Intermediate Report: Android Development Team

Jillian Andersen, Jordan Apele, David Ruhle, Kyle Wenholz

October 18, 2011

Contents

1 The Product **2**

2 Requirements **2**

2.1 Functional Requirements 2

2.2 Nonfunctional Requirements 2

3 Domain Analysis **2**

4 Coding Style Guide **2**

4.1 Working with Java 3

4.1.1 Style Rules 3

4.1.2 Code Rules 3

4.2 Working with JavaScript 4

4.2.1 Style Rules 4

4.2.2 Coding Rules 4

4.3 Working with HTML5 5

4.4 Working with CSS 6

4.5 Setting up the Development Environment 6

4.5.1 Android SDK 6

4.5.2 PhoneGap 6

4.5.3 Editing Environment 6

4.6 Retrieving the Source 7

5 Implementation Details **7**

5.1 PhoneGap 7

5.2 Interactions with Other Development Teams 7

5.3 Testing 7

6 Planning and Reflection **7**

6.1 Current State of the Project 7

6.2 Proposed Schedule 7

1 The Product

The final product of this development team will be a downloadable Android application for playing Pong via human motion. Receiving position data from a Wii Remote the application allows a user to move a paddle on screen with motion in physical space. The current concept is to host games over the internet and allow play between two players to proceed as a normal game of Pong. This may change in the future to include *enhanced modes* where players may retrieve power-ups, attack or complete any other number of non-standard actions. Aside from the gameplay, however, the Android application will host a suite of other features. Accessing user statistics, global statistics, help and support, changing aesthetic settings, adjusting the volume, and even navigating to the Vir-Pong website will all be possible from within the application. While the application developed by our team is targeted at the Android platform, our team is working closely with the iOS development group to support a cohesive and quality application across multiple platforms.

Installation and usage instructions as well as help and support will be found on the Android market or on the Vir-Pong site. These instructions will be targeted towards novice technology users so that our product may be enjoyed by all groups. Developer documentation generated during the development cycle will be available to all Vir-Pong employees and the general public as part of our open-source commitment. Where this documentation will be hosted is currently under consideration.

The final product of this team will integrate with the greater Vir-Pong ecosystem. Servers, devices, the website, and users will bring together a community of human Pong players, all enjoying our product. Our piece in this greater puzzle is to put that experience in the pockets of consumers and allow Pong to be played in the physical world.

2 Requirements

2.1 Functional Requirements

2.2 Nonfunctional Requirements

3 Domain Analysis

4 Coding Style Guide

The majority of the coding takes place within the *assets/www* folder of our application. It is, therefore, important that our team maintain a clear and concise style within this limited space. In general, folders should be titled in the CamelCase style (first letter a capital) and individual files should be likewise with the exception that the first letter is a lower case. Dashes may be permitted so long as it is used when there may be multiple editions of something (e.g. a "logo-blue.jpg" and "logo-red.jpg"). Further exceptions include any README files (used for build instructions) or versioned

files.

As to how many folders to have, if there exists a logical grouping between one file and several others (i.e. more than 2 files are related to one another) then these should be placed in a separate folder within the *www* directory. Files themselves and the code contained should follow the guidelines given below. While these guidelines are not quite as extensive as some resources (such as Sun's own Java style guide[1]) the brevity serves our team well.

4.1 Working with Java

While Java is not a primary language for our team, we will be strictly following conventions laid down by Sun and other programmers[1][12].

4.1.1 Style Rules

- All identifiers use letters ('A' through 'Z' and 'a' through 'z') and numbers ('0' through '9') only. No underscores, dollar signs or non-ascii characters (with one exception mentioned later).
- Class and interface names will use CamelCase beginning with a capital letter.
- Fields, local variables, methods and parameters will be named using CamelCase beginning with a lower case letter. The exception being that constants are in all capital letters spacing words with underscores.
- There will be no use of *break*.
- Use a separate line for an increment or decrement.
- All fields must be private, except for some constants.

A note on comments: all methods and classes should contain a standard Javadoc comment (text description and appropriate author, version, parameter and return tags). In-line comments are strongly encouraged to assist in readability of the code.

4.1.2 Code Rules

- Class elements will follow this order: fields, constructors, methods.
- Opening curly braces should be on the same line as what they are opening.
- Closing curly braces will be horizontally aligned with the line where the statement began.
- Indent each time a new bracket set is created. Indents should be four spaces.
- All control-flow statements must use brackets.

- Commas and semicolons are always followed by whitespace.
- Binary operators should have a space on either side.
- Parentheses should be used in expressions not only to specify order of precedence, but also to help simplify the expression. When in doubt, parenthesize.

4.2 Working with JavaScript

The majority of our JavaScript style guidelines are mimicked from Google[13]. For further and more detailed information, see their guide. To many new programmers, JavaScript seems very much like Java (even the name!), but it is important to note that these are different languages and we have several very different rules.

4.2.1 Style Rules

- In general, use $functionNamesLikeThis$, $variableNamesLikeThis$, $ClassNamesLikeThis$, $EnumNamesLikeThis$, $methodNamesLikeThis$, and $SYMBOLIC\_CONSTANTS\_LIKE\_THIS$.
- Start curly braces on the same line as what they are opening.
- Be sure to indent blocks by four spaces.
- Use blank lines to group logically related pieces of code.
- Use parentheses only when required.
- Prefer ′ over ” for strings.
- Use JSDoc annotations (@*priave* and @*protected*) where appropriate. Marking visibility is encouraged.
- Be sure to use JSDoc comments. A comment at the top of the file for authorship and general overview, comments for methods, and inline comments are encouraged. The first two are done using $/*...*/$ and the latter is //.
- Be sure to use @param and @return tags for methods and functions.
- Simple getters may have no description but should specify the returned values.

4.2.2 Coding Rules

- Always declare variables with *var*.
- Use $NAMES\_LIKE\_THIS$ for constants. Use @*const* where appropriate. Never use the *const* keyword.

- Always end lines with semicolons.
- Feel free to use nested functions but try to comment these to make them clear.
- Do not declare functions within blocks. Instead you may do the following:

```
if (x) {
  var foo = function() {}
}
```

- Avoid wrapper objects for primitive types.
- The keyword *this* is for object constructors and methods only.
- For-in loops are only for iterating over keys in an object/map/hash.
- Do not use multiline string literals. Instead, use concatenation when initializing such long strings.

4.3 Working with HTML5

HTML5 and CSS are so quickly evolving of late that style guides are not readily available. We have, however, compiled our own unique guide from some suggestions found on the Web Developer's Virtual Library[10]. We recommend keeping JavaScript and CSS code in files separate from the HTML. This is primarily to keep the code modular and sensible. Reading HTML and JavaScript in the same file can be confusing.

- Block-level tags are to the far left with content indented four spaces.
- Don't indent tags relative to their container.
- Line up multiple attributes with the "=" signs all in the same column.
- The home page is an index to other pages.
- Page designs should be consistent in appearance and structure.
- Choose a meaningful title for pages.
- ∗Unless it is an incredibly brief code-snippet, do not include JavaScript or CSS in the *html* file. Place this code in a separate file.
- Provide a *Home* link.

4.4 Working with CSS

CSS doesn't normally follow any particular style guide, but we have imposed a few general rules to keep things neat[11].

- Consider a table of contents at the top of the CSS file. This would reference labels or comments used as tags in the file. Use a tree structure.
- Because constants are not possible, define colors and typography used in comments at the top of the file. This allows you to reference these colors font styles later to be consistent.
- Organize properties alphabetically.

4.5 Setting up the Development Environment

In order to develop for Android, you will need the Android SDK, the PhoneGap software, and some editor. These pieces are relatively easy to set up, but because all systems are different, some personal configuration may be required. All of the software mentioned below can be found, with current links, at the PhoneGap Android page[9].

4.5.1 Android SDK

The Android SDK comes with an Android Emulator as well as Android libraries. To download the SDK, first check to make sure your operating system fulfills all of the system requirements. A list of of system requirements is available on the Android Developers website[6].

Next, download the Android SDK from the Android Developer website[4]. The instructions for installation are found on the same site but on the installation page[5]. **Note:** do not put a space in the folder name.

After that, you must add necessary components to the SDK. The instructions for that portion are found on the components page[3]. On your first run, you will be required to create an Android Virtual Device (AVD) that is simply a mock phone.

4.5.2 PhoneGap

The PhoneGap download is primarily a collection of tools that provide functionality for the HTML5 and JavaScript interface in a native application environment. That is, the PhoneGap jar, js, and xml files all serve to support the use of HTML5 and JavaScript coding as implementing the core functionality of the application.

4.5.3 Editing Environment

In theory, any development can be done from a text editor so long as you have access to the Android SDK and Java. It is highly recommended, however, that you use Eclipse[7]. You then may want to

install the ADT plugin for Android Development[2].

4.6 Retrieving the Source

In order to work with the source code of the project, you will likely want to use Git[8]. Using Git, you may clone the repository from `git@github.com:VirPong/human-pong`. You will then want to navigate into $Android/VirPong-Mobile/$ and create a folder called $assets$. Navigate into $assets$ and clone `git@github.com:VirPong/www`. Now you may open Eclipse and import the $VirPong-Mobile$ directory as an existing project. You may need to point the build path to your PhoneGap Jar file (located wherever you downloaded PhoneGap and then inside the Android folder). Once this is done, you may begin development! **Note:** an alternative to git is to download the repository from `https://github.com/VirPong/human-pong` and `https://github.com/VirPong/www`, placing the www repository in the same place mentioned above.

5 Implementation Details

5.1 PhoneGap

5.2 Interactions with Other Development Teams

5.3 Testing

6 Planning and Reflection

6.1 Current State of the Project

6.2 Proposed Schedule

References

[1] Code Conventions for the Java TM Programming Language. http://www.oracle.com/technetwork/java/codeconvtoc-136057.html, August 1999. [Online; accessed October-2011].

[2] ADT Plugin for Eclipse. http://developer.android.com/sdk/eclipse-adt.html#installing, 2011. [Online; accessed September/October-2011].

[3] Android SDK Components. http://developer.android.com/sdk/adding-components.html, 2011. [Online; accessed September/October-2011].

[4] Android SDK Download. http://developer.android.com/sdk/index.html, 2011. [Online; accessed September/October-2011].

[5] Android SDK Installation. http://developer.android.com/sdk/installing.html, 2011. [Online; accessed September/October-2011].

[6] Android SDK System Requirements, 2011. [Online; accessed September/October-2011].

[7] Eclipse Packages. http://www.eclipse.org/downloads/packages/release/helios/sr2, 2011. [Online; accessed September/October-2011].

[8] Github. https://github.com, 2011. [Online; accessed September/October-2011].

[9] PhoneGap for Android. http://www.phonegap.com/start#android, 2011. [Online; accessed September/October-2011].

[10] The WDVL Style Guide. http://wdvl.internet.com/Authoring/Style/Guides/WDVL.html, 2011. [Online; accessed October-2011].

[11] Vitaly Friedman. Improving Code Readability With CSS Styleguides. http://coding.smashingmagazine.com/2008/05/02/improving-code-readability-with-css-styleguides/, May 2008. [Online; accessed October-2011].

[12] Paul Wheaton. Java Programming Style Guide. http://www.javaranch.com/style.jsp, 2011. [Online; accessed October-2011].

[13] Aaron Whyte, Bob Jervis, Dan Pupius, Eric Arvidsson, Fritz Schneider, and Robby Walker. Google JavaScript Style Guide. http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml. [Online; accessed October-2011].