CS371m - Mobile Computing

Sensing and Sensors
Sensors

• "I should have paid more attention in Physics 41"
• Most devices have built in sensors to measure and monitor
  – motion
  – orientation (aka position)
  – environmental conditions
• sensors deliver raw data to applications
Sensor Framework

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

Sensor Framework Classes

• SensorManager
  – conduit between your classes and Sensors

• Sensors
  – abstract representations of Sensors on device

• SensorEventListener
  – register with SensorManager to listen for events from a Sensor

• SensorEvent
  – data sent to listener
Recall: Android Software Stack

- **Applications**
  - Home
  - Contacts
  - Phone
  - Browser

- **Application Framework**
  - Activity Manager
  - Window Manager
  - Content Providers
  - View System
  - Package Manager
  - Telephony Manager
  - Resource Manager
  - Location Manager
  - Notification Manager

- **Libraries**
  - Surface Manager
  - Media Framework
  - OpenGL | ES
  - FreeType
  - WebKit
  - SGL
  - SSL
  - libc

- **Android Runtime**
  - Core Libraries
  - Dalvik Virtual Machine

- **Linux Kernel**
  - Display Driver
  - Camera Driver
  - Flash Memory Driver
  - Binder (IPC) Driver
  - Keypad Driver
  - WiFi Driver
  - Audio Drivers
  - Power Management

**AND SENSOR MANAGER**

**AND SENSOR DRIVERS**
TYPES OF SENSORS
Types of Sensors

• Three main classes of sensors:
  • motion (acceleration and rotational forces)
    – accelerometers, gravity sensors, gyroscopes, and rotational vector sensors
  • environmental (ambient air temperature and pressure, illumination, and humidity)
    – barometers, photometers, and thermometers.
  • position (physical position of a device)
    – orientation sensors and magnetometers
Types of Sensors

• Hardware sensors
  – built into the device

• Software sensors
  – takes data from a hardware sensors and manipulates it
  – from our perspective acts like a hardware sensor
  – aka synthetic or virtual sensors
## Types of Sensors - Dev Phone - Older

| Sensor: KR3DM 3-axis Accelerometer, STMicroelectronics |
| Sensor: AK8973 3-axis Magnetic field sensor, Asahi Kasei Microelectronics |
| Sensor: AK8973 Orientation sensor, Asahi Kasei Microelectronics |
| Sensor: GP2A Light sensor, Sharp |
| Sensor: GP2A Proximity sensor, Sharp |
| Sensor: K3G Gyroscope sensor, STMicroelectronics |
| Sensor: Gravity Sensor, Google Inc. |
| Sensor: Linear Acceleration Sensor, Google Inc. |
| Sensor: Rotation Vector Sensor, Google Inc. |

- accelerometer, linear acceleration, magnetic field, orientation, light, proximity, gyroscope, gravity
<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ACCELEROMETER</td>
<td>A constant describing an accelerometer sensor type.</td>
</tr>
<tr>
<td>TYPE_ALL</td>
<td>A constant describing all sensor types.</td>
</tr>
<tr>
<td>TYPE_AMBIENT_TEMPERATURE</td>
<td>A constant describing an ambient temperature sensor type.</td>
</tr>
<tr>
<td>TYPE_GAME_ROTATION_VECTOR</td>
<td>A constant describing an uncalibrated rotation vector sensor type.</td>
</tr>
<tr>
<td>TYPE_GEOMAGNETIC_ROTATION_VECTOR</td>
<td>A constant describing the geo-magnetic rotation vector.</td>
</tr>
<tr>
<td>TYPE_GRAVITY</td>
<td>A constant describing a gravity sensor type.</td>
</tr>
<tr>
<td>TYPE_GYROSCOPE</td>
<td>A constant describing a gyroscope sensor type.</td>
</tr>
<tr>
<td>TYPE_GYROSCOPE_UNCALIBRATED</td>
<td>A constant describing an uncalibrated gyroscope sensor type.</td>
</tr>
<tr>
<td>TYPE_LIGHT</td>
<td>A constant describing a light sensor type.</td>
</tr>
<tr>
<td>TYPE_LINEAR_ACCELERATION</td>
<td>A constant describing a linear acceleration sensor type.</td>
</tr>
<tr>
<td>TYPE_MAGNETIC_FIELD</td>
<td>A constant describing a magnetic field sensor type.</td>
</tr>
<tr>
<td>TYPE_MAGNETIC_FIELD_UNCALIBRATED</td>
<td>A constant describing an uncalibrated magnetic field sensor type.</td>
</tr>
<tr>
<td>TYPE_ORIENTATION</td>
<td>This constant was deprecated in API level 8. use SensorManager.getOrientation() instead.</td>
</tr>
<tr>
<td>TYPE_PRESSURE</td>
<td>A constant describing a pressure sensor type.</td>
</tr>
<tr>
<td>TYPE_PROXIMITY</td>
<td>A constant describing a proximity sensor type.</td>
</tr>
<tr>
<td>TYPE_RELATIVE_HUMIDITY</td>
<td>A constant describing a relative humidity sensor type.</td>
</tr>
<tr>
<td>TYPE_ROTATION_VECTOR</td>
<td>A constant describing a rotation vector sensor type.</td>
</tr>
<tr>
<td>TYPE_SIGNIFICANT_MOTION</td>
<td>A constant describing a significant motion trigger sensor.</td>
</tr>
<tr>
<td>TYPE_STEP_COUNTER</td>
<td>A constant describing a step counter sensor.</td>
</tr>
<tr>
<td>TYPE_STEP_DETECTOR</td>
<td>A constant describing a step detector sensor.</td>
</tr>
<tr>
<td>TYPE_TEMPERATURE</td>
<td>This constant was deprecated in API level 14. use Sensor.TYPE_AMBIENT_TEMPERATURE instead.</td>
</tr>
</tbody>
</table>
Sensor Capabilities - Dev Phones - Older

KR3DM 3-axis Accelerometer - minDelay: 20000, power: 0.23
max range: 19.6133, resolution: 0.019153614
AK8973 3-axis Magnetic field sensor - minDelay: 16667, power: 6.8
max range: 2000.0, resolution: 0.0625
GP2A Light sensor - minDelay: 0, power: 0.75
max range: 3626657.8, resolution: 1.0
GP2A Proximity sensor - minDelay: 0, power: 0.75
max range: 5.0, resolution: 5.0
K3G Gyroscope sensor - minDelay: 1190, power: 6.1
max range: 34.906586, resolution: 0.0012217305
Rotation Vector Sensor - minDelay: 20000, power: 13.13
max range: 1.0, resolution: 5.9604645E-8
Gravity Sensor - minDelay: 20000, power: 13.13
max range: 19.6133, resolution: 0.019153614
Linear Acceleration Sensor - minDelay: 20000, power: 13.13
max range: 19.6133, resolution: 0.019153614
Orientation Sensor - minDelay: 20000, power: 13.13
max range: 360.0, resolution: 0.00390625
Corrected Gyroscope Sensor - minDelay: 1190, power: 13.13
max range: 34.906586, resolution: 0.0012217305
| SensorTest | GP2A Light sensor Sharp |
| SensorTest | GP2A Proximity sensor Sharp |
| SensorTest | BMP180 Pressure sensor Bosch |
| SensorTest | MPL Gyroscope Invensense |
| SensorTest | MPL Accelerometer Invensense |
| SensorTest | MPL Magnetic Field Invensense |
| SensorTest | MPL Orientation Invensense |
| SensorTest | MPL Rotation Vector Invensense |
| SensorTest | MPL Linear Acceleration Invensense |
| SensorTest | MPL Gravity Invensense |
| SensorTest | Rotation Vector Sensor Google Inc. |
| SensorTest | Gravity Sensor Google Inc. |
| SensorTest | Linear Acceleration Sensor Google Inc. |
| SensorTest | Orientation Sensor Google Inc. |
| SensorTest | Corrected Gyroscope Sensor Google Inc. |
Sensor Capabilities - Dev Phone - Newer

GP2A Light sensor – minDelay: 0, power: 0.75
max range: 646239.5, resolution: 1.0
GP2A Proximity sensor – minDelay: 0, power: 0.75
max range: 5.0, resolution: 5.0
BMP180 Pressure sensor – minDelay: 20000, power: 0.67
max range: 1100.0, resolution: 0.01
MPL Gyroscope – minDelay: 10000, power: 6.1
max range: 34.90656, resolution: 0.57246757
MPL Accelerometer – minDelay: 10000, power: 0.139
max range: 19.6133, resolution: 0.038344003
MPL Magnetic Field – minDelay: 10000, power: 4.0
max range: 8001.0, resolution: 0.012
MPL Orientation – minDelay: 10000, power: 10.239
max range: 360.0, resolution: 1.0E-5
MPL Rotation Vector – minDelay: 10000, power: 10.239
max range: 1.0, resolution: 1.0E-5
MPL Linear Acceleration – minDelay: 10000, power: 0.5
max range: 10240.0, resolution: 1.0
MPL Gravity – minDelay: 10000, power: 10.239
max range: 19.6133, resolution: 0.038344003
Rotation Vector Sensor – minDelay: 10000, power: 10.239
max range: 1.0, resolution: 5.9604645E-8
Gravity Sensor – minDelay: 10000, power: 10.239
max range: 19.6133, resolution: 0.038344003
Linear Acceleration Sensor – minDelay: 10000, power: 10.239
max range: 19.6133, resolution: 0.038344003
Orientation Sensor – minDelay: 10000, power: 10.239
max range: 360.0, resolution: 0.00390625
Corrected Gyroscope Sensor – minDelay: 10000, power: 10.239
max range: 34.90656, resolution: 0.57246757
Types of Sensors

• TYPE_ACCELEROMETER
  – hardware
  – acceleration in m/s$^2$
  – x, y, z axis
  – includes gravity
Types of Sensors

• TYPE_AMBIENT_TEMPERATURE
  – hardware
  – "room" temperature in degrees Celsius
  – no such sensor on dev phones

• TYPE_GRAVITY
  – software or hardware
  – just gravity
  – if phone at rest same as
    TYPE_ACCELEROMETER
Types of Sensors

• **TYPE_GYROSCOPE**
  
  – hardware
  
  – measure device's rate of rotation in radians / second around 3 axis

• **TYPE_LIGHT**
  
  – hardware
  
  – light level in lx,
  
  – lux is SI measure illuminance in luminous flux per unit area
Types of Sensors

• **TYPE_LINEAR_ACCELERATION**
  – software or hardware
  – measure acceleration force applied to device in three axes excluding the force of gravity

• **TYPE_MAGNETC_FIELD**
  – hardware
  – ambient geomagnetic field in all three axes
  – uT micro Teslas
Types of Sensors

• TYPE_ORIENTATION [deprecated]
  – software
  – measure of degrees of rotation a device makes around all three axes

• TYPE_PRESSURE
  – hardware
  – ambient air pressure in hPa or mbar
  – force per unit area
  – 1 Pascal = 1 Newton per square meter
  – hecto Pascals (100 Pascals)
  – milli bar - 1 mbar = 1hecto Pascal
Types of Sensors

• **TYPE_PROXIMITY**
  – hardware
  – proximity of an object in cm relative to the view screen of a device
  – usually binary (see range, resolution)
  – typically used to determine if handset is being held to person's ear during a call

• **TYPE_RELATIVE_HUMIDITY**
  – ambient humidity in percent (0 to 100)
Types of Sensors

• **TYPE_ROTATION_VECTOR (ABSOLUTE)**
  – hardware or software
  – orientation of device, three elements of the device's rotation vector

• **TYPE_TEMPERATURE**
  – hardware
  – temperature of the device in degrees Celsius
## Availability of Sensors

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Android 4.0 (API Level 14)</th>
<th>Android 2.3 (API Level 9)</th>
<th>Android 2.2 (API Level 8)</th>
<th>Android 1.5 (API Level 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ACCELEROMETER</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TYPE_AMBIENT_TEMPERATURE</td>
<td>Yes</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TYPE_GRAVITY</td>
<td>Yes</td>
<td>Yes</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TYPE_GYROSCOPE</td>
<td>Yes</td>
<td>Yes</td>
<td>n/a¹</td>
<td>n/a¹</td>
</tr>
<tr>
<td>TYPE_LIGHT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TYPE_LINEAR_ACCELERATION</td>
<td>Yes</td>
<td>Yes</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TYPE_MAGNETIC_FIELD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TYPE_ORIENTATION</td>
<td>Yes²</td>
<td>Yes²</td>
<td>Yes²</td>
<td>Yes</td>
</tr>
<tr>
<td>TYPE_PRESSURE</td>
<td>Yes</td>
<td>Yes</td>
<td>n/a¹</td>
<td>n/a¹</td>
</tr>
<tr>
<td>TYPE_PROXIMITY</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TYPE_RELATIVE_HUMIDITY</td>
<td>Yes</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TYPE_ROTATION_VECTOR</td>
<td>Yes</td>
<td>Yes</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TYPE_TEMPERATURE</td>
<td>Yes²</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Sensor Capabilities

- Various methods in Sensor class to get capabilities of Sensor
- minDelay (in microseconds)
- power consumption in mA (microAmps)
- maxRange
- resolution
Triggered Sensors

- Android 4.4, API level 19, Kit-Kat added *trigger* sensors
- Trigger sensors give a single reading and then become unregistered
  - **TYPE_SIGNIFICANT_MOTION**
  - **TYPE_STEP_COUNTER**
  - **TYPE_STEP_DETECTOR**
USING SENSORS EXAMPLE
Using Sensors - Basics

• Obtain the SensorManager object
• create a SensorEventListener for SensorEvents
  – logic that responds to sensor event
  – varying amounts of data from sensor depending on type of sensor
• Register the sensor listener with a Sensor via the SensorManager
• Unregister when done
  – a good thing to do in the onPause method
Using Sensors

```java
private void createSensor() {
    sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);

    showSensors();

    sensorManager.registerListener(sensorEventListener,
        sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
        SensorManager.SENSOR_DELAY_UI);
}
```

- `registerListener(SensorEventListener, Sensor, int rate)`
- `rate` is just a *hint*
- `SENSOR_DELAY_NORMAL`, `SENSOR_DELAY_UI`, `SENSOR_DELAY_GAME`, or `SENSOR_DELAY_FASTEST`, or time in microseconds (millionths of a second)
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
Listing Sensors on a Device to Log

```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());
    }
}
```
Simple Sensor Example

- App that shows linear acceleration
- options to display current
- ... or maximum, ignoring direction
- Linear Layout
- TextViews for x, y, and z
- Buttons to switch between max or current and to reset max
Sensor Coordinate System

• For most motion sensors:
  • +x to the right
  • +y up
  • +z out of front face
  • relative to device
  • based on natural orientation of device
    – tablet -> landscape
Accelerometer - Includes Gravity

• Sensor.  
  \textit{TYPE_ACCELEROMETER}

• Device flat on table

• \(g \approx 9.81 \text{ m/s}^2\)

• Clearly some error
Sensor Coordinate System

• Hold phone straight up and down:

<table>
<thead>
<tr>
<th>Accelerometer</th>
<th>Linear Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>x Axis</td>
<td>0.345</td>
</tr>
<tr>
<td>y Axis</td>
<td>9.81</td>
</tr>
<tr>
<td>z Axis</td>
<td>1.072</td>
</tr>
</tbody>
</table>

Current  Reset Max
Sensor Coordinate System

- Hold phone on edge
Sensor Coordinate System

- Hold phone straight up and down and pull towards the floor:
Getting Sensor Data

```java
private void createSensor() {
    sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);

    showSensors();

    sensorManager.registerListener(sensorEventListener,
        sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
        SensorManager.SENSOR_DELAY_UI);
}
```

• `registerListener`
  – `sensorEventListener`
  – Sensor -> obtain via SensorManager
  – rate of updates, a hint only, or microseconds (not much effect)

• returns true if successful
private SensorEventListener sensorEventListener =
new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        // Log.d(TAG, event + "");

        // accelerationValues[0].setText("" + event.values[0]
        if (displayCurrent)
            displayCurrent(event);
        else
            displayMax(event);

        // displayCurrentRotation(event);
    }
}
Recall, max range of linear acceleration on dev phone is 19.613 + gravity = 29.423 - a baseball pitcher throwing a fastball reaches 350 m/s^2 or more (various "physics of baseball" articles)
private void displayCurrent(SensorEvent event) {
    if (!zeroingComplete)
        gatherZeroData(event);

    for (int i = 0; i < accelerationValues.length; i++) {
        float value = event.values[i];
        value = ((int) (value * 1000)) / 1000f;
        accelerationValues[i].setText("" + value);
    }
}

• Lots of jitter
Linear Acceleration

- At rest of table
- Recall
- units are m/s²

![SensorTest](SensorTest)

<table>
<thead>
<tr>
<th>Axis</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x Axis</td>
<td>-0.017</td>
</tr>
<tr>
<td>y Axis</td>
<td>-0.084</td>
</tr>
<tr>
<td>z Axis</td>
<td>-0.034</td>
</tr>
</tbody>
</table>
Zeroing out

• Take average of first multiple (several hundred) events and average
  – shorter time = more error

• Potential error
  – should be 0 at rest

| SensorTest | i: 0, zerovalue: 7.4665865E-4 |
| SensorTest | i: 1, zerovalue: -0.003574672 |
| SensorTest | i: 2, zerovalue: -0.02909316 |

| i: 0, zerovalue: -0.0035472375 |
| i: 1, zerovalue: -0.0018564985 |
| i: 2, zerovalue: -0.022586245 |
Rate of Events

• 1000 events
• SensorManager.SENSOR_DELAY_UI
  – times in seconds: 21, 21, 21
  – 21 seconds / 1000 events
• SensorManager.SENSOR_DELAY_FASTEST
  – times in seconds: 21, 21, 21
• Recall delay of 20,000 micro seconds
• $2 \times 10^4 \times 1 \times 10^3 = 2 \times 10^7 = 20$ seconds
USING SENSORS
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
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
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
Sensor Best Practices

• Testing on the emulator
• Android SDK doesn't provide any simulated sensors
• 3rd party sensor emulator
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
• Must modify app so it uses Sensor Simulator library
Sensor Simulator

Choose Device
- Medium

Basic Orientation
- accelerometer
- magnetic field
- orientation

Extended Orientation
- linear acceleration
- gravity
- rotation vector
- gyroscope

Environment Sensors
- temperature
- light
- proximity
- pressure

Other Sensors
- barcode reader

Write emulator command port and click on set to create connection. Possible IP addresses:
- 10.0.2.2
- 128.83.141.69
- 192.168.1.74
Sensor Simulator

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

• Orientation of the device
• $x$ - tangential to ground and points roughly east
• $y$ - tangential to the ground and points towards magnetic north
• $z$ - perpendicular to the ground and points towards the sky

Orientation Sensor

• Deprecated
• Instead use the Rotation vector sensor
• int TYPE_ROTATION_VECTOR
SENSOR SAMPLE - MOVING BALLS
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

```java
public void onSensorChanged(SensorEvent event)
{
    // 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

```java
// 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

```java
//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(0xFFFFFFFF);
    canvas.drawLine(mX, mY, mX + vX * 15, mY, mPaint);
    mPaint.setColor(0xFF0000FF);
    canvas.drawLine(mX, mY, mX, mY + vY * 15, mPaint);
```
Sensor Sample - TBBT

• Inspired by http://tinyurl.com/bxzaspy and http://tinyurl.com/6nhvnnv
TBBT Sound Effect App

Shake the Device

CRACK !!!!!
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

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