**App Development for Smart Devices**

CS 495/595 - Fall 2012

**Lec #5: Android Sensors**

Tamer Nadeem Dept. of Computer Science

Page 2

**Objective**

**• Working in Background**

• Sensor Manager • Sensor Types

• Examples

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 3

**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

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 4

**Android Sensors**

• MIC

• Camera

• Temperature

• Location (GPS or Network)

• Orientation

• Accelerometer

• Proximity

• Pressure

• Light

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 5

**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

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 6

**Android Software Stack**

**Sensor Manager**

**CS 495/595 - App Development for Smart Devices Fall 2013**

**Types of Sensors**

**• Not every device has every kind of sensor**

**• Constants from Sensor class**

**• Sensor.TYPE\_ACCELEROMETER**

• hardware

• acceleration force in m/s2

• x, y, z axis

• includes gravity

Page 7 Fall 2013

**CS 495/595 - App Development 7**

**for Smart Devices**

**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

Page 8

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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

Page 9 Fall 2013

**CS 495/595 - App Development 9**

**for Smart Devices**

**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

Fall Page 10 2013

**CS 495/595 - App Development for Smart Devices**

**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

Page 11 Fall 2013

**CS 495/595 - App Development 11**

**for Smart Devices**

**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

Page 12 Fall 2013

**CS 495/595 - App Development 12**

**for Smart Devices**

**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

Page 13

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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)

Page 14 Fall 2013

**CS 495/595 - App Development 14**

**for Smart Devices**

**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

Page 15 Fall 2013

**CS 495/595 - App Development 15**

**for Smart Devices**

**Availability of Sensors**

Page 16 Fall 2013

**CS 495/595 - App Development 16**

**for Smart Devices**

Page 17

**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

**CS 495/595 - App Development for Smart Devices Fall 2013**

SensorManager Your App

Register Callback

Sensor Event

Sensor Event

Sensor Event

Page 18

**Android Software Stack**

**Sensor Manager**

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 19

**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.

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 20

**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**

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 21

**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.**

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 22

**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?**

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 23

**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);

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 24

**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

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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.

Page 25

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 26

**Sensor Values**

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 27

**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

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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.

Page 28

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 29

**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. } } };

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 30

**Accelerometer Data**

**CS 495/595 - App Development for Smart Devices Fall 2013**

**Collecting Sensor Data**

**How we have collected the data**

**Walking Experiment**

●

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

**Page 32 CS 495/595 - App Development for Smart Devices**

Fall 2013

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.

**Page 33 CS 495/595 - App Development for Smart Devices**

Fall 2013

Orientation sensor reading

Turn/ change of angle

Stride

Page 34

**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. } } };

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 35

**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.

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 36

**Questions?**

**CS 495/595 - App Development for Smart Devices Fall 2013**

**To DO**

• Example 1 (in slides)

• Example 2 (in slides)

• Example 3 (in slides)

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

Page 37

**CS 495/595 - App Development for Smart Devices Fall 2013**

**Example 1. Displaying Accelerometer and Orientation Data**

• Create an activity with accelerometer and orientation data.

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;

Page 38

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 39

**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); }

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 40

**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)); }

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 41

**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) {} }

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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>

Page 42

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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; } };

Page 43

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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); }

Page 44

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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(); } }); };

Page 45

**CS 495/595 - App Development for Smart Devices Fall 2013**

**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); }**

Page 46

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 47

**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;

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 48

**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); }

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 49

**Example 3. Compass App**

@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);

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 50

**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;

**CS 495/595 - App Development for Smart Devices Fall 2013**

Page 51

**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(); } } }

**CS 495/595 - App Development for Smart Devices Fall 2013**