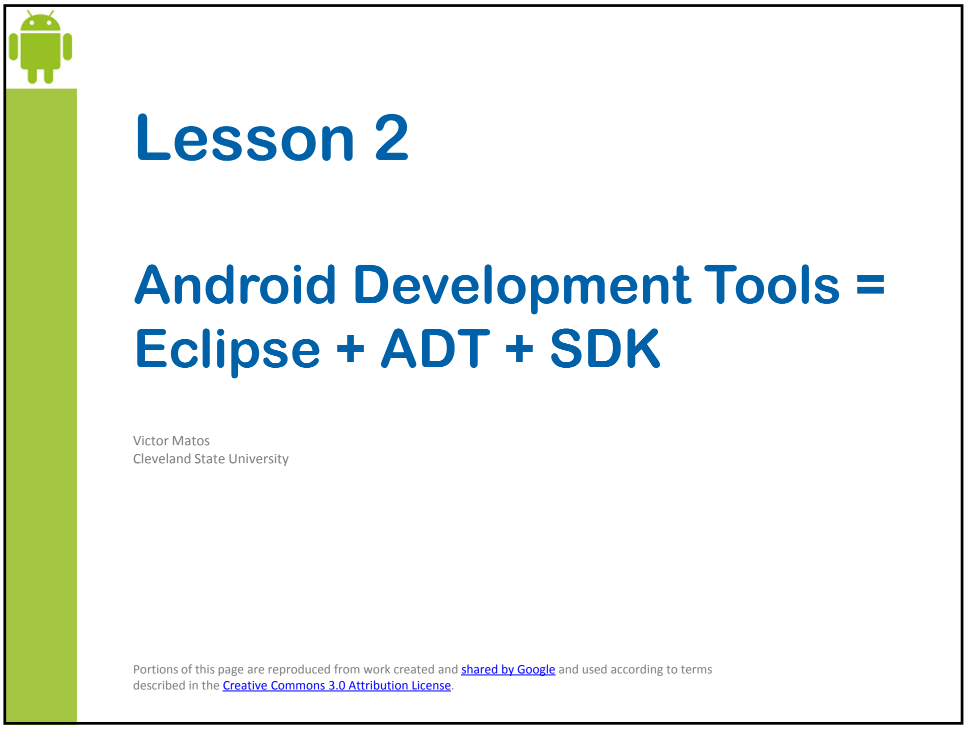
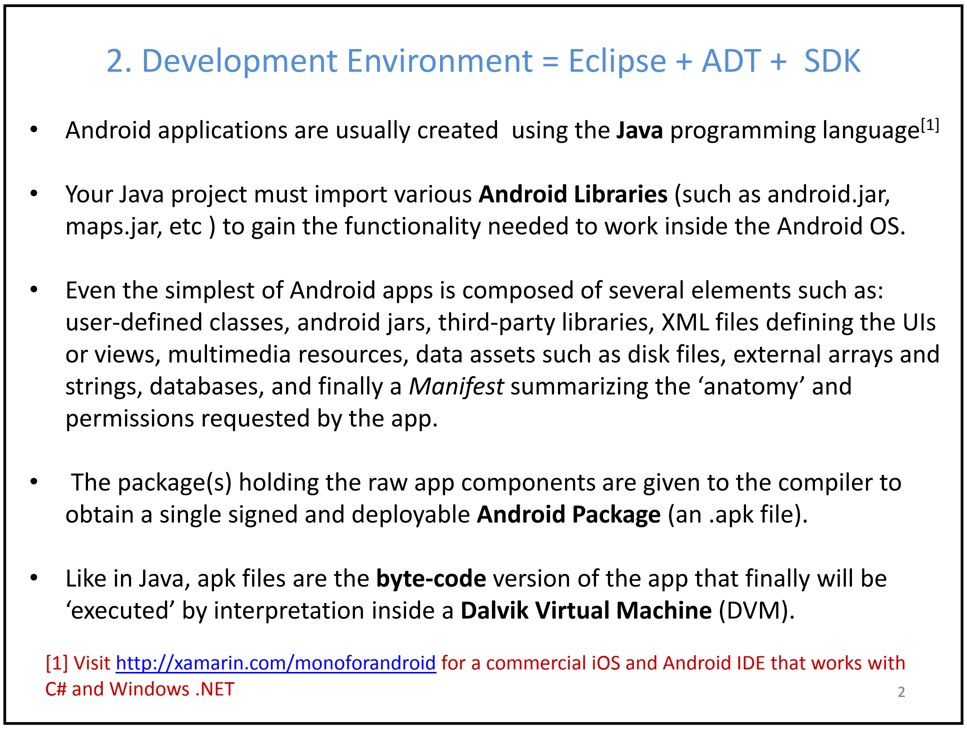


Lesson 2

Android Development Tools = Eclipse + ADT + SDK

Victor Matos
Cleveland State University



2. Development Environment = Eclipse + ADT + SDK

- Android applications are usually created using the **Java** programming language[1]
- Your Java project must import various **Android Libraries** (such as android.jar, maps.jar, etc) to gain the functionality needed to work inside the Android OS.
- Even the simplest of Android apps is composed of several elements such as: user-defined classes, android jars, third-party libraries, XML files defining the UIs or views, multimedia resources, data assets such as disk files, external arrays and strings, databases, and finally a *Manifest* summarizing the 'anatomy' and permissions requested by the app.
- The package(s) holding the raw app components are given to the compiler to obtain a single signed and deployable **Android Package** (an .apk file).
- Like in Java, apk files are the **byte-code** version of the app that finally will be 'executed' by interpretation inside a **Dalvik Virtual Machine** (DVM).

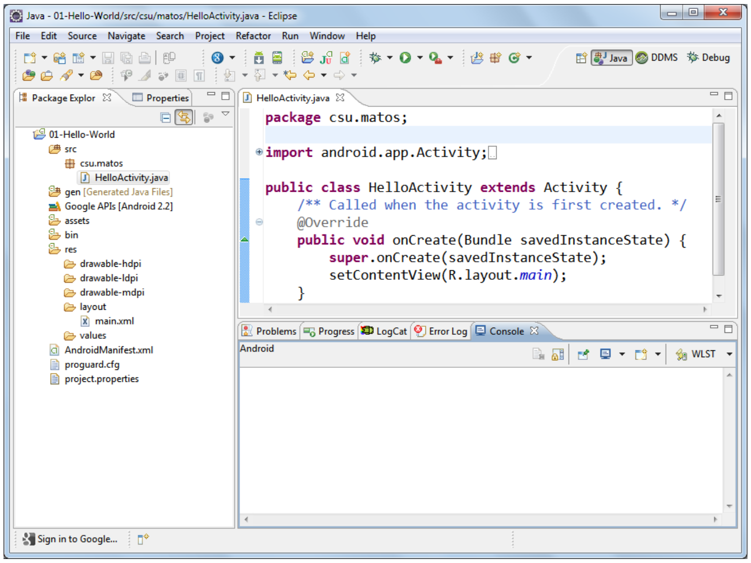
[1] Visit http://xamarin.com/monoforandroid for a commercial iOS and Android IDE that works with C# and Windows .NET

2. Development Environment = Eclipse + ADT + SDK

- Creating, organizing and managing the components of an Android app is better done using a ‘friendly’ workbench.
- The Android developer’s workbench typically includes the following tools:
  1. Eclipse IDE
  2. Android Development Tools (ADT), and
  3. Android System Development Kit (SDK)
- **Eclipse IDE** allows you to create and debug your Java code, and manage the various resources that normally are used in the making of an Android app.
- The **ADT plugin** extends Eclipse so you can easily reach the tools of the SDK through the use of menus, perspectives and icons seamlessly integrated in the Eclipse’s IDE.
- The **SDK** contains tools needed to transfer, profile, emulate, observe, and debug your applications which could run into any virtual or physical Android device.

2. Development Environment = Eclipse + ADT + SDK

Typical Layout of the Eclipse IDE for Android Development

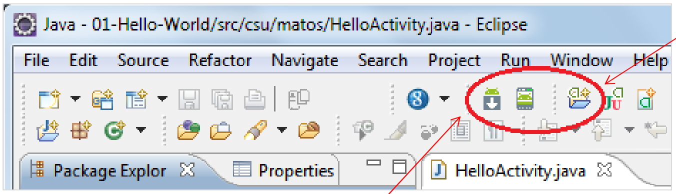


```
package csu.matos;

import android.app.Activity;

public class HelloActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

2. Development Environment = Eclipse + ADT + SDK

Typical Layout of the Eclipse IDE for Android Development
(details…)

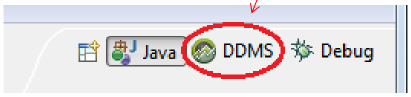
These icons are added to Eclipse by the ADT plugin



Opens Android SDK manager

Opens Android AVD Virtual Device Manager

Wizard creates a new Android Project



Opens DDMS Perspective
Dalvik Debugging Monitoring System

***Note**: The **DDMS** and **Hierarchy View** can be manually added by the user to Eclipse's tool bar*

2. Development Environment = Eclipse + ADT + SDK

SETUP
Prepare your computer – Install SDK: Windows, Mac, Linux

We assume you have already installed the Java JDK and Eclipse IDE in your computer

- Java JDK is available at:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Eclipse IDE for Java EE Developers is available at:
  http://www.eclipse.org/downloads/

The next instructions are given to:

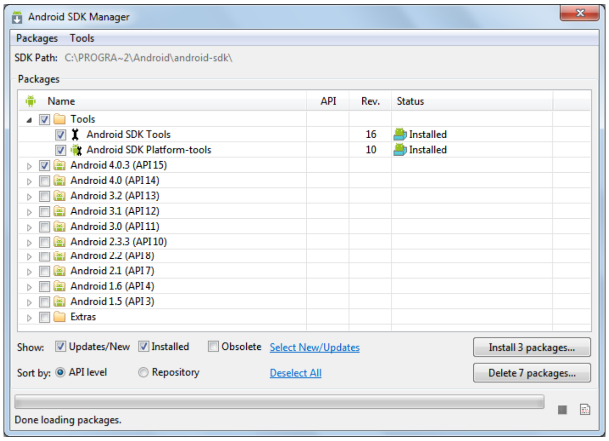
(a) User Wanting to Update their Older Android Workbench,
(b) First Time Users.

2. Development Environment = Eclipse + ADT + SDK

Aside Note:

SDKs are named after a dessert item. Available versions at the time of writing are:

1.5 Cupcake,
1.6 Donut,
2.1 Eclair,
2.2 Froyo,
2.3 Gingerbread [1],
3.x Honeycomb,
4.x Ice Cream Sandwich



Android SDK Manager

Packages Tools

SDK Path: C:\PROGRA~2\Android\android-sdk\

Packages

Name	API	Rev.	Status
☑ Tools			
☑ Android SDK Tools		16	Installed
☑ Android SDK Platform-tools		10	Installed
☑ Android 4.0.3 (API 15)			
□ Android 4.0 (API 14)			
□ Android 3.2 (API 13)			
□ Android 3.1 (API 12)			
□ Android 3.0 (API 11)			
□ Android 2.3.3 (API 10)			
□ Android 2.2 (API 8)			
□ Android 2.1 (API 7)			
□ Android 1.6 (API 4)			
□ Android 1.5 (API 3)			
□ Extras			

Show: ☑ Updates/New ☑ Installed □ Obsolete Select New/Updates

Sort by: ● API level ○ Repository Deselect All

Install 3 packages... Delete 7 packages...

Done loading packages.

[1] By March 2012 Gingerbread accounted for approximately 66% of the Android market share.
See page: http://www.appbrain.com/stats/top-android-sdk-versions

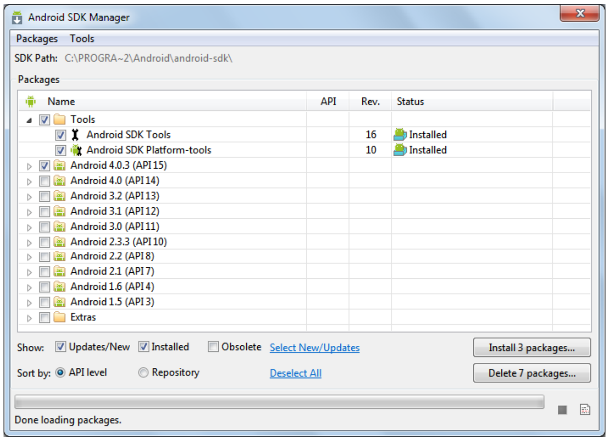
2. Development Environment = Eclipse + ADT + SDK

SETUP

(a) Users Wanting to Update an Older Android Workbench

If you are currently using the Android SDK, you just need to *update* to the latest tools or platform using the already installed *Android SDK and AVD Manager.*

1. Click on the *SDK Manager* icon.
2. You will see a form similar to the one on the right.
3. Select the Packages you want to install and wait until they are setup in your machine.



Android SDK Manager

Packages Tools

SDK Path: C:\PROGRA~2\Android\android-sdk\

Packages

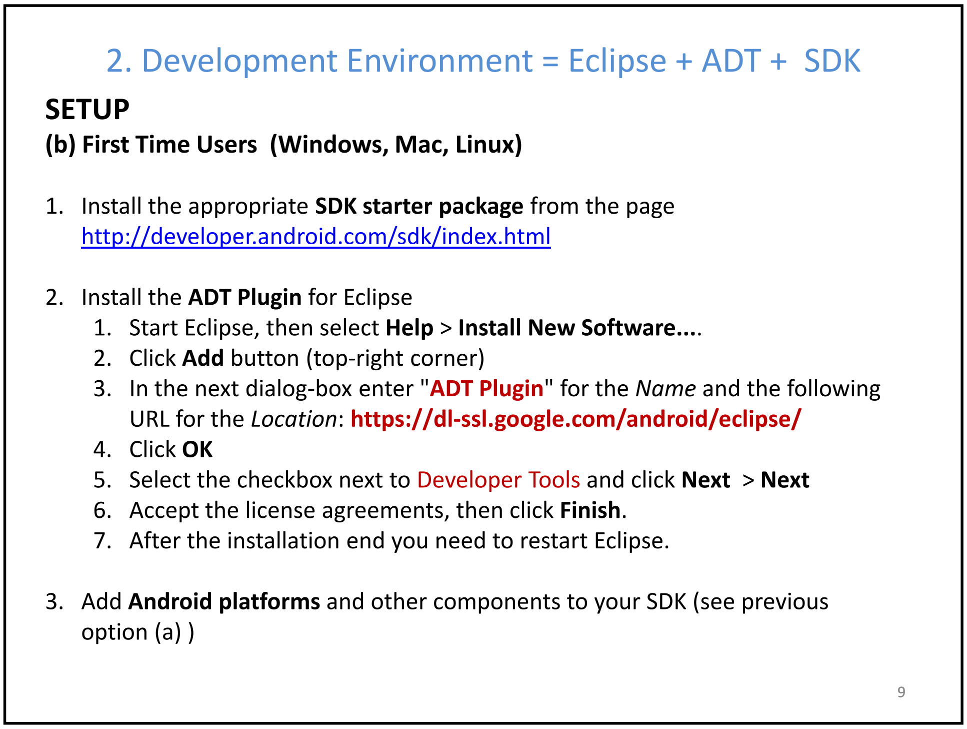
Name	API	Rev.	Status
☑ Tools			
☑ Android SDK Tools		16	Installed
☑ Android SDK Platform-tools		10	Installed
☑ Android 4.0.3 (API 15)			
□ Android 4.0 (API 14)			
□ Android 3.2 (API 13)			
□ Android 3.1 (API 12)			
□ Android 3.0 (API 11)			
□ Android 2.3.3 (API 10)			
□ Android 2.2 (API 8)			
□ Android 2.1 (API 7)			
□ Android 1.6 (API 4)			
□ Android 1.5 (API 3)			
□ Extras			

Show: ☑ Updates/New ☑ Installed □ Obsolete Select New/Updates

Sort by: ● API level ○ Repository Deselect All

Install 3 packages... Delete 7 packages...

Done loading packages.



2. Development Environment = Eclipse + ADT + SDK

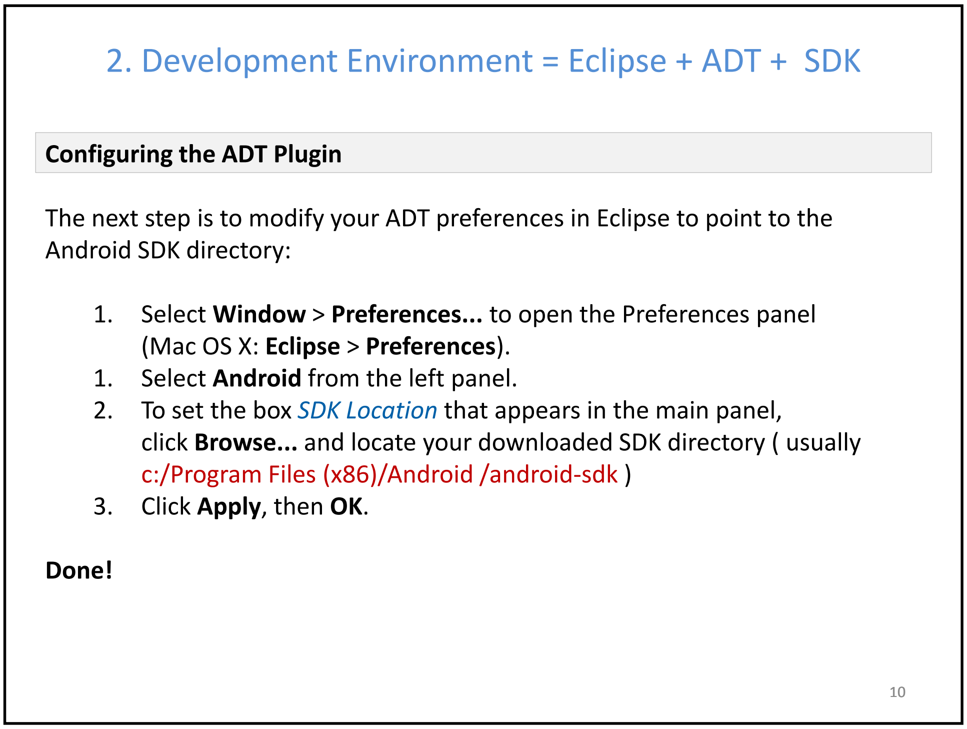
SETUP

(b) First Time Users (Windows, Mac, Linux)

1. Install the appropriate **SDK starter package** from the page http://developer.android.com/sdk/index.html

2. Install the **ADT Plugin** for Eclipse
   1. Start Eclipse, then select **Help** > **Install New Software....**
   2. Click **Add** button (top-right corner)
   3. In the next dialog-box enter "**ADT Plugin**" for the *Name* and the following URL for the *Location*: **https://dl-ssl.google.com/android/eclipse/**
   4. Click **OK**
   5. Select the checkbox next to Developer Tools and click **Next** > **Next**
   6. Accept the license agreements, then click **Finish**.
   7. After the installation end you need to restart Eclipse.

3. Add **Android platforms** and other components to your SDK (see previous option (a))



2. Development Environment = Eclipse + ADT + SDK

Configuring the ADT Plugin

The next step is to modify your ADT preferences in Eclipse to point to the Android SDK directory:

1. Select **Window** > **Preferences...** to open the Preferences panel (Mac OS X: **Eclipse** > **Preferences**).
1. Select **Android** from the left panel.
2. To set the box *SDK Location* that appears in the main panel, click **Browse...** and locate your downloaded SDK directory (usually c:/Program Files (x86)/Android /android-sdk)
3. Click **Apply**, then **OK**.

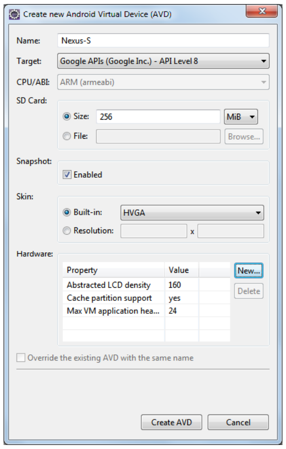
Done!

2. Development Environment = Eclipse + ADT + SDK

Creating an Android Virtual Device (AVD)

You should test your applications on a real phone (or tablet).
However, the SDK allows you to create realistic virtual devices on which your applications could be tested.

1. To create an emulator, click on the AVD Manager
2. Click **New**. The **Create New AVD** dialog appears.
3. Type the name of the AVD, such as "Nexus-S"
4. Choose a target (such as "Google APIs… API Level8").
5. Indicate how much memory the simulator will use.
6. Tick option box "Snapshot" to load faster.
7. Indicate screen size (HVGA is sufficient in general)
8. Optionally specify any additional hardware components (such as SD-card, camer, accelerometer, GPS,…)
9. Click **Create AVD**.



Create new Android Virtual Device (AVD)

Name: Nexus-S

Target: Google APIs (Google Inc.) - API Level 8

CPU/ABI: ARM (armeabi)

SD Card:
● Size: 256 MiB
○ File: Browse...

Snapshot:
☑ Enabled

Skin:
● Built-in: HVGA
○ Resolution: x

Hardware:

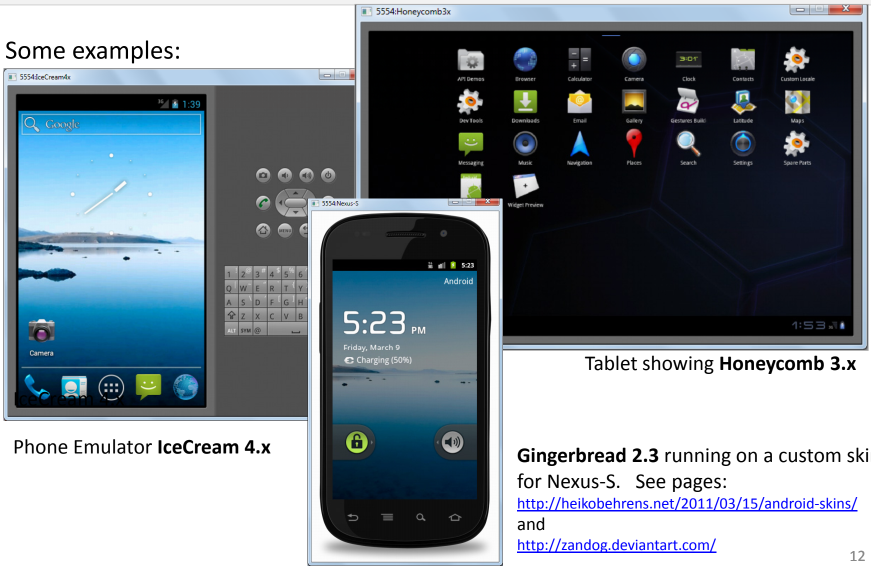
Property	Value
Abstracted LCD density	160
Cache partition support	yes
Max VM application hea...	24

□ Override the existing AVD with the same name

2. Development Environment = Eclipse + ADT + SDK

Creating Android Virtual Devices (AVD)

Some examples:



Tablet showing **Honeycomb 3.x**

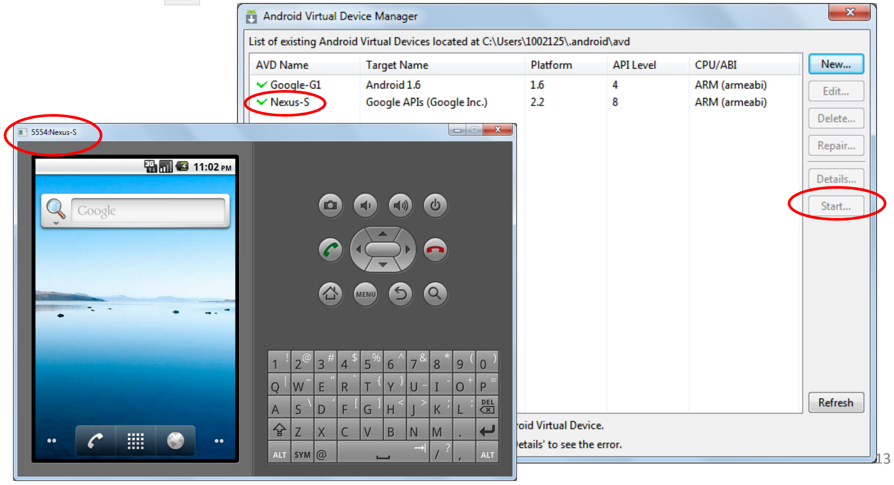
Phone Emulator **IceCream 4.x**

Gingerbread 2.3 running on a custom skin for Nexus-S. See pages:
http://heikobehrens.net/2011/03/15/android-skins/
and
http://zandog.deviantart.com/

2. Development Environment = Eclipse + ADT + SDK

Testing the Emulator

Click on the AVD Manager. Choose an emulator, click **Start.**



Android Virtual Device Manager

List of existing Android Virtual Devices located at C:\Users\1002125\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Google-G1	Android 1.6	1.6	4	ARM (armeabi)
✓ Nexus-S	Google APIs (Google Inc.)	2.2	8	ARM (armeabi)

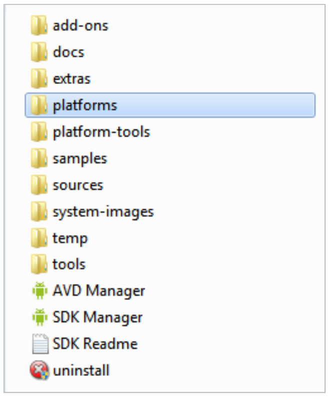
oid Virtual Device.

etails' to see the error.

Android Setup Tutorial

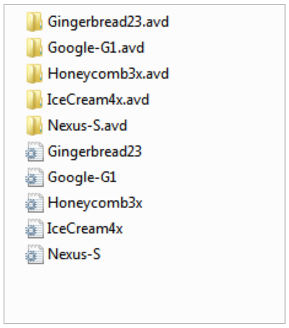
After you complete your setup look for the following two subdirectories in your file system

C:\Program Files (x86)\Android\android-sdk



add-ons
docs
extras
platforms
platform-tools
samples
sources
system-images
temp
tools
AVD Manager
SDK Manager
SDK Readme
uninstall

C:\Users\1002125\.android\avd



Gingerbread23.avd
Google-G1.avd
Honeycomb3x.avd
IceCream4x.avd
Nexus-S.avd
Gingerbread23
Google-G1
Honeycomb3x
IceCream4x
Nexus-S

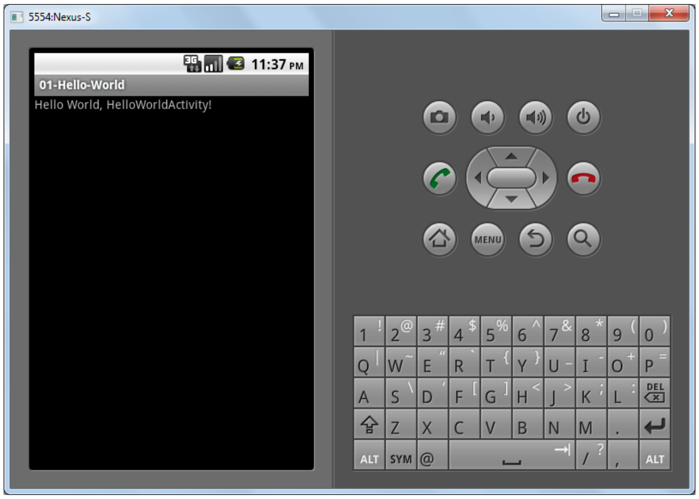
This folder contains your Android SDK, tools, and platforms

This directory holds your Virtual Devices (AVDs)

Testing Setup - Example: Hello World

Appendix. Creating an Android Project (made for SDK2.2 - Froyo)

An unabridged version of “Hello World”



01-Hello-World

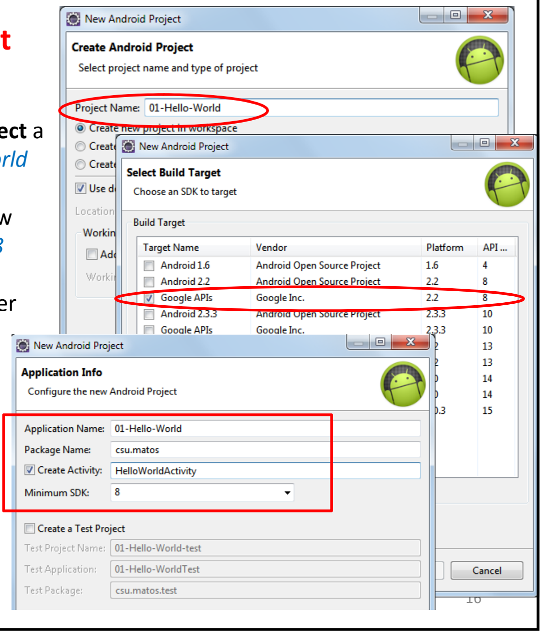
Hello World, HelloWorldActivity!

Testing Setup - Example: Hello World

Creating an Android Project

To create a new project:

1. Start **Eclipse**
2. Select **File** > **New** > **Android Project** a
3. Enter Project name: *01-Hello-World*
4. Click **Next**
5. On Select Build Target choose row *Google APIs Google Inc. 2.2 8*
6. Click **Next**
7. On the *Application Info* form enter
   Pacakage Name: *csu.matos*
   Check box *Create Activity*
   Activity name: *HelloActivity*.
   Min SDK Version: *8*.
   Click *Finish*.



Create Android Project

Select project name and type of project

Project Name: 01-Hello-World

● Create new project in workspace

○ Create

○ Create

☑ Use d

Location

Workin

□ Ad

Worki

Select Build Target

Choose an SDK to target

Build Target

Target Name	Vendor	Platform	API ...
□ Android 1.6	Android Open Source Project	1.6	4
□ Android 2.2	Android Open Source Project	2.2	8
☑ Google APIs	Google Inc.	2.2	8
□ Android 2.3.3	Android Open Source Project	2.3.3	10
□ Google APIs	Google Inc.	2.3.3	10
		2	13
		2	13
		0	14
		0	14
		0.3	15

Application Info

Configure the new Android Project

Application Name: 01-Hello-World

Package Name: csu.matos

☑ Create Activity: HelloWorldActivity

Minimum SDK: 8

□ Create a Test Project

Test Project Name: 01-Hello-World-test

Test Application: 01-Hello-WorldTest

Test Package: csu.matos.test

Testing Setup - Example: Hello World

OBSERVATION: Creating an Android Project using Eclipse

The *New Android Project* Wizard creates the following folders and files in your new project space:

- **src/** Includes your skeleton Activity Java file. All other Java files for your application go here.
- ***<Android Version>/*** (e.g., Android 2.2/) Includes the android.jar file that your application will build against.
- **gen/** This contains the Java files generated by ADT, such as your R.java file
- **assets/** This is empty. You can use it to store raw asset files.
- **res/** This folder holds application resources such as *drawable* files, *layout* files, *string* values, etc.
- **bin/** The bytecode (.apk) version of your app is stored here
- **AndroidManifest.xml** The Android Manifest for your project.
- **default.properties** This file contains project settings, such as the build target.

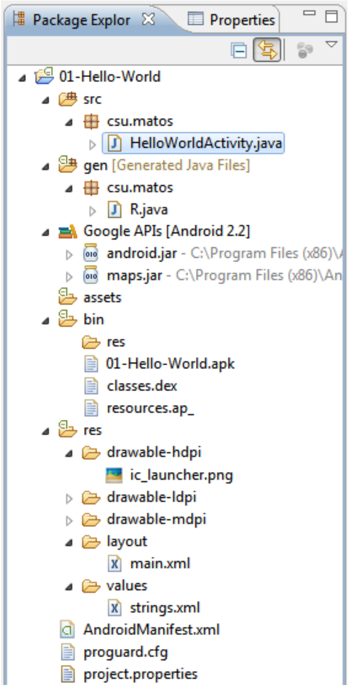
Testing Setup – Example: Hello World

Creating an Android Project

The following folders and files are created for the 01-Hello-World project.

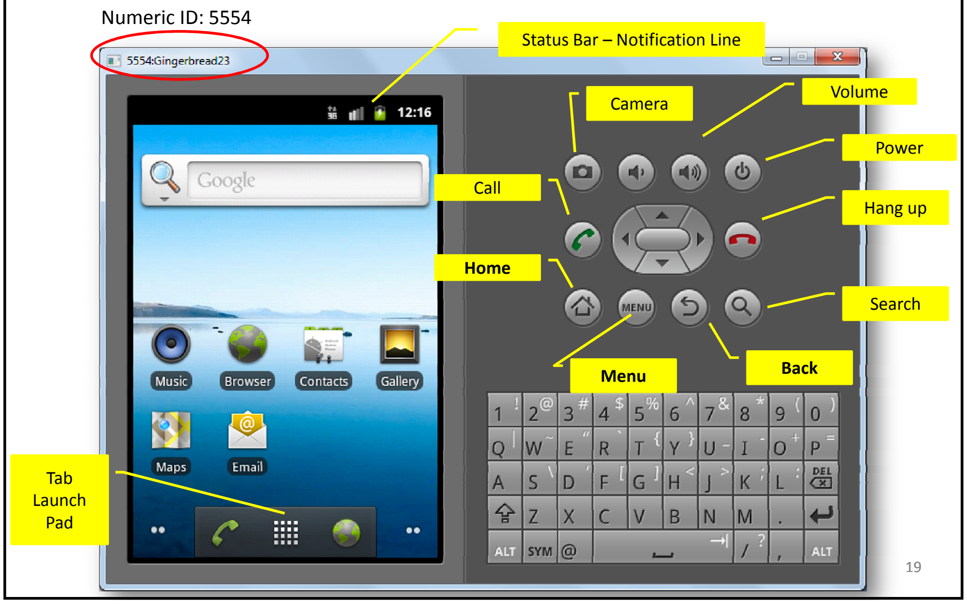
To test the application, position the cursor on the code panel, and then click on the ▶ *Run* menu button.

The fragment of code illustrated on page 4 is executed, and its effect on the emulator is shown on page 12.



```
01-Hello-World
  src
    csu.matos
      HelloWorldActivity.java
  gen [Generated Java Files]
    csu.matos
      R.java
  Google APIs [Android 2.2]
    android.jar - C:\Program Files (x86)\
    maps.jar - C:\Program Files (x86)\An
  assets
  bin
    res
    01-Hello-World.apk
    classes.dex
    resources.ap_
  res
    drawable-hdpi
      ic_launcher.png
    drawable-ldpi
    drawable-mdpi
    layout
      main.xml
    values
      strings.xml
  AndroidManifest.xml
  proguard.cfg
  project.properties
```

Android Emulator (v2.3 skin)



Android Emulator

Keyboard	OS function
Escape	Back button
Home	Home button
F2, PageUp	Menu (Soft-Left) button
Shift-F2, PageDown	Start (Soft-Right) button
F3	Call/Dial button
F4	Hangup / EndCall button
F5	Search button
F7	Power button
Ctrl-F3, Ctrl-KEYPAD_5	Camera button
Ctrl-F5, KEYPAD_PLUS	Volume up button
Ctrl-F6, KEYPAD_MINUS	Volume down button
KEYPAD_5	DPad center
KEYPAD_4	DPad left
KEYPAD_6	DPad right
KEYPAD_8	DPad up
KEYPAD_2	DPad down
F8	toggle cell network on/off
F9	toggle code profiling (when -trace option set)
Alt-ENTER	toggle FullScreen mode
Ctrl-T	toggle trackball mode
Ctrl-F11, KEYPAD_7	switch to previous layout
Ctrl-F12, KEYPAD_9	switch to next layout

Controlling the Android Emulator through (*your computer's*) keyboard keys

Keypad keys only work when *NumLock* is deactivated.