Final Report: Android Development Team

Jillian Andersen, Jordan Apele, David Ruhle, Kyle Wenholz

December 13, 2011

Contents

1 The Final 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 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.

- Replay is saved to the system database.

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 2 through 6 until a point is scored.
7. Hub logs point scored then resets game state to fresh.
   Repeat 2 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 replay 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 pauses the game and signals a disconnection from the game.
2. Player is prompted to leave the game.
3. The player goes back to main menu.

Initializing a Game - v3.0

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 software displays various options.
2. The player selects to begin a game.
3. The player is prompted with a list of various game rooms.
4. Player selects the desired game.

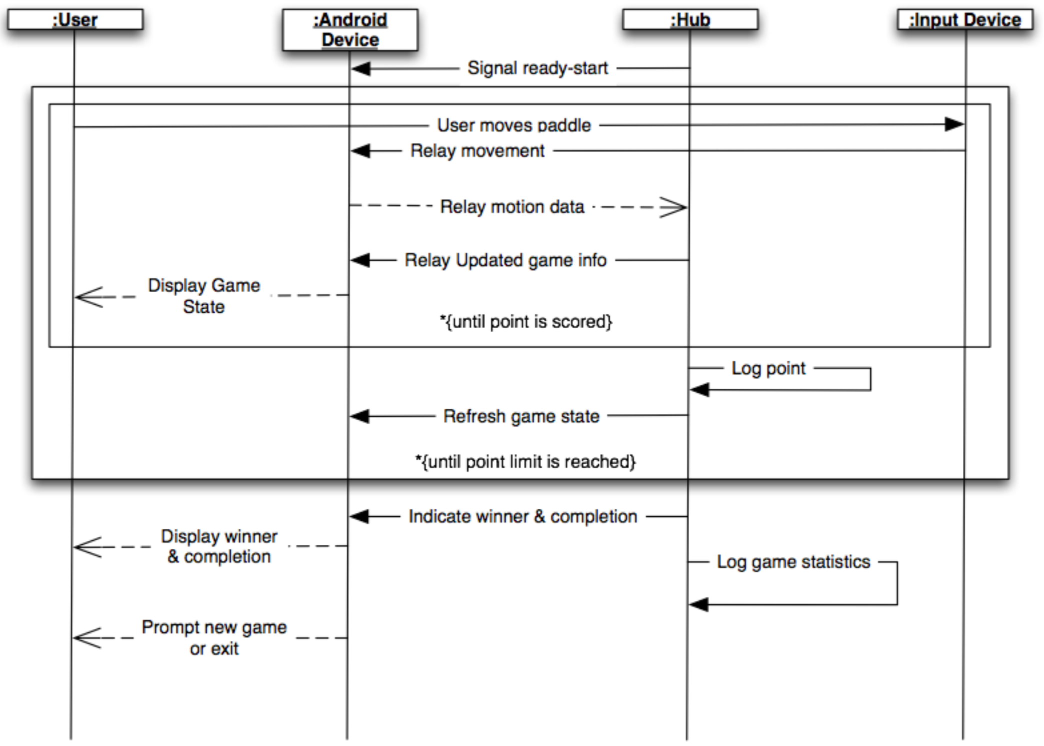

Figure 1: The role of our system in gameplay is described through the system sequence diagram above.

5. The device requests the selected game from the hub.
6. The player is notified of which paddle she is.
7. The hub waits for an opponent then signals a ready-start to device.
8. The device loads an initial game state displayed to the player.
9. Game begins.

Alternatives:
2a) The player selects the wrong action.

1. The player may elect to go back the the previous screen with a *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 7 of main scenario.

Connect to Input Device - v2.0

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 is prompted with options for input devices.
2. Player selects to use a Wii Remote, but Wii Remote is not connected yet.
3. Device directs player to instructions for pairing with a Wii Remote.
4. Following directions, the player opens the correct menu for pairing with a Wii Remote.
5. Player selects to pair with the Wii Remote.
6. System attempts to tether Wii Remote.
7. Player follows instructions from system to connect Wii Remote.
8. System accepts Wii Remote device and notifies user.

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.

6System attempts Wii Remote connection, and fails.

1. System alerts user that connection failed.
2. User is prompted to try again or back out.

Changing the Game Settings - v2.0

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

Alternatives:
3a) 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.

4a) The player decides not to change any settings.

1. The player selects the *return* function.

Viewing Replays - v2.0

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 watched their replay.

Scenario:

1. Main menu options are displayed to the player.
2. Player chooses to view replay.
3. Device requests list of available replays from server.
4. Database sends list of replays.
5. Device displays list of replays.
6. Player selects a replay to watch.
7. Device displays replay.
8. Player finishes the replay and hits the back button.
9. Device returns to main menu.

Alternatives:
??a) Unable to obtain available replays.

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

Editing 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 on main menu.
2. Device displays current login and pin number.
3. Player may change these current settings.
4. Player saves and exits editing account information.
5. System returns user to previous page.

2.2 Nonfunctional Requirements

3 Diagrammatic Depictions of the Product

Figure ?? shows the interactions that take place and the associations involved in playing a game of virtual pong on an android device. Throughout the game, the user 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 immediately sent to the hub. The hub sends back information on both the user'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 user to view their virtual 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.

3.1 Interaction Diagram

Though our code utilizes a number of design patterns, the code for our first use case, playing a pong game, uses three highly advantageous design patterns that help structure our code. One of the design patterns used was the observer pattern. In order to update paddle positions for each player efficiently, we chose to listen for events from the input device rather than constantly updating

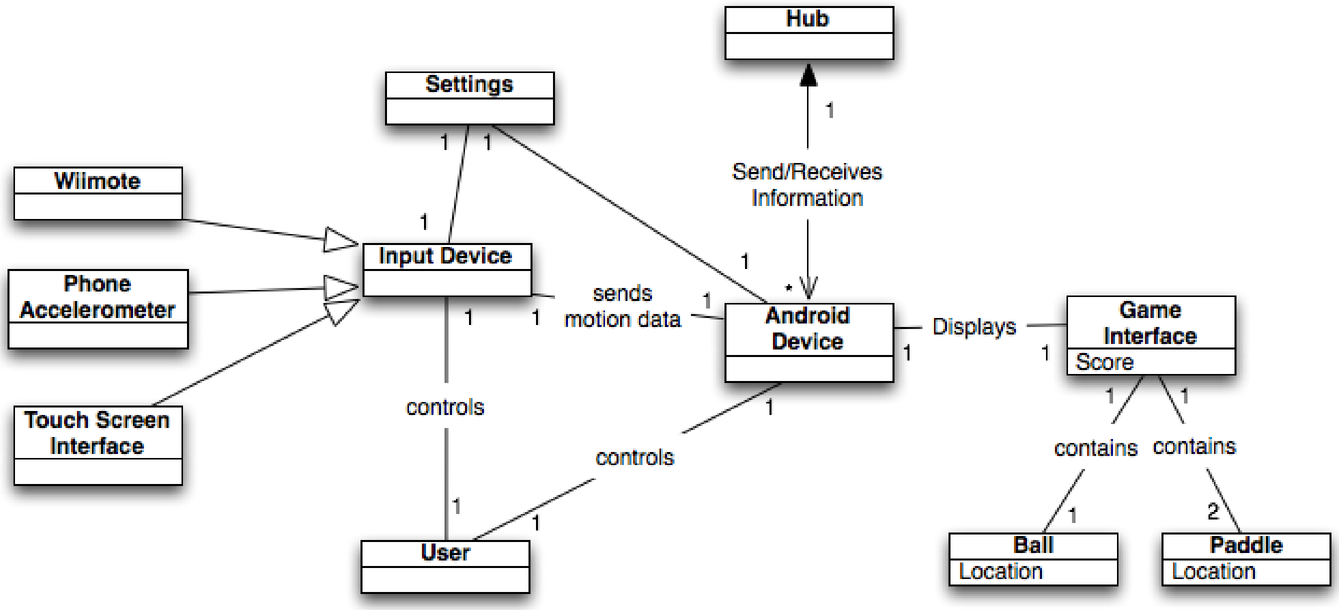

Figure 2: Class interactions and attributes outline the major system components.

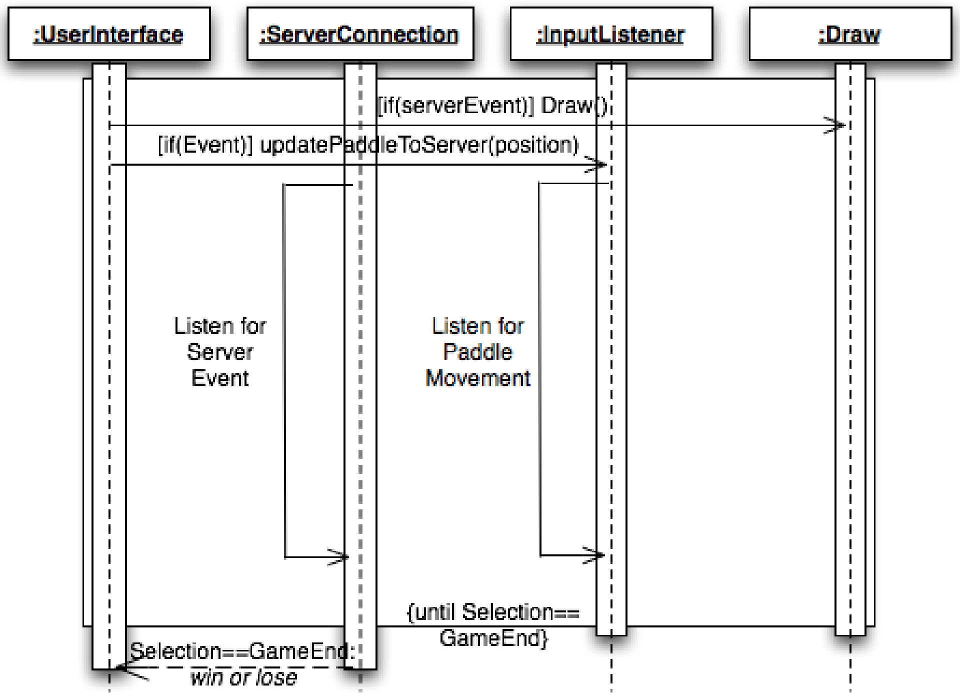

Figure 3:

over specific increments of time. We also utilized the information expert pattern in regards to drawing and redrawing the screen during game play. The server maintains the current game state, so the serverConnection class has knowledge of the paddle and ball locations. Consequently, the serverConnection class is given the responsibility of calling the draw method so that the screen will represent the current game state. We also maintain high cohesion in our Draw class. Draw is only focused on methods involving the act of drawing a figure on to the canvas. This makes the class highly cohesive and as a result, the code is easily understood and can be reused.

4 Implementation

5 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[**?**]) the brevity serves our team well.

5.1 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[**?**][**?**].

5.1.1 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.

5.1.2 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*.

5.2 Working with JavaScript

The majority of our JavaScript style guidelines are mimicked from Google[**?**]. 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.

5.2.1 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 (@*private* 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.

5.2.2 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.
- 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.

5.3 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[?]. 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.

5.4 Style Guide Removal

Due to the fact that we have redistributed tasks among the two phone teams, we have removed some items that no longer apply to the Android Development team. The following have been removed:

- CSS style guide section

6 Developer Documentation

6.1 Setting up the Development Environment

In order to develop for Android, you will need the Android SDK, the PhoneGap software, and some editor (for example, Eclipse). 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[8].

6.1.1 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[5].

Next, download the Android SDK from the Android Developer website[3]. The instructions for installation are found on the same site but on the installation page[4]. **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[2]. On your first run, you will be required to create an Android Virtual Device (AVD) that is simply a mock phone.

6.1.2 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. The download for PhoneGap can be found on the PhoneGap Android page[8].

6.1.3 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[6]. You then may want to install the ADT plugin for Android Development[1]. In addition, you may then want to install the plugin for PhoneGap Development[9].

An important note: When creating a PhoneGap application in Eclipse, it may be necessary to include the PhoneGap jar file in the libs folder of the application. To do this, right-click on the Eclipse project, and select "Build Path", then "Configure Build Path". If there is no PhoneGap jar file included, then select to "Add External JARs". Select the jar file downloaded earlier with the rest of the PhoneGap tools and click "Okay".

6.2 Retrieving the Source

In order to work with the source code of the project, you will likely want to use Git[7]. 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

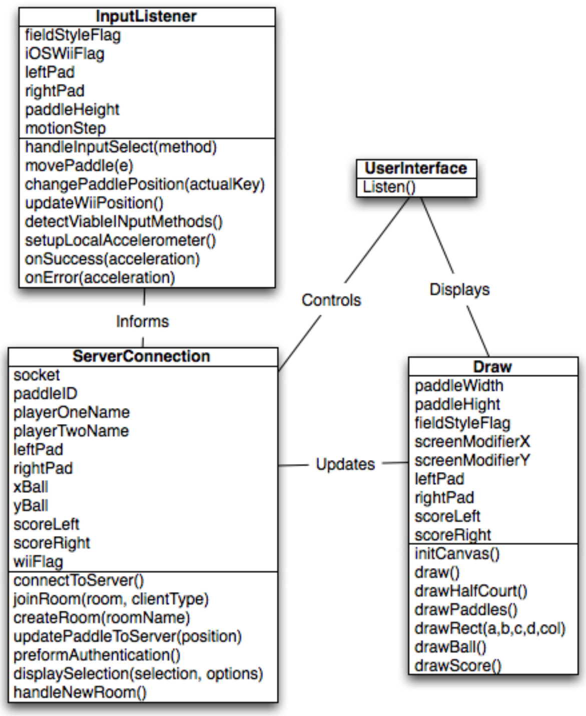

Figure 4: This model depicts the all the classes and their associations within our pong.js file. The ServerConnection class updates the Draw class, and continually calls to redraw the canvas. The ServerConnection also is informed by InputListener as to what input is being used and when it updates position. The UserInterface displays the refreshed canvas from Draw, and is controlled by ServerConnection as far as the game state.

`https://github.com/VirPong/human-pong` and `https://github.com/VirPong/www`, placing the *www* repository in the same place mentioned above.

7 Class Diagram

8 Database

In order to enforce security of the highest, the smart phones do not deal with the user password which is the only thing that allows access to personal information stored with the account. Instead the phones only store the user name and a pin number that grants the user limited access. This means that all major account settings and changes can only happen through the website. The account information saved on the phone uses HTML5 localstorage and more information and tutorials on localstorage can be found at `http://www.w3schools.com/html5/html5_webstorage.asp`. It uses JavaScript to store and access data and provides a more efficient replacement for cookies in webpages. It provides easy saving and accessing using a lookup key for each pin number and account name.

9 Reflections

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 had more manager meetings than at the beginning 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 have reached 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 couldnt 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.

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).

References

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

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

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

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

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

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

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

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

[9] *PhoneGap for Android with JSLint/JSHint 1.2*, `http://marketplace.eclipse.org/content/phonegap-android-jslintjshint`, 2011.