WIIMOTE IN RICH INTERNET APPLICATIONS

BACHELOR PROJECT

Timen Olthof
tpolthof@few.vu.nl
1553909

june 2008, VU Amsterdam
supervisor: Anton Eliëns
co-supervisor: dr. N. Silvis

Contents

1 Introduction

The Wiimote is a revolutionary game console controller made by Nintendo. It is normally used to control games on the Nintendo Wii game console. But since it uses Bluetooth to wirelessly connect to the Wii, it can also be connected to a normal PC. It can then be used to control Windows applications, but it is also possible to use it as an input device in Flash/Flex/ActionScript3 projects.

ActionScript3 allows one to program vector animations in a relatively easy way, but interacting with these animations using the mouse may not be very intuitive in some cases (for example when viewing or rotating a 3D representation of some prototype, or when 'throwing' a ball around in a game). In these situations, using the Wiimote to rotate the 3D model or throw the ball would most likely feel much more natural.

The goal of this project is to use the Wiimote to control or interact with ActionScript3 animations,and, more generally, Flex/ActionScript3 applications. First, I will introduce the Wiimote by showing its main features and functionalities. Then I will describe the Flex 3 SDK and ActionScript3, which are the main programming tools I used. Next, I will explain how realistic looking animations can be programmed in ActionScript3 using (approximations of) the laws of physics, discuss the ways in which the Wiimote can be used in practice to interact with ActionScript3 animations, and indicate some example applications, including an actionpaint program and an interactive tutorial on philosophy. Finally I will draw conclusions about the use of the Wiimote in ActionScript3 applications.

The appendix of this document contains a step-by-step practical guide on using the Wiimote in ActionScript3.

2 The Nintendo Wiimote

While Microsofts XBOX360 and Sony's Playstation3 are shipped with controllers that are much like their predecessors' constrollers, the Wiimote that is used to control the Nintendo Wii game console system is totaly unlike the controller of Nintendo's previous game system, the Nintendo Gamecube.

Hardware

The Wiimote is much more than an ordinary game controller. In addition to the standard array of buttons it also features an infrared sensor, an internal motion sensor, a speaker, four leds and a rumble motor. The layout of the Wiimote can be seen in figure 1:

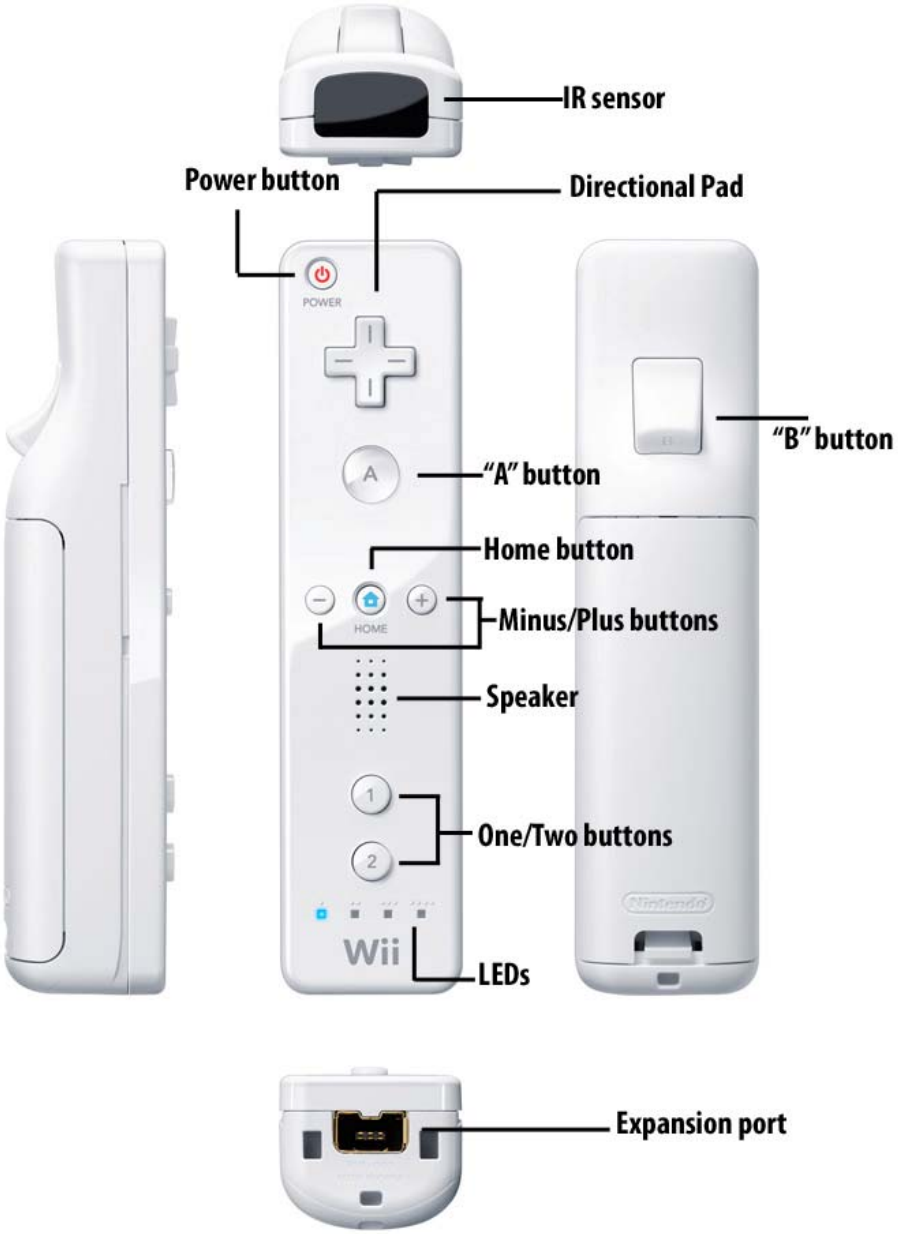

Figure 1: The Wiimote Layout

BUTTONS

There are twelve buttons placed on the Wiimote, one of which is the *power* button, that is used to turn the Wiimote off. The *directional pad* is in fact a collection of four buttons, that are most often used to move up, down, left or right. The most logical use of the *minus* and *plus* buttons is to zoom in or out. The last four buttons are the *one*, *two*, *A* and *B* buttons, which are most often used to perform an in-game action like jumping, attacking or triggering an event. The *B* button is placed at the bottom of the Wiimote so that it can be used as a shooting trigger.

INFRARED SENSOR

One of the most interesting innovations of the Wiimote is the embedded infrared sensor. This sensor can track up to four infrared light sources. There are two main methods of using the IR sensor.

The *normal* method would be to track the movement of the Wiimote based on an IR light source with a fixed position. This will require some converting calculations in the code, because the sensor measures the position of the light source relative to the Wiimote, while the value that is most often needed is the position of the Wiimote relative to the light source. When using this method, a Wii Sensor Bar[1] can be used as an IR light source.

Alternatively, the Wiimote can be placed at a fixed location enabling the tracking of the movement of up to four IR light sources. This method could be called the *inverted* IR tracking method. This method can be used to create for example an electronic whiteboard or head-tracking functionality, as has been shown by Johnny Chung Lee[2]

MOTION SENSOR

The Wiimote features a motion sensor that enables games to use the acceleration of the Wiimote to control software. It provides motion sensing on three axis, allowing the use of any movement the user as input values to influence animations or applications.

SPEAKER

The Wiimote contains a simple speaker to play back audio. I didn't use the speaker in this project.

LEDS

Four leds are used to indicate the status of the Wiimote. These leds are used both to provide identification in case multiple Wiimotes are used, as well as to indicate that the Wiimote is connecting to a host.

RUMBLE

The Wiimote includes rumble support by means of an internal motor that can generate vibrations that can be used to give the user force feedback on his actions or events that occur in the application.

---

[1] A Wii Sensor Bar is basicly a plastic bar that contains IR leds and batteries to power them. Multiple versions are available, both from Nintendo as well as from third party companies.

[2] `http://www.cs.cmu.edu/~johnny/projects/wii/`

Software

The Wiimote connects to a host (Wii or PC) using the Bluetooth protocol, so to connect the Wiimote to a PC, a Bluetooth chipset is required on the PC, either on the motherboard or on an external USB dongle. In this project, the Bluesoleil[3] Bluetooth driver is used to pair the Wiimote and the PC. Most Bluetooth chipsets are shipped (or are compatible) with the Bluesoleil driver software.

The WiiFlash[4] software is subsequently used to communicate between Flash and the Wiimote. The WiiFlash consists of a WiiFlash Server (written in C++/.NET) that handles the communication with the Wiimote, and a WiiFlash ActionScript API that can be used in Flex/ActionScript3 applications. Section 5 describes the use of the WiiFlash ActionScript API in more detail.

[3] http://www.bluesoleil.com/

[4] http://www.wiiflash.org

3 The Flex 3 SDK and ActionScript3

Adobe Flex is "a cross-platform development framework for creating Rich Internet Applications (RIAs)"[5]. It is a collection of multiple programming languages, applications such as a compiler, and other services. At the core of the Flex framework are the MXML markup language and the ActionScript3 programming language that is well known from the Adobe Flash product line. With the Adobe Flex Builder software, Adobe provides a complete IDE to build, debug and test applications. However, Adobe Flex Builder is not free[6]. Alternatively, the free Adobe Flex compiler can be used to compile Flex applications, because Flex applications can be built using any text editor. The Flex Builder IDE however features numerous coding and debugging advantages, so using the Adobe Flex Builder IDE is highly recommended. The Adobe Flex Builder IDE provides three main methods to create applications:

ActionScript3 ActionScript started out as a scripting language used for controlling simple 2D Flash animations, but the most recent version (ActionScript3) that is used with Flex is as close to Java as you can get, without actually being Java. It is a complete programming language that allows you to do pretty much everything you would like to do when building a Flex application.

MXML MXML is a markup language based on XML standards that can be used to define an application in an ordered, nested fashion. The <mx:Application/>-tag is used for the main application, and other components and services (such as Panels, Buttons, Transitions) and data structures (like Arrays or Strings) can be embedded within the application tag.

Design The Flex Builder IDE also features a 'design mode' that can be used to create graphical user interfaces using drag-and-drop components and setting their property fields.

Flex applications are compiled to .swf files that can be executed in the Flash Player 9 Virtual Machine on the client. The installed base of the Flash Player 9 is very large, so this plugin requirement should not be a big problem. Just-in-time compilation at the server side is also supported. In that case, the application is automatically recompiled whenever it or any of its assets change.

[5] `http://www.adobe.com/go/flex/`

[6] However, a free trial version of the Adobe Flex Builder IDE is available from the Adobe Flex website and the full version of Flex Builder is also available for free for educational purposes only.

4 Physics-driven animations

ActionScript3 provides everything that is needed to animate objects. Objects on the screen (*stage*) can be moved by changing their `x` and make `y` properties. But to move objects realistically, as if gravity is pulling them down for example, the laws of nature need to be modelled. In this section I will discuss the basics of physics-driven animations in ActionScript3. For an in-depth guidebook to ActionScript3 animation see the book *ActionScript3 Animation: making things move* by Keith Peters[7].

BASIC ANIMATION

In ActionScript3 (and in other Object Oriented programming languages) this can be done very well-organized, by having each (screen) object keep track of it's own *position* (`x`, `y`), *speed* (`vx`, `vy`) and *acceleration* (`ay`, `ay`)[8]:

```
object:Object = new Object();
object.x = 100;
object.y = 100;
object.vx = 0;
object.vy = 0;
object.ax = 3;
object.ay = 3;
```

Movement is then created by first updating the speed of the object based on the accelleration of the object, and then updating the object's x and y position based on the speed of the object:

```
object.vx = object.vx + object.ax;
object.vy = object.vy + object.ay;
object.x = object.x + object.vx;
object.y = object.y + object.vy;
```

These values need to be updated continuously to create motion in time. Most often this is done n times per second, where n is the framerate of the animation.

INTERACTION

With basic animation, the object only influences itself. Of course it is also possible to have objects interact with each other. Objects may repulse or attract each other, or may collide with each other. Those cases can be handled by adjusting *location*, *speed* and *acceleration* properties based on for example the sizes of the objects and the angle of impact (in case of a collision), or based on the distance between objects (in the case of attraction or repulsion).

The user may influence objects too, for example by picking up a ball and throwing it away. Or an object (for example a cat) may 'follow' the user's mouse, by basing it's own position on the position of the mouse pointer.

There may also be other 'forces' that influence objects, for example gravity that pulls objects down, and friction that slows down movement.

Finally, multiple objects that form a greater whole (for example parts of a leg) may interact with each other to create a natural movement.

---

[7] *ActionScript3 Animation: making things move* by Keith Peters, FriendsofED (Appress), 2007

[8] The properties need to be measured on both the x and y axis because we are working with two dimensional movement. Threedimensional movement would require additional properties (z, vz, az) to measure movement on the z axis.

5 Using the Wiimote in Flex/AS3 applications

The WiiFlash ActionScript API is the Flex/AS3 part of the WiiFlash software. It consists of a set of ActionScript3 classes that model the Wiimote and its functionallity. To use the Wiimote in a Flex/AS3 application a little programming has to be done: first, a `Wiimote` object has to be created:

```
var myWiimote:Wiimote = new Wiimote();
```

Then, the Wiimote object needs to connect to the real-life Wiimote:

```
myWiimote.connect();
```

Internally, a socket connection wil be opened to the WiiFlash server (that needs to be running at localhost).
Finally eventListeners need to be added to the Wiimote object:

```
myWiimote.addEventListener(WiimoteEvent.UPDATE, onUpdated);
myWiimote.addEventListener(ButtonEvent.A_PRESS, onAPressed);
myWiimote.addEventListener(ButtonEvent.A_RELEASE, onAReleased);
```

In this example, the `onAPressed()` method will be called when the A button is pressed, `onAReleased()` will be executed when the A button is released, and `onUpdated()` will be called each time the Wiimote's internal sensors supply new values from the Wiimote's sensors.

At any time the data that the Wiimote's sensors supply can be read out from the Wiimore object. Data that can be read out includes the state of all buttons (whether they are currently pressed or not), the data from the motion sensor, the number of ir points that are detected (and the locations of these points) and the battery level. It is also possible to make the Wiimote rumble by setting a rumbletimeout:

```
//is the a button pressed?
if (myWiimote.a) {}

//what is the battery level?
myWiimote.batteryLevel

//is the first ir point source detected?
if (myWiimote.ir.p1) {
  //then get its y coordinate and use
  //it to position the cursor     cursor.y = myWiimote.ir.y1;
}

//rumble for 300 milliseconds
myWiimote.rumbleTimeout = 300;
```

6 Sample applicatications for the Wiimote

The Wiimote can be used to interact with animations in a lot of different ways. I created a set of example applications to show possible Wiimote application interaction. The named examples can be downloaded from the project website[9].

POINTING DEVICE

Because of its IR sensing capabilities, the most obvious method of using the Wiimote to interact with animations is to use the Wiimote as pointing device. It may simply replace the mouse cursor, but may also have a more specific context based function. The Wiimote may for example be used to target game objects or as a digital 'flashlight', enlightening only the part of the scene at which de Wiimote is aimed. The Wiimote may then also be used to trigger actions while aiming.

Fireworks In the Fireworks sample application, the Wiimote is used as a pointing device to aim for a fireworks explosion.

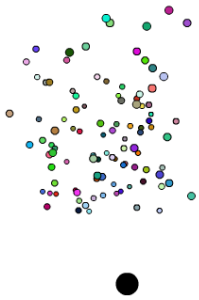

Figure 2: The Fireworks sample application

MOVEMENT

The Wiimote may also be used to move a game object, for example the player's avatar, or other objects that can be manipulated by the user, for example apples that drop from a tree and need to be put in a basket. These movement actions can be controlled intuitively with either the IR sensor, or the motion sensor. Of course the directional pad can also be used to control movement.

Trees / Trees2 In the Trees and Trees2 sample applications, the Wiimote is used to control movement through a tree forest.

Figure 3: The Trees sample application

INFLUENCE

The position, direction or movement of the Wiimote may be used to influence

[9] http://www.few.vu.nl/~tpolthof/wiimote/

animation objects. The Wiimote can be used to control the speed of thrown objects, the direction of moving objects or for example distribution of objects on the screen.

Fountain In the Fountain sample application, the Wiimote is used to influence the speed and direction of a fountain of particles.

Figure 4: The Fountain sample application

VIEW/ZOOMING/ROTATION

The Wiimote can be used to control the users view on the screen, and provide ways to alter the view by panning and zooming. Both the IR sensor and the motion sensor are very useful to control rotation of objects. This can of course be either 2D or 3D rotation.

Rotation1 In the Rotation1 sample application, the Wiimote is used to rotate a set of interconnected dots.

Cube In the Cube sample application, the Wiimote is used to rotate a Cube.

Rotation3 In the Rotation3 sample application, the Wiimote is used to rotate and zoom in to a set of red dots.

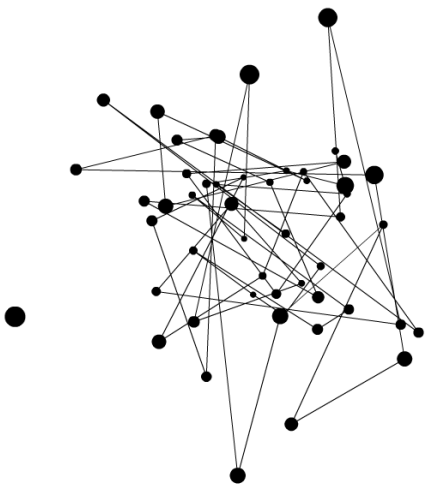

Figure 5: The Rotation1 sample application

FORCE FEEDBACK

Because of the Wiimote's internal motor, the Wiimote can be used to provide the user with force feedback. The Wiimote may for example rumble when the user hits a wall in a maze or when objects collide. Force feedback is implemented in a few of my example applications.

ARTISTIC/CREATIVE USE

The Wiimote can be used in creative or artistic projects. The Wiimote may for example function as a digital pencil or brush.

ActionPaint In the ActionPaint sample application, the Wiimote is used to paint in the style of Jackson Pollock[10].

Figure 6: The Wiimote Action Paint sample application

OTHER USES

There are many other uses for the Wiimote. The Wiimote could for example be used to control a slideshow during a presentation, or to interact with an interactive educational application, for example an interactive tutorial on philosophy[11]:

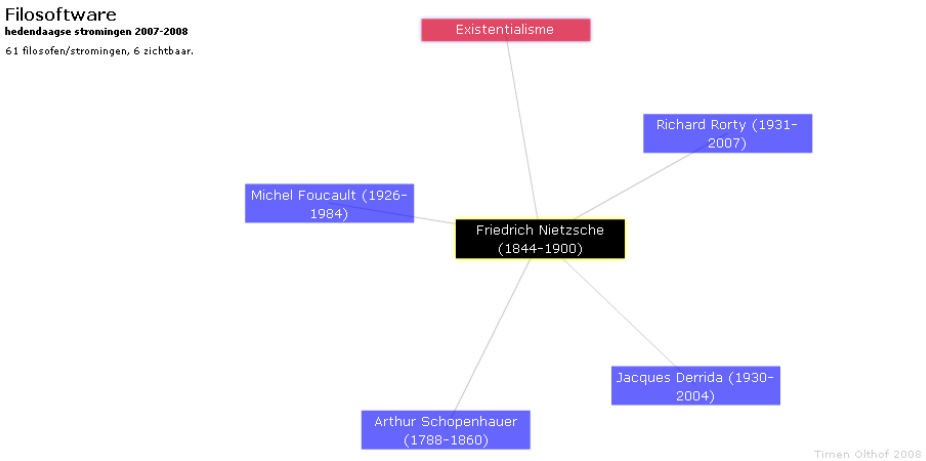

Figure 7: The Filosoftware interactive tutorial on philosophy

[10]Jackson Pollock (1912-1956): American abstract expessionist painter known for his action painting style.

[11]http://www.few.vu.nl/~tpolthof/filosoftware/

7 Conclusion

In this document I have described how the Wiimote can be used to interact with Flex/AcrionScript3 animations and applications. ActionScript3 proved to be very well equiped to be used for animation programming, and the extensible nature of the Flex framework allow for it to be used in conjunction with all kinds of technologies. The WiiFlash software is a great example of such a synthesis between ActionScript and a new technology. In practice the WiiFlash software has proven very workable. While the WiiFlash server does have some stability issues now and then, it generally works great without big problems.

However, using the Wiimote for online web applications is at this time not ideal, because of the fact that the WiiFlash software needs a WiiFlash Server external to the ActionScript3 application. An online visitor that would like to use the Wiimote would have to start this server by hand . In the future, if the boundary between web appications and desktop applications keeps fading away like it currently does, technologies like Adobe AIR may help to solve this problem. If future versions of the WiiFlash software would integrate the Bluetooth communication into the ActionScript3 API itself, this would of course also solve this problem.

The innovative design and interesting functionalities of the Wiimote provide opportunities to create new types of human computer interaction in Flex/AS3 applications. While there is definitely room for improvement in the WiiFlash software, the Wiimote technology does have a lot of potential, and is in most cases already very useful. In the meantime Nintendo has released yet another innovative game controller for the Wii game console system: the Wii Balance Board. This is a board on which you can stand, and use your body movement and weight to interact with Wii games.

A Practical guide to using the Wiimote in AS3

This is a step-by-step guide to using the Wiimote in AcrionScript3.

Step Zero: Requirements

To use the Wiimote in ActionScript 3 the following hard- and software is needed:

- a Wiimote
- a computer running Windows
- the .NET framework v3.0 or higher
- a Bluetooth chipset and driver[12]
- the WiiFlash software
- an ActionScript3 programming enviroment[13]
- a Wii Sensor Bar or another infrared light source
  (only needed in case you want to use the IR functionality)

Step One: Preparation

First, .NET, Bluesoleil and the Adobe Flex Builder need to be installed. This software can be obtained from:

.NET framework v3.0 or higher : `http://www.microsoft.com/downloads/details.aspx?FamilyID=10CC340B-F857-4A14-83F5-25634C3BF043\&displaylang=en`

Bluesoleil : `http://www.bluesoleil.com/`

WiiFlash : `http://www.wiiflash.org/`

Adobe Flex Builder : `http://www.adobe.com/go/flex_trial`[14]

Download and install these applications if they are not already installed. WiiFlash requires no installation. Please unzip the WiiFlash archive so that it is ready for later use. We will call the directory that you unzipped WiiFlash to `%WIIFLASHDIR%`.

Figure 8: The required software applications

[12] this guide uses Bluesoleil, because most chipsets support this driver
[13] this guide uses Adobe Flex Builder
[14] the full version of Flex Builder is also available for free for educational purposes

Step Two: Starting Bluetooth

When the software has been installed and is ready for use, it is time to start the Bluetooth service. If the Bluetooth chipset is on an external (usb) device, plug it in first. Then start the Bluesoleil software by doubleclicking its shortcut on the desktop or launch it from the Start menu.

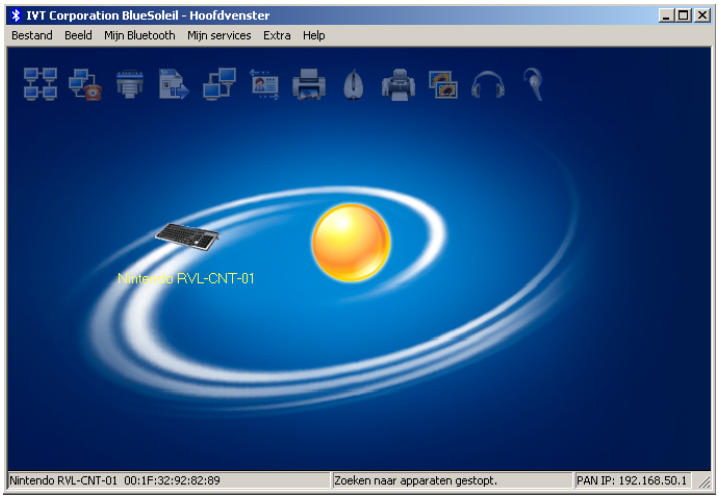

Figure 9: Bluesoleil has been started

Step Three: Connecting the Wiimote

To establish a connection between the Wiimote and the PC, they need to be paired. Simultaneously press the *one* and *two* buttons on the Wiimote. The four blue leds on the Wiimote should start blinking. Click the orange ball in the center of the Bluesoleil software to start searching for devices. The Wiimote should show up around the ball. If a lot of devices are detected, scroll to find the Wiimote in the list. Doubleclick on the Wiimote device in Bluesoleil to probe for its services. Then right click on the Wiimote and choose “Connect” → “Bluetooth Human Interface Device Service”. The first time you pair the Wiimote to a PC, drivers might need to be installed. This should happen automatically.

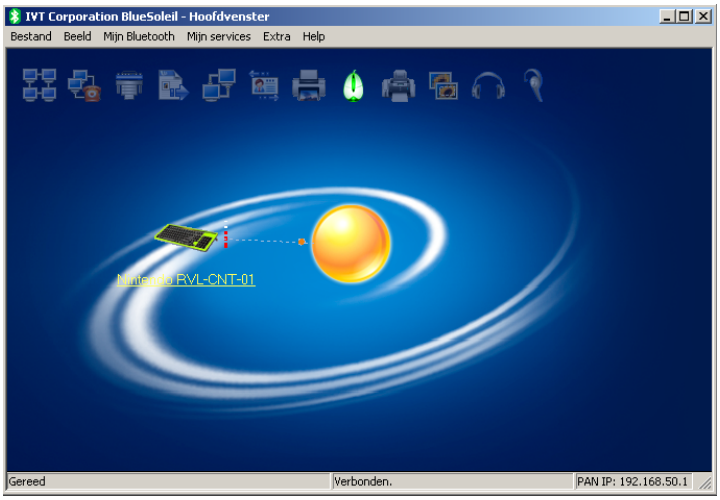

Figure 10: The Wiimote has been paired with the PC

Step Four: Starting the WiiFlash Server

Now that the Wiimote and PC have been paired, it is time to start the WiiFlash server by doubleclicking its icon in the `%WIIFLASHDIR%`. The Wiimote should start rumbling for a short period of time to indicate that it is connected to the WiiFlash server. The WiiFlash server will show itself, indicating the number of connected Wiimotes:

Figure 11: The WiiFlash server showing that one Wiimote is connected

Step Five: Testing

The WiiFlash software comes with a few example applications that can be used to test if everything works correctly. Browse to the `%WIIFLASHDIR%\cs3-examples` folder. Doubleclick the `Wiimote Demo.swf` to start the demo application:

Figure 12: The WiiFlash demo application

The demo application should indicate which buttons are pressed, and show the values that the Wiimote's sensors report.

Step Six: Programming

To use the Wiimote in your own applications, either Adobe Flex Builder, Adobe Flash or the standalone Flex SDK compiler can be used. In this example we use Flex Builder. Before the classes can be used in an application, the `org` folder in the `%WIIFLASHDIR%\api\source-classes` directory needs to be copied to the project path of your application. Also, for some projects, it might be necessary to link the `WiiFlash.swc` Flash Component file from the `%WIIFLASHDIR%\api\swc` directory to your project (File → Properties → 'Flex/ActionScript build path' → 'library path' tab → 'Add SWC...').

The WiiFlash ActionScript3 API centers around the `Wiimote` class that is used to represent a real-life Wiimote in the ActionScript3 code. The readings of the IR sensor can be obtained through the `IR` class, that can be accessed via the `ir` property of an `Wiimote` object. Other important WiiFlash API classes are the `WiimoteEvent` and `ButtonEvent` that are used to model the Events that are dispatched when using the Wiimote. Detailed documentation about the usage of the WiiFlash API classes is available in the `%WIIFLASHDIR%\wiiflash-docs` directory.

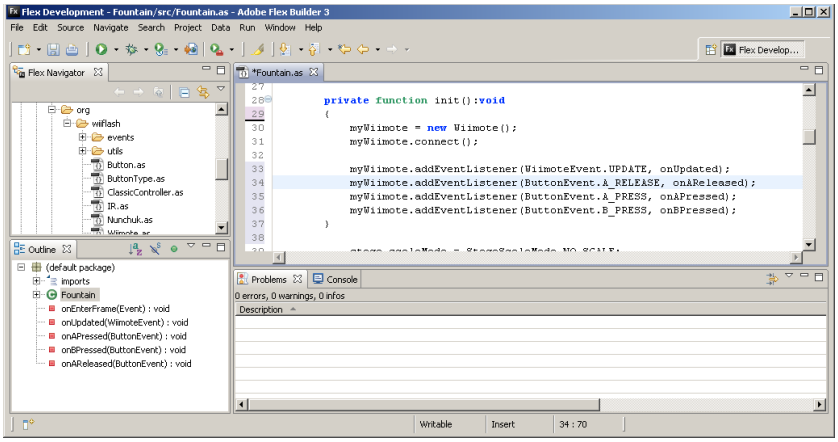

Figure 13: The Flex Builder GUI

For the basics of using of the Wiimote in Flex/ActionScript3 projects, see section 5 ("Using the Wiimote in Flex/AS3 applications") of the main document.

Using the IR sensor

To use the Wiimote's IR sensor, an infrared light source is required. While a candle might do the job, a more stable solution is using IR leds or a so called Wii sensor bar. A Wii sensor bar is a plastic bar containing IR leds, batteries to power them, an on/off switch and (normal light) status leds to indicate whether the sensor bar is powered on or not. These status leds are very useful, because IR light is invisible to the human eye:

Figure 14: Two Wii Sensor bars

Another option is to use an IR led array or some individual IR leds. This of course requires some soldering and/or fiddling with power sources:

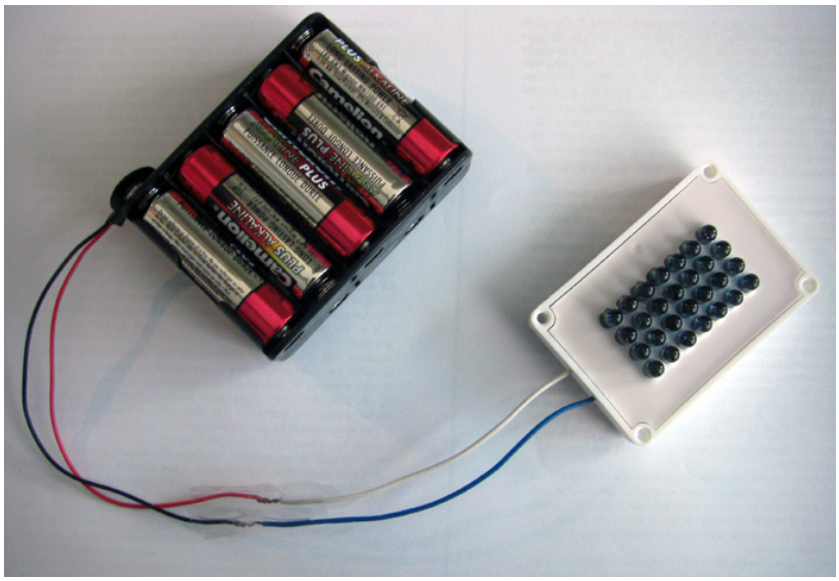

Figure 15: An IR-led array with power source

In my experience, the best option is to just use a sensor bar. The are cheap and they just work out-of-the-box. Using an IR led array or individual IR leds is only recommended if you have the time to tweak them, or if you like fiddling with electronics. Of course, individual leds can be used to create personalised control accesoires, like an IR LED pen (to use as pointer), an IR led cap for use with headtracking or for example a glove containing IR leds that can be used to interact with an application using hand gestures.