MANTECH Publications

Journal of Android iOS Development and Testing
Volume 1, Issue 1, January-April 2016

Setting up Android Development and iOS Environment on Your Computer and Working with Template Code Structure

Shashi Yadav, Rupesh Kumar

Samrat Ashok Technological Institute, Vidisha, M.P.

E-mail: *shashiyadav0049@gmail.com*

Abstract

Setting up the Android and iOS Development Environment in their computer is the first step that developers need to take for learning to develop the respective codes. In this paper, the basics of setting up an Android environment and working with a template code structure are provided for beginners.

By fully grasping information detailed in the different sections of this paper, it will be easier for Android and iOS code developers to understand the basics.

Keywords: *Android Development, Android, iOS, Template Code Structure*

INTRODUCTION

In this paper, you will learn to set up your development environment to be able to work with this paper's tutorials and examples. Then you will learn how to import this paper's existing SDK projects and templates into your favorite IDE, as you will do throughout this paper when following the different tutorials. And finally, this paper concludes with a quick tutorial that will help you to get familiar with the events of the template.

SOFTWARE REQUIREMENTS

This paper's content is built to run on iOS 5.x+ as well as for Android 2.x+, the latest and most stable versions of these two mobile operating systems at the time this paper was written.

For iOS Developers

To use this paper for iOS, all you have to do is to grab a copy of the latest iOS SDK available at http://developer.apple.com, and install it on your Mac. Out-of-the-box the iOS SDK provides a simulator with full GLES v2 support, so even if you do not have an iOS device, or do not have an

official iOS Developer Certification from Apple, you can still make full use of this paper.

For Android Developers

To set up your environment for Android, it is unfortunately not as easy as for iOS. First go to http://developer.android.com/sdk/installing.html and follow the instructions to install the Android SDK, Eclipse, and the ADT plug-in. Please note that the Android SDK version used for this paper was v2.3.4, but later versions should also work as well.

All the code in this paper uses C/C++, which means that you will have to install Android Native Code support. To fi nalize the installation of your development environment, follow these steps:

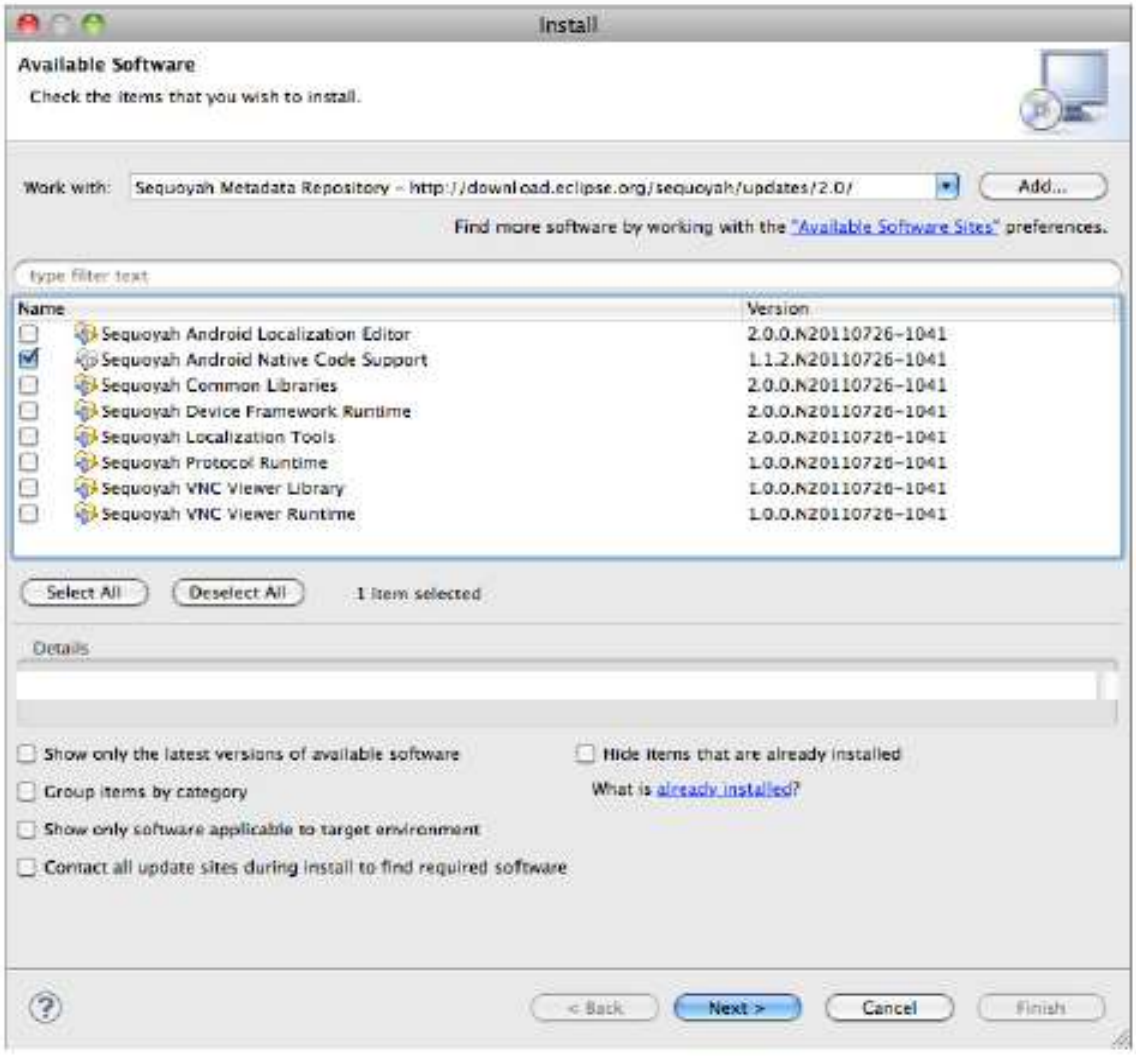

FIGURE 1-1: *Sequoyah Native Code Support plug-in*

1. Grab a copy of the Android NDK at the following address: http://developer.android.com/sdk/ndk/index.html. The version used at the time of writing this paper was r5c, but all examples and tutorials should work on later versions as well. Download the Android NDK zip package and decompress it on your machine where you have read and write access.

2. In order to compile and debug native code using Eclipse, you will need to install the Sequoyah plug-in. To do this, fi rst enable the repository that is located (from the Eclipse main menu) in:

   Help ➪ Install New Software ➪ Available Software Sites ➪ Sequoyah Metadata Repository. Then select the entry from the Work With combo box, and once the repository data is loaded, select and install the Sequoyah Android Native Code Support, as shown in Figure 1-1.

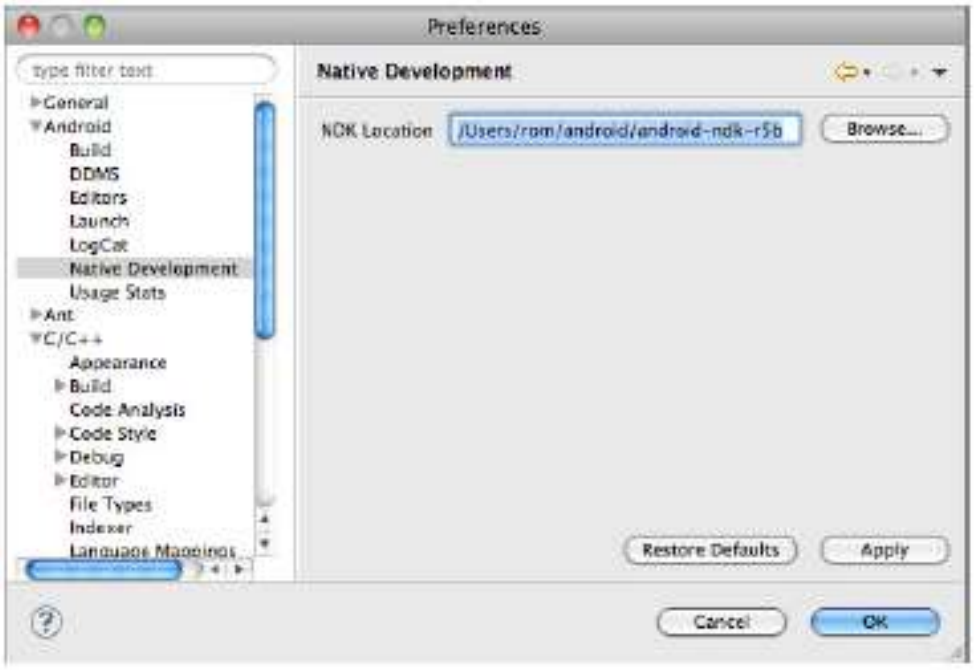

FIGURE 1-2: *Specify the location of the Android NDK*

3. Once Sequoyah is installed, go to (from the main menu): Eclipse Preferences ➪ Android ➪ Native Development and specify the location where you extracted the Android NDK in step 1, as shown in Figure 1-2.

Our Android development environment is now all set! However, please note that in order to use this paper with Android you will need an actual device with OpenGL ES 2.0 support. The emulator provided by the Android SDK supports only OpenGL ES 1.x, not OpenGL ES 2.0. So local deployment on the simulator is not possible on Android; only device deployment is supported when using GLES 2.

IMPORTING FILES AND PROJECTS

To be able load and rebuild the projects from elsewhere into your IDE, you will have to import them. To do this, just follow the instructions in the subsection that corresponds to the type of developer you are.

For iOS Developers

As usual for iOS developers, importing files is very easy. All you have to do to import a project into XCode is simply double-click the .xcodeproj file. To compile, simply click the Build & Run button.

For Android Developers

Things are a little bit more tedious if you're using Eclipse. You need to import this paper's projects as instructed in the following procedure. Of course, this procedure assumes that you have properly installed and configured Android SDK, Android NDK, Eclipse Classic, the ADT plug-in, and the Sequoyah Android Native Development plug-in.

Once you have configured all the necessary prerequisite files, follow these steps to import this paper's project files:

1. From the Eclipse main menu, select File ➪ New ➪ Android Project. The New Android Project dialog should appear.

2. In the Project name text box, enter the project name. Example: paper2-1.

3. Select the Create Project From Existing Source option.

4. Click the Browse button, and then select the existing Android directory inside the paper or template project. Example: /SDK/_paper2-1/Android.

5. Click the Finish button at the bottom of the dialog box. Figure 1-4 illustrates each of these steps.

Every time you want to open an existing Android project using Eclipse, you will have to go through this importing procedure.

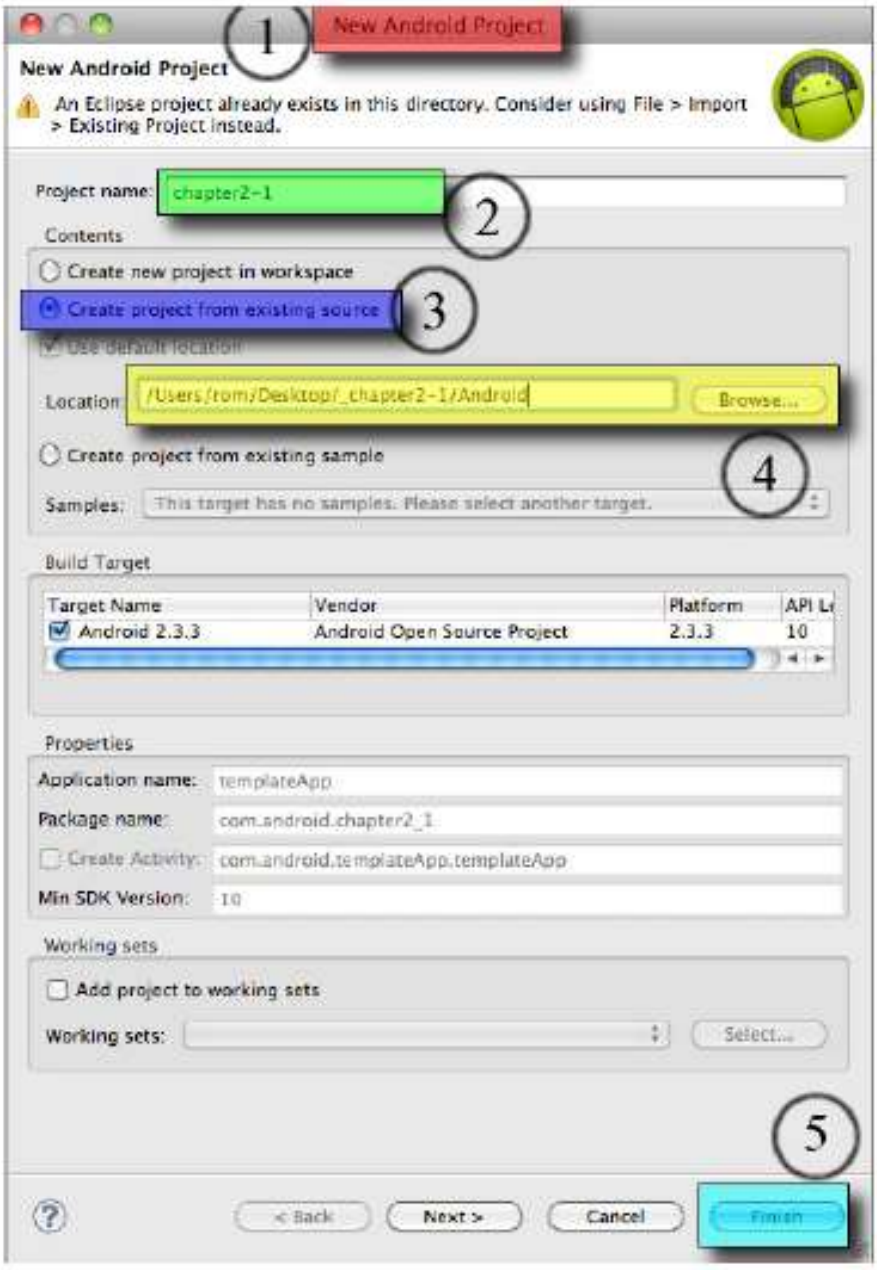

Figure 1-4: Importing an Android project into Eclipse

THE TEMPLATE

As briefly y mentioned earlier in this paper, you will work mostly with the template project that is provided inside this paper's SDK. This template is a C/C++ cross-platform project that initializes internally for you a vanilla, ready-to-use OpenGLES 2 context. In addition, the template provides an init and exit function callback, which you can just plug your creation and destruction code into.

The template also provides you with an easy-to-use callback mechanism that acts as a universal HUB to handle all the platform-specific c events for you. Using this mechanism, all you have to do is to link a function callback for the specific c event you want to intercept, and you'll receive updates for this event in real time. This mechanism covers all of the touche events such as ToucheBegan, ToucheMoved, ToucheEnded, as well as the accelerometer data.

In other words, everything is already set up for you. You can just go ahead and create the code as instructed in this paper's tutorials without having to worry about platform-specific issues. As the title of this paper says, it's time to get started! In order to get familiar with both the template and the type of tutorials you will be studying throughout this paper, follow these instructions:

1. Duplicate the template project directory at the root of the SDK and rename it template_test.

2. Load the template_test project (following the appropriate importing method for your platform as described previously) into your IDE, and then open the templateApp.cpp (for iOS developers, it is located under the templateApp directory inside the Project Navigator; for Android developers, you can fi nd it under the jni directory inside the Project Explorer panel).

3. Read the code comments that explain what each function is doing.

4. Uncomment the following callbacks from the initialization (TEMPLATEAPP templateApp = {}): templateAppToucheBegan, templateAppToucheMoved, and templateAppToucheEnded.

5. Move to the templateAppInit function and add the following code on the line before the end bracket of the function:

/* Use the built-in GFX cross-platform API to print on the console (XCode) or LogCat (Eclipse) that the execution pointer passes the templateAppInit function. */ console_print(“templateAppInit, screen size: %dx%d\n”, width, height);

6. On the line before the end bracket of the templateAppDraw function callback, add the following code block:

/* Specify that you want to use a chili red color to clear the screen and spice up your app. */ glClearColor(1.0f, 0.0f, 0.0f, 1.0f); /* Report that the execution pointer was here. */ console_print(“templateAppDraw\n”);

7. Add the following line before the end bracket of the templateAppToucheBegan function:

/* Print that the execution pointer enters the touche began function and print the touche XY value as well as the number of taps. */ console_print(“templateAppToucheBegan,” “touche: %f,%f” “tap: %d\n”, x, y, tap_count);

8. Repeat the same procedure as in step 7 for templateAppToucheMoved and templateAppToucheEnded, updating the console_print text with the appropriate callback function you are dealing with.

9. Move on to the templateAppExit function that has already been linked to the atexit built-in C function, and add the following line before the end bracket of the function: console_print(“templateAppExit...\n”);

10. Build and run the application. While the application is running, observe the console or LogCat (depending on which platform you are developing for). Touch the screen, move your finger around, and monitor in real time on the console how and in which sequence events are triggered internally.

CONCLUSION

By stepping through this paper, you now have your development environment set up. You have this paper's SDK resident on your drive and have learned how to fi nd your way around its architecture. You now know how to import new or existing projects into XCode or Eclipse, and have a good overview of what the default template project can do for you. You are now ready to embark on a very challenging journey in game and graphics programming. Before moving on to setting up your Android or iOS environement, make sure that you fully understand what has been covered in the different sections of this paper.

REFERENCE

1. Game and Graphics Programming for iOS and Android® with OpenGL® ES 2.0, by Romain Marucchi-Foino.

2. Learning Mobile App Development A Hands-on Guide to Building Apps with iOS and Android, by Jakob Iversen and Michael Eierman

3. Android Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides), By: Bill Philips & Brian Hardy

4. Android vs. iOS: Comparing the Development Process of the GQueues Mobile Apps, 1st ed. 2011 Edition, by Jeff Friesen and Dave Smith

5. Learning Mobile App Development, A Hands-on Guide to Building Apps with iOS and Android, by Jakob Iversen and Michael Eierman