Summation
May 2010, pp. 1-4
http://ripon.edu/macs/summation

Android Application Development and Implementation – 3 Dimensional Tic-Tac-Toe

Danial C. Hanson
Department of Computer Science – Ripon College

***Abstract*—This document discusses the Android Mobile Phone Operating System, and the processes involved with developing, implementing and deploying an application for it. Related subjects also covered include monetary opportunities, and androids relationship to java and xml.**

***Key Words*—Android Operating System, *Eclipse*, Mobile Phone, XML**

I. Introduction

The idea of installing a robust operating system on a mobile phone the size of a deck of cards would have been preposterous not long ago. Today it is commonplace and gaining ground. Android is Google's Open Source Mobile Software Environment. It consists of a Linux-kernel based operating system with the underlying code written in C and C++. It uses the Dalvik Virtual Machine to implement programs written in the 'Android' language which is a user-friendly combination of Java and XML. It's important to note that Android is not a hardware platform. For instance, there is no such thing as an 'Android Phone' only 'phones that can utilize the Android Software Environment'.

Developing applications for the Android Mobile Phone Operating System does not require as much preliminary work or in depth learning as one might initially assume. A working knowledge of programming and a willingness to try new things are the two main prerequisites. However, as with anything, it must first be understood just what it is that one is developing for.

Android is not difficult to pick up for anyone with a working knowledge of programming (this is especially true for users with prior java development experience). It is also very affordable to market, and holds many entrepreneurial opportunities for developers interested in taking advantage of an entirely open and new market. The development kit is free of charge and a twenty-five dollar license grants any developer freedom to publish as many applications as they would like on the world-wide 'Android Market'. Monetary opportunities consist of selling an application outright, offering a free 'lite' version and charging for the full version, or working with a program such as 'AdMob' which monopolizes a small portion of the applications screen space, but pays for the opportunity to do so.

The goal of our project was to create a three-dimensional four by four by four tic-tac-toe application with an attractive and accessible user interface and an intelligent opponent. At the time of this project's inception, we had already begun development of such a program in C++, so our plan was to complete said program development, and work to make the finished product available on Android compatible devices by converting it to the Android language.

This article will discuss the intricacies of Android application development and implementation including the *Eclipse* Development Environment, initializing a new Android Program, the steps carried out in turning a new 'blank canvas' into a fully functional Android application and finally how to test that program and make it available to use on Android capable handsets. We will also discuss the processes involved in the development of a 3-Dimensional Tic-Tac-Toe program for Android and things to watch out for when developing Android Apps.

II. The *Eclipse* Development Environment

When developing the Android programming language and a corresponding development environment, Google worked closely with the existing Java Software Development Environment *Eclipse*.

Currently in their 'Galileo' release, *Eclipse* is an open source java software development kit, conceived in 2006, which has many useful features and plug-ins including the free Android Development Tools plug-in developed by Google to allow greater ease of use to Android developers using the *Eclipse* environment.

The Android Development Tools plug-in "extends the capabilities of *Eclipse* to let the developer quickly set up new Android projects, create an application UI, add components based on the Android Framework API, debug their applications using the Android SDK tools, and even export their signed (or unsigned) APKs in order to distribute their application"[1].

This plug-in as well as instructions on installing all Android applicable *Eclipse* elements (and even *Eclipse* itself) can be found on the Android Developers webpage: http://developer.android.com/index.html.

Work presented to the college March 23, 2010. This work was supported in part by the Ripon College Math and Computer Science Department.

Danial Craig Hanson hopes to find a job in software development/engineering, specifically in the realm of java and/or android application programming. (phone: 715-220-7593; e-mail: dch4nson@gmail.com).

Once installed, *Eclipse* and the free Android extensions provide the user with full access to the Android function calls, variable types, packages, and debugger; as well as a fully customizable virtual phone emulator, or the ability to use an Android compatible handset, physically connected to the development machine via USB, to test completed applications prior to publishing.

III. Developing An Android Application

Developing a successful Android application consists of 5 main steps: new project creation, interface development, logic and functionality development, debugging and testing, and publishing and installation. The following section will describe the intricacies of each of these steps in detail, and attempt to educate the reader on the finer points of constructing a simple android application. The material discussed within section III will assume the user has installed and is utilizing the *Eclipse* development kit discussed in the prior section.

A. *New Project Creation*

Creating a new project in *Eclipse* is as simple as selecting 'File' → 'New' → 'Project' and then selecting 'Android Project'. At this point *Eclipse* will ask for a small number of items: the titles and necessary implications of these items is as follows (as gleaned from the Android Developers' Hello, Android example page [2]):

--*Project Name*: This will be the project's name with *Eclipse* and the name of the directory on the development machine that will contain the files.

--*Application Name*: This will be the human readable title of the application, and the one that will accompany the finished application on the handset.

--*Package Namespace*: Following the same guidelines as packages in Java, the package name is the namespace under which all code will be maintained and the stub activity will be generated. It is imperative that this remain unique across any and all packages installed on the given Android system.

--*Create Activity*: Although optional, this is almost always necessary; this will be the name of the class stub that the plug-in will generate as a subclass of Anrdroid's activity class.

--*Min SDK Version*: This specifies the minimum API level required by the application. For instance, if it is set it to 1, it will work on all level 1 devices and higher. If it is set it to 2, it won't be installable on level a 1 device; but the additional functionality made available with level 2 may be utilized in return for the lack of backwards compatibility.

Upon completion of these fields, clicking 'finish' will complete the project construction, allowing the user to begin application development.

Eclipse will create the application's package and make its resources available to the developer in the navigation panel of the program with the following folders/files:

--*src*: This folder contains the java file in which the logic and functionality coding will reside (discussed in subsection C).

--*gen*: This folder contains the R.java file which is automatically generated by *Eclipse* to provide the java file in the 'src' folder with the proper memory address of any necessary variables declared in the layout file.

--*Android X.X*: This folder will contain the android.jar file corresponding to the specified minimum SDK version. This will include all the built in android functions and variables that may be referenced in the logic.

--*Assets*: This folder is seldom used, but essentially can hold an unstructured hierarchy of files to be referenced later as as raw byte streams[3].

--*Res*: This folder contains the resources, including images to be used in the application, and the *.png icon file. This folder also contains the *.xml layout files, where user interface development will take place.

--*Android Manifest.xml*: This is an XML file that is automatically generated, but can allow access to many of the items specified at the inception of the project including Project and Application Name, package namespace, etc…

--*default.properties*: This is another auto-generated file that will not require any form of user interaction but must be constructed for the sake of the application.

B. *Interface Development*

Once the program template has been established, the next logical step is to design the user interface. This is done primarily in the layout.xml file found in the 'Res' folder. The coding will be done in XML. Once opened the layout.xml file will allow the user to 'preview' the layout via a tab at the bottom of the opened file rather than emulating the program every time they want to see how a change in code impacts the visual layout. This view is entirely customizable based on the developer's specifications and requirements.

There are four layout options to be utilized by the developer, each offering a different approach to layout initialization:

--*Frame Layout*: This layout is simple but virtually useless for any practical application as it stacks all icons in the upper left corner of the window, not allowing any further input on the developer's part.

--*Linear Layout*: This layout will allow the developer to align widgets either vertically or horizontally and assign 'weights' to them to adjust their size relative to other widgets within their row or column.

--*Table Layout*: This layout is row and column based and allows the developer to grow or shrink those rows and columns based on their own personal needs.

--*Relative Layout*: Allows the developer to assign absolute sizes to each widget and position them relative to the other widgets and the edges of the screen.

Once the developer has selected a layout, the next step is simply to design the user interface using the XML language until they've constructed what they feel to be an appropriate and easy to use UI.

C. Logic and Functionality Development

When coupled with the User Interface Development, the Logic and Functionality Development portion of the application development completes the 'main phase' of the project. Whereas with the user interface the developer designs the visual elements of the application, in the logic and functionality portion, they assign functionality *to* those visual elements. For instance, if a button is created in the layout file, what the application is supposed to *do* when the button is clicked will be defined in during the logic and functionality development phase.

The logic is written in the very user friendly java language, and the corresponding file can be found in the 'src' file. All common java functions can be referenced and utilized, as well as all Android functions contained within the API specific *.jar file in the 'android x.x' folder.

Any widgets defined in the layout.xml file may be referenced from within this code, and any necessary local variables for logic and calculation may also be freely initialized and utilized.

D. Debugging and Testing

As any developer knows, once the project is near completion, it must be tested to ensure that everything operates as expected. *Eclipse* makes this process very painless by incorporating a fully customizable virtual emulator. This can be setup exactly to the developer's specifications, or based on one of many included templates setup to emulate any of many existing Android compatible phones.

Once an emulator has been initialized, the developer must simply compile and run the program and the emulator will launch, allowing the developer to see how the application operates on a 'faux-phone'. Another option available to the developer is to simply connect an Android compatible phone to the development machine via a USB cable and run the program directly on that handset. This allows the developer to actually use the program on a physical handset and provides very useful feedback.

E. Publishing and Installation

The final step in successful android application is installing and publishing the application. Installation can be accomplished in one of three ways: emulation, direct file installation or market installation.

The first and easiest (although not practical for multiple devices) is to use the emulation method described above with the target handset. By connecting a handset to the development machine and running a completed application on said handset, the application will automatically be installed.

The second option is to locate the *.apk file in the project directory on the development machine's hard disk and transfer it to the target handset's memory. Once this is accomplished, the application can be installed by utilizing the free market available application *Apps Installer*.

The final method is to publish the application to the android market and install it from there. This is the best option for mass publication and profit gaining, although the initial twenty-five dollar investment is to be considered if this is the avenue the developer chooses to take.

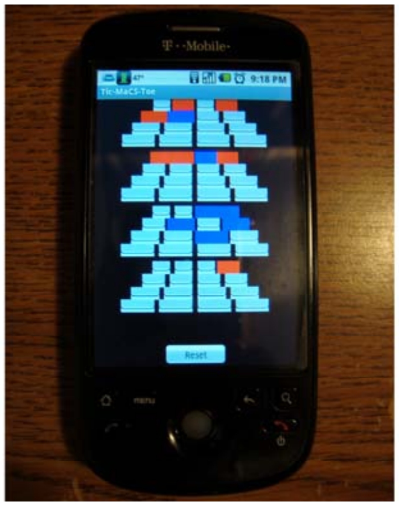

Figure 1. Tic-Macs-Toe live in action on the developers' test phone, a T-Mobile MyTouch 3G (a stripped down version of the HTC Magic).

These five steps encompass the basic procedures involved with application development and implementation for any basic android application. In the following section we will discuss the intricacies involved with the development of the three dimensional tic-tac-toe application.

IV. 3-Dimensional Tic-Tac-Toe

Remember from section I that the goal of our project was to create a three-dimensional four-by-four-by-four tic-tac-toe application with an attractive and accessible user interface and an intelligent opponent. We began by completing a program that we had already begun in C++ with the hopes of porting it into Android.

The first step to creating the android application was to develop the user interface. To accomplish this, we utilized the 'relative' layout format. We created un-clickable and invisible buttons with a width of 1px to serve as 'center posts' for the 64 clickable spots. The first of these posts was centered horizontally and aligned to the very top of the window, with two clickable buttons to the left, and two to the right. We then created another post located directly below the first, again with two clickable buttons to the left, and two to the right. This process was repeated until all 64 buttons were in place. The top row of each of the four 'sections' of buttons was thinnest, with each of the three successive rows growing slightly in width to give the impression of a three dimensional board.

The clickable spots consisted of a 3-D array of button widgets in the [x][y][z] format where [0][0][0] would be the top left corner and [3][3][3] would be the bottom right corner. Finally, a 'reset' button was placed at the bottom of the screen that would serve as just that and allow the user to begin a new game upon completion of one.

Next, another layout had to be created to serve as a text pop-up box. This layout consisted of a simple TextView widget, the text of which could be edited from within the java logic file, allowing it to be referenced anytime a popup textbox was necessary.

The logic coding was fairly simple when the time came as it was simply a matter of copying the completed and fully functional C++ code into the java file and changing the syntax as necessary to accommodate the Android language and the new button array.

In the finished product, the user would play blue versus the computer who would play red. Once a square was selected, it's on click listener would be turned off, making it unselectable, and the computer would use alpha-beta pruning with the mini-max algorithm to determine its next move. Once this had been done, the computer would check to see if a win (four in a row in any direction) or cat's game had occurred, and if so would display a message stating such and turn off *all* button listeners except the reset button which when clicked, would clear all buttons, reactivate all button listeners and begin a new game.

The majority of development went as smoothly as possible. The one major incident we encountered involved slow runtime due to memory restrictions. While the computer used to develop the original C++ program had 2GB of RAM and a 2.8Ghz processor, the phone only had 188MB of RAM and a 528Mhz processor, turning an instant computer response into an approximately two-minute wait when implemented on the target handset.

Memory restriction is the main bottleneck of mobile phone operating systems at this time and must be carefully considered when developing programs. In our case, we were lucky enough to be able to speed the program up without having to entirely rework the algorithm. In the C++ version, the algorithm was set to check '4-ply' meaning it would look at every possible combination of the next 4 moves, or, on an empty board 16,777,216 possible move combinations. We dialed this back down to 2, which meant a maximum of just 4,096 checks and the computer's response became instantaneous, yet it was still intelligent enough to be competitive with the user and in most cases still win the match.

V. Conclusion

Google's Android Operating System and its corresponding development environment have proven to be as user friendly and powerful as the search engine that popularized Google to begin with. Application development is a breeze, even for those new to the language, and the opportunities are nearly endless for freelance developers and entrepreneurs. I would like to personally encourage anyone with an interest in software and/or application development to take a look at Android as I truly believe the time will be well spent. The full code for the completed 4x4x4 Tic-Tac-Toe program can be found at the URLs listed below.

4x4x4 Tic-Tac-Toe Code Resources

[1] http://ripon.edu/academics/macs/summation/2010/supp/tix-java.pdf
[2] http://ripon.edu/academics/macs/summation/2010/supp/main-xml.pdf
[3] http://ripon.edu/academics/macs/summation/2010/supp/dialog-xml.pdf

Recommended Further Reading

[1] Reto Meier, Professional Android Application Development Wrox, 2008.

[2] Frank Ableson et al, *Unlocking Android: A Developer's Guide* Manning Publications, 2009.

Acknowledgment

I would like to personally acknowledge my partners in this endeavor Brad Bogenschutz, and Jerry Hardacre, as well as our faculty mentor McKenzie Lamb and the entire Ripon College 2010 Math and Computer Science Department, without whom this project never would have taken place. Special thanks is also due to the fine people at Google for developing such a user friendly software environment *and* great learning tutorials and reference material for the beginning Android developer.

References

[1] http://developer.android.com/sdk/*Eclipse*-adt.html
[2] http://developer.android.com/guide/tutorials/hello-world.html
[3] http://groups.google.com/group/android-developers/browse_thread/thread/879d4d9545f27b26/6032e4fe6941140c?lnk=raot&pli=1

Danial C. Hanson was born in Shakopee, MN in October of 1988. He graduated from Clear Lake Jr/Sr High School in Clear Lake, WI in 2006 and began attending Ripon College in Ripon, WI in the fall of 2006 where he is currently pursuing a Bachelors degree in computer science (tentative graduation May of 2010).

He has been employed by Ripon Medical Center in Ripon, WI as a Helpdesk Analyst since March of 2009 where he plans to continue employment until the fall of 2011, at which point he hopes to begin post-graduate education to obtain a Masters of Business Administration at a yet undecided institution.

Mr. Hanson is a member of the Sigma Chi fraternity and the BSA where he attained the rank of Eagle Scout. He enjoys spending time with his family, playing guitar, and playing with his Black Labrador 'Monty'.