Final Report: Android Development Team

Jillian Andersen, Jordan Apele, David Ruhle, Kyle Wenholz

December 12, 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.

- 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 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 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.

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

1. Android device signals a game pause to user.
2. User is prompted to wait or leave the game.

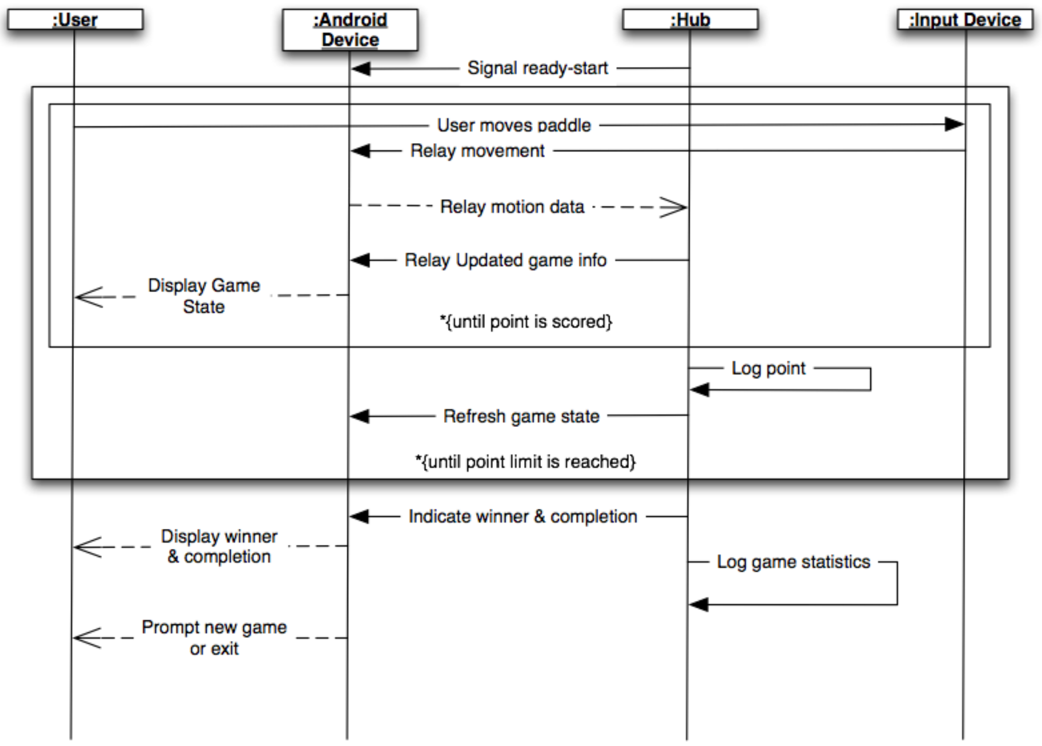

Figure 1: The role of our system in gameplay is described through the system sequence diagram above.

3. The user elects to wait.
4. Hub signals that the other player is reconnected.
5. Game resumes.
6. Return to 6 in main scenario.

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 device prompts the player to authenticate.
4. The player provides credentials.
5. The device verifies credentials with the hub.
6. The player is prompted with a list of various game rooms.
7. Player selects the desired game.
8. The device requests the selected game from the hub.
9. The player is notified of which paddle she is.
10. The hub waits for an opponent then signals a ready-start to device.
11. The device loads an initial game state displayed to the player.
12. 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.

4a) 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.

8a) 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 10 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 Statistics - 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 looked at her statistics.
- The player can compare hers to everyones.

Scenario:

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

Alternatives:
3a) 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 3.

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.
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 Nonfunctional Requirements

2.2.1 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.

2.2.2 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.

2.2.3 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.

2.2.4 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.

3 Diagrammatic Depictions of the Product

4 Implementation

5 Developer Documentation

5.1 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].

5.1.1 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.

5.1.2 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].

5.1.3 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".

5.2 Retrieving the Source

¡¡¡¡¡¡¡ HEAD 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 `https://github.com/VirPong/human-pong` and `https://github.com/VirPong/www`, placing the *www* repository in the same place mentioned above. ======= 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: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. ¿¿¿¿¿¿¿ upstream/master

6 Reflections

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.