Interfacing the WII remote with a Computer to capture Real-Time activity.

Abhishek Venkatesh
CS 6235
Real Time Systems
venky@cc

Girish Saini
CS 7470
Mobile And Ubiquitous Computing
gsaini@cc

ABSTRACT

Game companies are coming up with radical new ways for users to interact with games. An example is the Nintendo Wii. The Wii allows users to control game-play via the Wii Remote also popularly known as the Wiimote. The Wiimote has an Analog Devices ADXL330 3-axis accelerometer which transforms the user's physical motions to activities in the game. Recognizing and classifying these activities can be beneficial to game developers as a means of identifying activities and adjusting the games internal strategy accordingly. Likewise, it is also useful to log and identify motions that do not correspond to any useful activities within the game. These can be visualized as "garbage" classes that are analogous to belches, coughs of human speech. For the purpose of logging and classifying these activities we have interfaced the Wiimote with a Linux machine running Ubuntu 7.1, created a sample game using an open source version of the OpenGL utility toolkit (freeglut) to receive input from the Wiimote and logged and classified various motions that are interpreted with our game. We call the game WiiTT which is short for Wii Table Tennis.

INTRODUCTION

The Wii is the latest games console from Nintendo. Wii aims to change the face of gaming, with a revolutionary controller called the Wii Remote or the Wiimote and new types of games that are based on input from the Wiimote.

The Wiimote communicates with the Wii via a Bluetooth wireless link. The Wiimote can be queried with Bluetooth's Service Discovery Protocol. There are 12 buttons on the Wiimote. Four of them are arranged into a directional pad, and the rest are spread over the controller. The Wiimote also has a speaker, a force feedback mechanism, an IR sensor and an extension port for docking in two other types of controllers – The Nunchuk and the Classic Controller.

Perhaps the most intriguing feature of the Wiimote is that it has the capability to sense motion of the remote by a 3-axis linear accelerometer. This allows for a whole new way for gamers to interact with game consoles.

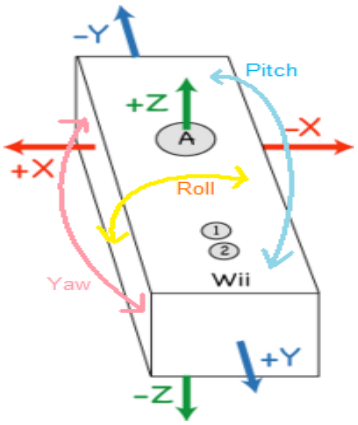

Figure 1: The figure shows the 6 degrees of freedom of the Wiimote.

To interface the Wiimote with our Linux machine we used an open source C-library called LibWiimote (download link http://downloads.sourceforge.net/libwiimote) that provides a simple API for communicating with the Nintendo Wiimote on a Linux system. The goal of this library is to provide a complete easy to use framework for interfacing applications with the Wiimote. The Library (Libwiimote) provides the following features:

Inputs:

- Read accelerometer data from the wiimote (0x31, 0x33, 0x35, 0x37).
- Read key states (0x30-0x37).
- Read battery status (0x20).
- Read nunchuk key, accelerometer and joystick states (0x34-0x37).
- Read IR-sensor data (0x32, 0x33, 0x36, 0x37).
- Read classic controller key and joystick states (0x34-0x37).

Outputs:

- Set/unset the wiimote LEDs (0x11).
- Enable/disable force-feedback effect (0x11).
- Play sounds on the built-in speaker (0x18).

The actual game that we have created is called WiiTT. It is a two player 3D table tennis (ping pong) game. All scene rendering is done via the open source version of the OpenGL utility Toolkit "freeglut".

6 degrees of freedom for Wiimote (see figure 1)

- 3 linear translation directions (X, Y, Z) using accelerometer readings.
- 3 rotation angles (pitch, roll, yaw)(the IR sensor is needed for this)

The accelerometers on Wiimote give readings even when it is stationary. This is because gravity is acting all the time. If all the forces of each accelerometer adds up to a value equal to g then the Wiimote is stationary.

The figures given in Motion Classification section were taken while consciously moving the WiiMote. For the purpose of collecting and analyzing data from WiiMote in real life gamming applications, we have written a PC version of Table tennis game which uses the WiiMote as the virtual TT bats. The game engine is based on OpenGL and Glut library which provides the basic support for drawing primitive shapes in an efficient way.

The rest of this report is organized as follows: Section I gives the instructions on how to play the game, section II talks about the game architecture, section III talks about the real-time data logging, section IV shows some graphs corresponding to different motions of the Wiimote and section V gives the conclusion. The **Appendix** has instructions on how to compile the project.

SECTION I
GAME INSTRUCTIONS

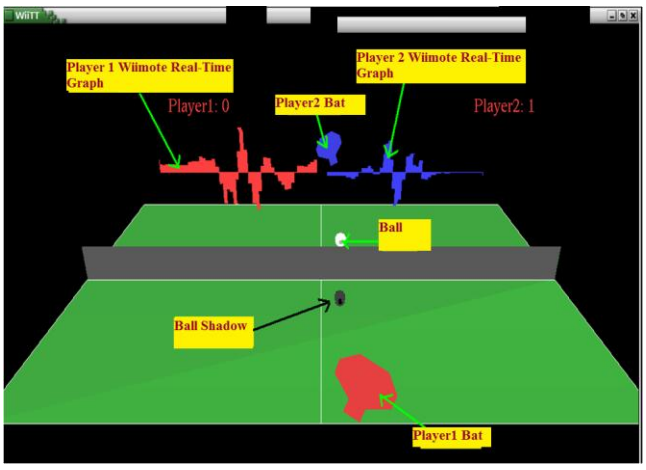

Figure 2: Game Screenshot: Show real-time graph and the game objects

The game is played between two persons each having a WiiMote to control the respective virtual bat on the computer.

Game instructions: (For compilation see Appendix)

1) Make sure a blue tooth dongle is plugged in and its driver is installed.
2) Invoke ./table_tennis from a terminal
3) Press buttons '1' and '2' together on both the Wiimotes.(The BD_ADDR are hardcoded for both the wiimotes and you will have to recompile the code for trying it out with different wiimotes(see init_wii.c and Appendix)
4) To serve press 'B'on the WiiMote.
5) To hit a ball to the left, just press 'A' on the WiiMote and swing it towards left (as if playing a game of table tennis in real world). Note the timing has to be accurate otherwise the ball will go in the opposite direction. This is explained by looking at linear motion figures in the appendix where the graph switches its maxima or minima direction. So the time at which the user presses the button 'A' should be prior to any movement and it should also be at the instant when the ball is very close to the bat.
6) To hit a ball to the right, just press 'A' on the WiiMote and swing it towards right (as if playing a game of table tennis in real world).
7) The angle (left or right) can be controlled in real-time by putting more force in step 2 or 3 while hitting the ball.
8) If a player hits the ball outside the table, the other player gets a point.
9) The service turn switches automatically after every 5 serves by a single player (starting with player 1).
10) The game ends when any of the players reaches the score of 11.

SECTION II
GAME ARCHITECTURE

The game can be divided into the following components;

1) Game Logic: This computes the motion of the ball, player1 bat, player2 bat, interpreting the wiimote readings, collision of bat and ball, updating the scores.
2) Rendering the game: Rendering is done using Opengl. See figure 2 for sample screenshot. It in-

cludes the real-time graphs of both wiimotes, player bats, ball, ball shadow, table and table net.

3) Reading the moves from the wiimote. We use the libwiimote library to interact with the wiimote.
4) Writing the raw wiimote readings to file which can be analyzed later.

The above functionality is done by means of three threads (See figure 3)

Thread 1(Main Thread): The main thread does all the rendering and also handles the logic of the game. The game has the following objects which are rendered on the screen for every frame: Bat1, Bat2, table, net, ball, ball shadow. Except the ball all other objects are represented by means of vertices (and the triangulation of the vertices). The ball is represented by means of a sphere. When the ball is in play the position of the ball is updated with every frame drawn. When the ball is hit by a bat it switches its direction, taking into consideration of the direction intended by the player (by swinging the Wiimote). When the ball comes close to a bat the main thread gets the reading from the data structures which are being constantly updated by thread 2 and 3(see below). This thread also keeps track of whose turn it is and the respective points of the players. For giving a better 3D look we also show the shadow of the ball.

Thread 2 and 3: The main thread cannot read the wiimote readings because it has to do the rendering. If it also reads the wiimote for readings we would not get a smooth rendering as the thread will block every time it tries to get a reading from the wiimote. So we have a dedicated sub-thread for each player which constantly checks for any keys, force or tilts readings from the Wiimote and updates the global structure. This global structure is read by the main thread (before rendering each frame) to identify player moves. The advantage of having a dedicated thread for interfacing the Wiimote is that the main thread concentrates on rendering so the graphics is smooth without any jerks or delays. These threads also log the readings into a file which can be analyzed later on.

FILE LOGGING FOR OFFLINE ANALYSIS:

The game generates 4 output files (2 corresponding to each player) Player1.log and Player2.log: These files contain the entire log of the respective player which happened during the game. A lot of data in these files is actually garbage.
Player1_log.catalog and Player2_log.catalog: This file represents the valid moves that happened during the game play. It has the line numbers corresponding to Player1.log which represent the valid move readings from the WiiMote.

We plotted a graph from Player1.log which is shown in figure 9. The periodic peaks represent the actual moves and rest of the information is garbage. The moves are periodic because the speed of the TT ball is fixed so the moves happen at regular intervals. Since the output from the WiiMote is jerky we average the readings and take the average of past 5 readings as a single reading. The effect of this is to remove the noise or jerky readings from the WiiMote.

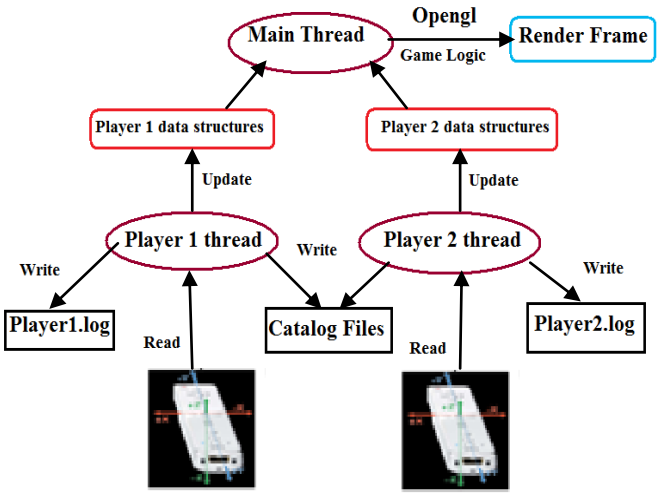

Figure 3

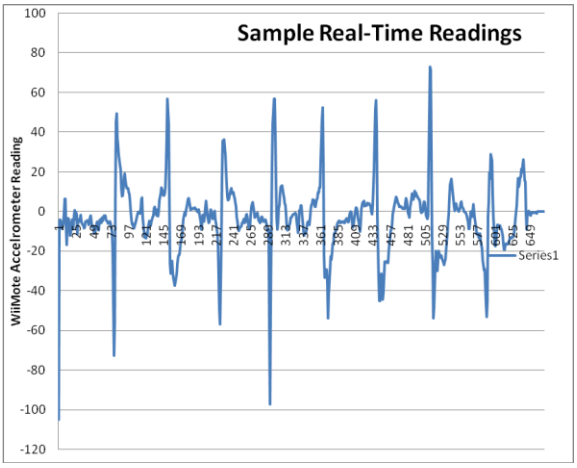

Figure 4

NOISE CONSIDERATION-SMOOTHING

The Wiimote accelerometers give very noisy readings. So we have to remove the noise before it can be interpreted for a particular move. The challenge is to remove the noise and interpret the moves in real-time. So we cannot apply complex filters which consume time. Also we cannot wait for the full graph cycle to complete to interpret the Wiimote reading as the player would experience a delay in the game from the instant he plays a shot and the instant when it is actually interpreted. It will also result in incorrect interpre-

tation of the player moves. We use simple averaging to remove the noise. The code snippet is given below. Another approach would have been to use a median filter. But the simple averaging of readings served the purpose for us in the ping pong game so we have used it.

```
if (wiimote_is_open(w)) {
for(int i = 0;i<SAMPLES_PER_MOVE;i++) {
  if (wiimote_update(w) < 0) {
                wiimote_disconnect(w);
                perror("Wiimote Disconneted Restart Game \n");exit(0);
 }
 w->mode.acc = 1;
                value[i].xval = w->axis.x;
                value[i].yval = w->axis.y;
                value[i].zval = w->axis.z;
                usleep(MOVE_SAMPLING_TIME);
                }
}
for(int j=0;i<SAMPLES_PER_MOVE;i++) {
                x += (value[i].xval - 130.0);
                y += (value[i].yval - 130.0);
                z += (value[i].zval - 130.0);
}
x /= SAMPLES_PER_MOVE;y /= SAMPLES_PER_MOVE; z /= SAMPLES_PER_MOVE;
//SAMPLES_PER_MOV=5
```

RESULTS: MOTION CLASSIFICATION

The Wiimote reports the instantaneous force in the x,y,z directions exerted on the controller by the player holding it or the surface it is resting on. However a free body in 3D has 6 degrees of freedom 3 directions of linear motion and 3 directions of angular motion. Based on the acceleration due to gravity and assuming no linear acceleration otherwise it is possible to calculate the exact orientation of the Wiimote. However if there is linear acceleration then orientation calculations will be skewed due to the Wiimote's inability to report both tilt and linear acceleration separately. This is one of the reasons why we believe an external IR sensor bar is required to figure out directional information. This also limits the usefulness to detect a complex linear and rotational motion.

We have however identified the tilt and linear motions reported by Wiimote's accelerometer and graphs classifying these motions during game play are reported below. Values are scaled by a constant amount for clarity.

Note

- The y axis in all the graphs below represents the magnitude of values reported by the accelerometer.
- The x-axis is the i^{th} reading being polled by the dedicated thread.

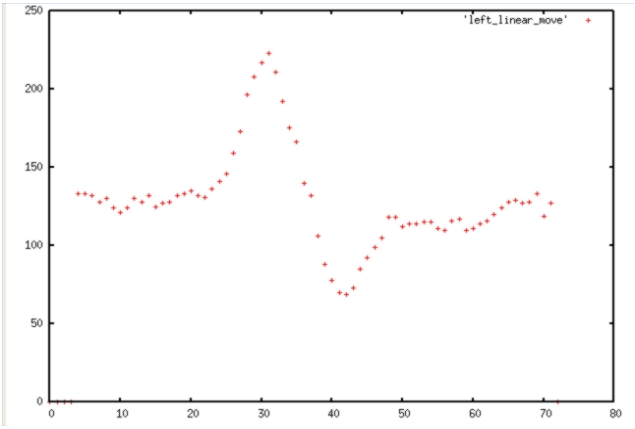

Figure 5: Accelerometer output for a pure linear left move.

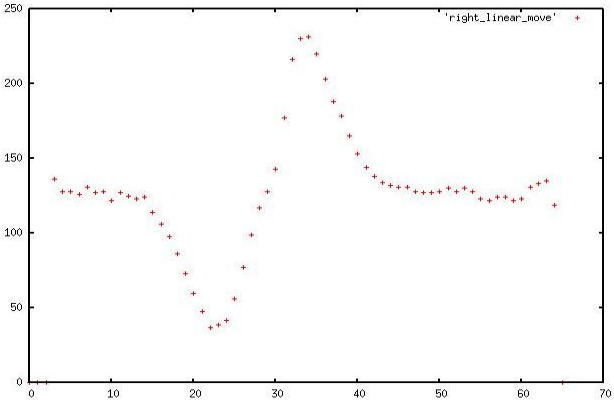

Figure 6: Accelerometer output for a pure linear right move.

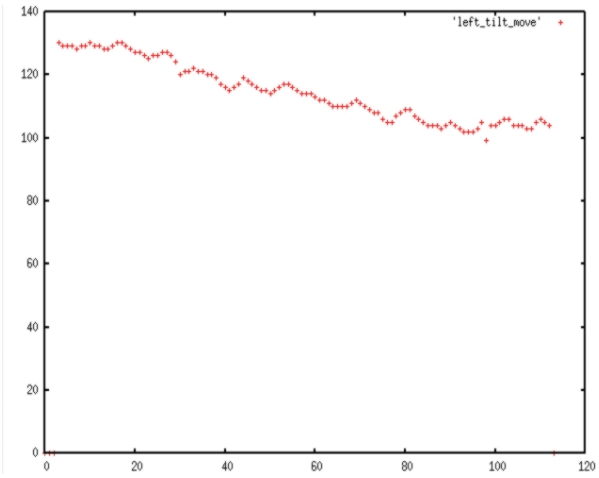

Figure 7: Pure Rotation about the z-axis in a clockwise direction.

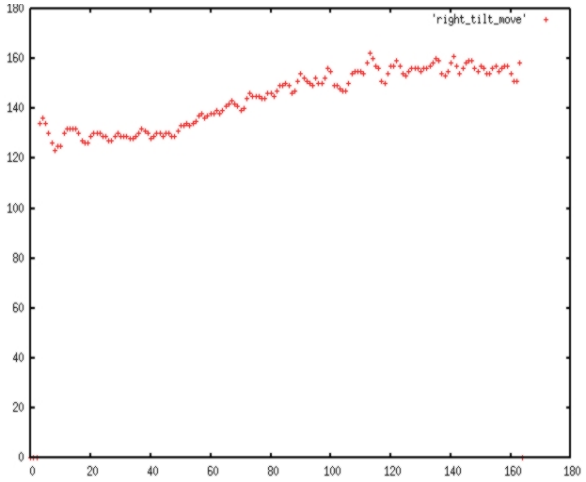

Figure 8: Pure Rotation about the z-axis in an anti-clockwise direction.

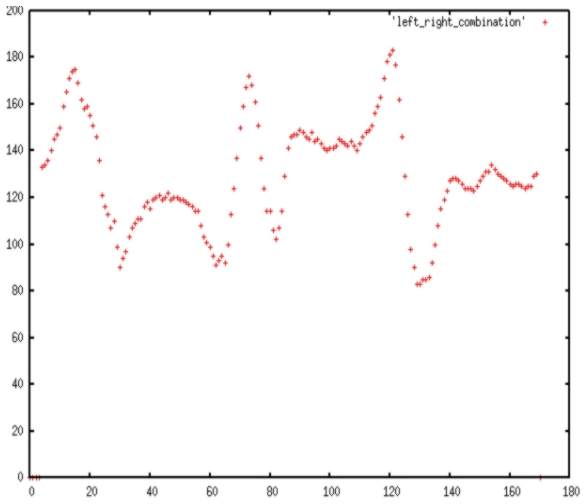

Figure 9: A series of linear moves

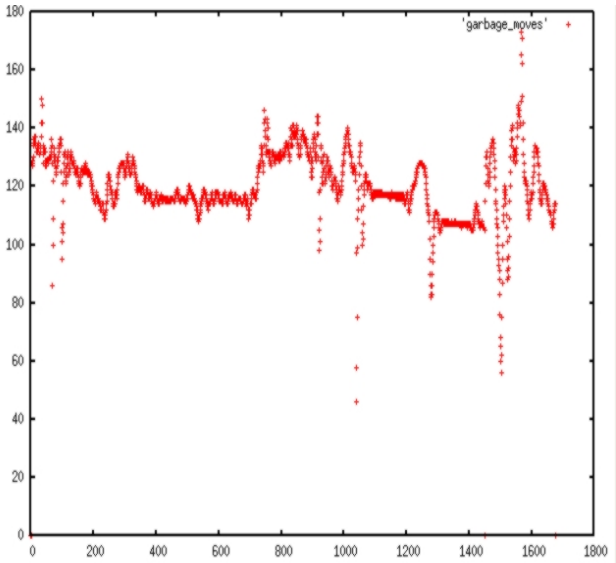

Figure 10: Accelerometer data corresponding to Idle or garbage activity when the Wiimote is in the hand of a player.

SECTION VI
CONCLUSION

In this project we used Wiimote as an input device for a custom Opengl based ping pong game on a computer. We used libwiimote to talk to the wiimote and get the player moves. The readings are also written to a file which can be analyzed later. Various graphs of the logged data are shown for the following movements: left, right, rotation and garbage. The real-time graphs of each wiimote are also shown in the game screen. The intensity of the shots can be varied to control the angle of the returned ball by a player. The game has a 3D look and feel and along with the wiimotes gives a feeling of real-life style experience of a ping pong (table tennis) game.

FUTURE WORK

- Setting up and Interpret the sensor bar's data.
- Transmit the wiimote data to an actual Nintendo console. This will allow logging activity for any game built for Nintendo.
- Analyze other wiimotes: Nunchuk, wheel , Zapper.
- More interesting games with different gestures
- Apply machine learning algorithms for gesture recogntion.
- Explore other applications of Wiimote for a computer. Since it can also act as a generic 3d input device, the possibilities are endless.

ACKNOWLEDGMENTS

We would like to acknowledge the contributions and Guidance of Calton Pu, David Minnen and Chris Skeels for the direction and support that they gave us for this project.

REFERENCES

1. http://www.wiili.org/index.php/Wiimote#Communication
2. http://libWiimote.sourceforge.net/
3. http://www.sparkfun.com/commerce/present.php?p=Wii-Internals
4. http://www.wiili.org/index.php/Motion_analysis
5. http://www.analog.com/en/subCat/0,2879,764%255F800%255F0%255F%255F0%255F,00.html

APPENDIX

Compiling the code

You would require freeglut to compile and run the game. If you have freeglut installed the simply run the following command to compile the project
$./compile_me
This will create a binary called table_tennis in the current directory.

Running the executable

1) Make sure a Bluetooth dongle is plugged in and its driver installed.
2) Invoke the binary "./table_tennis"
3) Press buttons '1' and '2' together on both the Wiimotes.

Please note: The BD_ADDR are hardcoded for both the wiimotes and you will have to recompile the code for trying it out with different wiimotes. You can change this in in-it_wii.c:
wiimote_connect(&wiimote1, "00:19:FD:4F:FF:47")
wiimote_connect(&wiimote2, "00:1A:E9:50:04:5B")