Escola Superior de Tecnologia e de Gestão
INSTITUTO POLITÉCNICO DE BRAGANÇA

Joaquim José de Carvalho Paulino

Expansão de Aplicações para Wiimote

Curso Informática de Gestão

Julho 2008

Expansão de Aplicações para Wiimote

Relatório da Disciplina de Projecto
Curso de Informática de Gestão
Escola Superior de Tecnologia e de Gestão

Joaquim Paulino

A Escola Superior de Tecnologia e Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Certifico que li este relatório e que na minha opinião, é adequado no seu conteúdo e forma como demonstrador do trabalho desenvolvido no âmbito da disciplina de Projecto.

_____

<Nome do Orientador> Orientador

Certifico que li este relatório e que na minha opinião, é adequado no seu conteúdo e forma como demonstrador do trabalho desenvolvido no âmbito da disciplina de Projecto.

_____

<Nome do Co-Orientador> Co-Orientador

Certifico que li este relatório e que na mmminha opinião, é adequado no seu conteúdo e forma como demonstrador do trabalho desenvolvido no âmbito da disciplina de Projecto.

_____

<Nome do Arguente> Arguente

Aceite para avaliação da disciplina de Projecto

Prefácio

O Wii, da Nintendo, destacou-se como a consola de videojogos que tirou as pessoas do sofá para as pôr a participar em jogos virtuais de ténis e futebol, tudo isto graças a um controlo remoto sensível ao movimento, o Wiimote.
A interacção homem máquina estabelece uma revolução com o crescente número de dispositivos que destroem os paradigmas actuais. O principal objectivo foi realizar um projecto algo diferente das práticas habitualmente propostas, aproveitar para investigar e explorar um pouco as características técnicas do Wiimote e investigar possíveis usos alternativos aos jogos e as suas possíveis aplicações noutros contextos. Neste sentido, este trabalho propõe a análise de três aplicações[1] que utilizam o Wiimote e a investigação de novas extensões destas mesmas aplicações sugerindo, produzindo e testando novas abordagens e estruturas de informação.

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

Agradecimentos

Agradeço ao Professor Pedro João Rodrigues por me ter proposto o tema que serviu de base à realização deste trabalho.

Estou particularmente grato pela disponibilidade demonstrada no acompanhamento das diversas fases deste projecto, fornecendo toda a informação e esclarecimentos necessários.

Ainda pelo facto de me ter facultado o material necessário para uma boa elaboração do mesmo.

Índice

1 Introdução 1

1.1 Enquadramento 1

1.2 Motivação 2

1.3 Objectivos 2

1.4 Estrutura do Relatório 2

2 Contextualização do Problema 3

2.1 Dispositivo Multi-Apontador em Quadro Branco Padrão 4

2.2 Definição dos Recursos e Ferramentas Utilizadas 4

2.2.1 Software Necessário 4

2.2.2 Hardware Necessário: 5

2.3 Seguidor Posição em Ambientes Imersivos 2D com Analogia de Percepção em 3D .. 6

2.4 Definição dos Recursos e Ferramentas Utilizadas 7

2.4.1 Software Necessário 7

2.4.2 Hardware Necessário 7

2.5 Controlo de Cursor Através dos Movimentos dos Dedos (Tipo Minority Report). 8

2.6 Definição dos Recursos e Ferramentas Utilizadas 8

2.6.1 Software Necessário 8

2.6.2 Hardware Necessário 8

3 Projecto e Desenvolvimento 10

3.1 Análise 10

3.2 A Aplicação WiimoteWhiteboard 11

3.2.1 Como Iniciar a Aplicação 11

3.2.2 Comunicação Wiimote-Computador Através de Bluetooth 11

3.2.3 O Programa BlueSoleil 11

3.2.4 Verificar a Compatibilidade do Adaptador Bluetooth 12

3.2.5 O Ficheiro WiimoteWhiteboard 13

3.3 Possiveis Erros e Soluções 14

3.4 Considerações Prévias 15

3.5 Limitações do WiimoteWhiteboard 16

3.5.1 Sombras na Superfície de Projecção 16

3.5.2 Emulação do Botão Direito do Rato 16

3.5.3 A Qualidade do Traço 16
3.5.4 Compatibilidade com Algumas Aplicações Windows 17
3.6 Requisitos 17
3.6.1 Requisitos Funcionais 17
3.7 Desenvolvimento 17
3.8 Modificações Introduzidas no WiimoteWhiteboard 18
3.9 Nova Versão para o WiimoteWhiteboard Utilizando Nova Caneta com Processador Digital 21
3.9.1 Novo Layout 22
3.10 Modificações na Nova Versão WiimoteWhiteboard 22
3.11 A Aplicação Desktop Vr e a Aplicação Wiimote Multipoint Grid 27
3.11.1 Testes Efectuados 27

4 Análise de Resultados 28

4.1.1 Análise de Desempenho e Resultados do Sistema 28
4.1.2 Comparação com Outros Sistemas 29
4.1.3 Aplicação Prática em Ambiente de Sala de Aula 29
4.1.4 Avaliação Geral 30

5 Conclusões 31

A Detalhes do Ficheiro Xml 32

A.1 Aspecto Geral Versão Inicial 32
A.2 Aspecto Geral da Nova Versão 32

B Programação do Chip 16F628A para Nova Versão da Caneta Digital 33

B.1 Aspecto Geral 33

6 Bibliografia 36

7 Anexos 37

7.1 Características LED IR Tsal 6200 e 6100 38
7.2 Características LED SFH 4231 39
7.3 Código Fonte da Aplicação WiimoteWhiteboard 40
7.4 Modificações ao Código Fonte Inicial 41
7.5 Diagrama do Chip 16F628A 42
7.6 Código Fonte Nova Versão para o WiimoteWhiteboard 43

Lista de Figuras

Figura 1 – Layout do Programa WiimoteWhiteboard 4

Figura 2 – Comando Wiimote 5

Figura 3 – Adaptador USB Bluetooth 5

Figura 4 – LED Emissor TSAL 6200 5

Figura 5 – Botão de Pressão. 5

Figura 6 – Resistência para Prevenir que Acima de 1,5 Volts o LED não se Danifique. 6

Figura 7 – Cabo Multifilar para Efectuar Ligações. 6

Figura 8 – Esquema para Construir uma Caneta Digital. 6

Figura 9 – Ligação das Componentes na Construção de uma Caneta 6

Figura 10 – LED Emissor Infravermelho SFH 4231 [OSRAM] 9

Figura 11 – Matriz de LEDs. 9

Figura 12 – Alimentador da Matriz de LEDs. 9

Figura 13 – Aspecto Geral da Aplicação BlueSleil 11

Figura 14 – Aplicação BlueSoleil Compatibilidade com Adaptador Bluetooth 12

Figura 15 – Aplicação BlueSoleil Quando da Incompatibilidade da Aplicação ao Adaptador 13

Figura 16 – Ficheiro de Arranque do WiimoteWhiteboard 13

Figura 17 – Layout do Wiimote WhiteBoard 14

Figura 18 – Caneta Digital com Identificação da Função dos Botões 22

Figura 19 – Layout da Nova Versão do WiimoteWhiteboard 22

Capítulo 1

1 Introdução

1.1 Enquadramento

O Wiimote, um pequeno comando sem fios que é fornecido com a consola Wii da Nintendo, veio revolucionar a forma de interagir entre o utilizador e a máquina. Este comando integra uma câmara de infravermelhos que possibilita fazer o seguimento posicional do Wiimote ou de um dispositivo de infravermelho de forma precisa.

É este último dispositivo que facilita a utilização deste comando em aplicações diferentes das originalmente criadas, as de interacção em jogos.

Este projecto passa pela análise, implementação e melhoria das três aplicações propostas por Johnny Chung Lee[2] que utilizam o Wiimote, são elas: Dispositivo multi-apontador em quadro branco padrão; Seguidor de posição para ambientes imersivos 2D com analogia de percepção em 3D; Controlo de cursor através dos movimentos dos dedos (Tipo Minority Report).

Das três aplicações a primeira foi a que obteve a maior atenção por parte do autor deste projecto por ser a que traz mais benefício na utilização diária. A comunicação entre o Computador e o comando Wiimote é feito através de Bluetooth recorrendo a drivers comuns.
Pode-se classificar o projecto Wiimote como um projecto de investigação e de desenvolvimento, pela área de análise envolvida e pelos objectivos práticos a atingir.

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

1.2 Motivação

A motivação para este projecto é a de investigar e alargar as reais capacidades deste comando Wiimote no contexto do projecto de Johnny Chung Lee.

Este projecto foi criado para dar continuidade à utilização das aplicações descritas dando-lhe uma maior independência de forma a melhorar a sua utilização.

Espera-se assim melhorar e cativar a sua utilização, bem como dinamizar e optimizar o processo de modelação e partição dos módulos.

Relativamente ao módulo de quadro interactivo pretende-se para além de uma substancial redução de custos ao nível da compra de ferramentas, um importante aumento de quadros interactivos, relativamente ao orçamento original, numa escola.

1.3 Objectivos

O objectivo deste trabalho visa aumentar a autonomia das ferramentas disponibilizadas por Johnny Chung Lee, analisar essas mesmas aplicações e melhorá-las para outras de carácter útil bem como ensaiar e optimizar o processo de utilização.

Espera-se por outro lado que seja um veículo para possíveis alternativas de baixo custo na aquisição de equipamentos.

1.4 Estrutura do Relatório

Este trabalho encontra-se estruturado em cinco capítulos dos quais, o primeiro é composto por esta introdução ao trabalho.

No segundo capítulo são apresentadas as características do Wiimote e o conceito das três aplicações estudadas assim como o material necessário à realização das mesmas.

O terceiro capítulo é apresentado uma análise das aplicações, a implementação da solução proposta no capítulo anterior e as experiências efectuadas bem como os resultados obtidos

O quarto capítulo apresenta a análise aos resultados às modificações realizadas nas respectivas aplicações.

O último capítulo contém as conclusões gerais deste trabalho, analisa os seus principais resultados e apresenta algumas perspectivas de desenvolvimentos futuros.

Capítulo 2

2 Contextualização do Problema

Os controladores sem fios vendidos pela Nintendo, doravante referidos como comando *Wiimote,* possuem características impar para a realização destes projectos uma vez que possuem uma câmara de infravermelhos (IR) que podem monitorar até quatro pontos de luz em simultâneo com uma resolução de 1024x768 pixels.

O Wiimote contém dois tipos de sensores que podem ser usados para a resolução de monitoramento [**KREYLOS, 2007]**:

Acelerómetro Linear: O Wiimote contém três acelerómetros que medem a aceleração linear ao longo de três eixos ortogonais. Os acelerómetros podem ser utilizados para medir directamente o movimento angular, medindo a direcção da gravidade, são também a base para o reconhecimento do gesto Wii usado em muitos jogos. Infelizmente, os acelerómetros por si só não podem ser usados para monitorizar a (relativa) posição e orientação do Wiimote no espaço.

Câmara IR: O Wiimote está equipado com uma câmara de luz infravermelha (IR) de 1024x768 pixels montado na frente do comando. O IR do Wiimote pode detectar e monitorizar até quatro fontes de luz infravermelha modulada a 100 Hz. A câmara está equipada com um processador que analisa a imagem IR, que identifica fontes de luz e que calcula as suas posições (x, y). Uma vez que a câmara é passiva, o Wiimote precisa para trabalhar de fontes de luz infravermelha (IR). A consola Wii vem com um chamado "sensor bar", que simplesmente emite luz infravermelha de dois aglomerados de IR LEDs montados em ambas as extremidades do sensor. Os acelerómetros podem ser usados para combinadamente, com a imagem de IR, medir os movimentos angulares.

É através deste último tipo de sensor (IR) que o trabalho agora apresentado vai incidir.
O Wiimote é o controlador padrão da consola Wii, da Nintendo (mas pode ser conectado a computadores através de uma interface Bluetooth).
Ele recebe a luz (pontos de IR) de uma caneta digital e transmite ao computador os movimentos feitos pelo utilizador [**WIIlI.ORG, 2007**]. De acordo com as posições destes pontos o Wiimote calcula a posição do cursor na superfície de projecção.
Além desse sensor, o Wiimote possui botões convencionais e pode controlar um cursor.
Todos estes dados são transmitidos por Bluetooth.

Aquilo que se espera obter é explorar, adicionar e demonstrar a funcionalidade do Wiimote nos três projectos apresentados.

2.1 Dispositivo Multi-Apontador em Quadro Branco Padrão

Tendo em conta a noção central do trabalho realizado por Johnny Chung Lee, a ideia básica consiste em conseguir transformar um quadro branco padrão em quadro interactivo com o simples emprego do comando Wiimote. Dado que este comando tem as características apresentadas no ponto anterior, unicamente teremos que construir canetas de infravermelhos, instalar o software disponibilizado e teremos um quadro interactivo em qualquer superfície de projecção. Inclui-se mesmo o monitor do próprio computador (como sendo um pseudo tablet-pc) ou até mesmo sobre a superfície de uma mesa para explicar que com esta tecnologia podemos ter até quatro pontos na mesma superfície, e isto tudo é conseguido a muito baixo custo, aproximadamente 60€.

2.2 Definição dos Recursos e Ferramentas Utilizadas

2.2.1 Software Necessário

- **WiimoteWhiteboard V 02** – O Programa em Visual C #, criado por Johnny Chung Lee que se encarrega de interpretar os dados enviados pelo Wiimote ao computador.

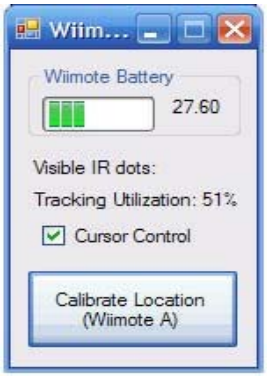

Figura 1 – Layout do Programa WiimoteWhiteboard

- **BlueSoleil 1.6.2.1 e 5.0** – Este programa encarrega-se de estabelecer a comunicação entre o computador e o comando Wiimote. Antes de instalar o programa é aconselhável consultar o manual de instalação do USB Bluetooth já que muitos requerem a instalação dos drivers antes de estabelecer a conexão do USB. Existem também alguns adaptadores USB que são incompatíveis com o BlueSoleil (consultar compatibilidade em http://www.wiili.org/index.php/Compatible_Bluetooth_devices)

- **Microsoft. Net Framework 2.0 –** É necessário ter instalado no computador o .Net Framework 2.0 ou superior para que o programa funcione. Um *Framework* pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projecto de software.

2.2.2 Hardware Necessário:

- **Computador com Windows Xp ou Windows Vista.**
- **Projector multimédia.**
- **Comando Wiimote** – Comando da consola de jogos Wii da Nintendo

Figura 2 – Comando Wiimote

- **USB Bluetooth** – Existem vários adaptadores Bluetooth no mercado nem todos são compatíveis com o BlueSoleil, no entanto é possível conectar-se com um qualquer adaptador através do painel de controlo do Windows.

Figura 3 – Adaptador USB Bluetooth

- **LED emissor IR –** O tipo de LED infravermelho é o principal comunicador entre o Wiimote e o computador, existem vários modelos que emitem luz com um ângulo de abertura maior que outros, com mais potência e com longitudes de onda distintas, neste caso é necessário escolher o mais adequado, os LEDs usados nas experiências efectuadas foram VISHAY TSAL 6200 e 6100 (**Anexo 7.1**)

Figura 4 – LED Emissor TSAL 6200

- **Botão de pressão** – No mercado existem vários modelos, foi escolhido este (o botão transborda 0,5 cm da sua base) porque parece ser o mais ergonómico para os dedos.

Figura 5 – Botão de Pressão.

- **Pilhas** – Qualquer tipo de pilhas de 1,5 V ou 3 V.
- **Resistência de 15 a 30 Ohm (1/4W) (usar a 3V)** – Elas servem para não danificar o LED quando é sujeito a um valor inadequado de tensão.

Figura 6 – Resistência para Prevenir que Acima de 1,5 Volts o LED não se Danifique.

- **Cabo multifilar –** Podem ser aplicados cabos de rede na concepção das canetas no entanto este cabo multifilar quebra menos que o cabo de rede.

Figura 7 – Cabo Multifilar para Efectuar Ligações.

- **Caneta digital com LED infravermelho** – Na figura seguinte mostra-se o esquema de uma caneta infravermelha indicando que tipo de resistência é necessário para cada caso de tensão.

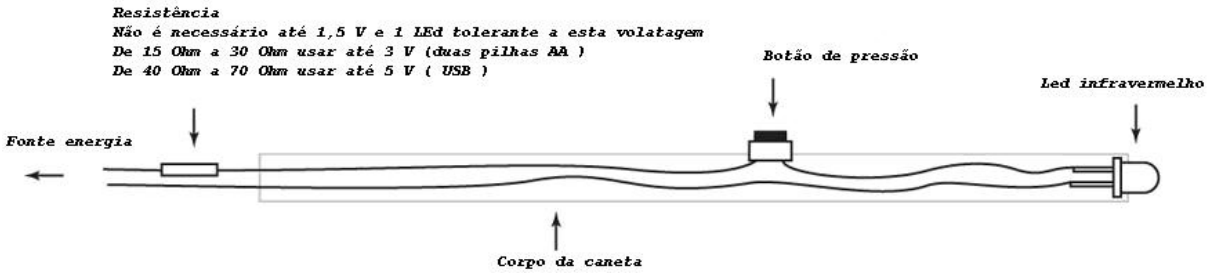

Figura 8 – Esquema para Construir uma Caneta Digital.

De seguida apresenta-se uma imagem real da construção de uma caneta seguindo o esquema apresentado.

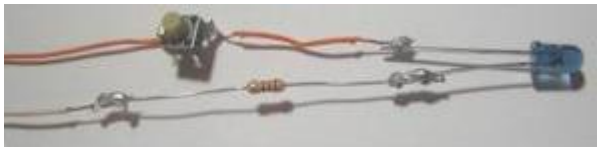

Figura 9 – Ligação das Componentes na Construção de uma Caneta

2.3 Seguidor Posição em Ambientes Imersivos 2D com Analogia de Percepção em 3D

O segundo projecto apresentado por Johnny Chung Lee, é criar ambientes virtuais usando para o efeito a câmara de infravermelhos do Wiimote de forma a demonstrar que com o

movimento do corpo podemos criar ambientes imersivos a duas dimensões com analogia de percepção a três dimensões.

Isto é possível através de dois LEDs montados num chapéu, desta forma pode-se acompanhar com precisão a localização do corpo (cabeça), isso transforma a projecção da imagem num ambiente virtual dinâmico. A projecção reage adequadamente ao movimento da cabeça e do corpo como se fosse um verdadeiro cenário 3D, criando uma ilusão de profundidade e espaço.

2.4 Definição dos Recursos e Ferramentas Utilizadas

2.4.1 Software Necessário

- **WiiDesktopVr** – O Programa em Visual C #, criado por Johnny Chung Lee que se encarrega de interpretar os dados enviados pelo Wiimote ao computador.
- **BlueSoleil 1.6.2.1 e 5.0**
- **Microsoft .Net Framework 2.0**
- **DirectX SDK (Microsoft) –** É uma colecção de APIs que tratam de tarefas relacionadas com a programação de jogos e grafismos para o sistema operativo Microsoft Windows, ou seja, é quem padroniza a comunicação entre software e o hardware gráfico e interpreta as instruções gráficas.

2.4.2 Hardware Necessário

- **Computador com Windows Xp ou Windows Vista.**
- **Projector multimédia.**
- **Comando Wiimote**
- **USB Bluetooth**
- **LED emissor IR**
- **Interruptor** – No mercado existem vários modelos, o interruptor estabelece a ligação com os LEDs.
- **Pilhas** (1.5V a 3V)
- **Resistência de 15 a 30 Ohm (usar a 3V)**
- **Cabo multifilar**

2.5 Controlo de Cursor Através dos Movimentos dos Dedos (Tipo Minority Report).

O terceiro projecto que Johnny Chung Lee nos apresenta, usa uma matriz de LEDs e algumas fitas reflectoras podemos usar a câmara infravermelha do Wiimote para localizar objectos, com os próprios dedos em duas dimensões, ou seja com umas pequenas fitas reflectoras ou com os próprios LEDs de infravermelhos podemos interagir com o computador com o simples agitar dos dedos, uma cena semelhante á do filme "Minority Report" neste caso o Wiimote pode monitorizar até quatro pontos em simultâneo.

2.6 Definição dos Recursos e Ferramentas Utilizadas

2.6.1 Software Necessário

- **Wiimote Multipoint Grid** – O Programa em Visual C #, criado por Jonhnny Chung Lee que se encarrega de interpretar os dados enviados pelo Wiimote ao computador.
- **BlueSoleil 1.6.2.1 e 5.0**
- **Microsoft .Net Framework 2.0**
- **DirectX SDK**

2.6.2 Hardware Necessário

- **Computador com Windows Xp ou Windows Vista.**
- **Projector multimédia.**
- **Comando Wiimote**
- **USB Bluetooth**
- **LED emissor IR –** O LED infravermelho (**Figura 10**) é o principal comunicador entre o Wiimote e o computador. Existem vários modelos que emitem luz com um ângulo maior que outros, com mais potência e com longitudes de onda distintas, neste caso foi necessário escolher o mais adequado, os LEDs usados na experiência efectuada foram: SFH 4231 (**Anexo 7.2**).

Figura 10 – LED Emissor Infravermelho SFH 4231 [OSRAM]

- **Matriz de LEDs** – Vários LEDs em série (**Figura 11**) totalizando uma tensão de 12 V.

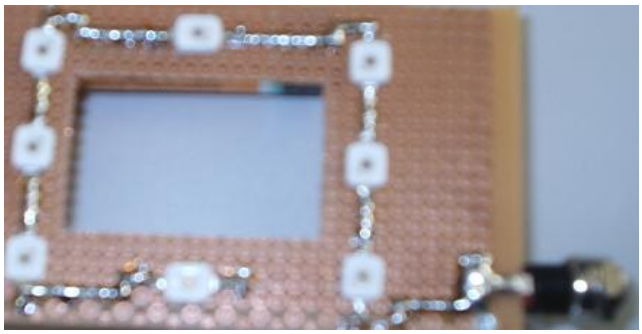

Figura 11 – Matriz de LEDs.

- **Transformador** – O alimentador (**Figura 12**) da Matriz de LEDs usa uma tensão de 12 Volts.

Figura 12 – Alimentador da Matriz de LEDs.

- **Fita reflectora** – O órgão reflector usado foi papel de alumínio

Capítulo 3

3 Projecto e Desenvolvimento

3.1 Análise

O elemento essencial detector dos nossos dispositivos IR, para o multi-apontador em quadro branco, é o Wiimote. Este é um comando que é usado para jogar com o Wii da Nintendo e que é caracterizado pela sua originalidade, pela capacidade de detectar os movimentos e pela capacidade de mover o cursor na superfície de projecção através de infravermelhos. O Wiimote pode ser comprado por cerca de 40 euros e utilizá-lo com o computador sem comprar a consola Wii.
Johnny Chung Lee ficou fascinado com as características técnicas do Wiimote e começou a investigar possíveis utilizações alternativas aos jogos. Um de seus múltiplos projectos que utiliza o Wiimote consiste precisamente em obter um quadro interactivo de baixo custo (Low-Cost Multi-Point Interactive Whiteboards Using the Wiimote) em http://www.cs.cmu.edu/~johnny/projects/wii/, ele desenvolveu o software necessário, para implementar o quadro interactivo de baixo custo. O programa é chamado WiimoteWhiteboard e é gratuito. Além disso, o código fonte está disponível (**Anexo 7.3**) para os programadores que o queiram melhorar. O software também permite:

- Converter o monitor do computador em ecrã táctil usando os mesmos princípios;
- Obter um Tablet-Pc (a imagem é projectada sobre uma superfície horizontal).

Vejamos como funciona este sistema de quadro interactivo de baixo custo:

1. Um computador é ligado a um projector que proporciona uma imagem em uma superfície.
2. O Wiimote deve estar fixo e a abranger toda a superfície de projecção.
3. Usar uma caneta digital com luz infravermelha para interagir com a imagem projectada.
4. O Wiimote tem uma câmara infravermelha e capta a luz infravermelha da caneta.
5. O Wiimote envia aquilo que vai captando por Bluetooth para o Computador.
6. O software de Johnny Chung Lee recolhe e interpreta os dados e actua movendo o cursor do computador ao pressionar o botão.

3.2 A Aplicação WiimoteWhiteboard

3.2.1 Como Iniciar a Aplicação

É necessário estabelecer entre o computador e o Wiimote a comunicação mediante Bluetooth usando a aplicação BlueSoleil ou outra compatível. Executar a aplicação WiimoteWhiteboard de Johnny Chung Lee, utilizar uma caneta digital previamente construída. O conselho que se dá é para ir trabalhando por fases, primeiro é importante estabelecer a comunicação Wiimote /computador.
Até não se conseguir estabelecer a comunicação não é possível experimentar a aplicação WiimoteWhiteboard.

3.2.2 Comunicação Wiimote-Computador Através de Bluetooth

Para estabelecer a ligação do computador com o Wiimote é necessário o adaptador Bluetooth, o software de gestão da comunicação, neste caso foi usado o BlueSoleil, e iniciar a comunicação com o computador.
Deve-se ter em atenção que nem todos os adaptadores Bluetooth funcionam correctamente quando comunicam com o Wiimote. Existe uma lista contendo os que são compatíveis em http://www.wiili.org/index.php/Compatible_Bluetooth_devices.

3.2.3 O Programa BlueSoleil

Regra geral é necessário instalar o programa BlueSoleil antes do adaptador Bluetooth, se o Windows tentar instalar novos controladores deve-se cancelar esse processo para não interferir com o BlueSoleil.
Para iniciar o programa, fazemos duplo clique no ícone que aparece na barra de tarefas, ao lado do relógio do Windows e aparece a janela principal do BlueSoleil (**Figura 13**).

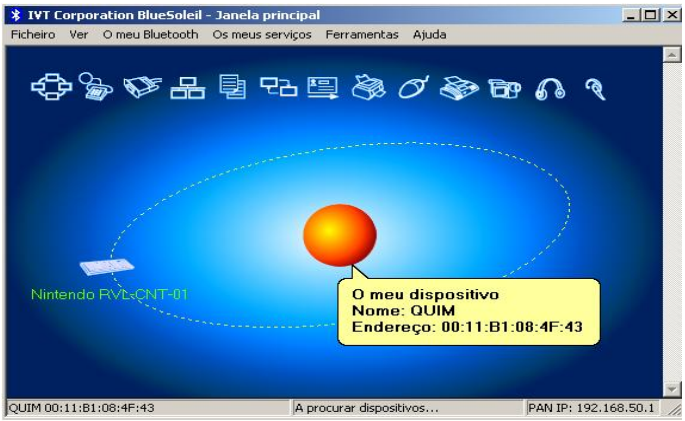

Figura 13 – Aspecto Geral da Aplicação BlueSleil

1. Dando um clique sobre o ponto central laranja o programa detecta todos os dispositivos Bluetooth ao seu alcance.
2. Um pouco antes ou depois pressiona-se as teclas 1 e 2 do Wiimote, isto serve para activar a ligação Bluetooth nesse instante os LEDs azuis piscam duração da ligação.
3. Em seguida aparece um ícone com o nome Nintendo-RVL-CNT-01, dá-se um clique no botão superior direito e selecciona-se a opção "Actualizar Serviços."
4. Depois de alguns segundos damos um clique no botão da barra superior no ícone do rato, e em seguida, aparece uma linha tracejada entre o Wiimote e o ponto laranja. Isso significa que nós já temos uma ligação ao computador. As luzes azuis do Wiimote acendem de forma fixa. O número de luzes acesas indica o estado das pilhas (com uma luz, é baixa energia).

Se durante o processo explicado acima as luzes do Wiimote não acenderem é necessário executar todo o processo descrito para realizar a comunicação com sucesso.

3.2.4 Verificar a Compatibilidade do Adaptador Bluetooth

Breve descrição para verificar a compatibilidade do adaptador com o BlueSoleil:

1. Conectar o Adaptador Bluetooth ao computador (este passo pode ser omitido no caso de existir um interno)
2. Executar o programa.
3. Situar o cursor em acima do ponto central laranja.
4. Se houver compatibilidade, aparece uma mensagem com o nome do computador e um endereço composto de pares de dígitos hexadecimal (**Figura 14**).

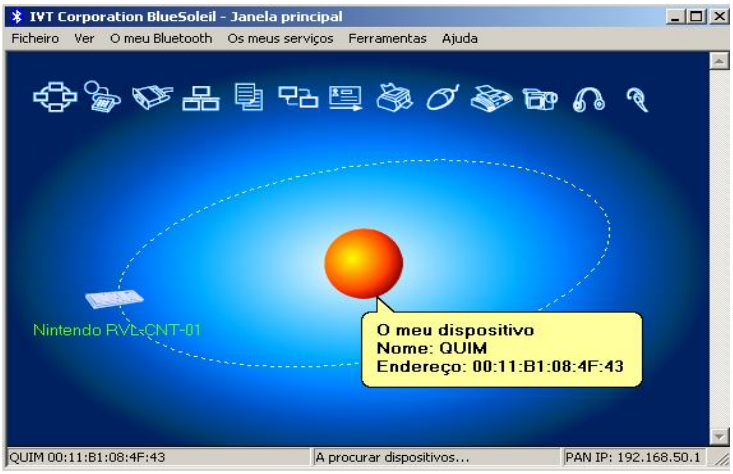

Figura 14 – Aplicação BlueSoleil Compatibilidade com Adaptador Bluetooth

5. Caso não se verifique compatibilidade, não aparece o nome do computador e o endereço aparece com todos os dígitos a zero (**Figura 15**).

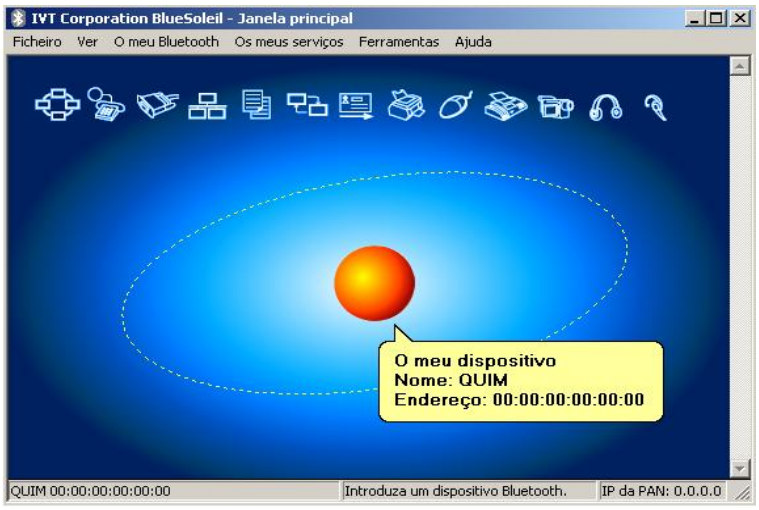

Figura 15 – Aplicação BlueSoleil Quando da Incompatibilidade da Aplicação ao Adaptador

Se verificarmos este último caso, existem duas opções:

1. Experimentar outro adaptador Bluetooth e verificar na data de aquisição se é compatível com o BlueSoleil.
2. Tentar realizar a comunicação com o software Bluetooth do Windows ou com o programa e os controladores que vêm com o adaptador adquirido.

Antes de continuar é necessário colocar o Wiimote apontado para a superfície de projecção e ter em conta que a câmara de infravermelhos que este possui tem um ângulo de visão de 45 graus, assim não devemos coloca-lo numa posição demasiado perto já que podemos correr o risco de este não abarcar toda essa superfície de projecção pelo contrário se o afastamos muito podemos perder o acompanhamento da luz infravermelha.

3.2.5 O Ficheiro WiimoteWhiteboard

O ficheiro WiimoteWhiteboard é a aplicação a executar conforme se pode ver na **Figura 16**.

Figura 16 – Ficheiro de Arranque do WiimoteWhiteboard

Depois de executar o ficheiro pretendido surge o layout da aplicação como se mostra na **Figura 17**.

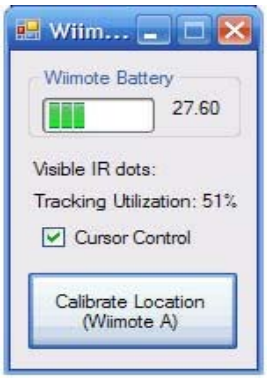

Figura 17 – Layout do Wiimote WhiteBoard

Antes de calibrar a superfície de projecção, passo indispensável para obter uma boa calibração da superfície, é aconselhável desligar o "Cursor Control" (cursor controlo) para verificar com o ponteiro infravermelho se o Wiimote abrange toda a área de projecção, como indicado anteriormente.
A ideia é a de manter o ponteiro emitindo luz infravermelha e avançar ao longo de todo o ecrã ou superfície, quando a Wiimote detecta luz infravermelha revela um número ao lado de "Visíble IR dots" (pontos infravermelhos visíveis), um 1 se apenas temos um ponteiro, um 1 e um 2, se usamos dois e são capturadas em simultâneo, e assim por diante.
Se mover-mos o ponteiro infravermelho num canto e desaparecer o número que foi atribuído, por exemplo, o 1, significa que este ponto não é captado pelo Wiimote, desta forma devemos ajustar a distância, deslocando-o para um lado ou para o outro. Teremos o Wiimote bem posicionado quando o ponteiro infravermelho esta ligado, e seja mostrado um 1 na "Visible IR dots".
Uma vez estabelecida a comunicação do Wiimote com o computador estamos em condições de poder testar a aplicação com a nossa caneta digital de infravermelhos.

3.3 Possiveis Erros e Soluções

Problema: A aplicação não pode iniciar correctamente.
"The application failed to initialize properly (0xc0000135)"

Solução: É necessário instalar Microsoft. NET 2,0

Problema: A aplicação não pode encontrar o Wiimote.
"Exception: Error Reading data from Wiimote…is it connected?"

Solução: A comunicação Wiimote – PC não foi estabelecida. É necessário fazê-lo previamente com BlueSoleil ou outros software de Bluetooth compatíveis.

Problema: Nada acontece quando você liga o ponteiro infravermelho em cima de um dos pontos vermelhos de calibração.
Esta ***solução*** pode ser devido a diferentes causas:

1. O ponteiro infravermelho não funciona. A maioria das câmaras digitais é sensível à luz infravermelha. Só temos de apontar o LED para a câmara e verificar se há ou não luz no LED deve-se obter um novo LED no segundo caso.

2. O Wiimote não pode ver o LED infravermelho, porque é fora do seu campo de visão. Usando a opção "Visível IR dots" comentada anteriormente será possível detectar se é esse o problema. Se assim for, temos de reajustar a posição do Wiimote. É importante que, enquanto efectuar tais controlos, nenhuma parte do nosso corpo interfira entre o Wiimote e a luz infravermelha.

3. A luz infravermelha não é brilhante o suficiente, porque não basta obter energia a partir de pilhas ou baterias. Em tais casos, devem ser substituídas por novas ou modificar o circuito eléctrico da caneta (menor resistência, mais tensão) ou alterar o tipo de LED infravermelho por um mais potente.

Problema: Linhas desenhadas com a caneta são irregulares e a precisão é pobre.

Solução: Lembramos que o Wiimote tem uma câmara infravermelha e quanto melhor é a visão que ele tem na superfície de projecção, melhor a resolução da caneta. Assim, o bom posicionamento do Wiimote é crucial para o sucesso, nem muito longe nem muito próximos. A opção "Visível IR dots" ajuda a encontrar a melhor posição (sabendo que o ângulo de visualização do Wiimote é de 45 graus) para cobrir toda a superfície sem distanciar-se muito. Também pode ser porque o LED infravermelho não é muito forte.

Problema: O cursor nem sempre responde aos movimentos e cliques da caneta, aparecendo em outras partes da superfície.

Solução: Verifique se não há outra fonte de luz infravermelha que atinge o Wiimote e interfira na comunicação. As chamas de uma vela, um isqueiro ou os próprios raios solares emitem luz infravermelha assim são possíveis fontes de interferências. Para verificar se existem, terá que se verificar a opção "Visível IR dots" não contém quaisquer números quando não utilizamos a nossa caneta. Se aparecer algum número, significa que há interferência que é preciso localizar e "remover".

3.4 Considerações Prévias

Existem dois factores que influenciam significativamente a qualidade deste projecto:

1. A posição do Wiimote em relação à imagem projectada: nem demasiado perto (risco de não abranger a totalidade da superfície), nem muito longe (perde resolução de monitoramento). Pode-se obter uma óptima posição utilizando a função "Visível IR dots" da aplicação WiimoteWhiteboard v0.2 de Johnny Chung Lee.
2. A qualidade da luz infravermelha emitida pelo LED: assim como existem lâmpadas de 40 ou de 100 watts e emitem mais ou menos luz, podemos usar um LED infravermelho mais ou menos potente. Quanto mais brilhante é o feixe infravermelho e maior abertura tiver, melhor capturado será pelo Wiimote e desta forma os resultados serão mais satisfatórios.

3.5 Limitações do WiimoteWhiteboard

Foram detectadas algumas limitações na utilização da aplicação WiimoteWhiteboard, de seguida enumeram-se algumas dessas limitações.

3.5.1 Sombras na Superfície de Projecção

É verdade que se pode apoiar a ponta da caneta sobre uma superfície com a imagem projectada, sem afectar a superfície. O que importa é saber se emite luz infravermelha ou não (que é captada pelo Wiimote). Isto implica um pequeno inconveniente inicial:
Temos de pressionar o botão da caneta, sempre que se queira fazer um clique e mantê-lo pressionado se for necessário desenhar ou arrastar objectos.
De facto, a principal desvantagem deste "modus operandi" é o de conseguir um ângulo sempre visível do Wiimote em relação ao LED infravermelho porque em muitos destes movimentos o corpo faz demasiada sombra sobre a projecção.
Uma possível solução para este problema seria a colocação do Wiimote numa posição superior à da superfície de projecção para que interferisse menos com a sombra que o utilizador faz na superfície de projecção.

3.5.2 Emulação do Botão Direito do Rato

Não tendo a capacidade de emular o botão direito do rato (versão original) já coloca um problema mais grave. Muitas vezes é necessário mostrar menus de contexto que aparecem somente desta maneira. Seria lógico definir o botão direito numa qualquer posição estática com a luz infravermelha acesa alguns instantes, simulando o botão direito do rato. É assim que funcionam os Tablet PC ou o PDA com ecrã táctil.

3.5.3 A Qualidade do Traço

Nas primeiras experiências efectuadas saltaram à vista as irregularidades acentuadas do traço quando usamos programas de desenho.
É pouco aconselhável usar a escrita com uma aplicação deste tipo, pois em determinadas situações o traço é imperceptível e muito pouco linear.

3.5.4 Compatibilidade com Algumas Aplicações Windows

Em geral, o WiimoteWhiteboard é compatível com a grande maioria das aplicações existentes. No entanto, existem alguns casos em que não o é. Um exemplo é dado com a aplicação PowerPoint 2007. Quando se quer entrar em modo de apresentação, esta opção não funcionava adequadamente.

3.6 Requisitos

3.6.1 Requisitos Funcionais

Para contrariar estas lacunas encontradas na aplicação WiimoteWhiteboard original foram identificadas as funcionalidades que se querem introduzir na aplicação para resolução destas limitações.

1. Adicionar dispositivo: O utilizador deve poder emular o botão direito do rato ao fim de alguns instantes.
2. Adicionar Filtro: O utilizador deve poder accionar filtro de forma a tornar o traço mais suave.
3. Verificar compatibilidade: Na aplicação Microsoft PowerPoint poder accionar o modo de apresentação de diapositivos.

3.7 Desenvolvimento

Para alterar o funcionamento do WiimoteWhitheboard foram necessárias modificações ao código fonte disponibilizado por de Johnny Chung Lee, este código desenvolvido em linguagem de programação C# que oferece poder, facilidade e flexibilidade é uma linguagem nativa para a plataforma. NET. o C # resolve o abismo entre linguagem de “baixo nível” e as linguagens “de alto nível”.
Apresenta-se um resumo das alterações realizadas ao código fonte inicial (**Anexo 7.4**):
Primeiro foram criadas as variaveis funcionais para o botão direiro do rato e para a funcionalidade do filtro.Em seguida definiram-se os parâmetros no ficheiro app.config (**Apêndice A**) do botão direito, do filtro e do tempo de disparo do botão direito.
E por fim foi alterado o valor da propriedade time do objecto mi para 500 para poderem funcionar bem alguns botões (ícones) do PowerPoint.
Para explicar um pouco as mudanças na caneta no que diz respeito à suavidade do traço, é necessário descrever como funciona o sensor infravermelho do Wiimote. Um emissor de luz

infravermelha é posicionado na superfície de projecção. O Wiimote capta essa luz como posições num dado intervalo. Depois de tirar a média do intervalo, o Wiimote tem a posição em que o utilizador está apontando a caneta, essa posição é decomposta em dois eixos X e Y que são utilizados para controlar a câmara do Wiimote. Desta forma, a câmara é controlada pelo movimento da caneta digital.

3.8 Modificações Introduzidas no WiimoteWhiteboard

Definição das variaveis adicionadas ao WiimoteWhiteboard:

```
public bool move_flag = true;
public int old_dx = 0;
public int old_dy = 0;
public int now_dx = 0;
public int now_dy = 0;
public bool IR_flag = false;
public int[] vxx = null;
public int[] vyy = null;
public int number_of_positions = 18 ;
public int move_threshold = 100;
public int ind = 0;
public int medx = 0;
public int medy = 0;
public int first_x = 0;
public int first_y = 0;
public bool filter_flag = true;
```

Parametros definidos e lidos a partir do ficheiro app.config em xml para configuração das novas funcionalidades da aplicação (apendice A):

```
 timer1.Interval =
Convert.ToInt32(System.Configuration.ConfigurationManager.AppS
ettings.Get("right_key_time"));
            number_of_positions =
Convert.ToInt32(System.Configuration.ConfigurationManager.AppS
ettings.Get("smooth"));
            move_threshold =
Convert.ToInt32(System.Configuration.ConfigurationManager.AppS
ettings.Get("move_threshold"));

//define the vectors to store positions
vxx = new int[number_of_positions];
vyy = new int[number_of_positions];
```

Preenchimento das posições do valor inicial do filtro:

```
if(filter_flag)
 {
  // fill the whole buffer with the first position
```

```
   for(int ix = 0; ix < number_of_positions; ix++)
   {
         vxx[ix] = first_x;
         vyy[ix] = first_y;
   }
}
```

Valor atribuido para accionamento do icone de modo de apresentação do PowerPoint:

```
     buffer[1].type = INPUT_MOUSE;
     buffer[1].mi.dx = 0;
     buffer[1].mi.dy = 0;
     buffer[1].mi.mouseData = 0;
     buffer[1].mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
     buffer[1].mi.time = 500; //500 to work fine with
     //some buttons in the PowerPoint
     buffer[1].mi.dwExtraInfo = (IntPtr)0;
     SendInput(2, buffer, Marshal.SizeOf(buffer[0]));
```

Prenchimento do vector de posições com a última posição detectada:

```
 if (filter_flag)
  {
    vxx[ind] = (int)(warpedX * 65535.0f / screenWidth);
    vyy[ind++] = (int)(warpedY * 65535.0f / screenHeight);
    if (ind == number_of_positions) ind = 0; //circular buffer
```

Calculo da média dos pontos do intervalo de valores:

```
    medx = 0;
    medy = 0;
    for (int ix = 0; ix < number_of_positions; ix++)
    {
      medx += vxx[ix];
      medy += vyy[ix];
    }
   // set the average position
    buffer[0].mi.dx = medx / number_of_positions;
    buffer[0].mi.dy = medy / number_of_positions;

    } // if filter on
 now_dx = buffer[0].mi.dx;
 now_dy = buffer[0].mi.dy;

 SendInput(1, buffer, Marshal.SizeOf(buffer[0]));
 // is there movement?
 if ((old_dx - now_dx) * (old_dx - now_dx) + (old_dy - now_dy)
* (old_dy - now_dy) > move_threshold)
move_flag = true;
old_dx = buffer[0].mi.dx; // store the actual dx to compare
//with the future dx
old_dy = buffer[0].mi.dy; // store the actual dy to compare
//with the future dy
                    }
                }
```

Lançamento do botão direito do rato ao fim de um segundo:

```
    private void timer1_Tick_1(object sender, EventArgs e)
        {
            if (cursorControl && IR_flag)
            {
                INPUT[] buffer = new INPUT[1];

                buffer[0].type = INPUT_MOUSE;
                buffer[0].mi.dx = 0;
                buffer[0].mi.dy = 0;
                buffer[0].mi.mouseData = 0;
                buffer[0].mi.dwFlags = MOUSEEVENTF_LEFTUP;
                buffer[0].mi.time = 0;
                buffer[0].mi.dwExtraInfo = (IntPtr)0;

                SendInput(1, buffer,
                Marshal.SizeOf(buffer[0]));

                buffer[0].type = INPUT_MOUSE;
                buffer[0].mi.dx = 0;
                buffer[0].mi.dy = 0;
                buffer[0].mi.mouseData = 0;
                buffer[0].mi.dwFlags = MOUSEEVENTF_RIGHTDOWN;
                buffer[0].mi.time = 0;
                buffer[0].mi.dwExtraInfo = (IntPtr)0;

                SendInput(1, buffer,
                Marshal.SizeOf(buffer[0]));

                buffer[0].type = INPUT_MOUSE;
                buffer[0].mi.dx = 0;
                buffer[0].mi.dy = 0;
                buffer[0].mi.mouseData = 0;
                buffer[0].mi.dwFlags = MOUSEEVENTF_RIGHTUP;
                buffer[0].mi.time = 0;
                buffer[0].mi.dwExtraInfo = (IntPtr)0;
               SendInput(1,buffer, Marshal.SizeOf(buffer[0]));
            }
            timer1.Stop();

        }
        // launch the right key event (using timer1) when the
cursor is stopped
        private void timer2_Tick(object sender, EventArgs e)
        {
            if (move_flag)
            {
                move_flag = false;
                timer1.Stop();
            }
            else
```

```
            {
                if (IR_flag)
                {
                    timer1.Enabled = true;
                    timer1.Start();
                }
```

CheckBox para accionar o filtro no layout da aplicação:

```
private void cbFilter_CheckedChanged(object sender, EventArgs
e)
{           if(cbFilter.Checked)
              filter_flag = true;
            else
              filter_flag = false;
}
```

CheckBox para accionar o botão direito do rato no Layout da aplicação:

```
private void cbRightButton_CheckedChanged(object sender,
EventArgs e)
        {
            if (cbRightButton.Checked)
                timer2.Enabled = true;
            else
                timer2.Enabled = false;
        }
```

3.9 Nova Versão para o WiimoteWhiteboard Utilizando Nova Caneta com Processador Digital

Depois de testar e analisar os resultados obtidos com a aplicação inicial e a modificação introduzida no código (**Anexo 7.6**), surgiu a ideia de alterar o modo de funcionamento da caneta electrónica de forma a tornar a sua utilização mais rápida e funcional.
Assim foi desenvolvida uma versão para a nova caneta novamente utilizando a linguagem de programação C#, a nova caneta (**Figura 18)** possui três botões, um botão é o clique normal, outro é o clique direito e o outro é um botão que pode ser programado com uma tecla como por exemplo a tecla Back Space do teclado.
Esta caneta é baseada num chip 16f628A (**Anexo 7.5)** o chip excita o LED através da criação de um código padrão (**Apêndice B**), estes códigos são enviados para o Wiimote a 100Hz, os códigos têm um campo de preâmbulo, um campo de acção, um campo de código e um bit de paragem.
No *layout* da aplicação o filtro pode ser desligado ou ligado.

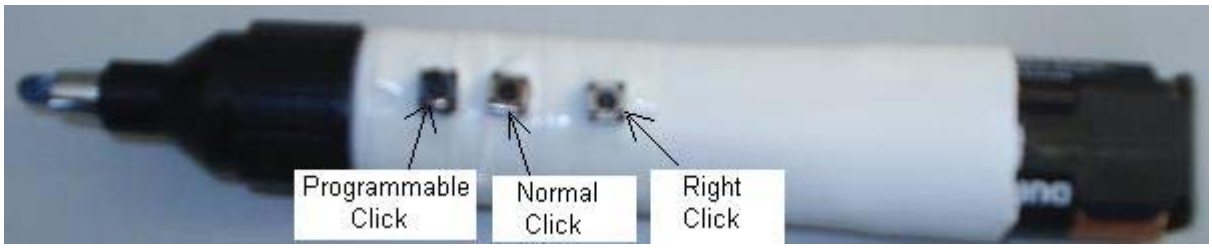

Figura 18 – Caneta Digital com Identificação da Função dos Botões

3.9.1 Novo Layout

De seguida apresenta-se o aspecto do novo *layout* (**Figura 19)** da nova versão do WiimoteWhiteboard, conforme se pode verificar o utilizador tem como opção ligar ou desligar a opção de filtro assim como todos os botões desactivando o Cursor Control.

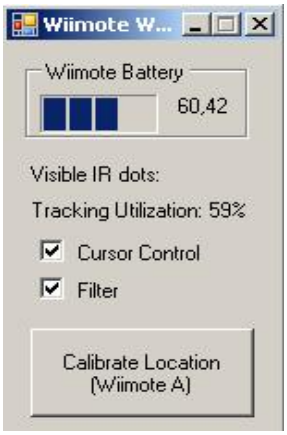

Figura 19 – Layout da Nova Versão do WiimoteWhiteboard

3.10 Modificações na Nova Versão WiimoteWhiteboard

Os segmentos de código mostrados a seguir retratam as principais vertentes das novas funcionalidades adicionadas ao sistema de WiimoteWhiteboard.
O filtro de suavização funciona com base na média posicional dos últimos *n* pontos detectados pelo Wiimote.
A detecção dos novos comandos, da nova caneta, é feita pela observação do estado dos bits de uma trama enviada pela caneta.

Variáveis para o botão direito e para a função de suavização do filtro:

```
public bool[] stateVector = new bool[8];
public bool rightClick = false;
public bool timer_Flag = false;
public int number_of_positions = 19;
public int [] vxx = null;
```

```
public int [] vyy = null;
public int ind = 0;
public int medx = 0;
public int medy = 0;
public int first_x = 0;
public int first_y = 0;
public int filter_flag = 1;
public byte key_code = VK_BACK;
```

Para carregamento do grau de suavização do filtro:

```
number_of_positions =
Convert.ToInt32(System.Configuration.ConfigurationManager.
AppSettings.Get("smooth"));
```

Para carregar o código de tecla do clique 3:

```
key_code =
Convert.ToByte(System.Configuration.ConfigurationManager.AppSe
ttings.Get("key_code");
```

Declaração e definição dos vectores para armazenar os 8 bits mais recentes da trama de IR:

```
vxx = new int[number_of_positions];
vyy = new int[number_of_positions];
stateVector[0] = false;
stateVector[1] = false;
stateVector[2] = false;
stateVector[3] = false;
stateVector[4] = false;
stateVector[5] = false;
stateVector[6] = false;
stateVector[7] = false;
```

Armazenamento dos últimos oito estados a 100Hz:

```
stateVector[0] = stateVector[1];
stateVector[1] = stateVector[2];
stateVector[2] = stateVector[3];
stateVector[3] = stateVector[4];
stateVector[4] = stateVector[5];
stateVector[5] = stateVector[6];
stateVector[6] = stateVector[7];
stateVector[7] = ws.IRState.Found1;
```

Código padrão recebido pelo Wiimote a 100Hz a partir do microprocessador da nova caneta:

preambulo: |1|0|1|0|
códigos de acção:
Right click = |1|1|
Programmable click = |0|0|
bit de paragem: |1|
exemplos de tramas: |1|0|1|0|1|1|1| ou |1|0|1|0|0|0|1|

Para se manter a compatibilidade das canetas originais com a função do *click* normal não foi considerado o preambulo |1|0|1|0| mas sim o |1|1|1|1|. Nesta situação, os restantes bits tem que estar também a 1.

Detecção dos novos botões:

```
// Testa o preâmbulo e o bit de paragem
if (cursorControl && stateVector[1] && !stateVector[2] &&
stateVector[3] && !stateVector[4] && stateVector[7])
{
int x = ws.IRState.RawX1;
int y = ws.IRState.RawY1;

float warpedX = x;
float warpedY = y;
warper.warp(x, y, ref warpedX, ref warpedY);

INPUT[] buffer = new INPUT[1];
buffer[0].type = INPUT_MOUSE;
buffer[0].mi.dx = (int)(warpedX * 65535.0f / screenWidth);
buffer[0].mi.dy = (int)(warpedY * 65535.0f / screenHeight);
buffer[0].mi.mouseData = 0;
buffer[0].mi.dwFlags = MOUSEEVENTF_ABSOLUTE |
MOUSEEVENTF_MOVE;
buffer[0].mi.time = 0;
buffer[0].mi.dwExtraInfo = (IntPtr)0;
SendInput(1, buffer, Marshal.SizeOf(buffer[0]));

rightClick = true;

if(!timer_Flag) // to avoid fast click repetitions
if (stateVector[5] && stateVector[6]) // Is the action code
//11? Then Right Click
{
buffer[0].type = INPUT_MOUSE;
buffer[0].mi.dx = 0;
buffer[0].mi.dy = 0;
buffer[0].mi.mouseData = 0;
buffer[0].mi.dwFlags = MOUSEEVENTF_RIGHTDOWN;
buffer[0].mi.time = 0;
buffer[0].mi.dwExtraInfo = (IntPtr)0;
```

```
SendInput(1, buffer, Marshal.SizeOf(buffer[0]));

buffer[0].type = INPUT_MOUSE;
buffer[0].mi.dx = 0;
buffer[0].mi.dy = 0;
buffer[0].mi.mouseData = 0;
buffer[0].mi.dwFlags = MOUSEEVENTF_RIGHTUP;
buffer[0].mi.time = 0;
buffer[0].mi.dwExtraInfo = (IntPtr)0;
SendInput(1, buffer, Marshal.SizeOf(buffer[0]));
}

// key click
if (!timer_Flag)// to avoid fast click repetitions
if (!stateVector[5] && !stateVector[6]) // Is the action code
//00? Then Key Code.
keybd_event(key_code, 0x45, 0, 0);

timer_Flag = true;
stateVector[0] = false;
stateVector[1] = false;
stateVector[2] = false;
stateVector[3] = false;
stateVector[4] = false;
stateVector[5] = false;
stateVector[6] = false;
stateVector[7] = false;
```

Detecção do início do botão normal (se o padrão do código enviado pela caneta só tiver 1s e o oitavo bit anterior não estava a 1) :

```
if (stateVector[1] && stateVector[2] && stateVector[3] &&
stateVector[4] && stateVector[5] && stateVector[6] &&
stateVector[7])
            {

if (!stateVector[0])//mouse down
                {
if (filter_flag == 1)
                        {
```

Posições de início do filtro:

```
//fill the whole buffer with the initial position
for (int ix = 0; ix < number_of_positions; ix++)
{
vxx[ix] = first_x;
vyy[ix] = first_y;
}
}
if (filter_flag == 1)
{
```

Prenchimento, no vector, do x e y da ultima posição observada pelo WiiMote:

```
vxx[ind] = (int)(warpedX * 65535.0f / screenWidth);
vyy[ind++] = (int)(warpedY * 65535.0f / screenHeight);
if (ind == number_of_positions) ind = 0; // circular buffer
```

Calculo da média posicional (filtro):

```
int medx = 0;
int medy = 0;
for (int ix = 0; ix < number_of_positions; ix++)
{
medx += vxx[ix];
medy += vyy[ix];
}

buffer[0].mi.dx = medx / number_of_positions;
buffer[0].mi.dy = medy / number_of_positions;

} //filtrar
```

Temporizador bloqueador de repetições rápidas de cliques (através da variável timer_Flag):

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            timer1.Stop();
            timer_Flag = false;
            rightClick = false;
        }
```

Temporizador (no papel de uma thread) para o lançamento do timer1 pela observação da timer_Flag

```
    private void timer2_Tick(object sender, EventArgs e)
        {
            if (timer_Flag)
            {
                timer1.Enabled = true;
                timer1.Start();
            }
        }
```

Botão de activação do filtro de suavização:

```
private void checkBox1_CheckedChanged(object sender, EventArgs
e)
        {
```

```
        if (checkBox1.Checked)
            filter_flag = 1;
        else
            filter_flag = 0;
    }
```

3.11 A Aplicação Desktop Vr e a Aplicação Wiimote Multipoint Grid

3.11.1 Testes Efectuados

Conforme a descrição efectuada para cada aplicação nos pontos 2.3 e 2.5 e na impossibilidade, por manifesta falta de tempo, de estudar e analisar as duas aplicações de forma a aumentar a sua autonomia e de, melhorá-las para outras de carácter útil, bem como ensaiar e optimizar o seu processo de utilização, estas duas aplicações apenas foram testadas de acordo com os princípios que foram descritos nos pontos anteriores relativamente a cada uma delas.

No entanto verificaram-se alguns erros no lançamento das duas aplicações, que foram resolvidos desactivando a opção “desactivar temas visuais” nas propriedades de cada uma delas no separador compatibilidade. De referir que na aplicação WiiDesktopVr verifica-se com clareza a noção de profundidade e espaço. Pois, como exemplo prático, quando nos aproximamos de uma janela temos uma perspectiva maior do espaço envolvente e quando nos afastamos temos uma perspectiva menor desse mesmo espaço. É desta forma muito simples que se consegue explicar aquilo que se vê quando usamos esta aplicação.

Na aplicação Wiimote Multipoint Grid houve dificuldade em adquirir a fita reflectora de modo a provar com maior precisão a detecção dos quatro pontos em simultâneo na projecção, no entanto usando papel de alumínio consegue-se esse resultado embora com mais dificuldade, usando os próprios LEDs das aplicações anteriores consegue-se mostrar esses pontos em simultâneo assim como diminuir, aumentar e rodar a grelha apresentada na imagem de projecção.

4 Análise de Resultados

4.1.1 Análise de Desempenho e Resultados do Sistema

O WiimoteWhiteboard resulta! Dos testes efectuados no primeiro contacto com o programa WiimotWhiteboard de Jonhnny Chung Lee tornou-se dificil pois a colocação do Wiimote interferia na boa utilização da projecção, depois de perceber a melhor colocação do Wiimote conseguiu-se ter um melhor dominio da mesma.
Embora,como já foi referido em pontos anteriores, tenham surgido alguns problemas no uso de algumas das aplicações de ambiente Windows, assim como o aspecto da suavidade do traço e a utilização do botão direito. Mesmo com estas limitações conseguiu-se obter um desempenho bastante aceitavel.
Analisando estas limitações no programa foi possivel modificar a partir do código fonte as limitações existentes e com isso introduzir o botão direito do rato quando utilizamos a projecção, este recurso entra em funcionamento ao fim de um segundo com a caneta digital imobilizada ou seja precionado o botão da caneta temos que a manter estática sem movimento durante um segundo para accionar o botão direito do rato. Com esta modificação na aplicação WiimoteWhiteboard o utilizador da superficie de projecção sente um maior domínio da mesma e uma maior flexibilidade no modo como pertende actuar, pois pode desligar e ligar o botão direito do rato no layout da aplicação. Com o botão direito o utilizador faz uma utilização das aplicações de modo mais rápido e confortável.
Quanto à suavidade do traço foram introduzidas as alterações correspondentes de modo a que o Wiimote capta-se a luz infravermelha como posições num dado intervalo de amostras posicionais. Depois de tirar a média do intervalo, o programa tem a posição média em que o utilizador está apontando a caneta, essa posição é decomposta em dois eixos X e Y que são utilizados para controlar a câmara do Wiimote. Desta forma, a câmara é controlada pelo movimento do ponteiro do rato de forma suave. Resolvida esta questão temos como resultado um traço muito mais liso e sem irregularidades.
O problema do PowerPoint foi resolvido colocando na classe do CursorControl a propriedade time do objecto mi com um tempo de 500ms. Desta forma conseguimos colocar em modo de apresentação de diapositivos um ficheiro no PowerPoint.
Com a nova versão o manuseamento das aplicações torna-se mais rápido, flexível e cómodo pois temos cada botão para sua função e evita-se ter a caneta imobilizada para que o clique direito seja disparado assim como a introdução do botão programado pode ser útil na eliminação de caracteres a partir da imagem projectada.
Nas duas outras aplicações disponibilizadas por Jonhnny Chung Lee, Controlo de cursor através dos movimentos dos dedos (Tipo Minority Report) e Seguidor de posição para ambientes imersivos 2D com analogia de percepção em 3D verificaram-se inicialmente os problemas descritos que foram resolvidos de modo a poder ter a perfeita percepção de que o Wiimote consegue monitorar até quatro pontos em simultâneo, que consegue localizar objectos com os dedos em planos a duas dimensões e que podemos acompanhar com precisão as imagens no monitor ou na projecção pois elas reagem ao movimento do corpo quando nos movemos, isto cria uma noção de profundidade e de espaço.
Para estas duas aplicações não foi desenvolvida qualquer alteração ao código por falta de tempo.

4.1.2 Comparação com Outros Sistemas

Após ver a utilização de quadros interactivos profissionais e até uma demonstração do quadro da Clasus concluímos que, apesar de serem todos diferentes em termos funcionais, o objectivo é sempre o mesmo: devolver às salas de aula uma maior interacção sendo possível simular qualquer ambiente, em qualquer disciplina, interagir dinamicamente com os alunos simplificando a partilha e a utilização dos diversos recursos e conteúdos produzidos.
Das utilizações referidas o WiimoteWhiteboard aproxima-se de todas, mas existem algumas diferenças de abordagem que serão referidas:
Relativamente à instalação/calibração é mais funcional o quadro profissional assim como na qualidade de imagem, embora com as modificações introduzidas na versão inicial do WiimoteWhiteboard a qualidade de imagem se aproxime bastante.
O WiimoteWhiteboard afasta-se definitivamente dos outros sistemas, no baixo custo de aquisição, na utilização para outros fins e na facilidade de uso.
O WiimoteWhiteboard apresenta uma nova abordagem, perante os sistemas profissionais, pois permite a criação/utilização independente de software, o programa não possui uma relação de interdependência de qualquer software, separando assim, a utilização de software próprio à utilização de software livre podendo o utilizador criar os seus recursos noutro qualquer software livre.

4.1.3 Aplicação Prática em Ambiente de Sala de Aula

Como professor de Tecnologias de Informação e Comunicação na Escola Secundária Emídio Garcia, o WiimoteWhiteboard foi usado para explicar conceitos de segurança na Internet, criar apresentações em PowerPoint e utilização de dois programas de física o Phun E o Microsoft Physics Illustrator.
Assim como na realização de exercícios interactivos produzidos em Hot Potatoes.
Neste segundo caso os alunos, tiveram a oportunidade de interagir com os conceitos anteriormente adquiridos em tempo real já que eles se encontram disponíveis na plataforma de ensino virtual da escola.
Ao usar o WiimoteWhitheboard, temos de dar sentido à letra *i* de interactivo porque não é um simples quadro convencional, o que queremos dizer é que existiram alguns contratempos iniciais na sua utilização, mas que foram ultrapassados com uma utilização mais intensa.
Um dos resultados verificados é que os alunos aprendem melhor experimentando.
Houve também oportunidade de divulgar estes projectos a um grupo de alunos e professores de Cabo Verde que visitaram a Escola Secundária Emídio Garcia.

4.1.4 Avaliação Geral

Como avaliação geral, consideramos que o projecto no que concerne à aplicação WiimoteWhiteboard, embora defira um pouco das propostas comerciais existentes no mercado, é uma possibilidade a encarar como uma solução de muito baixo custo e como alternativa portátil, no entanto, enquadra-se perfeitamente nos objectivos propostos inicialmente, cumprindo assim com o objectivo de estudar, e apresentar alternativas quanto à aquisição de novos equipamentos.
Com as melhorias efectuadas na aplicação inicial o uso desta tecnologia ficou mais consistente e capaz de se tornar um caso sério de sucesso.
Acreditamos que a evolução deste projecto é promissora, perante o entusiasmo e perspectivas dos possíveis utilizadores. Estas novas versões foram publicadas na Internet a fim de se mostrar as capacidades deste sistema ao público em geral.

5 Conclusões

Apesar do numero crescente de dispositivos lançados no mercado e a diminuição do custo de aquisição, a inovação na procura de novas ferramentas é sempre aliciante.
Este trabalho investigou algumas ferramentas e iniciativas desenvolvidas para utilização do dispositivo Wiimote da Nintendo, principalmente focado na aplicação Dispositivo Multi-apontador em quadro branco padrão de baixo custo (WiimoteWhiteboard).
Após a nalise da aplicação disponibilizada por Jonhnny Chung Lee, verificamos toda a sua funcionalidade, mas também detectamos algumas lacunas, foi essas que tentamos resolver e conseguimos resultados animadores.
Da actualização da aplicação resultou o incremento do botão direito do rato disparado depois de mantermos a caneta imobilizada ao fim de um segundo. Melhorou-se a qualidade do traço principalmente quando se utilizam programas de desenho, assim como as anomalias verificadas no PowerPoint .
Foi construida uma nova caneta digital de forma a termos a possibilidade de utilização dos botões de forma independente para uma melhor ergonomia, esta caneta tem a particularidade de apresentar um botão que pode ser programado com uma tecla como por exemplo a tecla *Back space*.
Este sistema deverá ser visto como possivel alternativa aos quadros interactivos profissionais pois para além de apresentarem uma redução muito significativa no preço final, em qualidade geral a sua de resolução é bastante satisfatória.
Este sistema pode ser encarado também como uma possibilidade de quadro intectativo portátil.
Desta forma os objectivos propostos foram cumpridos.
Remete-nos lembrar que, o projecto não acaba aqui, pois apenas demos o primeiro passo para algo que poderá ser grande.
Espera-se que os estudos que foram efectuados na área da interactividade contribuam de alguma forma para a criação de sistemas de muito baixo custo.

Apêndice A

A Detalhes do Ficheiro Xml

Este apêndice mostra os elementos que formam o ficheiro XML definido para configurar o módulo WiimoteWhiteboard e as aplicações que o utilizam. As secções seguintes são divididas por uma adição de uma *key* para suavizar o traço com um número de pontos definidos (19) para cálculo da média.
Uma *chave* para lançar o botão direito, indica que é necessário esperar esse tempo (um segundo para lançar o botão direito) (versão não micro programada).
Uma chave para indicar a sensibilidade ao movimento da caneta (versão não micro programada).
Na nova versão do WiimoteWhiteboard são mostrados os elementos que formam o ficheiro XML definido para configurar o módulo. As secções seguintes são divididas por uma adição de uma *key* para suavizar o traço com um número de pontos definidos (19) para cálculo da média.
Uma chave para carregar o código de tecla do clique 3.

A.1 Aspecto Geral Versão Inicial

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration><appSettings>
  <add key ="smooth" value="19"/> <!--  the filter smooth index  -->
  <add key ="right_key_time" value="1000"/> <!-- wait this time to launch
the right click-->
  <add key ="move_threshold" value="100"/> <!--  Sensitivity to the
movement  -->
</appSettings></configuration>
```

A.2 Aspecto Geral da Nova Versão

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key ="smooth" value="19"/> <!--  the filter smooth index  -->
    <add key="key_code" value="8"/> <!--  the key code to the click 3
|8=BackSpace -->
  </appSettings>
```

```
</configuration>
```

B Programação do Chip 16F628A para Nova Versão da Caneta Digital

Este apêndice mostra os dados relativos à programação do chip 16f628 para fazer uso do botão programável na nova versão da caneta digital.

B.1 Aspecto Geral

```
#include "ir_mod.h"
#bit RBIE = 0x0B.3
#bit RBIF = 0x0B.0

#byte T1CON = 0x10
#bit TMR1ON = T1CON.0

void main() {
  #asm
    movlw 0x0  //  weak pull up;
    option
  #endasm

  setup_vref(FALSE);

  TMR1ON = 0;

  disable_interrupts(INT_COMP);
  clear_interrupt(INT_COMP);

  RBIE = 1;

  set_tris_a(32);  //all output but mclear
  set_tris_b(0xF0); // MSBs (inputs) to allow the wakeup
  output_b(0xFF);

  while(1){RBIF = 0;
    if(input(PIN_B7) == 0)  // always on = normal click
       output_a(0xFF);
    else
     if(input(PIN_B6)==0)  //1010 11 1 = right click
     {

       output_a(0xFF);  // preamble
       delay_ms(10);
```

```
  output_a(0x0);
  delay_ms(10);

  output_a(0xFF);
  delay_ms(10);

  output_a(0x0);
  delay_ms(10);


  output_a(0xFF); // code
  delay_ms(10);

  output_a(0xFF);
  delay_ms(10);

  output_a(0xFF); // stop
  delay_ms(10);
}
else
if(input(PIN_B5)==0)  // 1010 00 1 = key click
{
  output_a(0xFF); // preamble
  delay_ms(10);

  output_a(0x0);
  delay_ms(10);

  output_a(0xFF);
  delay_ms(10);

  output_a(0x0);
  delay_ms(10);

  output_a(0x0); // code
  delay_ms(10);

  output_a(0x0); // code
  delay_ms(10);

  output_a(0xFF); // stop
  delay_ms(10);

}
else
if(input(PIN_B4)==0)  // 50hz
{
  output_a(0xFF);
  delay_ms(10);

  output_a(0x0);
  delay_ms(10);

  output_a(0xFF);
  delay_ms(10);

  output_a(0x0);
  delay_ms(10);
}
else
{
```

```
      output_a(0x0);
      sleep();
     }
   }
}
```

6 Bibliografia

KREYLOS, OLIVER. `*Homepage*. Disponível em http://graphics.cs.ucdavis.edu/~okreylos/ Último acesso em 13 de Junho de 2008.

LEE, JONHNNY CHUNG, *projects, Wii*. Disponível em http://www.cs.cmu.edu/~jonhnny/projects/wii/

MARQUES, P., PEDROSO, H., *C# 2.0.* FCA – Editora de Informática, LDA. 1ª Ed.2007

MICROSOFT. *DirectX. 2007*. Disponível em http://msdn.microsoft.com/directx/.

WIILI.ORG. *Wiimote. 2007*. Disponível em http://www.wiili.org/index.php?title=Wiimote&oldid=9285. Último acesso em 11 de Junho de 2008.

http://catalog.osram-os.com

http://www.vishay.com

7 Anexos

7.1 Características LED IR Tsal 6200 e 6100

7.2 Características LED SFH 4231

7.3 Código Fonte da Aplicação WiimoteWhiteboard

7.4 Modificações ao Código Fonte Inicial

7.5 Diagrama do Chip 16F628A

7.6 Código Fonte Nova Versão para o WiimoteWhiteboard