Studying The Wiimote

Exploration To Enhance The Feeling Of Interactivity In Applications

vrije Universiteit amsterdam

Danny Zulkarnain
Steve Stomp

September 2007

Index

Studying The Wiimote

Exploration To Enhance The Feeling Of Interactivity In Applications

1. Introduction

Despite the many revolutionary advancements in technology, the basic interface between the human and the computer system has received relatively little attention in terms of evolution. Recently Nintendo released a new video game console the Wii. This new console is controlled by a wireless Bluetooth controller, the Wiimote. Through its pointing and motion-sensing abilities Nintendo hoped to make a console accessible to people of all ages and abilities. Nintendo did not expected that users would re-engineer the device to do all sorts of things having nothing to do with playing videogames. Our aim is to study the Wiimote's features and possibilities to enhance the feeling of interactivity in either audiual and visual applications on the personal computer (Windows) platform.

This paper consist of four chapters, of which the introduction is the first. In the second chapter we will begin with giving a technical overview of the Wiimote along with how communication is set up. The third chapter discusses the project space, so which software and hardware are used. The paper is concluded with a chapter in which we give proposals for potential follow-up studies.

In able to set up a Wiimote with a computer the following things are needed:

- Wiimote (and Nunchuk)
- Bluetooth device
- Windows operating system
- LED sensor bar
- GlovePie *(optional)*
- MidiYoke *(optional)*

2. Input Device

2.1 Wiimote

The Wiimote is an one-handed wireless input device/controller and consist of the twelve buttons, four of which are arranged into a directional pad and the rest is spread over the device. It's symmetric design allows it to be used with either the left or right hand. Figure 2.1 gives overview of the layout. The expansion port on the bottom of the unit is used to connect the remote to auxiliary controllers which augment the input options of

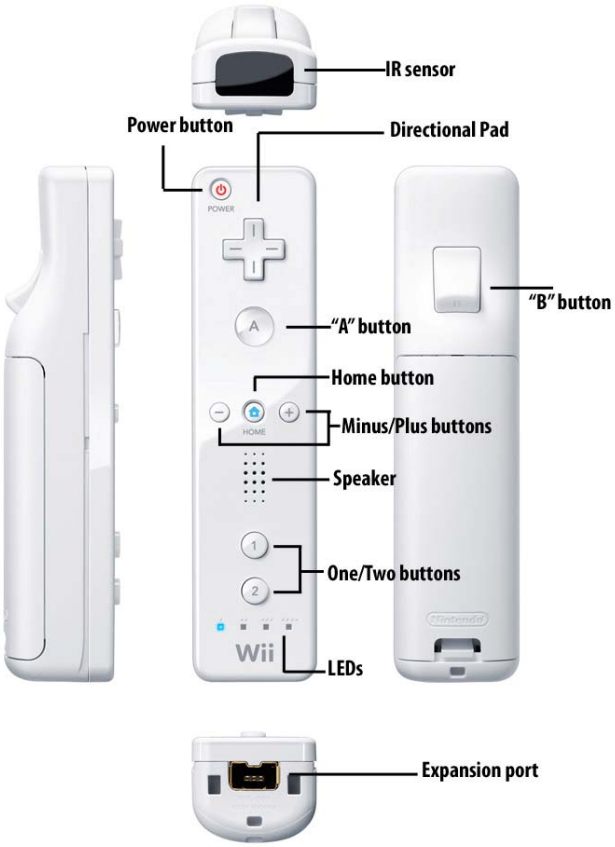

Figure 2.1, Wiimote

the Wiimote. Auxiliary controllers use the interface of the Wiimote to communicate with the host. One of the auxiliary controllers is the Nunchuk (see figure 2.2). It features an analog stick enabling more control options. The controller uses two AA batteries as a power source, which can power it for 30 hours if used with a auxiliary controller or 60 hours if used alone.

Figure 2.2, Nunchuk

As figure 2.1 shows the Wiimote contains four blue LEDs, these LEDs are used during normal use to indicate that the remote is in Bluetooth discoverable mode (all blinking), or to indicate the users number of the controller (only one illuminated). Furthermore the controller is provided with a rumble, via a motor with an unbalanced weight attached to it inside the Wiimote, which can be activated to cause the controller to vibrate. And it has a low-quality speaker used for short sound effects during use.

.

2.2 Communication

Communication with the host is accomplished through a Bluetooth wireless link. The Bluetooth controller is a Broadcom 2042 chip, which is designed to be used with devices which follow the Bluetooth Human Interface Device (HID) standard, such as keyboards and mice. Not all Bluetooth devices are compatible with the Wiimote. The following website gives a list for all working and non-working Bluetooth devices, http://wiibrew.org/index.php?title=List_of_Working_Bluetooth_Devices.

2.3 Motion Sensor

Internally the Wiimote has a integrated circuit called the ADXL330 accelerometer. This device has the ability to measure acceleration along three axis (see figure 2.3). The device is supported by springs built out of silicon, and through the force exerted on these springs the sensor measures the acceleration of the Wiimote. Due to the sign convention used, this quantity is proportional to the net force exerted by the user's hand on the Wiimote when holding it.

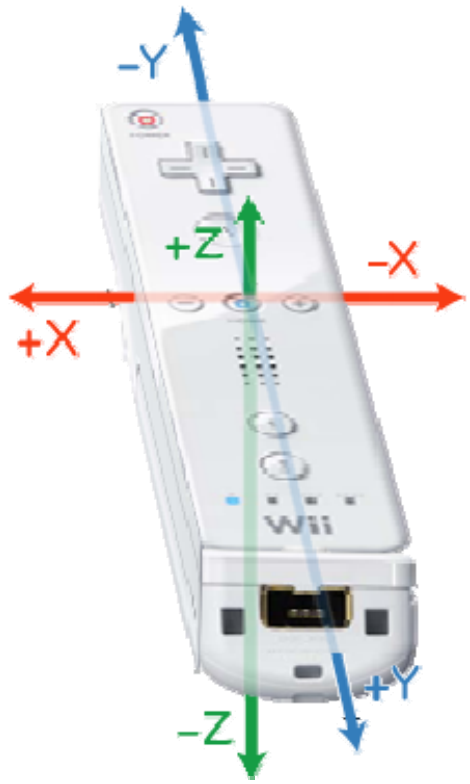

Figure 2.3, motion sensing

Besides motion-sensing abilities, with the accelerometer, the Wiimote is also augmented with an infrared image sensor on the front. This feature is designed to locate IR beacons within the user's field of view, and thus creating position-sensing abilities. These beacons are transmitted through the sensor bar. By tracking the locations of the beacons in contrast to the Wiimote sensor, the system can derive more accurate pointing information. The Nintendo sensor bar contains 10 LEDs, five on each side. The LEDs closest to the center are pointing inward and the farthest are pointing outward. The LEDs enable the Wiimote to calculate the distance from the bar and even the angle of the device with respect to the ground (tilting and rotating). Out of these position values other data (size and pixel value) can be requested. It is not necessary to use a sensor bar containing as many lights as the version of Nintendo. A bar containing two LEDs will be enough to enable the position-sensing abilities.

3. GlovePie

3.1 Application

GlovePie (Glove Programmable input emulator) is an application specifically designed to emulate joystick and keyboard input through running custom scripts. The application was initially designed for virtual reality gloves as a system for emulating joystick and keyboard input. However it has evolved to support all kinds of devices including the Nintendo Wiimote. By using GlovePie it is possible to handle and program the input from the Wiimote, to use this device to control programs that are used with a keyboard, mouse or joystick. GlovePie is only compatible with Windows operating systems, though there are other programs for utilizing the Wiimote in other operating systems (http://www.wiili.org)

3.2 Scripting

GlovePie uses an interface that resembles a text editor. In able to use GlovePie as an interpreter to interpret Wiimote signals, it is necessary to program scripts that correspond to specific keys or buttons. The syntax used in this program is similar to Java and BASIC. For example to assign the Wiimote's button A to keyboard button 'A' :

- Key.A = Wiimote.A

With GlovePie it is possible to use multiple Wiimotes, which can be distinguished by using consecutive numbers with a maximum of 8. For instance to call button 'A' from different Wiimotes and assign them to different keyboard entries.

- Key.X = Wiimote1.A
- Key.Y = Wiimote2.A

The available inputs of the Wiimote can be categorized into buttons, D-pad (directional-pad), acceleration, rotation, sensor bar, LEDs and rumble.

Buttons:

- Wiimote.A
- Wiimote.B
- Wiimote.Plus
- Wiimote.Home
- Wiimote.One
- Wiimote.One
- Wiimote.Two

D-pad:

- Wiimote.Up
- Wiimote.Down
- Wiimote.Left
- Wiimote.Right

Acceleration:

- Wiimote.RelAccX / Wiimot.RawAccX
- Wiimote.RelAccY / Wiimote.RawAccY

- Wiimote.RelAccZ / Wiimote.RawAccZ

Rotation
- Wiimote.Roll
- Wiimote.Pitch

Sensor Bar (# = 1,2,3,4)
- Wiimote.Dot#x
- Wiimote.Dot#y
- Wiimote.Dot#size
- Wiimote.Dot#vis

LEDs
- Wiimote.Led1
- Wiimote.Led2
- Wiimote.Led3
- Wiimote.Led4

Rumble
- Wiimote.Rumble

Speaker
- Wiimote.Frequency
- Wiimote.Volume
- Wiimote.Speaker
- Wiimote.Mute
- Wiimote.SampleRate

The Wiimote controls can be extended using the Nintendo Nunchuck (see chapter 2) to have extra controls.

Buttons:
- Wiimote.Nunchuk.Cbutton
- Wiimote.Nunchuk.Zbutton

Joystick:
- Wiimote.Nunchuk.JoyX
- Wiimote.Nunchuk.JoyY

Acceleration
- Wiimote.Nunchuk.RawForceX
- Wiimote.Nunchuk.RawForceY
- Wiimote.Nunchuk.RawForceZ
- Wiimote.Nunchuk.gx
- Wiimote.Nunchuk.gy
- Wiimote.Nunchuk.gz

Basic directional-pad script mapped to keyboard

// If the Wiimote is tilted over an angle lower than -20 degrees then keyboard button up is pressed

- Key.Up = (Wiimote1.Pitch <= -20 degrees)

// If the Wiimote is tilted over an angle higher than 20 degrees then keyboard button down is pressed

- Key.Down = (Wiimote1.Pitch >= 20 degrees)

// If the Wiimote is rotated over an angle lower than -45 degrees then keyboard button right is pressed

- Key.Left = (Wiimote1.Roll <= -45 degrees)

// If the Wiimote is rotated over an angle higher than 45 degrees then keyboard button right is pressed

- sKey.Right = (Wiimote1.Roll >= 45 degrees)

3.3 Scripts

3.3.1 Powerpoint

These are some basic PowerPoint scripts to control a PowerPoint presentation using a Wiimote.

//powerpoint-control: next slide

- Key.PageUp = (Wiimote1.Roll <= -45 degrees)
- Key.PageUp = Wiimote1.RelAccX <= -20 m per s *//alternate*

//powerpoint-control: previous slide

- Key.PageDown = (Wiimote1.Roll >= 45 degrees)
- Key.PageDown = Wiimote1.RelAccX >= 20 m per *s //alternate*

//powerpoint-control: 1th slide

- Key.One = Wiimote1.Minus
- Key.Enter = Wiimote1.Minus

//powerpoint-control: display/hide white screen

- Key.W = Wiimote1.A
- Key.Escape = Wiimote1.Home

3.3.2 Midi VJ-tool

In this script example the Wiimote signals are mapped to midi entry keys. With an midi emulator like MidiYoke (http://www.midiox.com/) it is possible to map the Wiimote-controls to midi signals.

- midi2.C0 = Wiimote.Up
- midi2.D0 = Wiimote.Left
- midi2.E0 = Wiimote.Right
- midi2.F0 = Wiimote.Down

- midi2.G0 = Wiimote.A
- midi2.A0 = Wiimote.B

- midi2.B0 = Wiimote.Minus

- midi2.C1 = Wiimote.Home
- midi2.D1 = Wiimote.Plus

- midi2.E1 = Wiimote.One
- midi2.F1 = Wiimote.Two

In this example a midi input is mapped to a combined operation of accelerating the Wiimote in the x-axis while pressing the A key.

- midi.g2 = (Wiimote1.RelAccX <= -20 m per s) and (Wiimote1.A)
- midi.gsharp2 = (Wiimote1.RelAccX >= 20 m per s) and (Wiimote1.A)

3.3.3 Halflife2[1]

This script uses the Wiimote, Nunchuck and a infrared source. In this script the Nunchuk is used to navigate through the scene and the Wiimote is used to control the weapons.

```
var.dummy = Wiimote.rawforcex

// Get Nunchuk axis locations. Range is -0.99 to 0.99
// Multiply by 100 to get whole numbers. (-99 to 99)
var.xNunchuk = Wiimote.Nunchuk.JoyX * 100
var.yNunchuk = Wiimote.Nunchuk.JoyY * 100
// X/Y offsets for Analog. If it's too sensative then make the numbers larger.
var.xOff = 7
var.yOff = 7
// Acceleration ammount for Nunchuk Reload (default = 17.0)
var.nunchukAccX = 19.0
// Blink rate for battery check.
var.Blink = 500ms
// Mouse IR Offsets - Use these for calibrating.
var.xOffset = 0
var.yOffset = 0
var.irAmount = 2
// Master Sensitivity (Default 80) - Change Not Recommended
var.smooth = 80
// Look Speed (Default 1)
var.speed = 1
// Speed When Aiming Down Sight (Default 1/2)
var.zoom = 1/2
// Less Sensitive Area Around Cursor (Default 40)
var.deadzone = 40

//
// Analog Movements
//
if var.xNunchuk > var.xOff and var.yNunchuk > var.yOff then
```

[1] This script is for Halflife2 created by Marco Ceppi.

```
  key.w = false
  key.a = false
  key.s = true
  key.d = true
  //debug = 'SE'
else if var.xNunchuk > var.xOff and var.yNunchuk < -var.yOff then
  key.w = true
  key.a = false
  key.s = false
  key.d = true
  //debug = 'NE'
else if var.xNunchuk < -var.xOff and var.yNunchuk < -var.yOff then
  key.w = true
  key.a = true
  key.s = false
  key.d = false
  //debug = 'NW'
else if var.xNunchuk < -var.xOff and var.yNunchuk > var.yOff then
  key.w = false
  key.a = true
  key.s = true
  key.d = false
  //debug = 'SW'
else if var.xNunchuk > var.xOff then
  key.w = false
  key.a = false
  key.s = false
  key.d = true
  //debug = 'Right'
else if var.xNunchuk < -var.xOff then
  key.w = false
  key.a = true
  key.s = false
  key.d = false
  //debug = 'Left'
else if var.yNunchuk < -var.yOff then
  key.w = true
  key.a = false
  key.s = false
  key.d = false
  //debug = 'Up'
else if var.yNunchuk > var.yOff then
  key.w = false
  key.a = false
  key.s = true
  key.d = false
  //debug = 'Down'
else if var.xNunchuk > -var.xOff and < var.xOff and var.yNunchuk < var.yOff and > -var.yOff then
  key.w = false
```

```
  key.a = false
  key.s = false
  key.d = false
  //debug =
else
  key.w = false
  key.a = false
  key.s = false
  key.d = false
  //debug
endif

//
// Game buttons.
//
// Wiimote
key.Shift = Wiimote.Up
mouse.WheelUp = Wiimote.Left
mouse.WheelDown = Wiimote.Right
key.e = Wiimote.Down

mouse.LeftButton = Wiimote.B
Wiimote.Rumble = Wiimote.B
mouse.RightButton = Wiimote.A

key.q = Wiimote.Minus
key.escape = Wiimote.Home
key.f = Wiimote.Plus

key.One = Wiimote.One
// Wiimote.Two handles Battery.

//
// Nunchuk
//
key.space = Wiimote.Nunchuk.ZButton
key.Ctrl = Wiimote.Nunchuk.CButton

//
// Reload
//
if Wiimote.Nunchuk.RawAccX > var.nunchukAccX or < -var.nunchukAccX then
key.r = true
key.r = false
//debug = 'VROOM!'
endif

Wiimote.leds = 0
```

```
//
// Battery Check!
//
// A full battery gives 0xC0 (192)
if Wiimote.Two == true then
  var.Batt = Wiimote.Battery / 48

  if true then
    wait 5 seconds
    // it sends an instruction that tells the Wiimote to actually
    // send the report.
    Wiimote.Report15 = 0x80 | Int(Wiimote.Rumble)
  endif

  // Display the battery level of your Wiimote using the four LEDs on the bottom.
  // Battery level is displayed in four levels increasing to the right, like a cell
  // phone battery gauge. As the battery gets close to the next level down, the LED
  // for the current level will blink.

  debug = "Battery level: " + 100*48*var.Batt/192 + "%"
  if 0 <= var.Batt <= 0.25 then
    Wiimote.Leds = 1
    wait var.Blink
    Wiimote.Leds = 0
    wait var.Blink
  elseif 0.25 < var.Batt<=1 then
    Wiimote.Leds = 1
  elseif 1 < var.Batt<=1.25 then
    Wiimote.Leds = 3
    wait var.Blink
    Wiimote.Leds = 1
    wait var.Blink
  elseif 1.25 < var.Batt<=2 then
    Wiimote.Leds = 3
  elseif 2 < var.Batt<=2.25 then
    Wiimote.Leds = 7
    wait var.Blink
    Wiimote.Leds = 3
    wait var.Blink
  elseif 2.25 < var.Batt<=3 then
    Wiimote.Leds = 7
  elseif 3 < var.Batt<=3.25 then
    Wiimote.Leds = 15
    wait var.Blink
    Wiimote.Leds = 7
    wait var.Blink
  elseif 3.25 < var.Batt<=4 then
    Wiimote.Leds = 15
  else
```

```
    Wiimote.Leds = 0
  endif
endif

//
// IR Mouse
//
If var.irAmount = 2 Then
  var.xPos = (Wiimote.dot1x + Wiimote.dot2x) / 2
  var.yPos = (Wiimote.dot1y + Wiimote.dot2y) / 2
Else
  var.xPos = Wiimote.dot1x
  var.yPos = Wiimote.dot1y
EndIf

If Wiimote.dot1vis Then
  // Locate Inrared Point Coordinates
  var.actualX = (1-(round(var.xPos) / 1024)) * Screen.Width
  var.actualY = ((round(var.yPos) / 768)) * Screen.Height
  // Determine Look Speed
  var.speedX = (((var.actualX / (Screen.Width / 2)) - 1) * var.smooth) + var.xOffset
  var.speedY = (((var.actualY / (Screen.Height / 2)) - 1) * var.smooth) + var.yOffset
  // Calculate Point-Range Multipliers
  If abs(var.speedX / var.deadzone) > 1 Then var.multX = 1 Else var.multX = abs(var.speedX / var.deadzone)
  If abs(var.speedY / var.deadzone) > 1 Then var.multY = 1 Else var.multY = abs(var.speedY / var.deadzone)
  // Scoped And Normal Multipliers
  If Wiimote.A = True Then
  var.speedX = var.speedX * var.zoom
  var.speedY = var.speedY * var.zoom
  Else
  var.speedX = var.speedX * var.speed
  var.speedY = var.speedY * var.speed
  EndIf
  // Move Cursor
  If abs(var.speedX) > 0 Then Mouse.CursorPosX = Mouse.CursorPosX + (var.speedX * var.multX)
  If abs(var.speedY) > 0 Then Mouse.CursorPosY = Mouse.CursorPosY + (var.speedY * var.multY)
  // Backup Last Motions
  var.lastX = var.speedX
  var.lasty = var.speedY
Else                    // If dot is not visible use last known value
  If abs(var.lastX) > 1 Then Mouse.CursorPosX = Mouse.CursorPosX + var.lastX
  If abs(var.lastY) > 1 Then Mouse.CursorPosY = Mouse.CursorPosY + var.lastY
EndIf
debug = 'Debug: SpeedY: ' + var.speedY + '  SpeedX: ' + var.speedX
```

3.4 Pros and Cons of GlovePie

Pros	Cons
Extensively programming experience is not needed to use GlovePie it is relatively easy to program.	Need several programs to connect Wiimote to a program, such as MidiYoke for midi input and PPjoy to connect all kinds of joystick devices.
It is also possible to use the Graphical User Interface to map keys. Using this interface it is only needed to press the keys that is desired to map to each other. However it is not error-free and it still necessary to define some range in variables, like the angle when using the rotation. The GUI just specifies the minimum and maximum range.	GlovePie is a little limited to connecting Wiimote-gestures/buttons to keys. In some cases it would be nice to be able to manipulate the variables or have direct access to the variables that are send from the Wiimote. For example mapping rotation angle of the Wiimote to acceleration.
Any keyboard, mouse or joystick controlled application/software can be controlled with a Wiimote.	When using only GlovePie, Wiimote mapping is restricted to keyboard or mouse input.
	When mapping to keyboard keys, you have to keep in mind that keyboard functions are shared with all kinds of programs. Because of this reason the program one wishes to control with the Wiimote has to be activated and on-top in the window layers.
	GlovePie can only be used on a windows operating system.
	Without the distribution of the script it can only be used on a single computer.

4. Conclusion

GlovePie enables users to actually choose any software they like to control with the Wiimote. Only thing needed are the exact scripts and possible other interpreters or emulators. The need for own created scripts and the fact that GlovePie has to run simultaneously with the application, does not make it an easy-to-use program. An interesting subject for further study would be investigating the possibilities of programming special pc-software or online-applications that are controlled by the Wiimote, without the use of GlovePie. Various software has been created, such as Virtual Drum Kit, BlueTunes (an application to control various media players without any scripting), etc. These programs can be found at www.brianpeek.com/blogs/pages/net-based-wiimote-applications.aspx. They are created with a C-Sharp library called Managed Library Wiimote which is developed by Brian Peek (blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx). This library gives extensive possibilities for the Wiimote. For example using it for motion capture[2] or interactive installations.

Furthermore it is also interesting to investigate the possibilities in combination with Virtools. Virtools supports a wide variety of 3D formats and allows users to create all sorts (web) applications. Recently the developer of Virtools has added Wii building block to its latest release (Virtools 4), allowing the access of WIi controllers on PCs with Virtools. Although Virtools is not an easy-to-use program it is still an interesting option for further study, because of the important fact that you can import all sorts 3D models including animation, which makes it very appropriate for creating Wiimote controlled games on PCs.

The last interesting subject is XNA. Microsoft recently released extension for its XNA development platform which allows users to use the Wiimote as an input device. This platform is used to create games on either PC or Xbox360. At the moment it can only be used for creating for PC, because the Xbox360 isn't equipped with Bluetooth.

[2] Using only the capabilities of the Wiimote sets certain boundaries. To detect motion it uses the sensor bar, so you always have to be in the field of view of the sensor bar.

5. Bibliography

- WiiLi (2007), visited on 5 August 2007 on http://www.wiili.org/index.php/Main_Page
- WiiBrew Wiki (2007), visited on 26 July 2007 on http://wiibrew.org/index.php?title=Main_Page
- GloviePie (2007), visited on 15 July 2007 on http://carl.kenner.googlepages.com/glovepie_download
- Midiox (2006), visited on 15 July 2007 on http://www.midiox.com/
- XNA (2007), visited on 5 August 2007 on http://msdn2.microsoft.com/en-us/xna/default.aspx
- Analog Devices (2007), visited on 5 August 2007 on http://www.analog.com/en/prod/0,2877,ADXL330,00.html
- Virtools (2007), visited on 5 August 2007 on http://www.virtools.com/
- Brain Peek (2007), visited on 6 August 2007 on http://www.brainpeek.com/blog
- Coding4Fun (2007), visited on 26 July 2007 on http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx