touchqode

jsug.at

University of Technology Vienna

October 25th 2010

Android Sensors

by

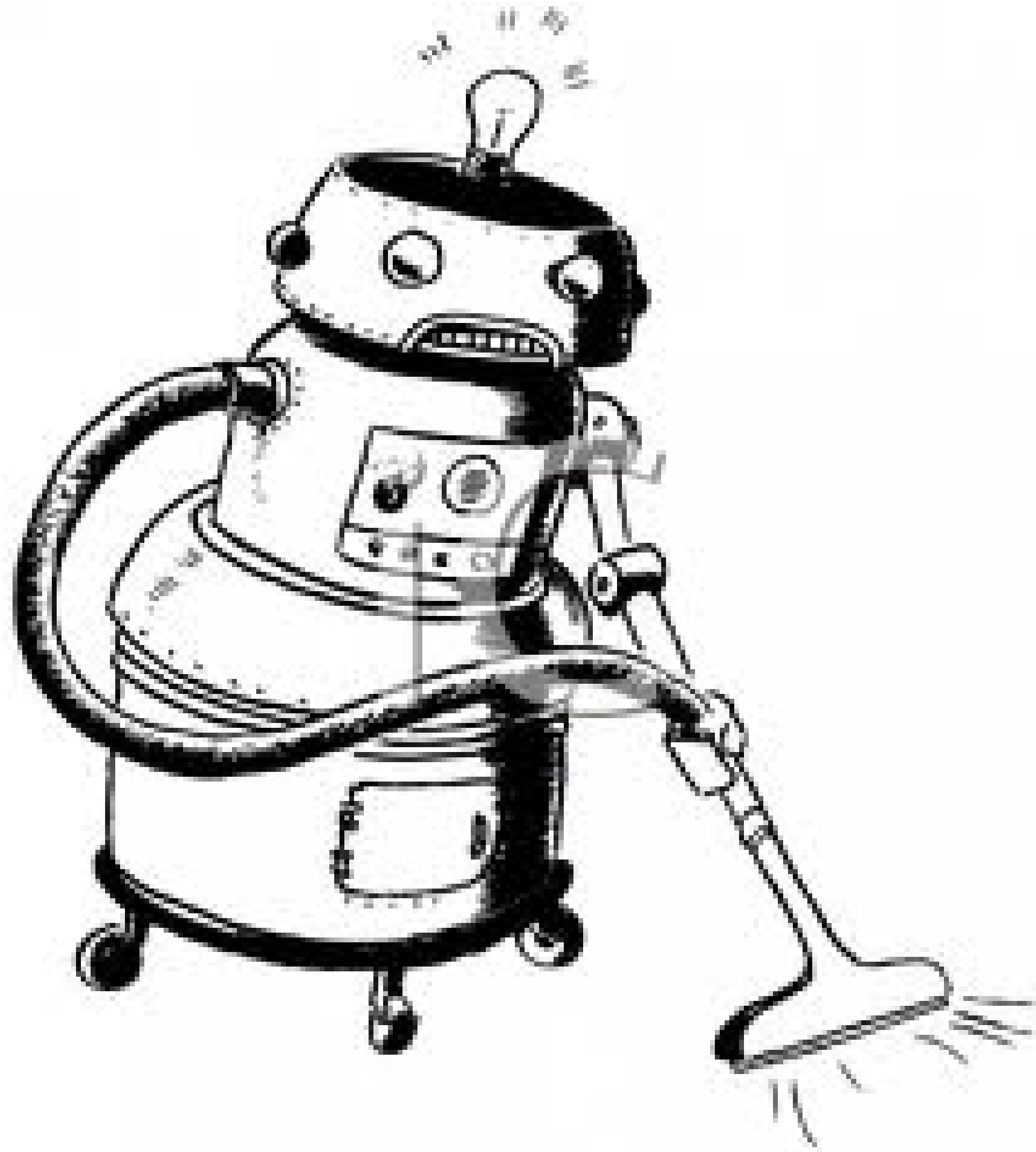
Stefan Varga, Michal Kostic

touchqode.com

touchqode

Why sensors?

touchqode



touchqode

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… ☺

touchqode

Presentation Outline

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

touchqode

Android 3rd in sales

Operating System	2Q10 Units	2Q10 Market Share (%)
Symbian	25,386.8	41.2
RIM BlackBerry	11,228.8	18.2
Android	10,606.1	17.2
iOS	8,743.0	14.2
Microsoft Windows Mobile	3,096.4	5.0

Source: http://www.gartner.com/it/page.jsp?id=1421013

touchqode

Overview of android phones

	Acceler.	Magnetic	Gyroscope	Light	Pressure	Proximity	Temperature	Camera
Nexus One	x	x		x		x		x
HTC Incredible	x	x		x		x		x
HTC Desire	x	x		x		x		x
HTC Evo	x	x		x		x		x
Motorola Droid	x	x		x		x		x
Samsung Galaxy S	x	x	?	x		x		x
Garminfone	x	x						x
HTC Hero	x	x					?	x
HTC Droid Eris	x	x		x		x		x
Motorola CHARM	x			x		x		x
Motorola DROID™ 2	x	x		x		x		x
Samsung Epic	x	x				x		x
Samsung Captivate	x	x				x		x
Sony Ericsson Xperia X10	x	x				x		
Motorola Backflip	x							x

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

API – example setup

```
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
	}
}
```

touchqode

API – example processing event

```
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!

touchqode

API – example cleanup

```
public class MainActivity extends Activity implements SensorEventListener {
	...
	protected void onPause() {
		...
		sm.unregisterListener(this);
	}
	...
	protected void onStop() {
		...
		sm.unregisterListener(this);
	}
	..
}
```

touchqode

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

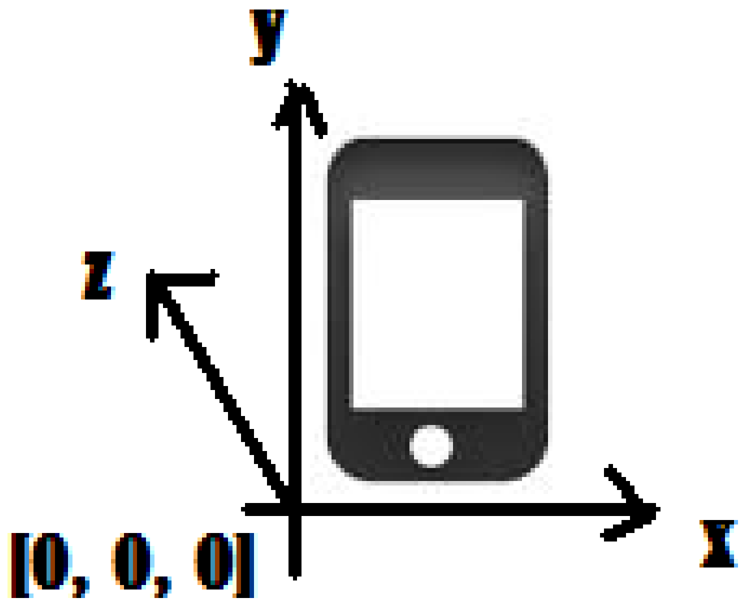
- Sensor.TYPE_TEMPERATURE
- values[0] = temperature

Pressure sensor

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

touchqode

Position sensors



z - pointing to the sky

touchqode

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

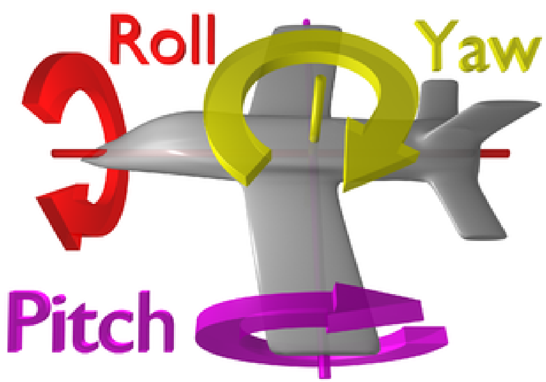
touchqode

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

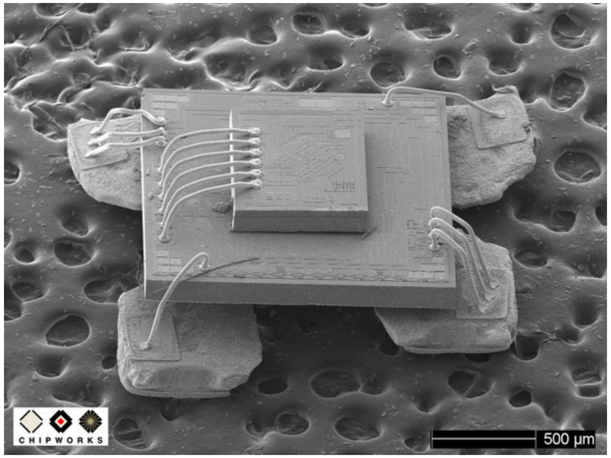
Orientation sensor

- 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



touchqode

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

touchqode

How to use the data – example

```
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]);
```

touchqode

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;
```

touchqode

Accelerometer noise - notes

- If it is too slow to adapt to sudden change in position, do more rapid changes when angle(accel, acceleration) is bigger
- You can throw away single values that are way out of average.
- The |acc| does not have to equal |g| !
- Kalaman filters – too complicated?

touchqode

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

touchqode

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