View this session at
https://developers.google.com/events/io/sessions/gooio2012/108/

The sensitive side of Android

Ankur Kotwal, Tim Bray, Tony Chan

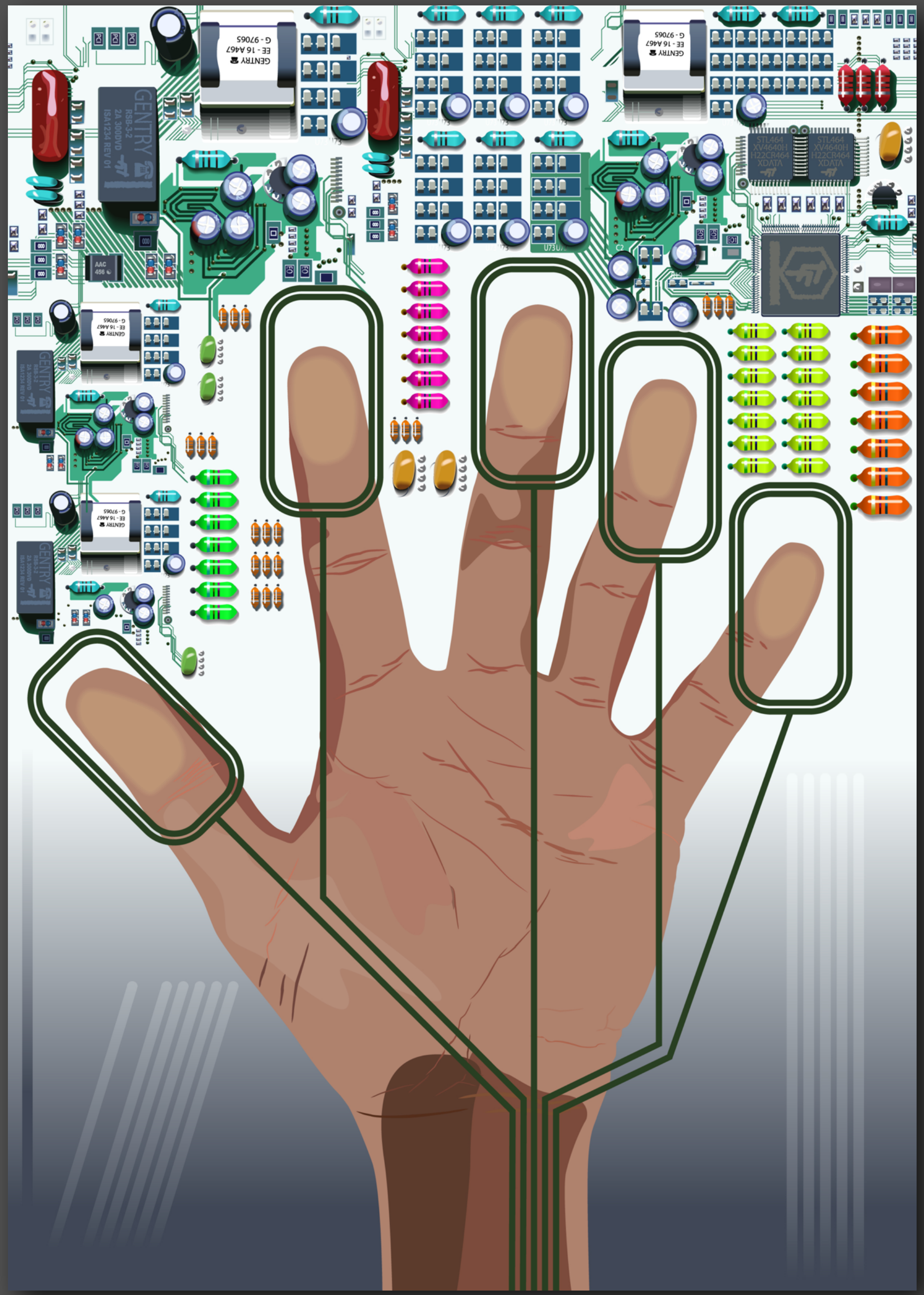
Android Developer Advocates

Demo

Sensitive Blackjack

- Dealer vs Player
- Aim is reach 21
  - Ace is 1 or 11
  - King, Queen, Jack are 10

Touch



Touch events

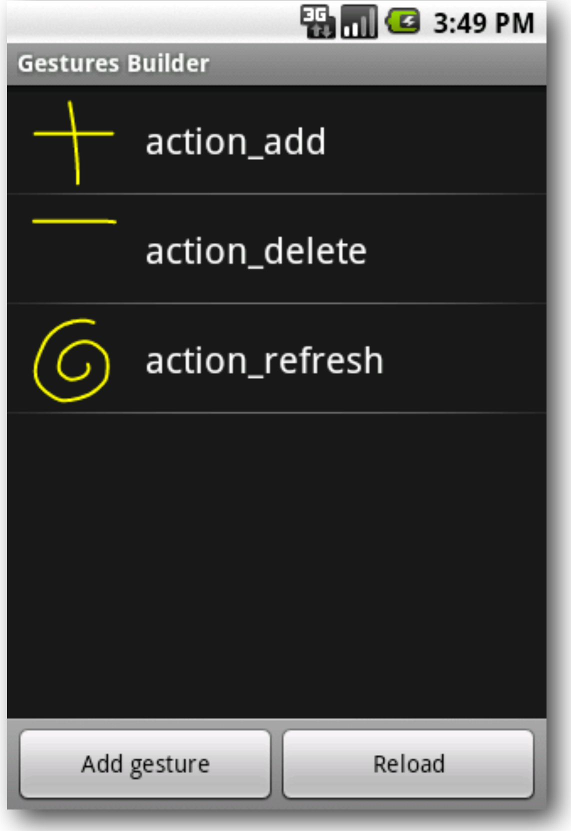
```
public boolean onTouchEvent(MotionEvent event)
```

Easy? Yeah, right.

GestureDetector

VelocityTracker

Gesture builder sample



Gestures Builder

action_add

action_delete

action_refresh

Recipe for multi-finger pinch (1)

```
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_CANCEL
            && event.getAction() != MotionEvent.ACTION_UP) {
        // Capture initial touch. Changed counts are new initial
        if (initialTouches == null || initialTouches.size() !=
                event.getPointerCount()) {
            for (int i = 0; i < event.getPointerCount(); i++) {
                // Capture each touch and calculate maximum distance
                initialDistance = ...
            }
        } else {
            // Capture new touches and calculate distance.
            for (int i = 0; i < event.getpointerCount(); i++) {
                currentDistance = ...
            }
        }
```

Recipe for multi-finger pinch (2)

```
} else {
  // Cancelled motion or last finger up. Clear state.
  initialTouches.clear();
  currentTouches.clear();
}
if (callback != null) {
  static final double DISTANCE_THRESHOLD = 0.3f;
  static final int MIN_POINTERS = 3;

  if (currentDistance < initialDistance * DISTANCE_THRESHOLD &&
      initialTouches.size() >= MIN_POINTERS) {
    callback.onMultiPinchShrink();
  } else if ((initialDistance * DISTANCE_THRESHOLD <
              currentDistance * DISTANCE_THRESHOLD) &&
             (currentTouches.size() >= MIN_POINTERS)) {
    callback.onMultiPinchGrow();
  }
```

Gotcha - close pointers can merge!

Telepathy

Best practices for using Sensors

- Tailor your rate

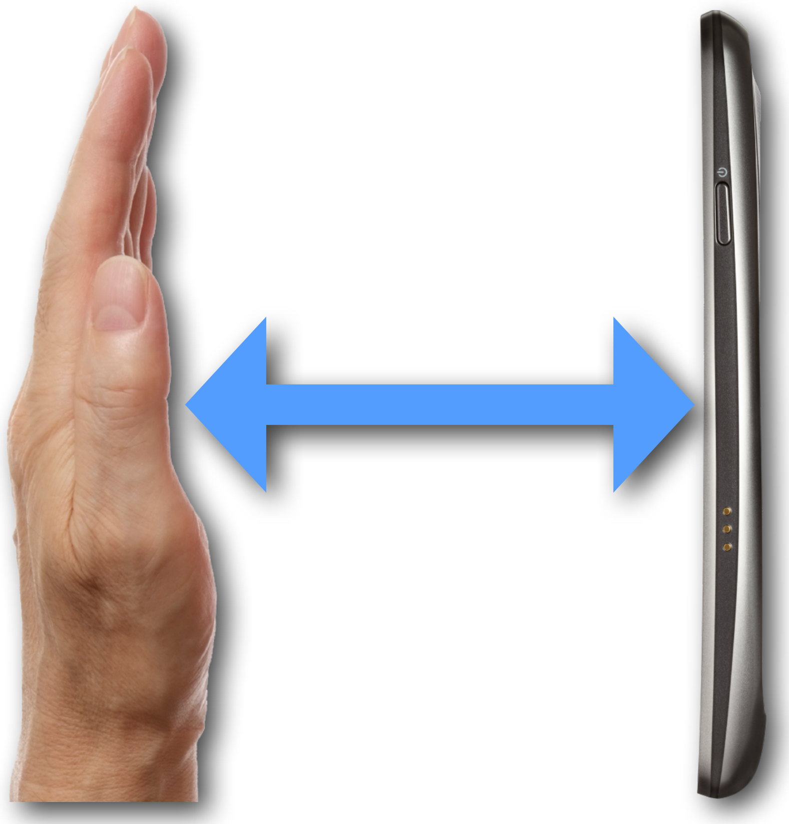
```
sensorManager.registerListener(listener,
   sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
   SensorManager.SENSOR_DELAY_UI);
```

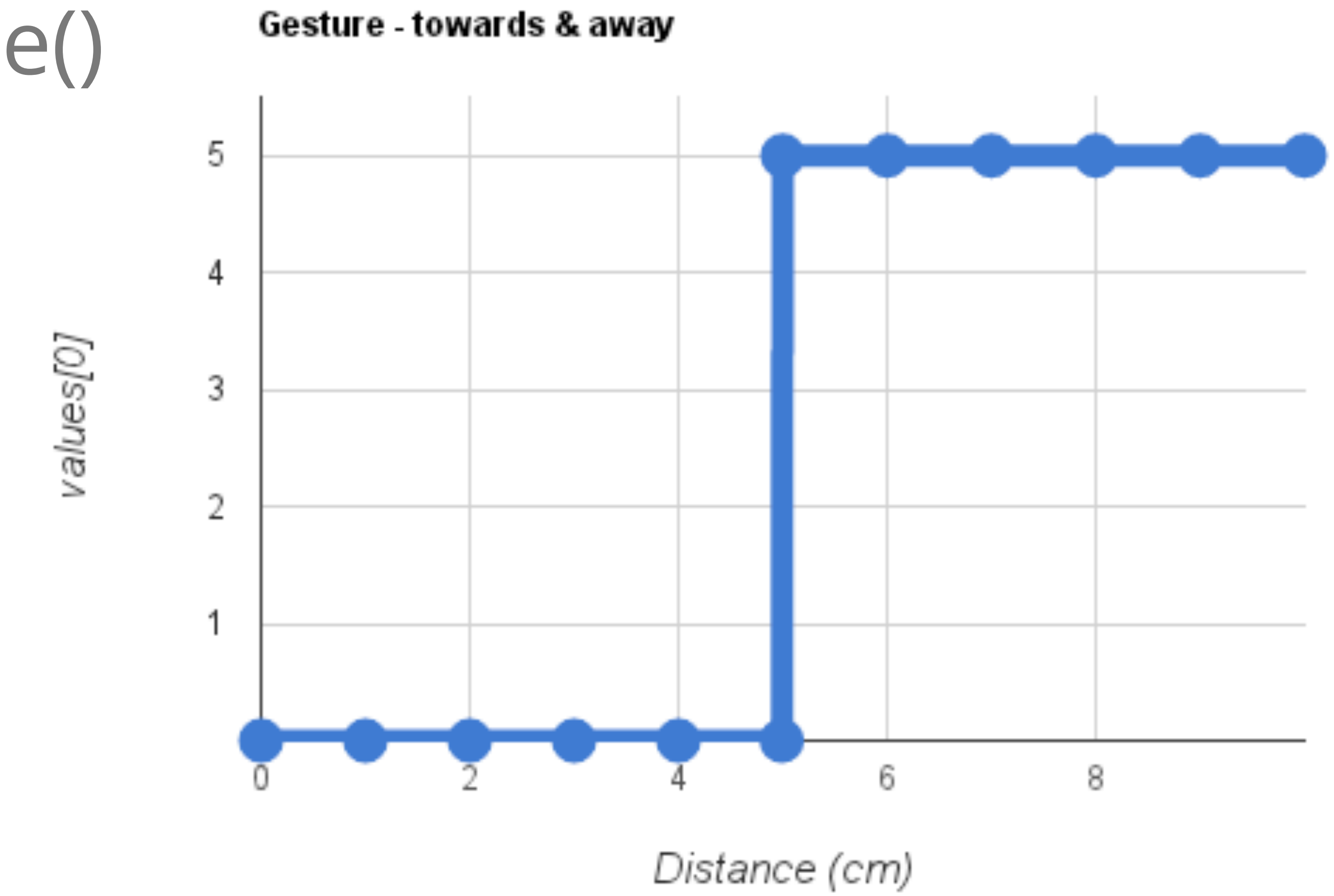
- Unregister aggressively

```
sensorManager.unregisterListener(this);
```

Proximity sensor gestures

- Found on phones to turn off screen
- Continuous or binary values
  - Safest to assume binary results
  - Sensor.getMaximumRange()



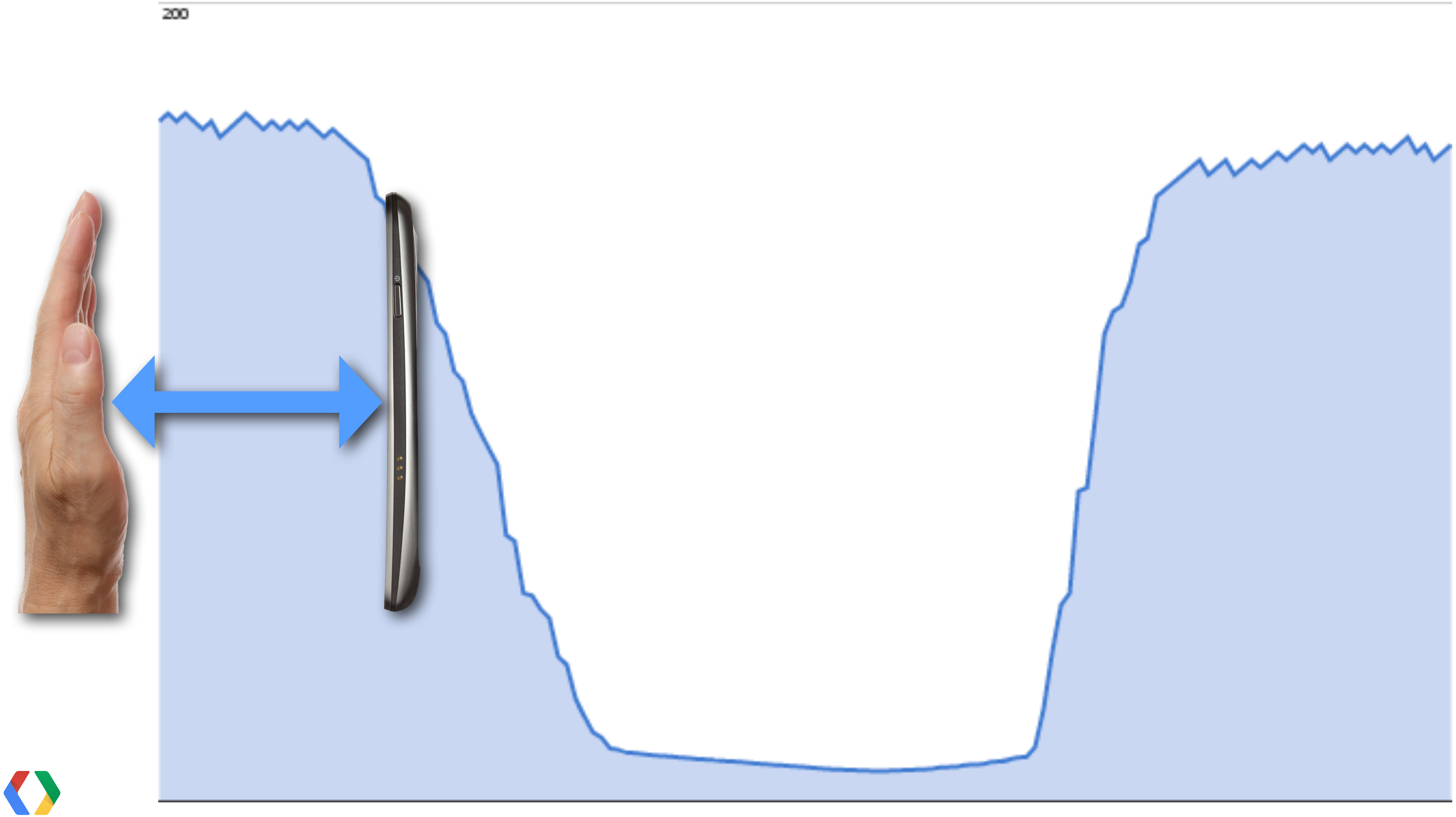


Light sensor gestures (1)

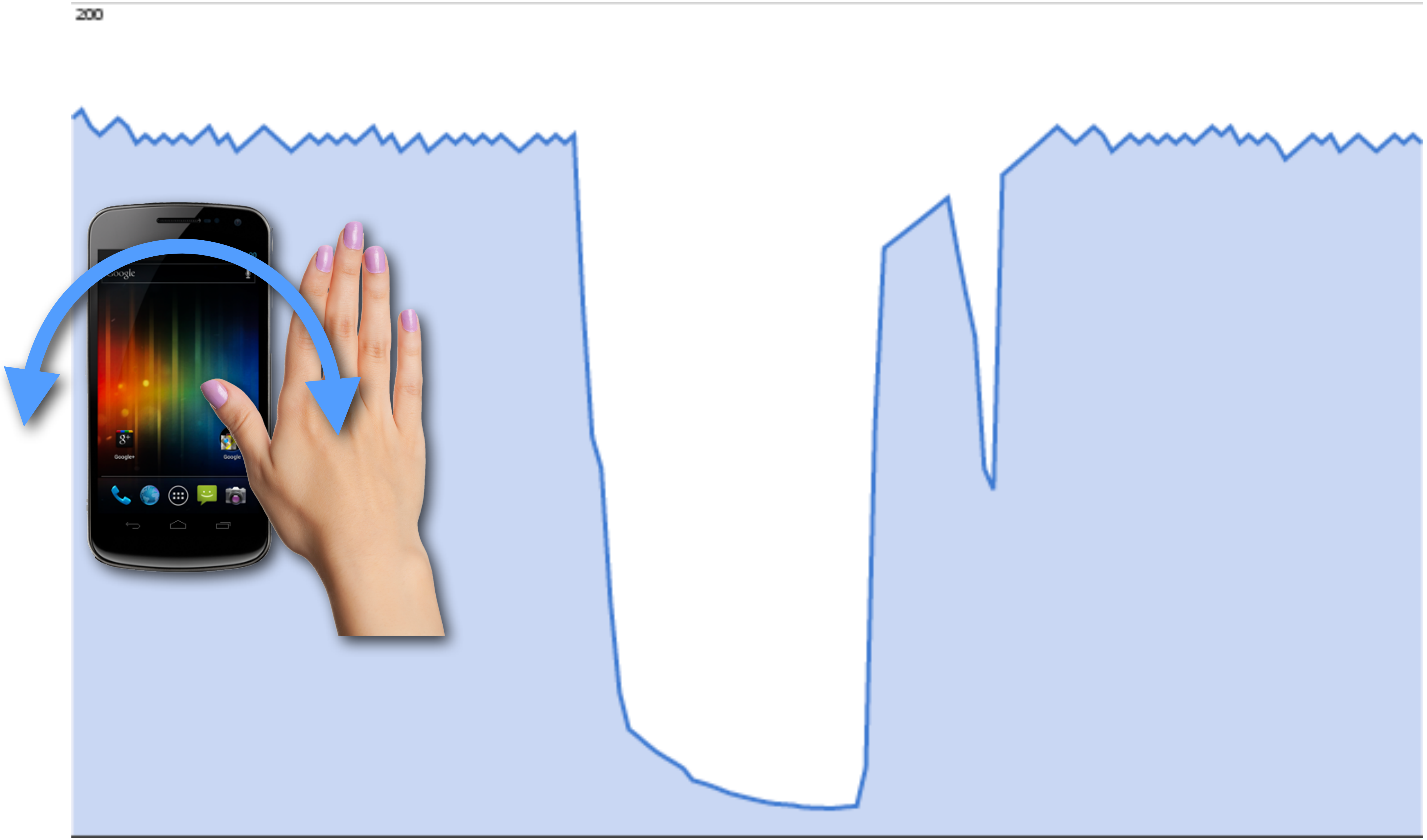
- Lumens per sq. meter
  - Continuous data
- Typically used for adjusting brightness

Illuminance (lux)	Light source
0.27	Full moon on a clear night
50	Family living room
320-500	Office lighting
32,000 - 130,000	Direct sunlight

Light sensor gestures (2)



Light sensor gestures (3)



Recipe for wave gesture

```
private static final int DATA_SIZE = 100;
private static long TIME_THRESHOLD = 500000000L; // 500 ms
private static float GESTURE_THRESHOLD = 0.2f;

public void onSensorChanged(SensorEvent event) {
    mData[mIndex] = (int) event.values[0];
    if (++mIndex >= mData.length) {
        mIndex = 0;
        mDataFull = true;
    }
    // Calculate max light but only after array is full.
    if(mDataFull == true && event.timestamp - mTimeOfCalc > TIME_THRESHOLD)
        for(int point : mData)
            if(point > mCurrentMaxLight)
                // Found new maxLight
    if(event.values[0] < mCurrentMaxLight * GESTURE_THRESHOLD)
        // Wave gesture triggered
}
```

Kinetics

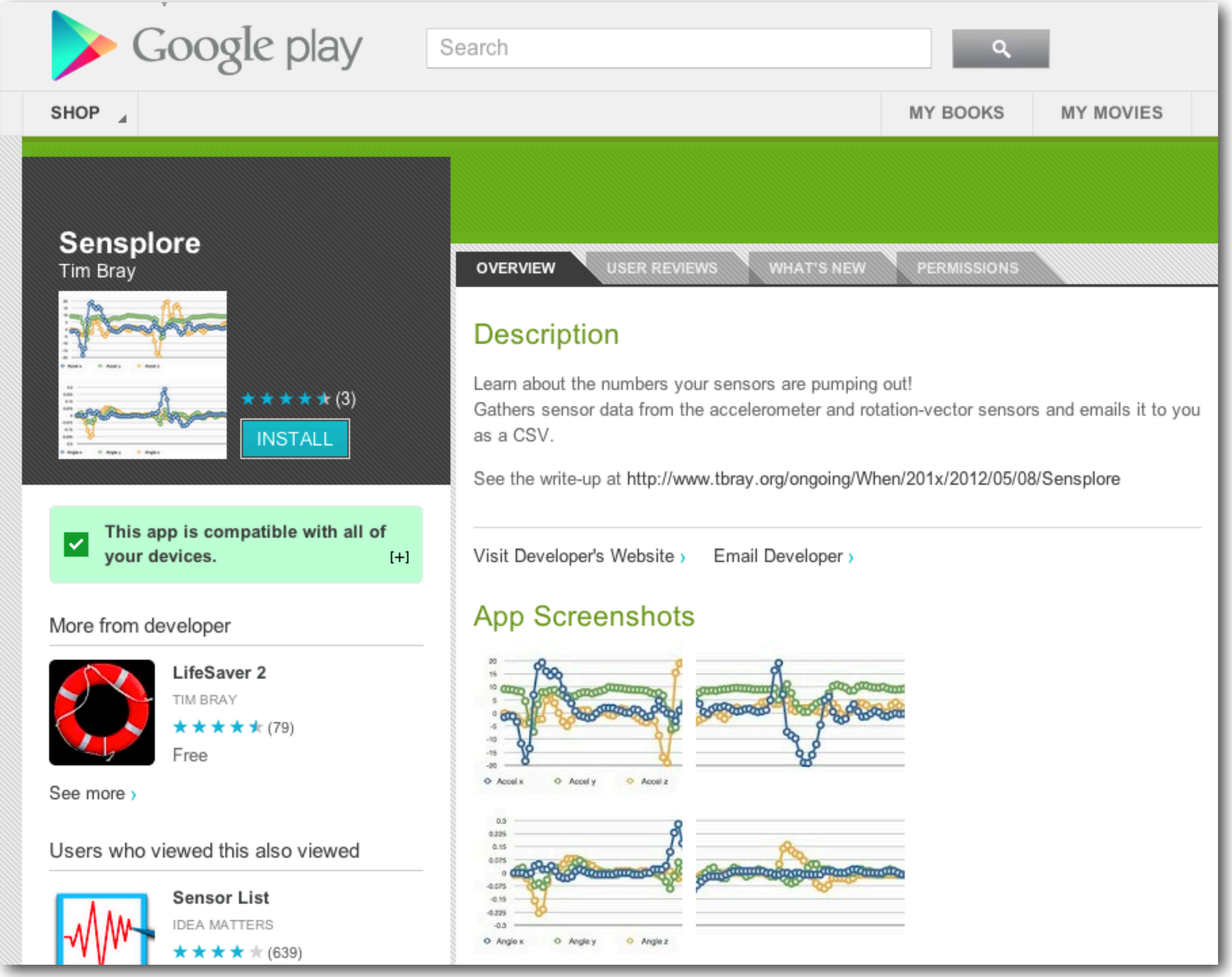
Idea: Controls-free/clean-screen apps

Tap!

Shake!

Wave!

Chop!



Sensplore

Tim Bray

★★★★★ (3)

This app is compatible with all of your devices.

Description

Learn about the numbers your sensors are pumping out!
Gathers sensor data from the accelerometer and rotation-vector sensors and emails it to you as a CSV.

See the write-up at http://www.tbray.org/ongoing/When/201x/2012/05/08/Sensplore

Visit Developer's Website › Email Developer ›

App Screenshots

More from developer

LifeSaver 2
TIM BRAY
★★★★★ (79)
Free

See more ›

Users who viewed this also viewed

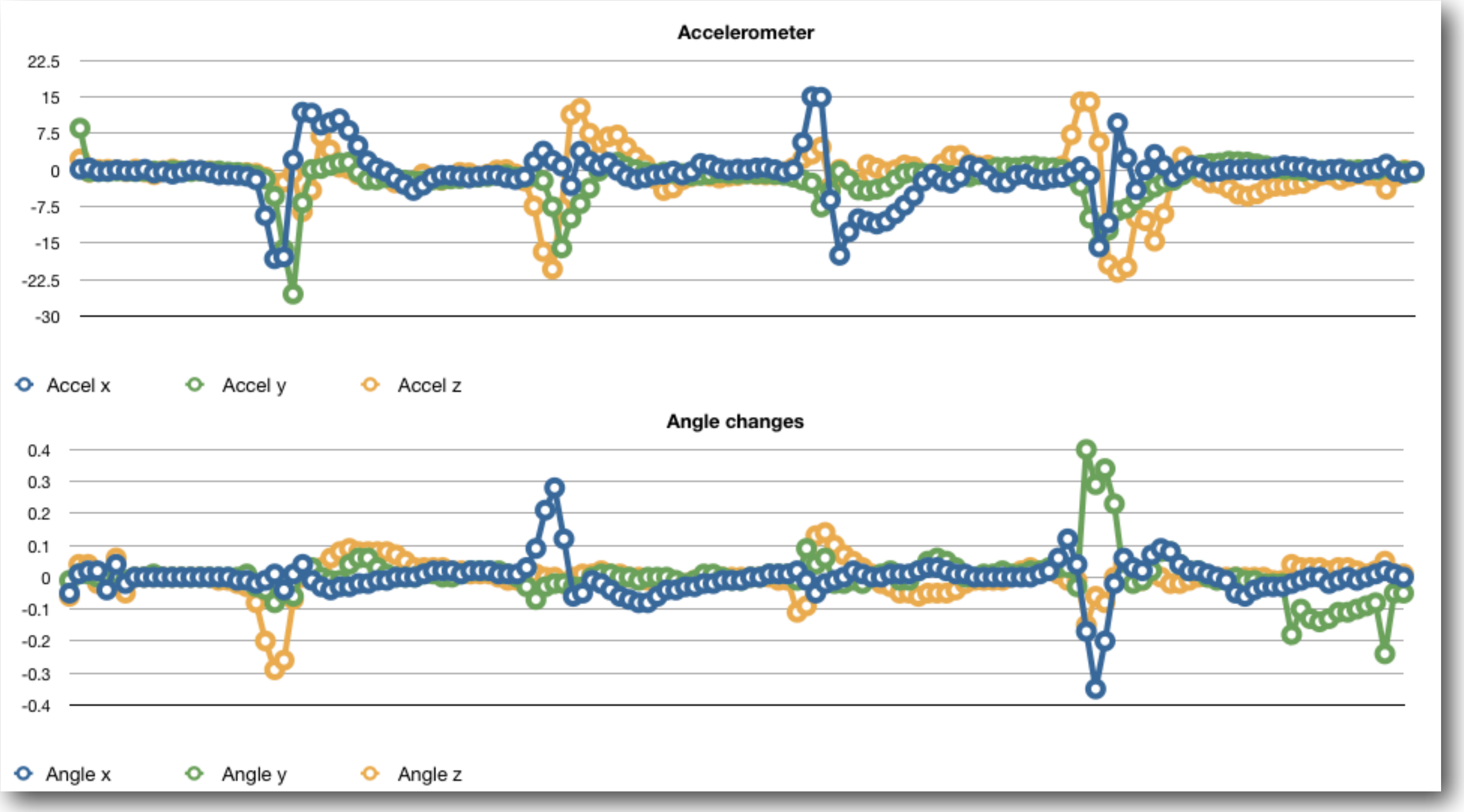
Sensor List
IDEA MATTERS
★★★★★ (639)

Kinetics-related Sensors

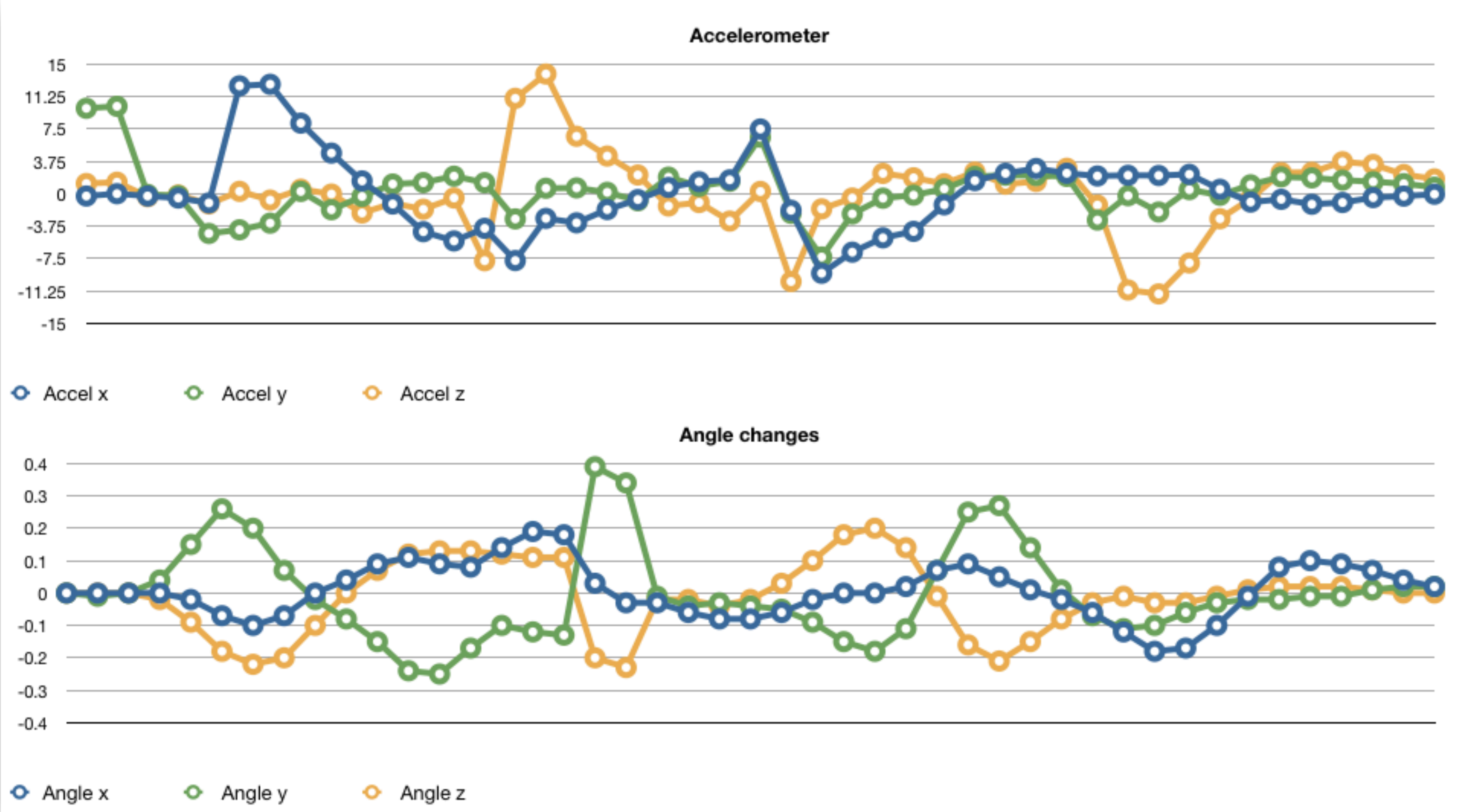
- Accelerometer ⇐ since API 3, includes gravity, very common
- Gravity ⇐ since API 9, synthetic
- Gyroscope ⇐ since API 3, less common
- Linear Acceleration ⇐ since API 9, synthetic, Accelerometer - Gravity
- Magnetic field ⇐ since API 3, A.K.A. “compass”
- Orientation ⇐ deprecated, unreliable, don’t use it
- Rotation Vector ⇐ since API 9, synthetic

Synthetic sensors got a lot better in ICS… *if* there’s a gyroscope.

The gyroscope matters

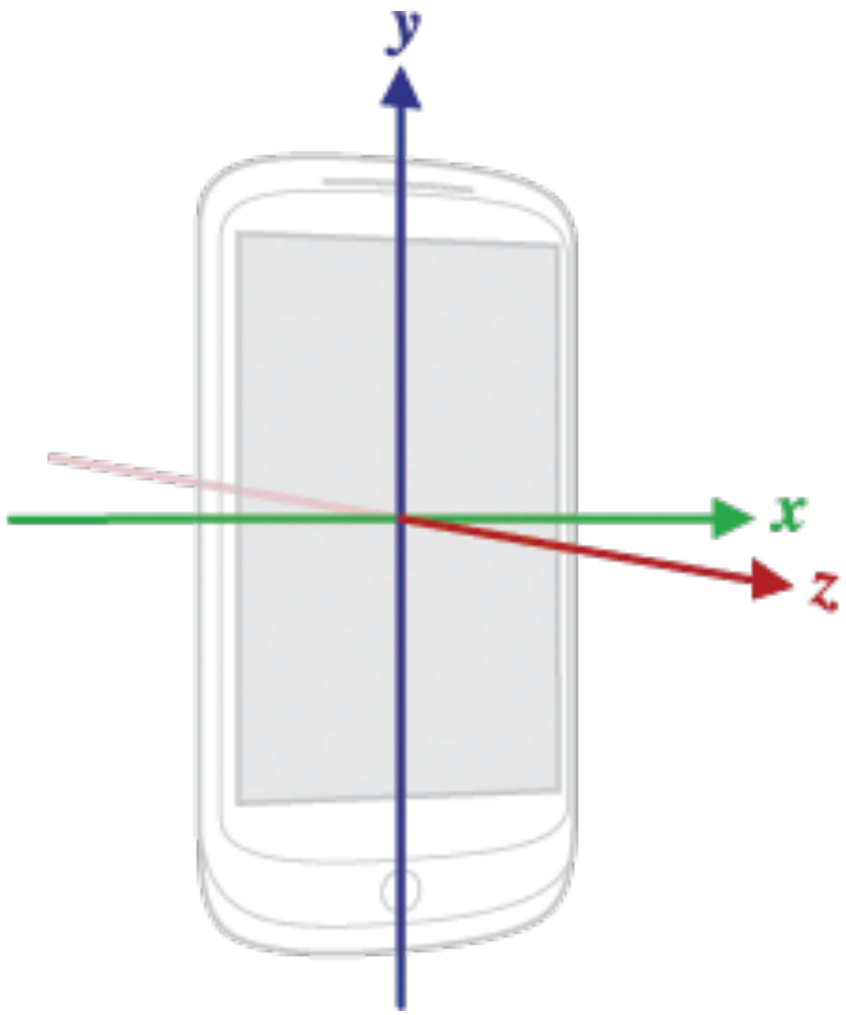


Galaxy Nexus
(with gyroscope)

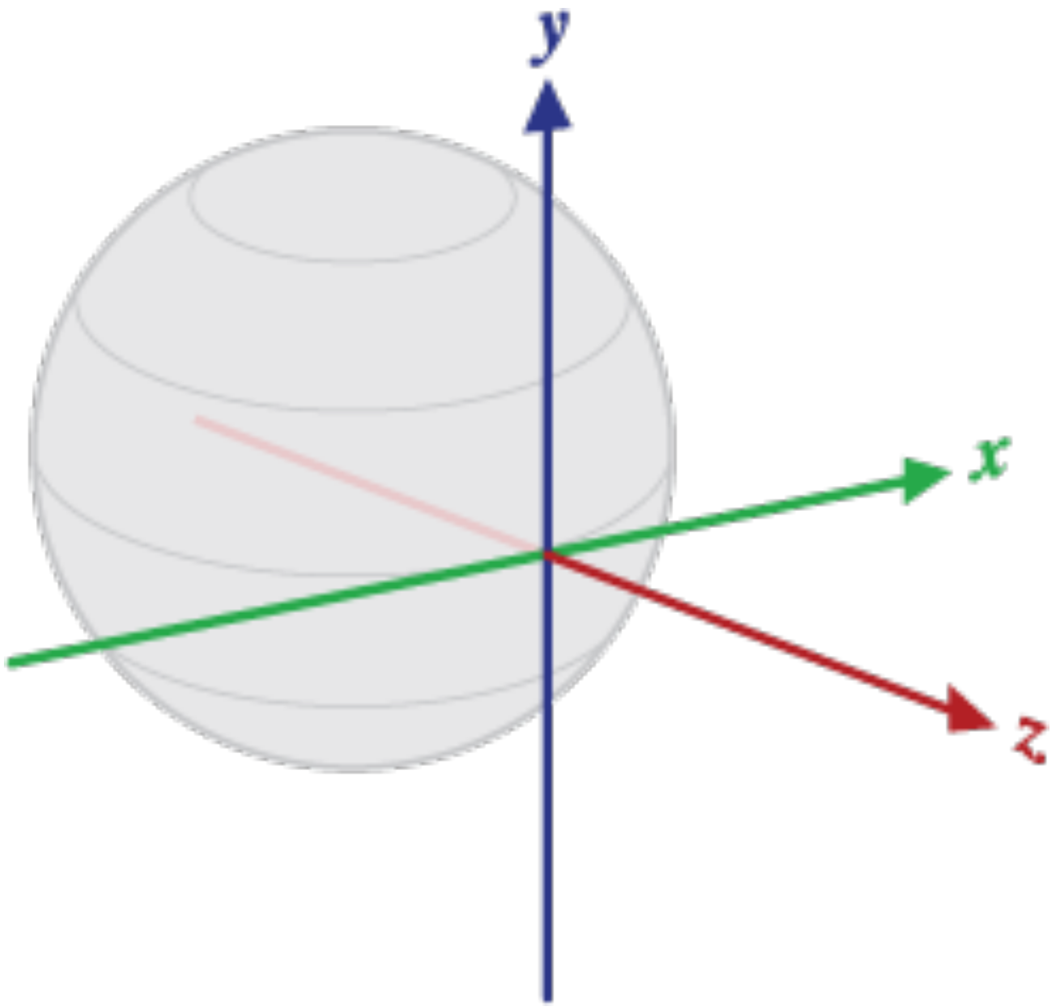


Nexus One
(no gyroscope)

Co-ordinate Systems

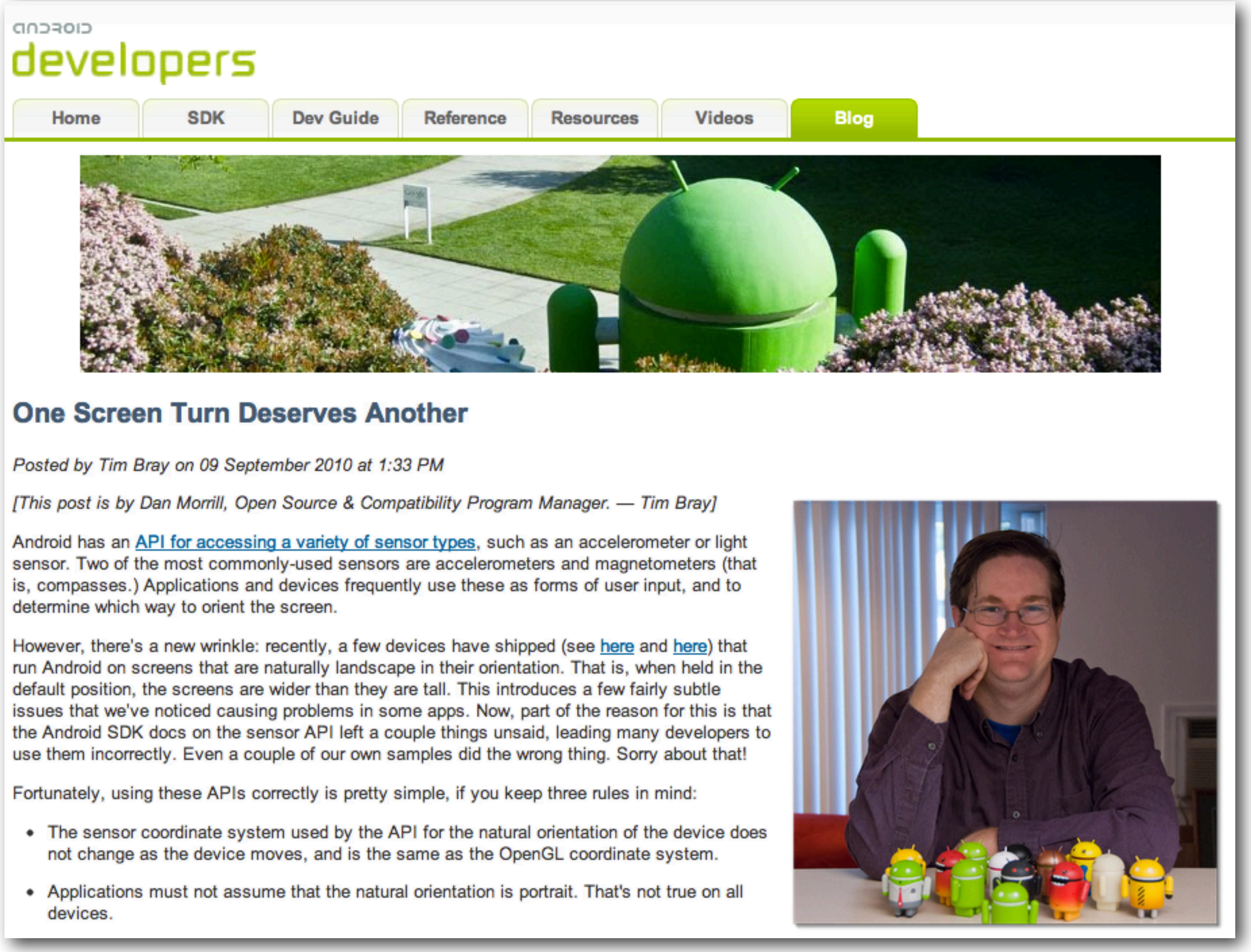


Device



World

Watch out for rotations!



One Screen Turn Deserves Another

Posted by Tim Bray on 09 September 2010 at 1:33 PM

[This post is by Dan Morrill, Open Source & Compatibility Program Manager. — Tim Bray]

Android has an API for accessing a variety of sensor types, such as an accelerometer or light sensor. Two of the most commonly-used sensors are accelerometers and magnetometers (that is, compasses.) Applications and devices frequently use these as forms of user input, and to determine which way to orient the screen.

However, there's a new wrinkle: recently, a few devices have shipped (see here and here) that run Android on screens that are naturally landscape in their orientation. That is, when held in the default position, the screens are wider than they are tall. This introduces a few fairly subtle issues that we've noticed causing problems in some apps. Now, part of the reason for this is that the Android SDK docs on the sensor API left a couple things unsaid, leading many developers to use them incorrectly. Even a couple of our own samples did the wrong thing. Sorry about that!

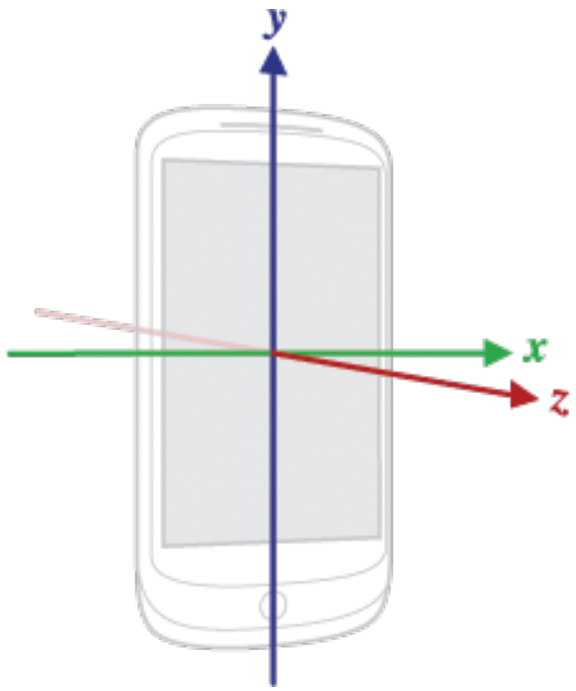
Fortunately, using these APIs correctly is pretty simple, if you keep three rules in mind:

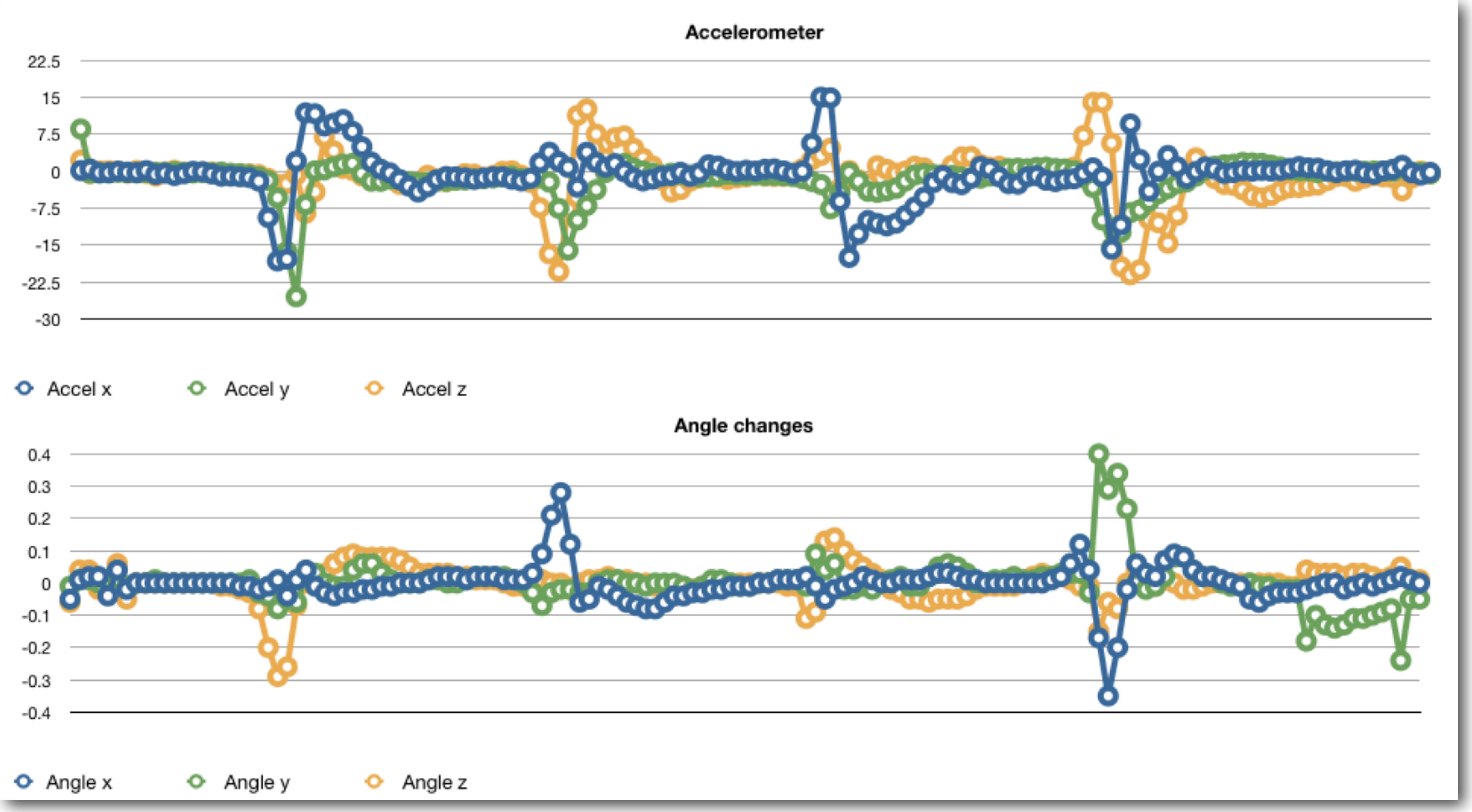
- The sensor coordinate system used by the API for the natural orientation of the device does not change as the device moves, and is the same as the OpenGL coordinate system.
- Applications must not assume that the natural orientation is portrait. That's not true on all devices.

More gotchas

- Power consumption
- Sampling rate
- Static and random variation
- Accelerometer data (unless you're in orbit)

Detecting flip-up: “Four chops”





Detecting Flip-up (1)

```
private class Listener implements SensorEventListener {

    @Override
    public void onSensorChanged(SensorEvent event) {
        switch (event.sensor.getType()) {
        case Sensor.TYPE_ROTATION_VECTOR:
            handleRotation(event);
            break;
        case Sensor.TYPE_GRAVITY:
            handleGravity(event);
            break;
        }
    }
}
```

Detecting Flip-up (2)

```
private void handleRotation(SensorEvent event) {

    final float[] last = mFlipper.last(), next = mFlipper.next();
    mFlipper.flip();
    SensorManager.getRotationMatrixFromVector(next, event.values);
    if (last == null) {
        return;
    }

    SensorManager.getAngleChange(mValues, next, last);
    final float deltaX = mValues[1]; // [z,x,y]
    final boolean plusX = (deltaX > THRESHOLD_ANGLE);
    final boolean minusX = (deltaX < -THRESHOLD_ANGLE);
    . . .
```

Detecting Flip-up (3)

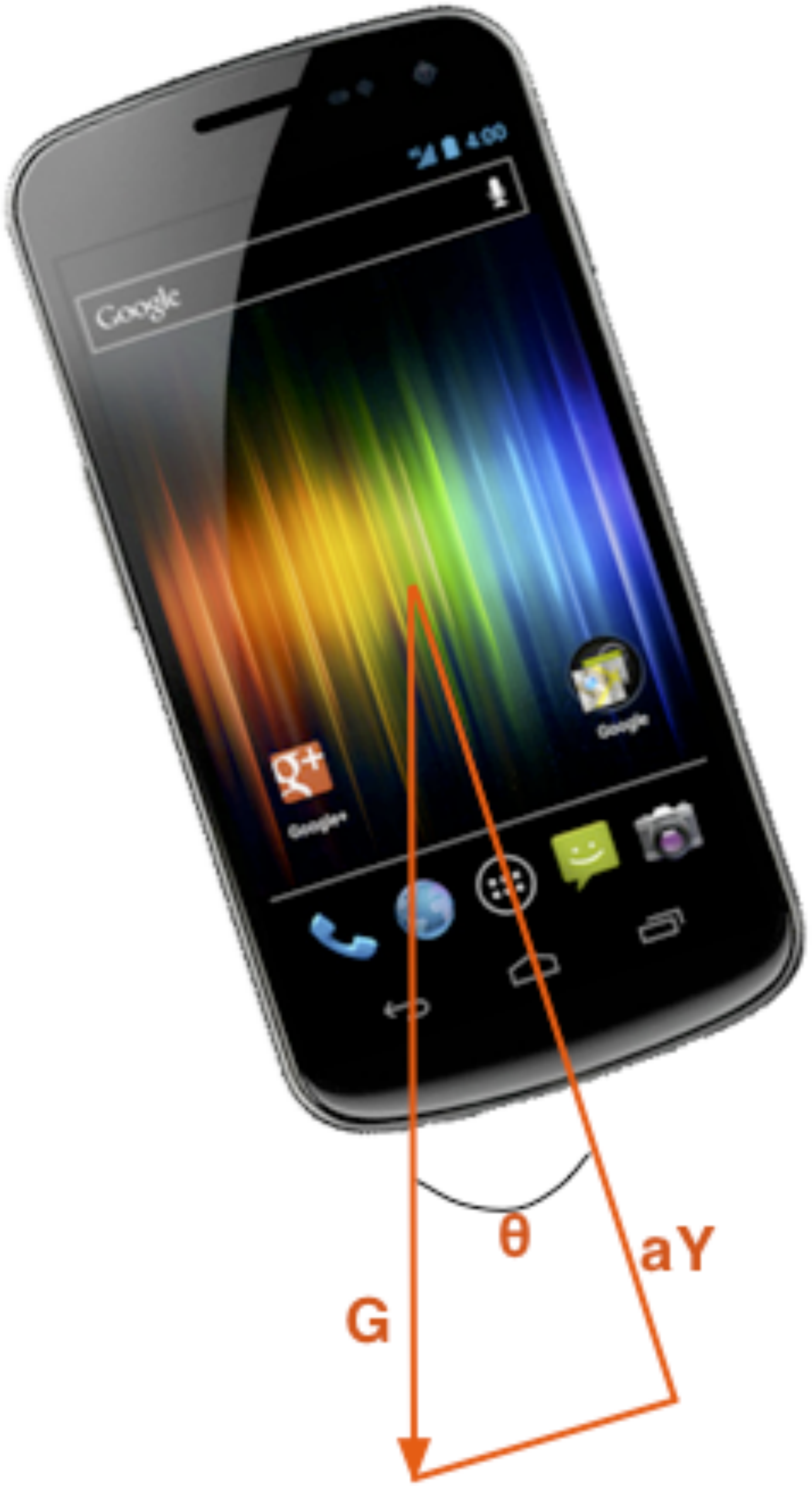
```
switch (mState) {
case STATE_AT_REST:
    // ignore first few events when arriving at rest;
    //  stabilizing after a gesture
    if (mStateDuration < THRESHOLD_REST_DURATION) {
        mStateDuration++;
    } else if (minusX) {
        mState = STATE_MOVING_FORWARD;
        mStateDuration = 1;
    } else if (plusX) {
        mState = STATE_MOVING_BACKWARD;
        mStateDuration = 1;
    }
    break;
    . . .
```

Detecting Flip-up (4)

```
case STATE_MOVING_FORWARD:
   if (minusX) {
       mStateDuration++;
   } else {
       if (mStateDuration > THRESHOLD_MOVING_DURATION) {
           send(FLIP_UP);
       }
       mState = STATE_AT_REST;
       mStateDuration = 1;
   }
   break;
```

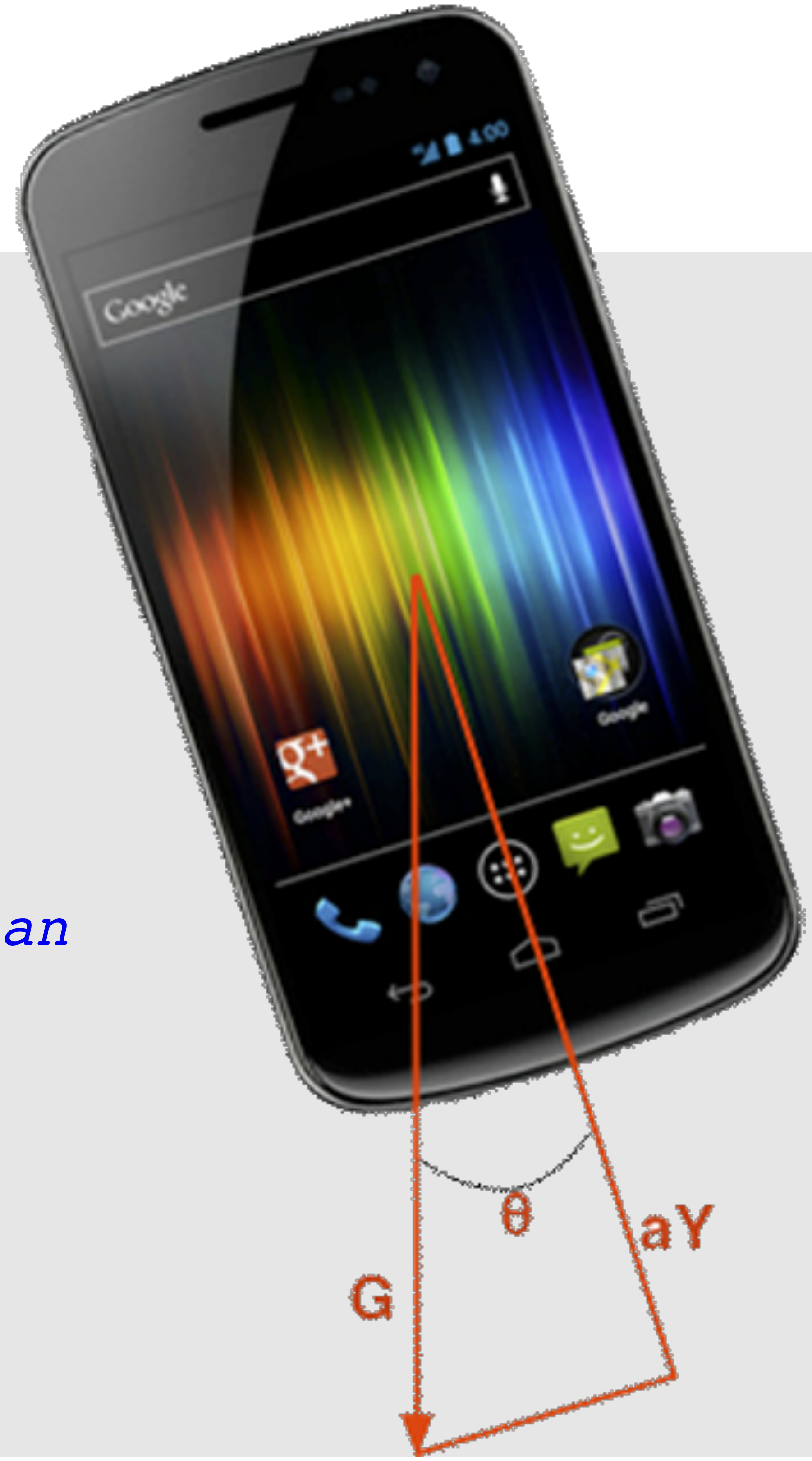
```
private void send(int kinetic) {
    stop();
    if (!mCustomer.kineticRecognized(kinetic)) {
        start();
    }
}
```

Which way is up?



Recipe for Tilt-tracking

```
private void onSensorChanged(SensorEvent e) {
    // avoid NaN's provoked by wonky sensor readings
    double gy = e.values[1] / SensorManager.GRAVITY_EARTH;
    if (gy > 1)
        gy = 1;

    // correct the range so it goes smoothly from 0 to 180
    //  degrees. The X value tells you, roughly speaking,
    //  whether you’re leaning left or right.  The Y value
    //  goes from 0 at the top to +90 whichever way you lean
    //  the device
    double theta = Math.acos(gy);
    if (e.values[0] < 0)
        theta = (Math.PI/2) - theta;
    else
        theta = (Math.PI/2) + theta;
    setTilt(mAverage.add(theta));
```



Tilting a Mostly-flat device

1. Collect ACCELEROMETER triples.
2. If the device is held flat and tilted, the X and Y values can be used directly as acceleration numbers.
3. These can be used to describe Verlet integration, see lonesock.net/article/verlet.html
4. Useful for games! See example at AccelerometerPlayActivity.java in the SDK samples.

Light Saber!?!

1. Collect LINEAR_ACCELERATION triples, but...
2. Unfortunately, the acceleration values are insufficiently accurate and steady to enable the use of Verlet, but...
3. You can usefully measure total acceleration, sqrt(aX^2 + aY^2 + aZ^2), to detect shakes and so on, but...
4. It’ll never rival a Wii or Kinect using current typical mobile-device sensors.

Recipe for a Compass (1)

```
private float[] accel;
private float[] mag;
public void onSensorChanged(SensorEvent event) {
    if (event.accuracy == SensorManager.SENSOR_STATUS_UNRELIABLE) {
        return;
    }

    switch (event.sensor.getType()) {
    case Sensor.TYPE_ACCELEROMETER:
        accel = event.values.clone();
        break;
    case Sensor.TYPE_MAGNETIC_FIELD:
        mag = event.values.clone();
        break;
}
```

Recipe for a Compass (2)

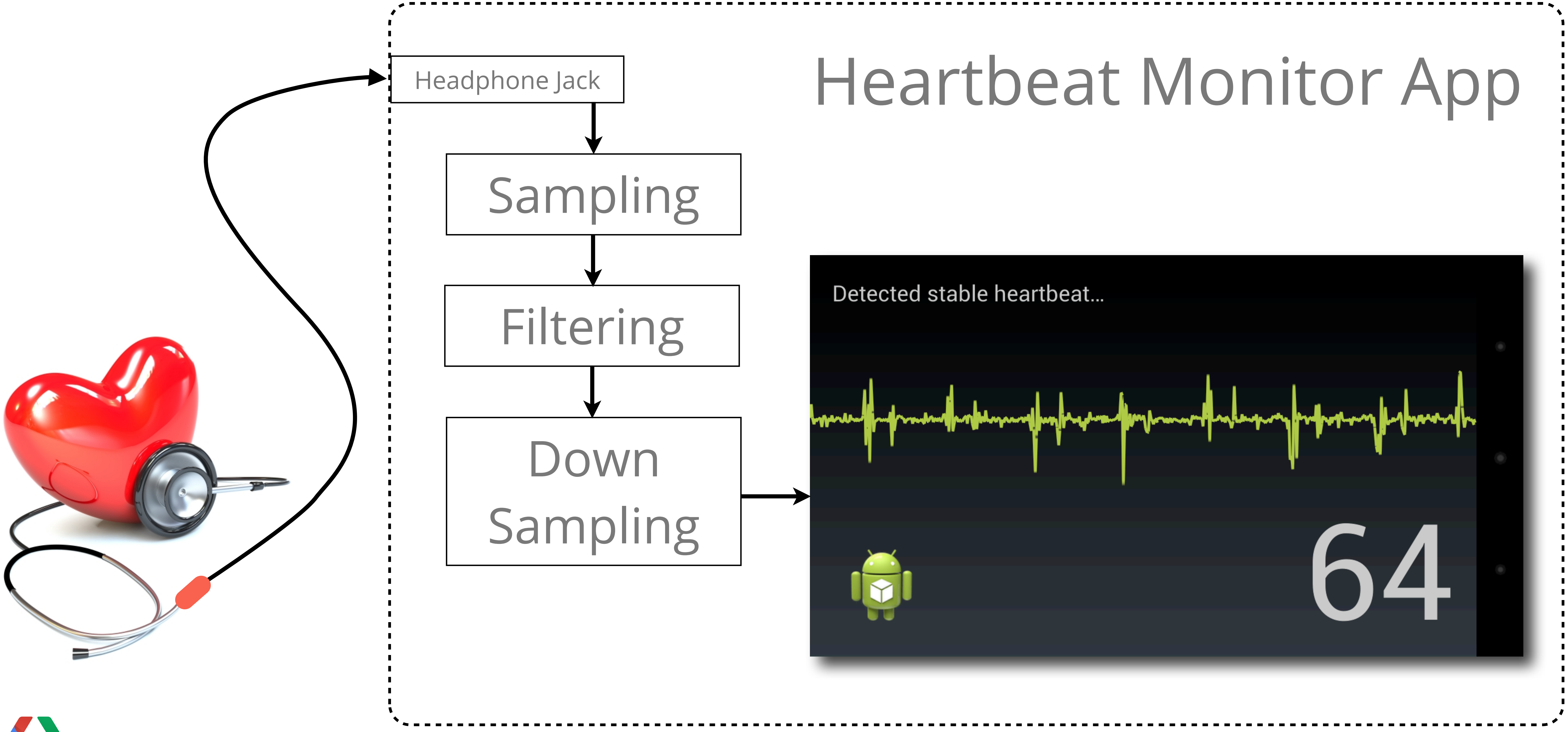
```
if (accel != null && mag != null) {
    float[] r = new float[16];
    float[] outR = new float[16];
    float[] euler = new float[3];
    if (true == SensorManager.getRotationMatrix(r, null, accel, mag)) {
        // Correct for orientation
        SensorManager.remapCoordinateSystem(r, SensorManager.AXIS_X,
            SensorManager.AXIS_Z, outR);
        SensorManager.getOrientation(outR, euler);

        // Convert to degrees
        compassValue = euler[0] * 180 / ((float) Math.PI);
    }
}
```

Audio

Demo

Heartbeat Monitor

Heartbeat Monitor



Heartbeat Monitor App

Headphone Jack

Sampling

Filtering

Down Sampling

Detected stable heartbeat...

64

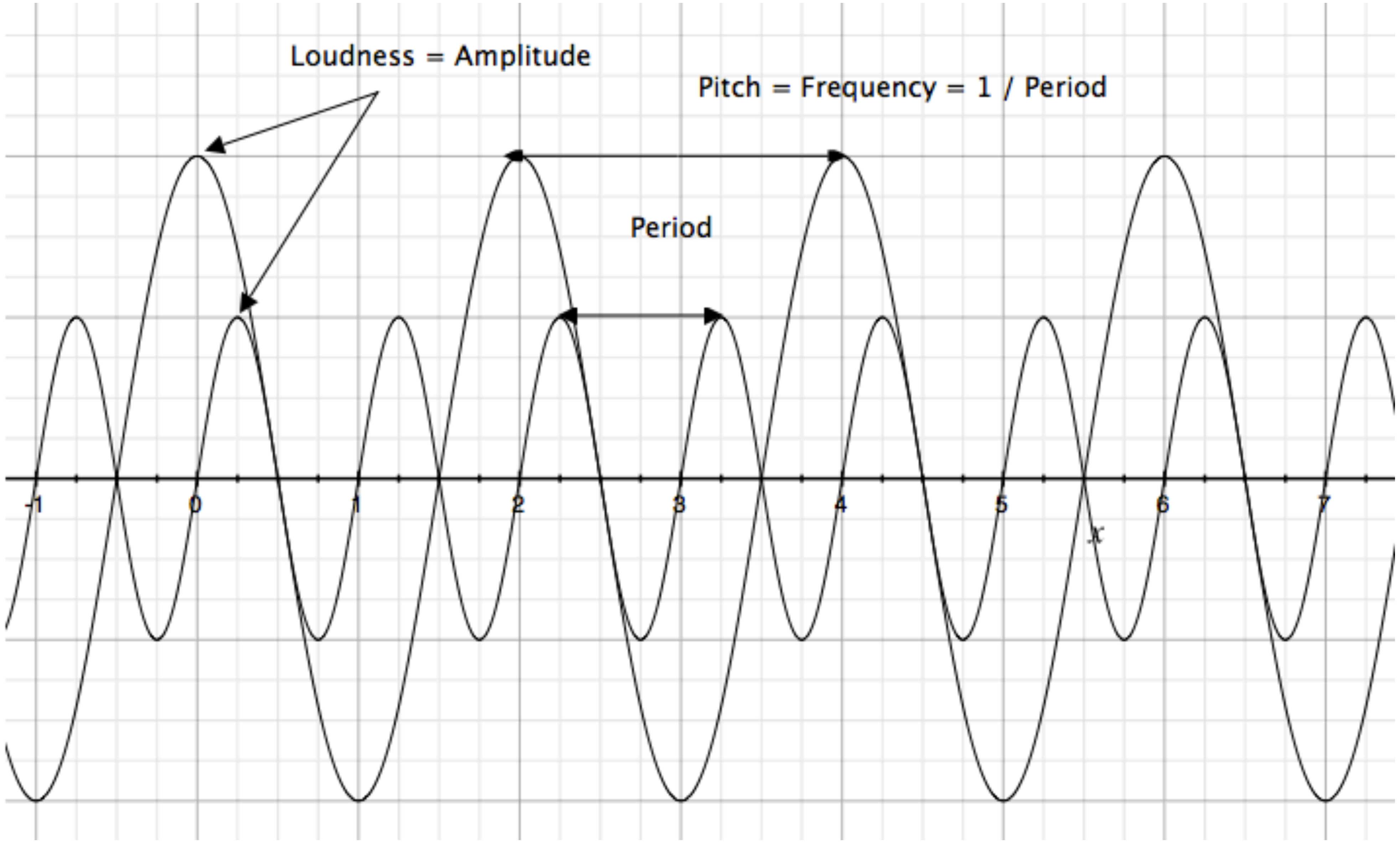
Microphone

- MediaRecorder - API 1, Android 1+
  - Capturing and encoding a variety of common audio formats
  - Use for regular audio recording

- AudioRecord - API 3, Android 1.5+
  - If you need to process the audio signals

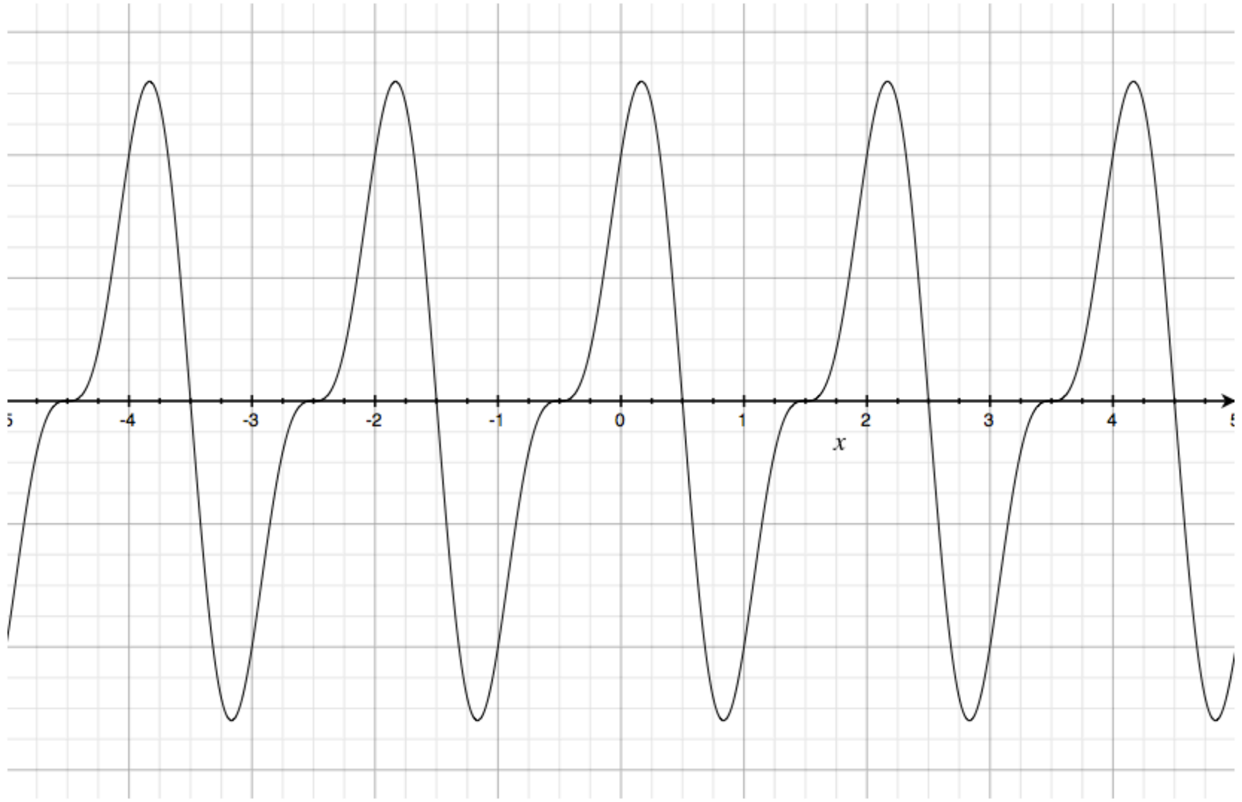
Sound Fundamentals

- Human hearing range: 20Hz - 20kHz
- Sound waves - Pitch and loudness

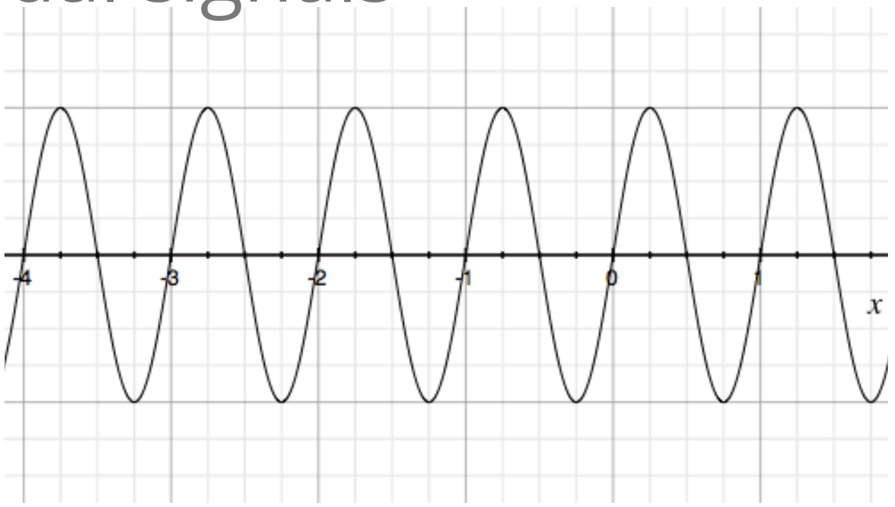


Sound Fundamentals

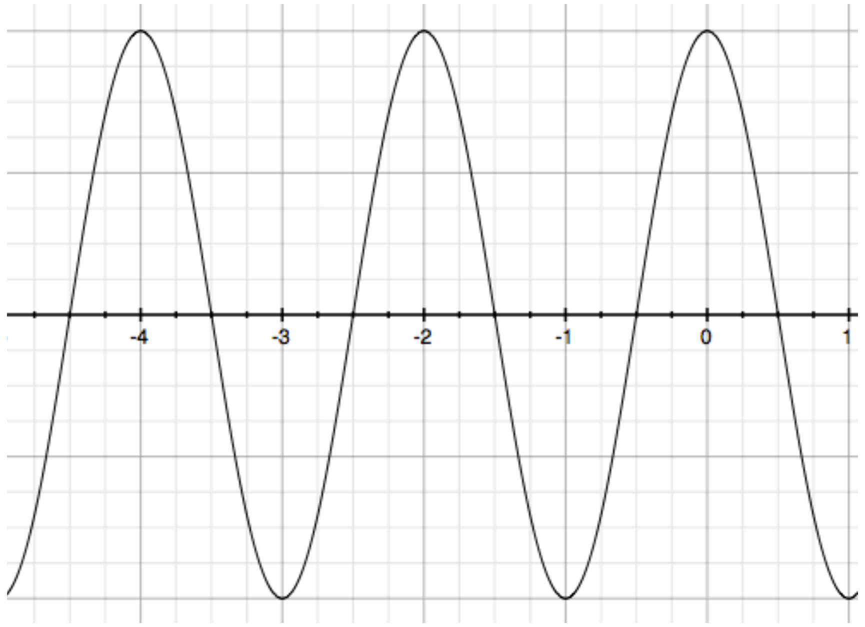
- Sound Wave - superposition of multiple sinusoidal signals



$$F(a_1x_1 + a_2x_2) = a_1F(x_1) + a_2F(x_2)$$

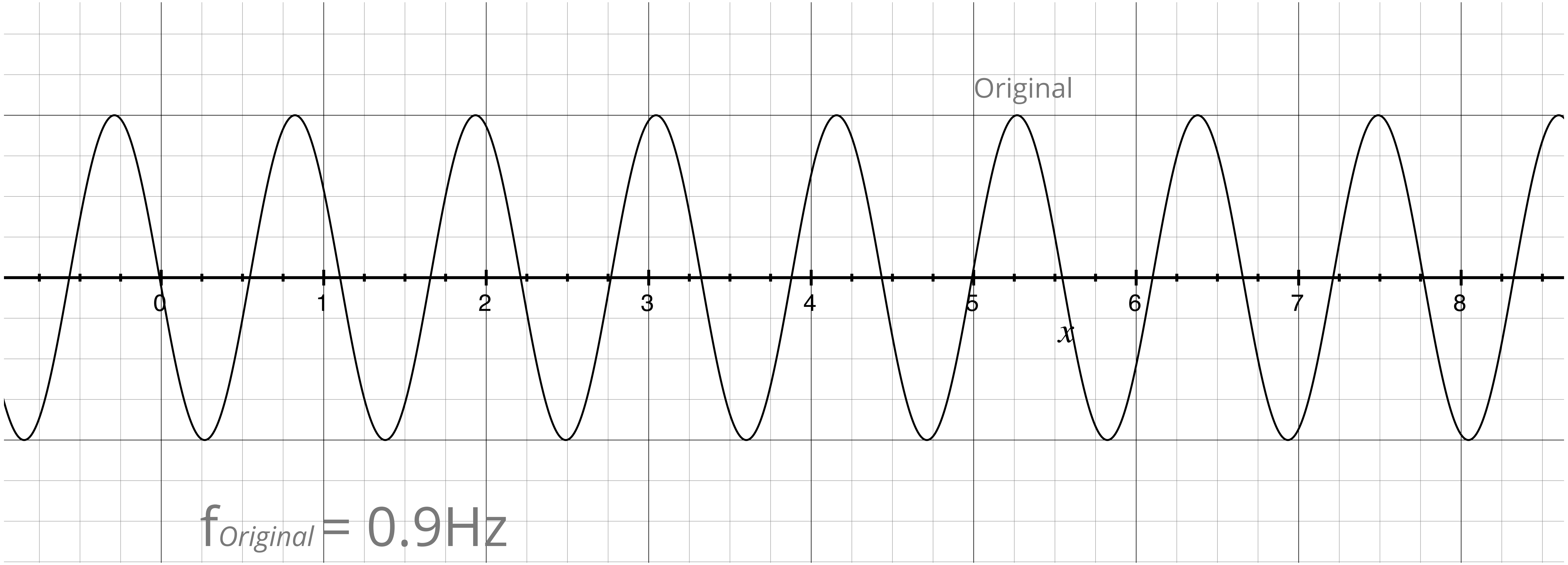


+



Sampling

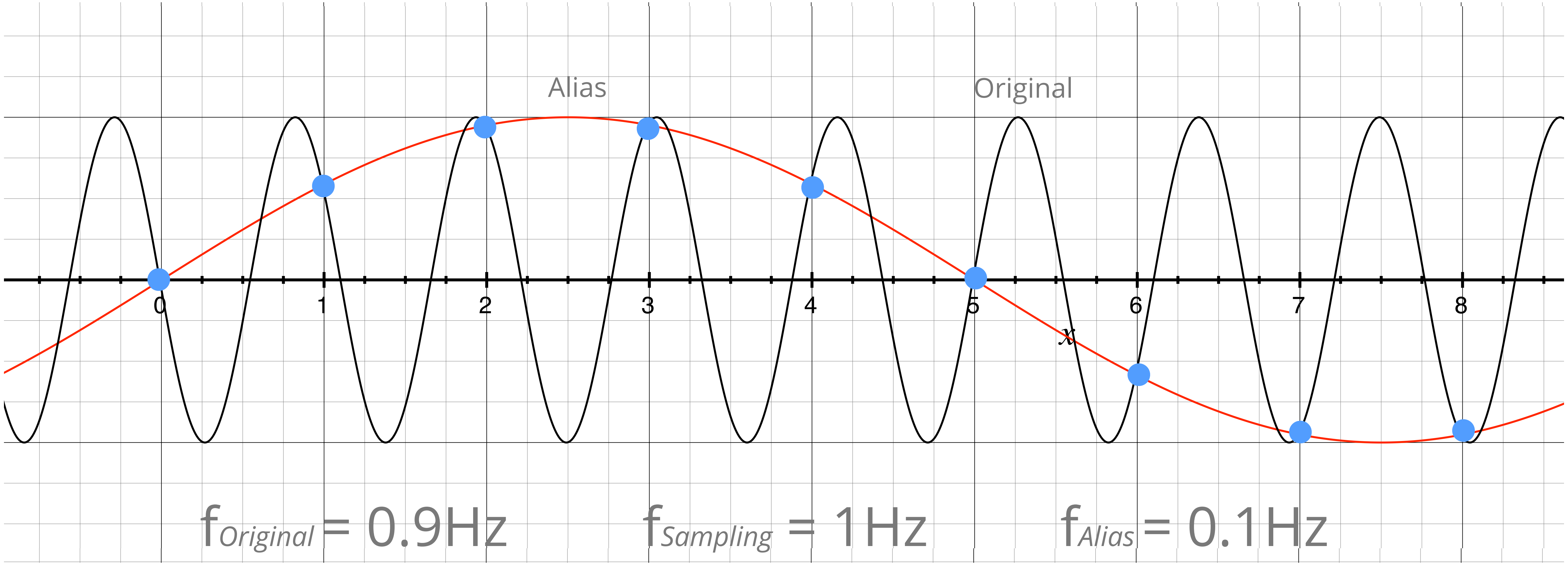
- All Android compatible devices support 44.1kHz sampling rate
- To avoid aliasing, sampling rate must be **> 2 * highest frequency component of your signal**



$f_{Original} = 0.9\text{Hz}$

Sampling

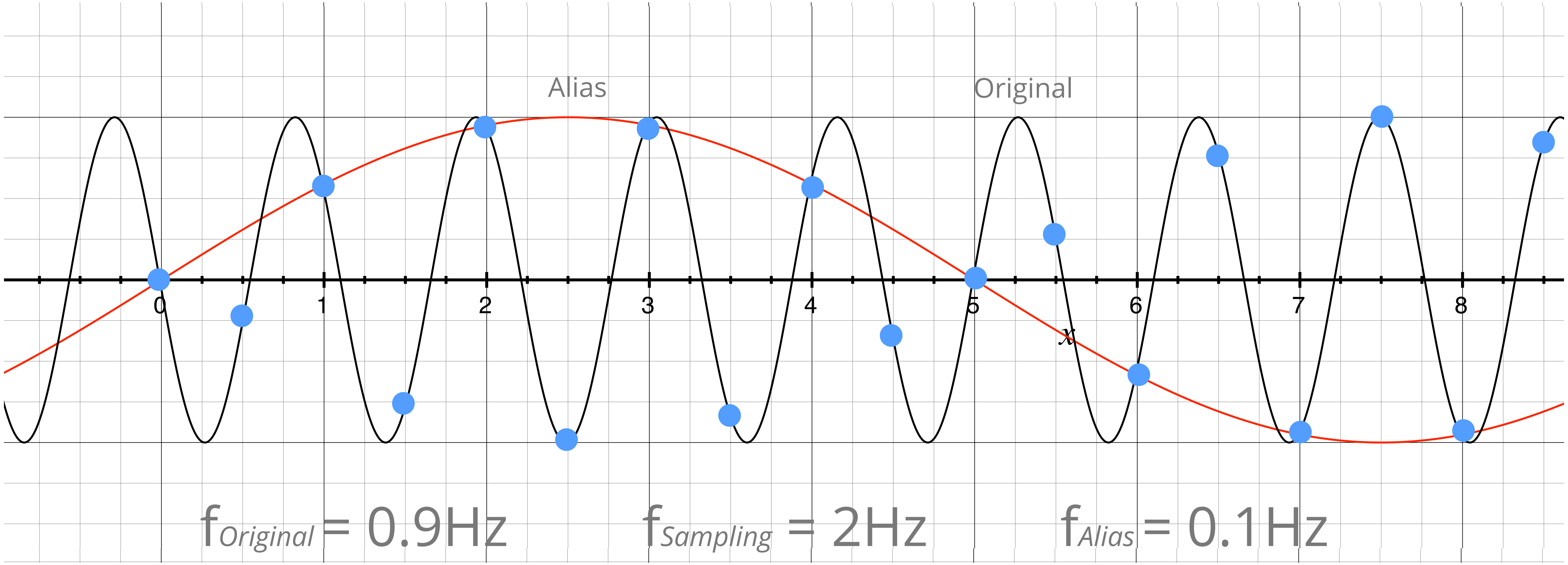
- All Android compatible devices support 44.1kHz sampling rate
- To avoid aliasing, sampling rate must be **> 2 * highest frequency component of your signal**



$f_{Original} = 0.9\text{Hz}$ $f_{Sampling} = 1\text{Hz}$ $f_{Alias} = 0.1\text{Hz}$

Sampling

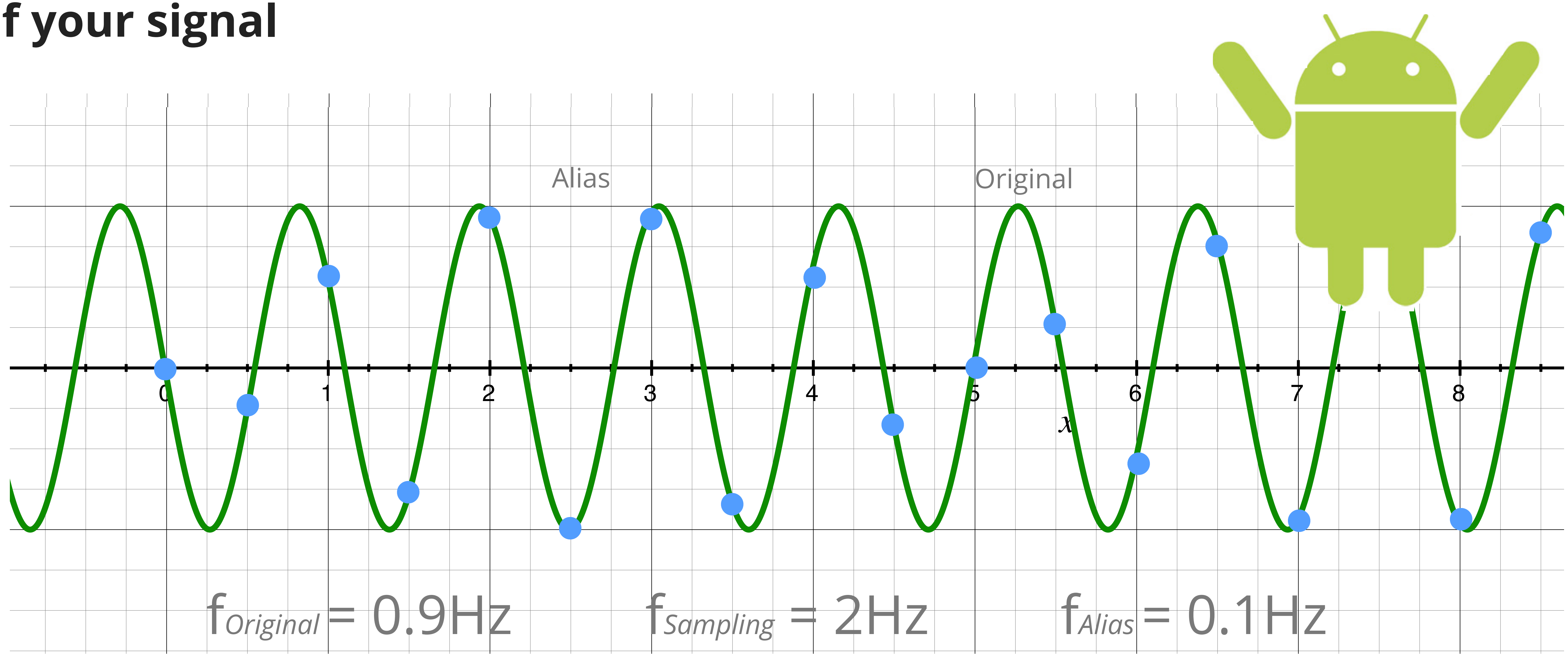
- All Android compatible devices support 44.1kHz sampling rate
- To avoid aliasing, sampling rate must be **> 2 * highest frequency component of your signal**



$f_{Original} = 0.9Hz$ $f_{Sampling} = 2Hz$ $f_{Alias} = 0.1Hz$

Sampling

- All Android compatible devices support 44.1kHz sampling rate
- To avoid aliasing, sampling rate must be **> 2 * highest frequency component of your signal**



$f_{Original} = 0.9Hz$ $f_{Sampling} = 2Hz$ $f_{Alias} = 0.1Hz$

Buffering

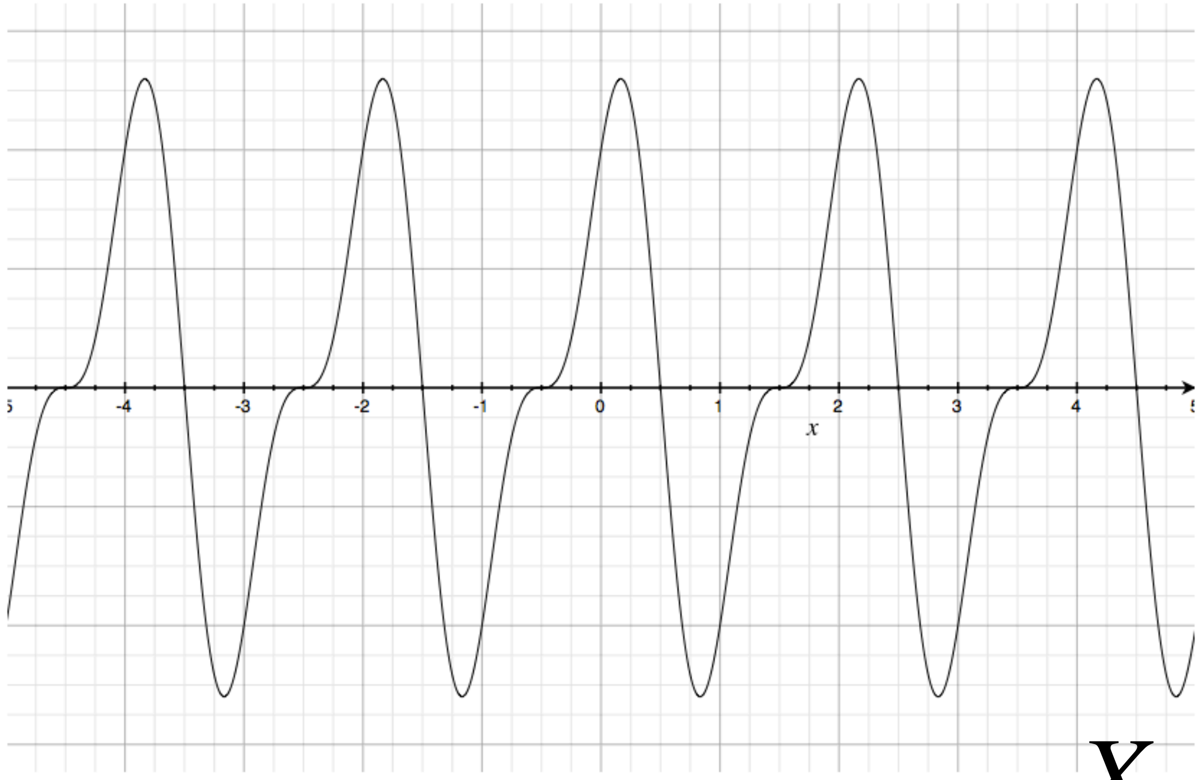
- Choice of buffer size
  - Responsiveness
  - Memory/CPU cycles
  - Tolerance to failures

- Common pitfalls
  - Always use `getMinBufferSize(int sampleRateInHz, ...)`
  - For ENCODING_PCM_8BIT, use `read(byte[] audioData, ...)`
  - For ENCODING_PCM_16BIT, use `read(short[] audioData, ...)`

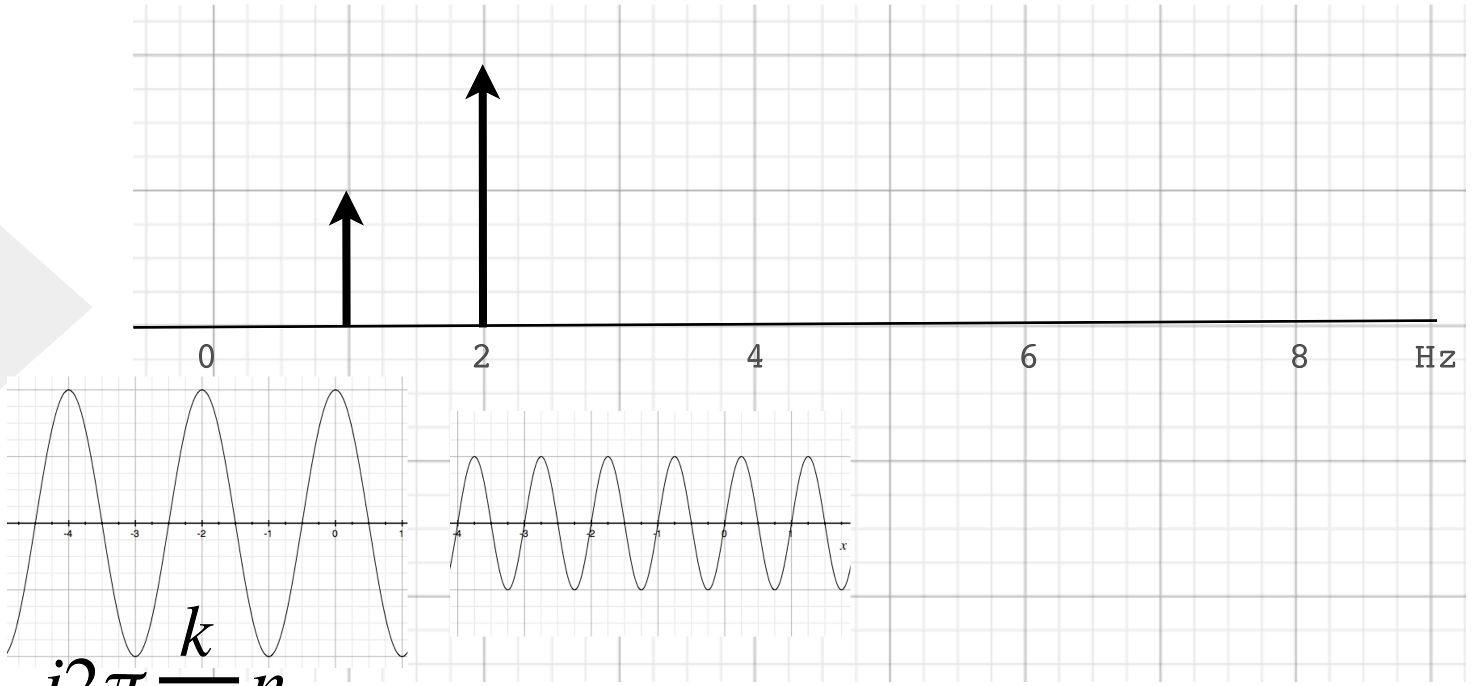
Audio Signal Processing

- Filter noises
- Spectral analysis
- Hard to do it in time domain but easy in frequency domain

Time Domain



Frequency Domain



$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-i2\pi \frac{k}{N} n}$$

Discrete Fourier Transform (DFT)

- Computing Intensive
- In practice: Fast Fourier Transform (FFT)
- Things to look out for
    - **Garbage Collection**
    - In-place algorithm
    - Native code

```
double[] real = new double[buffer.length];
double[] imaginary = new double[buffer.length];

~~Complex[] complex = new Complex[buffer.length];~~
```

Discrete Fourier Transform (DFT)

- Computing Intensive
- In practice: Fast Fourier Transform (FFT)
- Things to look out for
  - Garbage Collection
  - **In place algorithm**
  - Native code

```
FFT.fft(real, imaginary);
lowPassFilter(real, imaginary);
FFT.ifft(real, imaginary);
```

Discrete Fourier Transform (DFT)

- Computing Intensive
- In practice: Fast Fourier Transform (FFT)
- Things to look out for
  - Garbage Collection
  - In-place algorithm
  - **Native code**

Consider using NDK

Making Sense out of your Sensor Data

- **Thresholds**
- Time
- Statistics
  - Mean, Median, Mode, Range, etc
  - Moving/Weighted Average

```
if (magnitude >= HEARTBEAT_AMPLITUDE_THRESHOLD) { // Detect heartbeat peaks
if (diff < HEARTBEAT_FREQUENCY_THRESHOLD) return; // Filter peaks too close
if (count > HEARTBEAT_REPEAT_THRESHOLD) {         // Set new heartbeat rate
                                                  // if it repeats X times
```

Making Sense out of your Sensor Data

- Thresholds
- **Time**
- Statistics
  - Mean, Median, Mode, Range, etc
  - Moving/Weighted Average

```
// Calculate heartbeat rate
long now = System.currentTimeMillis();
long diff = now - mLastHeartBeatTime;
int heartbeat = (int) ((1000.0 / (now - mLastHeartBeatTime)) * 60)
```

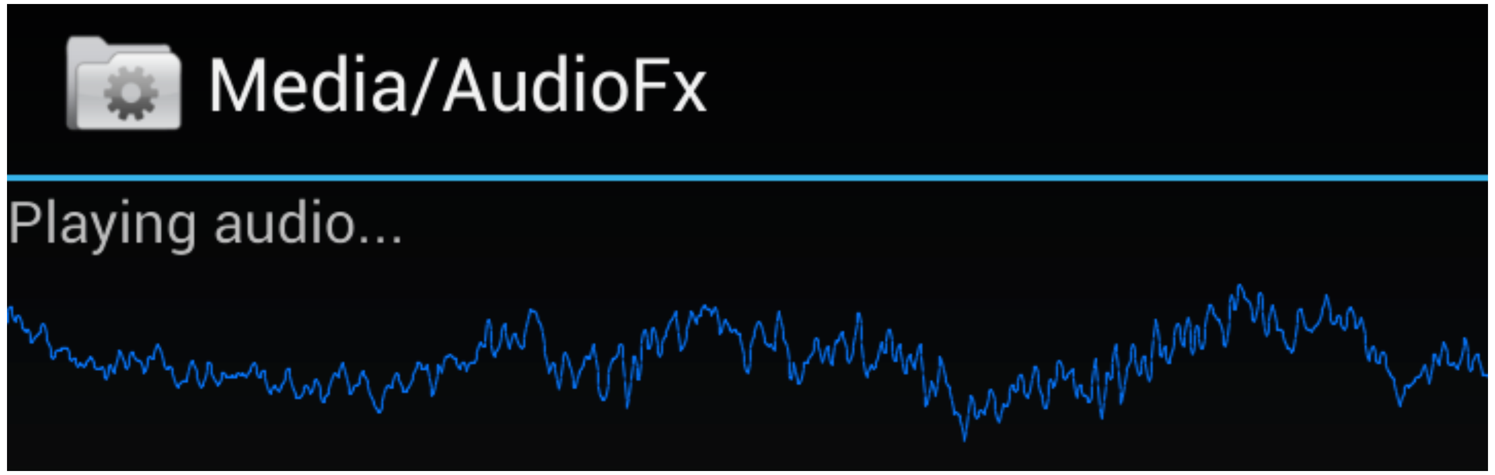
Making Sense out of your Sensor Data

- Thresholds
- Time
- **Statistics**
    - **Mean, Median, Mode, Range, etc**
    - **Weighted/Moving Average**

```
Statistics are your friend
```

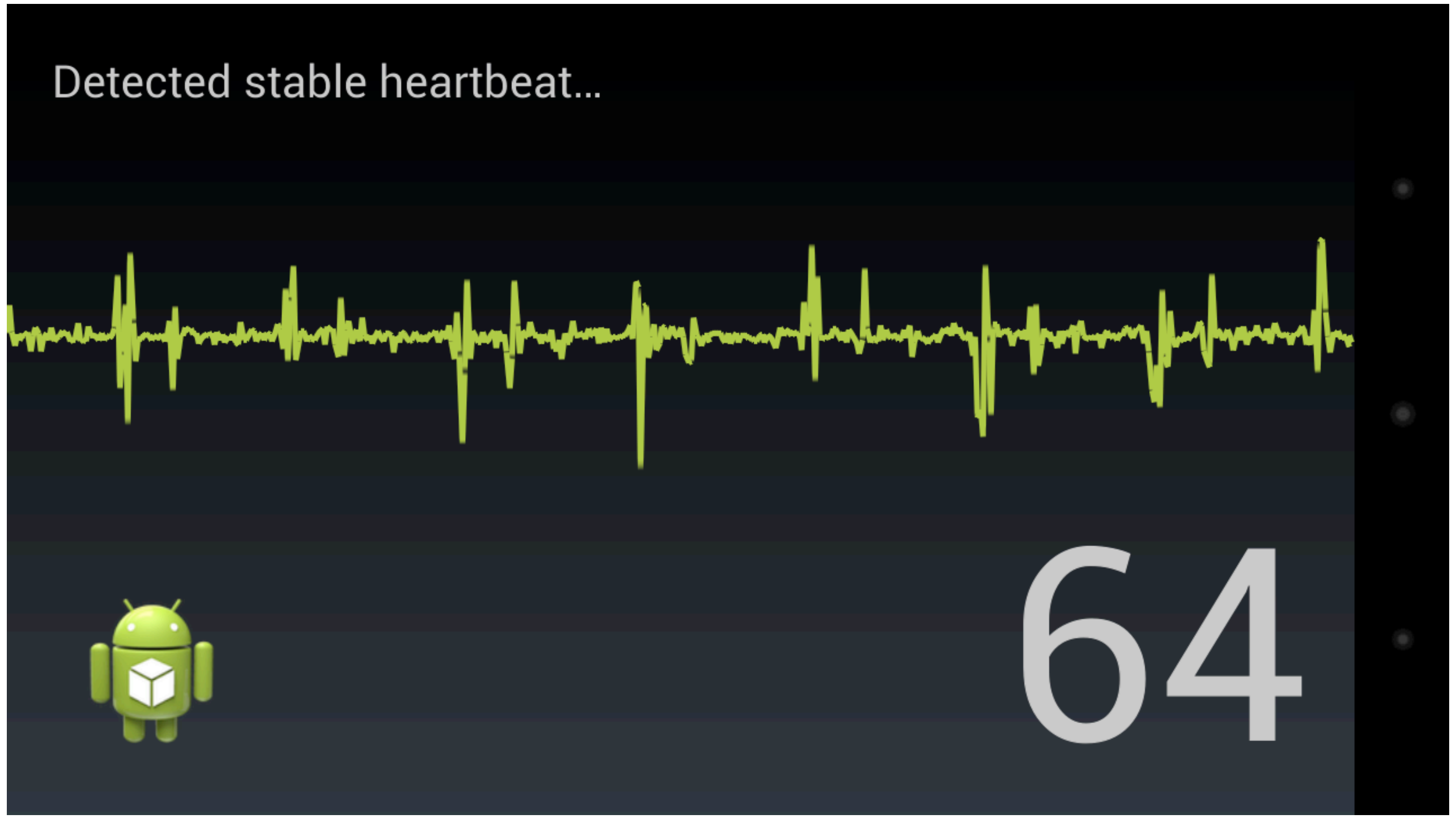
Visualization

- android.media.audiofx - Visualizer
- Custom visualizer
  - Down-sample data for display
  - Use a circular buffer to store the display data to achieve scrolling effect



Media/AudioFx

Playing audio...



Detected stable heartbeat...

64

Thank You!

http://developer.android.com/training

+Android Developers on Google+

+Tim Bray

+Tony Chan

+Ankur Kotwal

View this session at
https://developers.google.com/events/io/sessions/gooio2012/108/

Google
Developers