App Development for Smart Devices

CS 495/595 - Fall 2012

Lec #5: Android Sensors

Tamer Nadeem
Dept. of Computer Science

Objective

- **Working in Background**
  - Sensor Manager
  - Sensor Types
  - Examples

What is a Sensor

- A converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument …
- Sensors have been used in cellphones since they were invented …
  - Microphone, number keys
  - Instead of carrying around 10 separate devices, now you just need 1

Android Sensors

- MIC
- Camera
- Temperature
- Location (GPS or Network)
- Orientation
- Accelerometer
- Proximity
- Pressure
- Light

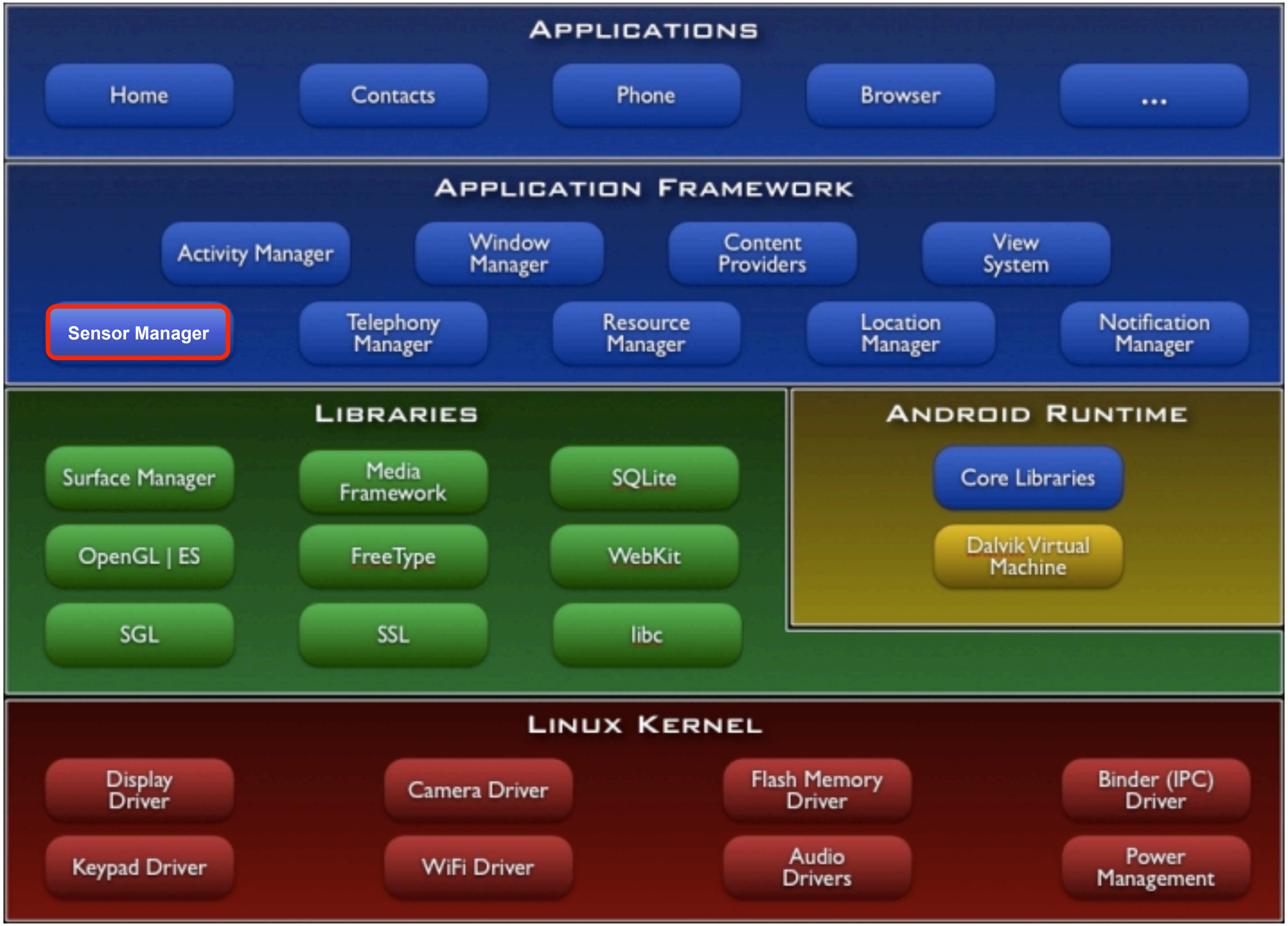
Android.hardware Package

Support for Hardware classes with some interfaces

- **Camera:** used to set image capture settings, start/stop preview, snap pictures, and retrieve frames for encoding for video.
- **Camera.CameraInfo:** Information about a camera
- **Camera.Parameters:** Camera service settings.
- **Camera.Size:** Image size (width and height dimensions).
- **GeomagneticField:** Estimate magnetic field at a given point on Earth and compute the magnetic declination from true north.
- **Sensor:** Class representing a sensor.
- **SensorEvent:** Represents a Sensor event and holds information such as sensor's type, time-stamp, accuracy and sensor's data.
- **SensorManager:** SensorManager lets you access the device's sensors.

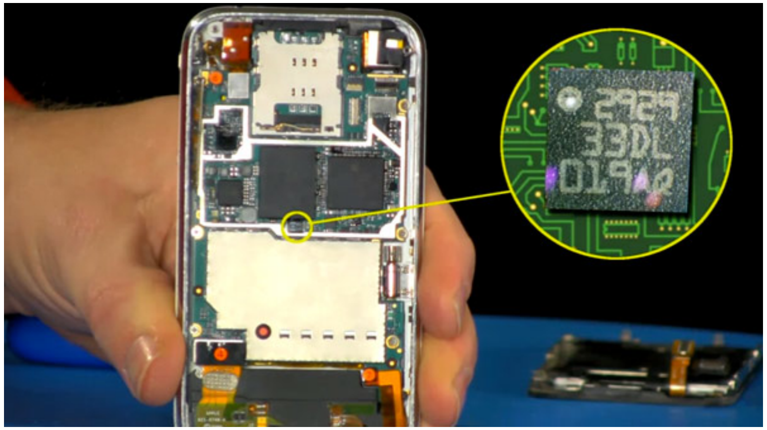
http://developer.android.com/reference/android/hardware/package-summary.html

Android Software Stack

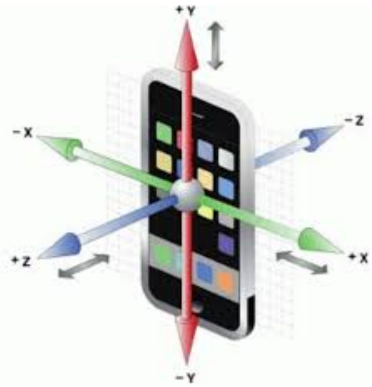


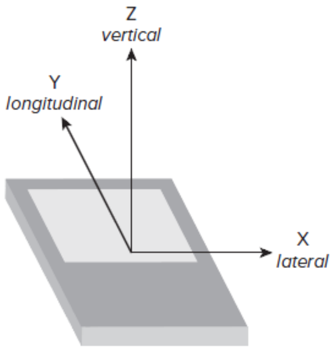
Types of Sensors

- **Not every device has every kind of sensor**
- **Constants from Sensor class**
- **Sensor.TYPE_ACCELEROMETER**
  - hardware
  - acceleration force in m/s^2
  - x, y, z axis
  - includes gravity

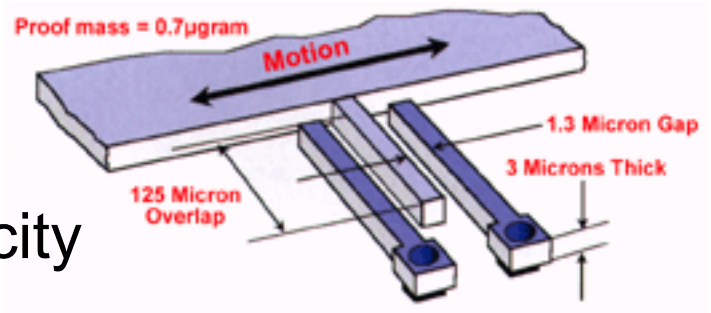


Accelerometer Sensor





- Acceleration is defined as the rate of change of velocity.
- Accelerometers measure how quickly the speed of the device is changing in a given direction.
- Detect movement and corresponding speed's rate of change.
- Accelerometers do not measure velocity



- How does it work?
  - The "proof mass" shown above is allowed to move in a plane.
  - The attached fingers form a capacitor with the two plates around it.
  - The rate of change of the capacitance is measured and translated into an acceleration

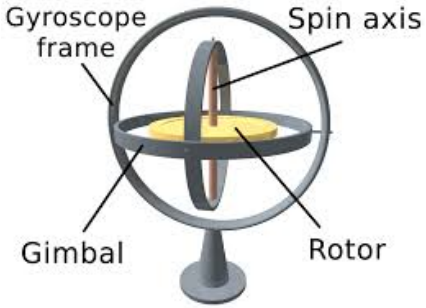
Types of Sensors

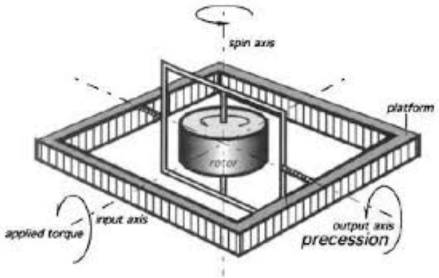
- **Sensor.TYPE_AMBIENT_TEMPERATURE**
  - hardware
  - "room" temperature in degrees Celsius
  - no such sensor on dev phones

- **Sensor.TYPE_GRAVITY**
  - software or hardware
  - just gravity
  - if phone at rest same as TYPE_ACCELEROMETER

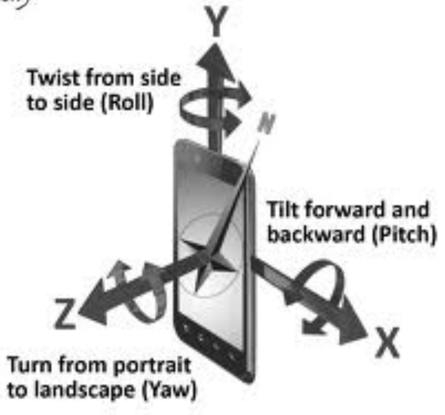
Types of Sensors

- **Sensor.TYPE_GYROSCOPE**
  - hardware
  - measure device's rate of rotation in radians / second around 3 axis





- **Sensor.TYPE_LIGHT**
  - hardware
  - light level in lx,
  - lux is SI measure illuminance in luminous flux per unit area


Y
Twist from side to side (Roll)
N
Tilt forward and backward (Pitch)
Z
X
Turn from portrait to landscape (Yaw)

Types of Sensors

- **Sensor.TYPE_LINEAR_ACCELERATION**
  - software or hardware
  - measure acceleration force applied to device in three axes excluding the force of gravity

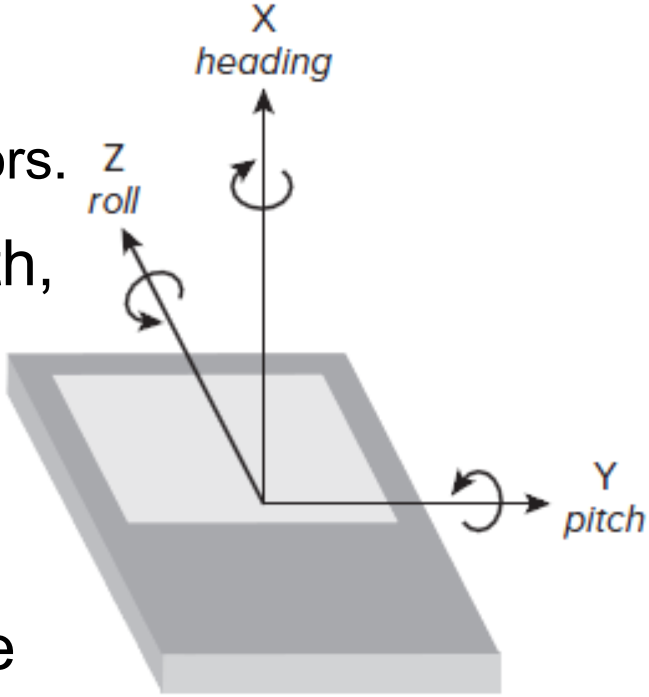
- **Sensor.TYPE_MAGNETC_FIELD**
  - hardware
  - ambient geomagnetic field in all three axes
  - uT micro Teslas

Types of Sensors

- **Sensor.TYPE_ORIENTATION [deprecated]**
  - software
  - measure of degrees of rotation a device makes around all three axes
- **Sensor.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

Orientation Sensor

- Orientation Sensor is a combination of the magnetic field Sensors, which function as an electronic compass, and accelerometers, which determine the pitch and roll.
- Two alternatives for determining the device orientation.
  - Query the orientation Sensor directly
  - Derive the orientation using the accelerometers and magnetic field Sensors.



- **x-axis (azimuth)** 0/360 degrees is north, 90 east, 180 south, and 270 west
- **y-axis (pitch)** 0 flat on its back, -90 standing upright.
- **z-axis (roll)** 0 flat on its back, -90 is the screen facing left

Types of Sensors

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

- **Sensor.TYPE_RELATIVE_HUMIDITY**
  - ambient humidity in percent (0 to 100)

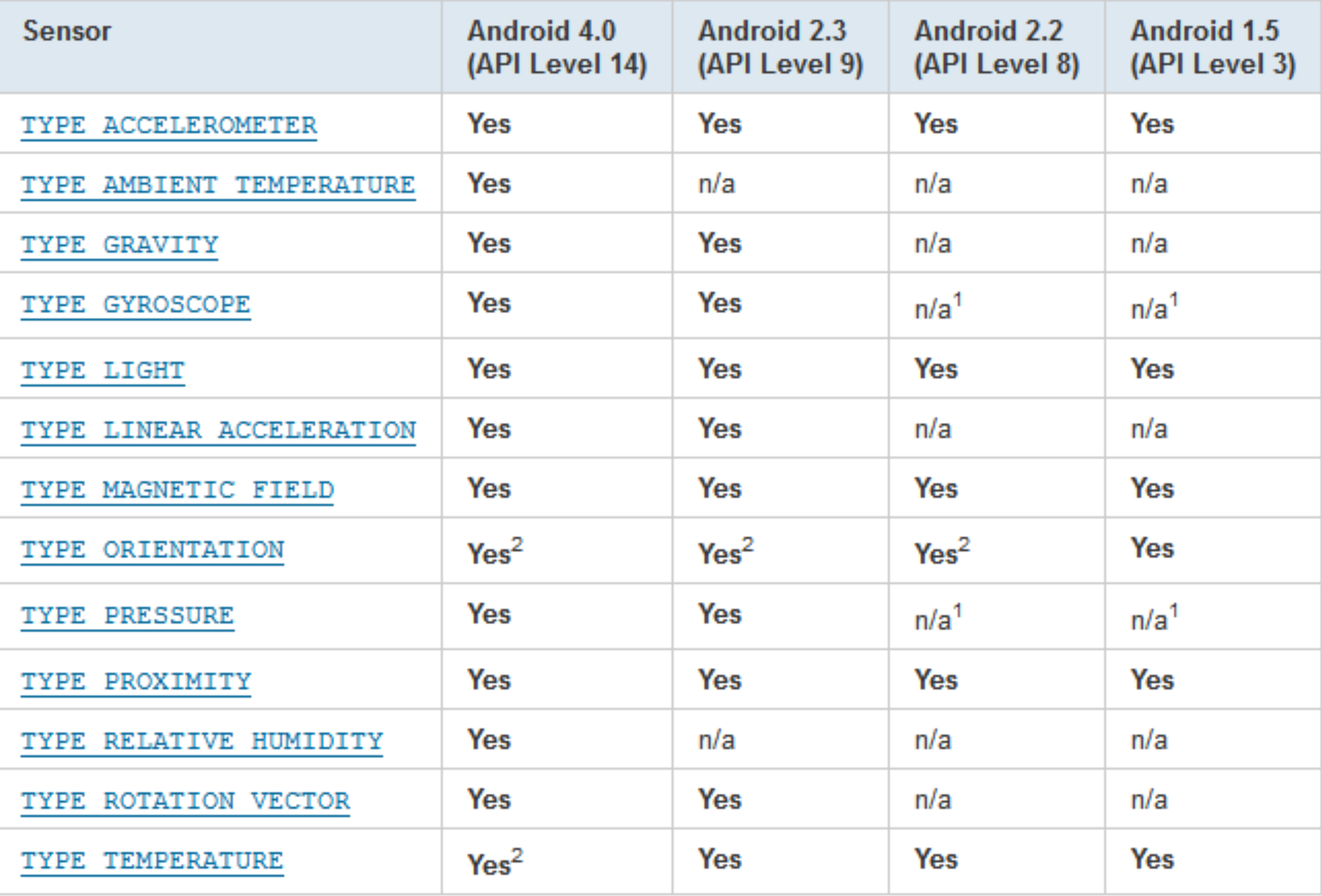
Types of Sensors

- **Sensor.TYPE_ROTATION_VECTOR**
  - hardware or software
  - orientation of device, three elements of the device's rotation vector

- **Sensor.TYPE_TEMPERATURE**
  - hardware
  - temperature of the device in degrees Celsius

http://developer.android.com/reference/android/hardware/Sensor.html

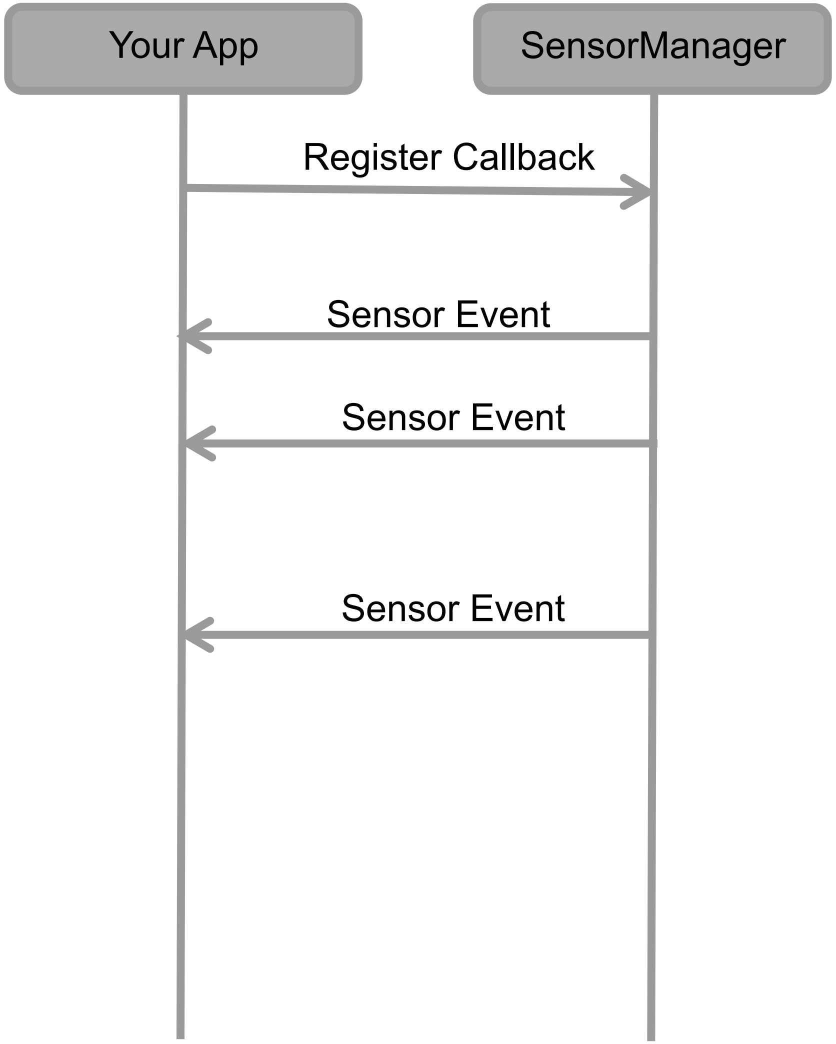
Availability of Sensors



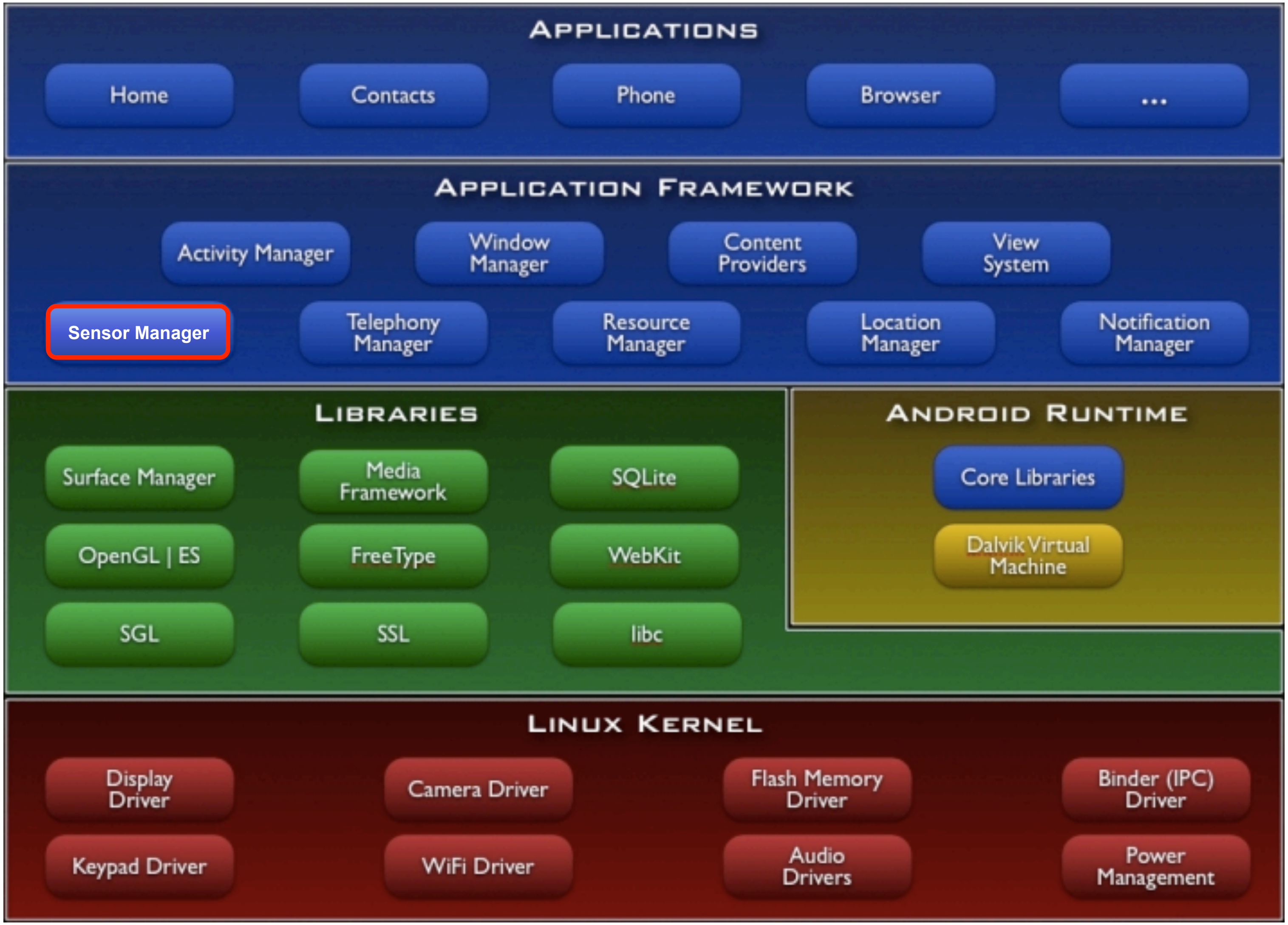
Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a[1]	n/a[1]
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes[2]	Yes[2]	Yes[2]	Yes
TYPE_PRESSURE	Yes	Yes	n/a[1]	n/a[1]
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes[2]	Yes	Yes	Yes

Async Callbacks

- **Android's sensors are controlled by external services and only send events when they choose to**
- **An app must register a callback to be notified of a sensor event**
- **Each sensor has a related XXXListener interface that your callback must implement**
  - E.g. LocationListener



Android Software Stack



Sensing & Sensor Manager

- Device specific

- ServiceManager provides access to Sensor Manager Service
- Use Context.getSystemService(SENSOR_SERVICE) for access

```
String service_name = Context.SENSOR_SERVICE;
 SensorManager sensorManager = (SensorManager) getSystemService(service_name)
```

- Note that you should disable any sensors you don't need, especially when activity paused.
  - System will not disable automatically when screen turns off
  - Battery will drain quickly otherwise.

SensorManager's Methods

- **Sensor getDefaultSensor(int type)** Use this method to get the default sensor for a given type
- **List<Sensor> getSensorList(int type)** Use this method to get the list of available sensors of a certain type
- **boolean registerListener(SensorEventListener listener, Sensor sensor, int rate)** Registers a SensorEventListener for the given sensor.
- **void unregisterListener(SensorEventListener listener, Sensor sensor)** Unregisters a listener for the sensors with which it is registered.

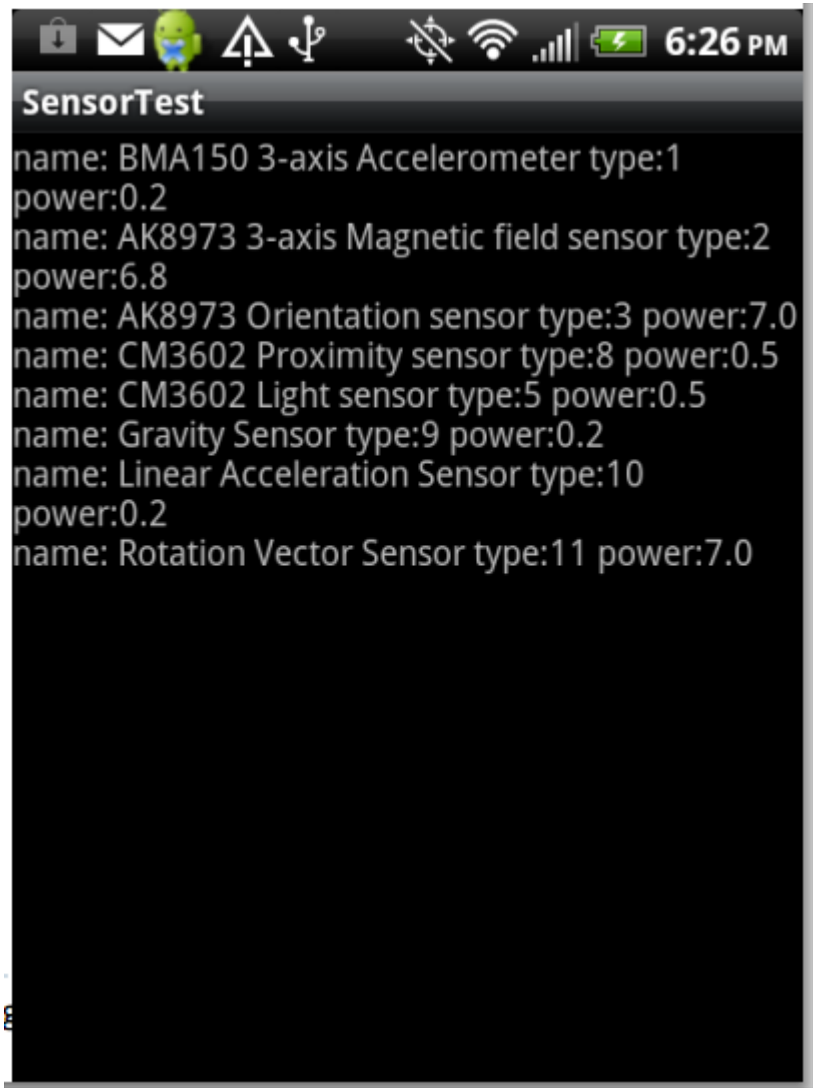
http://developer.android.com/reference/android/hardware/SensorManager.html

Sensor's Methods

- **public float getMaximumRange () - maximum range of the sensor in the sensor's unit.**
- **public int getMinDelay () - the minimum delay allowed between two events in microsecond or zero if this sensor only returns a value when the data it's measuring changes.**
- **public String getName () - name string of the sensor.**
- **public float getPower () - the power in mA used by this sensor while in use.**
- **public float getResolution () - resolution of the sensor in the sensor's unit.**

getPower() Methods

- The device’s battery has a 1500 mA
- Under normal use, the battery lasts 10hours.
- If we use orientation, rotation vector, & magnetic field sensors
- How long would it last now?



```
name: BMA150 3-axis Accelerometer type:1
power:0.2
name: AK8973 3-axis Magnetic field sensor type:2
power:6.8
name: AK8973 Orientation sensor type:3 power:7.0
name: CM3602 Proximity sensor type:8 power:0.5
name: CM3602 Light sensor type:5 power:0.5
name: Gravity Sensor type:9 power:0.2
name: Linear Acceleration Sensor type:10
power:0.2
name: Rotation Vector Sensor type:11 power:7.0
```

Checking for Sensors

```
Sensor defaultGyroscope = sensorManager.getDefaultSensor
          (Sensor.TYPE_GYROSCOPE);
 //(Returns null if none)


//Or, get a list of all sensors of a type:
List<Sensor> pressureSensors =
          sensorManager.getSensorList(Sensor.TYPE_PRESSURE);


//Or, get a list of all sensors of a type:
List<Sensor> allSensors =
          sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Listening for Sensors

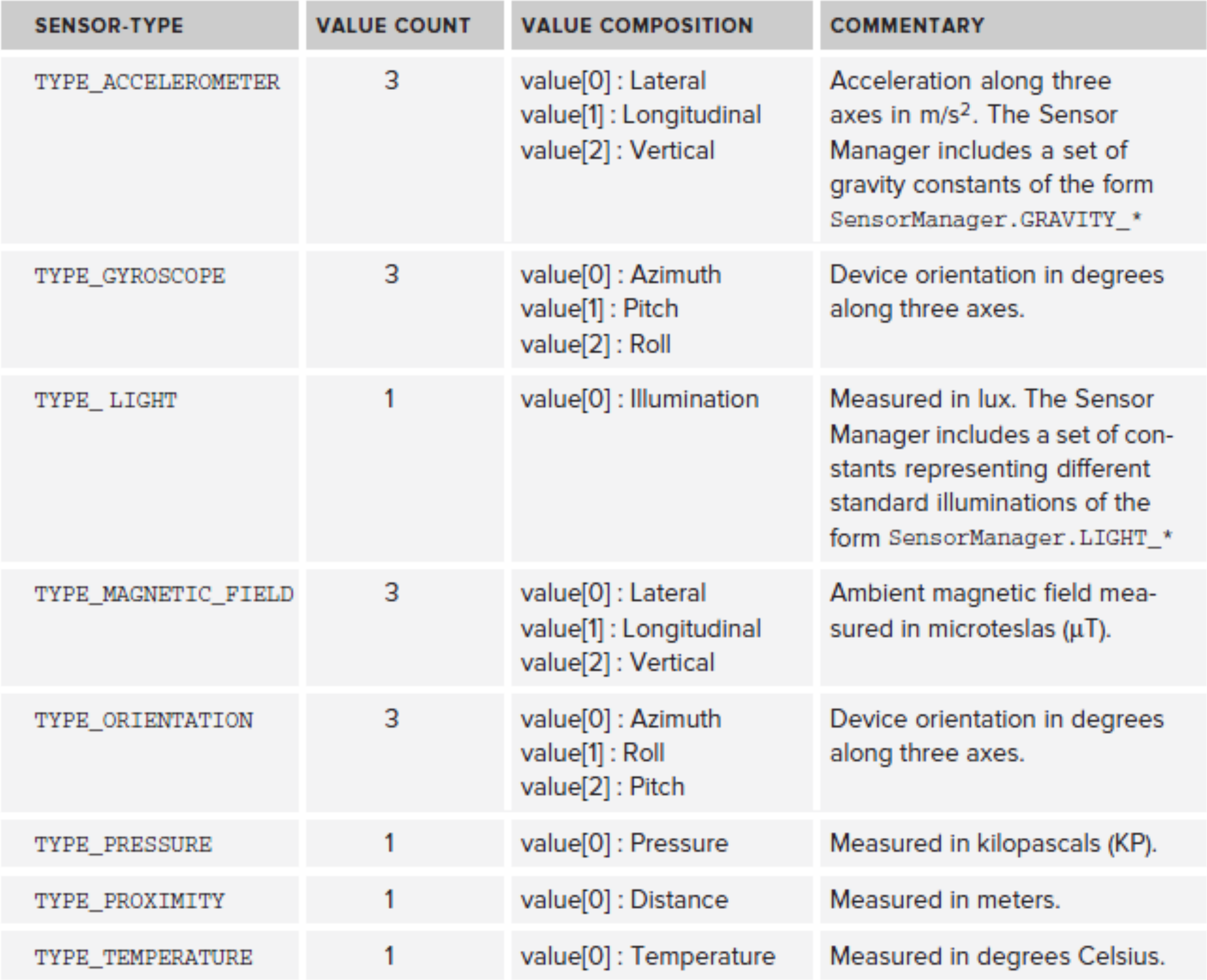
```
final SensorEventListener mySensorEventListener = new SensorEventListener() {
    public void onSensorChanged(SensorEvent sensorEvent) {
        // TODO Monitor Sensor changes.
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO React to a change in Sensor accuracy.
    }
}
```

- Accuracy:
  - SensorManager.SENSOR_STATUS_ACCURACY_LOW
  - SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM
  - SensorManager.SENSOR_STATUS_ACCURACY_HIGH
  - SensorManager.SENSOR_STATUS_ACCURACY_UNRELIABL

SensorEvent

- SensorEvent parameter in the onSensorChanged method includes four properties used to describe a Sensor event:
  - ***sensor:*** The sensor that triggered the event.
  - ***accuracy:*** The accuracy of the Sensor when the event occurred.
  - ***values:*** A float array that contains the new value(s) detected.
  - ***timestamp:*** The time in nanosecond at which the event occurred.

Sensor Values



SENSOR-TYPE	VALUE COUNT	VALUE COMPOSITION	COMMENTARY
TYPE_ACCELEROMETER	3	value[0] : Lateral value[1] : Longitudinal value[2] : Vertical	Acceleration along three axes in m/s^2. The Sensor Manager includes a set of gravity constants of the form `SensorManager.GRAVITY_*`
TYPE_GYROSCOPE	3	value[0] : Azimuth value[1] : Pitch value[2] : Roll	Device orientation in degrees along three axes.
TYPE_ LIGHT	1	value[0] : Illumination	Measured in lux. The Sensor Manager includes a set of constants representing different standard illuminations of the form `SensorManager.LIGHT_*`
TYPE_MAGNETIC_FIELD	3	value[0] : Lateral value[1] : Longitudinal value[2] : Vertical	Ambient magnetic field measured in microteslas (μT).
TYPE_ORIENTATION	3	value[0] : Azimuth value[1] : Roll value[2] : Pitch	Device orientation in degrees along three axes.
TYPE_PRESSURE	1	value[0] : Pressure	Measured in kilopascals (KP).
TYPE_PROXIMITY	1	value[0] : Distance	Measured in meters.
TYPE_TEMPERATURE	1	value[0] : Temperature	Measured in degrees Celsius.

Register

```
// Usually in onResume
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
sensorManager.registerListener(mySensorEventListener, sensor,
          SensorManager.SENSOR_DELAY_NORMAL);

// Usually in onPause
sensorManager.unregisterListener(mySensorEventListener);
```

- Update Rate:
  - SensorManager.SENSOR_DELAY_FASTEST
  - SensorManager.SENSOR_DELAY_GAME
  - SensorManager.SENSOR_DELAY_NORMAL
  - SensorManager.SENSOR_DELAY_UI

Accelerometer, Compass, & Orientation

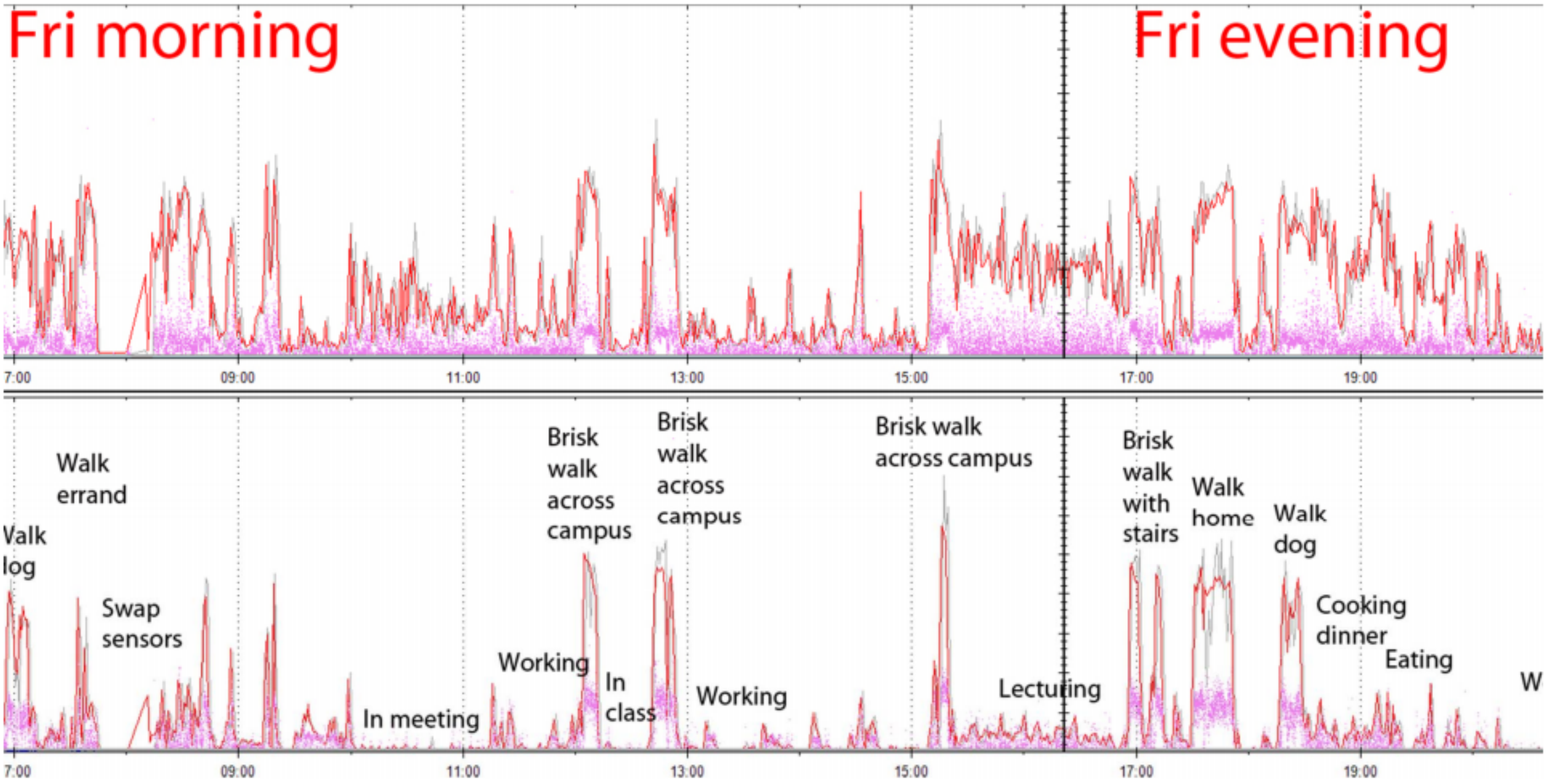
- Allow you to:
  - ➤ Determine the current device orientation
  - ➤ Monitor and track changes in orientation
  - ➤ Know which direction the user is facing
  - ➤ Monitor acceleration—changes in movement rate—in any direction

- Open possibilities for your applications:
  - ➤ Use these with a map, camera, and location-based services to create augmented reality interfaces.
  - ➤ Create user interface that adjust dynamically to suit device orientation.
  - ➤ Monitor rapid acceleration to detect if a device is dropped or thrown.
  - ➤ Measure movement or vibration (e.g., locking application).
  - ➤ User interface controls that use physical gestures and movement.

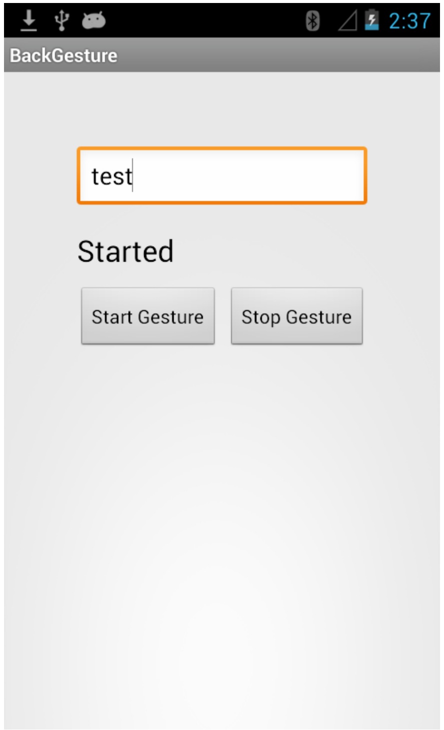
Listener for Changes (Accel)

```
public void setupSensorListener() {
     SensorManager sm =
            (SensorManager)getSystemService(Context.SENSOR_SERVICE);
      int sensorType = Sensor.TYPE_ACCELEROMETER;
      sm.registerListener(mySensorEventListener,
            sm.getDefaultSensor(sensorType),
            SensorManager.SENSOR_DELAY_NORMAL);
}

final SensorEventListener mySensorEventListener = new SensorEventListener() {
    public void onSensorChanged(SensorEvent sensorEvent) {
          if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
              float xAxis_lateralA = sensorEvent.values[0];
              float yAxis_longitudinalA = sensorEvent.values[1];
              float zAxis_verticalA = sensorEvent.values[2];

              // TODO apply the acceleration changes to your application.
          }
    }
};
```

Accelerometer Data

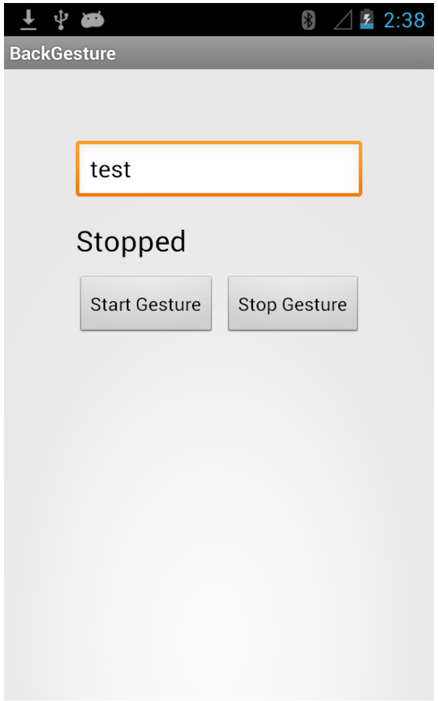


Collecting Sensor Data



test

Started



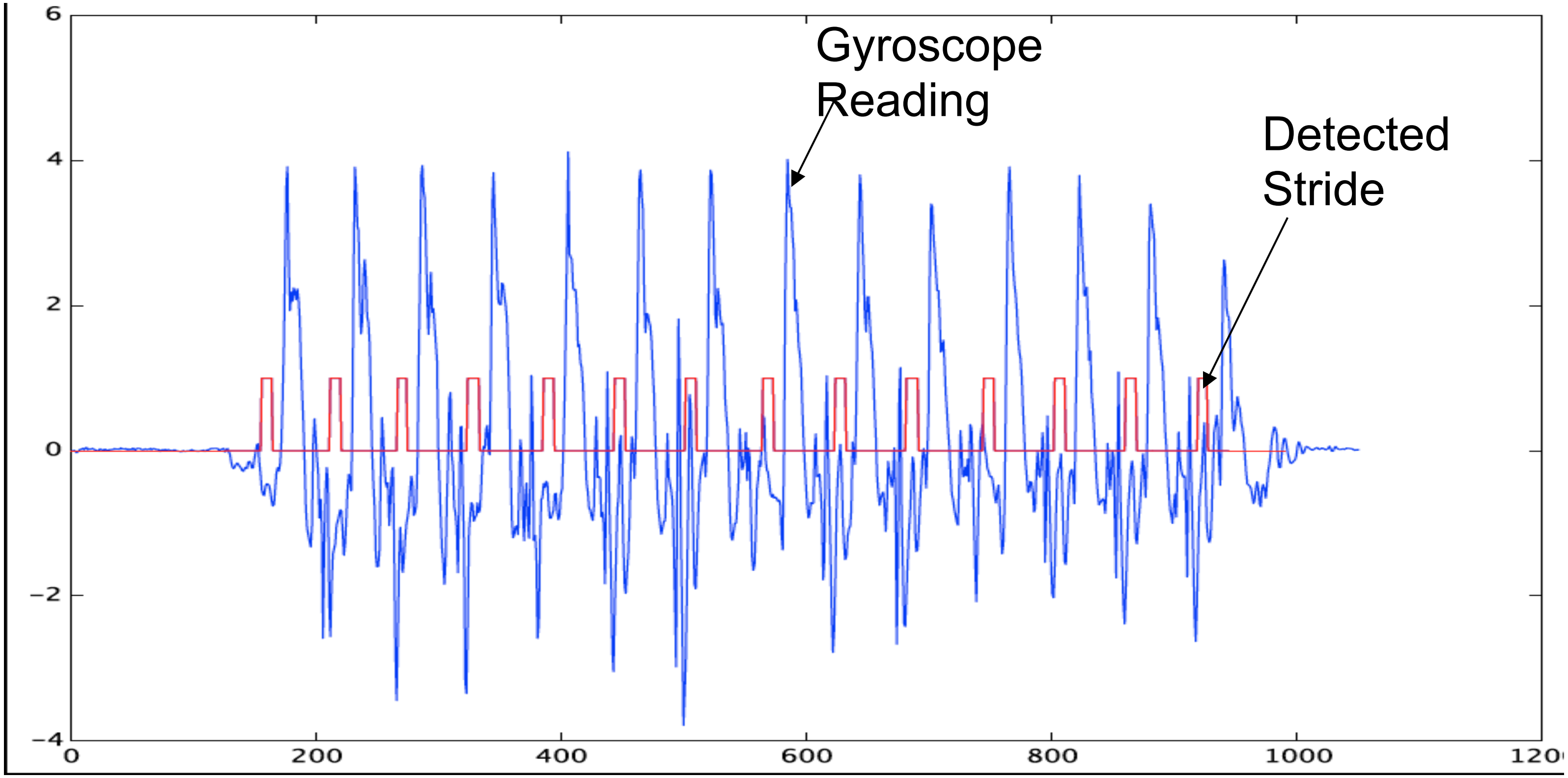
test

Stopped

How we have collected the data

Walking Experiment

- We use the sensors (Accelerometer+Gyroscope) reading to count the stride.

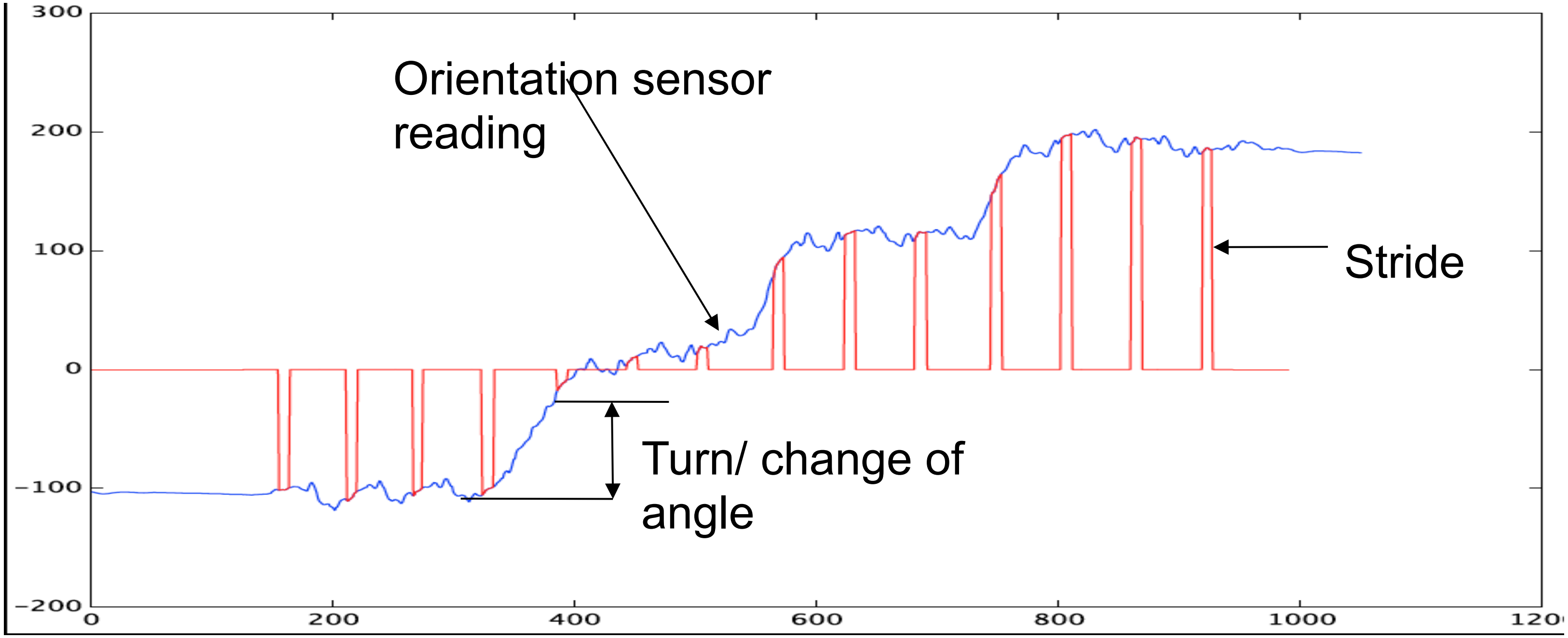


Gyroscope Reading

Detected Stride

Walking Experiment

- We use the orientation and magnetic field sensor to detect the turns.
- We limit each turn to fixed angles(i.e. 0,45,90,135,180,225,270 degree) for simplicity.



Orientation sensor reading

Stride

Turn/ change of angle

Listener for Changes (Orientation)

```
public void setupSensorListener() {
    SensorManager sm =
        (SensorManager)getSystemService(Context.SENSOR_SERVICE);
    int sensorType = Sensor.TYPE_ORIENTATION;
    sm.registerListener(mySensorEventListener,
        sm.getDefaultSensor(sensorType),
        SensorManager.SENSOR_DELAY_NORMAL);
}

final SensorEventListener mySensorEventListener = new SensorEventListener() {
    public void onSensorChanged(SensorEvent sensorEvent) {
        if (sensorEvent.sensor.getType() == Sensor.TYPE_ORIENTATION) {
            float headingAngle = sensorEvent.values[0];
            float pitchAngle = sensorEvent.values[1];
            float rollAngle = sensorEvent.values[2];

            // TODO apply the orientation changes to your application.
        }
    }
};
```

Controlling Vibration

- Vibration is an excellent way to provide haptic user feedback.
- Applications needs the VIBRATE permission in application manifest:

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

- Example:

```
String vibratorService = Context.VIBRATOR_SERVICE;
Vibrator vibrator = (Vibrator)getSystemService(vibratorService);

long[] pattern = {1000, 2000, 4000, 8000, 16000 };
vibrator.vibrate(pattern, 0); // Execute vibration pattern.
vibrator.vibrate(1000); // Vibrate for 1 second.
```

Questions?

To DO

- Example 1 (in slides)

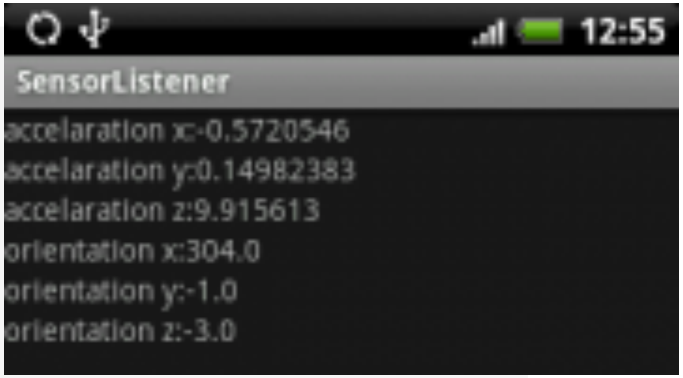
- Example 2 (in slides)

- Example 3 (in slides)

- **Assignment #3: Assignment Tracker App v2.0**

Example 1. Displaying Accelerometer and Orientation Data

- Create an activity with accelerometer and orientation data.



SensorListener

```
accelaration x:-0.5720546
accelaration y:0.14982383
accelaration z:9.915613
orientation x:304.0
orientation y:-1.0
orientation z:-3.0
```

```
package com.exercise.AndroidSensorList;

import android.app.ListActivity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class SensorTest extends Activity implements SensorEventListener {

    SensorManager sensorManager = null;

    //for accelerometer values
    TextView outputX;
    TextView outputY;
    TextView outputZ;

    //for orientation values
    TextView outputX2;
    TextView outputY2;
    TextView outputZ2;
```

Example 1. Displaying Accelerometer and Orientation Data

```
 @Override
 public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
     setContentView(R.layout.main);

     //just some textviews, for data output
     outputX = (TextView) findViewById(R.id.TextView01);
     outputY = (TextView) findViewById(R.id.TextView02);
     outputZ = (TextView) findViewById(R.id.TextView03);

     outputX2 = (TextView) findViewById(R.id.TextView04);
     outputY2 = (TextView) findViewById(R.id.TextView05);
     outputZ2 = (TextView) findViewById(R.id.TextView06);
}
```

Example 1. Displaying Accelerometer and Orientation Data

```
@Override
 protected void onResume() {
     super.onResume();
     sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                sensorManager.SENSOR_DELAY_GAME);
     sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                sensorManager.SENSOR_DELAY_GAME);
 }

@Override
 protected void onStop() {
      super.onStop();
      sensorManager.unregisterListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER));
      sensorManager.unregisterListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION));
 }
```

Example 1. Displaying Accelerometer and Orientation Data

```
public void onSensorChanged(SensorEvent event) {
    synchronized (this) {
        switch (event.sensor.getType()){
            case Sensor.TYPE_ACCELEROMETER:
                outputX.setText("x:"+Float.toString(event.values[0]));
                outputY.setText("y:"+Float.toString(event.values[1]));
                outputZ.setText("z:"+Float.toString(event.values[2]));
                break;
            case Sensor.TYPE_ORIENTATION:
                outputX2.setText("x:"+Float.toString(event.values[0]));
                outputY2.setText("y:"+Float.toString(event.values[1]));
                outputZ2.setText("z:"+Float.toString(event.values[2]));
                break;
        }
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {}
}
```

Example 2. Creating a G-Forceometer

- Create a simple device to measure g-force using the accelerometers to determine the current force being exerted on the device.
- Forceometer Activity & Layout (main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical“  android:layout_width="fill_parent"
            android:layout_height="fill_parent">
    <TextView android:id="@+id/acceleration“  android:gravity="center"
            android:layout_width="fill_parent“  android:layout_height="wrap_content"
            android:textStyle="bold“   android:textSize="32sp"
            android:text="CENTER“   android:editable="false"
            android:singleLine="true“  android:layout_margin="10px"
    />
   <TextView android:id="@+id/maxAcceleration“   android:gravity="center"
            android:layout_width="fill_parent“   android:layout_height="wrap_content"
            android:textStyle="bold“   android:textSize="40sp"
            android:text="CENTER“   android:editable="false"
            android:singleLine="true“   android:layout_margin="10px"
    />
</LinearLayout>
```

Example 2. Creating a G-Forceometer

- Within Forceometer Activity class, create instance variables

```
SensorManager sensorManager;
TextView accelerationTextView;
TextView maxAccelerationTextView;
float currentAcceleration = 0;
float maxAcceleration = 0;
```

- Within Forceometer Activity class, create a new SensorEventListener implementation

```
private final SensorEventListener sensorEventListener = new SensorEventListener() {
    double calibration = SensorManager.STANDARD_GRAVITY;
    public void onAccuracyChanged(Sensor sensor, int accuracy) { }
    public void onSensorChanged(SensorEvent event) {
        double x = event.values[0];
        double y = event.values[1];
        double z = event.values[2];
        double a = Math.round(Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2) + Math.pow(z, 2)));
        currentAcceleration = Math.abs((float)(a-calibration));
        if (currentAcceleration > maxAcceleration)
            maxAcceleration = currentAcceleration;
    }
};
```

Example 2. Creating a G-Forceometer

- Update the onCreate method to register your new Listener for accelerometer updates using the SensorManager.

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      accelerationTextView = (TextView)findViewById(R.id.acceleration);
      maxAccelerationTextView = (TextView)findViewById(R.id.maxAcceleration);
      sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
      Sensor accelerometer =
                  sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
      sensorManager.registerListener(sensorEventListener, accelerometer,
                  SensorManager.SENSOR_DELAY_FASTEST);
}
```

Example 2. Creating a G-Forceometer

- Create a new updateGUI method that synchronizes with the GUI thread based on a Timer before updating the Text Views

```
private void updateGUI() {
    runOnUiThread(new Runnable() {
        public void run() {
            String currentG = currentAcceleration/SensorManager.STANDARD_GRAVITY + "Gs";
            accelerationTextView.setText(currentG);
            accelerationTextView.invalidate();

            String maxG = maxAcceleration/SensorManager.STANDARD_GRAVITY + "Gs";
            maxAccelerationTextView.setText(maxG);
            maxAccelerationTextView.invalidate();
        }
    });
};
```

Example 2. Creating a G-Forceometer

- Update the onCreate method to start a timer that’s used to update the GUI every 100ms:

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      accelerationTextView = (TextView)findViewById(R.id.acceleration);
      maxAccelerationTextView = (TextView)findViewById(R.id.maxAcceleration);
      sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
      Sensor accelerometer =
                  sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
      sensorManager.registerListener(sensorEventListener, accelerometer,
                  SensorManager.SENSOR_DELAY_FASTEST);

      Timer updateTimer = new Timer("gForceUpdate");
      updateTimer.scheduleAtFixedRate(new TimerTask() {
           public void run() {
                  updateGUI();
           }
      }, 0, 100);
}
```

Example 3. Compass App

```
package com.example.android.apis.graphics;

import android.content.Context;
import android.graphics.*;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Config;
import android.util.Log;
import android.view.View;

public class Compass extends GraphicsActivity {

    private static final String TAG = "Compass";

    private SensorManager mSensorManager;
    private Sensor mSensor;
    private SampleView mView;
    private float[] mValues;
```

Example 3. Compass App

```
    private final SensorEventListener mListener = new SensorEventListener()
{
        public void onSensorChanged(SensorEvent event) {
            if (Config.DEBUG) Log.d(TAG,
                    "sensorChanged (" + event.values[0] + ", " +
event.values[1] + ", " + event.values[2] + ")");
            mValues = event.values;
            if (mView != null) {
                mView.invalidate();
            }
        }

        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
    };

    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        mSensorManager =
(SensorManager)getSystemService(Context.SENSOR_SERVICE);
        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
        mView = new SampleView(this);
        setContentView(mView);
    }
```

Example 3. Compass App

```java
    @Override
    protected void onResume()
    {
        if (Config.DEBUG) Log.d(TAG, "onResume");
        super.onResume();

        mSensorManager.registerListener(mListener, mSensor,
                SensorManager.SENSOR_DELAY_GAME);
    }

    @Override
    protected void onStop()
    {
        if (Config.DEBUG) Log.d(TAG, "onStop");
        mSensorManager.unregisterListener(mListener);
        super.onStop();
    }

    private class SampleView extends View {
        private Paint   mPaint = new Paint();
        private Path    mPath = new Path();
        private boolean mAnimate;

        public SampleView(Context context) {
            super(context);
```

Example 3. Compass App

```
        // Construct a wedge-shaped path
        mPath.moveTo(0, -50);
        mPath.lineTo(-20, 60);
        mPath.lineTo(0, 50);
        mPath.lineTo(20, 60);
        mPath.close();
    }

    @Override protected void onDraw(Canvas canvas) {
        Paint paint = mPaint;

        canvas.drawColor(Color.WHITE);

        paint.setAntiAlias(true);
        paint.setColor(Color.BLACK);
        paint.setStyle(Paint.Style.FILL);

        int w = canvas.getWidth();
        int h = canvas.getHeight();
        int cx = w / 2;
        int cy = h / 2;
```

Example 3. Compass App

```
            canvas.translate(cx, cy);
            if (mValues != null) {
                canvas.rotate(-mValues[0]);
            }
            canvas.drawPath(mPath, mPaint);
        }

        @Override
        protected void onAttachedToWindow() {
            mAnimate = true;
            if (Config.DEBUG) Log.d(TAG, "onAttachedToWindow. mAnimate=" +
mAnimate);
            super.onAttachedToWindow();
        }

        @Override
        protected void onDetachedFromWindow() {
            mAnimate = false;
            if (Config.DEBUG) Log.d(TAG, "onDetachedFromWindow. mAnimate=" +
mAnimate);
            super.onDetachedFromWindow();
        }
    }
}
```