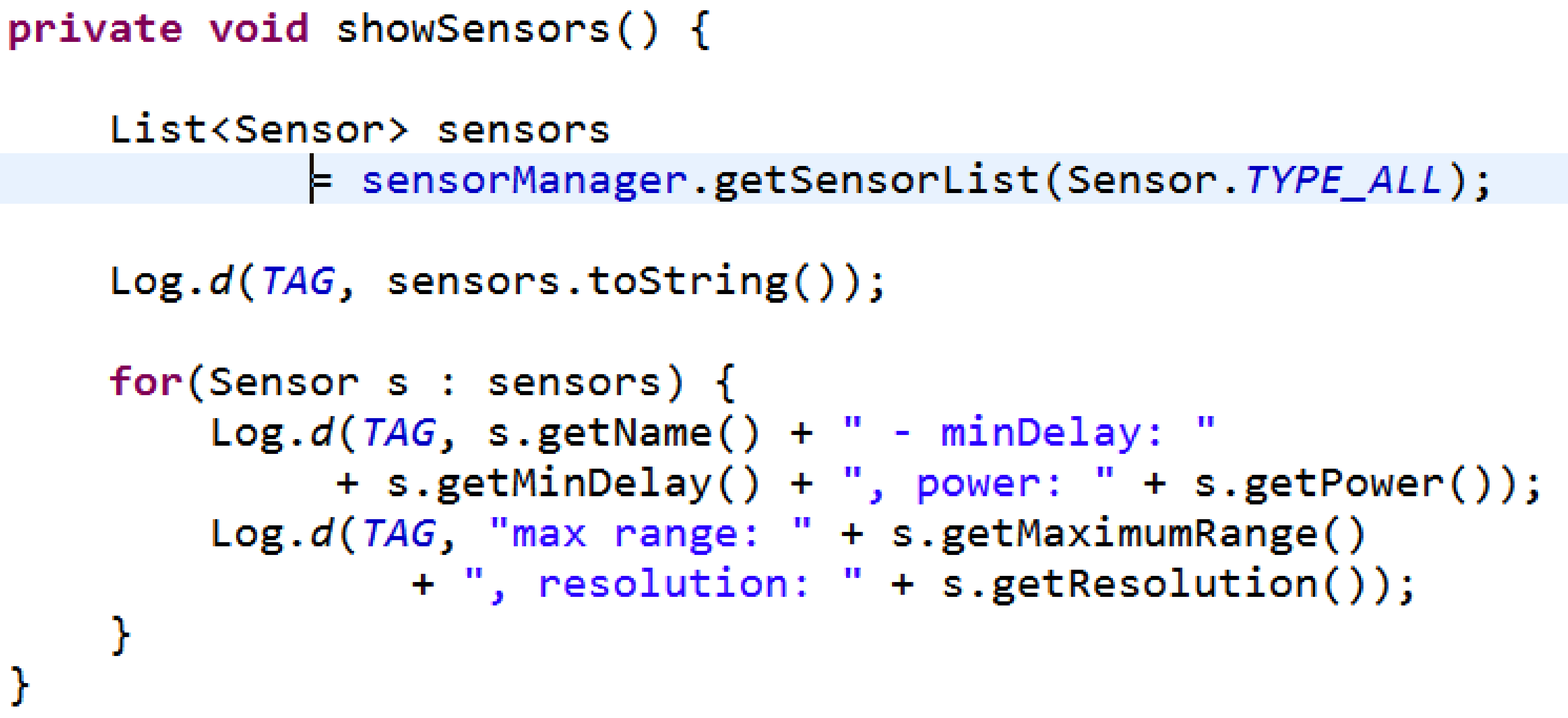
SensorManager

- Use SensorManager



```
private SensorManager mSensorManager;
private PowerManager mPowerManager;
private WindowManager mWindowManager;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get an instance of the SensorManager
    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

    // Get an instance of the PowerManager
    mPowerManager = (PowerManager) getSystemService(POWER_SERVICE);
```

Listing Sensors on a Device



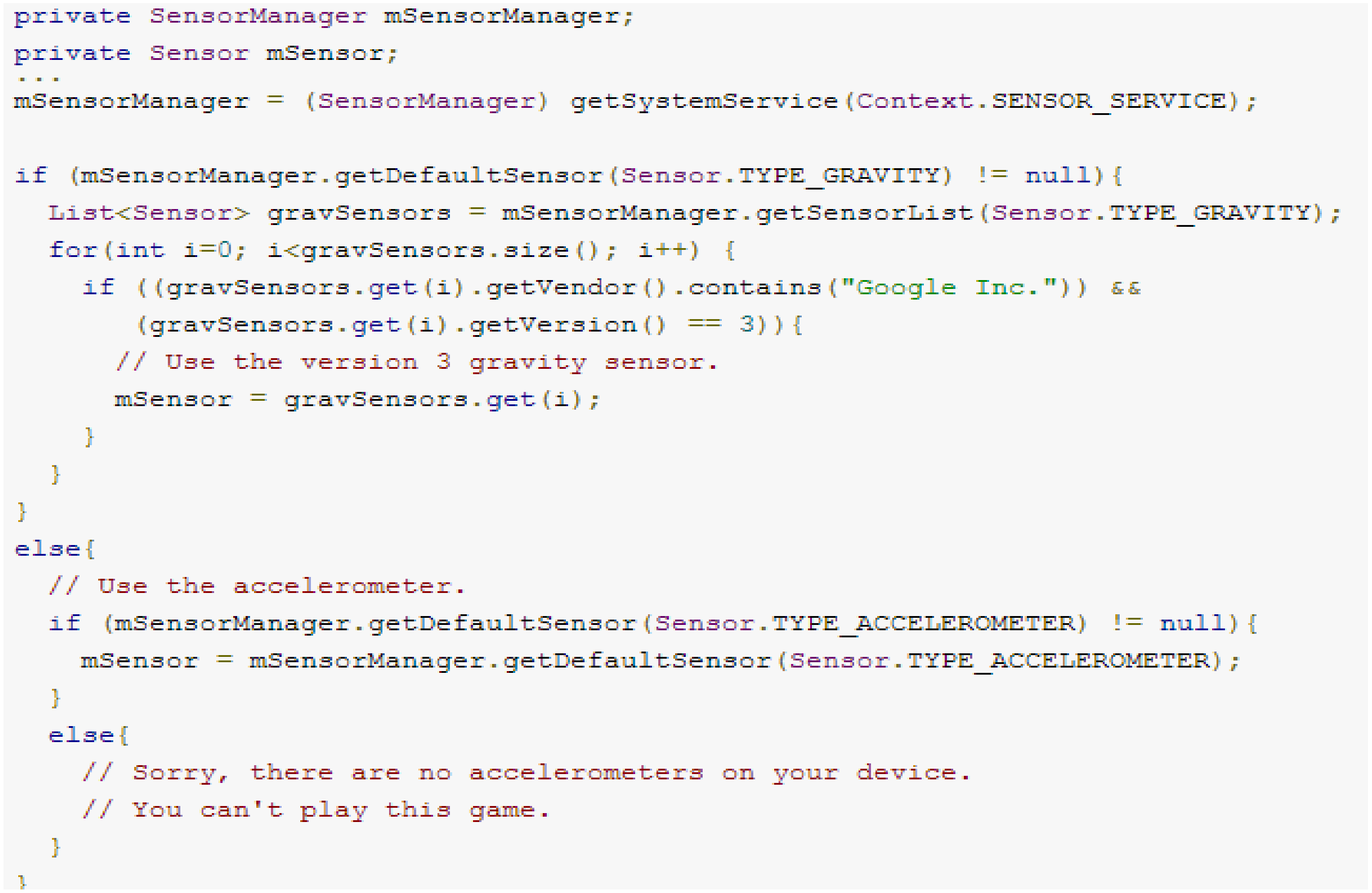
```java
private void showSensors() {

    List<Sensor> sensors
            = sensorManager.getSensorList(Sensor.TYPE_ALL);

    Log.d(TAG, sensors.toString());

    for(Sensor s : sensors) {
        Log.d(TAG, s.getName() + " - minDelay: "
             + s.getMinDelay() + ", power: " + s.getPower());
        Log.d(TAG, "max range: " + s.getMaximumRange()
                + ", resolution: " + s.getResolution());
    }
}
```

See GetSensorList application

Sensor Capabilities

- Various methods in Sensor class to get capabilities of Sensor
- minDelay (in microseconds) between 2 events
- power consumption in mA (milliAmps)
- maxRange (of return values)
- getVendor ()
- getVersion ()

Choosing Specific Sensors



```java
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

if (mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null){
  List<Sensor> gravSensors = mSensorManager.getSensorList(Sensor.TYPE_GRAVITY);
  for(int i=0; i<gravSensors.size(); i++) {
    if ((gravSensors.get(i).getVendor().contains("Google Inc.")) &&
       (gravSensors.get(i).getVersion() == 3)){
      // Use the version 3 gravity sensor.
      mSensor = gravSensors.get(i);
    }
  }
}
else{
  // Use the accelerometer.
  if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null){
    mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
  }
  else{
    // Sorry, there are no accelerometers on your device.
    // You can't play this game.
  }
}
```

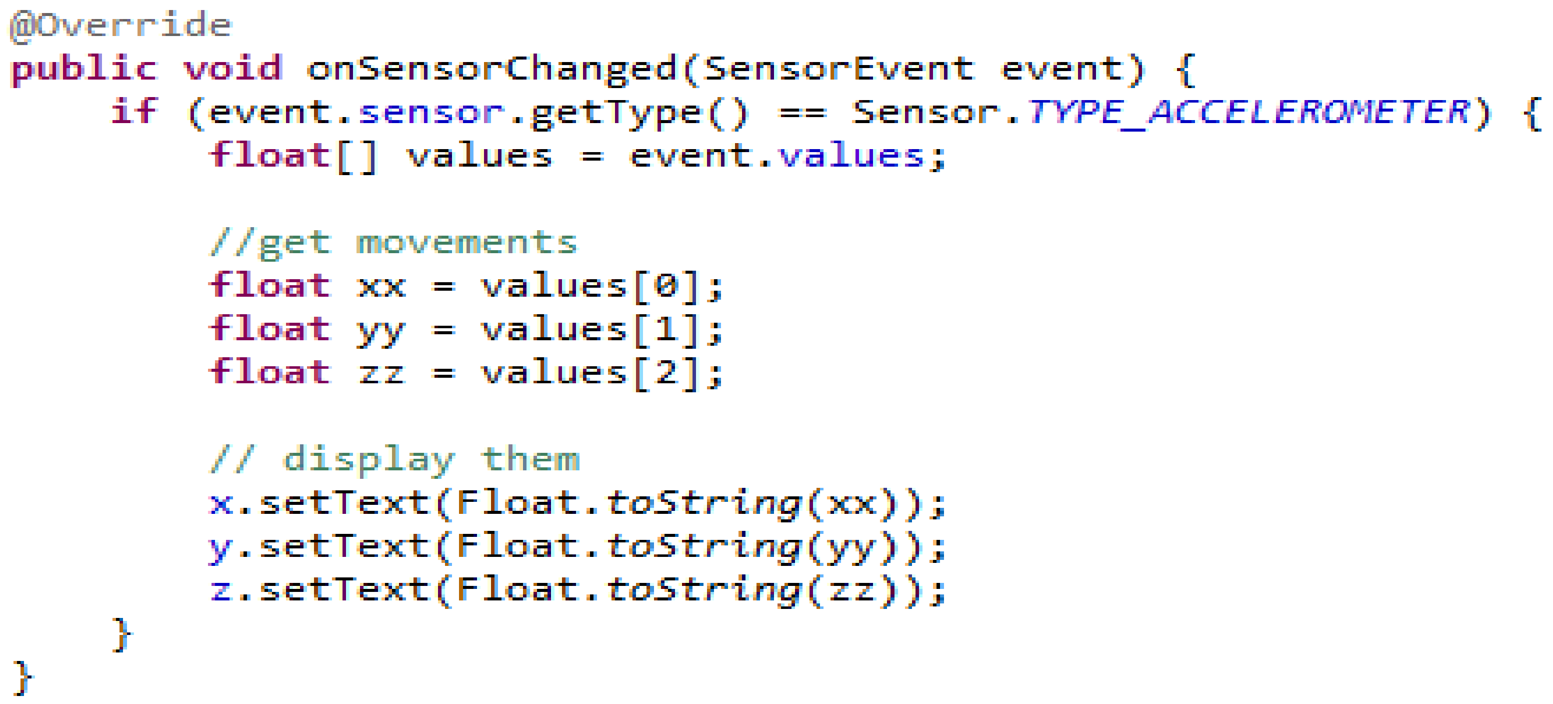
Using Sensors

- Obtain the *SensorManager* object
- create a *SensorEventListener* for *SensorEvents*
  - logic that responds to sensor event
  - various amounts of data from sensor depending on type of sensor
- Register the listener (with a particular sensor) in onResume()
- UnRegister the listener in onPause()

SensorEventListener

- Interface with two methods:
  - void onAccuracyChanged (Sensor sensor, int accuracy)
  - void onSensorChanged (SensorEvent event)
    - Sensor values have changed
    - this is the key method to override
  - don't hold onto the event
    - part of pool and the values may be altered soon

OnSensorChanged

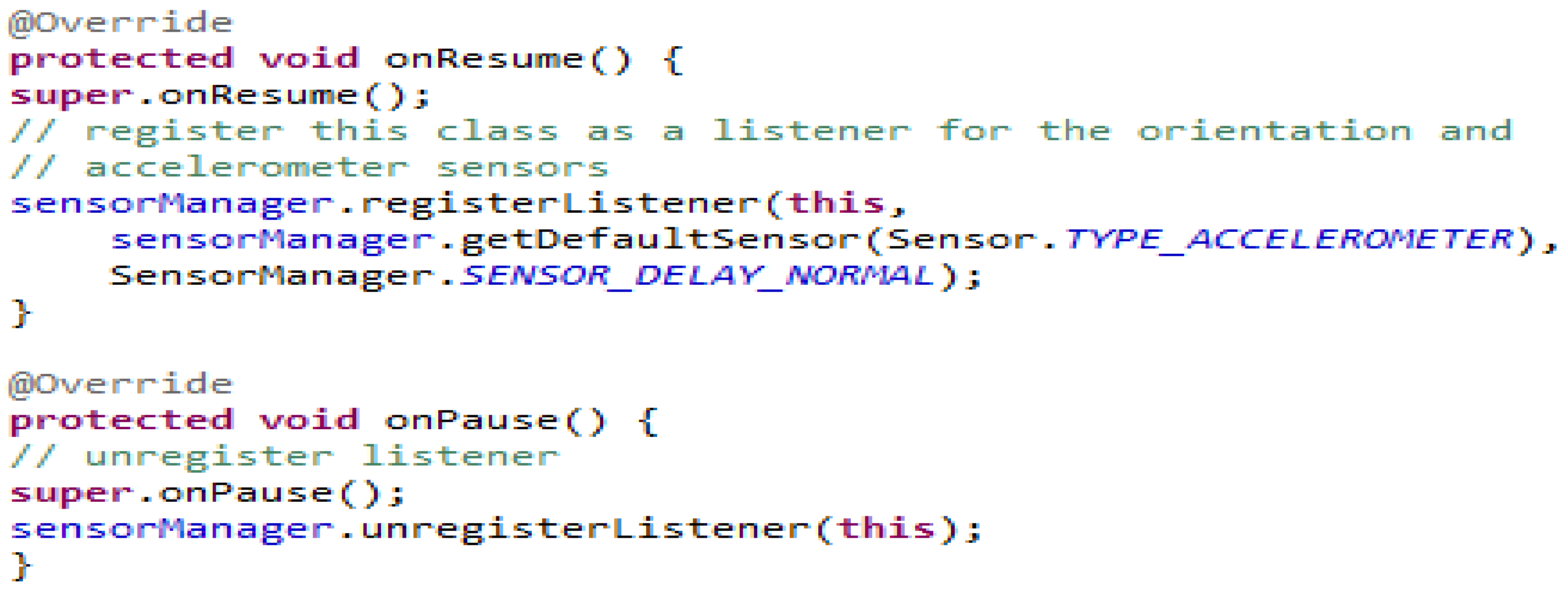


```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float[] values = event.values;

        //get movements
        float xx = values[0];
        float yy = values[1];
        float zz = values[2];

        // display them
        x.setText(Float.toString(xx));
        y.setText(Float.toString(yy));
        z.setText(Float.toString(zz));
    }
}
```

- Lots of events here, so do not block method

(Un)Registering a listener

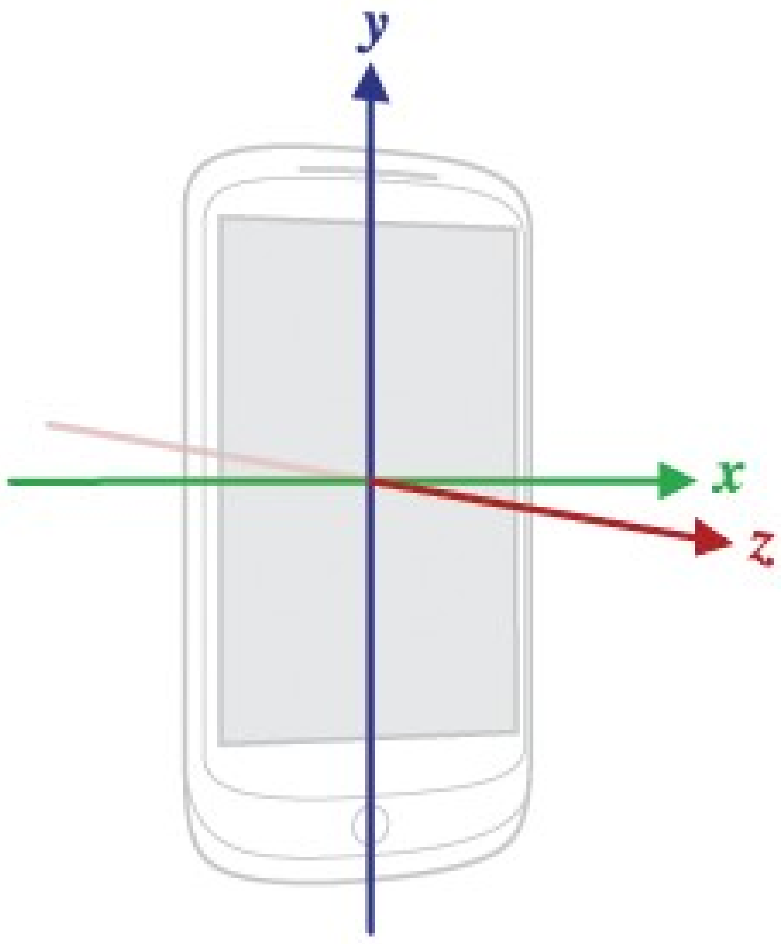
- IMPORTANT
  - Register in onResume()
  - Release in onPause()



```
@Override
protected void onResume() {
super.onResume();
// register this class as a listener for the orientation and
// accelerometer sensors
sensorManager.registerListener(this,
    sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
    SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
// unregister listener
super.onPause();
sensorManager.unregisterListener(this);
}
```

Sensor Coordinate System

- For most motion sensors
- +x to the right
- +y up
- +z out of front face
- relative to device



AccelerationTest
Simple Sensor Example

- App that demos linear acceleration

Odds and Ends

- Register/Unregister sensor listeners
- Harder to use emulator
- Don’t block onSensorChanged()
- Don’t use deprecated sensor methods or types
- Verify that sensors are there before using

- http://www.vogella.com/articles/AndroidSensor/article.html
- http://developer.android.com/guide/topics/sensors/sensors_overview.html