

Joel Clemens  
Senior Thesis Project Plan  
Virtual Wiiality

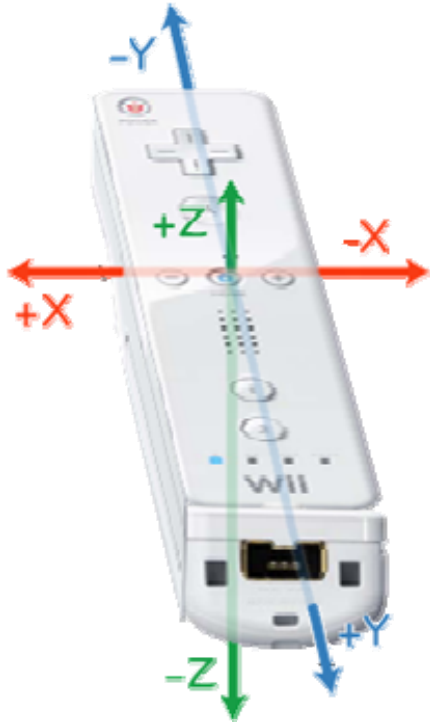


## **Idea - Virtual Wiiality**

Inspired by the limitations of conventional computer monitors and lack of affordable VR systems, I wanted to develop an affordable head tracking system using a head mounted display and the Nintendo Wiimote. The idea starts with simple head movement tracking, but I want to expand that into full motion tracking if possible. This is where a virtual universe would be a mirror of reality. The user's forward walking movement would cause forward movement of the same speed in the computer along with the other directions.

## **Research**

The primary source of information dealing with interfacing a Wiimote and a computer is [www.wiili.org](http://www.wiili.org). The Wiimote has 3 accelerometers for X-Y-Z translational motion tracking. An accelerometer is a transducer, (device that converts real life analog events into electrical signals that can be read by computers), that detects a scale of force (or acceleration), positive or negative. A force that is always acting on the Wiimote is gravity (unless it is in free fall in a vacuum, where all forces would be zero). Because the Wiimote has 3 accelerometers, force can be detected when it is moved (accelerated) forwards, backwards, left, right, up, or down, (or a combination). This adds up to the often mentioned "6 axis" that the Wiimote has. It also has an infrared (IR) camera that is setup to track IR emitters in a two-dimensional plane. The IR tracking is used for cursor movement. The IR camera has a filter so only infrared light can be seen. It is programmed to track movement patterns that can be from any IR source, but it is the most reliable when tracking IR LEDs. Below is an image depicting the direction of the motion sensor's orientation.



(in GlovePIE, the Z and Y axis are switched)

The Wiimote can be interfaced with a computer with a standard Bluetooth adapter. Programs are already available that can read the data that the Wiimote is transmitting. A list of compatible and non-compatible devices is available on the Wiili website (see appendix for link). This project will use the scripting program GlovePIE because of its flexibility. GlovePIE is a program for Windows written by Carl Kenner to emulate devices on a computer. PIE stands for Programmable Input Emulator and was originally developed to utilize the P5 VR glove. Any common device's output data can be read, and put in place of another device. (Common devices such as keyboard, mouse, joystick, and even the Wiimote.) GlovePIE runs scripts in a loop 40 times each second (by default).

## Designs

The priority will be placed on coding/scripting in this project. A secondary design task would be to create a virtual environment that matches the room that is used for

demonstration. The user can interact with solid objects that exist in reality and virtual reality, and witness things that can only exist in the virtual reality.

The secondary design cannot exist without properly functioning code. This project can be broken up into 3 major coding milestones:

1. head tracking while sitting stationary, looking in one general direction
2. head tracking with 360 degrees of motion, (vertical and horizontal).
3. full motion tracking (user can physically walk forward in both real and virtual worlds)
  - a. minimally forward and backward motion
  - b. turning head results in turning body in virtual world

As of this writing, some limited success has been achieved in a first milestone using basic mouse emulator scripts, 2 emitters. This resulted in about 30 degrees of head tracking.

Here is a sample script for basic mouse movement using only the IR functions and 2 IR emitters.

```
-----  
//puts data from Wiimote buttons into mouse buttons  
Mouse.RightButton = Wiimote.B  
Mouse.LeftButton = Wiimote.A  
Mouse.MiddleButton = Wiimote.Home  
  
//if IR emitters are visible, display on Wiimote lights  
Wiimote.Led1 = Wiimote.dot1 vis  
Wiimote.Led2 = Wiimote.dot2vis  
  
//maps Wiimote left and right buttons to keyboard (added for power point)  
key.Left = Wiimote.Left  
key.Right = Wiimote.Right  
  
//if statement allows regular mouse to be used if Wiimote cannot track any IR  
//sources  
if (Wiimote.dot1 vis = true) and (Wiimote.dot2vis = true)
```

```

//regular mouse cannot be used while data is being put into mouse
//coordinates (there is conflict)
//mouse coordinates start at top left of screen, Wiimote cursor co-ordinates
//start at
//top left of screen, and uses 1024x768 resolution.
//this code says put the average position of the 2 tracked IR sources into
mouse //system
Mouse.DirectInputX = 1024-((Wiimote.dot1x + Wiimote.dot2x)/2)
Mouse.DirectInputY = ((Wiimote.dot1y + Wiimote.dot2y)/2)
endif
-----

```

The Wiimote can track up to 4 emitters at a time, however, many more than that will be needed to track 360 degrees of motion. To accomplish this, multiple LED emitters are required to surround the user. Experimentation needs to take place to see how fast the Wiimote can handle new dots to track. Also, the code will be significantly different because rather than dumping the average co-ordinates of the dots into the mouse input, the difference or change of position of the dots need to increment or decrement the values that go to the mouse input.

The same IR tracking can be used for vertical rotation (looking up and down), however, the amount of LEDs that are required would increase exponentially. A function that is available in GlovePIE is a pre calculated force vector from gravity that gives the current angle of down on a given axis. This angle does not change with a swivel motion caused by looking left and right (known as yaw). But, the angle of gravity along the forward/backward axis changes when tilting up and down (pitch). This angle value can be translated into a value that the mouse can use when looking up and down in the game environment. (Since the character in a game cannot typically look up/down past + or -90 degrees, straight forward being 0). If full motion tracking were implemented, forward and backward acceleration could give erroneous data on the angle because gravity would no longer be the only force in the equation. Not only that, but if the user looks down at the

ground, or up at the (virtual) sky, there will not be anything for the Wiimote to track. This means that the user could not look up (or down) and to the left (or right) at the same time. If a full array of LEDs were created to surround the user, this would be possible, however, if the users walks forward/backward (with full motion tracking), the LEDs will start to move in relation to the Wiimote. This would cause the code to act as if there was head movement, when in fact it is body movement. (Though while looking down/up, there is an angle value along the left/right axis that becomes available that could be used for left and right tracking).

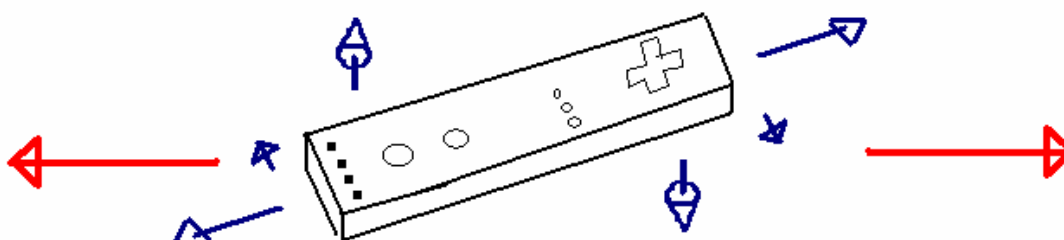
Because the Wiimote cannot distinguish the difference between the force of gravity and other forces from movement, accurate motion tracking has proven to be quite difficult. In theory, the Wiimote's accelerometers can detect the acceleration of a walk or a run, and the deceleration of the same walk or run. This data can be calculated to a value than can put input into a joystick emulator (PPjoy – see appendix). Physics dictate that one needs the same acceleration in the opposite direction to return to a complete stop. The problem with initial test code was that, only 1 axis could be used at a time. (ideally forward and backward). Relative (to compensate for gravity) values on this axis were read and used to increment the joystick value, however upon deceleration, the joystick did not decrement back to 0. This likely to be caused by the imperfect, non-robotic movement of human motion. The start-up and finishing force values did not equal because they were not completely on the same axis at the beginning and end of the test motion. Please refer to the following diagrams for further explanation.



If the Wiimote was moved perfectly straight, then only the forward accelerometers would be used. Calculating a value to put into a joystick would be very simple. (Blue lines being the accelerometers and the red lines be the total force exerted on the Wiimote.

However

It is impossible to keep the Wiimote straight at all times and all axes must be taken into effect when calculating the force. (Force Vectors)



Red Force (actual motion) is measured by all 3 sensors.  
 And can be calculated by sum of the intensities and angles of the accelerometers  
 $F_x + F_y + F_z = T(\text{total})$

This method is used within GlovePIE to calculate the current angle of gravity.

If the Wiimote, (or similar device – see appendix) could somehow detect rotation along the vertical axis (as when turning your head left and right), then IR tracking would not be necessary, and much more freedom could be realised if full motion tracking were to work. Currently the Sony Playstation 3 six axis controller is capable of this, however, it does not have the same level of community support as the Wiimote.

### Parts and costs

- Wiimote 45\$
- head mounted display 200\$
- Bluetooth adapter 20\$
- GlovePIE - free

- PC – pre-existing
- IR emitters – maximum 20\$
- Code – costs only time
- Virtual environment – pre-existing (will use Source Engine by Valve software)

**Schedule and Priorities**

- 1 get multiple LEDs to track left and right rotational movement
- 2 develop code that allows up and down rotational movement based on direction of gravity (angle)
- 3 test in environment, create keyboard/mouse free system of interaction in 3d game. (uses the WiiNunchuck) (phase 1 complete)
- 4 develop forward backward walking motion tracking (phase 2 complete)
- 5 create custom environment to demonstrate (final phase complete)

Optional things, relative position tracking, wireless video, proper headset

**Timeline**

Task	January	February	March	April
Step 1a	Build IR emitter array	Create system that allows full movement without keyboard -this implements the WiiNunchuck	Full motion tracking	Finalize documentation for online publishing and presentation
Step 1b	Write code for rotational motion		Full motion tracking	
Step 2	Develop code for up/down motion or expand IR array to encompass user.	Test on existing environment (Valve game)	Full motion tracking	Present project
		Start initial attempts for full motion tracking		
Extra non-critical stuff	Clean up wires and ensure durability in hardware.	Start learning about environment design Investigate status of PS3 controller as Wiimote alternative	Complete 3d demonstration environment	(optional wireless video modal)

## **Risks**

It is very difficult to estimate exact time consumption when writing code that still involves a learning curve. Things rarely work in the desired way immediately and often have a random amount of time until they do work, (if at all). The biggest challenge in this project is the full motion tracking, but knowing that it is theoretically possible gives some encouragement.

If yaw were to become available somehow (such as through the PS3 controller), then it should not be too difficult to convert this project to use such device. However, a risk could be that this doesn't happen, and I would be faced with the challenges of the Wiimote not having yaw. (Rotating left/right.)

## **Appendix**

[http://www.wiili.org/index.php/Compatible\\_Bluetooth\\_Devices](http://www.wiili.org/index.php/Compatible_Bluetooth_Devices)

(list of Bluetooth devices that work, and don't work)

<http://www.geocities.com/deonvdw/PPJoy.htm>

(PPjoy joystick emulator)

<http://www.pabr.org/sixlinux/sixlinux.en.html>

(Example of PS3 controller used on computer in similar ways as the Wiimote.)

[http://www.wiili.org/forum/use-4-ir-sensors-get-full-range-of-motion---\)-t2832.html](http://www.wiili.org/forum/use-4-ir-sensors-get-full-range-of-motion---)-t2832.html)

(GlovePIE developer Carl Kenner's comments on head tracking)