Intermediate Report: Android Development Team

Jillian Andersen, Jordan Apele, David Ruhle, Kyle Wenholz

October 27, 2011

Contents

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 player 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 users 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 player 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 Analysis

2.1 Functional Requirements

2.1.1 Playing a Game - v1.0

Actors:

- Player
- Hub
- Android device
- Input device

Preconditions:

- Player is authenticated into the system and has successfully requested a game from the hub.
- Input device is tethered to the Android device and is ready to submit motion data.

Postconditions:

- A winner has been determined.

- Player statistics are recorded by the hub system.

Scenario:

1. Hub signals a ready-start and the game begins.
2. Player observes ball and opponent's paddle motion on Android device.
3. Player responds with appropriate motion, detected by the input device and relayed to the Android device.
4. Android device relays motion data to the hub.
5. Hub recalculates ball and paddle position then sends updated information to Android device.
6. Player's Android device displays the current information.
   Repeat **??** through 6 until a point is scored.
7. Hub logs point scored then resets game state to fresh.
   Repeat **??** through 7 until score limit is reached.
8. Hub indicates winner to Android device.
9. Android device displays winner to player and announces game complete.
10. Hub logs game statistics for later access.
11. Player is prompted to play another game or exit back to the home screen.

Alternatives:
5a) Hub signals that connection to other player has been lost.

1. Android device signals a game pause to player.
2. Player is prompted to wait or leave the game.
3. The player elects to wait.
4. Hub signals that the other player is reconnected.
5. Game resumes.
6. Return to 6 in main scenario.

8a) The player's Android device loses connection with the hub after score limit has been reached.

1. The hub waits for a specified recovery time.
2. Upon connection lost, the player's Android device begins recovery mode, trying to reconnect with the hub and indicating this to the player.
3. After reconnecting with the hub, the game's winner is announced.
4. Player is prompted to verify this, due to the connection lost.
5. Both players agree, so hub logs the game statistics.
6. Player is prompted to play another game or exit back to the home screen.

8b) The game duration exceeds the maximum time.

1. Hub alerts players that game is reaching overtime.
2. Player is given option to continue playing, call a tie, or postpone the game.
3. Player selects to call a tie.
4. The game ends and hub logs results.
5. Player is given option to play another game or return to home screen.

Figure 1 depicts the sequence of actions during a game. The Hub signals to the Android Device that the game is ready to start, then the player is able to move the paddle using their chosen Input Device. That paddle movement is sent from the Input Device to the Android Device, which relays the motion data to the Hub. It is then the Hub's responsibility to relay the game status back to the

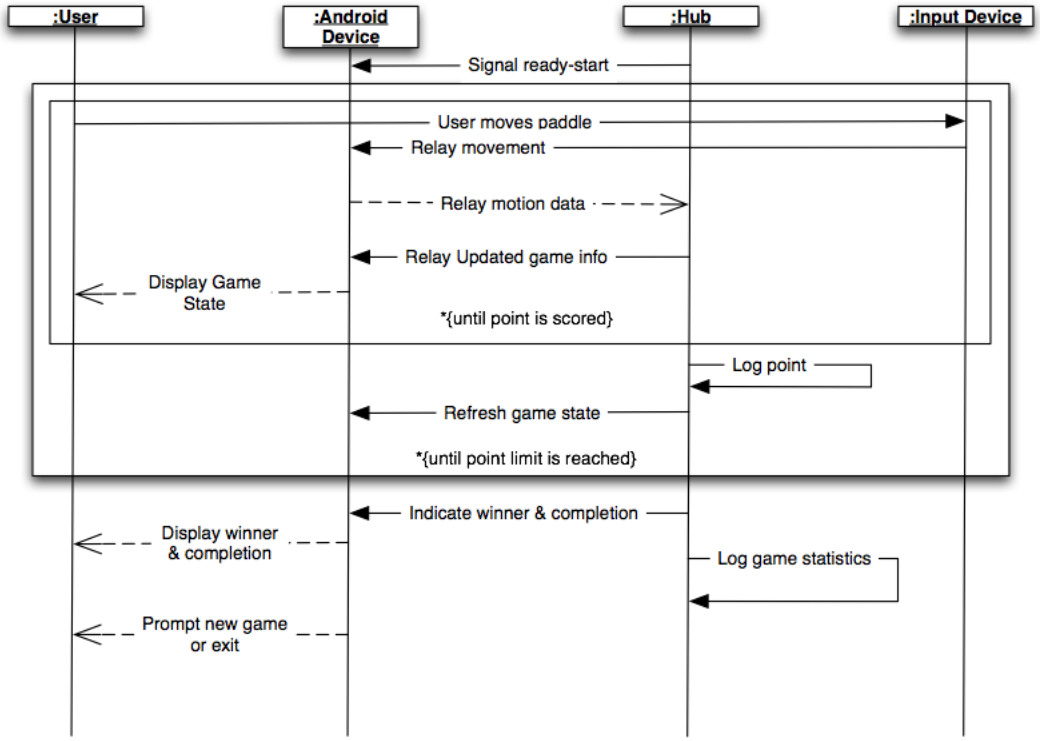

Figure 1: The role of our system in gameplay is described through the system sequence diagram above.

Android Device so that it may display the game state. This relay of information is repeated until a point is scored. The Hub will log the point, then refresh the game state and send this new state to the Android Device. After a refresh, the player is able to move the paddle again and continue play. Our system remains in this cycle until the maximum number of points is reached. The Hub indicates the winner and completion of the game to the Android Device, which then informs the player. Finally, the Hub logs the game statistics and the Android Device prompts the player for a new game or exit.

Initializing a Game - v2.0

Actors:

- Player
- Hub
- Device

Preconditions:

- The application is installed and open on the device.
- The player has already created an account with Vir-Pong.
- The input device is tethered to the Android device and is ready to submit motion data.

Postconditions:

- The hub is relaying game information.

- The system is using a display to relay the game state.

Scenario:

1. The player selects to begin a game.
2. The device prompts the player to authenticate.
3. The player provides credentials.
4. The device verifies credentials with the hub.
5. The device requests a game from the hub.
6. The hub selects an opponent and signals a ready-start to device.
7. The device loads an initial game state displayed to the player.
8. Game begins.

Alternatives:
1a) The player selects the wrong action.

1. The player may elect to go back the the previous screen with a *return* function.

3a) The player provides incorrect credentials.

1. The device cannot authenticate into the hub.
2. The device displays a message that tells the player that the entered credentials were incorrect.
3. The device prompts the player to authenticate.
   Repeat 1-3 until the Player selects the *return* function.

5a) The system can not connect to the hub.

1. The device sends the player an error message that prompts the player to retry or exit.
2. Player chooses to retry.
3. System connects to hub.
4. Return to 6 of main scenario.

Connect to Input Device - v1.0

Actors:

- Player
- Device
- Wii Remote

Preconditions:

- System is installed on the device and has launched.

Postconditions:

- Input device (Wii Remote) is selected and prepared for game use.

Scenario:

1. Player selects start game.
2. Device prompts player for input method: Wii Remote, device accelerometer, or touchscreen.
3. Player elects to use the Wii Remote.

4. System attempts to tether Wii Remote.
5. Player follows instructions from system to connect Wii Remote.
6. System accepts Wii Remote device and notifies user.
7. System proceeds to game launch.

Alternatives:
2a) Player selects phone accelerometer.

1. System asks player to make certain the device has an built in accelerometer.
2. Player indicates that device has accelerometer.
3. System connects to device accelerometer.
4. System proceeds to game launch.

2b) Player selects touch screen interface.

1. System asks player to touch a box displayed on-screen to ensure that a touch screen is available.
2. Player touches box.
3. System proceeds to game launch.

4System attempts Wii Remote connection, and fails.

1. System alerts user that connection failed.
2. Return to step 2 of main scenario.

Changing the Game Settings - v1.0

Actors:

- Player
- Device

Preconditions:

- The application is installed and open on the device.

Postconditions:

- The newly changed settings have been saved and will be applied to future game play.

Scenario:

1. Player selects a *change settings* function.
2. Player changes the setting(s).
3. Player selects a *save and apply* function.
4. Device saves changes and applies them to future game plays.
5. Device returns to the main menu.

Alternatives:
2a) The player selects a non-valid entry for a setting.

1. The device displays an error message that tells the player he entered a non-compatible value.
2. The device returns the setting to a default state.

3a) The player decides not to change any settings.

1. The player selects the *return* function.

Viewing Stats - v1.0

Actors:

- Player
- Website
- Device

Preconditions:

- Our application is installed on the device and has launched.
- The player has already created an account with Vir-Pong and is logged in.

Postconditions:

- The player has looked at their statistics.
- The player can compare his to everyones.

Scenario:

1. Player chooses view stats option
2. Device requests personal score from website.
3. Website sends personal stats.
4. Device displays personal stats and CompareStats option.
5. Player selects CompareStats option.
6. Device navigates to website with all player stats listed.
7. Player finishes viewing stats and hits back button.
8. Device returns to system state SelectStats.

Alternatives:
2a) Unable to contact website.

1. Display connection error message.
2. Player may utilize a *retry* or *continue* button.
3. Player selects *retry*.
4. Return to 2.

Edit Account Information - v1.0

Actors:

- Player
- System
- Vir-Pong Website

Preconditions:

- The application is installed on the device and has launched.

Postconditions:

- Player has edited account information.

Scenario:

1. Player selects Edit Account information.
2. System opens phone browser to "Vir-pong Edit Account" information page.
3. Player begins editing account information.
4. Player saves and exits editing account information.
5. System returns user to previous page.

2.2 Usability Requirements

In order for the application to provide an experience enjoyable to a wide user base, the following usability requirements should be met:

- Players should have their statistics saved and then made available for viewing.
- Players should be able to change game settings(difficulty, ball shape, etc.).
- Users should be able to register an account through the application.
- The application needs to be available for download via the website or on the Android Market.
- The application should have the option to view a game in progress.
- It would be nice to make the game environment modifiable (i.e. theming).
- When there is not a second available human player (or connection to the hub is impossible) there should be an option for playing versus the computer in training mode.

2.3 Reliability Requirements

A working product is important, but reliability ensures a consistent experience. With this in mind, our design is striving for security of private user information and the ability to pause the game. The latter is focused on creating a fault tolerant solution for network connectivity. A strict testing policy will help to alleviate many other possible issues.

2.4 Performance Requirements

The ability to adjust the display for different screens will allow for better performance of the software on different Android devices. The system must also minimize latency when contacting the server for game information or receiving information from the Wii Remote. There are no other connections that are quite as important.

2.5 Supportability Requirements

Installation, game play, and resource usage should all be properly documented in a conveniently located and navigated user-manual. A list of known compatible devices included in the manual will assist in keeping potential customers informed about our product. There should also be some contact information for support and help when the available documentation is not enough.

In keeping with the project's open source roots, it will be important to make all source code well documented for the general public; make the source code readily available; give credit to non-employees that have contributed to our application; and provide substantial developer documentation.

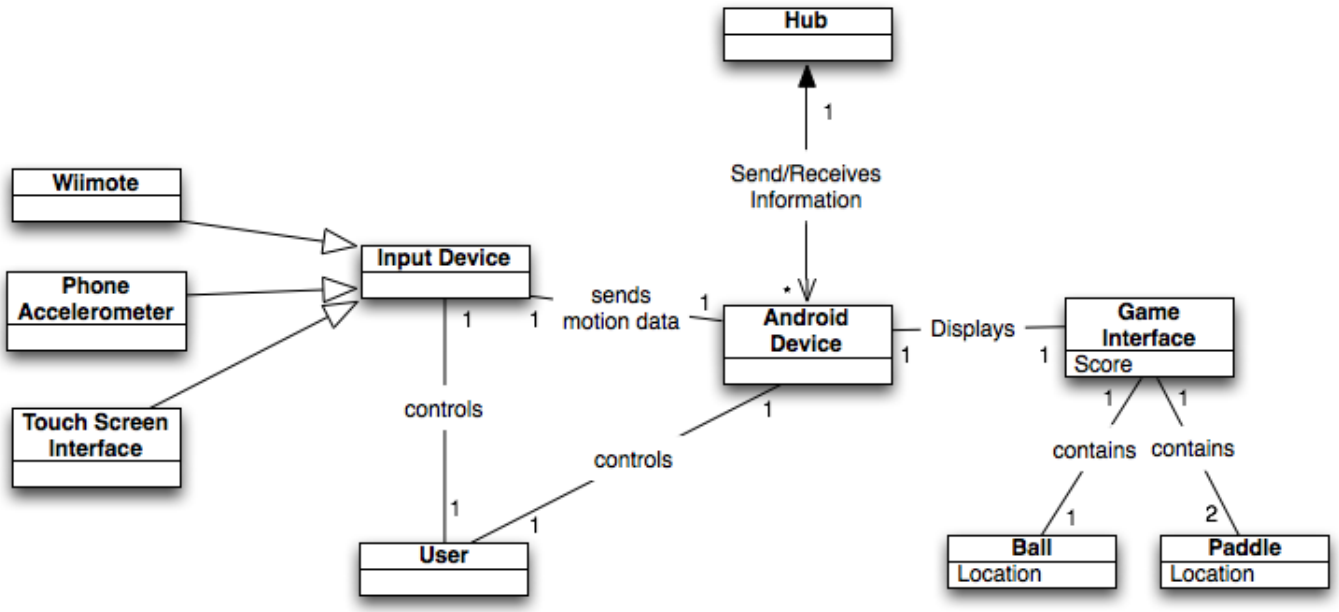

Figure 2: Class interactions and attributes outline the major system components.

3 Domain Analysis

Figure 2 shows the interactions that take place and the associations involved in playing a game of Pong on an android device. Throughout the game, the player controls an input device that will pass motion data to the android device. There are three different types of input that can be used by the user: the Wii Remote, the phone accelerometer, or the touch screen interface. The motion data from the input device is sent to the android device and from there is immediately sent to the hub. The hub sends back information on both the player's and an opponent's positions as well as the position of the ball. Other relevant data is sent to and from the android device and hub, including score data and a signal that a point has been scored. To allow the player to view their Pong game, the android device relies on a game interface to graphically display the constantly updating state of the game. This game interface contains a graphical representation of two paddles and one ball that respond to information sent from the hub and also keeps track of the game's current score.

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][13].

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).
- In general, use $methodNamesLikeThis$, $variableNamesLikeThis$, $ClassNamesLikeThis$, and $SYMBOLIC\_CONSTANTS\_LIKE\_THIS$.
- Use a separate line for an increment or decrement.
- All fields must be private, except for some constants.
- Class elements will follow this order: fields, constructors, methods.
- Limit the scope of local variables.
- Initialize objects as late as possible.
- Use Strings with care.
- Avoid wrapper classes.

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 Coding Rules

- 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.
- There will be no use of *break*.

4.2 Working with JavaScript

The majority of our JavaScript style guidelines are mimicked from Google[14]. 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 $SYMBOLIC\_CONSTANTS\_LIKE\_THIS$.
- Avoid using many global variables.
- 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.
- 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 //.
- Use JSDoc annotations (@*priave* and @*protected*) where appropriate. Marking visibility is encouraged.
- 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.
- Use *onclick* instead of *javascript* : for anchors.

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[11]. 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[12].

- 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[10].

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[7].

Next, download the Android SDK from the Android Developer website[5]. The instructions for installation are found on the same site but on the installation page[6]. **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[4]. 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[8]. You then may want to install the ADT plugin for Android Development[3].

4.6 Retrieving the Source

In order to work with the source code of the project, you will likely want to use Git[9]. Using Git, you may clone the repository from `git@github.com:Vir-Pong/human-pong`. You will then want to navigate into $Android/Vir-Pong-Mobile/$ and create a folder called *assets*. Navigate into *assets* and clone `git@github.com:Vir-Pong/www`. Now you may open Eclipse and import the $Vir-Pong-Mobile$ 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/Vir-Pong/human-pong` and `https://github.com/Vir-Pong/www`, placing the *www* repository in the same place mentioned above.

5 Implementation Details

5.1 Basic Design

The core technology of the Android application is PhoneGap[2]. PhoneGap basically creates a shell on top of native devices, allowing developers to design the application using HTML5 and JavaScript. This avoids the need to use native languages, such as Java or Objective C, and gives developers the same power used to generate content on the web. Using PhoneGap allows us to combine with the iOS team and design a single application.

Our application is structured such that individual "web pages" will represent different features of the application: separate pages for the home screen, game playing, settings, etc. JavaScript files provide the functionality behind each of these features and CSS creates the styling. The separation of the system into layout, style and functionality with specific languages allows us to use the power of languages built specifically for each piece of our whole. While this requires a wider knowledge

base, the benefits are far more important to our dedicated team. Working with other teams' systems is made slightly easier through this choice of technology.

The WebUI and Server teams are so closely connected to internet technologies that our use of similar technologies creates a stronger basis on which to interface. The Android application, as it stands, will not be storing any user information. All player account and statistical information will be handled by the databases constructed by other teams. This keeps information secure and centralized. In the future, it may be possible that the application will hold local login information for several users, as a convenience.

As the design of our application progresses, there may be changes to the overall implementation structure. The current implementation, however, is designed to be rapidly prototypable and easy to work with other teams. Going forward, we hope to keep improving on our use and understanding of the technologies at hand.

5.2 Testing

At this stage of development, many of our features are split across different pages of the application. This has allowed us to test each feature in isolation. As development progresses, we are constantly bringing new features into sections of the application we consider as *main*. This process keeps us from breaking the functioning pieces of the system, while progressing quickly on new features.

Our general testing strategy is to use each test page as a suite of unit tests. Buttons allow us to interact with functions and test each feature individually. This may sound tedious, but it is effective and simulates unit testing in the best way we know. Going forward, we will be keeping certain test pages to allow us to retest functionality. Final builds will, of course, lack these pages.

Members of the team are encouraged to consistently bring their code to others for verification. This keeps members informed about each other's progress, and it forces members to write cleaner, more comprehensible code. Our ability to snapshot code into the repository has also helped in this effort. Each snapshot is verified to be tested and functioning, with documentation marking any rough edges. These code checks represent each end of a testing cycle.

6 Planning and Reflection

The biggest challenge involved with a project of this nature is the variety of interactions between all of the groups. The Android device in particular relies on communication with the Wii Remote, server, and website, so consistent communication is vital. Therefore, if one team manager does not make an effort to communicate, the entire project suffers. We finally realized the importance of human communication and have planned more manager meetings for the rest of the semester. In addition, managers have become more prompt with emailing and this has made it much easier to stay in touch and keep everyone updated on the status of the project. Another challenge has been a general lack of motivation due to a failure to create a sense of individual accountability. It was nearly

impossible to measure how much work each individual in every group had done; consequently, team members were not highly motivated to work at a fast pace. As a result, some teams would move ahead of the others and then feel a great deal of frustration when they had to wait for other teams to catch up. This is an extremely difficult challenge to overcome, but the situation has improved because we are at the point where it is apparent if a team member is not contributing a satisfactory amount. The increased manager communication also causes the group leaders to hold each other more accountable for their groups progress. A more specific challenge for the Android team was getting the iOS development team to agree to implement PhoneGap; this was difficult because their team members were divided on the issue. This was especially frustrating because our team could not move forward until the iOS team had made their decision. However, we finally convinced them that the use of PhoneGap has a number of important benefits for both teams.

If we were to begin this project again, there are a number of things we would have done differently. First of all, we would have written the APIs for Android to server and Android to Wii Remote communication at the very start of the project. This would have allowed our team to more forward, even when the other teams were stuck. This change would also have helped us be clearer in our expectations of what we needed from the Wii Remote team. Another beneficial change would been restructuring the Android and iOS development teams. Since we are using PhoneGap, the user interface for the game will be the same for both systems, so it would have been advantageous to instead divide the teams between local programming (the interface, pong game, features) and communications (namely between the server and the device, but also the Wii Remote to the device).

6.1 Past Goals

Thursday the 29th of September

- Complete a preliminary class diagram Status: Completed
- Create a prototype of the Android game interface Status: Completed
- Create an interface that would be compatible with both Android and iOS devices Status: Completed 10/11 because the iOS team had not yet committed to using PhoneGap.
- Finish Bluetooth Plugin Status: Delayed because the Wii Remote team wished to put all of its effort into setting up the BluezIME application instead of integrating their code into the PhoneGap plugin.

Thursday the 6th of October

- Server Communication to and from Android Status: Delayed but completed 10/10 due to unforeseen difficulties that were encountered during final testing.
- Ability to capture wii data on interface Status: Completed

Monday the 10th of October

- Commit all code and documentation to GitHub Status: Completed

Thursday the 13th of October

- Create a working demo of the communication from Wii Remote to Android to server and then back to Android Status: Completed
- Be able to draw on Canvas Status: Completed

Monday the 24th of October

- Be able to query the database from the client side to verify usernames and passwords Status: Completed

Friday the 28th of October

- Finish an intermediate report for the Android team Status: Will be completed the 28th

6.2 Future Goals and Deadlines

- **Monday the 31st of October**: Present a working demo of the core functionality of our product. This includes some Wii Remote to Android connectivity as well as Android to server connectivity, a user interface for the phones, and a working website with a login page as well as other key features.

- **Friday the 4th of November**: Have a more robust server connection so that the desired data is being passed from the devices to the server and back. Also we should have a robust API for the game logic. Both of these tasks will be handled by Jillian and Kyle. We also will fully integrate the desired functions of BluezIME with our PhoneGap application. Jordan and David will complete this task.

- **Friday the 11th of November:** Start enhancing the user interface of the game. This includes the ability to maintain and display score and sizing the paddles' movement in relation to the devices screen size. Those tasked to handle this will be Kyle and David. Jillian and Jordan will refine the user login process.

- **Friday the 18th of November:** Allow the player to select settings for their game play. Kyle and Jordan will work on this task. We will also work on other secondary features such as viewing the players game stats and changing account information. This will be handled by David and Jillian

- **Tuesday the 25th of November:** Clean up and embellish the applications user interface and test the game a great deal. We will all help in both of these activities.

- **Friday the 2nd of December:** Have all core functionality completed and satisfactorily tested. Have as many secondary features completed and tested as possible. Make sure all documentation is complete and the code follows our style guides.

- **Friday the 9th of December:** Code freeze. Begin pulling together the components for our final product presentation.
- **Wednesday the 14th of December:** Present our product to management.

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

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

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

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

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

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

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

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

[10] PhoneGap for Android. http://www.phonegap.com/about, 2011. [Online; accessed September-2011].

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

[12] 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].

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

[14] 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].