Android Sensors

by
Stefan Varga, Michal Kostic

touchqode.com

jsug.at
University of Technology Vienna
October 25th 2010
Why sensors?
Applications

• Resizing screen / tilt
• Environment adjustment of apps, user comfort
  — Adjustment in cinema, prediction of movement
• Gaming
• AR
• AR Gaming
• AR Navigation
• Bar codes
• Geo – tagging, grafitti, recomendations..
• Network of objects, locations and people, 3D social
• Giant distributed sensor system
  — Noise mapping
• .. And anything you can imagine... 😊
Presentation Outline

1. Introduction + API
2. Simple sensors
3. Position
4. Camera
5. About us – touchqode.com
Android 3rd in sales

<table>
<thead>
<tr>
<th>Operating System</th>
<th>2Q10 Units</th>
<th>2Q10 Market Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbian</td>
<td>25,386.8</td>
<td>41.2</td>
</tr>
<tr>
<td>RIM BlackBerry</td>
<td>11,228.8</td>
<td>18.2</td>
</tr>
<tr>
<td>Android</td>
<td>10,606.1</td>
<td>17.2</td>
</tr>
<tr>
<td>iOS</td>
<td>8,743.0</td>
<td>14.2</td>
</tr>
<tr>
<td>Microsoft Windows Mobile</td>
<td>3,096.4</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Source: [http://www.gartner.com/it/page.jsp?id=1421013](http://www.gartner.com/it/page.jsp?id=1421013)
# Overview of Android phones

<table>
<thead>
<tr>
<th></th>
<th>Acceler.</th>
<th>Magnetic</th>
<th>Gyroscope</th>
<th>Light</th>
<th>Pressure</th>
<th>Proximity</th>
<th>Temperature</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus One</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>HTC Incredible</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>HTC Desire</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>HTC Evo</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Motorola Droid</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Samsung Galaxy S</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Garminfone</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Hero</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>HTC Droid Eris</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Motorola CHARM</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Motorola DROID™ 2</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Samsung Epic</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Samsung Captivate</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Sony Ericsson Xperia X10</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Backflip</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From web sources - might not be complete, plus some brands have several versions of their phones with different hw setups!
API (I.)

• Package: android.hardware

• Classes:
  – SensorManager – android service
  – Sensor – specific sensor
  – SensorEvent – specific event of the sensor = data
public class MainActivity extends Activity implements SensorEventListener {
    ..
    private SensorManager sm = null;
    ..
    public void onCreate(Bundle savedInstanceState) {
        ..
        sm = (SensorManager) getSystemService(SENSOR_SERVICE);
    }
    protected void onResume() {
        ..
        List<Sensor> typedSensors = sm.getSensorList(Sensor.TYPE_LIGHT);
        // also: TYPE_ALL
        if (typedSensors == null || typedSensors.size() <= 0) ... error...
        sm.registerListener(this, typedSensors.get(0),
               SensorManager.SENSOR_DELAY_GAME);
        // Rates: SENSOR_DELAY_FASTEST, SENSOR_DELAY_GAME,
        // SENSOR_DELAY_NORMAL, SENSOR_DELAY_UI
    }
}
public class MainActivity extends Activity implements SensorEventListener {

    ..

    private float currentValue;
    private long lastUpdate;

    ..

    public void onSensorChanged(SensorEvent event) {
        currentValue = event.values[0];
        lastUpdate = event.timestamp;
    }

    ..

}

It is recommended not to update UI directly!
public class MainActivity extends Activity implements SensorEventListener {

    ... 
    protected void onPause() {
        ...
        sm.unregisterListener(this);
    }
    ...
    protected void onStop() {
        ...
        sm.unregisterListener(this);
    }
    ..
}
Light sensor

• Sensor.TYPE_LIGHT
• values[0] = ambient light level in SI lux units
• SensorManager’s constants
  – LIGHT_CLOUDY: 100
  – LIGHT_FULLMOON: 0.25
  – LIGHT_NO_MOON: 0.001
  – LIGHT_OVERCAST: 10000.0 (cloudy)
  – LIGHT_SHADE: 20000.0
  – LIGHT_SUNLIGHT: 110000.0
  – LIGHT_SUNLIGHT_MAX: 120000.0
  – LIGHT_SUNRISE: 400.0
Proximity sensor

- Sensor.TYPE_PROXIMITY
- values[0]: Proximity sensor distance measured in centimeters (sometimes binary near-far)
Temperature sensor

• Sensor.TYPE_TEMPERATURE
• values[0] = temperature
Pressure sensor

- Sensor.TYPE_PRESSURE
- values[0] = pressure
- no constants
Position sensors

z - pointing to the sky
Magnetic sensor

- Sensor.TYPE_MAGNETIC_FIELD
- values[3] = in micro-Tesla (uT), magnetic field in the X, Y and Z axis
- SensorManager’s constants
  - MAGNETIC_FIELD_EARTH_MAX: 60.0
  - MAGNETIC_FIELD_EARTH_MIN: 30.0
Accelerometer sensor

- TYPE_ACCELEROMETER
- Values[3] = m/s^2, measure the acceleration applied to the phone minus the force of gravity (x, y, z)
- GRAVITY_EARTH, GRAVITY_JUPITER, GRAVITY_MARS, GRAVITY_MERCURY, GRAVITY_MOON, GRAVITY_NEPTUNE
Orientation sensor

• TYPE_ORIENTATION
• Deprecated
  – (use getOrientation (float[] R, float[] result))
• Values[3] – (Azimuth, Pitch, Roll) – angles 0-360
  – azimuth, rotation around the Z axis
  – pitch, rotation around the X axis
  – roll, rotation around the Y axis
• Different from plane yaw, pitch, roll
  (different axes and clockwise-ness)
Gyroscope sensor

- TYPE_GYROSCOPE
- Measure the orientation of a device
- Detect all rotations, but only few phones have it
- Values[] – iPhone gives radians/sec., and makes it possible to get the rotation matrix
Accelerometer vs. Gyroscope

• Accelerometer
  – senses linear movement, but worse rotations, good for tilt detection,
  – Does not know difference between gravity and linear movement, shaking, jitter can be filtered out, but the delay is added

• Gyroscope
  – measure all types of rotation
  – not movement
  – does not amplify hand jitter

• A+G = both rotation and movement tracking possible
How to use the data – the maths

• `SensorManager.getRotationMatrix(matrixR, matrixI, matrixAccelerometer, matrixMagnetic);`

• `matrixR` – rotation matrix $R$
  − device coordinates -> world's coordinates
  − $R^t = R^{-1}$

• `matrixI` - inclination matrix $I$
  − rotation around the X axis
  − `getInclination(I)` – computes geomagnetic inclination angle in radians
How to use the data – example

```java
float[] matrixR = new float[9];
float[] matrixI = new float[9];
SensorManager.getRotationMatrix(
    matrixR, matrixI,
    matrixAccelerometer, matrixMagnetic);
float[] lookingDir = MyMath3D.matrixMultiply(matrixR,
    new float[] {0.0f, 0.0f, -1.0f}, 3);
float[] topDir = MyMath3D.matrixMultiply(matrixR,
    new float[] {1.0f, 0.0f, 0.0f}, 3);
GLU.gluLookAt(gl,
    0.4f * lookingDir[0], 0.4f * lookingDir[1], 0.4f * lookingDir[2],
    lookingDir[0], lookingDir[1], lookingDir[2],
    topDir[0], topDir[1], topDir[2]);
```
Open GL

• The rotation matrix can be used with open GL
  – Directly load into `glLoadMatrixf(float[], int)`
  – With some computations `gluLookAt(..)`
Special cases

• Unexpected results
  – free fall
  – north pole
  – acceleration
  – other sources of magnetic field present
Accelerometer noise - simple

const float kFilteringFactor = 0.1f; // play with this value until satisfied
float accel[3]; // previous iteration

// acceleration.x,.y,.z is the input from the sensor

accel[0] = acceleration.x * kFilteringFactor + accel[0] * (1.0f - kFilteringFactor);
accel[1] = acceleration.y * kFilteringFactor + accel[1] * (1.0f - kFilteringFactor);
accel[2] = acceleration.z * kFilteringFactor + accel[2] * (1.0f - kFilteringFactor);
result.x = acceleration.x - accel[0];
result.y = acceleration.y - accel[1];
result.z = acceleration.z - accel[2];

Return result;
Accelerometer noise - notes

• If it is too slow to adapt to sudden change in position, do more rapid changes when $\text{angle(accel, acceleration)}$ is bigger

• You can throw away single values that are way out of average.

• The $|\text{acc}|$ does not have to equal $|\text{g}|$ !

• Kalaman filters – too complicated?
Calibration

• Phone laying on the table rarely gives $[0, 0, -1]$ on accelerometer
• Adding negative vectors is not the right idea
• Useful solution is the use of rotation matrix
Apps to play with

• Any compass app
  – I like the “Marine Compass”

• Sensor reading apps
  – It’s simple – make your own 😊

• Some are at androidsensors.com