immersion

**Peter van der Linden**
**Android Technology Evangelist**

Jan 2014

NASDAQ: IMMR

# Code to go!

## Writing your first Android app

# Agenda

1. Compilation tools
2. Importing an existing project
3. The folders that make up an app
4. GUI Basics
5. Run it!

Slides online at:

1 **Compilation tools**

2 Importing an existing project

3 The folders that make up an app

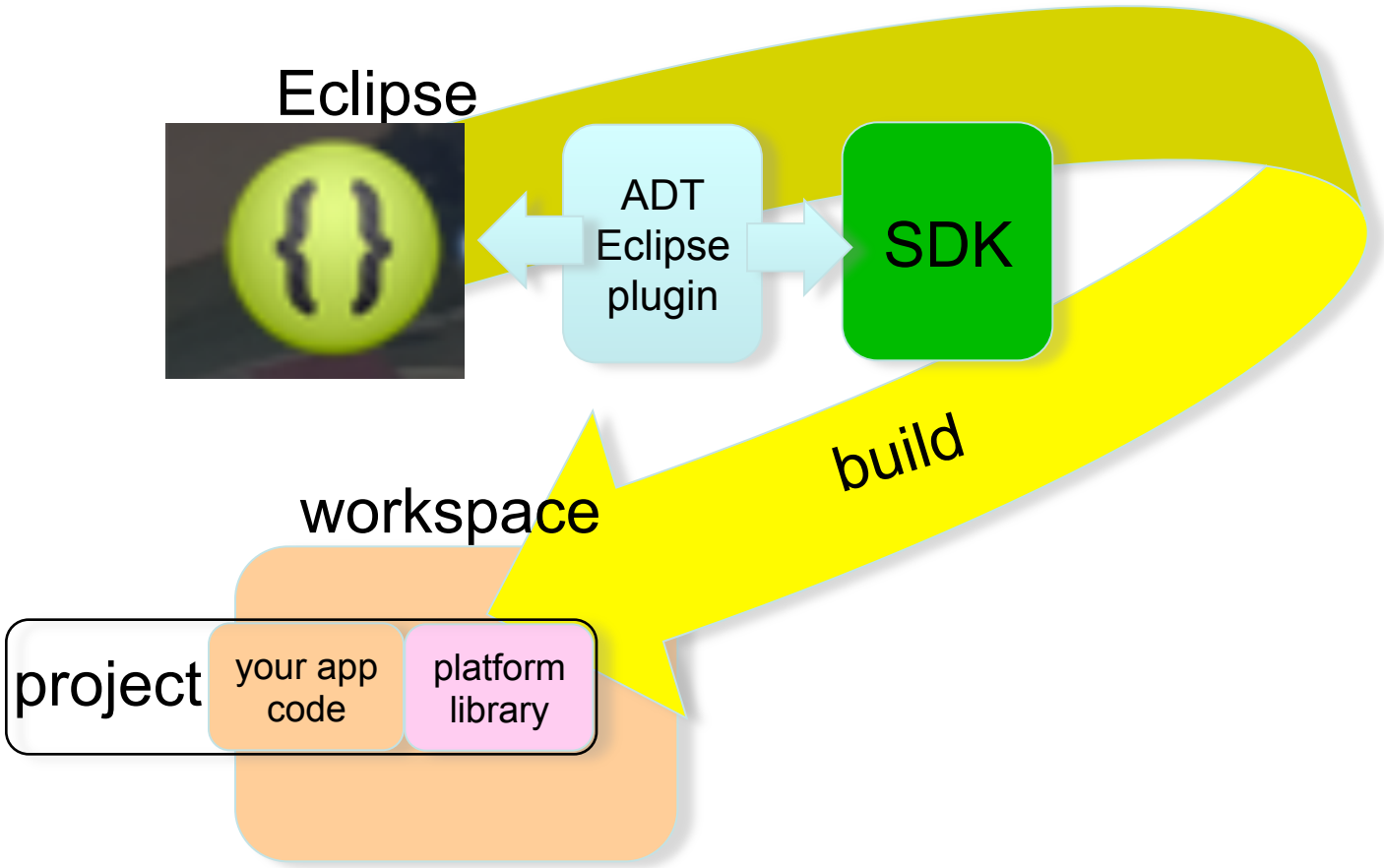
4 GUI Basics

5 Adding some Views

# Compilation tools

- Technologies:
  - Java, XML, SQLite, OpenGL, embedded development
- Tools
  - Eclipse (and Android Studio, based on IntelliJ IDE)
  - Android SDK
  - Platform libraries

# Compilation tools



# Eclipse main screen

Package Expl | Type Hierarc

- confer
- confer1
  - src
    - com.example
      - ChooseRoom.java
        - ChooseRoom
          - onCreate(Bundle) : void
      - ShowMap.java
  - gen [Generated Java Files]
  - Android 2.1
  - Android Dependencies
  - assets
  - bin
  - res
  - AndroidManifest.xml
  - project.properties
- confer2
- cryptex
- feels
  - src
    - com.example.feels
      - MainActivity.java
        - MainActivity
          - handleButton(View) : void
          - onCreate(Bundle) : void
  - gen [Generated Java Files]
  - Android 4.3
  - Android Private Libraries
  - assets
  - bin
  - libs
  - res
  - AndroidManifest.xml
  - ic_launcher-web.png
  - proguard-project.txt
  - project.properties
- FillIt
- FindNumber
- FlipAnim
- Game1

activity_main.x | MainActivity.ja | ChooseRoom.java | MainActivity.ja

```
package com.example.feels;

import java.util.Random;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void handleButton(View v) {
        RelativeLayout rl = (RelativeLayout) findViewById(R.id.mylayout);
        int r=0, g=0, b=0;
        Random ran = new Random();
        g = ran.nextInt(0x100);
        r = ran.nextInt(0x100);
        b = ran.nextInt(0x100);
        Toast.makeText(this, "r="+r+ ",g="+g + ",b="+b, Toast.LENGTH_SHORT).show();
        int randomColor = 0xFF000000 + (r<<24) + (g<<16) + (b);
        rl.setBackgroundColor( randomColor);
    }


}
```

Problems | Javadoc | Declaration | Console | LogCat | Lint Warnings | Error Log

Workspace Log

type filter text

| Message | Plug-in | Date |
|---|---|---|

226M of 486M

# Eclipse main screen

Package Expl | Type Hierarc

- confer
- confer1
  - src
    - com.example
      - ChooseRoom.java
        - ChooseRoom
          - onCreate(Bundle) : void
      - ShowMap.java
  - gen [Generated Java Files]
  - Android 2.1
  - Android Dependencies
  - assets
  - bin
  - res
  - AndroidManifest.xml
  - project.properties
- confer2
- cryptex
- feels
  - src
    - com.example.feels
      - MainActivity.java
        - MainActivity
          - handleButton(View) : void
          - onCreate(Bundle) : void
  - gen [Generated Java Files]
  - Android 4.3
  - Android Private Libraries
  - assets
  - bin
  - libs
  - res
  - AndroidM
  - ic_launch
  - proguard-
  - project.pr
- FillIt
- FindNumber
- FlipAnim
- Game1

activity_main.x | MainActivity.ja | ChooseRoom.java | MainActivity.ja

```
package com.example.feels;

import java.util.Random;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void handleButton(View v) {
        RelativeLayout rl = (RelativeLayout) findViewById(R.id.mylayout);
        int r=0, g=0, b=0;
        Random ran = new Random();
        g = ran.nextInt(0x100);
        r = ran.nextInt(0x100);
        b = ran.nextInt(0x100);
        Toast.makeText(this, "r="+r+ ",g="+g + ",b="+b, Toast.LENGTH_SHORT).
        int randomColor = 0xFF000000 + (r<<24) + (g<<16) + (b);
        rl.setBackgroundColor( randomColor);
    }


}
```

Your source code file

Your projects

# Using Eclipse

- Eclipse video tutorials

  http://eclipsetutorial.sourceforge.net/totalbeginner.html

  http://www.vogella.de/articles/Eclipse/article.html

- Eclipse “Perspective” reset

  Window > Reset Perspective > Yes

1 Compilation tools

**2 Importing an existing project**

3 The folders that make up an app

4 GUI Basics

5 Run it!

# What if I did not download any platforms?

Then do it now.  Download Platform 14, and use that throughout.

In Eclipse, click on Window > Android SDK Manager

Under “Android 4.0 (API 14) Click on “SDK Platform”

Then click “Install”

These are large 100MB downloads – don’t download more than you need till you are back on your home network

# What if I did not put the SDK tools in my path?

Take a demerit for Gryffindor, and add the folders now.

**MacOS** – edit file ~/.bash_profile to add these 2 directories to PATH by adding this at the end of the file (use names for *your* PC!)

```
export PATH=$PATH:/Users/plinden/android-sdk-macosx/
  tools:/Users/plinden/android-sdk-macosx/platform-tools
```

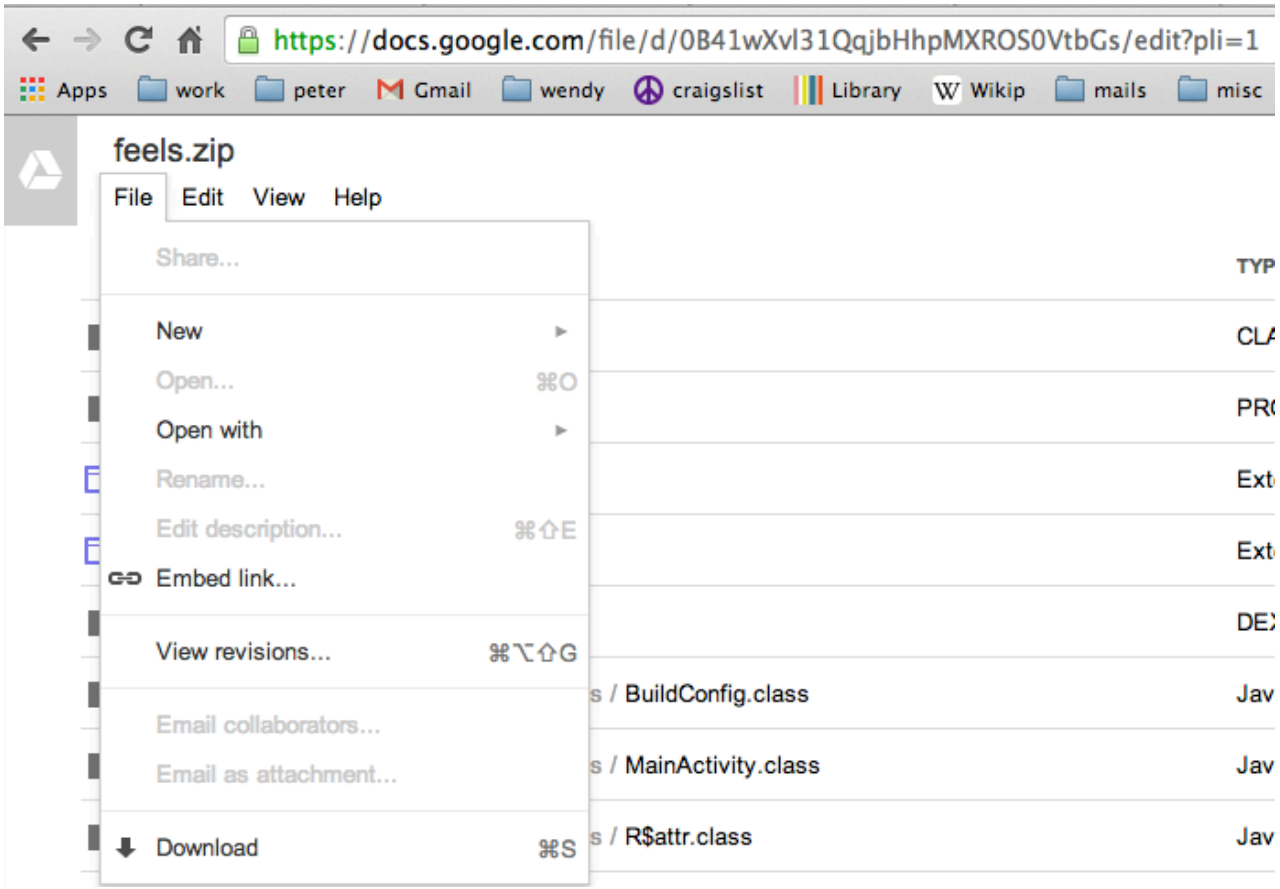
**Linux** – add the SDK two folders, tools and platform-tools, to your PATH in your shell initialization file (file varies with the shell you use).

**Windows** – path environment variable is set somewhere under control panel. Google “Windows 7 set env variable” (windows 8 etc)

# Get my existing project “feels” into Eclipse

Download the zip file from http://goo.gl/TN2Hpq

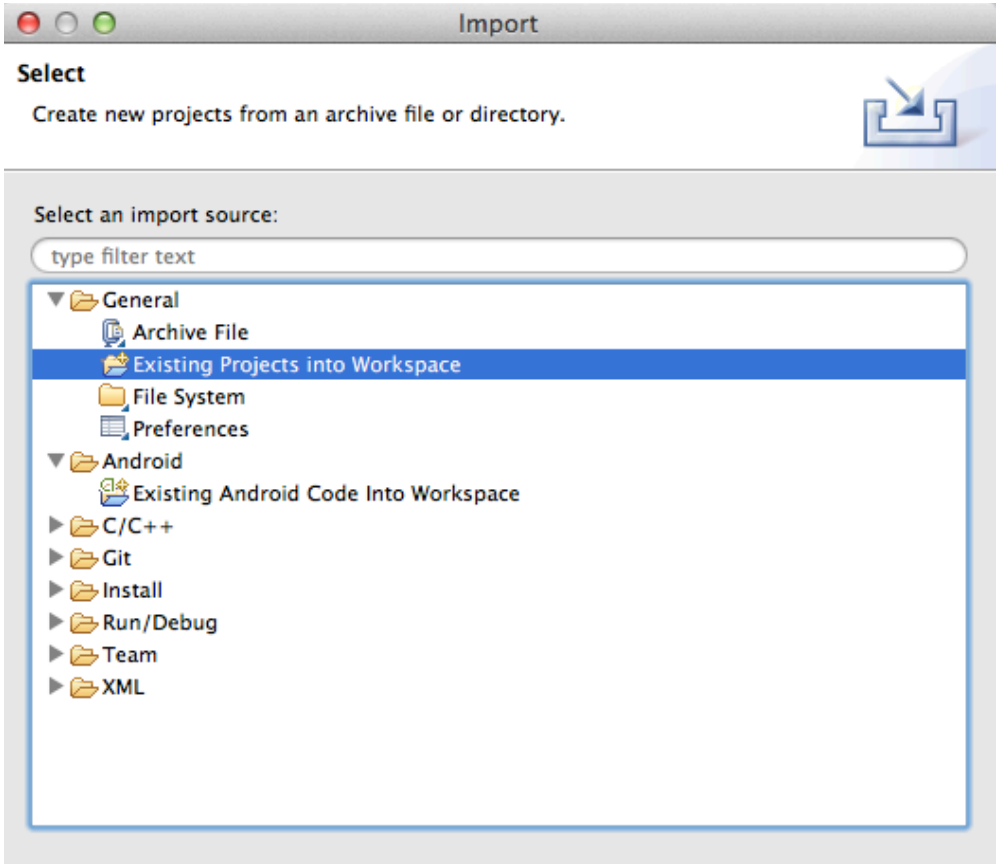
On that page, hit File > Download



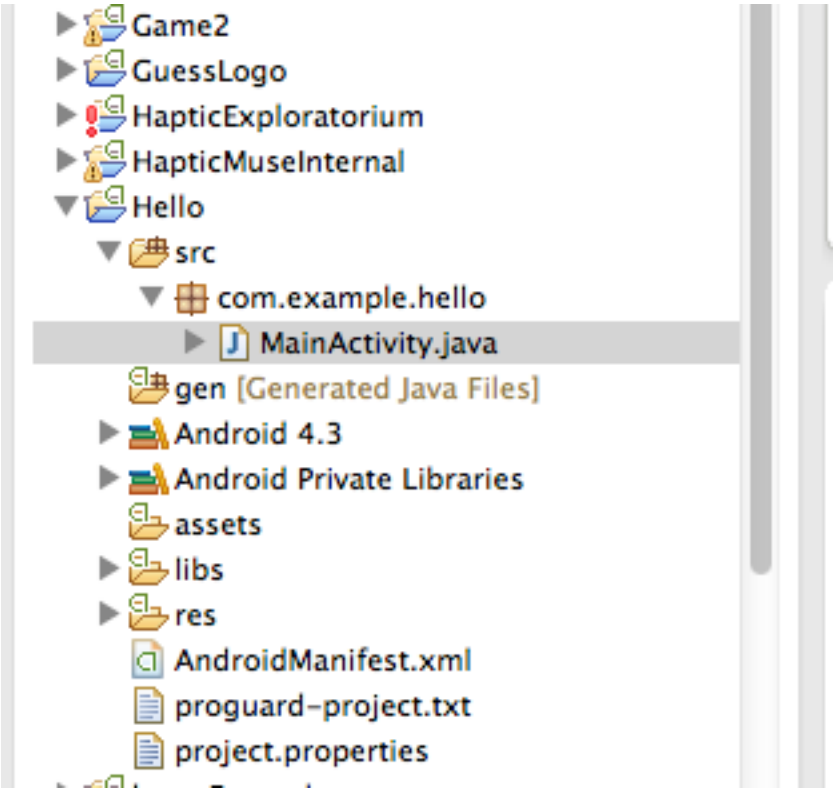
# Importing existing project into Eclipse

Unzip the downloaded feels.zip file somewhere handy

File > Import … > Existing files into workspace



# Imported project appears in Eclipse




You can expand folders by clicking right pointing triangle

If project has errors, click Project > Clean > OK

# Agenda

1 Compilation tools

2 Creating a new project

**3 The folders that make up an app**

4 GUI Basics

5 Adding some Views

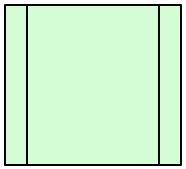
6 Execution tools

# Files that make up a Mobile App

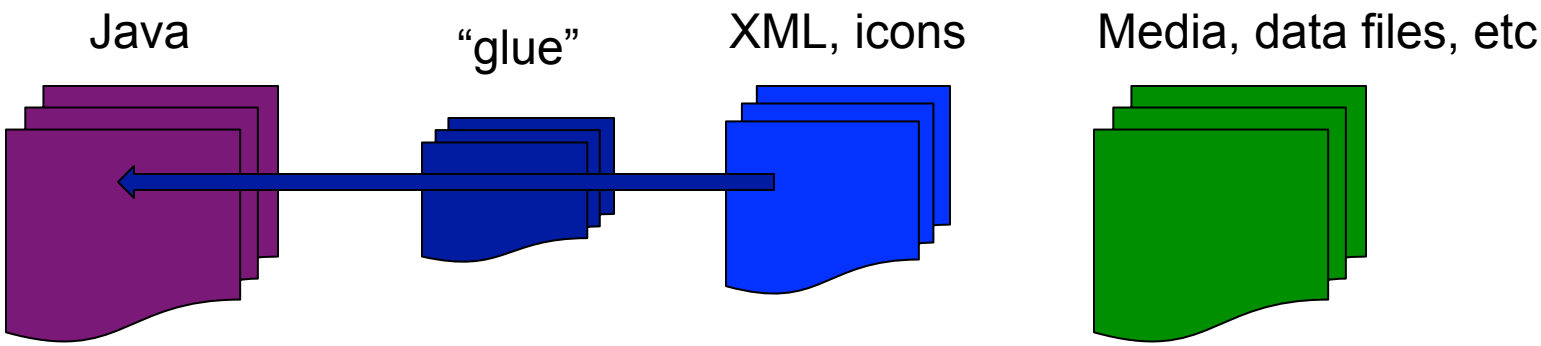
- **Java files**
- **Resource files**
  - Png files for icons (up to 5 different screen resolutions)
  - XML files to specify the GUI layout and controls
  - XML files to hold literal strings
  - A project manifest file in XML
- **Asset files (photos, music, video…, other files not compiled or localized)**

# Ingredients of an App

- Source code for every Android app has:

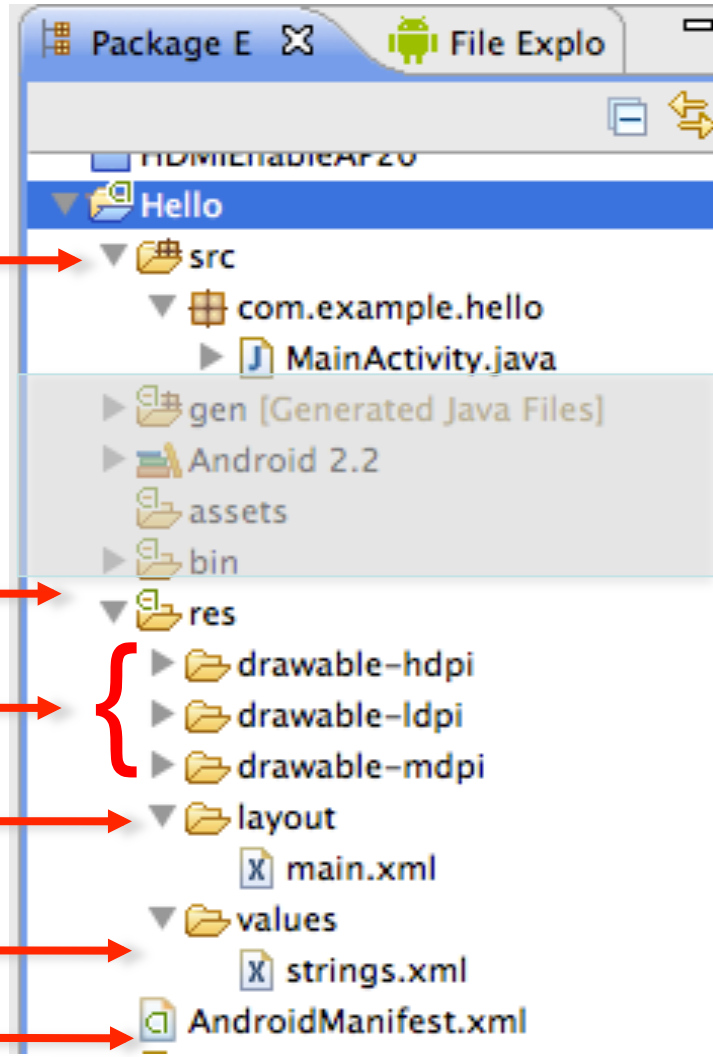


AndroidManifest.xml
describes the app overall, features used, version, etc




- App binary is an .apk file (zip format)
- contains the compiled version of these files

# The “Top 2 folders” – src, res




- **Java files**
- **Resource files**
- Png files for icons (1-5 dpi’s)
- XML files for GUI layout
- XML files to hold literal strings
- A project manifest file in XML
- Restriction: *filenames under res folder must contain only lowercase a-z, 0-9, or _.*

1 Compilation tools

2 Importing an existing project

3 The folders that make up an app

**4 GUI Basics**

5 Run it!

# GUIs in Android

## Views (Widgets, Controls)

- E.g. Button, CheckBox, TextView, ProgressBar,
- About 70 basic controls

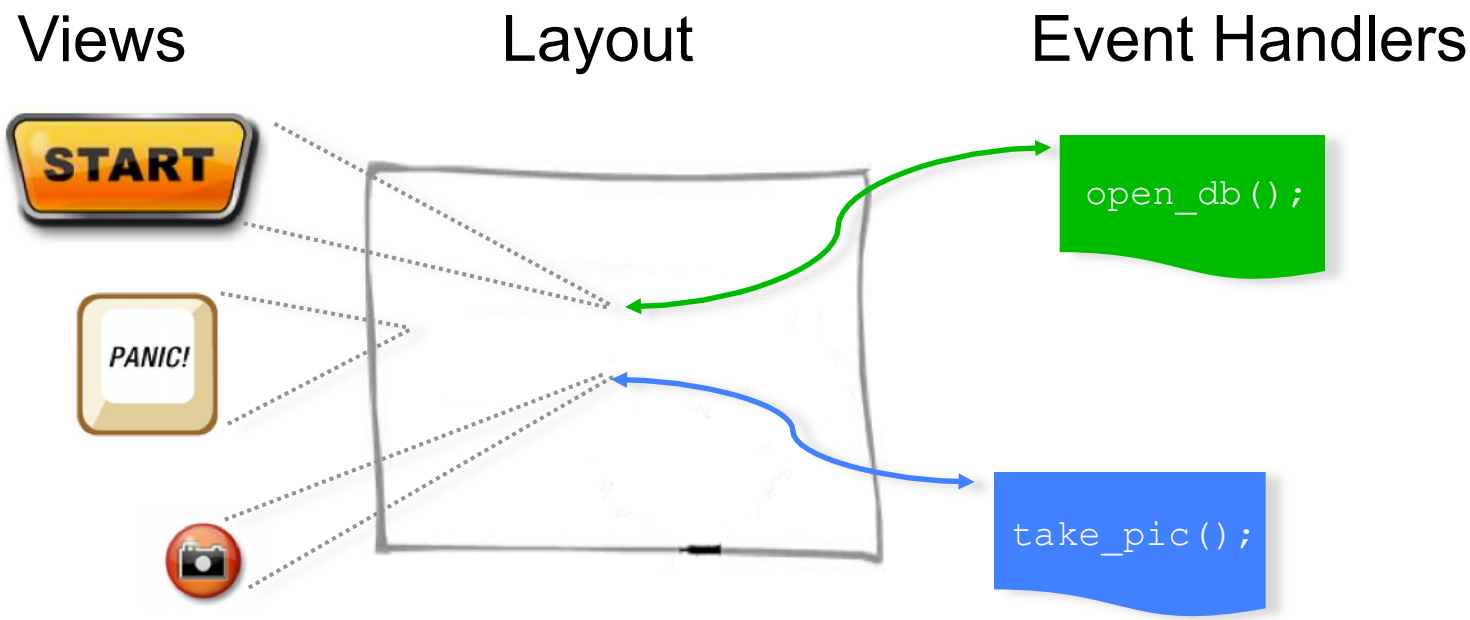
## Layouts

- Defines where each View is on a screen
- One XML file per screen
- “alternative resources” – diff layout for portrait vs land

## Event handlers

- When a user “operates” a View, it fires an event
- Developer writes event handler code to process it

# GUIs in Android - diagram




- how it looks
- what events it fires
- specified in XML in a layout file

- position on screen
- specified in XML file
- the Activity sets this file as its “content view” (how it looks)

- what happens when view is clicked
- written in Java

# Some Views in more detail

# Peter's handy-dandy XML cheat-sheet

| What it's called | What it looks like |
|---|---|
| Declaration | `<?xml version="1.0" encoding="utf-8"?>` |
| Element | `<someTagName attributes> nested_elements </someTagName>` |
| Element | `<someTagName attributes />` |
| Attribute | `someName="someValue"` |
| Comment | `<!-- some commentary here -->` |
| Namespace Declaration | `Xmlns:someNamespaceName="someURI"` |
| Android namespace declaration | `xmlns:android= "http://schemas.android.com/apk/res/android"` |
| Attribute name from android namespace | `android:layout_width="fill_parent"` |

# Getting into XML

- There is an XML file defining the GUI objects on its screen/ Activity

```
<?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="@string/hello”  />

</LinearLayout>
```

# XML for a View

- Every View must have a value for android:layout_width and _height

Tells layout manager how much room you want
for the WIDTH and the HEIGHT of the component

- *"fill_parent"* magic word that says "greedy – as much as possible"
  "match_parent" is also used.
- *"wrap_content"* magic word that says "frugal – as little as possible"

```
<TextView
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@string/hello"  />
```

# Gluing XML names to Java code

The XML names in res folder are visible in Java namespace!
The glue code is generated for you.

Create an ID name for an XML element with this attribute

```
<TextView   android:id="@+id/myTV"
```

In Java, get hold of that XML-declared TextView by:

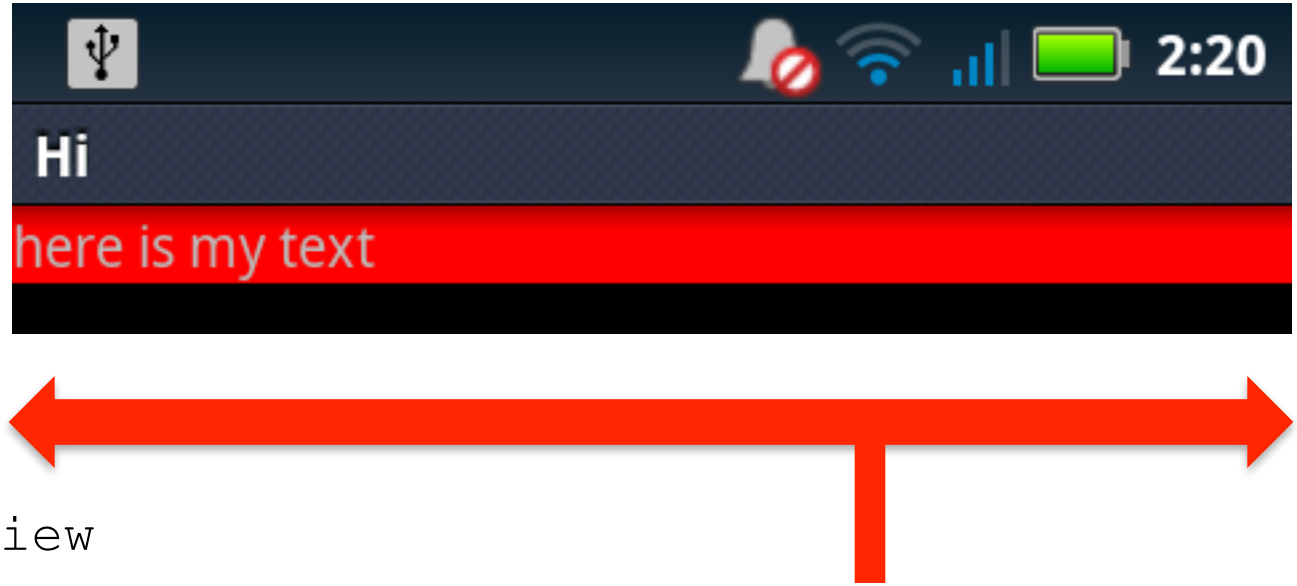
```
TextView tv = (TextView) findViewById( R.id.myTV );
```

In XML, get hold of that XML-declared TextView by:

```
<Button   android:layout_below="@id/myTV"
```

# android.widget.TextView

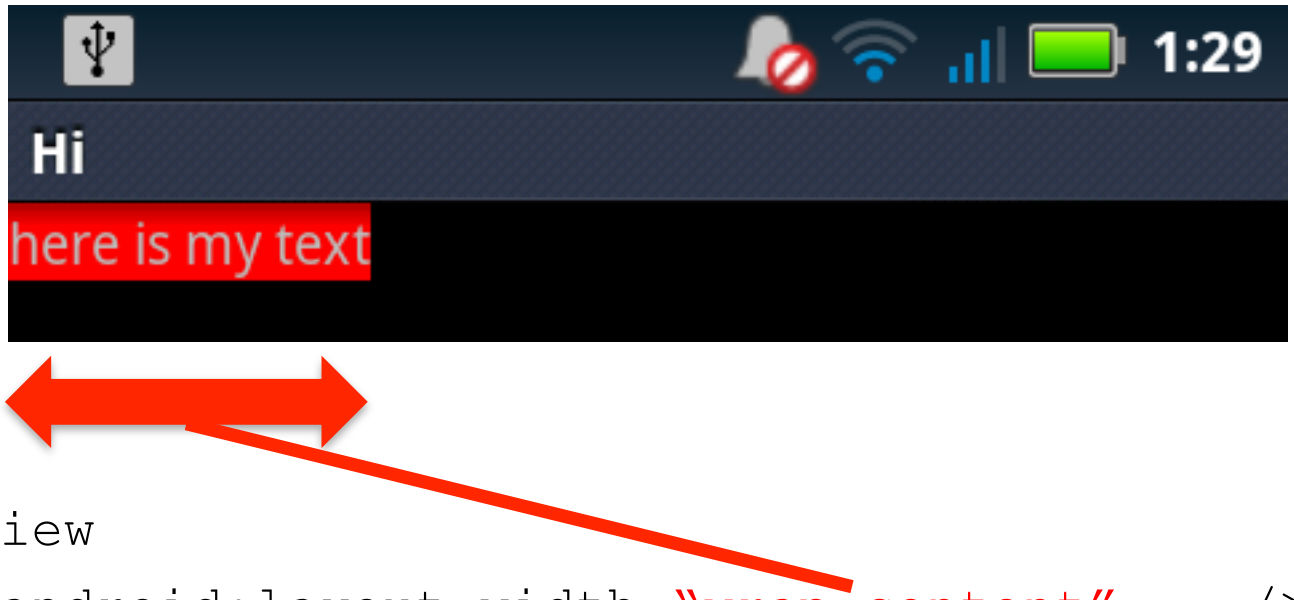
- Appearance:




```
<TextView
        android:layout_width="fill_parent"     />
```

# android.widget.TextView

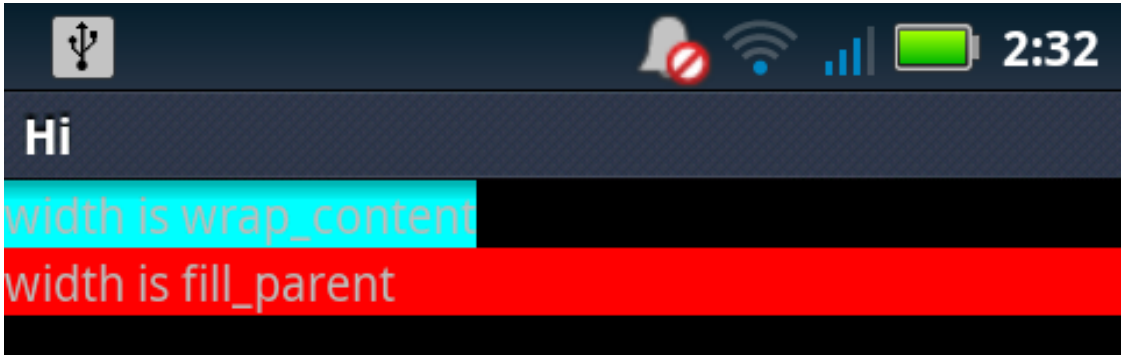
- Appearance:




```
<TextView
        android:layout_width="wrap_content"     />
```

# android.widget.TextView

- Appearance:



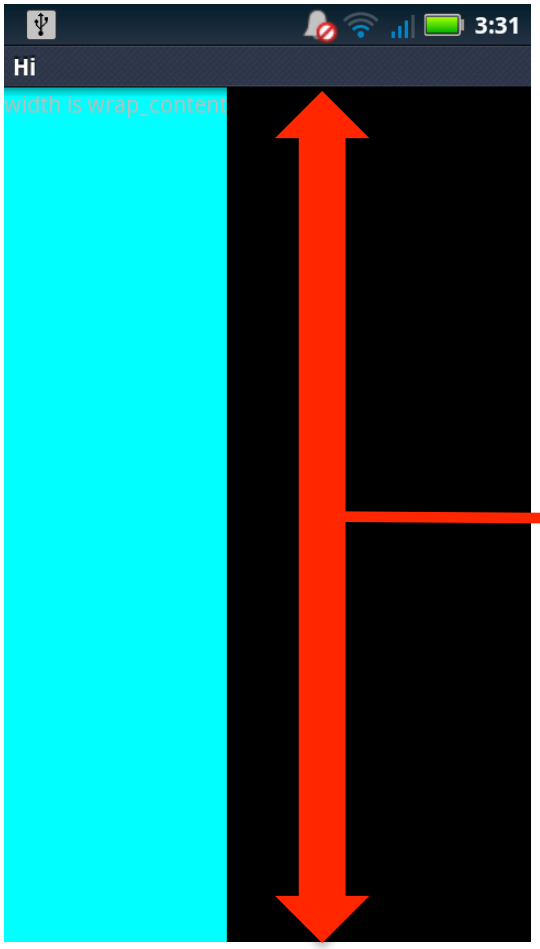

- XML in res/layout/myname.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#0FF"
    android:text="width is wrap_content" />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#FF0000"
    android:text="width is fill_parent" />
```

# Easy to make mistakes!

- Appearance:




XML in res/layout/myname.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:background="#0FF"
    android:text="width is wrap_content" />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#FF0000"
    android:text="width is fill_parent" />
```

# Attributes for android.widget.TextView

- Use the Android Developer Docs
- http://developer.android.com/reference/android/R.styleable.html#TextView
- There are about 75 attributes for TextView

Guide | Reference | Resources | Videos | Blog

Filter by API Level: 15

public static final int[] **TextView**

Attributes that can be used with a TextView.

Includes the following attributes:

| Attribute | Description |
|---|---|
| android:autoLink | Controls whether links such as urls and email addresses are automatically found and converted to clickable links. |
| android:autoText | If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors. |
| android:bufferType | Determines the minimum type that getText() will return. |
| android:capitalize | If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. |
| android:cursorVisible | Makes the cursor visible (the default) or invisible. |
| android:digits | If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept. |
| android:drawableBottom | The drawable to be drawn below the text. |
| android:drawableEnd | The drawable to be drawn to the end of the text. |
| android:drawableLeft | The drawable to be drawn to the left of the text. |
| android:drawablePadding | The padding between the drawables and the text. |
| android:drawableRight | The drawable to be drawn to the right of the text. |
| android:drawableStart | The drawable to be drawn to the start of the text. |
| android:drawableTop | The drawable to be drawn above the text. |

# android.widget.Button

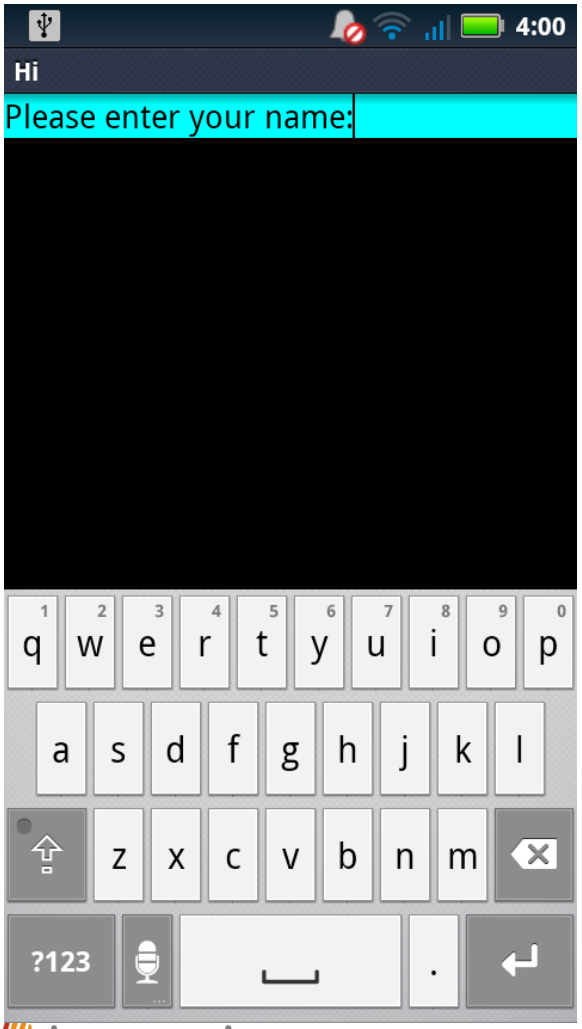
- Appearance:


- XML

```
<Button android:layout_width="fill_parent"
        android:layout_height="wrap_content”
        android:text="@string/brew"
        android:id="@+id/bt" />
```

# android.widget.EditText

- Appearance:




```
<EditText    someAttributes   />
```

- Subclass of TextView
- No new attributes of its own
- Requires the usual layout attribs
- Click in the field to get keyboard
- And type away…

# Event Handlers in more detail

# android.widget.EditText Event Handler

- Gives you the contents of entire field for each keypress
- Implement android.view.View.OnKeyListener
- Only has 1 method:

```
onKey(View v, int keyCode, KeyEvent event)
```

- Register your listener with:

```
myedittext.setOnKeyListener( objOnKeyListener );
```

# android.widget.EditText key listener

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final EditText et = (EditText) findViewById(R.id.et);
        OKL my_okl = new OKL();
        et.setOnKeyListener(my_okl);
    }
}
class OKL implements OnKeyListener {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
            (keyCode == KeyEvent.KEYCODE_ENTER)) {
          // Perform action only for "return" key press
          EditText et = (EditText) v;
          Log.i("Hi app", et.getText().toString());
          return true;  // have "consumed event"
        }
        return false;   // have not consumed event
    }
}
```

# android.widget.Button

- Gives you an event when clicked
- Implement android.view.View.OnClickListener
- Only has 1 method:

```
onClick(View v)
```

- Register your listener with code:

```
mybutton.setOnClickListener( objOnClickListener);
```

- Or register your listener with XML attribute:

```
<Button … android.onClick="doAction" />

public void doAction(View v) { … }
```

# android.widget.Button event handler

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button bt = (Button)findViewById(R.id.bt);
        CL my_cl = new CL();
        bt.setOnClickListener(my_cl);
    }

    public void doAction(View v) { // button has been pressed
        Log.i("Hi app", v.toString() + " pressed,doAction called");
    }

class CL implements OnClickListener {
    public void onClick(View v) {
          Log.i("Hi app", v.toString() + " pressed");
    }
}
```

# Debugging

First choice – the debugger

Here are some quick ‘n dirty alternatives to see what is going on in your code

# Always have “adb logcat” running in a terminal!

```
01-22 18:31:30.520 11946 11946 W dalvikvm: threadid=1: thread exiting with uncaught exception (group=0x40018560)
01-22 18:31:30.559 11946 11946 E AndroidRuntime: FATAL EXCEPTION: main
01-22 18:31:30.559 11946 11946 E AndroidRuntime: java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.hi/
      com.example.hi.HiActivity}: android.view.InflateException: Binary XML file line #7: Error inflating class Checkbox
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1696)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:1716)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread.access$1500(ActivityThread.java:124)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread$H.handleMessage(ActivityThread.java:968)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.os.Handler.dispatchMessage(Handler.java:99)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.os.Looper.loop(Looper.java:130)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread.main(ActivityThread.java:3806)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at java.lang.reflect.Method.invokeNative(Native Method)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at java.lang.reflect.Method.invoke(Method.java:507)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:839)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:597)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at dalvik.system.NativeStart.main(Native Method)
01-22 18:31:30.559 11946 11946 E AndroidRuntime: Caused by: android.view.InflateException: Binary XML file line #7: Error inflating class Checkbox
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.view.LayoutInflater.createViewFromTag(LayoutInflater.java:581)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.view.LayoutInflater.rInflate(LayoutInflater.java:623)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.view.LayoutInflater.inflate(LayoutInflater.java:408)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.view.LayoutInflater.inflate(LayoutInflater.java:320)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.view.LayoutInflater.inflate(LayoutInflater.java:276)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at com.android.internal.policy.impl.PhoneWindow.setContentView(PhoneWindow.java:256)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.Activity.setContentView(Activity.java:1703)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at com.example.hi.HiActivity.onCreate(HiActivity.java:19)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1047)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1660)
01-22 18:31:30.559 11946 11946 E AndroidRuntime:   ... 11 more
01-22 18:31:30.559 11946 11946 E AndroidRuntime: Caused by: java.lang.ClassNotFoundException: android.view.Checkbox in loader
      dalvik.system.PathClassLoader[/data/app/com.example.hi-1.apk]
```

# Logging

```
import android.util.Log;

      Log.i(”Activity ID", “message to log, i=” + i);
```

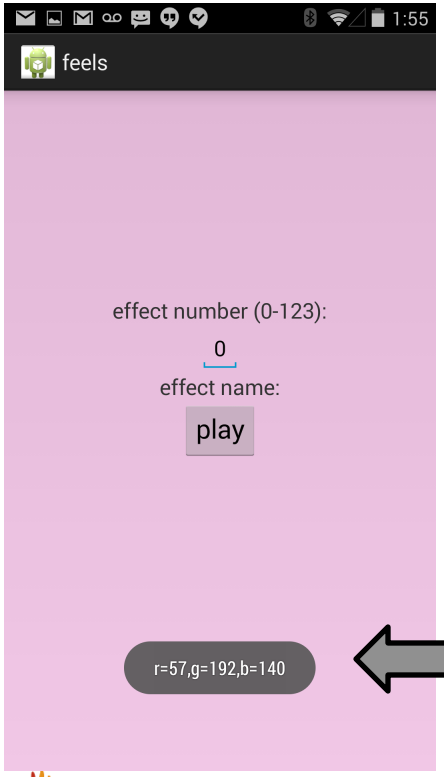
In a shell, run

```
$ adb logcat
```

# Toast

Toast – an easy way to make text "pop up" on screen. Do it when in UI thread in Activity.

```
import android.widget.toast;
 …
Toast.makeText( this,
                "my string",
                Toast.LENGTH_LONG ).show();
```




this is the toast pop-up

1 Compilation tools

2 Creating a new project

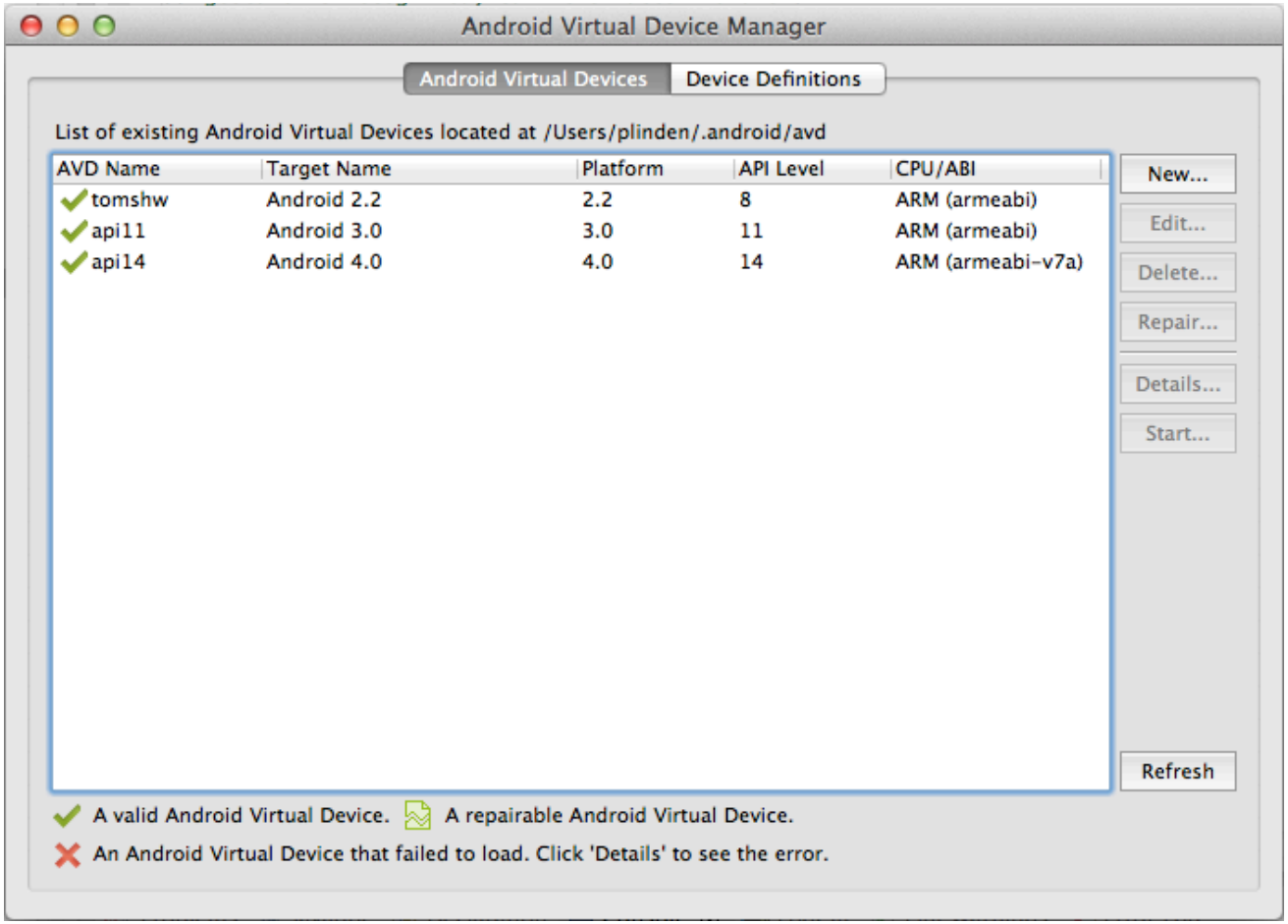
3 The folders that make up an app

4 GUI Basics

**5 Run it!**

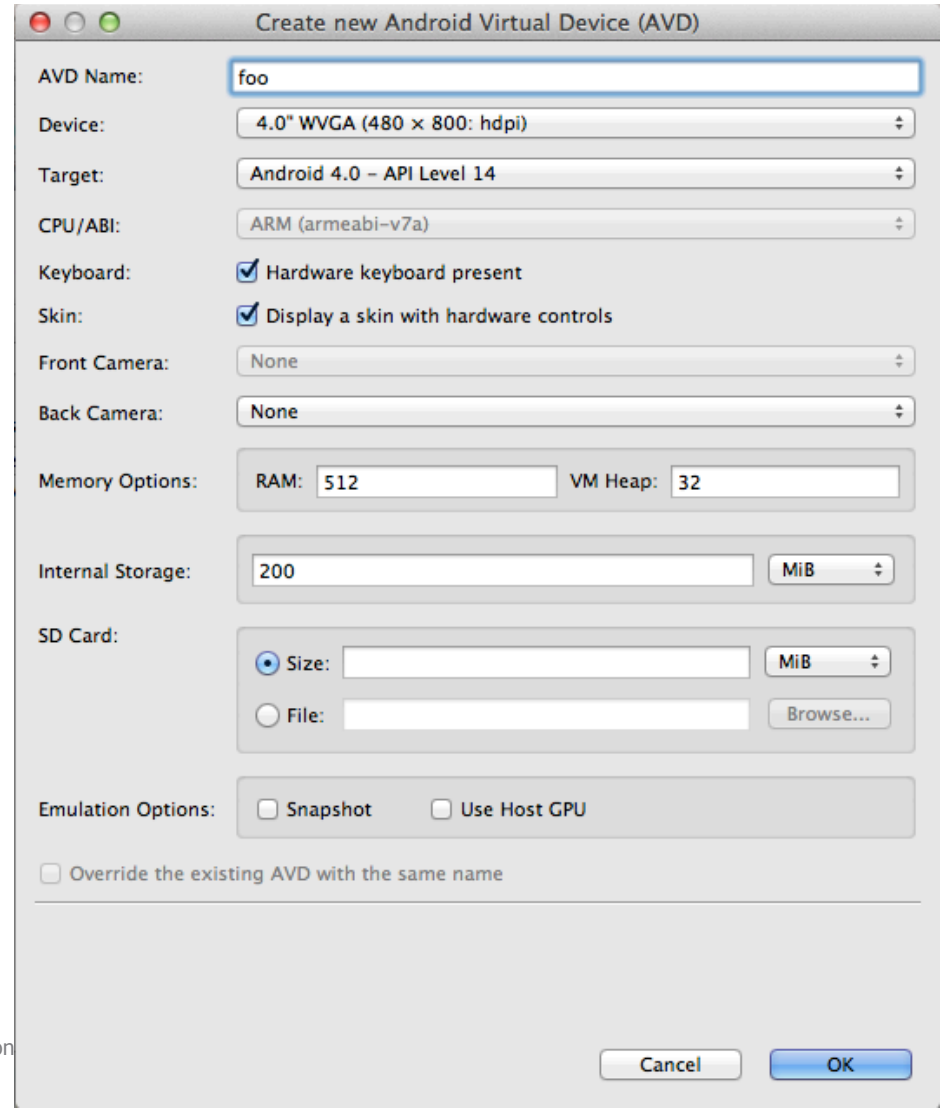
# Create a virtual device

Window > Android Virtual Device Manager



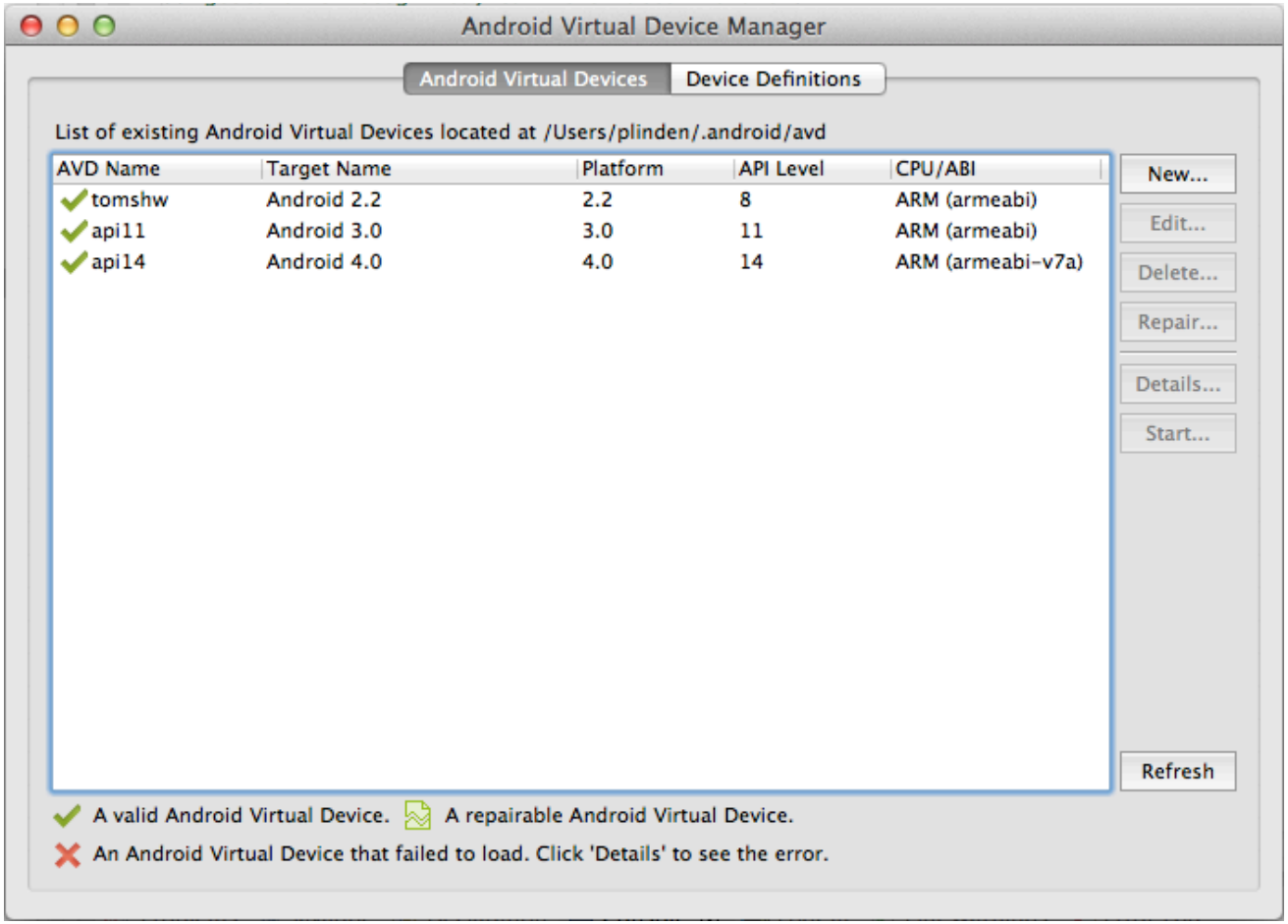
# Create a virtual device

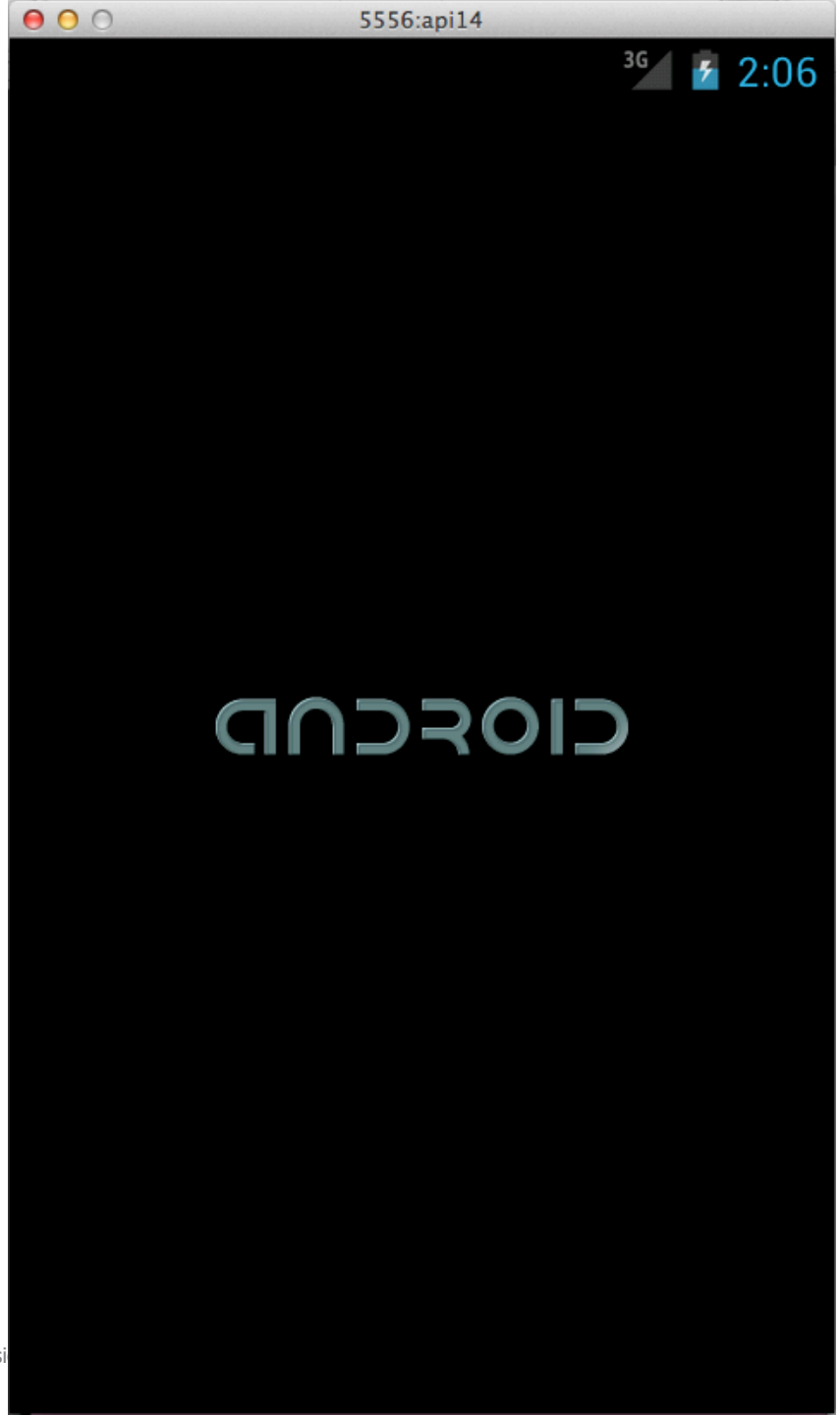
The only config that really matters is the platform API level.  Here API 14.



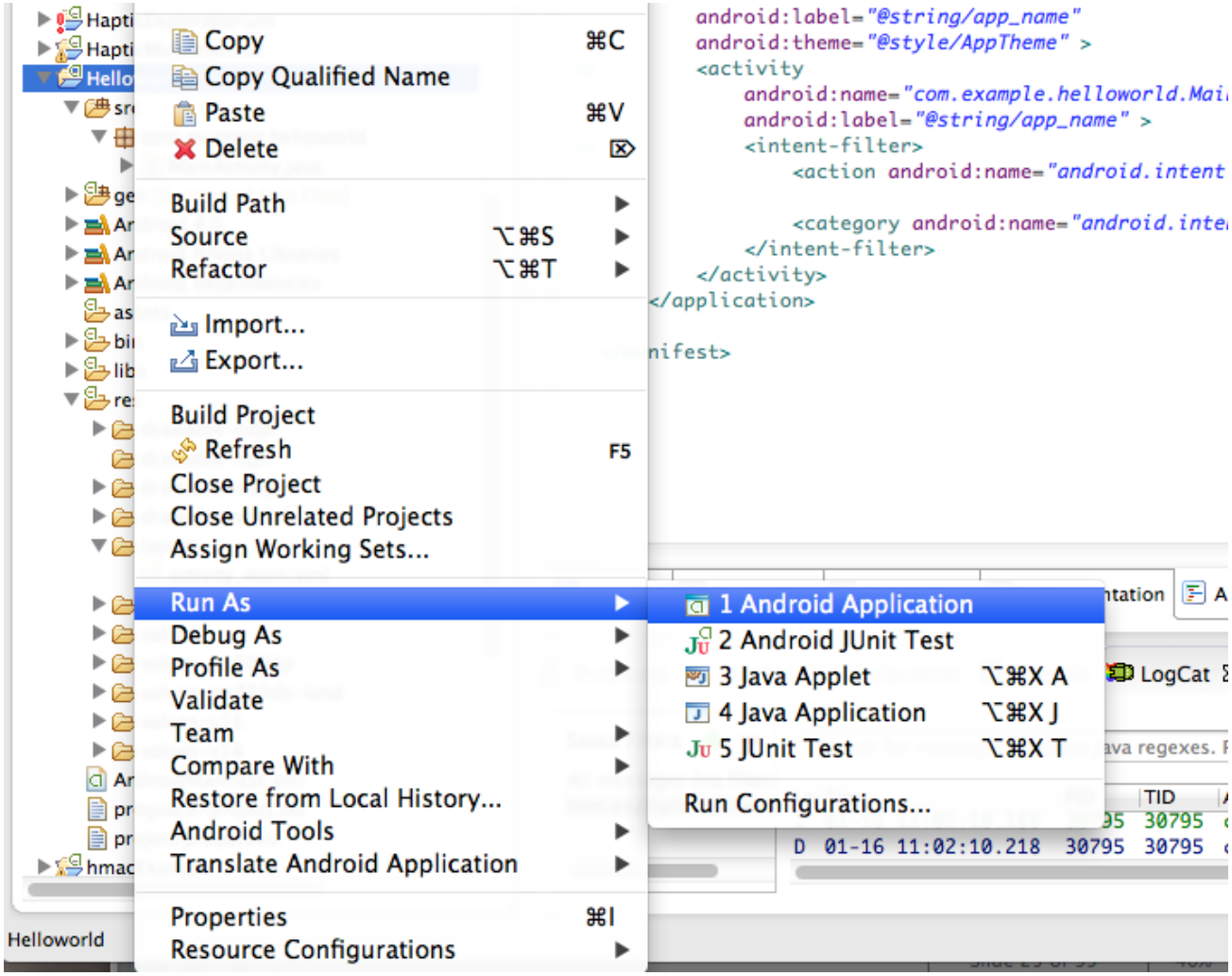
# Create a virtual device

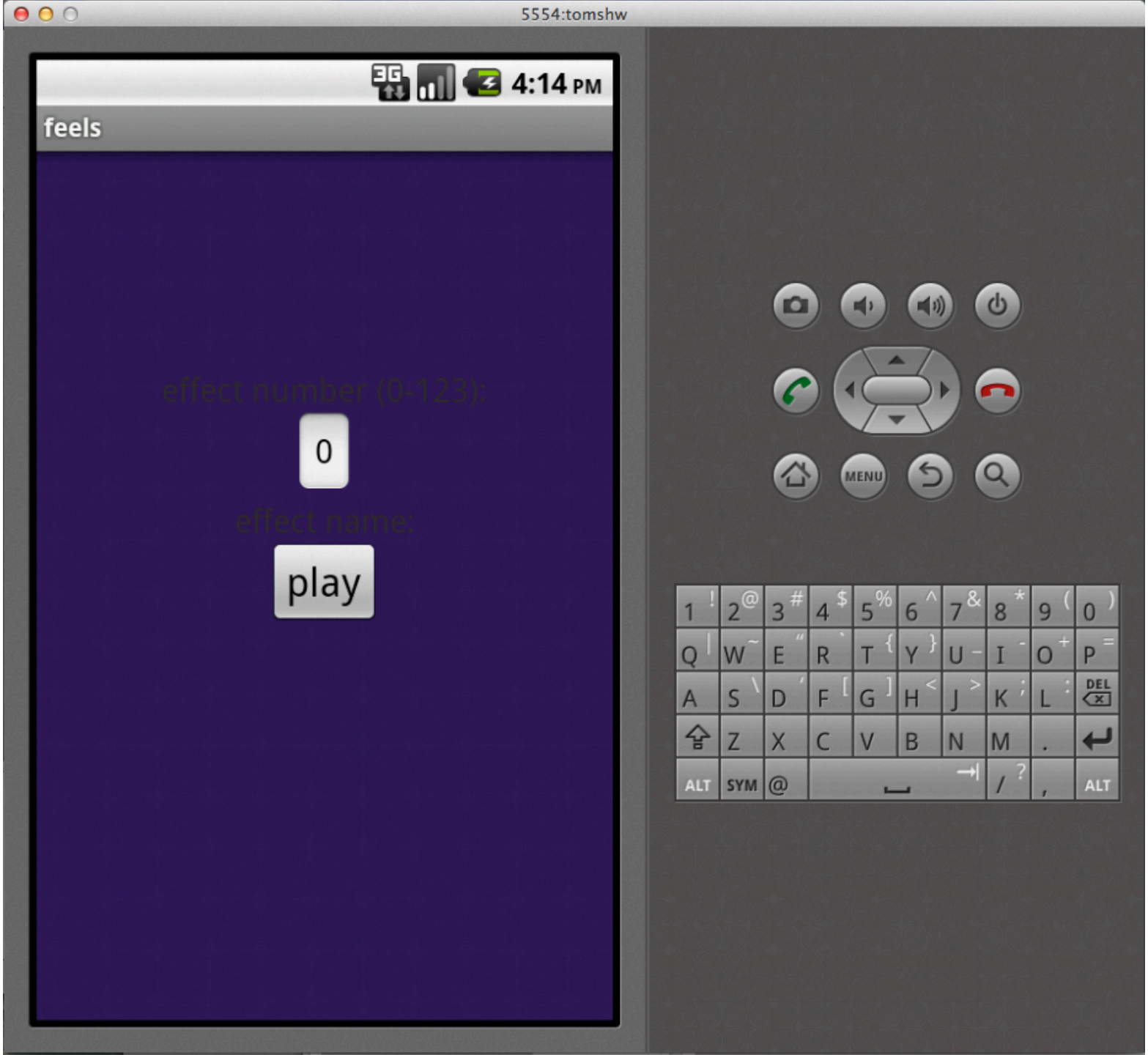
Window > Android Virtual Device Manager





# Run app on phone / tablet




5554:tomshw
4:14 PM
feels
effect number (0-123):
0
effect name:
play
MENU
ALT
SYM
ALT

# Well Done!

- Well done!
- We're done!
- Q & A welcome

# Connect with Immersion

#HapticsDev

like “ImmersionDeveloper”

search “Immersion Corporation”

# Some great Android resources

- http://developer.android.com
- http://developer.immersion.com
- http://stackoverflow.com
- Web search for keywords “Android notification tutorial”
- Have a great time with this!