Android Concepts and Programming

TUTORIAL 1

Kartik Sankaran

kar.kbc@gmail.com

CS4222 Wireless and Sensor Networks

[2nd Semester 2016-17]

23rd January 2017

Agenda

PART 1: Introduction to Sensors

PART 2: Introduction to Android

- Installation of Android SDK
- Simple Android Apps: *HelloWorldApp* and *ButtonApp*

Goal of the Tutorials:

- Help you quickly get started with Android programming.
- Cover concepts not explained well on the Android website.

Tutorial 2: Selected advanced Android concepts

Introduction to Sensors

Introduction to Sensors

1. Sensors available on Android phones
2. Sensor and location APIs
3. Phone and world co-ordinate systems
4. App to detect if the phone is face up
5. Getting the phone's orientation
6. Euler angles and gimbal lock
7. App to detect a simple phone gesture
8. Sampling rate and power
9. Noise and drift

Sensors available on Android phones

1. **Motion Sensors**
   - Accelerometer (also: Gravity, Linear Accl)
   - Gyroscope
   - Rotation Vector
2. **Position Sensors**
   - Magnetic Field
   - Orientation
   - Proximity
3. **Environmental Sensors**
   - Temperature, Light, Pressure, Humidity
4. **Others**
   - GPS, Camera, Microphone, Network

May be physical (hardware) or virtual (software). Some sensors are derived from others.

http://www.youtube.com/watch?v=C7JQ7Rpwn2k

Understanding the physics behind the sensors

These sensors have different APIs

Sensors APIs

```
SensorManager manager = (SensorManager)
    getSystemService( Context.SENSOR_SERVICE );
Sensor light = manager.getDefaultSensor( Sensor.TYPE_LIGHT );

protected void onResume() {
    manager.registerListener( this, light,
                              SensorManager.SENSOR_DELAY_NORMAL);
}
protected void onPause() {
    manager.unregisterListener( this );
}
public void onSensorChanged( SensorEvent event ) {
    float lux = event.values[0];
}
```

Returns null if sensor not present

Sampling Rate

Listener for sensor changes

Location APIs

```
LocationManager manager = (LocationManager)
    getSystemService( Context.LOCATION_SERVICE );
List <String> providers = manager.getProviders( true );

manager.requestLocationUpdates( “gps”,
    TIME_PERIOD,
    MIN_DISTANCE_MOVED,
    this );

manager.removeUpdates( this );

public void onLocationChanged( Location location ) {
    double latitude = location.getLatitude();
}
```

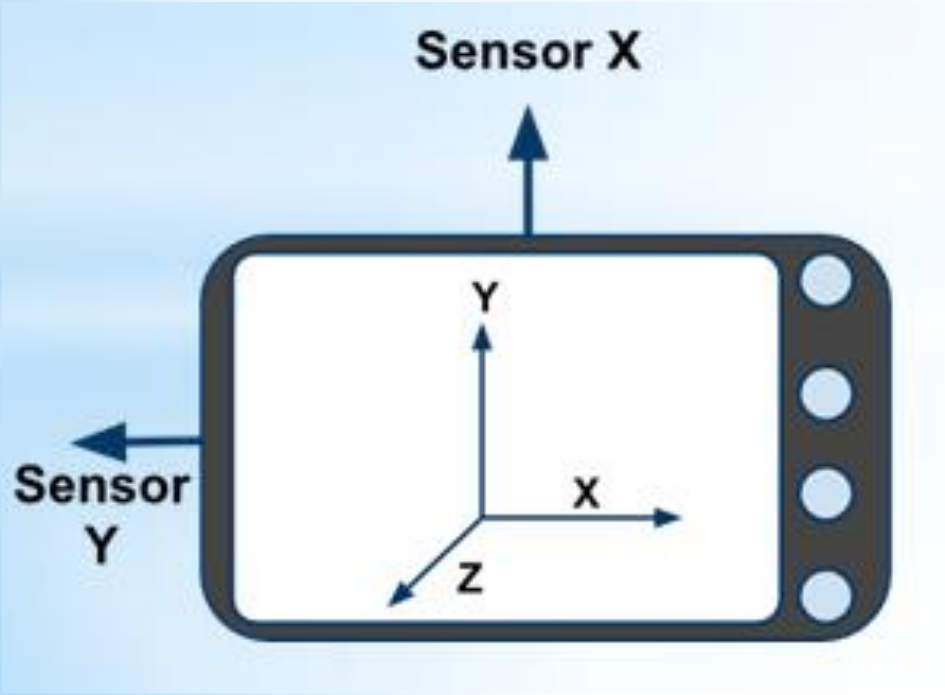
Returns subset of {“gps”, “network”, “passive” }

Sampling Rate
(0 for max frequency)

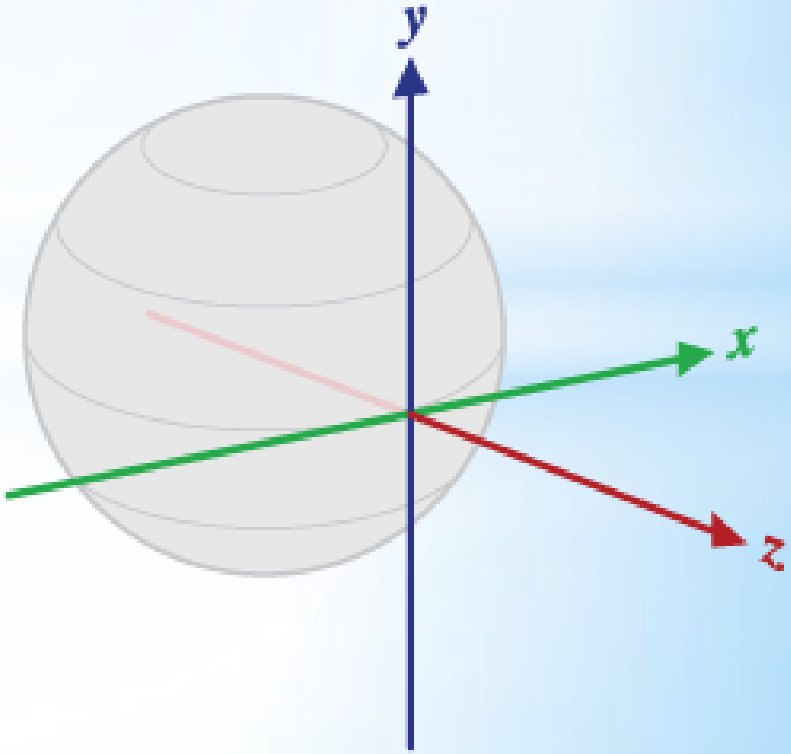
Listener for location changes

Phone and world co-ordinate systems

1. Phone co-ordinate axes (smartphone + tablet) – ***used by sensors***
2. World co-ordinate axes
3. Screen (virtual world) co-ordinate axes



remapCoordinateSystem()
[Used when the screen and phone co-ordinate axes do not match, e.g. during landscape screen rotate]



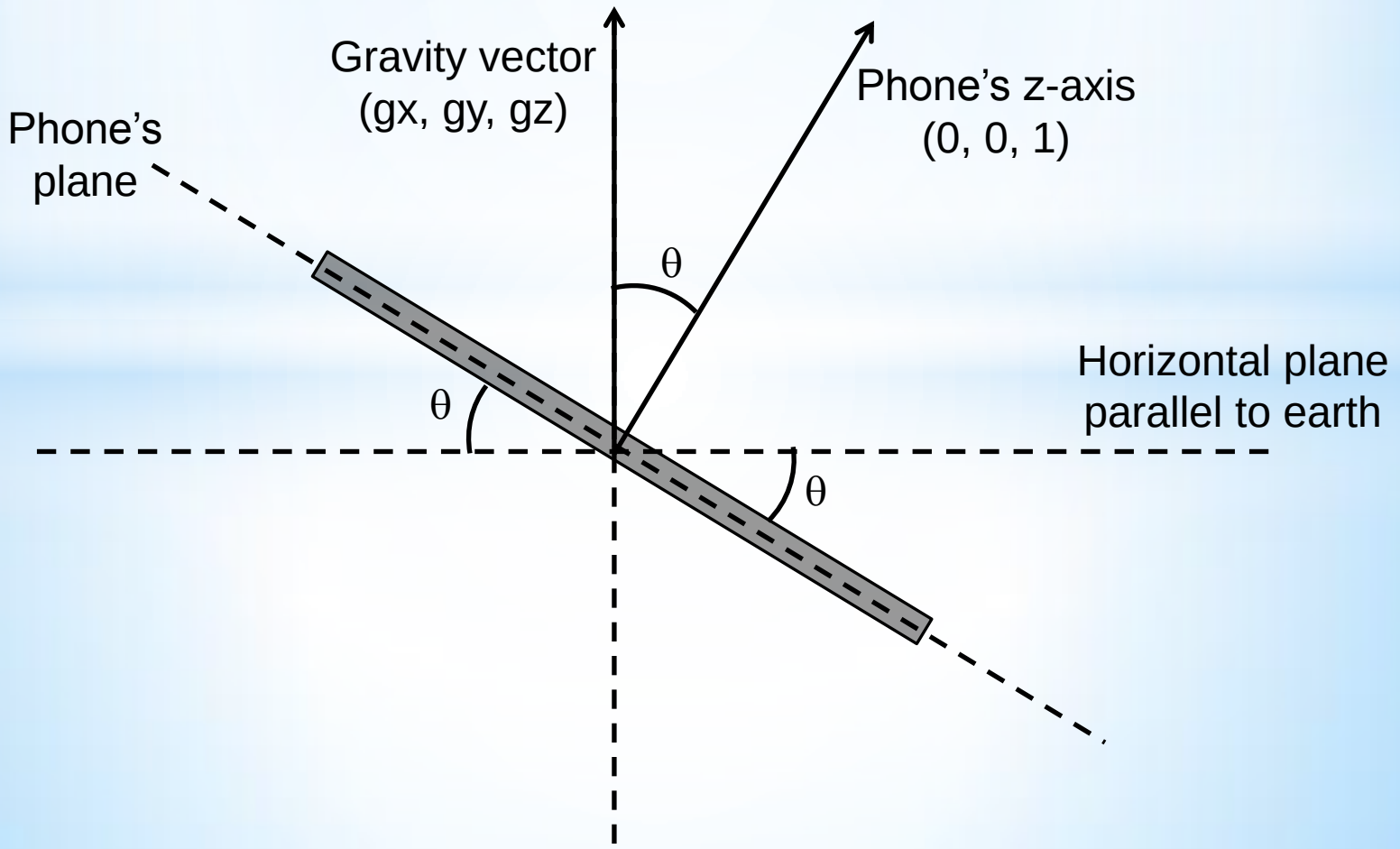
getRotationMatrix()
[Phone → World]

http://android-developers.blogspot.sg/2010/09/one-screen-turn-deserves-another.html

App to detect if the phone is face up

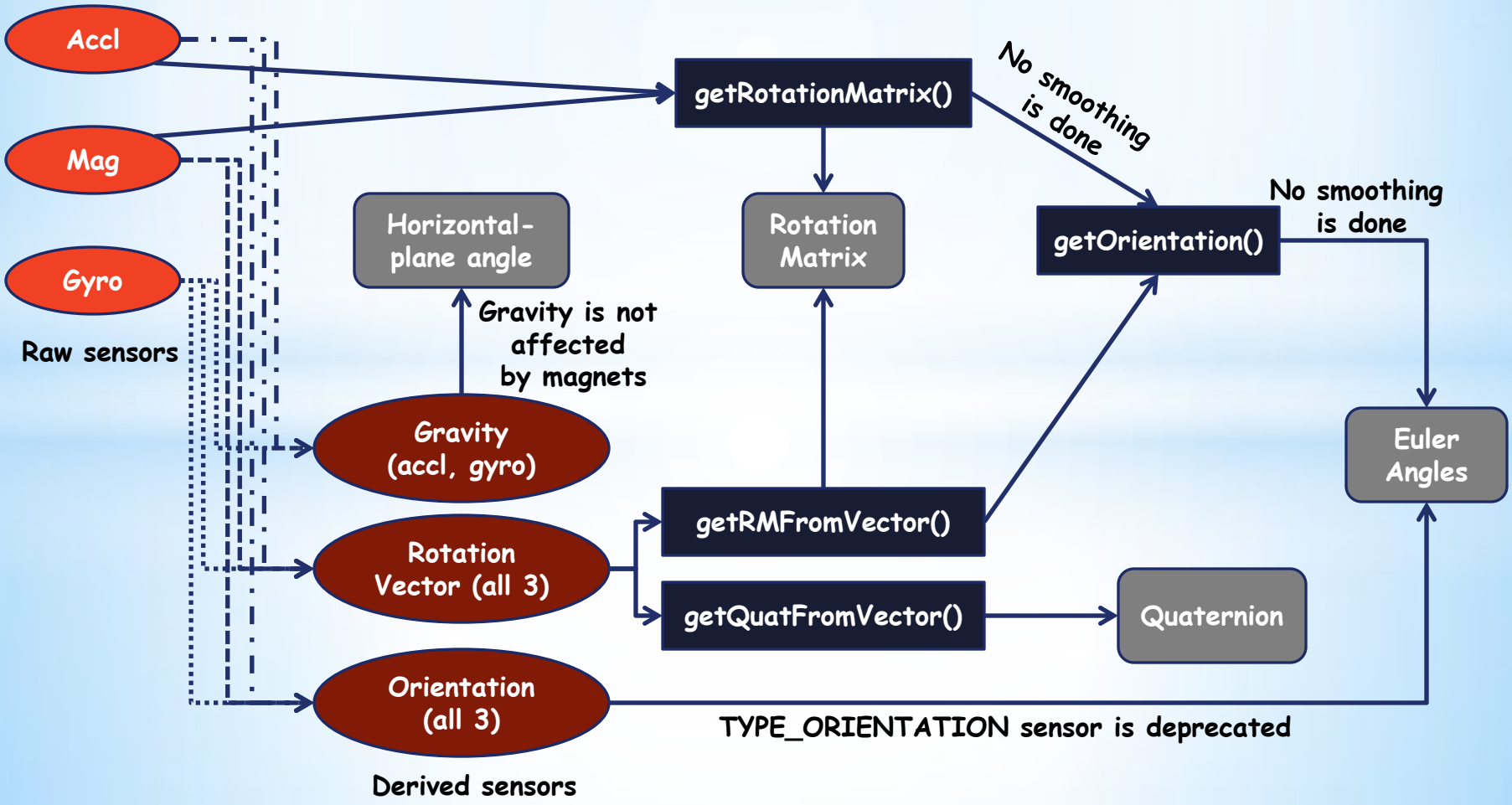
App that displays angle of the phone with the horizontal plane

- **Calculates the angle between the gravity vector and phone's positive z-axis (w.r.t. the phone's co-ordinate system)**



Getting the phone's orientation

Representations: Euler angles, rotation matrix, rotation vector, quaternion



Euler angles and gimbal lock

Euler angles ('intrinsic rotation', i.e. w.r.t. phone's co-ordinate axes):

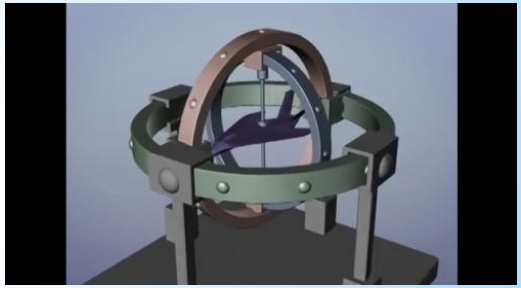
1. **Azimuth/heading/yaw (rotation about negative Z-axis): -180 to +180 deg**
2. **Pitch (rotation about negative X-axis): -90 to +90 deg**
3. **Roll (rotation about the Y-axis): -180 to +180 deg**

Gimbal lock: Two rotations become semantically equivalent at a particular phone position.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}$$



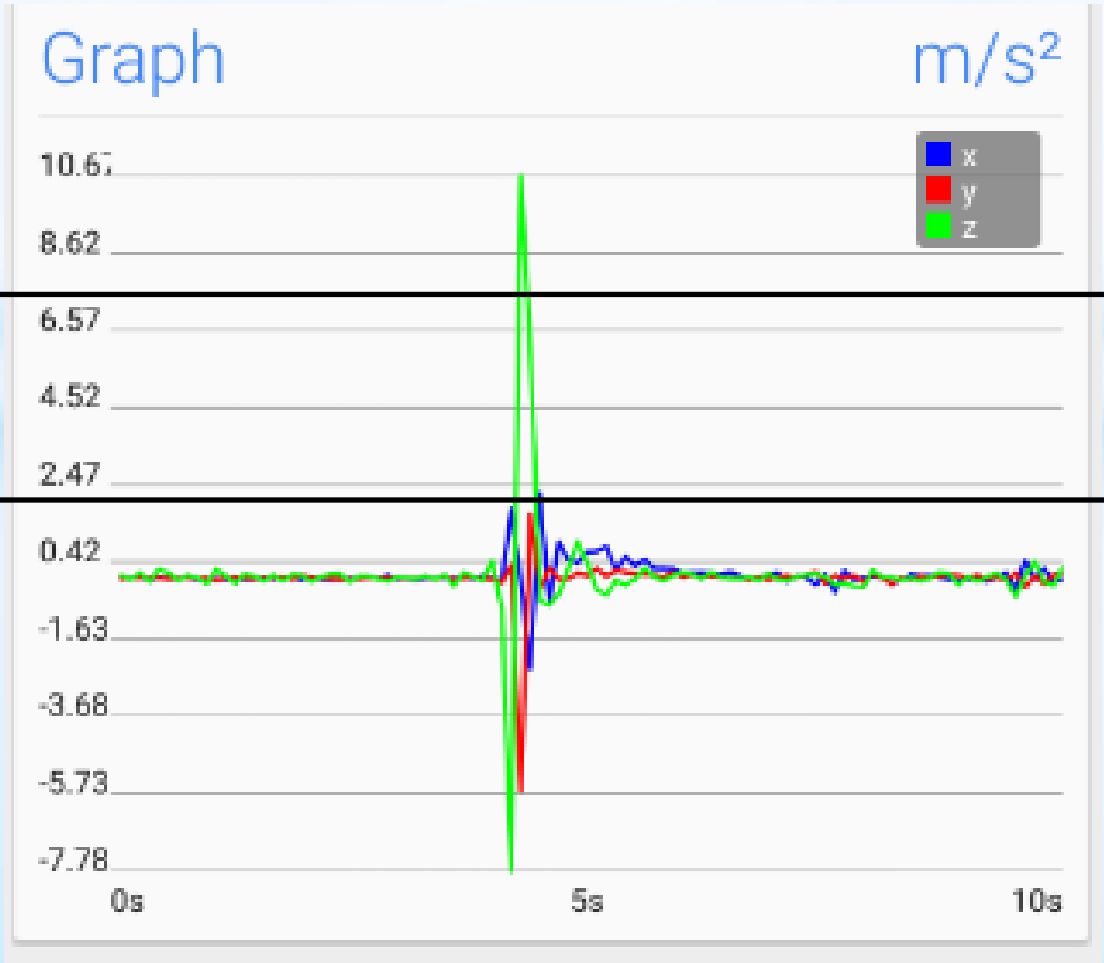
When pitch is 90 deg

Gimbal lock: https://www.youtube.com/watch?v=zc8b2Jo7mno&sns=em

App to detect a simple phone gesture

Ad-hoc technique using two thresholds, on the linear accl value on the phone's z-axis (assuming phone is face up and parallel to the ground).

Better approaches include machine learning and dynamic time warping.



Sampling rate and power

- Depends on event frequency (Eg: Walking, Occupancy)
- Power consideration (Eg: Accl 20 Hz v/s 1 Hz)

Android gotchas:

- Android delivers data at faster rate than specified
- Actual sampling rate can vary with time (*onSensorChanged()*)

Noise and drift

1. Noise (Eg: Accl)
2. Drift (Eg: Gyro)

Noise and drift

Example –

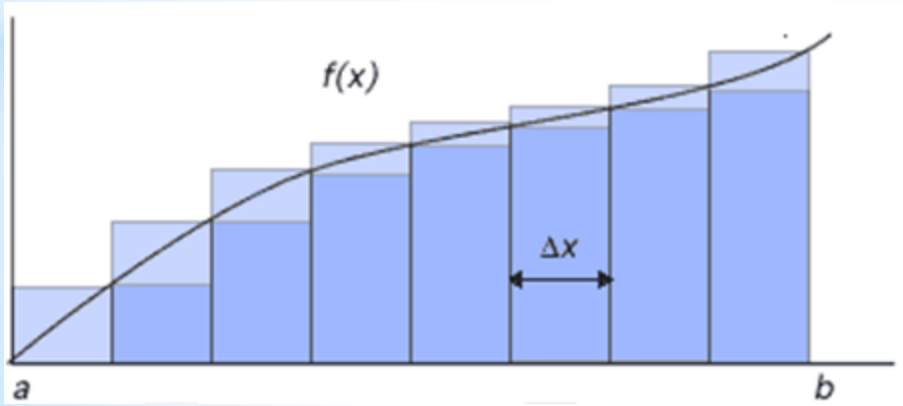
Integrating Accelerometer (*m/sec²*) → Velocity

v/s

Very Noisy

Integrating Gyroscope (*radian/sec*) → Angle rotated

Less noisy, but drifts with time



Combine Accl + Gyro: Sensor Fusion

Q: Can Accl be used to get distance?

Break

Introduction to Android

Introduction to Android

What is Android?

- **Linux-based Operating System for mobile devices**
- **Developed by Android, Inc. (later purchased by Google)**
- **First Android phone in October 2008 (HTC)**
- **Multiple companies producing Android phones:**

Samsung, HTC, LG,

Motorola, ASUS,

and many others

Introduction to Android

Code name	Version number	Initial release date	API level
Alpha	1.0	September 23, 2008	1
Beta	1.1	February 9, 2009	2
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4
Eclair	2.0 - 2.1	October 26, 2009	5 - 7
Froyo	2.2 - 2.2.3	May 20, 2010	8
Gingerbread	2.3 - 2.3.7	December 6, 2010	9 - 10
Honeycomb	3.0 - 3.2.6	February 22, 2011	11 - 13
Ice Cream Sandwich	4.0 - 4.0.4	October 18, 2011	14 - 15
Jelly Bean	4.1 - 4.3.1	July 9, 2012	16 - 18
KitKat	4.4 - 4.4.4	October 31, 2013	19
Lollipop	5.0 - 5.1.1	November 12, 2014	21 - 22
Marshmallow	6.0 - 6.0.1	October 5, 2015	23
Nougat	**7.0 - 7.1.1**	**August 22, 2016**	**24 - 25**

Introduction to Android

Typical specs:

- **2 GB RAM, 32 GB flash storage, 1 GHz multi-core processor, GPU**
- **Have variety of sensors!**
- **Limited battery : approx 2300 mAh (varies)**

Mobiles have limited memory (not anymore!):

- **Android can *kill* an App taking too much memory**
- **Each App has an *upper limit on heap* space: 32 MB on Galaxy Nexus**

Major battery drainers:

- **Display, Processor, GPS, Cellular data**

Introduction to Android

How does Android manage Applications?

- ***Multiple*** **applications can run at same time**
- **Only *one instance* of an App runs at a time**
- **Typical buttons: Home, Back, Menu, Search**

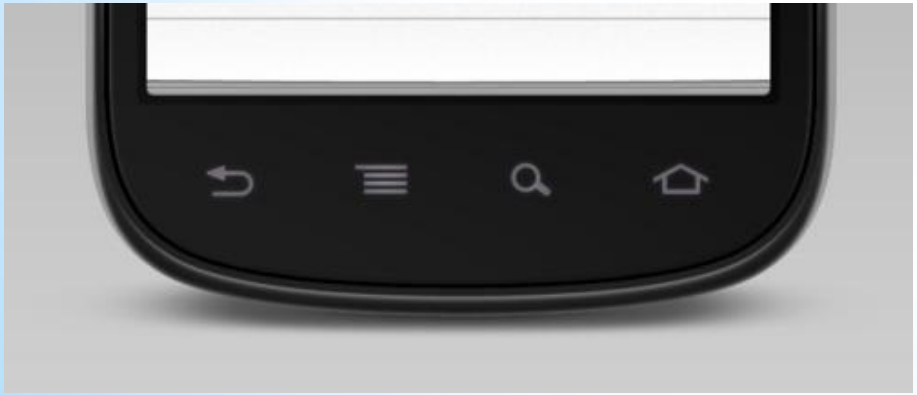
  Home button:

  Similar to *minimizing* a window (state *retained*)

  Back button:

  Similar to *closing* a window (state *destroyed*)

- **After App destroyed, its *still in memory*!**



Introduction to Android

How to write Applications?

- Apps are mostly written in Java.
- Compiled to Dalvik (dex) code.

  (Dalvik is a village in Iceland)

Each App runs:

- On Dalvik VM
- Separate process
- Separate User ID (for security)

Why Dalvik?

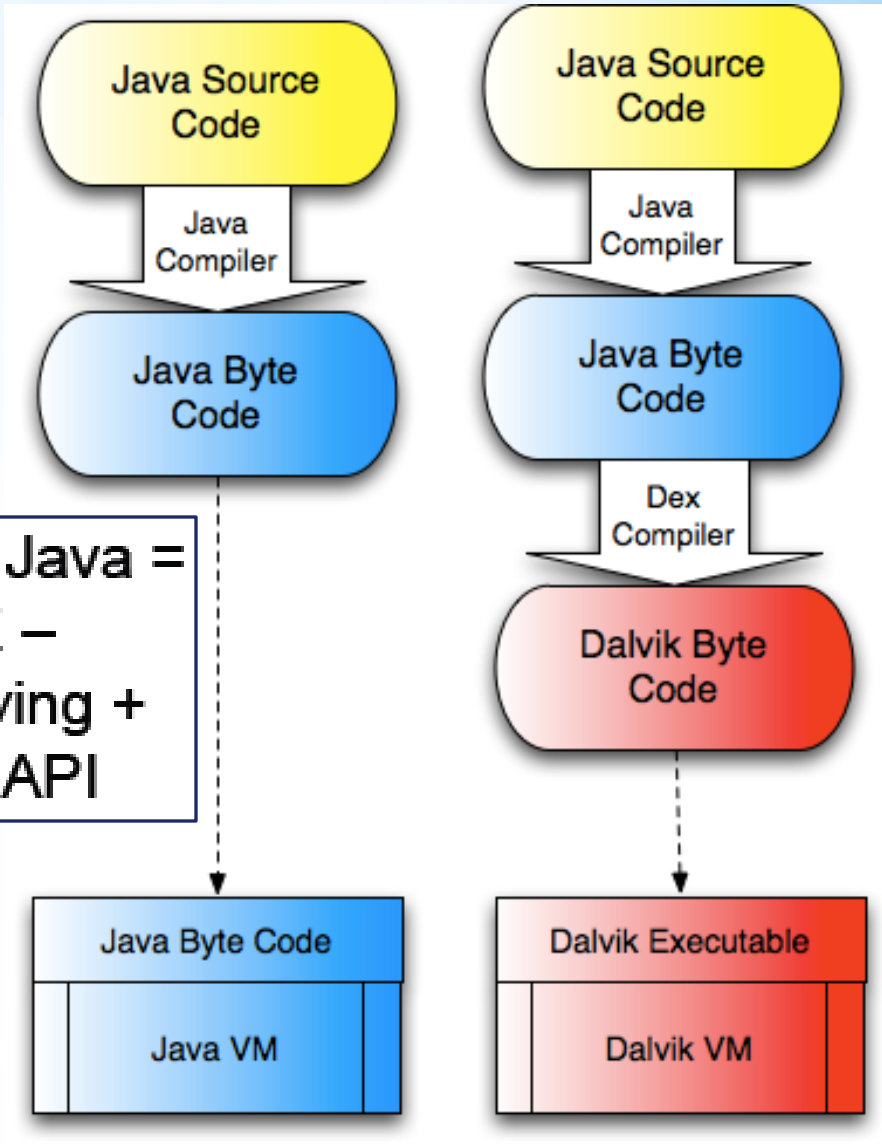
- Memory Optimized

  (Jar 113 KB ➔ Dex 20 KB)
- Avoid paying money to Sun (Oracle)

Why not directly convert Java ➔ Dex?

- Use 3rd party Java libraries

Android Java = Java SE – AWT/Swing + Android API



(Image from Marakana's slides on Android Internals)

Your first Android program

Installation Steps:

1. **Install Java (Version 7 or above)**
2. **Install Android SDK tools**
   - **http://developer.android.com/sdk/index.html**

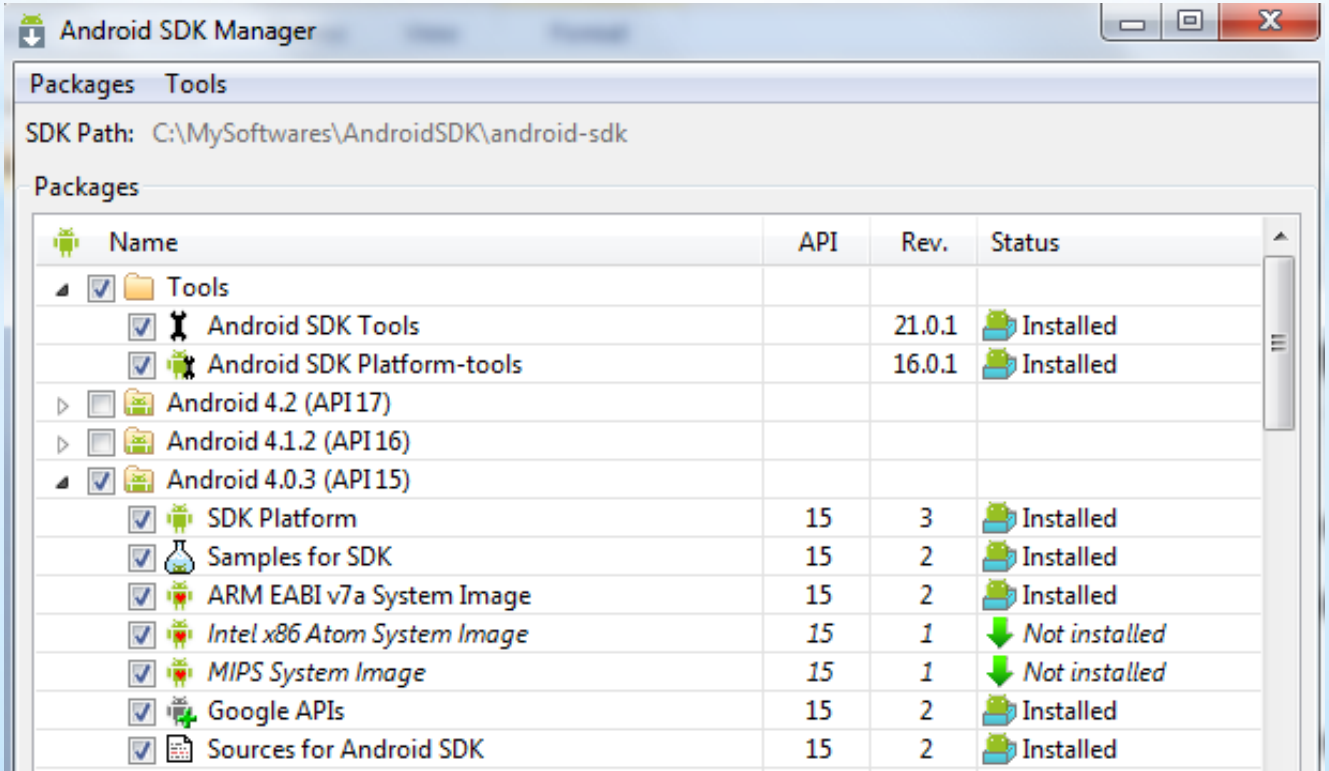
If any problem, then email/meet me

Platform	SDK tools package	Size
Windows	tools_r25.2.3-windows.zip	292 MB (306,745,639 bytes)
Mac	tools_r25.2.3-macosx.zip	191 MB (200,496,727 bytes)
Linux	tools_r25.2.3-linux.zip	264 MB (277,861,433 bytes)

Your first Android program

Installation Steps:

3. **Install Android 'APIs'** (***needs a good internet connection, takes a while***)
   - **http://developer.android.com/sdk/installing/adding-packages.html**
   - **SDK Manager.exe** (***at root of SDK installation folder***)
   - **Target API (check your device) + SDK tools + SDK platform tools**



Your first Android program

Installation Steps:

4. Install an IDE of your choice

a. Android Studio (replaced Eclipse, based on IntelliJ)

- Android Studio bundle (~1 GB) has SDK + IDE + Lollipop API

http://developer.android.com/sdk/index.html

Platform	Android Studio package	Size
Windows (64-bit)	android-studio-bundle-145.3537739-windows.exe Includes Android SDK **(recommended)**	1,674 MB (1,756,130,200 bytes)
	android-studio-ide-145.3537739-windows.exe No Android SDK	417 MB (437,514,160 bytes)
	android-studio-ide-145.3537739-windows.zip No Android SDK, no installer	438 MB (460,290,402 bytes)
Windows (32-bit)	android-studio-ide-145.3537739-windows32.zip No Android SDK, no installer	438 MB (459,499,381 bytes)
Mac	android-studio-ide-145.3537739-mac.dmg	434 MB (455,263,302 bytes)
Linux	android-studio-ide-145.3537739-linux.zip	438 MB (459,957,542 bytes)

Your first Android program

Installation Steps:

4. Install an IDE of your choice

b. IntelliJ (approx 190 MB)

- **http://www.jetbrains.com/idea/download/index.html**

c. Emacs (or any text editor of your choice)

d. Eclipse (your project may implode randomly)

Your first Android program

Installation Steps:

5. **Install phone's device drivers (*Mac users skip this step*)**

   a. **Windows (For Nexus 4, download Google USB Driver using SDK manager)**

   - **Install N4 driver: http://developer.android.com/tools/extras/oem-usb.html**
   - **Galaxy S3: http://www.samsung.com/us/support/owners/product/SCH-I515MSAVZW**

   b. **Linux (Create the file** /etc/udev/rules.d/51-android.rules)

   - **http://developer.android.com/tools/device.html**
   - **SUBSYSTEM=="usb", ATTR{idVendor}==“1004", MODE="0666", GROUP="plugdev"**

Your first Android program

Creating an Android project:

1. **Using command line**

```
$ android list targets
```
(to get target IDs)

```
$ android create project --target 9 --name HelloWorld --path
./HelloWorld --activity HelloWorldActivity --package
nus.cs4222.helloworld
```

2. **Using an IDE**

- **File ➔New Project**
- **Fill in the details, similar to above**

Your first Android program

Project Structure:

src: **Your source code**

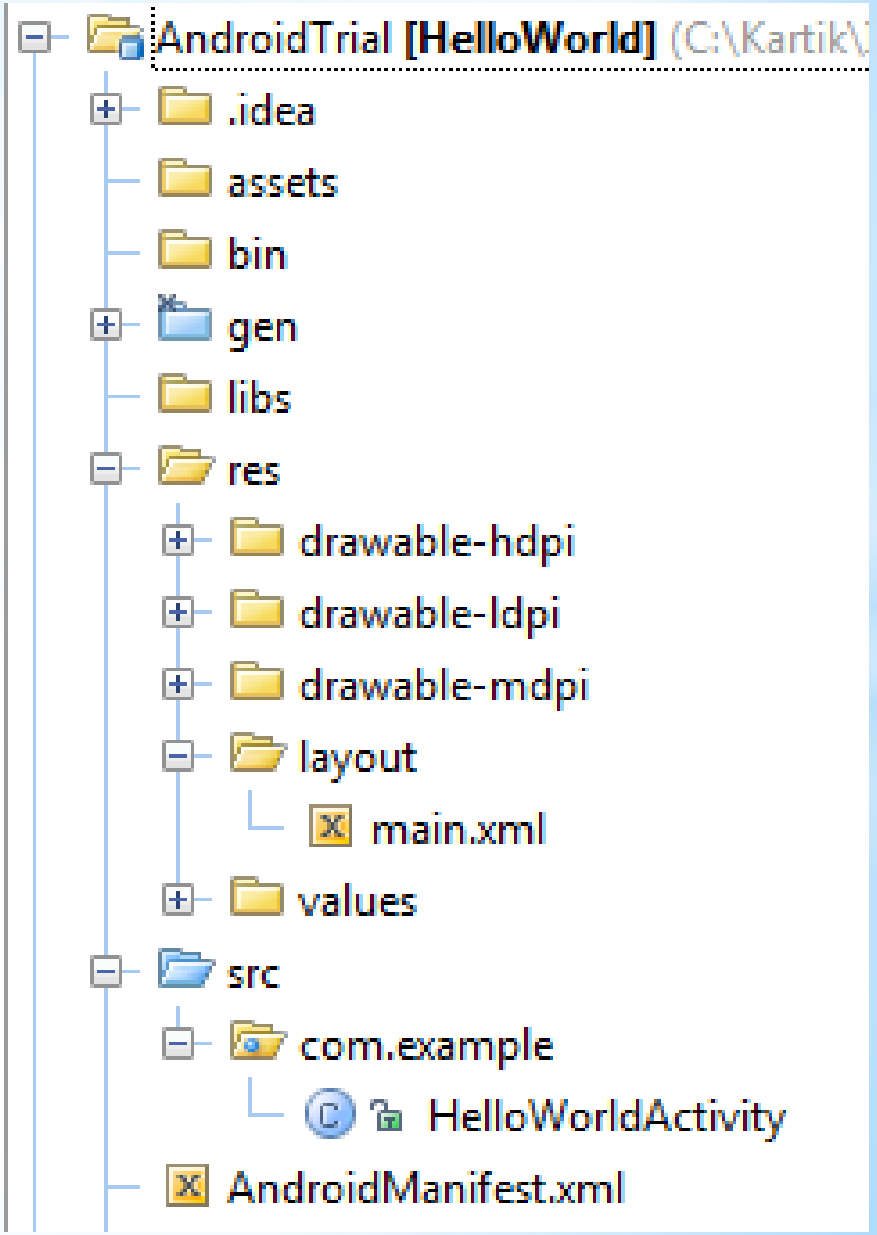
res/layout/main.xml: **Your GUI**

libs: **Library jars**

Manifest: **App description**

In Android, the GUI screen is called an *'Activity'*

(will be covered in detail in another tutorial)



Your first Android program

res/layout/main.xml : (*already created by default*)

```
<?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:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Hello World!" />
</LinearLayout>
```

Your first Android program

HelloWorldActivity.java : (*already created by default*)

```
public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate( Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );

        // Specify what GUI to use (res/layout/main.xml)
        // 'R' stands for 'Resources'
        setContentView( R.layout.main );
    }
}
```

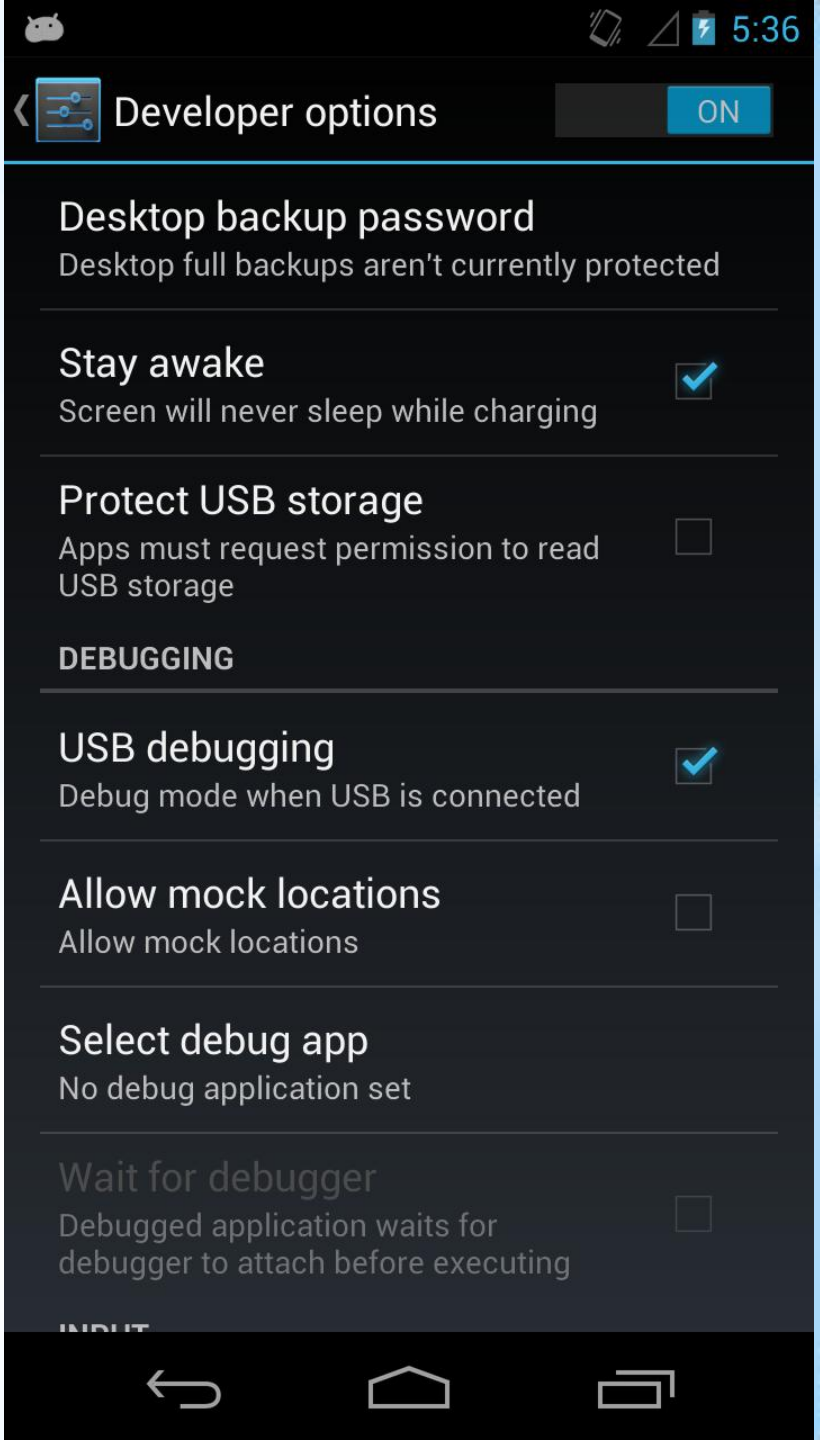
Tap ‘Build Number’ in your phone settings 7 times to enable Developer options.

Turn on USB debugging on the phone:

Settings ➔ Developer Options

Enable USB Debugging

Make sure you *installed the USB Driver* for this phone!



Build (compile) your App:

1. Using command line:

```
$ ant debug
```

```
$ gradle assembleDebug
```

2. Using an IDE:

   'Build' button/menu

- *App: bin/HelloWorld-debug.apk*

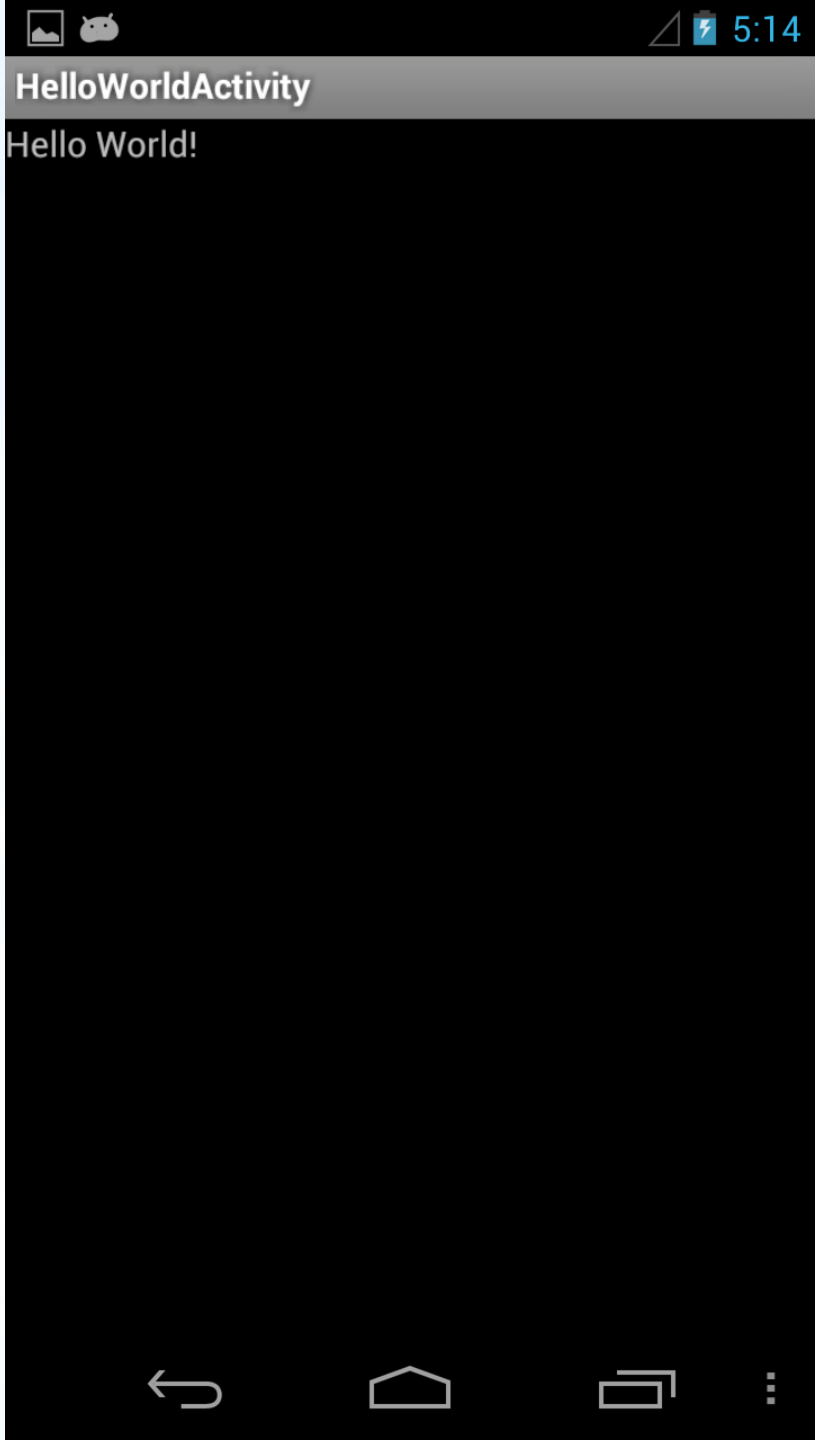
Install App on phone:

1. Using command line:

```
$ adb install -r
      bin/HelloWorld-debug.apk
```

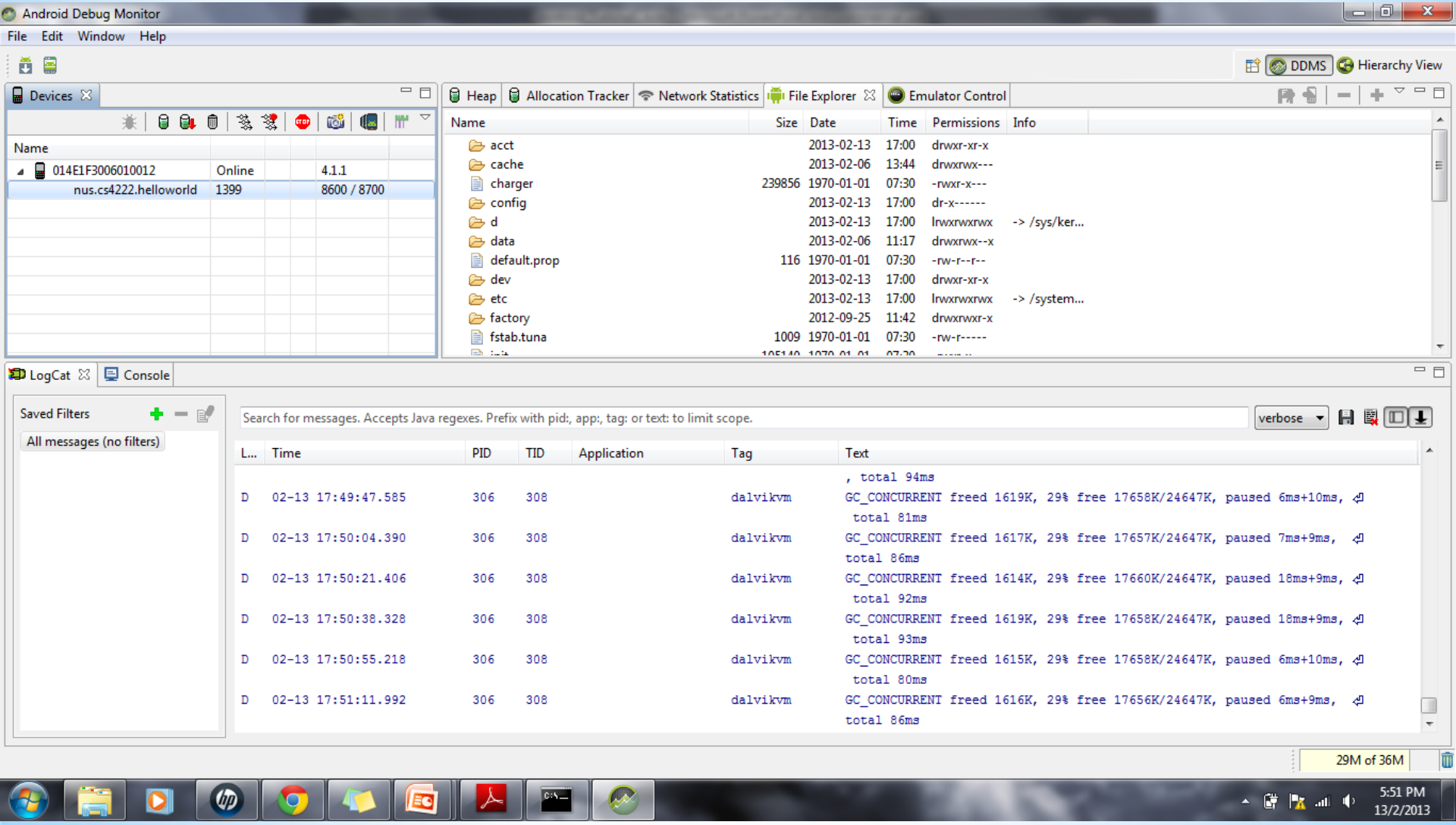
2. Using an IDE:

   'Install/Run' button/menu



Debugging (use the *filter* field most of the time):

Monitor.bat (AKA ddms.bat)



Your first Android program

Adding more GUI 'widgets':

- ***EditText***: **User can enter text here**
- ***Button***: **Pressing it triggers an action**
- ***TextView***: **Displays some text**

1. Enter text here

2. Press the button

3. Text gets appended here

Your first Android program

res/layout/main.xml : (*add Button and EditText widgets)*

```
<LinearLayout ...>
  <EditText android:id="@+id/EditText_Message"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="" />
  <Button android:id="@+id/Button_Send"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="Send Message" />
  <TextView
      android:id="@+id/TextView_Message"
      ... />
</LinearLayout>
```

Widgets have IDs
"@" Referring to a resource
"+" Adds this ID to the R class

GUI layout defined using XML

Your first Android program

HelloWorldActivity.java :

```
public class HelloWorldActivity extends Activity {
  /** Called when the activity is first created. */
  public void onCreate( Bundle savedInstanceState ) {
    ...
  }

  // Text View (displays messages)
  private TextView textView_Message;
  // Edit Text (user enters message here)
  private EditText editText_Message;
  // Button to trigger an action
  private Button button_Send;
}
```

GUI widgets in Java code

Your first Android program

```
public class HelloWorldActivity extends Activity
        implements View.OnClickListener {
  public void onCreate( Bundle savedInstanceState ) {
    ...
    // Get references to the GUI widgets
    textView_Message =
      (TextView) findViewById( R.id.TextView_Message );
    editText_Message =
      (EditText) findViewById( R.id.EditText_Message );
    button_Send =
      (Button) findViewById( R.id.Button_Send );
    // Set the listener for button clicks
    button_Send.setOnClickListener( this );
  }
}
```

Your first Android program

HelloWorldActivity.java :

```
public class HelloWorldActivity extends Activity
        implements View.OnClickListener {
  public void onCreate( Bundle savedInstanceState ) { ... }
  ...
  public void onClick( View v ) {
    // Change the text view
    String enteredText = editText_Message.getText();
    String oldText = textView_Message.getText();
    String newText = oldText +
                     "\n New Message: " + enteredText;
    textView_Message.setText( newText );
  }
  ...
}
```

Your first Android program

Android online tutorials:

http://developer.android.com/guide/components/fundamentals.html

http://developer.android.com/training/basics/firstapp/index.html

Note: **Many Android programmers directly jump into coding.**

However, it is more *important to understand the concepts first* before writing complex Apps. Otherwise, coding becomes a messy headache.

Questions?

Thank You!