Beyond the Android Emulator: Bonus Chapter

Updated March, 2012

This bonus chapter contains information about loading your application on a real Android device, debugging on the device, and publishing your application to Google Play (the Android Market).

Note: This content was originally written for our 1st edition course, but it still applicable to future editions.

Copyright Notices:

Disclaimer:

Homeschool Programming, Inc, and their officers and shareholders, assume no liability for damage to personal computers or loss of data residing on personal computers arising due to the use or misuse of this course material. Always follow instructions provided by the manufacturer of 3rd party programs that may be included or referenced by our courses.

Note: Any red highlights below are added for emphasis and are not present on the actual pages!

Lesson One: Deploying to a Real Device

Now that you have learned the basics of creating Android applications, you may already have a few great ideas for some killer apps. Once you create your apps, how do you get them to the customer? The best method for getting your applications in front of some users is to put it on the Google Play market. But before you attempt to upload your application to Google Play, you really should test it out on an actual device!

Android Devices

There are a few things to do before you can put your application on a device. The first item of business, obviously, is access to an Android device. In order to test your application, you will need to have an Android device that has at least the minimum version of the Android OS that your application requires. If you have created an application that requires Android 2.3, you cannot test it on a device that is running Android 1.6!

Once you have found a compatible device, you will need to makes some changes to the device settings. On your device, go to the Settings application and choose "Applications". On the Applications screen, enable the check mark in the "Unknown Sources" option. This will allow your device to install "non-Google Play" applications.

You will also need to turn on USB Debugging on the device. This will enable you to debug the application as it is running on a real device. To do this, open the Settings application and click on "Applications" again. This time, you will need to choose "Development" from the menu and then make sure there is a checkmark in the "USB Debugging" option.

USB Drivers

Once you have configured the Android device, you will need a way to connect this device with your computer. The best way to do this is with a USB cable connected between the computer and the device. You will need to locate the USB cable that came with your device (or find any other standard USB cable with the right-sized connector).

If you are using a Mac computer, you can just connect the USB cable from your Mac to the Android device and the Mac will automatically recognize your device. If you are using a Windows computer, the process is a little bit more complicated. Windows computers require the installation of a USB driver that can communicate between your specific device and the Windows operating system. This USB driver can typically be found on the device manufacturer's website. The Android Developer website has a very comprehensive list of device manufacturer website links located here: http://developer.android.com/sdk/oem-usb.html. These links should take you directly to a website that will have some information on downloading the USB driver that is appropriate for your device. You will need to know the exact manufacturer and model of your Android device in order to find the proper USB driver.

Once you find the driver, follow the manufacturer's installation instructions to get the driver installed on your machine. When this process is complete, Windows should recognize your device and you should be ready to go!

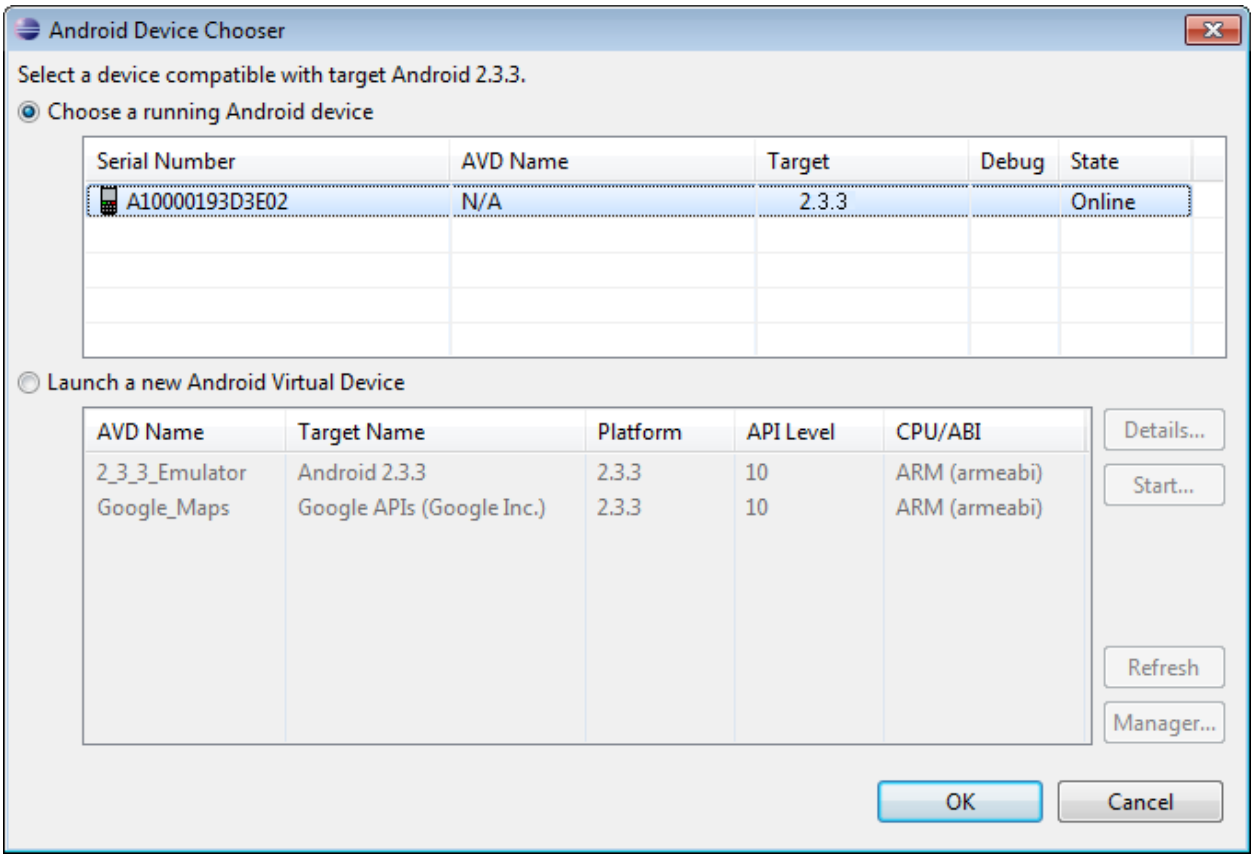
Running the Application on a Device

Now that you have configured your device, installed your USB driver and connected the device to your computer, you are ready to run and debug an application on the device. In Eclipse, first make sure that your application is marked as "debuggable" in the "AndroidManifest.xml" file. Then switch to the DDMS perspective to view the device in the list of available devices. If you do not see your device listed in the DDMS list, you have not connected or configured the USB connection properly.

If you do see the device listed, then you are ready to run your application. You will need to check your Run Configurations to make sure that you will be able to select the real device as the target device when you hit the "Run" button in the Eclipse software. Select "Run → Run Configurations" from the Eclipse menu.

This will bring up the Run Configurations screen. Choose the "Target" tab and then make sure that the "Manual" option is selected in the Deployment Target Selection Mode section and click on "Apply".

Now when you run your application, a screen will appear that will allow you to choose your target device. When this screen appears, choose the real device and then click "Start". This will attempt to install and start the application on your attached Android device.



This is a very powerful tool in performing final debugging on your Android application. When your application is working perfectly on your device, it's time to upload it to Google Play! We'll discuss how to do that in the next few lessons.

Lesson Two: Configuring Application for Publishing

Now that you have finished and tested your killer Android application, it's time to upload it to Google Play and make your millions!

Pre-Build Checklist

Before you build and upload your application, there are some simple checks that you should perform to make sure the application is ready for prime-time.

The first file that you should check is the "AndroidManifest.xml" file. This file requires certain elements in order to be accepted by Google Play. You should open up this file and make sure the following elements are included:

- **android:versionCode** – This is an integer value that represents the current version number for your application. The Android system maintains this value as a number so that it can be checked programmatically by other applications. You should start with a value of 1, and incrementally increase the number with any updates that you provide for the users. This version code value is not typically the same as the version information that the user sees on the screen.

- **android:versionName** – This is the string value that represents the version value that the user sees in the application. This value is typically in the form: <major>.<minor>.<point>, where the major value is the major release number, minor is the minor release number and the point value represents a slight change in versions. So your application could start out with a version name of "1.0.0". Then you could create a slight update and change the version name to "1.1.0" or "1.0.1". The next major release (with big changes to the application) would be version name "2.0.0".

- **android:icon** – This is the icon image for your application. This is the image that will appear on the device's Home Screen, Launcher menu, Manage Applications screen and so on. You will want to make sure this attribute points to the correct image for your application.

- **android:label** – This is the default name for your application, which will appear in the Google Play screen, the Home Screen, and any application list on the device. You should make sure this attribute is set to a short, meaningful name. You don't want to spend all this time on your application only to have it show up in a list as "MyApp"!

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="teencoder.androidprogramming"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".Main" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

In addition to these basic attributes, you will want to make sure that certain other elements are handled correctly in the manifest. First, you will want to turn off the debugging in the application by removing the "debugging=true" line in the manifest file. Then you should take a look at all of your <uses-permission> tags to make sure that they are all still required and/or relevant for your application. You should not request permissions that you program does not need to work properly.

Finally, examine the <uses-sdk> tag in the manifest. This tag should have two main attributes: "android:targetSdkVersion" and "android:minSdkVersion". The targetSdkVersion value should indicate the SDK version number for which that the application was primarily designed. In most of our examples, we were targeting the 2.3.3 version of the Android OS, which corresponds to the SDK version number 10. The "minSdkVersion" on the other hand, is the minimum SDK version on which your application will run. These values do not need to be the same! In fact, it's fairly common to target a higher SDK version and still allow your application to run on an older version.

Google Play will automatically filter the applications that are shown to a user based on the SDK version of their device. If you have a device with Android 1.6 installed, you will not be shown any applications that have a "minSdkVersion" of 5 or above. If you have a device with Android 2.2 installed, you cannot install any applications that have a "minSdkVersion" of 9 or above, and so on.

This means that in order to make your application available to more users in the Google Play market, you may want to lower your "minSdkVersion" number. Just make sure that you are not using any functionality that is not available in the minimum SDK version.

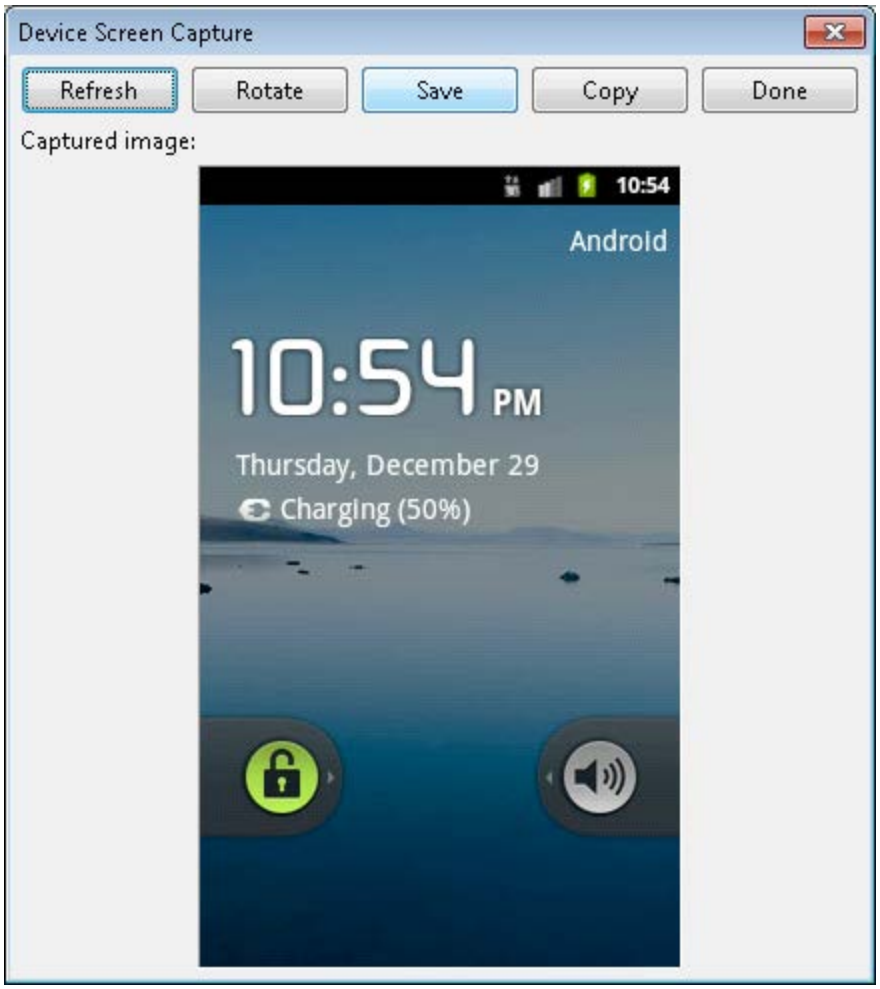
Verifying Images

Images are very important on an Android device. You will want to make sure that you have provided appropriate-sized images for any graphics that your application will display. This includes making images that can be used in any of the available resolutions on the target devices. Making sure your application images will look great on any device is a good way to make an application impressive to your potential users.

In addition to the basic images in your application, you will need to create a special high-resolution application icon that will be displayed in the Google Play application. This icon should be a 512 pixels x 512 pixels version of your application launcher icon, saved as a 32-bit PNG.

You will also need to provide at least two screenshots for your application. These screenshots are used to give the user an idea of what the application looks like before they decide to download and install it from Google Play. It's been proven time and time-again that including at least two good-quality screenshots will result in more downloads and installations of your application.

Your application screenshots can be 320w x 480h, 480w x 800h, or 480w x 854h and should be saved as 24-bit PNG or JPG files with no transparency. You can add up to 8 screenshots for each application. The easiest way to create these screenshots is to run your application in the emulator, and then use the "Screenshot" button in the DDMS perspective.



The Screenshot button is found at the top of the DDMS Perspective and looks like a camera image. When you click on this button, you will see a Device Screen Capture window appear on the screen. Note that it may take a few seconds for your screen image to appear.

If you don't like the image that has been captured, you can go back to your emulator window and set up a new screen. When you are ready to capture the image again, just hit the "Refresh" button in the Screen Capture application.

Once you have that perfect screenshot, you can click on the "Save" button to save the image to your computer. You can also choose the "Copy" button to save the image to the clipboard. Then you can paste it into the image editor of your choice.

Packaging and Signing Application

There is one last step to complete before we can package our application for public use. All Android applications must be digitally signed in order to be placed in the Google Play market. What does this mean? This certification provides some assurance to your users that you are a trusted programmer whose code will not do anything malicious to their devices. In addition, any updates that you offer for your application must have the same digital signature as the original application. This assures the user that only the original developer can change or update their application.

The digital signature consists of a public and private key value. The public key value remains on the application. The private key value remains with the developer. When the user installs an update, the two keys are compared to make sure they match.

So do you need to buy an expensive certification from a third-party company? No! All Android applications are capable of being self-signed. This means that you can create your own certification. In fact, the Eclipse software actually makes this process very easy!

To see full-color, screen-by-screen documentation on signing and packaging your applications, please read the "Signing_Android_Apps.pdf" document from our website.

Lesson Three: Getting Your Application to the User

Publishing to Google Play

Now that you have your application signed and packaged, you are ready to fling it to the eager masses. One of the best ways to get your application in front of a large pool of users is to publish it to the Google Play market. Google Play is a place where users can browse applications, read and write reviews, and install applications. This view is always filtered to show only applications that will works on the connected device. This is why it's important to mark your "minSdkVersion" as low as you can, in order to get your application in front of the most users.

In order to publish your application on Google Play, you will need to create a Google Play account. Please see our "Creating_a_Google_Play_Account.pdf" document from our website for a description of this process.

Free Applications vs. Paid Applications

So now that you have your Google Play account, you will need to make a decision. Do you charge a fee for your application? Or do you offer it for free? There are pros and cons for each option.

If you choose to charge a fee for your application, you will begin making money as soon as your application sells to the first user. This is definitely a pro in the instant-gratification column. However, there are some down-sides to charging for applications. First of all, you will need to create a Google Wallet Merchant account in order to collect the fee from your users. This is a fairly simple procedure, but you will have to provide some personal information to Google. Typically, you will need to provide your Social Security number, a credit card and some bank account information. Google will take care of collecting the payment for you and will send the money to your bank account within 24-48 hours of the sale (minus a small fee, of course!).

Why does it take at least 24 hours to receive your payment? The Google Play market has a 24 hour guarantee for all applications purchased through the market. If the user decides they don't like your application within those 24 hours, they can request a full refund. This means that Google will typically wait this long before processing your fee.

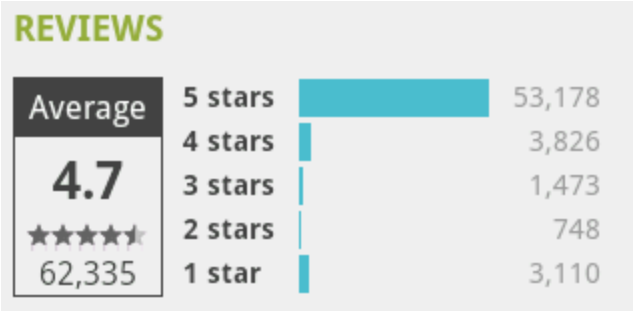
Another downside to paid applications is that they statistically result in far fewer installs than free applications. Users are often wary of spending even a small amount of money on an application made by a relatively-unknown developer.

Free applications, on the other hand, result in many more downloads and installs. Statistics typically show that after a user views your application on Google Play, there is about a 50% - 80% chance that they will download and install the application. So this means many more installs of your application, but how does this work for you, the developer? Most free applications include advertising that pays the developer a fee for each time the ad is displayed or clicked by the user. To add these advertisements, you can sign up for free accounts at places like Google AdSense. They will give you detailed instructions on how to setup the ads within your application. Believe it or not, this free application and advertising model will often make more money than a paid application!

One of the more popular sales models is to offer a free application with advertising and then offer to upgrade the user for a small fee to an application that runs without the advertisements. This allows the user to test out your application for free and upgrade when they like it enough to get rid of those pesky ads!

Viewing Application Progress

Once your application is uploaded to Google Play, you can check its progress frequently. Your customers will tell you if they like your application or if they think it could be improved. Customers will rate your program with a star rating, leave comments about the app and might even offer error reports. This is important information! A responsive developer is one that is highly-successful. Pay attention to what your customers have to say and use this information to learn and to build a better application.



Other Methods of Installing Apps

The Google Play market is not the only way to get your application into the hands of users. You can actually take your APK file and install it directly on a device. In order for this to work, the user must go into the "Settings" application, choose the "Applications" option and then make sure that the "Allow Unknown Sources" option is checked. This will allow "non-Google Play" applications to be installed on the device.

Once this is done, you can distribute your application by putting the APK on an SD card, sending it through email to the device, or by hosting it on your own web server. Once the file is on the device, the user only needs to click on the APK file to automatically launch the install process.

Using this method allows you to offer your application without paying the $25 Google Play fee.

Amazon App Store for Android

A recent addition to the Google Play world is the Amazon App Store for Android. This is Amazon's attempt to enter the lucrative app market for these devices. You can view the store at: http://www.amazon.com/appstore or you can install the Amazon App Store app on your Android device. The App Store is still relatively new, but it is gaining popularity quickly.

You can add applications to the App Store by registering as a developer on the Amazon site. The cost is around $99 per year, but they almost always have some sort of deal where you can get your first year for free. For more information on adding your own applications to the Amazon App Store, you can view their developer page at: https://developer.amazon.com/public/.