Excerpt from GlovePIE 0.29 Documentation

When using your Wiimote for your video game, the Wiimote transmits data to the computer using a wireless communication protocol called Bluetooth. GlovePIE is the software that interprets this Bluetooth data and translates it into key strokes. The pages below deal with Wiimote implementation. The full document can be found by downloading the GlovePIE zip file (http://www.wiili.org/index.php/GlovePIE).

==============================================

Wiimote (Nintendo Wii Remote)

If your computer has Bluetooth, you can control games with the Nintendo Wii Remote. The Wiimote has accelerometers, buttons, an infra-red camera, LEDs, vibration, a speaker, and storage for Miis. Or you can connect a Nunchuk (with joystick, two buttons, and accelerometers) or Classic Controller (with 9 digital buttons, two analog shoulder buttons, two joysticks and a D-Pad). Every feature is now supported.

Bluetooth

You need a Bluetooth adapter. Don’t pay more than $10 US for one. Cheap ones for old Bluetooth versions work fine for me. Different ones come with different Bluetooth drivers called Bluetooth Stacks. Some adapters allow you to install other stacks, some don’t. All Bluetooth stacks are very buggy. But different stacks have different bugs. Most Bluetooth stacks will now work with GlovePIE, but you may need to try the Trouble-Shooter > Bluetooth Fix menu in GlovePIE.

If you have the BlueSoleil Bluetooth Stack, you will be able to connect a Wiimote whenever GlovePIE is loaded, just by pressing the 1+2 buttons on the Wiimote at the same time. The blue B icon will turn green to show that it is connected. Note: BlueSoleil is buggy and unstable, and this feature may cause GlovePIE to stop responding for a while, or may not work sometimes, or sometimes causes BlueSoleil to beep the PC speaker. If this feature causes problems for you, you can disable it by checking the TroubleShooter > No Auto-Bluetooth Connect menu in GlovePIE.

If you have BlueSoleil, you will also be able to show the BlueSoleil window by choosing the CP-Settings > Bluetooth menu in GlovePIE.

If you don’t have BlueSoleil, then those two features won’t work, and you will need to start your Bluetooth program manually, and connect the Wiimotes manually. You need to start the Bluetooth program, hold down the 1+2 buttons on the Wiimote, tell your Bluetooth program to search for devices, then when it finds a device either tell it to search for services and connect to the HID service, or tell it to connect. There is no PIN (as far as we know) so tell your Bluetooth program to skip the PIN step, or not to use a PIN. Only after it is completely connected, can you release the 1+2 buttons.

You can disconnect a Wiimote at any time, whatever Bluetooth Stack you use, by holding down the Wiimote’s power button. GlovePIE does not have to be running for that.

Only connected Wiimotes will show up in GlovePIE.
Using the Wiimote in GlovePIE

You can use the GUI to assign the Wiimote’s actions, or you can use scripting with the “Wiimote” object. The GUI has an automatically detect input feature which will work with the Wiimote or keyboard, or mouse, or joystick, or P5 Glove. The GUI is easy, but it can’t edit or create anything too complex. You should use scripting to make fancier scripts.

GlovePIE supports up to 8 Wiimotes (in theory) although only 7 per adapter. Sometimes there will be phantom Wiimotes that show up in GlovePIE but aren’t really there.

In scripts you can access Wiimotes with “Wiimote” followed the number of the Wiimote, followed by a dot, then followed by the Wiimote property you want to use. eg.
Wiimote1.Home
If you leave out the number then it will assume you mean Wiimote1.

Buttons

The Wiimote has the following buttons, which are either true or false:

Up, Down, Left, Right,
A, B,
Minus, Home, Plus
One, Two

The Power button, and the Sync button (hidden under the battery compartment) aren’t implemented yet.

Motion Sensing

The Wiimote also has three accelerometers which measure forces/accelerations, in each of the three dimensions. Force and Acceleration are basically the same thing, since the mass is constant (F=ma). In GlovePIE they are labelled like this: X is to the right (of the wiimote), Y is up (from the wiimote buttons), and Z is forwards (or wherever the wiimote is pointing). This is the left-handed Direct3D system.

If you drop a Wiimote and it is in freefall, all the forces will be zero. BUT if you are holding a wiimote up so that it doesn’t fall, then you are pushing it upwards with a force equal to the force of gravity, but in the upwards direction. If you hold the Wiimote flat then the force holding it up will be in the +Y direction, but if you point the Wiimote at the ceiling then the force holding it up will be in the +Z direction. This helps us measure the Wiimotes tilt as well as its acceleration.

There are three values that you should never use anymore, because they are obsolete and only for backwards compatibility: RawForceX, RawForceY, and RawForceZ. If your script is currently using them, then replace them with the calibrated force values below. Remember
the calibrated force values don’t need an offset, and they are about 1/30 the size of the RawForce values.

There are three calibrated force values:

gx, gy, and gz

They can be zero for no force/acceleration, 1 for the same amount of force/acceleration as gravity, or any value in between or outside that range. There is also the vector version:

\( \mathbf{g} \)

The accelerations above are effectively measured in Gs. You can use them for measuring actual acceleration, or for measuring tilt.

If you prefer acceleration values in other units, you can use these values:

RawAccX, RawAccY, and RawAccZ.

They are measured in Metres Per Second Per Second. But GlovePIE supports units so you can do things like this:

RawAccX in miles per hour per second

to get the value in different units.

The RawAcc values still include gravity.

If you want to get the real acceleration without gravity then you should use the following values instead:

RelAccX, RelAccY, and RelAccZ

They are not very accurate because it is hard for GlovePIE to separate out the force of gravity, but it makes a good attempt.

All these acceleration values are relative to the Wiimote’s orientation. So Z is wherever the Wiimote is pointing, not necessarily a fixed direction.

**Rotations**

The Wiimote doesn’t contain gyros (BOO!!), so it has no way of determining the yaw rotation without using an imitation sensor bar. But it can sort-of tell which way is down, based on the force of gravity. This allows it to measure pitch and roll rotations. You can use the following values:

Pitch, Roll

SmoothPitch, SmoothRoll
Excerpt from GlovePIE 0.29 Documentation

The angles are all in degrees.

Pitch is the vertical angle. It is 90 when the wiimote is pointing at the ceiling, -90 when it is pointing at the ground, 0 when it is flat, and a value in between for angles in between.

Roll is how much the top is tilted towards the right. It is -180 or +180 when the Wiimote is upside-down, -90 when it is on its left side, +90 when it is on its right side, and 0 when it is flat.

Pitch and Roll attempt to filter out accelerations. This may make them slightly jagged. You can use SmoothPitch and SmoothRoll if you don’t mind getting the wrong result if the wiimote moves, but you do want it to be smooth.

There is also a rotation matrix, for those who prefer it. It is a 3x3 Direct3D style rotation matrix:

RotMat

Sensor Bar

The sensor bar is just a bunch of Infra Red lights which are always on. You can make your own fake sensor bar with candles, Christmas tree lights, or Infra-Red remote controls with a button held down. Or you can order a wireless sensor bar off the internet, or you can build your own.

You can read the position of the infra-red dots that the Wiimote can see with:

wiimote.dot1x, wiimote.dot1y

...wiimote.dot4x, wiimote.dot4y

You can tell whether an infra-red dot can be seen with Wiimote.dot1vis to Wiimote.dot4vis

You can tell the size of a dot (between 0 and 15) with Wiimote.dot1size to Wiimote.dot4size

The size will always be reported as 16 if your script uses the Nunchuk or Classic Controller, because the report needs to contain more information.

LEDs

You can set the 4 LEDs on the Wiimote by setting:

Wiimote.Leds
to a value between 0 and 15. It is binary.

Or you can set:

to either true or false individually.
**Force Feedback**

You can activate force feedback by setting:

Wiimote.Rumble
to either true or false

**Speaker**

You can play sounds on the Wiimote simply by setting these values:

Frequency, Volume

If you set Frequency to 0 it will switch the speaker off. If you set it to non-zero it will switch the speaker on again. The frequency is measured in Hz. Don’t confuse frequency with pitch, they are different in this context.

You can set the volume to a value between 0 and 2, (between 0% and 200% in other words). I don’t recommend setting it above 100% though, unless you want to go deaf.

You can also turn the speaker on and off by setting either of these values:

Speaker, Mute

Setting speaker to true switches on the speaker (slowly) and setting it to false switches off the speaker. Setting Mute to true mutes the speaker, and setting it to false un-mutes it. Using ”Mute” works faster than using “Speaker”. Note that you don’t need to turn the speaker on if you set the frequency, because setting the frequency to non-zero does it automatically.

Turning sound on and off with mute is also faster than doing it by changing the frequency to 0.

You can also set the sample rate with this variable:

SampleRate

Currently GlovePIE only supports sample rates around 3640 Hz. Sample rates must be multiples of 280 added or subtracted from that value. The default is 3640 Hz. The sample rate is also measured in Hz. The sample rate will automatically increase if you set the frequency higher than 1820 Hz. There isn’t much point in changing the sample rate.

**Nunchuck and Classic Controllers**

To tell whether a Nunchuk or classic controller is plugged in, use these values:
Wiimote.HasNunChuck
Wiimote.HasClassic
Wiimote.Expansion

The first two are true or false. The third one is a number: 0 means none, 1 don’t know what kind yet, 2 means Nunchuk, 3 means Classic Controller.

**Nunchuk**

The Nunchuk has accelerometers too! Most of the same acceleration values and rotation values are the same as the Wiimote, so see the description in the sections above. The difference is the Nunchuk accelerometers have higher precision and lower range.

The Nunchuk also has two buttons. NOTE: Unlike a playstation controller, pushing in the joystick is not a button, don’t try it. The two buttons are:

CBButton
ZButtonDown

Unlike the other controllers, you need the word “Button” on the end, because Z is the name of an axis and I’m reserving it for future axis use.

The Nunchuk also has a joystick. It uses the following values:

JoyX, JoyY

JoyX is -1 to the left, 0 in the middle, and +1 to the right.
JoyY is -1 up, 0 in the middle, and +1 down.

These are the same values used by PPJoy.

The Nunchuk does NOT have vibration, or a speaker.

**Classic Controller**

The classic controller does NOT have accelerometers or rumble, or a speaker!

The classic controller does have an unexplained lock button in the top centre, which physically opens and closes two slots in the back. The two slots can’t connect to anything and seem totally pointless. The lock button can’t be read.

You can read the following buttons:

a, b, x, y
Minus, Home, Plus
Up, Down, Left, Right
ZL, ZR
LFull, and RFull

They are either True or False.

The L button and the R button (LFull, and RFull above) also have analog versions:

L, and R

They will be between 0 and 1. Although mine won’t go below 0.03 when fully released, and yours may not reach 1 when fully pressed.

There are also two joysticks. I’m calling the one on the left Joy1, and the one on the right Joy2.

Joy1X, Joy1Y,
Joy2X, Joy2Y

They are between -1 and 1, the same as the Nunchuk joystick described above.

**Low level stuff**

You can read the data reports from the Wiimote with:

```python
deploy = Wiimote.InByteString
```

Or you can read them individually with:

```python
var.a = Wiimote.InReport
var.b = Wiimote.InByte1
```

You can send data reports to the Wiimote with this function (it has no dot):

```python
WiimoteSend(1, 0x12, 1, 0x33)
```

The first parameter of WiimoteSend is the Wiimote number you want to send it to, normally 1. The second parameter is the report number. The remaining parameters are the report payload. In this case, the report tells the Wiimote to start vibrating, and to send data via report 0x33.

The one-byte reports can also be sent another way by setting one of these values (it will only send the report when the value changes):

```python
```

You can set bytes in the Wiimote’s onboard memory (where Miis are stored) with the WiimotePoke command, and read bytes from its memory with the WiimotePeek command.

```python
WiimotePoke(1, 0x04B00033, 0x33)
var.value = WiimotePeek(1, 0x04B00033)
```
The first parameter is optional, and it is the Wiimote number. The second parameter is the address. The third parameter is the value you want to set it to (between 0 and 255).

Note that the Nunchuk and Classic Controller have their own memory space which is mapped to the Wiimote’s when they are plugged in.

Don’t use these low level functions unless you know what you are doing and have read the www.wiili.org/Wiimote page. One person has bricked their nunchuk by experimenting with writing to all its addresses (although that is how we discovered how the nunchuk works).

**Multiple Wiimotes**

You can tell how many Wiimotes there are with Wiimote.Count

You can access a particular wiimote by putting a number after the word “wiimote” and before the dot. For example:

Enter = wiimote2.A

You can tell whether a particular Wiimote exists with Wiimote.Exists

You can’t read the Serial number (as text) anymore with Wiimote.Serial, because I disabled it. It rebooted computers with BlueSoleil when you tried to read the serial.

Don’t forget to sync both Wiimotes with the PC before you try to use them.