LARGE GROUP GAMES WITH A MOTION- AND ORIENTATION-SENSING GAME CONTROLLER

ILHAM ABIYASA SUHARDI

Master of Science

Digital Media
Universität Bremen

August 2008

Ilham Abiyasa Suhardi: *Large Group Games with a Motion- and Orientation-Sensing Game Controller,* Master of Science, © August 2008

SUPERVISORS:
Prof. Jörn Loviscach
Prof. Jürgen Friedrich

To my parents,

who always lovingly support my journey in life.

To my wife,

who has always been the source of my inspiration.

ABSTRACT

Large group gaming can be described as a gaming activity played by a group of people gathering at a certain place. In this project, group game is defined as a big size video game played by hundreds of players. I would like to introduce a large group gaming utilizing the Wii Remote, a wireless game controller from Nintendo Wii. The controller is equipped with a motion and infrared sensor allowing the players to control the game by using their hand gestures or pointing at the screen.

The research goal can be formulated into three questions: (1) How can a system support a large number of Wii Remotes? (2) What kind of possible interaction can be done using the Wii Remote in group gaming? (3) How does the user perform using those interaction methods?

To address these questions, a group game prototype is developed. The system consists of a game server and several input clients which connect the Wii Remotes. Several playtests are done using the Wii Remote as a pointing device, a motion sensing controller, and an ordinary gamepad.

Keywords: Wii remotes, Wiimote, large group interaction, large group gaming, game design, motion sensor, pointing device, video game interaction.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. I wish to express my gratitude to my first supervisor, Prof. Joern Loviscach who was abundantly helpful and offered invaluable assistance, support and guidance.

I am thankful to my second supervisor, Prof. Juergen Friedrich, for his generous assistance during this time and also previous project.

Special thanks to all my friends from Master of Digital Media Bremen class 2004-2007, especially the Stammtisch group.

I would like also to thank Marion Wittstock for her guidance and help during my study in Bremen.

Finally, I take this opportunity to express my profound gratitude to my beloved wife and parents for their moral support and patience.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ACRONYMS

API	Application Programming Interface
IR	Infra Red
LED	Light Emitting Diode
IDE	Integrated Development Environment

1

INTRODUCTION

1.1 LARGE GROUP GAMING

Large group gaming can be described as a gaming activity played by a group of people gathering at a certain place. In this project, group game is defined as a big size video game. The system requires a large screen to display the game and is played by hundreds of players. The game usually takes place in a big hall with a large screen or cinema where the audience sits facing the screen.

From the previous large group interaction projects, there are several ways for the player to interact and participate in the game:

1. Using a bi-color paddle [Cin01] [Car94]. The player shows the paddle and the mounted camera will track the number of paddle shown by the players.

2. Tracking the audience movement. The mounted camera tracks the audience movement. The audience leans their body to the left or right to control the game [MAPS02].

3. Pointing laser pointer to the screen [MAPS02]. The player point his/her laser pointer to the screen and the mounted camera will track the laser points.

4. Using additional wireless device [FP04]. The device is small enough so the user can hold it in one hand or wear it. It is motion sensitive and able to send RF signal every time the player shakes the device.

All of them are using simple interaction methods since the audience has a little or none experience in gaming.

1.2 MOTIVATION

In this project, a new interaction method is introduced. The large group game system is played using a game controller from Nintendo, the Wii Remote. This game controller is different from the others due to its motion sensing and IR sensor feature. The IR feature enables it to be a pointing device.

Nintendo promotes a new way of playing game using the Wii Remote with their new game console, the Nintendo Wii. It is played by swinging, shaking, pointing, or tilting the controller. The effort is successful since the Nintendo Wii attracts casual gamers and people who never play games before [SK08].

I would like to introduce the use of Wii Remote in large group gaming. These are the reasons why I believe that Wii Remote is suitable for large group gaming:

1. It's an inexpensive game controller. For about 40 Euros, one can get a device equipped with a good motion and infrared sensor, eleven buttons, vibration feature, and Bluetooth connectivity.

2. People can relate Wii Remote with gaming activities. Nintendo have done a big marketing to attract people who never play video game before. There is a bigger chance that people knows Wii Remote and relates it with video game.

This project focuses on development and experiment of a large group game played by using Wii Remotes. The goal of this research can be formulized as these three questions:

1. How can a system support a large number of Wii Remotes?

2. What kind of possible interaction can be done using the Wii Remote in group gaming?

3. How does the user perform using those interaction methods?

To address these questions, a prototype of a group game is developed and several tests are done using the Wii Remote to control the game.

1.3 SYSTEM OVERVIEW

My system is designed to connect as many as possible Wii Remotes to support large number of players. Each player uses his or her Wii Remote to control the game. The system consists of a game server and several input clients. The input clients are responsible in connecting

the Wii Remotes while the game server is responsible for displaying and updating the game based on the input data from the connected Wii Remotes.

To explore the interaction using Wii Remote in a large group gaming, a game is designed and implemented on top of the system. The game can be played by tilting, pointing, or pushing the Wii Remote's buttons. The game will be played in single player mode and group mode, where two to three players are grouped in a team.

Finally, a prototype version of this system is implemented and tested by a smaller group of audience. The player's performance and behavior during the game will be recorded.

2

BACKGROUND

This chapter describes the background of this project. Several related projects will be explored, followed by the Wii Remote and the reason of choosing the Wii Remote for group gaming interaction.

2.1 RELATED WORK

There are three related projects related to large group gaming. These projects are focusing on simple interaction method which can be played by a wider range of audience.

1. Cinematrix, an interactive system by Rachel Carpenter and demonstrated in 1991.
2. Maynes-Aminzade's audience interaction work, which introduced three techniques for audience interaction using image processing.
3. A large group musical interaction using disposable wireless device by Feldmeier.

2.1.1 *Cinematrix Interactive Entertainment System*

At SIGGRAPH in 1991, Loren and Rachel Carpenter showed the Cinematrix, an interactive system which allowed an audience of 4000 people to control the onscreen game using paddles [Car94] [Cin01]. The audience sat in a big hall and the game is projected on a big screen, just like in a movie theater. Each of them was given a paddle which has two reflective sides—the red and green side (see Figure 2.1a). The audience controlled the game by showing the red or the green side of

(a) The paddle **(b)** The audience

Figure 2.1: The audience use paddles to control the game in Cinematrix [Cino1]

the paddle. Using an attached video camera on the ceiling, the system captured the audience paddles status, and updated the game.

Based on the audience position and the number of red and green paddles shown by the audience, several games can be played by a large group of people. One of them is *Dog and Cat*, a variant of the old video game PONG. The audience is divided into two teams, based on their sitting position. Each team control a fence to protect the cats from a dog which run back an fourth (see Figure 2.2). The fence acts like the paddle from the PONG that can bounce away the dog when it hit the fence. The team shows the green side of the paddle to raise the fence position and the red side to lower it.

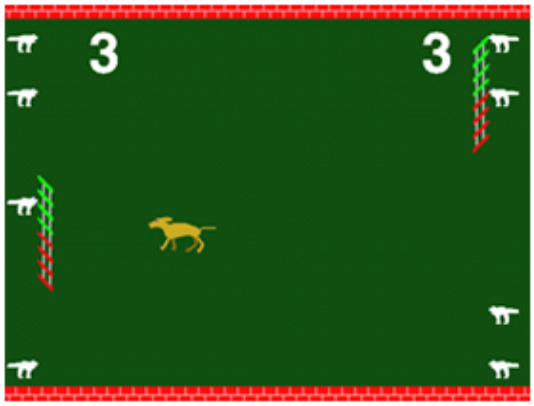
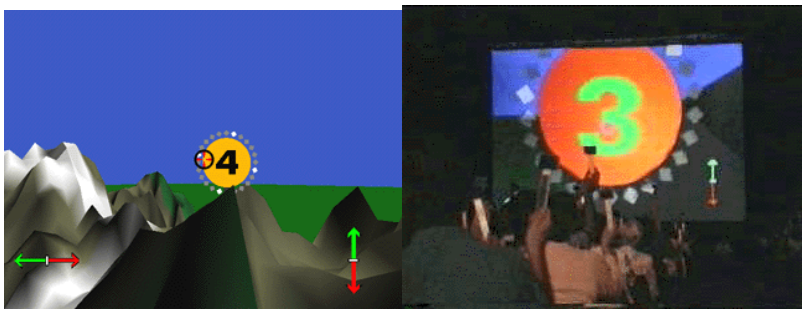


Figure 2.2: DOG and CATS from the Cinematrix [Cino1]

Another team based game from Cinematrix is the Air Strike, a flight simulator. The audience is divided by two groups based on their seat. This time they are not competing but working together to fly a plane.

The left side group controls the horizontal movement of the plane while the right side group controls the vertical movement. The audience flies the plane to avoid obstacles, goes through some floating rings, and lands the plane safely.

The most interesting part about these games is the team coordination. There is a spontaneous non-verbal communication among the team member to decide the movement. The effect is quite exciting and perfect for team building.



(a) A screen shot of Air Strike **(b)** The audience playing Air Strike

Figure 2.3: In Air Strike, the audience fly a plane avoiding obstacles, flying through several floating rings, and land the plane safely [Cino1]

2.1.2 *Maynes-Aminzade's Work*

In 2002, Dan Maynes-Aminzade, presented three techniques for interactive audience participation [MAPS02]. The setting was still using a video camera but unlike the Cinematrix, the audience didn't use paddles. These three techniques are:

1. **Audience movement tracking**. The audience leans left or right to move and control the onscreen objects.
2. **Object shadow tracking**. The audience bats a beach ball into the air while its shadow, projected on the screen, used as a cursor.
3. **Laser Pointer tracking**. The audience points the laser pointer to the screen and the camera tracked the laser dots on the screen.

Maynes-Aminzade made several experiments and tested them on movie theaters with audience of 150 to 600 college students. The games were projected on the screen and the audience played them using those three techniques.

The audience movement tracking allows the audience to control the game by just leaning the body to the left or right, just like in Figure 2.4. The system captures the audience movement (leaning movement) and calculates the average movement. Using this technique, the audience played the game PONG by controlling the paddle, or to steer a race car in the game Pole Position, a classic F1 racing game.

Figure 2.4: The audience leans their body left or right to control the game [MAPS02]

The second technique from Maynes-Aminzade uses a beach ball. As the audience bats the beach ball into the air, the ball casted a shadow on the projected screen (see Figure 2.5).Maynes-Aminzade choose to track the shadow (instead the ball directly) since it is easy to identify the shadow regardless the size and the type of the ball. The system tracks the ball's shadow and uses it as a cursor or game object. On the experiment, the audience was playing Missile Command, where they controlled the ball's shadow to destroy the falling missiles from the sky.

The third technique requires each member of audience to point his/her laser pointer to the screen. One of the games that were played using this technique was the Whack-A-Hamster. The audience directed their laser at the hamsters to hit them. Other interesting uses of laser pointer are the live audience poll and trivia game. The participants point their laser to the displayed choices and the bar graphs continually update to show the audience's preferences until the time limit is reached.

2.1.3 *Disposable Wireless Device for Group Musical Interaction*

On 2004, Feldmeier and his team created a group musical interaction system which allows a large group of people to participate in an in-

Figure 2.5: Using the beach ball's shadow to play Missile Command [MAPS02]

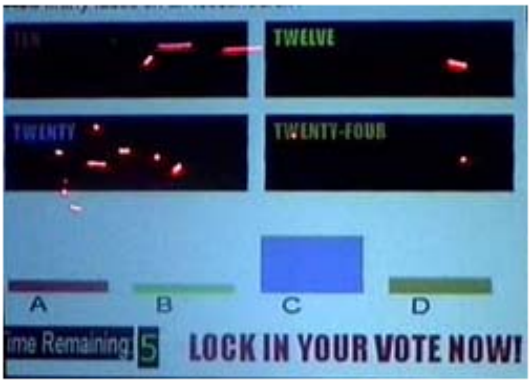
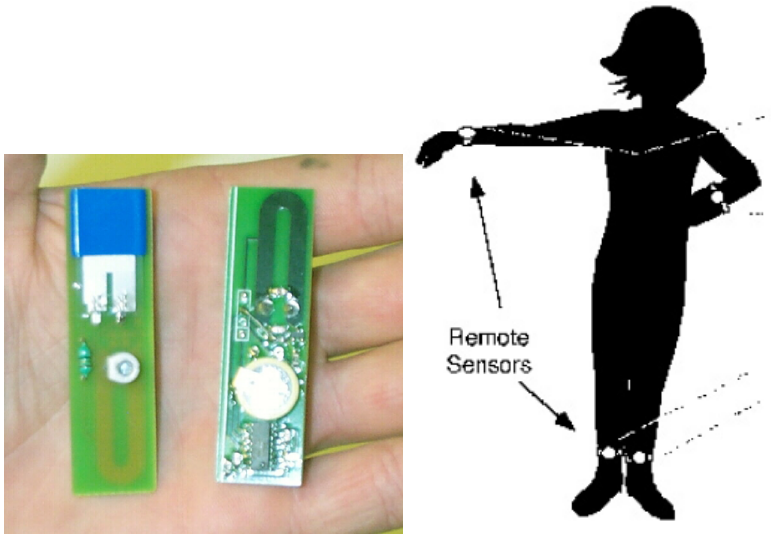


Figure 2.6: Live voting system using laser pointers [MAPS02]

teractive musical performance [FP04]. They built a wireless electronic sensor which is small and cheap so that it is reasonable to give away to a large number of people. They were able to create a working prototype that cost less than 3 dollar per unit (see Figure 2.7a). The controller is small enough that the user can hold or wear it. It can detect extreme motion such as dancing or clapping and transmit the radio-frequency (RF) pulse.

The main part of the system is the processing unit. It collects the RF data from the wireless sensors, analyzes the dance movement, and generates music and event lighting based on it (see Figure 2.8). The sensors' RF pulse are collected by several receiver stations and sent to the MIDI converter. The MIDI converter generates MIDI signals and sends it to the Macintosh G4 computer. These signals will be analyzed by its activity level and rhythmic features. These parameters will be mapped to musical content or lighting control information. By making



(a) The device

(b) Each participant can wear more than one devices

Figure 2.7: A working prototype of wireless sensor by Fredmeier [FP04]

the musical and lighting system responsive to the dance, the audience will have a feeling of controlling music.

The Feldmeier's system is suitable for heavy rhythmic dance music, such as house, techno, trance, and hardcore [Fel02].

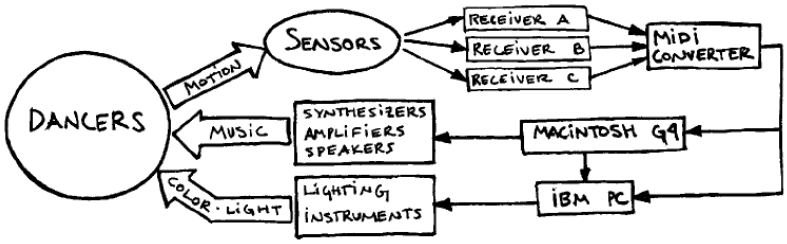


Figure 2.8: System block diagram from Feldmeier's system [Fel02]

2.2 WII REMOTE

In this section, I will explore Wii Remote and the idea why Wiimote is suitable and can be used for large group gaming.

2.2.1 *Nintendo Wii*

On 2006, Nintendo has released a new next generation console, the Nintendo Wii. The most innovative feature from Wii is the revolutionary controller, which is called Wii Remote or Wiimote, which allows players to control the game using their hand movement or gesture.

Figure 2.9: *Left:* Wii Remote (from [Nino7]). *Right:* Advertisements from Nintendo, promoting family and group gaming for Nintendo Wii (image taken from `http://us.wii.com/experience_gallery.jsp`)

Nintendo carefully design this control scheme in order to attract a broader audience, especially the non-gamer ones. This effort is followed by the marketing campaigns and the game design. Nintendo is always advertising its console or Wii games with a group of people or a family playing together in a living room with a relax and fun atmosphere. That is because the games for Wii are more casual and social game like Wii Sports, Mario Party, Wario Ware, or Brain Academy, which contain mini-games and can be played in a group.

2.2.2 *Capability*

Wii Remote is a game controller that has shape like a normal television remote control. Its motion sensor is able to detect 3-axis movement at 100Hz. It also has an IR sensor (a 1024x768 infrared camera) and able to recognize up to four infrared points, also at 100Hz. This mechanism allows the Wii Remote to be used as a pointing device [Wii07a].

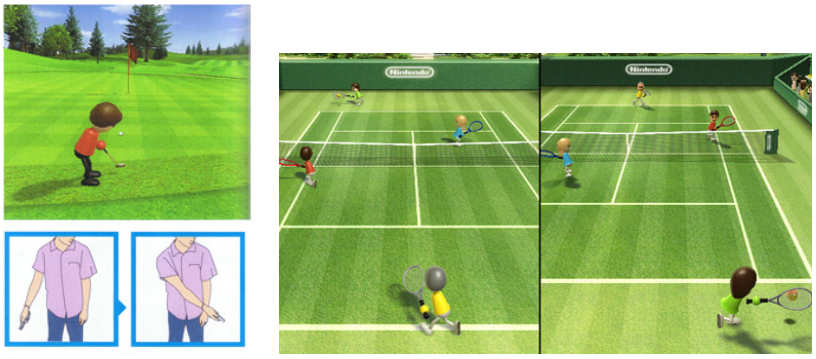
Wii Remote has several buttons which are a directional pad, three action buttons, and a trigger button. It has also a speaker, four blue LEDs, and vibration features for the feedback.

The Wii Remote's technical issues will be explored more on the Chapter 3.

2.2.3 *Wii Remote in Commercial Games*

Let's explore how the Wii Remote is used in several commercial games.

1. **Wii Sports**. This is a sports game developed by Nintendo. This game is bundled with the console Nintendo Wii, except in Japan. It contains five sports games: tennis, baseball, bowling, golf, and boxing.
   These five games are designed to be played using the Wii Remote motion sensing features. The player moves the Wii Remote mimicking the action in the real sport: swinging the golf club, a tennis racket, a baseball bat, or a bowling ball.



(a) Wii Golf **(b)** Wii Tennis

Figure 2.10: Tennis and golf game from Wii Sports. Images are taken from Nintendo website (http://www.nintendo.com/games/detail/1OTtO06SP7M52gi5m8pD6CnahbW8CzxE)

2. **Mario Kart Wii**. Mario Kart Wii is a kart racing game developed by Nintendo, featuring the characters from the Super Mario Bros series. It is played by tilting the Wii Remote as if as it is steering wheel. The Wii Remote is covered by an additional plastic case so it is shaped like a small steering wheel. This accessory is called Wii Wheel (Figure 2.11).

   The player holds the Wii Wheel on the air and steer it to the left or right as if as he/she is steering a kart. The game can also be

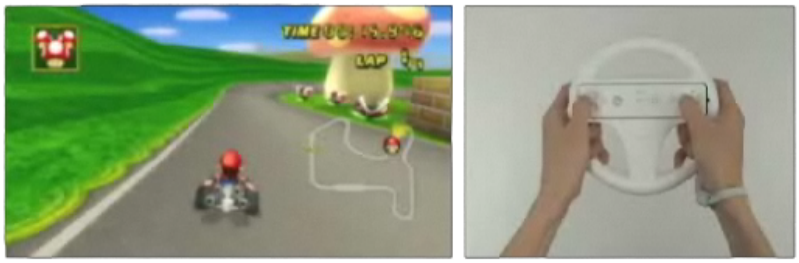


Figure 2.11: Playing the Mario Kart Wii using the Wii Wheel. It can be played by using the Wii Remote only. Image taken from http://www.mariokart.com/

played by not using the Wii Wheel; just using the Wii Remote. The player holds the Wii Remote horizontally and tilts it to the left and right.

3. **Trauma Center: Second Opinion**. Trauma Center: Second Opinion is a surgical simulation game developed by Atlus. In this game, the player plays a role of a doctor or a surgeon at a hospital. The game mission is to save the patient life by doing a surgery in the operating room.

Figure 2.12: Trauma Center, a surgical simulation game for Nintendo Wii. Image taken from http://www.atlus.com/trauma_center/

Using the Wii Remote as a pointing device, the user points the Wii Remote as if as it is a surgical knife, scalpel, or other medical instruments. The player is trained to act and decide quickly,

what to do next and which medical tools should be used, before the patient's life vital reach zero.

2.2.4 *Wii remote as an input device*

Other than its technical specification, there are several other reasons why I believe Wii Remote is suitable for large group gaming, particularly this project.

1. It is a game controller and the audience will immediately think game as soon as they hold the device. This is good in order to build the gaming atmosphere for the audience.

2. Wii Remote allows intuitive control scheme for gaming. The use of motion control is more intuitive and appealing to a wider range of audiences than just pressing the gamepad buttons. For example, in the Wii Tennis game, the player has to swing the controller mimicking the actions of swinging a tennis racket in order to hit the ball. The same mechanism is also use in the game Wii Golf, Wii Baseball, and Wii Bowling. The players swing the Wii Remote as if as they are swinging a golf club, baseball bat, or a bowling ball. This kind of control scheme makes the Wii games are easy to learn and attract the people who never play console game before, like the elders.

3. It is possible to substitute the Wii Remote with the PlayStation 3 controller, the SIXAXIS (Figure 2.13), which also has Bluetooth and motion sensing ability. However, it is a two-handed device and has 12 buttons, which people associated it with complex controller for playing hardcore games [Ada06]. Hardcore games are hard and complex games, which require a lot of gaming experiences to play. This is also the problem of current game platforms. Its controller is getting more complex, which could mean more buttons, and could intimidate the non-gamer audience.

   On the other hand, the single hand controller and Wii advertisements encourage the player to think Wii Remote as a light saber, tennis racket, fishing rod, drum stick, or a Mexican maracas. People strongly relate Wii with casual and social game. There are many Wii games which are just collections of simple mini games and can be played in a group. Although Sony have been released several casual and social games on their PlayStation 2 before (like Singstar, BUZZ, and Eye Toy Play series), it is Nintendo which strongly emphasize the games that enjoyable for the whole family member [Iva07].

Figure 2.13: SIXAXIS, the PlayStation 3 wireless controller [Ento6]

2.3 PROJECTS USING WII REMOTE

The Wii's rising popularity is also followed by more and more researches and projects using the Wii Remote. Some people use the controller to generate electronic music that can be used by a DJ [Sofo7], to navigate an immersive 3D environment [Engo7], or even to conduct a virtual orchestra [BTL+07].

The Wii Remote is popular among the researchers or hobbyists because it is inexpensive, equipped with a built-in motion sensor, and it can connect to a PC using the Bluetooth technology. Several programming libraries and tutorials are also available in the Internet. Several websites which are not affiliated with Nintendo, such as WiiLi (`http://wiili.org`), provide links to several drivers and programming libraries for Mac or PC. These online resources allow programmers to develop software which can retrieve the input data from the Wii Remote.

These are short reviews of several projects using Wii Remote.

2.3.1 *Percussion project by Belcher*

Belcher made a prototype of Wii *Percussive* [Belo7] which generates percussion sounds—particularly drums sounds. The user holds and moves the Wii Remote as if it is a drumstick. When the system recognizes the movement as a drum stroke, it will play a sampled snare drum voice.

The system consists of an Apple Mac Pro and a Wii Remote (see Figure 2.14).

2.3.2 *Expressive Percussion Instrument*

On 2008, Heise and Loviscach from Hochschule Bremen created a percussion instrument simulator using Wii Remote [HLo8]. Unlike Belcher's project, this project focuses more on realistic percussion simula-

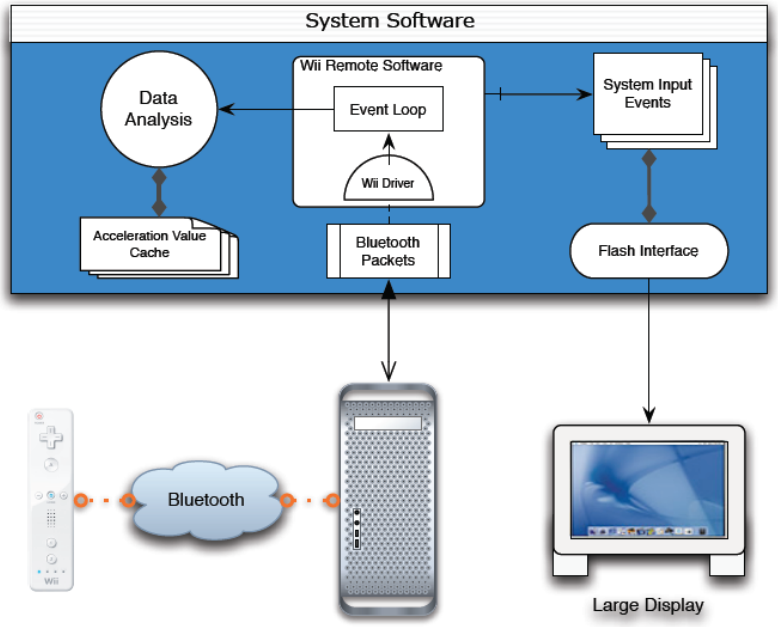


Figure 2.14: System diagram for Feldmeier's Percussive [Felo2]

tion. It simulates three percussive instruments: maracas, rainstick, and dugi dugi (Figure 2.15). These three devices have one common characteristic— they generate sound from collision of beads, whether it's beads to beads or beads to membrane.

This project generates the sound using physical simulation of particle collision; in this case, it's the beads. It utilizes commonly used game development technology, the physics engine and audio processor. The Newton Game Dynamics physics engine and AGEIA PhysX (hardware acceleration for physics engine) are used to calculate and simulate the percussive particles while the PortAudio is the audio processor which generates sound.

The Wii Remote is used to get the user input. The user shakes, rotates, or twirls the Wii Remote as if as it is a real maracas, rainstick, or a dugi dugi. The software uses the input from the motion sensor to calculate the force/impact and velocity which is done by the user with the Wii Remote.

The calculated input data is used by the physics engine to simulate the movement of the particles. Each particle moves and collides with other particles or with the simulated shell/membrane. Every collision data will be processed by the audio processor to generate sound.

Figure 2.15: A dugi dugi, a rainstick, and a maraca. These instruments are simulated in a project by Heise and Loviscach [HL08]

2.3.3 *Pinocchio: Conducting a virtual symphony orchestra*

A team from Technical University of Munich, has created a system called Pinnochio [BTL+07], which allows the user to conduct a virtual orchestra by using a regular conductor's baton or a Wii Remote. The virtual orchestra starts its performance when it receives a start command, which is issued by a special baton gesture. The orchestra changes tempo and volume according to the conductor's gesture. The system can recognize a start and stop gesture, and able to extract tempo and volume information from an easy-to-learn conducting motion.

2.3.4 *WiiArts*

WiiArts is an collaborative and expressive art project using Wii Remote, experimenting with video, image, and audio processing [LKGM08]. The users can participate or interact with the art installations using the Wii Remote's pointing or motion sensing features.

WiiArts applications are built using Max/MSP/Jitter. The art installation is projected on a big screen and up to three users can interact and work together on experimental image and sound processing. WiiArts contains four art installations: *Illumination*, *Beneath*, *WiiBand*, and *Time Ripples*.

In *Illumination*, the projection screen is a shared painting canvas, where up to three persons can draw simultaneously by pointing the Wii Remote. The system will draw the canvas by tracing the lines with live image of candlelight (Figure 2.16). The candlelight image source is captured from three burning candles using the webcams.

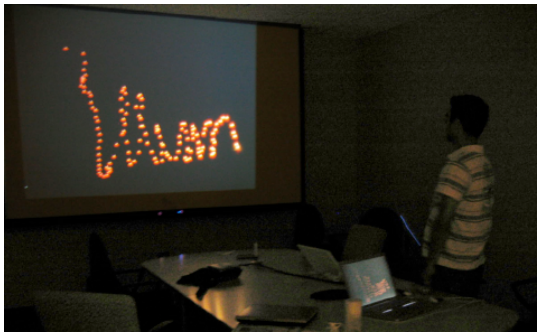


Figure 2.16: *Illumination*, painting the screen with live image of candlelight [LKGM08]

In *Beneath*, the Wii Remote acts like a flashlight. The users point the screen to expose portion of the image hidden in the dark layer. This experiment encourage collaboration between players which up to three players can play together to hunt a hidden item by pointing the Wii Remotes to the screen.

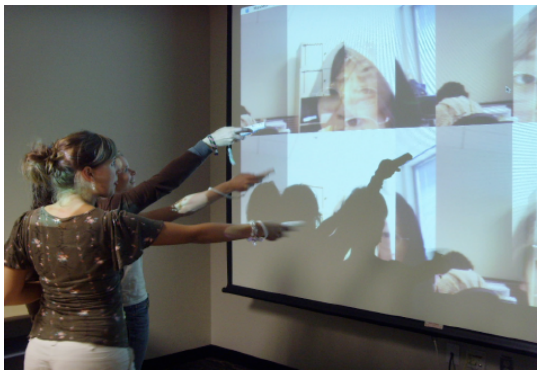


Figure 2.17: Three users interaction with *Time Ripple* [LKGM08]

Collaboration between users is also encouraged in *WiiBand*, a real time MIDI audio processing experiment. Each user moves their Wii Remote to change the musical instrument, the sound's pitch and volume. Together, the users can make a musical performance, just like a band. The system also displays visual feedback on the screen to accompany the performance.

2.3.5 *Johnny Chung Lee's Wiimote Projects*

Johnny Chung Lee, a PhD graduate student of the Human-Computer Interaction Institute at Carnegie Melon University, made several exper-

iments utilizing the Wii Remote's infrared camera [Lee07]. The user does not hold the Wii Remote, but put it on a table with the infrared sensor facing the user. The user wears the infrared sources while the Wii Remote acts as the infrared camera. Based on this mechanism, he made three applications which can be downloaded for free at his website (http://www.cs.cmu.edu/~johnny/projects/wii/).

The first application is the finger tracking. Lee uses an infrared LED array and stick reflective tape on his fingers. The Wii remote will track the reflected infrared from the fingers. This will allow the user to interact with a system by waving his/her hands in the air, similar to the interaction seen in the movie *Minority Report* (Figure 2.18).

Figure 2.18: Finger tracking using infrared source and Wii Remote by Johnny Chung Lee (image taken from http://www.cs.cmu.edu/~johnny/projects/wii/)

The second application is a low cost interactive whiteboard. By pointing the Wii Remote to a projection screen or LCD display, the user can have a low cost interactive whiteboard by using a pen white that has an IR LED in the tip (Figure 2.19). If the screen is projected on a table surface, then it will work as a tablet display. Up to four pens can be used at the same time.

The third application is head tracking for desktop VR displays. The user wears a head mounted sensor bar and the Wii Remote will track the user's head location. The system will render the view based on the user's head position and movement which create a realistic illusion of depth of space.

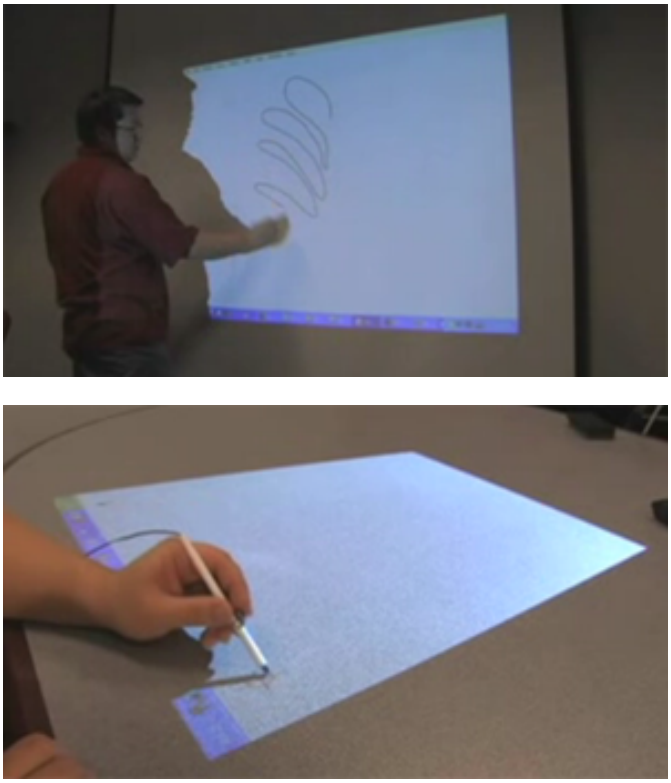


Figure 2.19: Low Cost Interactive Whiteboard by Johnny Chung Lee which can also be used for interactive tablet display (image taken from http://www.cs.cmu.edu/~johnny/projects/wii/)

3

TECHNICAL ISSUES

This chapter describes several technical aspects from Wii Remote, how it can be used on a large group gaming system, and how the system can connect a big amount of Wii Remotes. In the end of this chapter, the system architecture of large group gaming using Wii Remote will be explored.

3.1 BLUETOOTH LIMITATION

The Wii Remote uses Bluetooth technology to make wireless connection with the console Nintendo Wii. It is possible to pair a Wii Remote to a PC via Bluetooth connection. However, due to the Bluetooth limitation, a Bluetooth device can only connect to maximum seven other devices at the same time [Bha01].

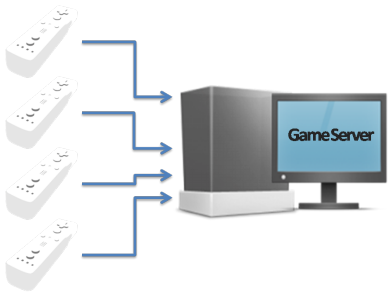


Figure 3.1: Using Bluetooth, one PC could connect up to four Wii Remotes

Despite the fact that a Bluetooth device can connect up to seven devices, a recent experiment by a research group from Carnegie Melon

University proved that one PC can only connect to six Wii Remotes at the same time [Kha07]. The sixth and fifth Wii Remote are not stable enough so only four Wii Remotes can be connected (Figure 3.1). The console Nintendo Wii can only connect maximum four Wii Remotes simultaneously.

3.2 CONNECTING MORE WII REMOTES

Having four controllers is definitely not enough for our goal to make a large group gaming. To solve this problem, I try to connect several PCs together, where each PC will have several connected Wii Remotes. If one PC can connect up to four Wii Remotes then theoretically, connecting four PCs together will allow the system to have input from 16 connected Wii Remotes (see Figure 3.2).

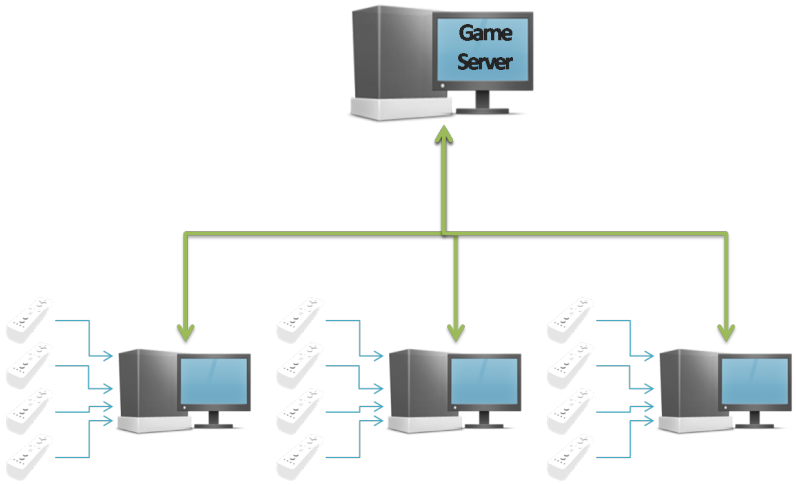


Figure 3.2: Several PCs, each with four connected Wii Remotes, are connected together via network

Our large group gaming system needs to have a client-server architecture in order to connect several PCs via computer network. The network and communication issues, including the our system architecture will be explored in the end of this chapter (section 3.6).

To connect a PC to several Wii Remotes, a Bluetooth capable hardware or dongle is needed. Once the connections are made, the system needs a Bluetooth driver and a programming library which can access the data from the connected device. By using the programming library, developers are able to make an application which able to make Bluetooth connection with the Wii Remote, retrieving the data from the

Wii Remote, and even activate several feedback functions like speaker, LEDs, and force feedback.

Before using the programming library, the Wii Remote needs to be paired with the PC. This process can be done using the Bluetooth software which is provided by the operating system. Until the end of this thesis time frame, there is no programming library which is able to pair the Wii Remotes by themselves (without manual setup from operating system).

3.2.1 *Pairing the Wii Remotes*

Wii Remote uses a non-secure, non-pairing Bluetooth connection which will give a lot of trouble to pair with, especially using the default Microsoft Windows Bluetooth stack. Not all Bluetooth adapters or driver could work with the Wii Remote [Pee07] [Wii07b]. The Wii community websites WiiLi (`http://www.wiili.org`) provide a database of working Bluetooth dongles and drivers with Wii Remote (`http://www.wiili.org/index.php/Compatible_Bluetooth_Devices`). Other sources of working and non-working Bluetooth devices and drivers can be found at WiiBrew website (`http://wiibrew.org/wiki/List_of_Working_Bluetooth_Devices`).

The pairing process of the Wii Remotes started by pressing and holding the Wii Remote's button '1' and '2' together. The Wii Remote's LEDs will start blinking and it will be detected by the Microsoft Windows Bluetooth software (or other Bluetooth software). The Wii Remote will be displayed as a Bluetooth device named **Nintendo RVL-CNT-01**. It must be paired without using any passkey.

For more details how to pair the Wii Remote with a Windows machine, see Section A.1.

3.2.2 *Wiimote Programming Library*

There are several programming libraries available for free on the Internet. Those libraries are not official from Nintendo and made by hobbyists who want to explore the possibility of using Wii Remote for their projects.

Community websites `http://en.wikipedia.org/wiki/Wii_homebrew` and `http://www.wilii.org` have several links to libraries, available for a wide range of platforms and programming languages. These are some of the popular libraries/drivers:

1. **WiinRemote** (`http://onakasuita.org/wii/`), a Wii Remote driver for Windows. It is one of the first available drivers for PC. The

author, a Japanese with the initial **tokkyo**, build and released the driver just one day after the Nintendo Wii was launched for the first time in Japan (2 December 2006).

2. **DarwiinRemote** (http://blog.hiroaki.jp/programming/darwiinremote/), a driver for Mac OS X, based on the source code of the WiinRemote.

3. **GlovePIE** (http://carl.kenner.googlepages.com), a driver for Microsoft Windows. It uses scripts that allow games and applications to use Wii Remote.

4. **wiimote-api** (http://code.google.com/p/wiimote-api/), a Wii Remote library written in C.

5. **Brian Peek Managed Wiimote Library** (http://www.brianpeek.com/), a popular library by Brian Peek for C# using .Net technology. A lot of projects are using this library, including Johnny Chung Lee's projects. The source code is available on the website.

6. **WiiYourself** (http://wiiyourself.gl.tter.org/), a native C++ library which supports most features of Wii Remote including the Bluetooth stacks and audio. It is based on the source code from Brian Peek.

The system uses Brian Peek's library because it is simple to use and it supports C#.net, which is the programming language for my system prototype development. I use Brian Peek's library version 1.5.2 which can connect up to four Wii Remotes. This is the feature that is needed most since most other libraries can only connect to only one Wii Remote.

For more details about Brian Peek Wiimote library, see Section A.3.

3.3 WII REMOTE MOTION AND TILT SENSOR

The Wii Remote has an integrated 3-axis linear accelerometer ADXL330 from Analog Devices [Wii07a]. The component, positioned under the big button 'A', is able to measure a range of ±3g with 10% sensitivity [Wii08]. This 3-axis system allows the Wii Remote Wii to have six degree of freedom: three linear movement directions (X, Y, Z) and three rotation angles (pitch, roll, yaw), as shown in the figure 3.3.

The Wii Remote accelerometer measures the linear acceleration in a free fall of reference. This means that if the Wii Remote is in free fall, it will report zero acceleration (X, Y, Z are zero). If the remote is put on top of a plain surface with the 'A' button facing up (just like on

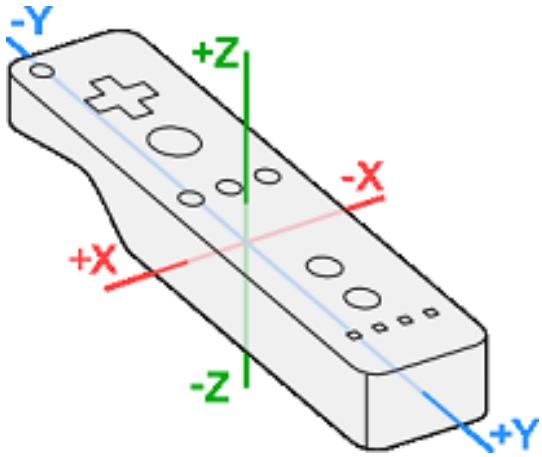


Figure 3.3: Wii Remote Coordinate System and Tilt Sensing (image taken from http://wiibrew.org/index.php?title=Wiimote)

the Figure 3.3), it will report an upward acceleration (with positive Z value, zero X, and zero Y) equal to the gravity acceleration g. Based on this, I can calculate the tilt from the acceleration outputs, although this will only be accurate if the Wii Remote is reasonably still.

3.4 WII REMOTE INFRARED SENSOR

Nintendo Wii includes a plastic bar called *Sensor Bar*, as seen in Figure 3.4a. In order to use Wii Remote as a pointing device, the Sensor Bar must be centrally placed, under or on top of the television.

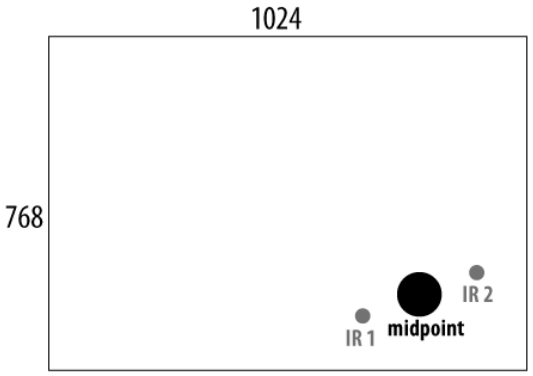
(a) The Sensor Bar

(b) Infrared LEDs on the Sensor Bar

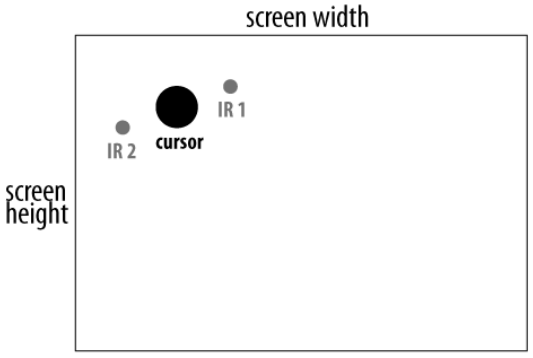
Figure 3.4: Wii Sensor Bar with its Infrared LEDs [Nino7] [Wiko7]

The Sensor Bar is not actually a sensor. It is a 20 cm bar which has ten infrared LEDs, with five infrared LEDs being arranged at each end of the bar. Figure 3.4b shows the infrared LEDs captured from the Sensor Bar by using a digital camera. The Wii Remote, using its PixArt Optical Sensor, will capture the infrared from the Sensor Bar [Wii06]. Wii Remote uses Sensor Bar so the pointing device can be used regardless of size or the type of the screen.

When the user points his/her Wii Remote to the screen (Sensor Bar), the controller, using its 1024 x 768 infrared sensor, will track up to four infrared spots and send the data to the console. After identifying two spots and the distance between those spots, the Wii CPU will calculate the cursor or pointed position.



(a) IR Points captured by Wii Remote's IR sensor



(b) The IR points and the cursor position on the screen

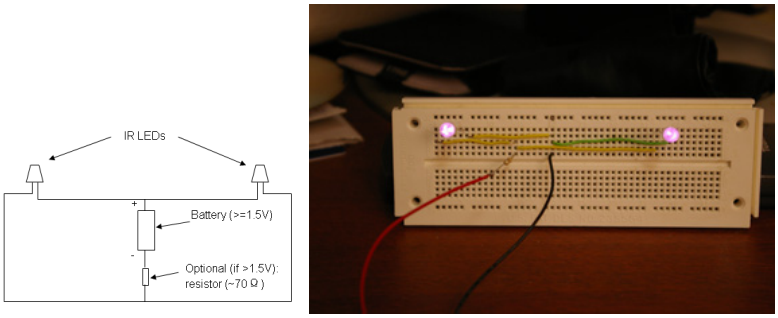
Figure 3.5: A simple conversion from IR positions captured by the Wii Remote's IR sensor (a) to cursor position on the screen (b). The cursor position is basically the midpoint of two IR points.

GlovePIE, one of the Wii Remote drivers, provides several scripts for calculating the cursor position based on the infrared points [Ken07].

It has scripts that use the data from the accelerometer. A simple and commonly used method to find the cursor position is by calculating the midpoint from the two received Infrared points. The points are normalized to the screen size to get the exact cursor position. The Figure 3.5 shows a simple conversion from the IR points to the cursor position on the screen.

3.5 SENSOR BAR MODIFICATION

The simplicity of Sensor Bar enables it to be replaced by any infrared resources, such as a pair of candle or flashlight [Wik07] [Hac07]. There are many guides for making your own Sensor Bar by using two or more Infrared LEDs and some AA batteries. By making the Sensor Bar replacement, we don't have to own a Nintendo Wii to activate the Sensor Bar.



(a) The Diagram **(b)** The Simple Sensor Bar

Figure 3.6: A Simple Do-It-Yourself Sensor Bar using a pair of infrared LEDs (image taken from http://wiibuilder.com/archives/6)

Another reason to modify the Sensor Bar is the limited view area of the Wii Remote's infrared sensor. In the project Wiimedia [SGR07], the view area of Wii Remote infrared sensor was measured and the result was about 36 degrees. Using the standard Sensor Bar, the user can only use Wii Remote with a distance between 2 to 4 m from the sensor bar or screen (see Figure 3.7). This is because the Wii Remote needs to detect minimum two infrared points in order to calculate the pointed position.

For large group games, it is necessary to support more than four players. In order to make the audience feel comfortable, they need to sit much further from the game screen. This is a problem if the system uses the standard Sensor Bar which has only 4 m distance. Therefore,

it is necessary to expand the distance between the pair infrared points to more than 20 cm. By making the distance further, the Wii Remote range will be extended to more than 4 meter. In order to achieve that, the IR LEDs need to be replaced by stronger ones.

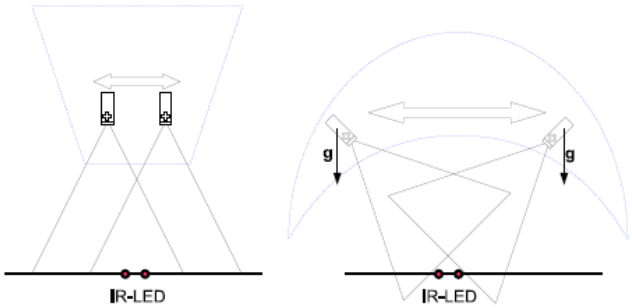


Figure 3.7: The Wii Remote needs to detect two infrared spots in order to be used as a pointing device accurately [SGR07]

3.6 SYSTEM ARCHITECTURE

My group gaming system has client-server architecture, formed by one game server and several input client machines. The input client is responsible for getting the input data and handling the Wii Remotes using Bluetooth connection. The game server handles the game engine and display. The input client sends the Wii Remote data to the game server via network connection (see Figure 3.8).

3.6.1 *Input Client*

Each input client is responsible to make Bluetooth connection with the Wii Remotes, retrieving the input data from the connected Wii Remotes, and send the data to the game server. The input client machine can handle up to four Wii Remotes. Therefore, for every four Wii Remotes, the system needs one input client machine. For example, to handle 15 Wii Remotes, the system needs a minimum of four input client machines.

The input client machines are placed among the audience to divide them into small groups of four based on their position. This will handle the weak Bluetooth signal when the audience is in a radius more than 10 meters.

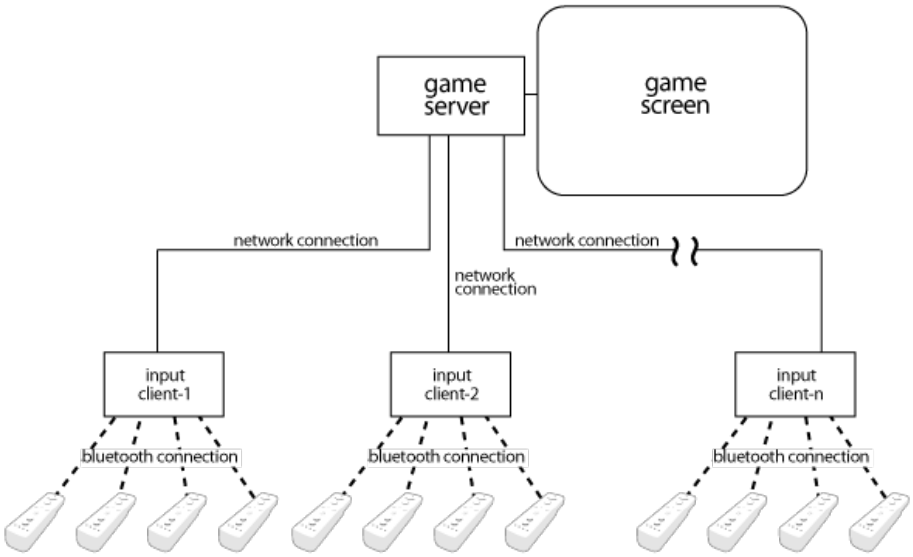


Figure 3.8: The system architecture of Large Group Gaming using Wii Remote

Each input client machine has its own ID. Every connected Wii Remote will get a local ID from the input client. Both input client ID and the Wii Remote ID are sent to the game server along with the input data. The game server can identify the source of the input data using these two IDs.

The input client gets the input data from the connected Wii Remote and sends the data to the server. Both tasks will be executed for 15 to 60 times per seconds so the game will be responsive to the user's Wii Remote. The input data will be displayed also on the screen of the input client machine.

The implementation of the network issues, such as the network protocol and the sent packet data, will be explained on the implementation chapter, subsection 5.1.6.

The input client is managed by a game operator. The game operator has to make sure that the player's Wii Remote is connected to the input client before the game session starts.

3.6.2 *Game Server*

The game server is responsible for processing all the input data from the input clients, updating the game, and displays the game to the screen. Just like the input client, the game server is managed by the game operator. He/she is responsible for moderating the audience, starting the game, and ending a game session.

There are other things which the game server does:

1. Maintaining game status like player's score, game timer, game character position, and game item position.
2. Handling collision detection between game character and game items. The game server will increase the player score whenever a character hit an item.
3. Managing player database, which contains player ID, Wii Remote ID, game character, and which group the player belongs to (for playing in the group mode).
4. Sending feedback data to the connected Wii Remote via the input client. For the feedback, the Wii Remote's vibration, LEDs, or speaker can be used.
5. Processing the input data from the Wii Remote and calculating the character speed. The game server considers the current game mode (single or group mode) and the input mode (pointing, tilting, and gamepad) for the character speed calculation.

3.6.3 *Stage Installation*

These are additional hardware or settings of the input client and game server machines, which are required for running a large group gaming using Wii Remote:

1. **Game screen**. The screen size must be from 20 inch to a size of a cinema screen, depending on the number of audience.
2. **Sensor Bar or IR Sources**. The IR sources must be placed in the center and below the screen. The size and the arrangement of the IR sources depend on the size of the audience and the range of their seats. All players' Wii Remotes should be able to detect at least two IR points in order to be used as a pointing device.
3. **Audience**. The audience sits in several rows in front of the large screen. The minimum distance from the screen depends on the audience numbers (the game screen should be visible by all audience) and the IR sources.
4. **Input client machines**. The input client machine should be placed near the group or the players (less than 10 m). Several input client machines can be placed among the audience in order to cover all Wii Remotes and to support zoning. These machines are operated by the game operator, especially in the beginning of the event, in order to get all Wii Remotes connected.

5. **Game server machine**. The game server machine is placed near the game screen. If the game supports sound feature, then the game server could be equipped with speakers placed near the game screen.

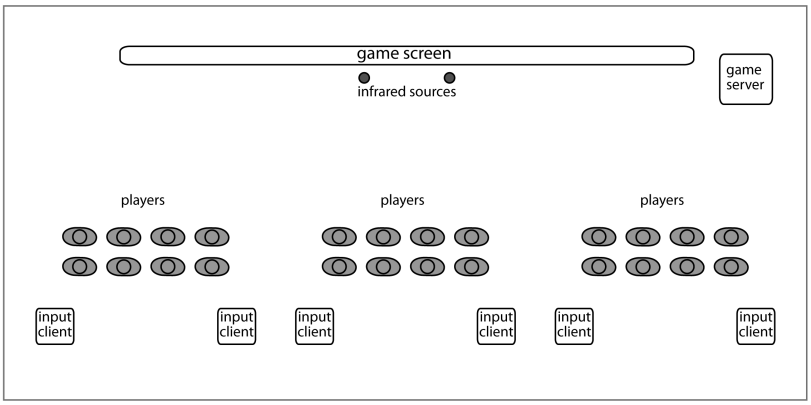


Figure 3.9: The stage installation for large group gaming using Wii Remote, with the position of the game screen, IR sources, the audience, the game server, and the input client machines

4

GAME AND INTERACTION DESIGN

To test the concept of the group gaming system using Wii Remote, a game prototype is designed and implemented on top of the system. The audience will control the game using their Wii Remotes. The game is designed to be simple to focus more on testing the concept of large group gaming using Wii Remote.

There are three interaction methods using Wii Remote (pointing, tilting, and pushing the buttons) which will be explored and used to control the game.

4.1 GAME IDEA

The idea of this game is to control the game character movement, collecting as many as possible items that appear randomly on the screen at a certain given time. When the timer ends, the character who collects the most items wins the game.

The game is inspired by real world fruit harvest festival or mushroom picking. Sometimes people compete in collecting the most fruits or mushrooms in order to make the job more fun and enjoyable.

4.2 TARGET AUDIENCE

The genre of this game is fast-action. The player has to decide the character movement quickly within a limited time while competing with other players.

The main target group of the game is casual gamers (with one to four hours a day playing game), specifically young male or female with age between six to 30 years old. Therefore, it is necessary to simplify the game design and control in order to support the main target group.

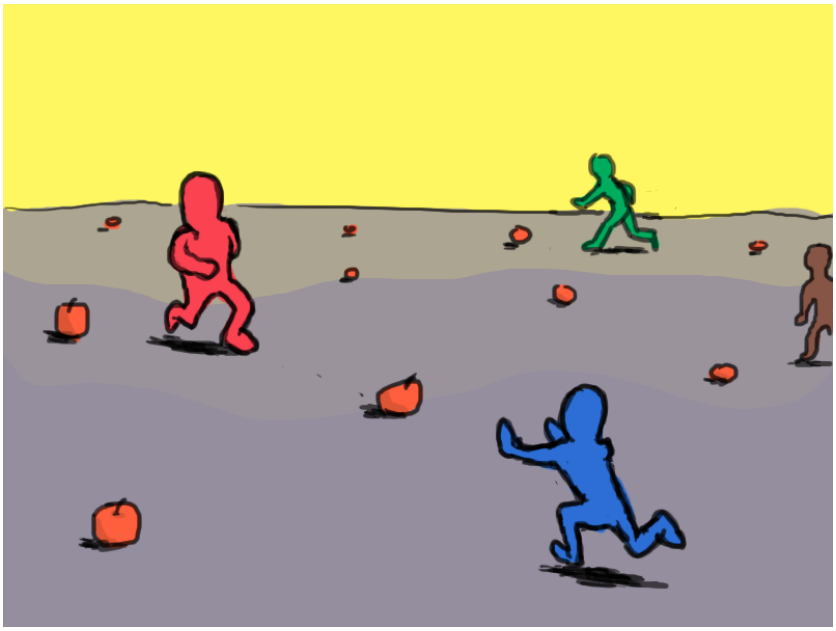


Figure 4.1: The conceptual picture of the game. Several game characters are running around catching the items

4.3 GAME MECHANIC

These are the basic game mechanics:

1. The game character has the ability to run at a different speed and to any direction. The player task is to control the character speed and direction. When the player does not give any command, the game character will stand still. The character velocity and running direction are based on the interaction or input methods which will be explained on subsection 4.5.

2. All game characters will appear at the same time on the screen. It is necessary for each character to have a differentiator (such as color or shape) so the player can recognize his/her game character.

3. The game area is the area where game characters and game items are rendered. The game characters and items cannot move or appear outside this game area.

4. The game character collects the game item by hitting it while moving. The hit item will disappear and the player will gain a score.

5. There are 10 to 20 small game items which appears randomly on the game area. When a character hits an item, the item will

disappear. Less than half a second later, the system will add another item and put it randomly on the game area. This is done to maintain a constant item number inside the game area.

6. One game session runs for 20 to 40 seconds. When the time is up, the player who collects the most items (highest score) wins. It is possible to have more than one winner if they all have the same score. Every time the game session restarts, the score will be reset.

4.4 THE GAME OBJECTS

This subsection describes the details and visualization of every game object.

4.4.1 *The Game Character*

The game character is the player representation in the game. One game character can represent one player or several players in the group mode. The character is visualized as a cartoonish human character with tiny body and large head. This type character is suitable with the game theme and it will make game atmosphere more fun and enjoyed by the main target group.

In addition to a different color and shape for each game character, an indicator arrow is put below the character as shown on Figure 4.2. Each game character will have a different arrow color. Players will know their characters since they get the color code before the game starts.

The character has two actions/animations: standing and running. When the character runs, it is facing the running direction. While running, the character can pick an item by hitting it without losing any speed. When a character hits another character, it will stop running. The character will still doing its running animation/movement without changing position (running on the spot) until the other character moves away or the player change the running direction.

The indicator arrow indicates the running speed and direction. The arrow points to the running direction and its length indicate how fast the character moves. This indicator arrow gives a better feedback to the user, avoiding missing characters.

Figure 4.2: The game character and its indicator arrow. It is used to give the player feedback of the character's movement.

4.4.2 *Game Item and Area*

Based on the game theme, a fruit is picked as the game item. To simplify the game, every game item has the same shape and will give the same score when it is picked.

The game area is basically the screen area where the system draws the game objects and characters. It is possible to use the whole screen but it is better to leave out some space for other game indicators such as scores and timer indicators.

Even though the system could draw other objects like non-playable background objects or obstacles, it only uses plain and empty game area. Adding unnecessary objects will only make the game more complex and distract the audience.

4.4.3 *The Game Objects Proportion and Visualization*

The game uses a screen with 1024 pixel width and 768 pixel height. It is the common size of a projection screen. The system could have supported more than one monitor but this game is designed to use one game screen.

The screen size must be considered when designing the game objects size, especially the character. If it is too big, the screen will easily be crowded with game objects. If it's too small, the player will have a difficult time to recognize the objects.

The game character will be displayed as a sprite with height between 100 to 150 pixels, and width between 50 to 60 pixels. Using this size, the screen can display up to 25 game characters without overcrowding the screen because only half of the screen area is filled. This will be enough for a prototype.

However, for a real large group gaming, the system needs to support 80 to 100 players at a time. A solution for this problem is to use group mode which enables several players to control one game character. The group mode will be discussed on subsection 4.6.

The game item size should be smaller than the game character. If it is too big, the game will be too easy because the player can hit the item easily.

The game item size will be 30 to 40 pixel width/height. Using this size, the system can draw up to 80 items.

The amount of items depends on how many game characters appear on the screen. For this game, the item number is twice as much as the characters. Based on the proportion diagram below (Figure 4.3), having 26 items for 13 players gives the game character enough room for moving and collecting item.

4.5 INTERACTION DESIGN USING WII REMOTE

Another purpose of this game is to test three interaction methods which are commonly used for playing game by using the Wii Remote. These interaction methods are:

1. **Pointing**. The Wii Remote is used as a pointing device. The player points it to the screen and the cursor will appear. This method requires a Sensor Bar or two IR points/source.

2. **Tilting**. The player holds the Wii Remote and tilts it to the left, right, backward, or forward.

3. **Gamepad**. In this method, the Wii Remote is used as if it is a normal gamepad, as shown in Figure 4.7. The game will abandon any input from the motion or IR sensor and only accept input from the buttons. The player holds the Wii Remote horizontally with both hands and use his/her thumbs to press the buttons. The left hand thumb pushes the directional pads and the right thumb pushes the button '1' or '2'.

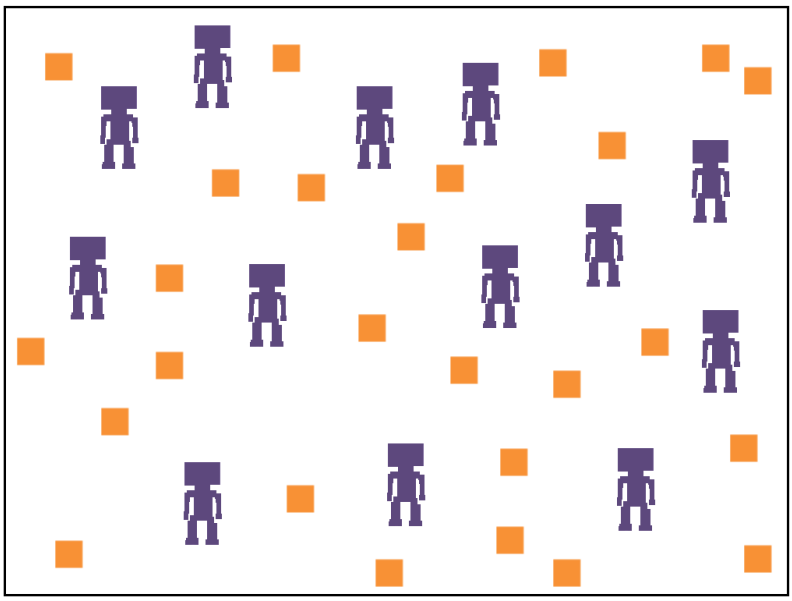


Figure 4.3: The comparison between the game characters, items, game area, and the screen. The small orange rectangles represent the game items.

All players use the same input method at the same time. There can be only one active input method. The input method can only change before or after a game session, not during the game play.

How can those input methods control the game? This issue will be explained on the following subsections.

4.5.1 *Pointing*

The character's speed and direction are determined by the distance between the game character and cursor (see Figure 4.4)

The character moves only when the Wii Remote button 'A' is pressed. When no button is pressed, the character will stand still. This will make the control mechanism easier for the player. If the character runs all the time, it will be harder for player to control the movement.

The character moves as if it's chasing the cursor. It will stop running once it is on the same position as the cursor. In order to make the character running all the time, the player has to move the cursor and keep a distance between the cursor and the game character.

One basic algorithm to determine the speed is to calculate the distance between the current character position and the pointed position by the cursor. The further the character with the pointed position, the faster it will move. The closer the character with the pointed position, the slower it will be. The distance will have maximum and minimum

Figure 4.4: The character's speed and direction are determined by the distance between the game character and cursor

threshold value to prevent the character speed exceeding the maximum speed and to make the character stop when it too close with the cursor or pointed position.

The basic algorithm of character speed calculation using the cursor is described in Listing 4.1.

Listing 4.1: Algorithm of character speed calculation using Wii Remote's cursor position

```
cursorPosition ← ProcessIRdata()

distanceX ← playerChar.x − cursorPosition.x
distanceY ← playerChar.y − cursorPosition.y
distance ← Math.SquareRoot(distanceX² + distanceY²)

{ Calculate speed with distance threshold }
if (distance < MinimumDistance) then
    playerChar.speed ← 0.0
else if (distance > MaximumDistance) then
    playerChar.speed ← MaximumSpeed
else
    playerChar.speed ← (distance / MaximumDistance) ×
        MaximumSpeed
end if
```

```
{ Calculate character direction using arc tangent }
playerChar.direction ← Math.Atan2(distanceX, distanceY)
```

4.5.2 *Tilting*

The user holds the Wii Remote vertically and tilts the controller to the left, right, forward, or backward. The game character's speed and direction are determined by how far and to which direction the user tilts the Wii Remote.

There are two ways to hold the Wii Remote for this input method: by using one hand, and both hand (see Figure 4.5). The first way is quite clear: holding the Wii Remote with one hand and tilting it in the air. The Wii Remote's buttons are facing the player so the player's thumb can easily press the button 'A'.

The second way is like the first method, but the player puts the bottom of the Wii Remote on top of the other hand surface. The player tilts the Wii Remote as if as it is a joytick or a shift gear of a car. The bottom part of Wii Remote, which is supported by the player's second hand, acted as an axis and does not move or only move slightly when the Wii Remote is tilted. More experienced players do not have to use this method because they might be able to hold the Wii Remote more stabile and able to use only one hand for tilting.

(a) Using one hand

(b) using the second hand to support the Wii Remote

Figure 4.5: Two ways of holding the Wii Remote for tilting

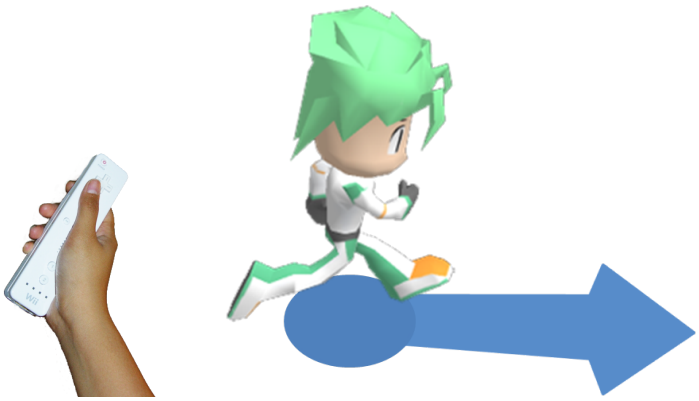
As described in Section 3.3 and on Figure 3.3, the Wii Remote motion and tilt sensor return the value of axis X, Y, and Z, ranging from -3.0 g

to 3.0 g. Tilting the Wii Remote without shaking or giving an additional force gives a value between -1.0 g to 1.0 g.

The default or base position for the Wii Remote in this input method is the Wii Remote stands vertically with IR sensor facing upwards. In this base position, the tilt sensor values {X, Y, Z} will be close to {0, -1, 0}.

From this position, tilting the Wii Remote to the left will get the X value closer to -1 while tilting it to the right will make it closer to +1. Tilting the Wii Remote forward will make the Z value closer to +1, while tilting it backward will make the value -1.

The game character speed can be calculated by adding X and Z values from the tilt sensor. Both values are added as two vectors and the result vector can be used to calculate the character direction and velocity. The value Y is abandoned. It is possible that the sensor value is smaller than -1.0 g or bigger than 1.0 g, especially when the player shakes or tilts the Wii Remote with force. Therefore, the system will make sure that the X and Z value is in the range by applying minimum and maximum value threshold.



(a) Tilt to the right **(b)** The character runs to the right

Figure 4.6: The game character's speed and direction are determined by how far and which direction the user tilts the controller.

Unlike the interaction method using cursor or pointing, the user does not have to press or hold any key to make the char run. Tilting the Wii Remote is enough to make the character moves. To make the char stand still, just hold the Wii Remote as vertical as possible. The algorithm of this calculation is shown in Listing 4.2.

Listing 4.2: Algorithm of character speed calculation using the Wii Remote tilt sensor values

```
tiltX ← wiimoteTiltData.x;
tiltZ ← wiimoteTiltData.z;

{ Value threshold }
if (tiltX < −1.0) then
    tiltX ← −1.0
else if (tiltX > 1.0) then
    tiltX ← 1.0
end if

if (tiltZ < −1.0) then
    tiltZ ← −1.0
else if (tiltZ > 1.0) then
    tiltZ ← 1.0
end if

{ calculate speed with value threshold }
sumVector ← Math.SquareRoot(tiltX^2 + tiltZ^2)
if (sumVector < MinimumValue) then
    playerChar.speed ← 0.0
else if (sumVector > MaximumValue) then
    playerChar.speed ← MaximumSpeed
else
    playerChar.speed ← MaximumSpeed * sumVector;
end if

{ Calculate character direction using arc tangent }
playerChar.direction ← Math.Atan2(distanceX, distanceY)
```

4.5.3 *Gamepad*

The user holds the controller horizontally, and pushes the directional pad to control the game character (Figure 4.7). This method utilizes the buttons and abandons the Wii Remote's motion and pointing sensing feature. The game character can only move to eight different directions because there are eight directions available on the Wii Remote's directional pad.

The character moves when the directional pad is pressed. The character moves with acceleration. This means the running speed increases until the maximum speed while the button is pressed.

The reason I use this interaction method is to test the player who is familiar with normal gamepad, or for an experienced and hardcore gamer. The Wii Remote is designed to be able to play in this hori-

Figure 4.7: How to hold a Wii Remote like a gamepad.

zontal mode or 'sideways' mode by Nintendo. One of the Nintendo games, Super Smash Bros, supports gamepad mode besides the input methods utilizing the shake feature in order to facilitate both amateur gamer and hardcore game. The amateur gamer would prefer tilting and shaking, focusing on fun. The hardcore game would prefer ordinary gamepad, focusing on score and performance.

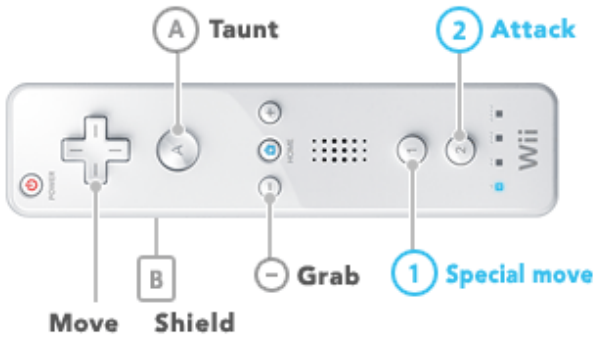


Figure 4.8: Sideways input mode in Nintendo's Super Smash Bros Brawl. Image taken from http://www.smashbros.com/en_us/howto/basic/basic07.html

4.6 GROUP MODE

In group mode, one game character will be controlled by a group of two to four players. All players in the group must work together to decide the character movement.

This group mode facilitates cooperation and social activity between the players. For most people, one of the reasons they play games is to have a social experience with their friends or family [Rou01]. In Cinematrix, there is a game where audience were cooperating to fly a plane. There were a rich conversation as the audience were shouting to suggest the best movement. This communication also happened in Aminzade system. While the audience pointed their laser in a trivial game, other players were shouting to persuade other players to point their laser to a certain position.

The group mode is a way to reduce the number of game characters on the screen. Since there is limitation of displayed game character, it is a problem when the number of player is more than 40. By making one game character controlled by four persons, the system will be able to support 100 players by displaying 25 game characters.

The game can be played using any of three interaction methods which have been explained before: pointing, tilting, or gamepad mode. By summing up the input vectors of all players in the group, the game character's running speed and direction can be determined.

The game character could end up frozen or not moving at all if all the resulting input vectors is zero or near zero. Imagine if two players are controlling a character. One of them is trying to move the char to the left while the other is trying to move it to the opposite direction. The character will move nowhere. This is the fun part of the large group gaming. Based on some experiments from the Cinematrix system, this group mode could initiate a verbal or non-verbal communication.

In group mode, each game character will have more than one arrow indicator (Figure 4.9). Each arrow represents the input data from every player who controls the character. This could be a potential problem if there are too many players in one group. There will be too many arrows and it will confuse the players. Two possible solutions are to reduce the arrow size or limit the number of players in the group.

In the previous large group gaming or interaction (Cinematrix, Aminzade, and Feldmeier), they did not used any individual avatar or indicator, only the sum or total input of the audience. This could cause doubt in the player's mind, whether they are really controlling the game. That's why there are a few players who intentionally make some strange input in order to see whether their effort affects or disturbs the total group input [MAPS02] [Cin01]. This is the reason why my system displays the indicator arrow for individual feedback.

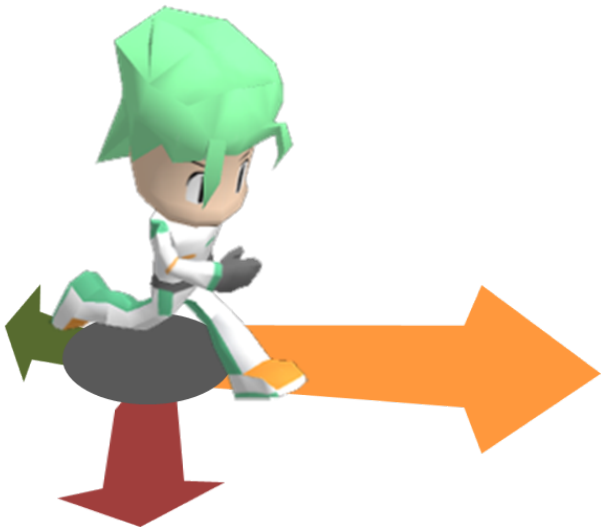


Figure 4.9: Several players can take control of a game character's movement

4.7 OTHER GAME VARIATIONS

The game mechanics described in the previous section are just the basic rules. It is mainly for casual audience and testing the concept of group gaming utilizing Wii Remotes.

It is possible to make other variation of the game by adding more rules, more complex game mechanism, and more objects. These will make the game more complex thus challenges more experienced player.

These are several possible additional rules or game objects for another game variation:

1. The ability to attack or stun. A character has the ability to do a special move to stun other characters thus slowing down their movement. If the hit character is carrying an item, the item will be dropped and the opponent can grab it away. To do this special movement, the player need to do a movement combination such as pressing a button while shaking the Wii Remote, or pressing a sequence of buttons.

2. One can defend his/her character from the attack by activating a special movement which is generated by a special input sequence using Wii Remote.

3. The ability to jump to reach a further distance. This can be done by combining several input methods like holding a button while shake the Wii Remote vertically.

4. Teaming up, where one team contains several game characters which are controlled by individual player. Forming a team makes

the attacking and defending more complex. Each team can do offensive or defensive movement by forming a special formation.

5. In the group mode, the game character can be replaced with something big and slow like a ship or a vehicle. By making the indicator arrow smaller, one character/vehicle movement can be controlled by eight players. The game rules can be changed to a ship race game to support this character visualization.

5

IMPLEMENTATION

As described on Chapter 3, my group gaming system for Wii Remote contains two major parts: the game server and the input client. In this chapter, the implementation of both parts including the game which is described on previous chapter will be explored.

The target of implementation is to build a prototype version of the system so it can be used to test the concept of large group gaming using Wii Remote. The prototype version supports a smaller audience, with a maximum of 16 players.

5.1 INPUT CLIENT

5.1.1 *Hardware Specification*

A PC-based computer is used for the input client machine. The machine must have sufficient processing power to get input from Wii Remote, process the input data, and send them to the server 30 times per second. Table 5.1 contains the hardware specification used for the development and testing.

The most important hardware for the input client machine is the Bluetooth capable hardware. For that purpose, a USB Bluetooth dongle or a laptop with built-in Bluetooth hardware is used. Since not all Bluetooth dongle or hardware is compatible with the Wii Remote, I make sure to use only the Bluetooth hardware and driver which are listed on `http://www.wiili.org` or `http://www.wibrew.org`.

Table 5.1: Minimum Hardware Specification for Input Client Machine

OS	Windows XP Home Edition
Processor	1.6Ghz single core processor
Memory	768Mb
Bluetooth Hardware	Acer BT-600 and built-in Blueooth hardware for Sony Vaio Laptop (VGN-SZ23)
Bluetooth Driver	BlueSoleil version 5.0.5 and built-in Sony Vaio laptop Bluetooth driver
Networking	LAN or Wireless LAN

5.1.2 *Software Specification*

The development of input client is using C# and .Net technology from Microsoft. For the IDE, the Microsoft Visual Studio 2008 Express Edition is used for editing the source codes, compiling, and debugging. It can be downloaded for free and it is a highly recommended software to develop application utilizing C# and .Net technology.

The details of the software enviroment and programming libraries used for the development is shown on Table 5.2.

Table 5.2: Development Environment for Input Client Software

Programming Language	C#
IDE	Visual Studio 2008 Express Edition
Additional Libraries	.Net 2.0, Brian Peek WiimoteLib v 1.5.1

5.1.3 *Implemented Features*

These are the implemented features on the prototype version:

1. Ability to detect and connect available Wii Remotes. The application can only connect with the paired Wii Remotes. Up to four Wii Remotes are supported by the input client.
2. Once the Wii Remotes are connected, the client will fetch the input data from them. This is done 30 times per second.
3. The input client manages the connected Wii Remotes by assigning them a local ID (from one to four).

4. The user is able to assign the input client a number for an ID. This ID will be used to differentiate several input clients when making connection with the Game Server.

5. The input client displays the information of the input data from the connected Wii Remotes. This information is updated 30 times per second.

6. The user is able to give the input client the IP address of the game server and the port number. Both are needed in order to make a connection with the game server. If no information provided, then the default data will be used.

7. The input client is able to establish a network connection with the game server. A socket connection with the protocol UDP is open based on the given IP address and port number.

8. Once the connection is open, the input client will send all input data to the server. This is done 30 times per seconds, until the user stops the connection or an error occurs.

5.1.4 *Connecting Wii Remotes*

Establishing a connection by pairing the computer with the Wii Remote is done using the Microsoft Windows Bluetooth Setup or other Bluetooth software or drivers such as Bluesoleil. See Appendix A, Section A.1 for a step-by-step pairing a Wii Remote to a PC.

The pairing process has to be done in order to make the Wii Remote available to the programming library. Up until the end of the time frame of this thesis, there are no programming library yet which can make connection and pairing the Wii Remotes without using additional Bluetooth driver or software.

To process the input data from the paired Wii Remote, the input client uses Brian Peek library. The latest version of this library supports multiple Wii Remotes and is able to access other features from the Wii Remote (battery level, vibration, and extension hardware like Nintendo Nunchuk). The input client only uses the basic features, which are:

1. Connecting the Wii Remote. Calling the method `FindAllWiimotes()` will connect all paired Wii Remotes on the machine and store the Wii Remotes into a list. Up to four Wii Remotes can be connected.

2. Getting the input data: two IR points, motion data from tilt sensor, and all button status. The IR points are two integer points X

and Y coordinate with range from (0,0) to (1023, 767).
The motion data are three floating points X, Y, and Z (one data for each axis). Each value is normalized from -3.0 to 3.0 which is the gravity value g. It can return a value smaller than -3 or bigger +3 if the Wii Remote is shaken instead of tilted, but the data is not valid to be used.

3. Turning on the LED on Wii Remote. This is done to indicate the Wii Remote numbering (will be explained on the next sub section)

For more details and the source code of how to use Brian Peek's Wii Remote library, please see Section A.3.

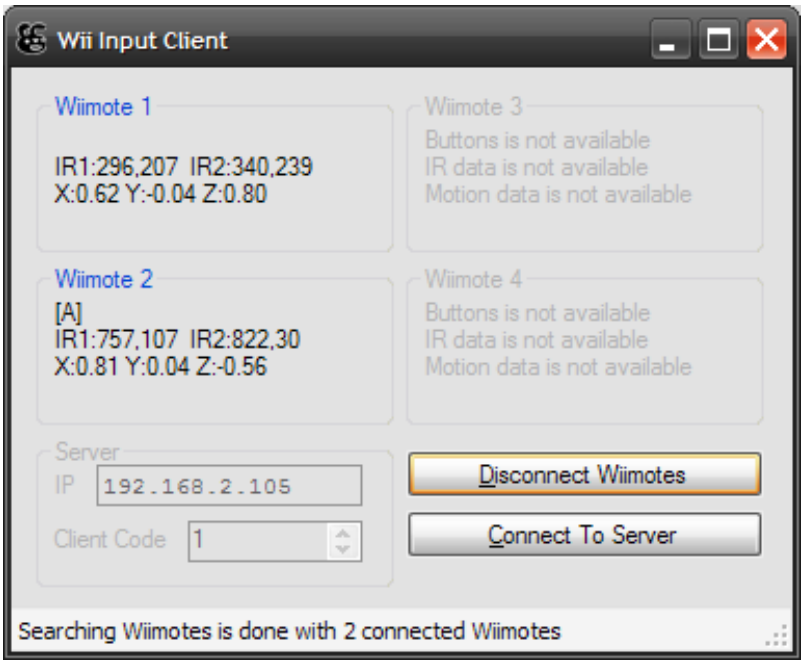


Figure 5.1: A screenshot of input client application, with two connected Wii Remotes. The second Wii Remote is holding the button A.

5.1.5 *Wii Remote and Input Client Identification*

Each input client machine has a number ID starting from one to n (n is the maximum number of active input client during the game). This number is given manually by the user. Therefore, it is the user's responsibility to maintain the uniqueness of input client's ID.

Each Wii Remote connected to the input client will be given a local ID from one to four. The input client is responsible giving the Wii Remote local ID. Since each Wii Remote has four LEDs, the input client just turn on the LED to indicate the Wii Remote local ID.

For example, an input client has three connected Wii Remotes. Those Wii Remotes are given local ID 1, 2, and 3. Therefore, the Wii Remote with local ID 1 will have its first LED on, the Wii Remote with local ID 2 will have its second LED on, and so on.

The game server identifies each Wii Remote using the global Wii Remote ID which can be calculated by using the Wii Remote Local ID combined with the input client ID. Since the maximum number of Wii Remote on each input client is four, then the system can calculate each Wii Remote global ID by using this formula:

$$\begin{aligned}\mathtt{GlobalWiiRemoteID} = ((\mathtt{InputClientID} - 1) \times 4) \\ + \mathtt{WiiRemoteLocalID}\end{aligned}$$

For example, a Wii Remote with a local ID 2 is connected to an input client with ID 3. The global ID of this Wii Remote will be:

$$((3 - 1) \times 4) + 2 = 10$$

The game server will assign a global ID 10 for this Wii Remote, and it will be used to control and update the game character. Therefore, it is important for the user to know on which input client his/her Wii Remote connect to. Otherwise, the player cannot identify correctly his/her global ID and which game character or cursor he or she is in control.

Every Wii Remote will have a unique global ID. This global ID can also be used to identify each player since every player can only use one Wii Remote. It is the duty of the game operator to tell the player his/her global ID after the player's Wii Remote is connected.

5.1.6 *Network Connection*

The input client and the game server use socket connection with UDP protocol for the network communication. There are several reasons why UDP is well suited for this system:

1. It is fast, lightweight, and has lower latency. This is needed since every input client will send the data 30 times a second to the game server.

2. UDP is a connectionless protocol, which means there is no need of establishing a connection before and after sending the data.

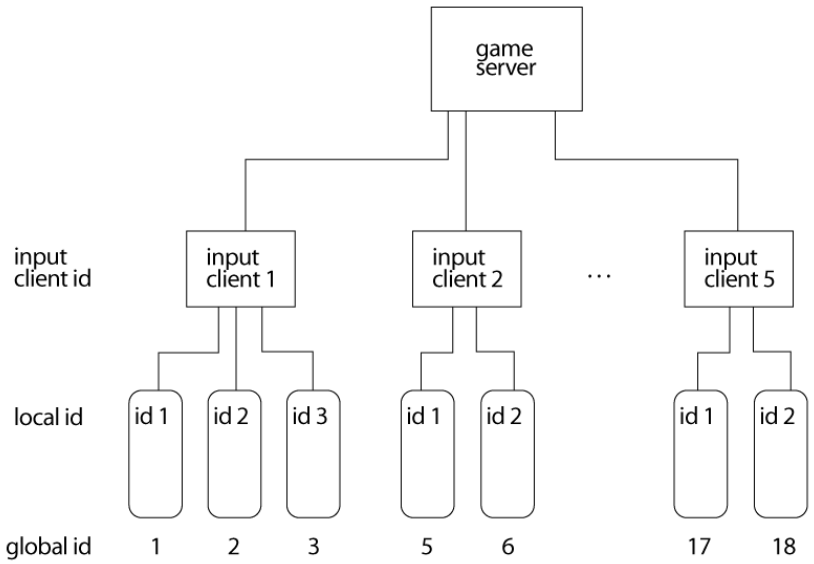


Figure 5.2: Wii Remote and Input Client Identification. The game server calculate each Wii Remote's golbal ID by using the Wii Remote local ID and the input client ID

The input client can immediately send the data to the game server without knowing whether the game server is available or not.

The input client creates an UDP connection and sends the data using the given game server's IP address and port. By default, the system use port number 5100.

Each network packet data is 46 bytes, containing binary data of the Wii Remote input data. It is designed to be small and compact so it can send the data 30 times per second without burdening the game server and network. Each packet contains only input data from one Wii Remote. If the input client has four connected Wii Remotes, it will send four packets of data; each contains the data from Wii Remote 1, 2, 3, and the Wii Remote 4. Table 5.3 shows the details of the packet data byte per byte.

5.2 GAME SERVER

5.2.1 *Hardware Specification*

The hardware specification of the game server is higher than the input client machine. Besides handling the input data sent by the input clients, the game server also handles the game engine and visualization. The game is visualized in 3D; therefore, the machine must have a

Table 5.3: Data packet sent by the input client to the Game Server

Byte	Data	Range Value
0	input client ID	1 to 255
1	Wii Remote local ID 1 to 4	1 to 4
2	Button A status	1 or 0
3	Button B status	1 or 0
4 - 11	Tilt Sensor axis X	64 bit floating point
12 - 19	Tilt Sensor axis Y	64 bit floating point
20 - 27	Tilt Sensor axis Z	64 bit floating point
28	IR point 1 availability	1 or 0
29 - 32	Coordinate X of IR1. Only valid if byte 28 is 1	32 bit signed integer
33 - 36	Coordinate Y of IR1. Only valid if byte 28 is 1	32 bit signed integer
37	IR point 2 availability	1 or 0
38 - 41	Coordinate X of IR2. Only valid if byte 37 is 1	32 bit signed integer
42 - 45	Coordinate Y of IR2. Only valid if byte 37 is 1	32 bit signed integer
46	D-pad button status. 1st bit: D-pad UP, 2nd bit: D-pad DOWN, 3rd bit: D-pad LEFT, 4th bit: D-pad RIGHT	0000b to 1111b

3D-enabled graphics card. For the prototype, sound feature is not available. Table 5.4 contains the hardware specification which was used for the development and testing.

Table 5.4: Minimum Hardware Specification for Game Server Machine

OS	Windows XP Home Edition
Processor	1.6Ghz single core processor
Memory	768Mb
Graphics Card	DirectX 9c-compatible graphics card with 3D acceleration, 64MB VRAM
Networking	LAN or Wireless LAN
Others	20 inch Display, a pair of candles or IR LED for Sensor Bar replacement

For the game screen, a 20 inch LCD monitor is used, which is good enough for prototype. The ideal scenario would be a large projected screen. The sensor bar is implemented using two pair of candles which are placed below the monitor. The candles are arranged to be 20 cm apart, simulating the IR source from the Nintendo's Sensor Bar.

5.2.2 *Software Specification*

The game server is developed using C# and .Net technology from Microsoft. The used IDE is Microsoft Visual Studio 2008 Express Edition.

The game server handles the 3D graphics and the game engine. An additional programming library for 3D visualization is used in order to take advantage of the 3D-acceleration hardware from the graphics card.

The game server mainly uses the graphic engine IrrlichtNETCP, a C# version of Irrlicht Engine (`http://irrlicht.sourceforge.net`). The library can be downloaded for free at their website at `http://irrlichtnetcp.sourceforge.net/`. The version 0.8 is used for the prototype implementation. Additional libraries for the game server development can be found on Table 5.5.

5.2.3 *Implemented Features*

These are the implemented features on the prototype version:

Table 5.5: Development Environment for Game Server Software

Programming Language	C#
IDE	Visual Studio 2008 Express Edition
Additional Libraries	.Net 2.0, IrrlichtNetCP v0.8, DirectX SDK August 2007

1. The game server visualizes/renders the game characters, game area, and game objects in 3D using its 3D engine.
2. The game server displays the player cursors when the current input mode is pointing.
3. The game server is able to open network connection to receive the packet data sent by the input clients. It processes the data and calculates the character speed based on the input data and the current game mode (single or group mode) and input mode (pointing, tilting, and gamepad).
4. The game server manages the game objects including the collision detection, removing or adding the game objects on the game area.
5. The game server manages the game phase: starting, resetting, and stopping the game.
6. The game server updates the game screen with rate 30 frame per seconds.
7. The game server record the speed and position of all active game characters during the game to a text file. This record data will be used to analyze the character movement and player performance (score) based on the input method.

5.2.4 *Game Objects*

The game is visualized in 3D. 3D environment is chosen because the game area and the game character size should be adjustable (smaller or bigger) depending on the size of audience. This is done by adjusting the position of 3D camera in the game environment. The further 3D camera from the game objects, the smaller the game objects will be rendered and the wider the game area will be.

These are the description for each game objects:

1. The game characters and the game objects are implemented as billboards. Billboard is a technique in computer graphics or 3D game programming for rendering a 2D image in 3D world. The image or sprite is rendered as a texture of a flat rectangular plane. The plane is adjusted and rotated so it always faces the camera. Billboard is chosen over a 3D model to reduce the number of polygons drastically so the system can render up to hundreds of objects in 3D world without a heavy performance drop.

2. The game character is drawn using a billboard with width and height of 256 unit (unit in 3D world). The billboard texture will be the character sprites/images. Each sprite is a PNG image with 128 x 128 pixel size. The sprites will be changed for every frame of the character animation.

3. Each game character has two animation sets: running and standing animation. Each animation set contains four sprites for each moving direction. Since there are eight moving directions, the total sprites are 32 for each animation set. The Figure 5.3 shows several sprites from the game character.

4. The maximum character speed is 10 unit per frame, which is 600 unit per second. If the Wii Remote is used as a pointing device, the character will run at the maximum speed when the distance between the character and the cursor is more than 400 point. If the Wii Remote's tilt feature is used, the player has to tilt the Wii Remote around 70 to 90 degrees to get the maximum speed.

5. On the gamepad mode, the character move using speed acceleration. The character will run at its maximum speed when the player holds the directional button for 1.5 seconds.

6. The game object is also using a billboard. For this game, there is only one type of object which only uses one still sprite (no animation).

7. The game area is a 3D rectangular plane with the game characters and the game objects on top of it. The size of the plane is 1600 x 1600. As shown in Figure 5.4, the position of the 3D camera is 1000 point away from the plane with 1000 point higher from the plane. Therefore, the system can render the whole game area/plane on the screen.

8. The game cursor is implemented using a 2D image. The cursor image size is 64 x 64 pixel size and the system draws it to the screen with different color for each player.

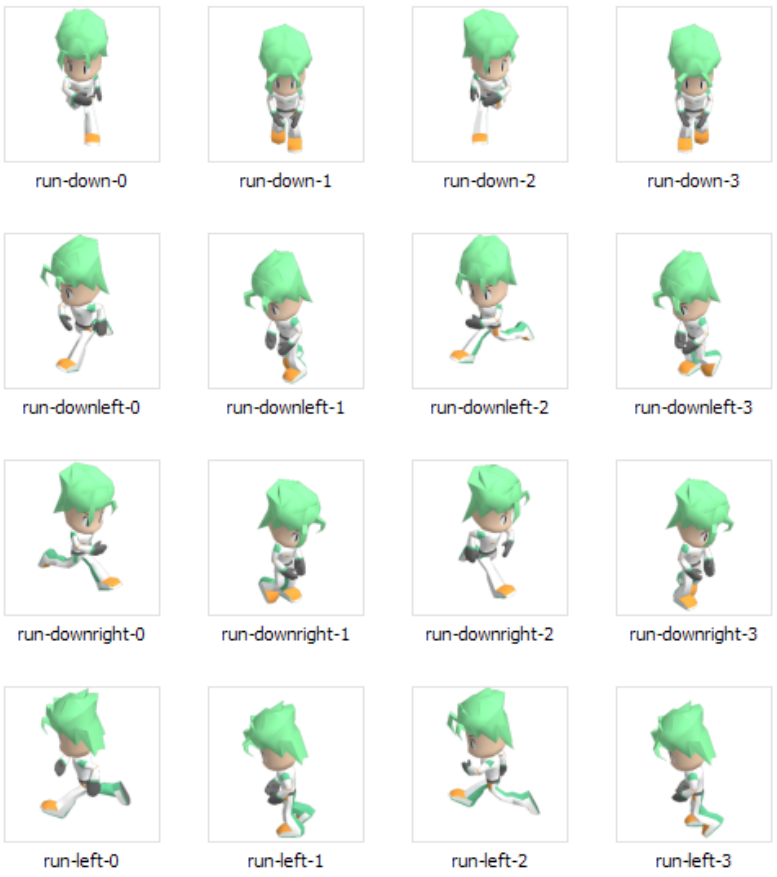


Figure 5.3: A screenshot of the game character's sprites

5.2.5 *Game Server Modules*

The game server contains several modules, such as modules for managing the game character, processing the input data, and displaying the game menu.

1. **Game Manager**, the main class (source code: `Game.cs`). It manages all other classes, creating a 3D device using functions from the Irrlicht 3D library, maintains the game loop/cycle at rate 60 frames/times per second, and changing the game status (start, stop, pause, and game over).

2. **Screen Manager**, managing the screen menu and layout (source code: `Screen.cs`). It displays the game menu, the game timer, and player scores.

3. **Input Manager**, managing the network communication with the input clients (source code: `InputManager.cs`). It handles a special thread that receives all the binary data from the input client

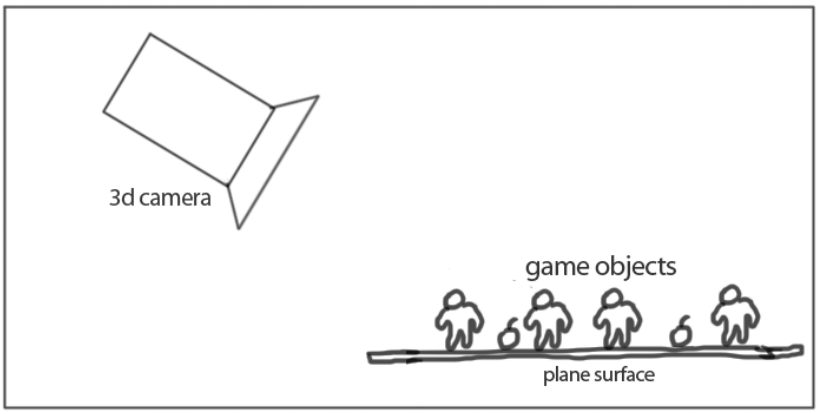


Figure 5.4: Game area (a 3D plane) and a 3D camera. The plane is 1600 x 1600 size and the camera is 1000 pixel away from it. The whole area of the game area will be rendered on the screen

via UDP protocol, translate the binary data to Wii Remote input data so they are available for other modules.

4. **Cursor Manager**, a module which calculates, manages, and displays the player cursor when the Wii Remote is being used as a pointing device. The source code files are `CursorManager.cs` and `WiiCursor.cs`.

5. **Camera Manager**, a module for the 3D camera (source code: `Camera.cs`). The module only handles a 3D camera and in this game, the camera is fixed and does not move.

6. **World Manager**, managing the game area (source code: `WorldManager.cs`). It creates the 3D plane for the area and provides the area information so the game characters and the game objects are always inside the game area.

7. **Animation Set**, loads and manages all the images file which are going to be used for the character animation sprites. The source code is `AnimationSet.cs`.

8. **Character Manager**. This module manages the game character (source code: `CharacterManager.cs`). It also translates the Wii Remote input data, which is provided by the module Input Manager, to character speed and movement direction based on the current input mode. It also animates and updates the character position on the game area.
   This module contains the implementation of character speed calculation algorithms which are described on previous chapter.

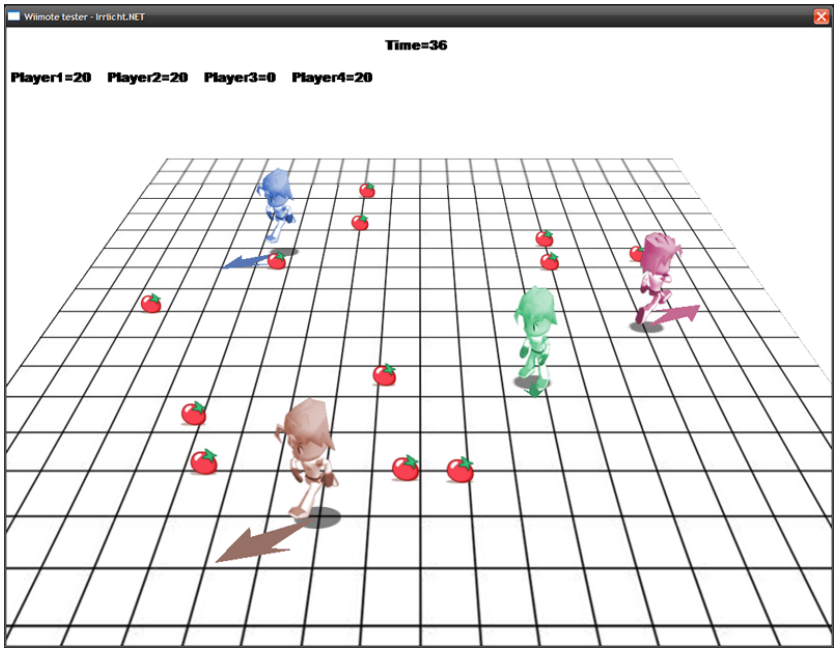


Figure 5.5: Game screenshot with four game characters, single mode, and using tilt input mode.

9. **Object Manager**, managing the game objects (source code: `Object-Manager.cs`). It adds/removes objects from the game area and detects collision between the game character and the game object.

10. **System Recorder**, an additional module which records the character speed and movement during the game (source code: `System-Recorder.cs`). The data is logged to a text file and will be use later for testing process.
    The system recorder will start recording once the game starts. It records the position and speed of all game characters every 0.5 second. Every time a character hit an item, the item position is also recorded.
    By analyzing the record data, the player scores and how well the player controlling the movement of game character using the Wii Remote will be known

6

TESTING AND EVALUATION

This chapter describes the testing procedure and the result of system prototype implementation described on Chapter 5. The test result is analyzed to check whether the system and the concept works properly and to explore the user behaviour/performance playing group gaming using Wii Remote.

6.1 TESTING GOAL

There are two aspects which will be tested and evaluated:

1. The system requirements: The ability of input client to make connection with Wii Remotes and send the input data to server, and the ability of the server to process the input data correctly.
2. Interaction testing: The users play the game using three interaction methods in two game modes (single and group mode). There are five questions to be answered for this testing:
    a) How well does the player control the character by pointing, tilting, or using the gamepad?
    b) How well does the player control the character in group mode?
    c) Which input method preferred by the players?
    d) Which input method gives the player his/her best performance?
    e) How do the team communicate or coordinate when playing using the group mode?

6.2 METHODOLOGY

This is how the test process works:

1. The input client is tested to make connection with one to four Wii Remotes.
2. When the connection is established, the Wii Remote is tilted and its button is pressed. The input client should display the input data and it will be compared with real status from the Wii Remote (which button is pressed or which direction it is tilted).
3. The game server is tested to connect with two or three input clients. Each input client will have two, three, or four connected Wii Remotes. The game server will display the processed data from the input client to the logging screen.
4. The participants are invited to play the game. Each participant brings his/her own Wii Remote which will be connected to several available input client machines.
5. Once the system is set (all Wii Remote is connected to the input clients, and all input clients are connected to the game server), the audience will be asked to do several input sequences (button pushing, tilting, or moving the cursor). This process is to validate the input data, and also to make the players getting used with the controller and how to control the game.
6. The player plays the game using the three input methods (tilt, point, and gamepad) and two game modes (single or group). For each combination of input and game mode, three game sessions will be played and tested.
7. The game server records the player score during the game. The player behavior during the game will be noted as well.

6.3 TESTING ENVIRONMENT

During the testing day, only six people and six Wii Remotes are available. Therefore, the testing procedure was modified in order to get the best result using this limited resource.

6.3.1 *Audience*

The audience consists of six people: five male and one female. The age range is from 22 to 29 years old. Four of them are students and the

others have full time job in IT and design industry. They play game on their Nintendo Wii one to three hours per week.

6.3.2 *Hardware Settings*

The testing process uses three laptops, which specification defined on Table 6.1. All three machines are using Windows XP Home Edition. The laptop Sony is the game server machine while two other laptops are the input client.

Table 6.1: Laptop Specification Used for Testing

Laptop Brand	Acer Aspire 2000	Acer Inspire 800	Sony Vaio VGN-SZ23
Processor	Intel Centrino 1.5Ghz	Intel Centrino 1.3Ghz	Intel Centrino Duo 1.66Ghz
Memory	768Mb	512Mb	1Gb
Bluetooth Hardware	Acer BT-600	Acer BT-700	built-in Blueooth
Bluetooth Driver	BlueSoleil v5.0.5	BlueSoleil v5.0.5	Built-in Bluetooth Driver

A 20 inch external LCD monitor is used for the game screen. For the Sensor Bar replacement, a pair of candle tea light is used. Both candles are put below the monitor with 10 cm apart. The audience sits in a line which is about 2 meter away from the screen. The game server machine is setup near the game screen while the other two machines are setup behind the audience. The Figure 6.1 shows the position of the game screen, audience, the machines, and the sensor bar during the testing.

6.3.3 *Input Clients Set Up*

Since there are only six Wii Remotes and three machines available for the testing, three combinations of hardware settings are set up to test the input client. Each combination uses different number of input client machines and the number of connected Wii Remotes on each input client machine.

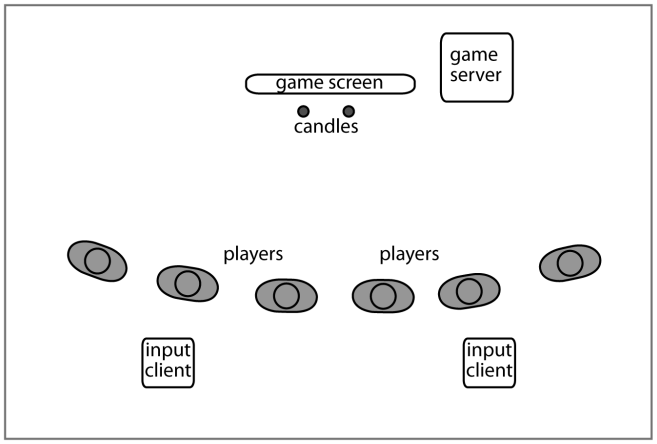


Figure 6.1: The Players, game screen, game server, and the input client machines (viewed from top)

These are the three combinations of the hardware setting used in the testing:

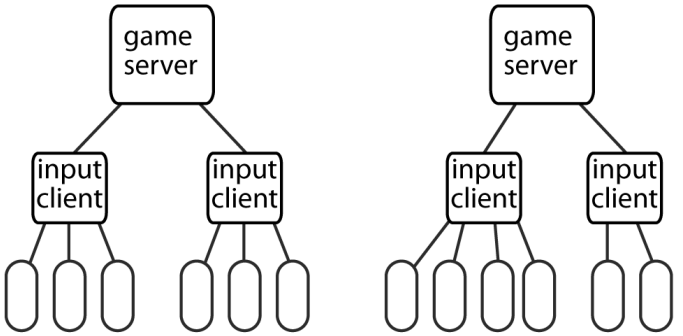
1. **Two input clients with three Wii Remotes**. Each input client machine is connected with three Wii Remotes.
2. **Two input clients with four Wii Remotes**. One of the input client machines is connected with four Wii Remotes while the other has only two Wii Remotes. This is to test the input client handling the maximum number of Wii Remote.
3. **Three input clients**. The game server machine also functions as an input client. Each machine, including the game server, will have two connected Wii Remotes. This is to test the game server with as many as possible connected input clients. It is also to test the input client and game server on one machine.

The Figure 6.2 illustrates the hardware combinations used for testing.

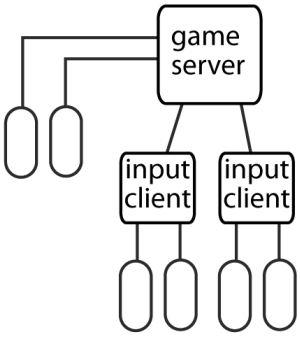
6.4 TESTING THE INPUT CLIENT

The input client is tested using three hardware settings combinations which have been explained before. The success criteria for this testing are:

1. The input client is able to make connection with The Wii Remotes
2. The input client is able to process and displays the Wii Remote data correctly.



(a) Hardware Combination 1 **(b)** Hardware Combination 2



(c) Hardware Combination 3

Figure 6.2: The hardware setting combination used for testing. (a) Two input clients with three Wii Remotes each, (b) Two input clients with four and two Wii Remotes each, and (c) Three input client, with game server and input client on the same machine.

6.4.1 *Pairing Several Wii Remotes*

The pairing process is done by using the Bluetooth software such as Bluesoleil Bluetooth or Built-in Sony Bluetooth software. However, there is a major problem when several Wii Remotes are trying to get paired to the input client.

When the button '1' and '2' are pressed together, the Wii Remote LEDs will blink for 20 seconds. While blinking, the Bluetooth software will be able to detect the Wii Remote and put it into a list of detected devices. From the list, the game operator has to choose which Wii Remote he/she wants to pair with the input client.

If several input clients are close enough (less than 10m), it is possible that a Wii Remote is detected and appears on two or more input client machines. This Wii Remote will only be paired to only one input client. However, sometimes the game operator accidentally connects the Wii

Remote to the wrong input client which in the end causing problem to the player for getting the wrong global ID.

The Figure 6.3 shows a possible case of two Wii Remotes connected to the wrong input clients.

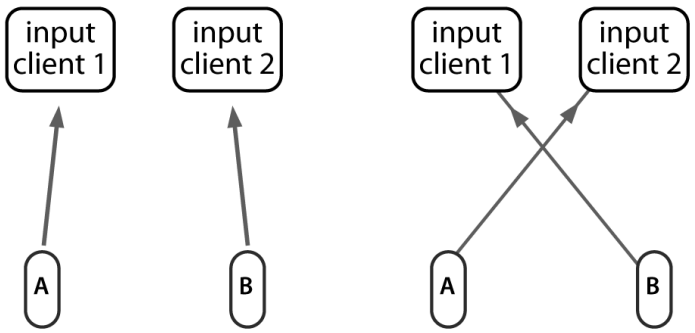


Figure 6.3: Two Wii Remotes connect to the wrong input clients.
(a) The Wii Remote A will be connected to input client 1 while the Wii Remote B will be connected to input client 2,
(b) Accidentally, the Wii Remote A is connected to input client 2. Both player and game operator might not realize this case.

For example, say there are two input clients with ID one and two. A player named Robert wants to connect to the input client 1 so he goes to the game operator at the input client 1. At the same time, another player named Julia is trying to connect to input client 2. She goes to the game operator at the input client 2. They both activate the Wii Remote pairing at the same time so both Robert's and Julia's Wii Remotes appear at both input client machines.

Accidentally, the game operator connects/pairs Robert's Wii Remote to input client 2 which is supposed to connect with the input client 1. The game operator thinks that it is Julia's Wii Remote which is connected to the input client 2. The game operator also connects Julia's Wii Remote to the input client one and thinks that it is Robert's Wii Remote.

This caused problem because the game operator gives them the wrong global ID. Robert is being told that his global ID is 1, which is false since his Wii Remote global ID is 4 and actually connected to input client 2 (not to the input client 1). Julia, who has been told her that global ID is 4, has her Wii Remote accidentally connected to input client 1 and now has global ID 1.

This problem can be solved if the game operator knows this fault connection and immediately exchange Julia's Wii Remote with Robert's. However, it is hard to tell which Wii Remote is connected to a specified input client since every Wii Remote has the same name when it is identified by the machine. The only differentiator is the Bluetooth address which is hard to remember.

The only solution which was done during the testing was connecting the Wii Remote one device at a time. It is not a big problem because there were only six Wii Remotes. However, this is not the case for a real large group gaming which could have up to 50 Wii Remotes. The pairing process of a big number of Wii Remotes will be a hard and time consuming process.

6.4.2 *Displaying the Wii Remote Data*

After all the Wii Remotes are connected to the input clients, the players are instructed to tilt and push the Wii Remote buttons at a certain way. The input client machine will display the raw data from the Wii Remote and it's compared with the player movement or Wii Remote status. This is done in order to test if input client process the Wii Remote data correctly.

These are the instructions which are given to the audience, including the expected values from the input client:

1. **Tilt to the left**. The player holds the Wii Remote at its base position (stands vertically with A button facing the player) and slowly tilts it to the left until it reach an angle around 90 degree. The X value from the motion sensor will slowly change from nearly 0 to nearly -1 and reach -1 when the angle is equal or bigger than 90 degree. The player then tilts the Wii Remote back to its base position, and the X value back to 0.

2. **Tilt to the right**. The criteria are the same as before, but the player tilts to right hand side. The motion sensor value X will slowly change from 0 to +1.

3. **Tilt forward**. The player holds the Wii Remote at its base position and slowly tilt it to the front (the same direction with where the player is facing to) until it reach an angle around 90 degree. The Z value from the motion sensor will slowly change from nearly 0 to nearly +1 and reach +1 when the angle is equal or bigger than 90 degree. The player then tilts the Wii Remote back to its base position, and the Z value is back to 0.

4. **Tilt backward**. The criteria are the same as tilting the Wii Remote forward, but the player tilts the Wii Remote backward. The motion sensor value Z will slowly change from 0 to -1.

5. **Push the button A**. The player press and hold the button A until he/she is being told to release it. Only the button A is tested since it is the only button used in the game.

6. **Points to the candles**. The user points the Wii Remote to the candles, and move around a little bit until the input client can detect both IR points. The input client will give a pair of coordinates if both IR points are detected. The cursor function is not tested on this phase because it is the game server which is able to process the IR data to cursor position. This will be tested later, on game server testing.

The result of this test proved that the three combinations of input clients are able to connect and process the Wii Remote. When the Wii Remote is tilted to the left or right, the displayed data shows that the motion sensor X has a value ranged between -1 when tilted to the left and +1 when tilted to the right. The motion sensor Z shows that it has a value ranged between -1 to +1 when the Wii Remote is tilted to the back and front.

There is a case that one player holds the Wii Remote at the wrong position so the X value is reversed (-1 for tilting to the right and +1 for the opposite direction). Another case is when the player tilts the Wii Remote too fast or with power so the X value is bigger than +1 or smaller than -1. This is because the Wii Remote is not only able to detect tilt but also the force which is generated by moving the Wii Remote with power (like swinging or shaking the Wii Remote).

The input client machines have also passed the button pushing test. All input clients in all combination are able to detect if the button A is being pressed or not. There are no noticeable delays between the button pressing and the input client shows the button status.

The input client is also able to show that the Wii Remote is able to detect two IR points. However, most of the players (five out of six players) having a little difficulty in finding the spot where the Wii Remote can detect two IR points.

6.5 TESTING THE GAME SERVER

For the first phase, the game server is tested using the same method as the input client. The players tilt the Wii Remote, push the button 'A', and point the Wii Remote to the candles. The game server is also

able to display the received input data from the input clients before it's processed into character speed and direction. The displayed result will be compared with the expected values. This is to check if the input data sent by the input clients has received correctly by the game server.

The game server has passed the first phase of the test. The game server displays the same value as the one displayed on the input clients. This is a proof that the input clients are able to send the data to the game server and the game server is able to receive it correctly. This test proves that it is possible to have the input client and the game server on one machine without sacrificing the game server performance.

The second phase is to test the cursor function. The player is instructed to move the cursor around the game screen. This is also to make them get used to controlling the cursor. They have to move the cursor from center to the top left corner of the screen, and move slowly to the top right corner, bottom right, bottom left, top left, and back to the center (Figure 6.4).

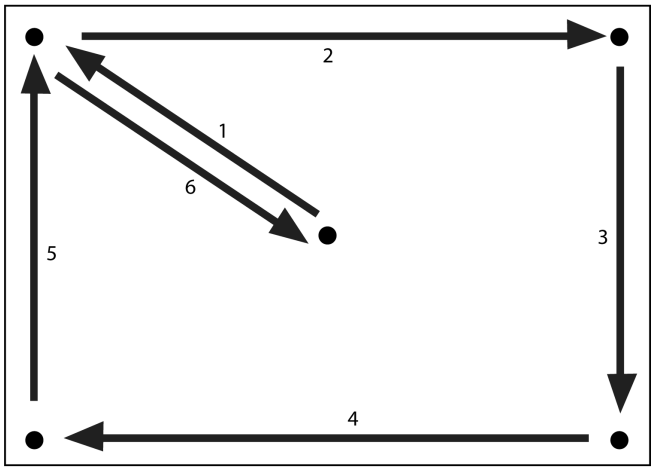


Figure 6.4: Cursor path during movement testing. The user has to point the and move the cursor to the specified path during movement test

Since there are only six players, the test is done by one player at a time. Before the test begin, each player is given chance to warm up by trying to move the cursor until he/she is ready.

All players are able to finish the task (able to move the cursor to all given points). However, based on the observation during the testing, the cursor movement is not very smooth. This caused difficulty for most players. All players agree that controlling the cursor is hard.

These are the common problems which make the cursor movement not smooth and hard to control:

1. The cursor movement feels jumpy. Sometimes the cursor suddenly jumps to another position.
2. Non-responsive cursor. The cursor stops moving/responding to the Wii Remote movement.
3. Too sensitive. The cursor moves too much by a very small movement. This will make the Wii Remote movement range much smaller and the cursor will harder to control.

These problems will be discussed on Section 6.7.

6.6 PLAY TEST

All players play the game together and only the hardware combination with two input clients (each with four and two Wii Remotes) is used to run the game. For each input method, the player plays three game sessions (40 seconds each) and three game modes: single mode, group mode with 2-player team, and group mode with 3-player team. Since there are three input modes, the audience plays 27 game sessions during the testing.

The game server records the game character movement and also the player score during the game. The player behavior is also noted. For every three game sessions, each player is asked these three questions:

1. Is it easy to control the game character using this input method?
2. How do you like this input method compare to previous input method?
3. Any additional comment about this input method?

These questions are for the general opinion of the user about the input method. In the end of the game (after all game sessions are finished), every player is asked again another questions:

1. Which input method do you like the best?
2. Do you find the game is too easy or too hard?
3. Any additional comments about the game?

These questions are asked in order to get the player's opinion about the game design and using Wii Remote for group gaming. On the following subsections, the test result is presented based on the input method.

6.6.1 *Input Method Pointing*

Table 6.2 shows the score of testing the game using the Wii Remote as a pointing device. Every time a player collects a game item, he/she will gain 10 point score. The game is played in single player mode, group mode with two players on each team, and group mode with three players on each team. For every game mode, three game sessions, with 40 seconds each, are played and every player's score is recorded.

Table 6.2: Players' scores using the pointing method

Player	Single Mode			2-Player Team Mode			3-Player Team Mode		
1	130	150	150	130	180	180	130	110	170
2	160	280	300						
3	130	180	190	150	190	200			
4	150	180	200				120	140	150
5	170	210	250	130	170	200			
6	140	150	180						

By average, the player score in this testing is the lowest one compare to the next two other input methods. All six players commented that it is really hard to control the character movement using the cursor and they don't like this input method.

The previous problems of jumpy, non-responsive, and too sensitive cursor make the character movement harder to control. Controlling the cursor movement itself is already a difficult thing to do, and not to mention that the player has to maintain the distance between the character and the cursor.

Two players feel confused in controlling the game character. They say that pointing cursor while holding the A button to make the character running is not an easy task for them. It feels unnatural to drag the character around.

Sometimes, the cursor suddenly disappears from the screen which makes the game character stop moving. This problem occurs when the cursor position is near the edge of the screen, which makes it even harder to move the game character around the edge or border area.

This is a problem regarding the range of Wii Remote IR sensor and also the algorithm of cursor calculation. If the player moves the Wii

Remote too wide, then the Wii Remote will lose one or both IR points. Since the cursor calculation requires both IR points, the new cursor position cannot be calculated.

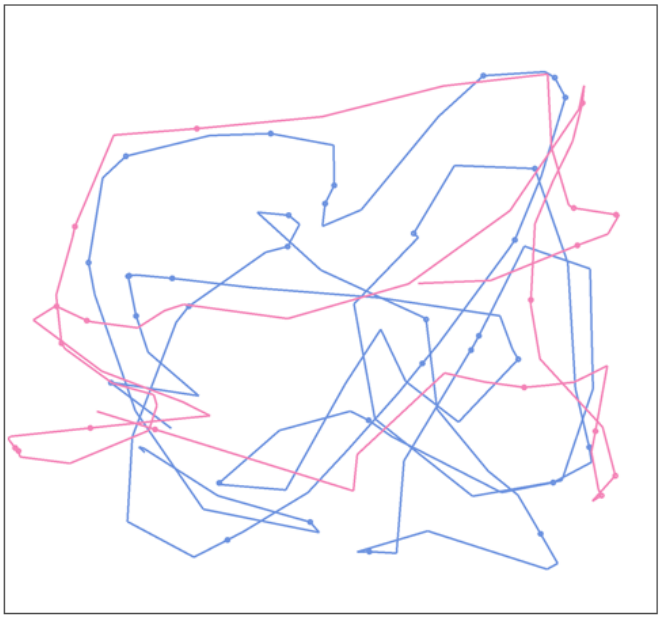


Figure 6.5: Two game characters movement controlled by using the pointing method. The number of characters in this Figure is reduced to two characters to make it easy to understand.

However, the player quickly realizes this problem and learns how to adjust. After the first game session using the pointing method, most players avoid moving their cursor near the edge of the screen. They only move the cursor around the center of the screen. From the recorded data, most of the game characters only move in the middle of the game area and abandon all game items near the border area.

The Figure 6.5 shows the movement of two game characters played using the pointing method. The game area is viewed from top and the number of tracks is reduced to two characters to make the figure easy to understand.

In the group mode, it is even harder to control the character. Based on the test result, the more players in a team, the slower or the more chaotic the character movement will be.

6.6.2 *Input Method Tilting*

The Table 6.3 shows the player's score using the tilting method for playing the game.

Table 6.3: Players' scores using the tilting method

Player	Single Mode			2-Player Team Mode			3-Player Team Mode		
1	250	300	320	380	420	400	350	370	320
2	450	400	450						
3	300	290	410	360	400	420			
4	280	300	320				330	300	340
5	380	370	460	300	370	360			
6	270	290	330						

By average, the player scores are much better than the input method pointing. The player is able to control the character movement fluently. Using this input method, the players start to enjoy the game and more cheerful than the previous input method (pointing).

All six players like this input method. They said that the control is more natural, easy to understand, and more precise. It is easy to make the character move faster, slower, or to stop the character.

From the recorded data, the character movement is more fluent and smooth. Unlike in the pointing method, the character is able to move to the corner or to the area near the edge area. See Figure 6.6 for the character movement diagram.

In the group mode, the player score are slightly lower than the single mode. The more players in a team/group, the lower the average score will be, although the test result does not show a big difference between 2-players group and the 3-players group.

For the group mode, the average score using this input method is the highest compare to the other two input methods. This is because the player is better in controlling speed and direction. All players also agree that this group mode is far more enjoyable than the group mode using the input method pointing or gamepad.

The player coordinates with the other team members by using verbal communication like shouting for a suggestion of the best direction. The character tends to move to the nearest game item.

Figure 6.6: Two game characters movement controlled using the tilting method. Unlike the previous method, the characters are able to move to area near the border/edge of the game area.

6.6.3 *Input Method Gamepad*

The Table 6.4 shows the score from playing the game by holding the Wii Remote horizontally and only pushing the buttons, as if it is a gamepad.

The average player score for single mode is the highest compare to the other two input method for the same mode. All players' best scores in single mode are from this input method. However, five out of six players still prefer the tilt method over using the gamepad. They feel uncomfortable holding the controller in the sideway position.

All of them agree that it is much easier to control the character using the gamepad method. Controlling the character movement is more precise and predictable. Unlike in pointing and tilting method, the character will move to the direction exactly to a specified or given direction by the player. This will make the player feel they have more control in character movements.

For example, pressing the left directional button will make the character move to the left. In tilt method, tilting the Wii Remote to the left sometimes make the character moves a little bit to the upper-left and not exactly to the left.

Unlike the single mode, the average score for the group mode using the gamepad is lower than the tilting.

Table 6.4: Players' scores using the gamepad/sideway method. No tilting or pointing, just pushing the Wii Remote's directional buttons

Player	Single Mode			2-Player Team Mode			3-Player Team Mode		
1	350	440	450	340	340	370	270	310	300
2	600	550	610						
3	440	470	490	270	330	380			
4	430	500	480				280	290	280
5	530	580	560	270	300	310			
6	370	430	470						

During the 2-players-team mode, it is quite often that the game character suddenly stops moving. This is because both players of the team is giving two opposite movement directions at the same time, thus the total sum of both speed vectors is zero.

This is very likely to happen in gamepad mode compared to tilt or cursor mode. In the tilt or cursor mode, the character can move to any direction and with any velocity between zero to the maximum speed. In gamepad mode, the character movement is limited to eight directions and the speed is only zero or the maximum speed.

In 3-players-team mode, this case never happens. The only way to get a zero-sum speed in 3-player team if and only if all three players are moving to three different directions with 120 degree angle between each direction. It is not possible since there are only eight moving directions with 45 degree difference each. Therefore, 120 degree difference for each player will not be achieved and the character will never get zero-sum speed vector.

6.7 ANALYZING THE PROBLEMS

In this section, the problem of Wii Remote cursor and connecting and pairing the Wii Remote to the input client will discussed. Then, the game design aspect will be analyzed for future improvement.

6.7.1 *Unsteady Cursor Movement*

Unsteady or jumpy cursor is a problem when the cursor makes sudden movement to another position. It is also a common problem on several Wii commercial games. Red Steel, one of the First Person Shoot-

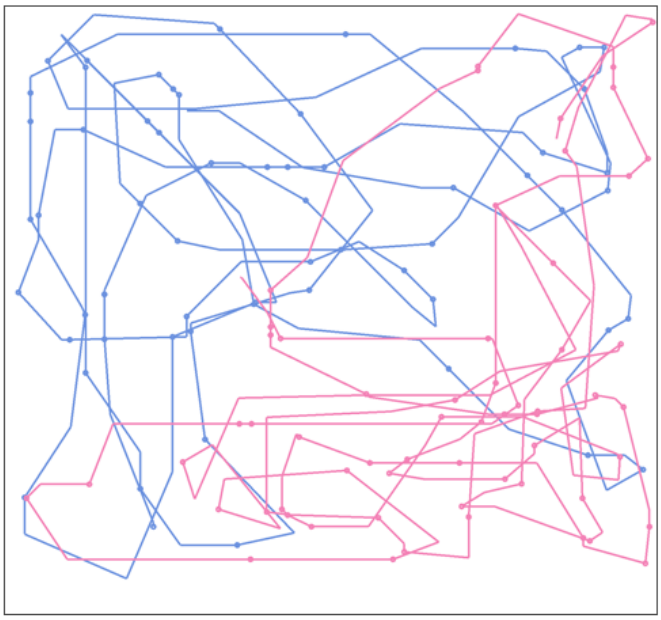


Figure 6.7: Two game characters movement controlled using the gamepad method. The number of tracks is reduced to two characters to make the figure easy to understand.

ing games for Nintendo Wii by Ubisoft, is controlled by pointing the Wii Remote. A video on Youtube (http://www.youtube.com/watch?v=T3lVFEuqStQ) shows a player playing the game Red Steel on his Nintendo Wii. The cursor has jumpy problem which makes the aiming harder.

From several game forums like GameSpot (http://www.gamespot.com), this jumpy cursor movement could occur if the Sensor Bar is placed too near or too far from the Wii Remote, or there are some other interfering IR sources like direct sunlight, Christmas lights, or incandescent light bulb. Even a reflection from a shiny marble table near the Sensor Bar could interfere.

In this project testing, there was a case that the Wii Remote ran out of battery during the game. When it's on low battery power, the Wii Remote will not be able to send a stabile input data.

These are two common reasons of the Wii Remote fail to detect two IR points:

1. The IR source is not strong or stabile enough. Once the game is already 30 minute, the candle length will be half and the light is not clearly visible to the Wii Remote. This can be fixed by replacing the candles with stronger and stabile IR sources like IR LEDs.

Figure 6.8: A screenshot of the game Red Steel, one of the first-person-shooter game for Nintendo Wii. Image is taken from http://www.redsteelgame.com/

2. The Wii Remote is too close to the IR sources so both IR points will be harder to detect. The Wii Remote also could easily lost one of its two detected IR points when moved with big range. Sometimes the player unconsciously stretched their arms too far while pointing with the Wii Remote. Even if the player sits 2 meter away from the candle, it will make the distance between the Wii Remote and the IR sources shorter.

A solution for a better cursor movement is to use a better algorithm. The current used algorithm requires two IR points to calculate the cursor position. This algorithm is not suitable for controlling the game character since it is not stabile and very sensitive, which make the character movement hard to control.

There are several algorithms still able to calculate the cursor position even if the Wii Remote has lost one of its two IR points. Some algorithms even use the values from motions sensor to predict the cursor movement. The cursor value needs to have a threshold to prevent it to be too sensitive to tiny movement like a small shake. GlovePIE is one of the good sources which provides several scripts for that purpose [Keno7].

6.7.2 *Game Design Aspect*

Two players were confused in controlling the game character using the pointing method (cursor). Both has a little experience in gaming before

and only play their Nintendo Wii about one hour per week. One way to solve this difficulty is to make the player to press the button only one time (instead of holding it) in order to move the character.

The player only has to push the button once to move the character to the specified point. The player prefers the sense of giving order to the game character to move to the pointed position rather than dragging the character using their Wii Remote. This is more natural since pointing the Wii Remote while holding a button on one hand will be too hard for inexperienced players.

That is also the reason that on some Wii Games which rely heavily on pointing (such as Red Steel) use an additional hardware, the Nintendo Nunchuk. The player holds the Nunchuk with the other hand. On this Nunchuk, there is an analog joystick which can be used to control the character movement.

Figure 6.9: Nintendo Nunchuk, an additional hardware for Wii Remote, is equipped with a analog joystick, one button, and also with motion sensing [Nino7]

The game feedback like score and game timer is also another problem. Most of the time, the player is unaware of the game timer and their own or other player's score. Sometimes they are surprised that the game is suddenly over and they just realized that they have won the game.

The players are too busy tracking and controlling the game character and targeting the game objects. In the pointing method, the players also have to track the cursor movement. Displaying the score on top area of the screen is too far from the player's focus, which are their own game character and the surrounding area near the character.

For a better user interface, the game needs an indicator which does not require the user to move their eye focus away from the game character. One solution for game timer feedback is to use non-visual feedback like beeping the sound or vibrating all Wii Remotes when the game will be finished in less than 10 seconds.

For the game score, one solution is to display it on top of the game character's head. To simplify the visualization, instead of score, only the player ranking is displayed on top of the character. The player ranking is based on the rank of the player score during the game play and it may change during the game play. By displaying the ranking, the player will be more motivated to defend or to make his/her ranking higher.

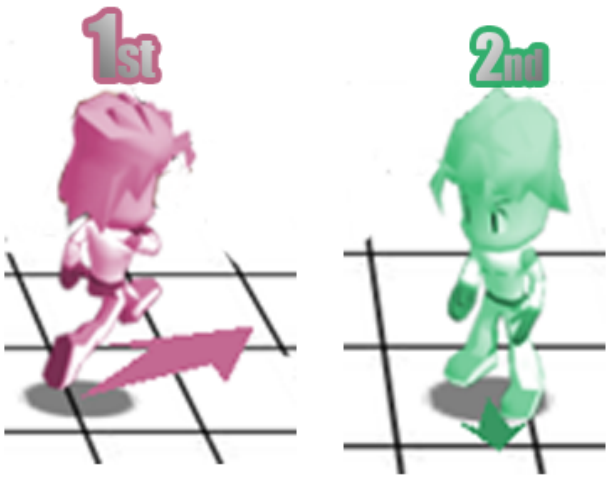


Figure 6.10: Game character with the ranking displayed on top of the character's head

6.8 CONCLUSION

These are the conclusions based on comparing the average scores of the players, the player's answers of the given questions, and the noted behavior during the testing.

The Table 6.5 and the chart on Figure 6.11 show the average scores of all players based on the game mode (single/group mode) and the three input methods. These are the conclusion based on the results:

1. For the single mode, the input method gamepad gives the highest average player score, followed by tilting and pointing. How-

Table 6.5: Average player scores from the experiment

	Single Player	2-Player Team	3-Player Team
Pointing	183.33	170.00	136.67
Tilting	342.78	378.89	335.00
Gamepad	486.11	323.33	288.33

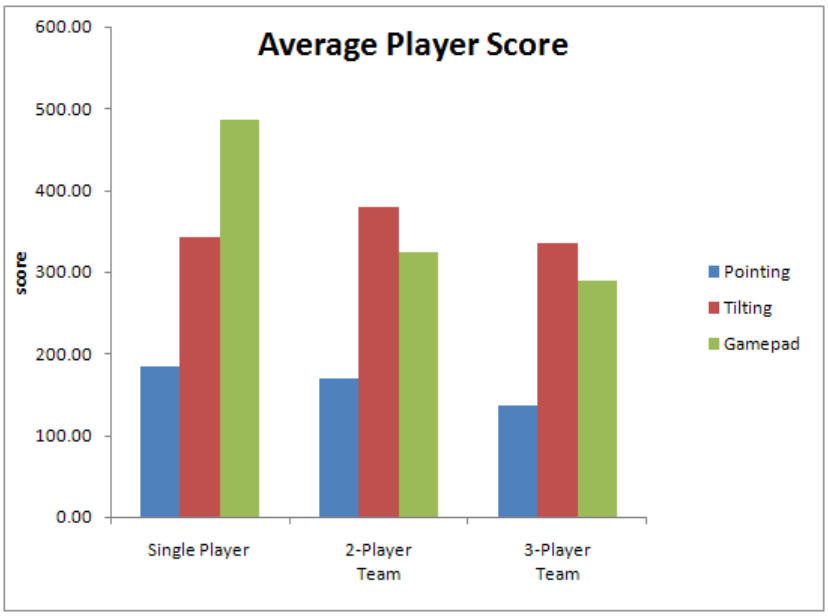


Figure 6.11: Player's average score. The number of players is six and three game sessions are played for each game mode.

ever, during the group mode (with two or three players on each team), the input method tilting gives the highest score compare to gamepad. The pointing method gives the worst score for both single and group modes.

2. Based on the player's opinion, five out of six players prefer the tilting over the input method gamepad.

3. All six players don't like the pointing method.

4. Except for the tilt mode, the average score for the group mode is always lower than the single player mode. For the gamepad mode, the players score even drops drastically from 468.11 in single mode to 323.33 in group mode. This is because in the gamepad mode, the character is likely to stop moving because of zero-sum vector speed (as explained on subsection 6.6.3) which

makes the character suddenly stops and the movement becomes less smooth.

5. In the group mode, the more player in a team, the lower the score will be. This is because the character movement is determined by several players. Each player can only control maximum half or a third of the character speed, depends on the number of player in a team. In order to the game character to move at a maximum speed, all players in a team must tilt the Wii Remote or move the cursor to the same direction.

6. Despite of the lower score, all six players enjoy the group mode. They said that it is more fun to play in a team, especially when they are allowed to shout.

7

CONCLUSION

This final chapter describes the conclusion and several improvements for the future development of large group gaming utilizing Wii Remote.

7.1 EXPERIMENT RESULTS

1. Based on the experiment result, the game server is able to connect up to three input clients, with each input client has up to four connected Wii Remotes. Theoretically, the prototype is able to handle up to twelve connected Wii Remotes.
2. The large group gaming can be played using the three interaction methods which can be done by Wii Remote: pointing, tilting, and using the Wii Remote as if as it is a gamepad (only pushing the buttons).
3. For the pointing method, it is important to have:
   a) a good algorithm to calculate the cursor position,
   b) a stabile and strong IR sources, and
   c) a good IR source placement so it can be reached by the Wii Remotes from a wider range.
4. The gamepad-style method gives the player more precise movement control. This is because most gamers are already familiar with the gamepad input scheme (controlling the movement by pressing the directional pad/button). Holding the Wii Remote in sideways/gamepad mode for long time is not so convenient for most players.

5. Although the gamepad mode gives the highest player score, most players prefer the tilting method. The tilting method gives more smooth and fluent movement control. It is easy to use and the user feels more natural tilting the Wii Remote to control the game character.

7.2 FUTURE IMPROVEMENTS

These are several points for the improvement of this project or further study regarding large group gaming using Wii Remote.

1. Using Wii Remote for the large group gaming is an expensive solution for group gaming. It needs more resources since one input client machine is needed for every four Wii Remotes. For example, five PCs are needed to support 20 users; therefore 25 PCs would be needed for 100 users. This problem might be solved using the advance Bluetooth technology which can connect up to 255 devices (compared to the current Bluetooth technology which is limited to maximum seven devices).

2. A better system or mechanism is needed to improve the pairing process between the Wii Remote and the PC (input client machine). On the prototype, the pairing process is a hard and time consuming process. The Wii Remotes have to be paired one by one. Pairing several Wii Remotes simultaneously, could cause the problem of connecting the Wii Remote to the wrong input client.

3. The new system should be able to manage Wii Remotes using its Bluetooth address. It should be a lower level application which can access the operating system's HID interface and the Bluetooth driver, so it will be able to pair the Wii Remote without using external Bluetooth software.

4. The system needs to be tested with a bigger audience (more than 60 players). The current prototype is only able to handle up to 16 game characters (up to 48 players with 3-player-team mode). Therefore, several modifications on the game server need to be done on the prototype, especially on the modules related to the game characters.

5. By supporting a larger number of audience, there would be four testing possibilities:
    a) The machine performance issue, whether one game server is enough to handle all data sent by more than 30 input clients.

b) Bluetooth interference, whether it is going to be a frequency or interference problem if there are 50 Wii Remote sending or connecting at the same time via Bluetooth.

c) The group mode, to test how many maximum player should be in one group.

d) The game visualization, how small the game character or object should be if there will be 50 game characters at the screen at the same.

A

APPENDIX: IMPLEMENTATION AND SOURCE CODES

This will contain some parts of the source code of the implementation.

A.1 PAIRING A WIIMOTE WITH COMPUTER

Before the programming library can connect and access and the Wii Remote input data, the Wii Remote needs to get paired with the PC. This has to be done using the Microsoft Windows Bluetooth software or external software.

A.1.1 *Microsoft Windows Bluetooth Software*

This is step-by-step instructions how to pair/connect a Wii Remote to a PC using the Microsot Window's Bluetooth software.

1. Make sure the PC's Bluetooth hardware is on.
2. Go to **Control Panel** » **Bluetooth Devices** and click "Add" a new Bluetooth Device.
3. Hold down the button '1' and '2' on the Wii remote. The LEDs will start blinking and the Wii Remote is ready to be paired.
4. Check the box next to "My device is set up and ready to be found", then "Next". The software will search for the available Bluetooth Devices and put them into the list.
5. The Wii Remote will be identified by a device name **Nintendo RVL-CNT-01**. Click on it and then click the button "Next".
6. Select "Don't use a passkey", then click "Next". You may need to hold down the Wii Remote's button 1 and 2 again here.

The Microsoft Windows will install and add the Wii Remote to the connected devices. For connecting another Wii Remote, repeat these steps on it.

To test if the Wii Remote is connected, use the small test application included with the Brian Peek's Wii Remote library. Go to the Section A.3.1 below for more details.

Depends on the Bluetooth hardrware and driver, pairing the Wii Remote using the Microsoft Bluetooth software may fail. If it fails, please use other Bluetooth hardware or software. Please check the compatible Bluetooth hardware and software on the list from `http://www.wiili.org/index.php/Compatible_Bluetooth_Devices` or `http://wiibrew.org/wiki/List_of_Working_Bluetooth_Devices`.

A.1.2 *Bluesoleil Bluetooth software*

This is step-by-step instructions how to pair/connect a Wii Remote to a PC using the Bluesoleil Bluetooth software, which is used in this project.

1. Make sure the PC's Bluetooth hardware and the Bluesoleil software are on.
2. Hold down the button '1' and '2' on the Wii remote. The LEDs will start blinking and the Wii Remote is ready to be paired.
3. While the lights are blinking on the WiiMote click on the orange sphere in the BlueSoleil program. It will search for the available Bluetooth Devices and will display them around the orange sphere on the program's display.
4. The Wii Remote will be identified by a device name **Nintendo RVL-CNT-01**. Right click it and select **Connect** » textbfBluetooth Human Interface Device Service (Figure A.1).
5. The program will automatically install drivers for the WiiMote. Eventually you will be asked to either Continue or Stop the installation because this has not passed Windows Logo testing. Click Continue Anyway.
6. A successful pairing will have a green line attached from you WiiMote to the orange sphere (Figure A.2)

To test if the Wii Remote is connected, use the small test application included with the Brian Peek's Wii Remote library. Go to the Section A.3.1 below for more details.

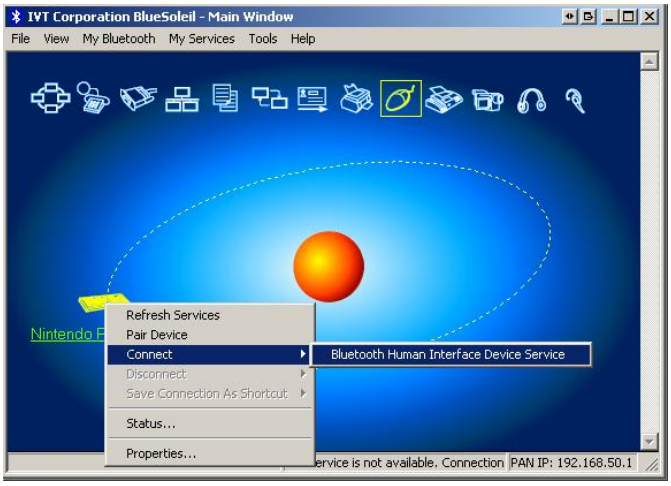


Figure A.1: Pairing a Wii Remote using Bluesoleil

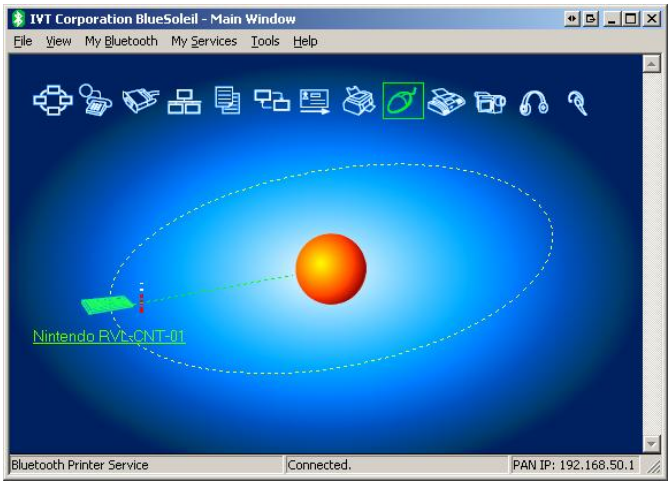


Figure A.2: A successful Wii Remote pairing using Bluesoleil

A.2 COMPILING THE PROJECT SOURCE CODES

A.2.1 *Downloading the Files*

The source code of this project can be downloaded from urlhttp://www.abiamy.com/abiyasa/thesis/. There you can see two files: **`WiiGroup-InputClient.zip`** and **`WiiGroup-GameServer.zip`**. The `WiiGroup-InputClient.zip` contains the binary and the source code of the input client and the other file is for the game server.

A.2.2 *Additional Libraries*

Before you can start running the Game Server and the Input Client, you have to install these additional runtime libraries:

1. Microsoft DirectX 9.0c Runtime.
2. Microsoft .Net 2.0.

Both can be downloaded from the Microsoft website. If you want to re-compile the source code, these libraries are required:

1. DirectX SDK. This project uses DirectX SDK version August 2007 which can be downloaded at Microsoft DirectX SDK webpage[1].
2. IrrlichtnetCP. A 3D graphics for .Net. This project uses the version 0.8 which can be downloaded at `http://sourceforge.net/project/showfiles.php?group_id=176021&package_id=203751`
3. Microsoft Visual Studio 2008 Express Edition C#. This IDE is needed to open the project files which are included in this project source code files.

A.3 BRIAN PEEK WIIMOTE LIBRARY

You can download the latest version of this library at Brian Peek personal website (`http://www.brianpeek.com`) or directly at his Wii Remote project page (`http://www.brianpeek.com/blog/pages/wiimotelib.aspx` or `http://www.wiimotelib.org/`).

A.3.1 *Testing the Connected Wii Remotes*

If you download the binary version of the library (not the source code), you will get a small test application called WiimoteTest (on WiimoteLib v.1.5.2). Execute that file to test the connected Wii Remote. The application will connect, get, and display the data from the Wii Remote. It shows all functionalities which are supported by this library.

A.3.2 *Connecting Wii Remotes*

Before connecting, we should call the `FindAllWiimotes()`. This method will find all the paired WIi Remotes and put them into a list.

1 Microsoft DirectX SDK webpage: `http://www.microsoft.com/downloads/details.aspx?FamilyID=529f03be-1339-48c4-bd5a-8506e5acf571`

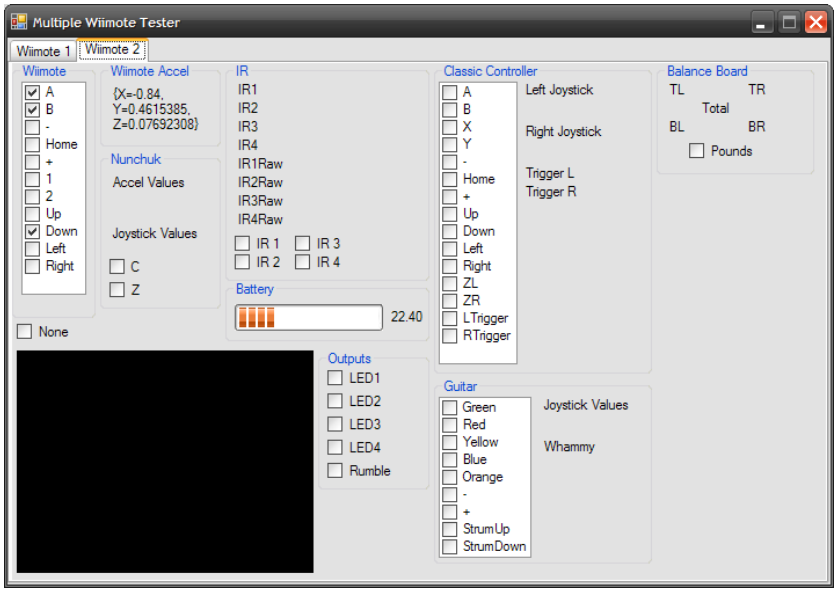


Figure A.3: Brian Peek's Wiimote Tester application

Listing A.1: Listing the paired Wii Remote using Brian Peek Library

```
WiimoteCollection wiimoteCollection = new
    WiimoteCollection();

try
{
    wiimoteCollection.FindAllWiimotes();
}
catch(WiimoteNotFoundException ex)
{
    // Handle error
    MessageBox.Show(ex.Message, "Wiimote not found error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
catch(Exception ex)
{
    // Handle error
    MessageBox.Show(ex.Message, "Unknown error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

The paired Wii Remotes are now stored on the list `wiimoteCollection`. Now we have to connect each Wii Remote from the wiimoteCollection. The Wii Remote object can be accessed from the wiimoteCollection by using `wiimoteCollection[x]`, where x is the index of the Wii Remote in the list.

This code below is to make connection with all Wii Remote objects on the wiimoteCollection.

Listing A.2: connecting Wii Remote using Brian Peek Library

```
bool isWiimoteConnected = false;
for (int i = 0; i < wiimoteCollection.Count; i++)
{
    try
    {
        wiimoteCollection[i].Connect();
        isWiimoteConnected = true;
        numOfConnectedWiimotes++;
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message, "Error in connecting Wiimote–"
            +
            i.ToString(), MessageBoxButtons.OK, MessageBoxIcon.
                Error);

        isWiimoteConnected = false;
    }

        // turn on the Wii Remote's LED
    if (isWiimoteConnected)
    {
        switch (i)
        {
            case 0:
                wiimoteCollection[i].SetLEDs(true, false, false,
                    false);
                break;

            case 1:
                wiimoteCollection[i].SetLEDs(false, true, false,
                    false);
                break;

            case 2:
                wiimoteCollection[i].SetLEDs(false, false, true,
                    false);
                break;

            case 3:
                wiimoteCollection[i].SetLEDs(false, false, false,
                    true);
                break;
```

```
        }

        // Set the kind of input data from Wii Remote
        wiimoteCollection[i].SetReportType(InputReport.IRAccel,
            true);
    }
}
```

A.3.3 *Reading Wii Remote Data*

From the code A.2, we know how to set turn on/off the Wii Remote LED by using `wiimoteCollection[i].SetLEDs()`.

The code below shows how to get the Wii Remote input data.

Listing A.3: Getting the Wii Remote data using Brian Peek Library. The data is processed and sent to the game server

```
// Structure returned by the Wii Remote
WiimoteState theWiimoteState;

// The Wii Remote Button status
WiimoteLib.ButtonState theButtonStatus;

// The Wii Remote motion sensor values
WiimoteLib.AccelState theMotionStatus;

// The Wii Remote IR status and points
WiimoteLib.IRState theInfraredStatus;

// Prepare the packet data which will be sent to the game server
byte[] sendBytes = new byte[64];

// the first byte is for the input client ID
sendBytes[0] = clientCode;

bool buttonA, buttonB;
bool dpadUp, dpadDown, dpadLeft, dpadRight;
byte dpadStatusByte;
double motionX, motionY, motionZ;
bool ir1enabled, ir2enabled;
int ir1x = 0, ir1y = 0, ir2x = 0, ir2y = 0;
StringBuilder buttonStatus = new StringBuilder(64);

// read the wiimote data
for (int i = 0; i < numOfConnectedWiimotes; i++ )
{
    theWiimoteState = wiimoteCollection[i].WiimoteState;
```

```
// get the button status
theButtonStatus = theWiimoteState.ButtonState;
buttonA = theButtonStatus.A;
buttonB = theButtonStatus.B;
sendBytes[1] = (byte)i;
sendBytes[2] = (byte)(buttonA ? 1 : 0);
sendBytes[3] = (byte)(buttonB ? 1 : 0);

// get the motion status
theMotionStatus = theWiimoteState.AccelState;
motionX = theMotionStatus.Values.X;
motionY = theMotionStatus.Values.Y;
motionZ = theMotionStatus.Values.Z;

// convert them into bytes
byte[] xdata = BitConverter.GetBytes(motionX);
byte[] ydata = BitConverter.GetBytes(motionY);
byte[] zdata = BitConverter.GetBytes(motionZ);
for (int j = 0; j < 8; j++)
{
    sendBytes[4 + j] = xdata[j];
    sendBytes[12 + j] = ydata[j];
    sendBytes[20 + j] = zdata[j];
}

// Get the first IR point
theInfraredStatus = theWiimoteState.IRState;
ir1enabled = theInfraredStatus.IRSensors[0].Found;
if (ir1enabled)
{
    sendBytes[28] = (byte)1;

    ir1x = theInfraredStatus.IRSensors[0].RawPosition.X;
    ir1y = theInfraredStatus.IRSensors[0].RawPosition.Y;

    byte[] irxdata = BitConverter.GetBytes(ir1x);
    byte[] irydata = BitConverter.GetBytes(ir1y);
    for (int j = 0; j < 4; j++)
    {
        sendBytes[29 + j] = irxdata[j];
        sendBytes[33 + j] = irydata[j];
    }
}
else  // no IR 1 data is available
{
    sendBytes[28] = (byte)0;
```

```
}

// Get the second IR point
ir2enabled = theInfraredStatus.IRSensors[1].Found;
if (ir2enabled)
{
    sendBytes[37] = (byte)1;

    ir2x = theInfraredStatus.IRSensors[1].RawPosition.X;
    ir2y = theInfraredStatus.IRSensors[1].RawPosition.Y;

    byte[] irxdata = BitConverter.GetBytes(ir2x);
    byte[] irydata = BitConverter.GetBytes(ir2y);
    for (int j = 0; j < 4; j++)
    {
        sendBytes[38 + j] = irxdata[j];
        sendBytes[42 + j] = irydata[j];
    }
}
else  // no IR2 data
{
    sendBytes[37] = (byte)0;
}

// get the D-pad buttons status
dpadStatusByte = (byte)0;
dpadUp = theButtonStatus.Up;
dpadDown = theButtonStatus.Down;
dpadLeft = theButtonStatus.Left;
dpadRight = theButtonStatus.Right;
if (theButtonStatus.Up)
{
    dpadStatusByte |= (byte)0x01;
}
if (theButtonStatus.Down)
{
    dpadStatusByte |= (byte)0x02;
}
if (theButtonStatus.Left)
{
    dpadStatusByte |= (byte)0x04;
}
if (theButtonStatus.Right)
{
    dpadStatusByte |= (byte)0x10;
}
sendBytes[46] = dpadStatusByte;
```

```
// Show the wiimote status and data to the form
buttonStatus.Length = 0;
if (buttonA)
{
    buttonStatus.Append("[A]");
}
if (buttonB)
{
    buttonStatus.Append("[B]");
}
if (dpadUp)
{
    buttonStatus.Append("[Up]");
}
if (dpadDown)
{
    buttonStatus.Append("[Down]");
}
if (dpadLeft)
{
    buttonStatus.Append("[Left]");
}
if (dpadRight)
{
    buttonStatus.Append("[Right]");
}
string motionStatus = "X:" + motionX.ToString("F") +
    " Y:" + motionY.ToString("F") + " Z:" + motionZ.ToString("F
        ");
string infraredStatus = "";
if (ir1enabled)
{
    infraredStatus += "IR1:" + ir1x.ToString() + "," + ir1y.
        ToString() + " ";
}
if (ir2enabled)
{
    infraredStatus += " IR2:" + ir2x.ToString() + "," + ir2y.
        ToString() + " ";
}
DisplayWiimoteData(i + 1, buttonStatus.ToString(), motionStatus
    , infraredStatus);

// send it to the server (if enable)
if (isSendToServer)
{
```

```
        // Send the data to server
        networkClient.Send(sendBytes, sendBytes.Length, sendPoint);
    }
}
```

A.3.4 *Disconnecting Wii Remotes*

Finally, we need to disconnect the Wii Remotes from the application.

Listing A.4: Disconnecting the Wii Remotes using Brian Peek Library

```
for (int i = 0; i < wiimoteCollection.Count; i++)
{
    // turn off LED
    wiimoteCollection[i].SetLEDs(false, false, false, false);
    wiimoteCollection[i].SetReportType(InputReport.Buttons, false);

    wiimoteCollection[i].Disconnect();
}

wiimoteCollection = null;
```

For more details or complete documentation of Brian Peek's library, please check the Brian Peek WiimoteLib page (http://www.brianpeek.com/blog/pages/wiimotelib.aspx or http://www.wiimotelib.org/).

BIBLIOGRAPHY

[Ada06] Ernest Adams. The designer's notebook: Ps3 versus Wii - the designer's perspective. http://www.gamasutra.com/features/20061222/adams_01.shtml, December 2006. (Cited on page 14.)

[Bel07] Justin Ryan Belcher. Embodied interfaces for interactive percussion instruction. Master's thesis, Virginia Polytechnic Institute and State University, May 2007. (Cited on page 15.)

[Bha01] Pravin Bhagwat. Bluetooth: Technology for short-range wireless apps. *IEEE Internet Computing*, 5(3):96–103, 2001. (Cited on page 21.)

[BTL+07] Bernd Bruegge, Christoph Teschner, Peter Lachenmaier, Eva Fenzl, Dominik Schmidt, and Simon Bierbaum. Pinocchio: conducting a virtual symphony orchestra. In *ACE '07: Proceedings of the international conference on Advances in computer entertainment technology*, pages 294–295, New York, NY, USA, 2007. ACM Press. (Cited on pages 15 and 17.)

[Car94] L. Carpenter. Cinematrix, video imaging method and apparatus for audience participation. us patents #5210604 (1993) and #5365266 (1994), 1994. (Cited on pages 1 and 5.)

[Cin01] Cinematrix interactive entertainment system. http://www.cinematrix.com, 2001. (Cited on pages 1, 5, 6, 7, and 44.)

[Eng07] Engadget. Wiimote used to navigate immersive 3d environments. http://www.engadget.com/2007/04/05/wiimote-used-to-navigate-immersive-3d-environments/, 2007. (Cited on page 15.)

[Ent06] Sony Computer Entertainment. Sixaxis wireless controller. playstation.com. http://www.us.playstation.com/PS3/Accessories/SCPH-98040, 2006. (Cited on page 15.)

[Fel02] Mark Feldmeier. Large group musical interaction using disposable wireless motion sensors. Master's thesis, Cambridge, MA : MIT Media Laboratory, 2002. (Cited on pages 10 and 16.)

[FP04] Mark Feldmeier and Joseph A. Paradiso. Giveaway wireless sensors for large-group interaction. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1291–1292, New York, NY, USA, 2004. ACM Press. (Cited on pages 1, 9, and 10.)

[Hac07] Hack a Wii. http://hackawii.com/, 2007. (Cited on page 27.)

[HL08] Sebastian Heise and Joern Loviscach. A versatile expressive percussion instrument with game technology. pages 393–396. IEEE ICME 2008, 2008. (Cited on pages 15 and 17.)

[Iva07] Tony Ivan. The "buzz" around social gaming. next generation - interactive entertainment today. http://www.next-gen.biz/index.php?option=com_content\&task=view\&id=6568\&Itemid=2, July 2007. (Cited on page 14.)

[Ken07] Carl Kenner. Glovepie. http://carl.kenner.googlepages.com/glovepie, 2007. (Cited on pages 26 and 77.)

[Kha07] Yekaterina Kharitonova. Cra dmp student in computing research. http://www.cs.cmu.edu/~ykk/week2.html, 2007. (Cited on page 22.)

[Lee07] Johnny Chung Lee. Wiimote projects. http://www.cs.cmu.edu/~johnny/projects/wii/, 2007. (Cited on page 19.)

[LKGM08] Hyun-Jean Lee, Hyungsin Kim, Gaurav Gupta, and Ali Mazalek. Wiiarts: creating collaborative art experience with Wiiremote interaction. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 33–36, New York, NY, USA, 2008. ACM. (Cited on pages 17 and 18.)

[MAPS02] Dan Maynes-Aminzade, Randy Pausch, and Steve Seitz. Techniques for interactive audience participation. In *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications*, pages 257–257, New York, NY, USA, 2002. ACM Press. (Cited on pages 1, 7, 8, 9, and 44.)

[Nin07] Nintendo. Wii:inside the system. http://www.nintendo.com/systemswii, 2007. (Cited on pages 11, 25, and 78.)

[Pee07] Brian Peek. Managed library for Nintendo's Wiimote. http://www.brianpeek.com/, March 2007. (Cited on page 23.)

[Rou01] Richard Rouse. *Game Design: Theory and Practice*. Wordware Publishing, 2001. (Cited on page 44.)

[SGR07] Akihiko Shirai, Erik Geslin, and Simon Richir. Wiimedia: motion analysis methods and applications using a consumer video game controller. In *Sandbox '07: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, pages 133–140, New York, NY, USA, 2007. ACM Press. (Cited on pages 27 and 28.)

[SK08] Hiroshi Suzuki and Masaki Kondo. Nintendo bigger on Wii sales. http://www.financialpost.com/story.html?id=690214, July 2008. (Cited on page 2.)

[Sof07] Softpedia. Dj Wiimote. http://news.softpedia.com/news/DJ-Wiimote-45598.shtml, 2007. (Cited on page 15.)

[Wii06] WiiSpot. The Wiimote: Nitty gritty. http://wiispot.com/the-wiimote-nitty-gritty/137/, June 2006. (Cited on page 26.)

[Wii07a] WiiLi. Wiimote. http://www.wiili.org/index.php/Wiimote, 2007. (Cited on pages 11 and 24.)

[Wii07b] WiiLi. Wiimote drivers. http://www.wiili.org/index.php/Wiimote_driver, 2007. (Cited on page 23.)

[Wii08] Wiire.org. Wii reverse engineered. http://www.wiire.org/Chips/ADXL330, 2008. (Cited on page 24.)

[Wik07] Wikipedia. Wii remote. http://en.wikipedia.org/wiki/Wii_Remote, August 2007. (Cited on pages 25 and 27.)