Android Installation Guide

In order to install the Android SDK, you can either follow this link, or do as the following:

Android SDK Installation

• Follow this link to download the SDK. Please notice that the SDK that come with the Phone is 2.0.1. Therefore, DO NOT choose the build target as 2.1 for your program because then you won't be able to install the application on your Droid phone. Use the SDK 1.6.
After you agree to the terms of the SDK license, click on the link that corresponds to your OS (Linux, Mac OS X, Windows) and download the zip file. It is recommended that you download everything on Windows, since our motorola usb driver that needs to be installed on your machine is available for windows (Go to "Motorala USB driver" section for more info).

• Unzip the archive at a suitable location in your machine. We will refer to this location as $ANDROID HOME from now on. By default the files are unzipped into a folder called
android-sdk-<platform>-<release> <build> and has subdirectories tools, samples and docs(Windows versions have another folder for the usbdriver).

– On Linux, edit your .bash profile to add tools to your path
export $PATH=$PATH:$ANDROID HOME/tools
– On Mac, the process is similar to linux. Create a .bash profile if you dont have one already
– On Windows, right-click on "My Computer" and select Properties. Under the Advanced tab, hit the Environment button. Double-click on Path in the window that comes up. Add the full path to the tools directory here.

• Remember that if you update your Android SDK, you also need to update the paths that have been set.

JDK Installation

Download JDK 6 Update 19 from the following link. Please note that you are required to install JDK and not just the JRE.

Eclipse Installation

Download “Eclipse IDE for Java Developers (92 MB)” from the following link. Unzip the downloaded file at an appropriate location.
NOTE: If you are using Linux, it is recommended that you use your package manager (synaptic/apt-get for ubuntu and other debian based distros and yum for fedora and other redhat based distros) to install eclipse. Also ensure that your version is Ganymede or Eclipse 3.4)

Install Android Development Tools

For the android development tools, go to Help -> Install New Software
Click on Add and
Add https://dl-ssl.google.com/android/eclipse/ as URL
Now you will see Android Development Tools in the available software. Select and install this package.
After the installation and restart, you need to update the preferences for ADT. This can be done by going to Window -> Preferences (Eclipse -> Preferences on Mac), select Android. For the SDK Location, click browse and find the location where the android sdk was unzipped. Click Apply and then Ok.
On the toolbar, click on Android SDK and Android Manager,
Then click on Available Packages and select Android SDK 1.6 and Install.

Then in the left menu Virtual Devices -> New. Enter the following
Name: Android-1.6-Emulator
Target: Android 1.6
Size: 64 MB
Then click on Create AVD.

Motorola USB driver

Your operation system needs to install the USB driver so it can recognize the device. The drivers provided here are for windows. If you are using any other OS, you need to look for the proper driver to install.
For 32-bit Windows:
http://infolab.usc.edu/projects/GeoTrust/download/Handset_USB_Driver_32_v4.2.4.msi
For 64-bit windows:
http://infolab.usc.edu/projects/GeoTrust/download/Handset_USB_Driver_x64_v4.2.4.msi

Phone Registration

Each phone needs a Google account. Each group needs to create a google account with the following format.

First name: infolab
Last name: USC
Account: infolabXX.usc@gmail.com
Password: infolabXX
XX stands for 2 digits number which is assigned to your phone.
Secondary email: infolab.usc@gmail.com

Hello World Application

In this project (here is also a link), we will learn to create our first application in Android - outputting “Hello World!” on the screen of an Android screen.
Outline
• Create a new project
• Fill out the project details
• Edit auto-generated source code template to display some output

Creating the Project

• Create a new Android project via the File -> New -> Project menu item. If the Android Plugin for
Eclipse has been successfully installed, the resulting dialog should have a folder labeled “Android” which should contain a single entry “Android Project’.
Select “Android Project” and click Next.

• Fill out the project details.

• Edit the auto-generated source code
After the plugin runs, you’ll have a class named HelloAndroid (found in your package HelloAndroid > src > com.android.hello). It should look like this:

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

```
}
}
**********************************************************************
```

This compiles right away, but lets see what's happening underneath.

Construct the UI

Take a look at this revised code, below, and make the same changes to your HelloAndroid.java file. We'll dissect it line by line:

```
**********************************************************************
package com.android.hello;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class HelloAndroid extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
TextView tv = new TextView(this);
tv.setText("Hello, Android");
setContentView(tv);
}
}
**********************************************************************
```

Tip: If you forgot to import the TextView package, try this: press Ctrl-Shift-O (Cmd-Shift-O, on Mac).
This is an Eclipse shortcut to organize importsit identifies missing packages and adds them for you.
In Android, user interfaces are composed of hierarchies of classes called Views. A View is simply a drawable object, such as a radio button, an animation, or (in our case) a text label. The specific name for the View subclass that handles text is simply TextView.
Here's how you construct a TextView:

```
**********************************************************************
TextView tv = new TextView(this);
**********************************************************************
```

The argument to TextView's constructor is an Android Context instance. The Context is simply a handle to the system; it provides services like resolving resources, obtaining access to databases and preferences, and so on. The Activity class inherits from Context. Since our HelloAndroid class is a subclass of Activity, it is also a Context, and so we can pass the this reference to the TextView. Once we've constructed the TextView, we need to tell it what to display:

```
**************************************************************************
tv.setText("Hello, Android");
**************************************************************************
```

Nothing too surprising there. At this point, we've constructed a TextView and told it what text to display. The final step is to connect this TextView with the on-screen display, like so:

```
**************************************************************************
setContentView(tv);
**************************************************************************
```

The setContentView() method on Activity indicates to the system which View should be associated with the Activity's UI. If an Activity doesn't call this method, no UI is present at all and the system will display a blank screen. For our purposes, all we want is to display some text, so we pass it the TextView we just created. There it is "Hello, World" in Android! The next step, of course, is to see it running.

Running an application

The Eclipse plugin makes it very easy to run your applications. Begin by selecting the Run -> Open Run Configurations menu entry. Highlight the "Android Application" entry and click the icon in the top left corner (the one depicting a sheet of paper with a plus sign in the corner) or simply double-click the "Android Application" entry. You should have a new launcher entry name "New Configuration". Change the name to something more meaningful like "Hello Android". The plugin will automatically scan your project for Activity subclasses, and add each one it finds to the drop-down list under the "Activity:" label. Since your "Hello, Android" project only has one, it will be the default, and you can simply continue.
Click the "Apply" button. That's it you're done! Click the Run button, and the Android Emulator should start. Once it's booted up your application will appear.