

CS378 - Mobile Computing

Sensing and Sensors Part 2

Using Sensors

- Recall basics for using a Sensor:
 - Obtain the *SensorManager* object
 - create a *SensorEventListener* for *SensorEvents*
 - logic that responds to sensor event
 - Register the sensor listener with a *Sensor* via the *SensorManager*

Sensor Best Practices

- Unregister sensor listeners
 - when done with Sensor or activity using `sensor paused (onPause method)`
 - `sensorManager.unregisterListener(sensorListener)`
 - otherwise data still sent and battery resources continue to be used

Sensor Best Practices

- Testing on the emulator
- Android SDK doesn't provide any simulated sensors
- 3rd party sensor emulator
- <http://code.google.com/p/openintents/wiki/SensorSimulator>

SensorSimulator

- Download the Sensor Simulator tool
- Start Sensor Simulator program
- Install SensorSimulator apk on the emulator
- Start app, connect simulator to emulator, start app that requires sensor data

Sensor Simulator

The screenshot shows the Sensor Simulator application window. The title bar reads "SensorSimulator". The interface is divided into several sections:

- Top Left:** Two icons, one representing a sensor and the other a terminal window.
- Top Right:** Minimize, maximize, and close window controls, along with a gear icon and a help icon.
- Control Panel:** Three radio buttons for "yaw & pitch" (selected), "roll & pitch", and "move". Below them is a small 3D visualization of a sensor.
- Sensors Panel:** A tabbed interface with "Sensors" selected. It contains:
 - Choose Device:** A dropdown menu set to "Medium".
 - Basic Orientation:** Three buttons: "accelerometer", "magnetic field", and "orientation".
 - Extended Orientation:** Four buttons: "linear acceleration", "gravity", "rotation vector", and "gyroscope".
 - Environment Sensors:** Three buttons: "temperature", "light", "proximity", and "pressure".
 - Other Sensors:** One button: "barcode reader".
- Output Panel:** A text area showing sensor data:

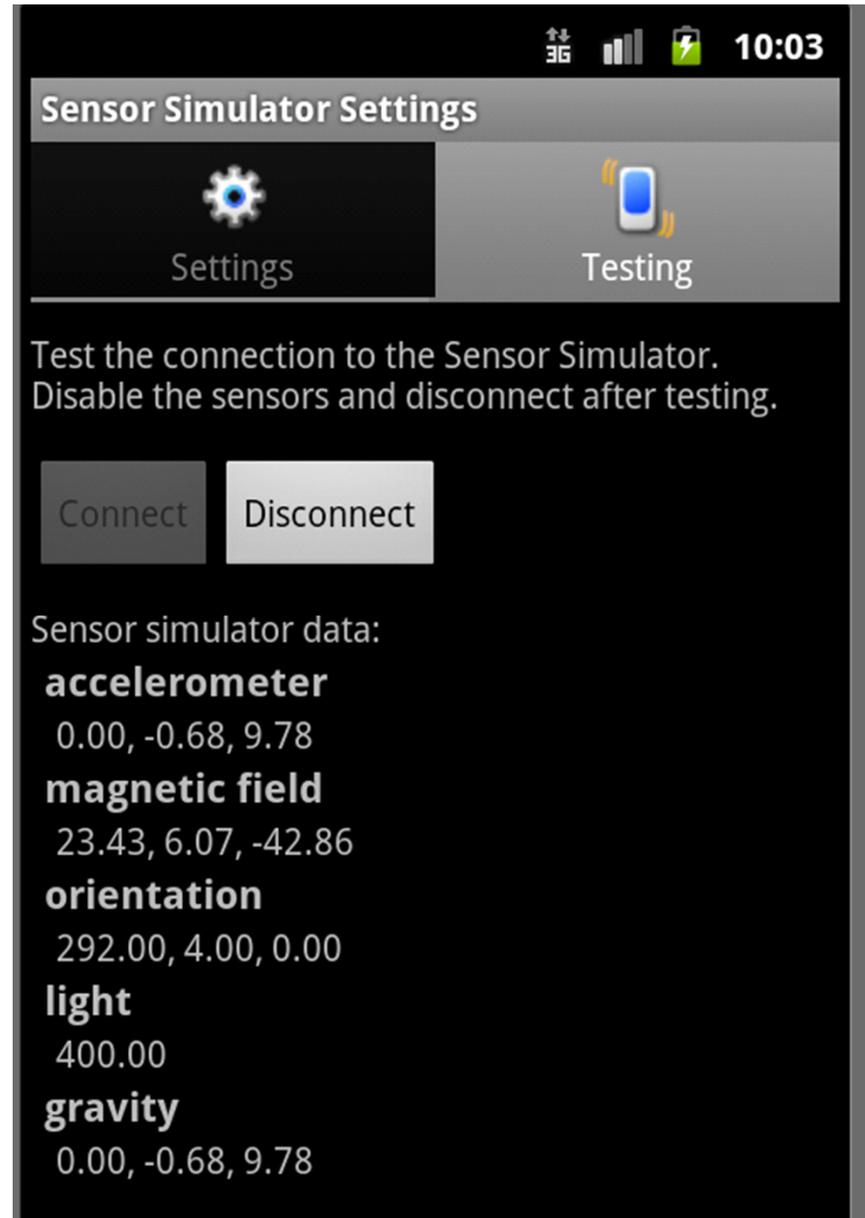
Sensor update: 13.00 ms

accelerometer: 0.00, -0.68, 9.78
magnetic field: 23.43, 6.07, -42.86
orientation: 292.00, 4.00, 0.00
light: 400.00
gravity: 0.00, -0.68, 9.78
- Terminal Panel:** A text area with instructions: "Write emulator command port and click on set to create connection. Possible IP addresses:" followed by a list of IP addresses:

10.0.2.2
128.83.141.69
192.168.1.74

Sensor Simulator

- Mouse in Sensor Simulator controls phone, feeds sensor data to emulator
- Can also record sensor data from device and play back on emulator



Sensors Best Practices

- Don't block the `onSensorChanged()` method
 - recall the resolution on sensors
 - 50 updates a second for `onSensorChange` method not uncommon
 - when registering listener update is only a hint and may be ignored
 - if necessary save event and do work in another thread or asynch task

Sensors Best Practices

- verify sensor available before using it
- use `getSensorList` method and type
- ensure list is not empty before trying to register a listener with a sensor

Sensors Best Practices

- Avoid deprecated sensors and methods
- TYPE_ORIENTATION and TYPE_TEMPERATURE are deprecated as of Ice Cream Sandwich

Sensor Best Practices

- Testing on the emulator
- Android SDK doesn't provide any simulated sensors
- 3rd party sensor emulator
- <http://code.google.com/p/openintents/wiki/SensorSimulator>

SensorSimulator

- Download the Sensor Simulator tool
- Start Sensor Simulator program
- Install SensorSimulator apk on the emulator
- Start app, connect simulator to emulator, start app that requires sensor data

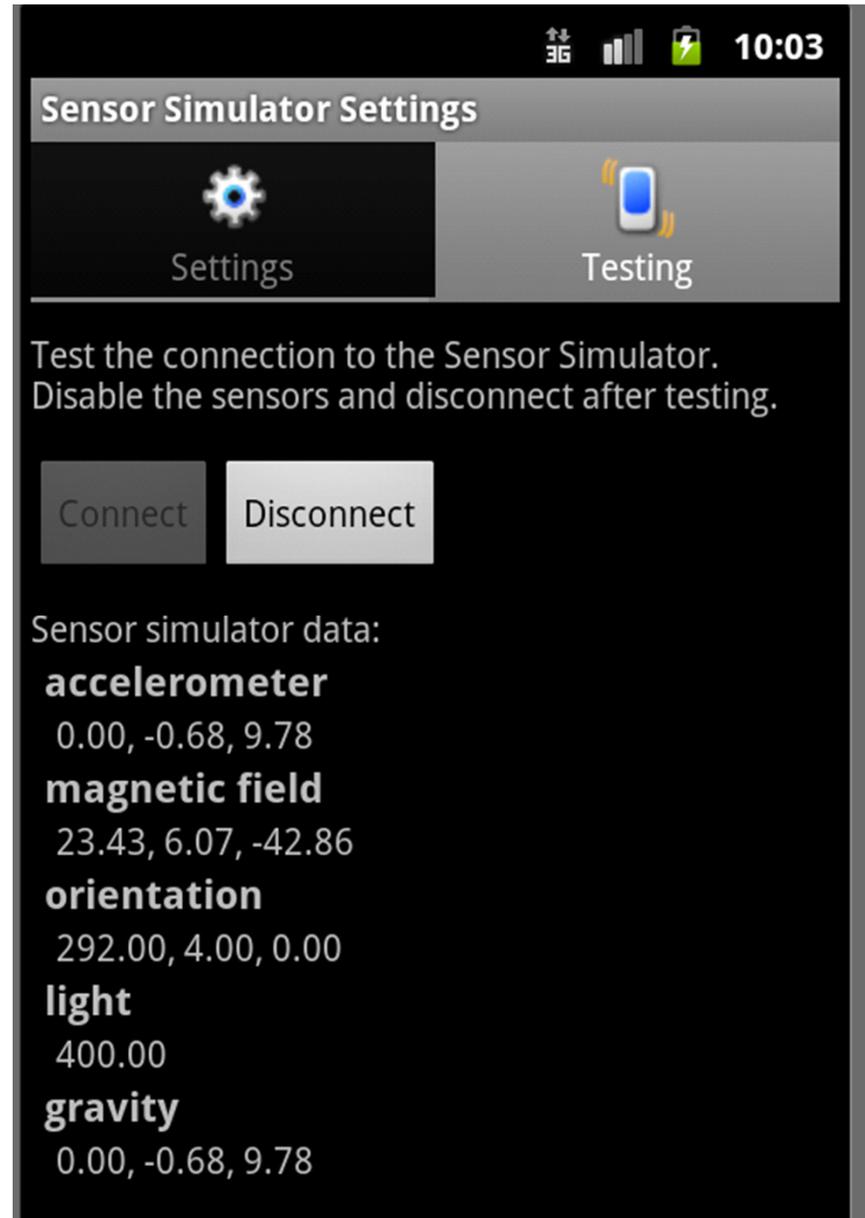
Sensor Simulator

The screenshot shows the Sensor Simulator application window. The title bar reads "SensorSimulator". The interface is divided into several sections:

- Top Left:** Two icons, one representing a sensor and the other a terminal window.
- Top Right:** A gear icon for settings and a question mark icon for help.
- Navigation Tabs:** "Sensors", "Scenario Simulator", "Quick Settings", and "Sensors Parameters".
- Left Panel:** Contains three radio buttons: "yaw & pitch" (selected), "roll & pitch", and "move". Below them is a small 3D visualization of a sensor. Further down, it shows "Sensor update: 13.00 ms" and a list of sensor data: "accelerometer: 0.00, -0.68, 9.78", "magnetic field: 23.43, 6.07, -42.86", "orientation: 292.00, 4.00, 0.00", "light: 400.00", and "gravity: 0.00, -0.68, 9.78". At the bottom, it provides instructions to "Write emulator command port and click on set to create connection. Possible IP addresses:" followed by "10.0.2.2", "128.83.141.69", and "192.168.1.74".
- Main Area:** Features a "Choose Device" dropdown menu set to "Medium". It is divided into three sensor categories:
 - Basic Orientation:** Includes buttons for "accelerometer", "magnetic field", and "orientation".
 - Extended Orientation:** Includes buttons for "linear acceleration", "gravity", "rotation vector", and "gyroscope".
 - Environment Sensors:** Includes buttons for "temperature", "light", "proximity", and "pressure".
 - Other Sensors:** Includes a button for "barcode reader".

Sensor Simulator

- Mouse in Sensor Simulator controls phone, feeds sensor data to emulator
- Can also record sensor data from device and play back on emulator



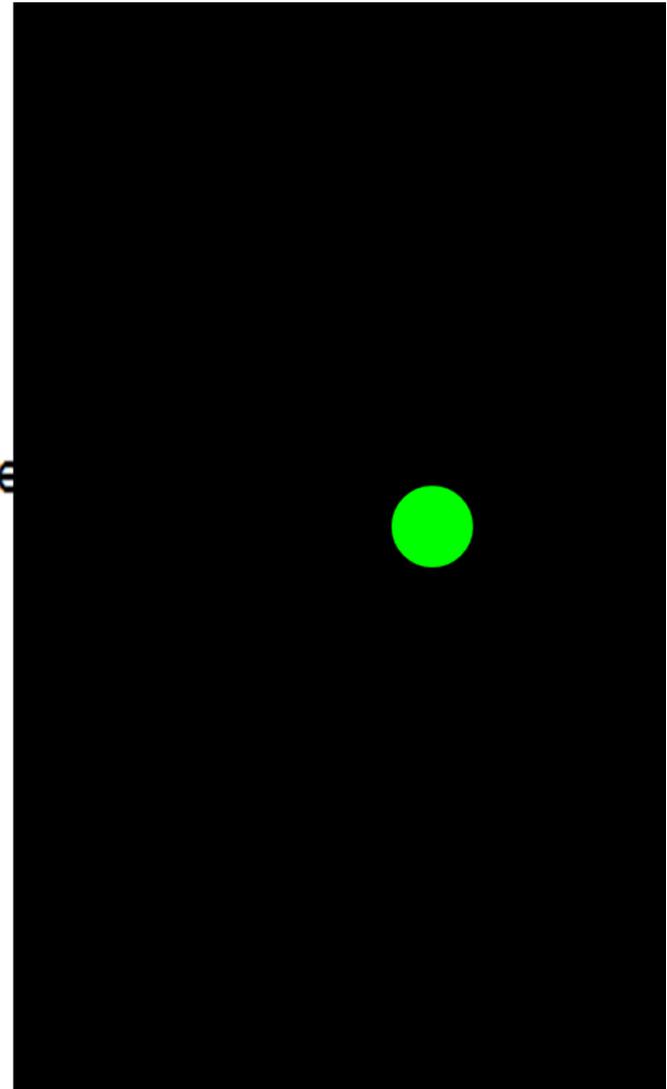
Sensor Sample - Moving Ball

- Place ball in middle of screen
- Ball has position, velocity, and acceleration
- acceleration based on linear acceleration sensor
- update over time, based on equations of motion, but fudged to suit application

Sensor Sample - Moving Ball

- Gross Simplification
- velocity set equal to acceleration

```
public void onSensorChanged(SensorEvent e) {
    //set ball speed based on phone tilt
    // speed set equal to acceleration
    mBallVelocity.x = -event.values[0];
    mBallVelocity.y = event.values[1];
}
```



Sensor Sample - Moving Ball

- Alternate Implementation

```
// try more realistic movement
float xA = -event.values[0];
float yA = event.values[1];
float aveXA = (xA + mPrevXAcc) / 2;
float aveYA = (yA + mPrevYAcc) / 2;
long currentTime = System.currentTimeMillis();
long elapsedTime = currentTime - mPrevTime;
mBallVelocity.x += aveXA * elapsedTime / 1000 / ACC_FUDGE_FACTOR;
mBallVelocity.y += aveYA * elapsedTime / 1000 / ACC_FUDGE_FACTOR;

mPrevXAcc = xA;
mPrevYAcc = yA;
mPrevTime = currentTime;
```

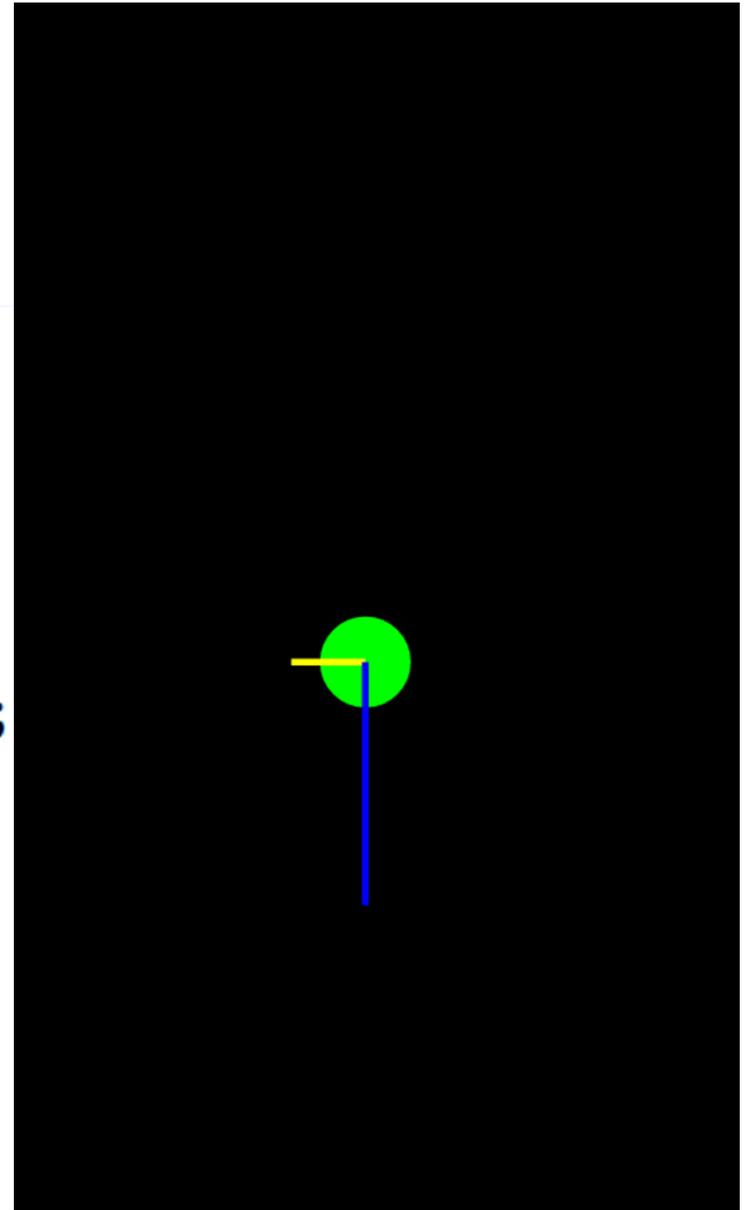
- position updated in separate thread
which redraws the view

Sensor Sample

- Draw lines for x and y velocities

```
//called by invalidate()
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    mPaint.setStrokeWidth(1);
    mPaint.setColor(0xFF00FF00);
    canvas.drawCircle(mX, mY, mR, mPaint);

    mPaint.setStrokeWidth(3);
    mPaint.setColor(0xFFFF00);
    canvas.drawLine(mX, mY,
        mX + vX * 15, mY, mPaint);
    mPaint.setColor(0xFF0000FF);
    canvas.drawLine(mX, mY,
        mX, mY + vY * 15, mPaint);
}
```



Demo Using SensorSimulator

The screenshot shows the SensorSimulator application window. At the top left, there are two icons: a blue oval with yellow lines and a black square with a white prompt character. Below these are three radio buttons: "yaw & pitch" (selected), "roll & pitch", and "move". In the center is a 3D wireframe model of a rectangular robot with a green vertical line. To the right, there are tabs for "Sensors" and "Scenario Simulator". Below the tabs, there is a "Choose Device" section with a dropdown menu set to "Medium". Underneath is a "Basic Orientation" section with three blue buttons: "accelerometer", "magnetic field", and "orientation". At the bottom left, there is a "Sensor update: 10.00 ms" label and a list of sensor data: "accelerometer: -2.53, 0.85, 9.44", "magnetic field: 30.73, -15.38, -35.25", "orientation: 255.00, -5.00, 15.00", and "light 400 00".

SensorSimulator

yaw & pitch roll & pitch move

Sensors Scenario Simulator

Choose Device

Medium

Basic Orientation

accelerometer

magnetic field

orientation

Sensor update: 10.00 ms

accelerometer: -2.53, 0.85, 9.44
magnetic field: 30.73, -15.38, -35.25
orientation: 255.00, -5.00, 15.00
light 400 00

Sensor Sample - TBBT

- Inspired by <http://tinyurl.com/7rnbg5> and <http://tinyurl.com/6nhvnnv>



TBBT Sound Effect App



Responding to Events

```
private class LinAccListener implements SensorEventListener {
    public void onSensorChanged(SensorEvent event) {
        if(event.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];
            float acc = (float)Math.sqrt( x * x + y * y);
            // Log.d("BBT", "" + acc);
            if(acc > 31) {
                Log.d("BBT", "" + acc);

                if(soundPlayer != null && !soundPlayer.isPlaying()) {
                    soundPlayer.start();
                    picture.setImageResource(R.drawable.crack);
                }
            }
        }
    }
}
```

Changing Images

- Use of an Image View
- Initial Image set in onCreate
- new image set in onSensorChange
- register listener with MediaPlayer
- on completion reset image

```
@Override  
public void onCompletion(MediaPlayer mp) {  
    picture.setImageResource(R.drawable.shake);  
}
```