UNIVERSIDADE DA BEIRA INTERIOR
Covilhã | Portugal

Projecções Interactivas na Sala de Aulas

Vasco Miguel Alves dos Santos

Submetida à Universidade da Beira Interior para a obtenção do grau de Mestre em Engenharia Informática

Orientado pelo Prof. Doutor Frutuoso Gomes Mendes da Silva

Departamento de Informática
Universidade da Beira Interior
Covilhã, Portugal
http://www.di.ubi.pt

Agradecimentos

Quero agradecer ao meu orientador, o Prof. Doutor Frutuoso Gomes Mendes da Silva, por me ter dado esta oportunidade, bem como pelo seu apoio e ajuda em todas as etapas, desde a escolha da Tese à escrita da Dissertação.

Quero agradecer também aos restante docentes do Departamento de Informática da Universidade da Beira Interior que contribuíram, directa ou indirectamente, para que eu atingisse esta etapa.

Aos meus pais, irmão, primos e avôs, em especial ao meu avô José Maria Alves Sainhas que nos deixou à pouco tempo, agradeço por todo o apoio que me deram nos bons e maus momentos.

Por fim, agradeço a todos os meus amigos e colegas que sempre estiveram dispostos a apoiar-me e ajudar-me em tudo o que fosse preciso.

Resumo

As Tecnologias de Informação e Comunicação (TIC) têm um papel chave nos métodos de ensino. O uso de quadros interactivos é uma dessas tecnologias que tem vindo a aumentar nos últimos anos. No entanto, a maioria das soluções disponíveis no mercado têm um preço alto, o que significa que é muito difícil ter um quadro interactivo em cada sala de aula. Recentemente, Johnny Lee mostrou que é possível ter um quadro interactivo em qualquer superfície plana usando apenas um comando Wiimote e um emissor de infravermelhos, i.e., um quadro interactivo de baixo custo (ou Wiimote Whiteboard) disponível por menos de 100€, excluindo o projector. Baseada nesta tecnologia, foi criada uma aplicação de suporte às aulas que permite ao professor/apresentador tirar apontamentos (notas) sobre apresentações e guardá-las em ficheiro. Deste modo, também é possível disponibilizar estes apontamentos aos estudantes. Assim, os estudantes passam a ter os apontamentos em formato digital, em vez de utilizarem o caderno, permitindo uma maior partilha de recursos e ideias entre estudantes e professores. Esta aplicação permite escrever, desenhar e reconhecer texto e formas geométricas. Para além disso, também foi desenvolvida outra aplicação que permite editar as notas à posterior. Com esta aplicação é possível complementar as notas tiradas durante uma aula/apresentação, corrigindo, apagando ou adicionando mais notas. Esta aplicação dá mais liberdade ao professor/apresentador pois permite que as notas melhoradas estejam disponíveis aos alunos. Além disto, estendeu-se as capacidades do Wiimote Whiteboard tornando possível controlar o computador à distância usando um segundo comando Wiimote. Isto foi alcançado mapeando várias teclas chave nos botões do Wiimote. Deste modo, o professor/apresentador pode controlar o computador à distância, tornando o quadro mais interactivo, i.e., cancela quase na totalidade a dependência que um utilizador tem do rato e teclado do computador. Isto permite ao utilizador estar perto do quadro onde, além de o poder usar sem limitações, também pode controlar à distância o que quer mostrar no computador (e.g., Powerpoint,

imagens, vídeos, etc.). Por fim, e dado que um normal quadro interactivo é normalmente acompanhado de software educativo, foi desenvolvida uma aplicação visual de suporte às aulas de Introdução à Programação. Esta aplicação permite criar fluxogramas e gerar o correspondente pseudo-código e código fonte em Linguagem C e, deste modo, permite ensinar os fundamentos básicos da Programação.

Palavras chave

Projecção Interactiva, Quadro interactivo de baixo custo, Wiimote, Interacção Humana com o Computador, Wiimote Whiteboard.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API: Application Programming Interface

IU: Interface do Utilizador

IV: Infravermelho

LCD: Liquid Crystal Display

OCDE: Organização de Cooperação e Desenvolvimento Económico

PC: Personal Computer

PDF: Portable Document Format

PISA: Programme for International Student Assessment

SDK: Software Development Kit

TIC: Tecnologias de Informação e Comunicação

UE: União Europeia

Wiimote: Wii Remote Control

WPF: Windows Presentation Foundation

XAML: eXtensible Application Markup Language

XPS: XML Paper Specification

Capítulo 1

Introdução

Hoje em dia, o uso das Tecnologias de Informação e Comunicação (TIC) tem tido, na sociedade actual, um papel fundamental nas escolas e nos métodos de ensino. O recurso a apresentações (e.g., em PowerPoint), a e-books, a quadros interactivos, ao ensino à distância através de vídeo-conferência, a software educacional, etc., em vez dos habituais quadros pretos e livros de papel, tem-se tornado, cada vez mais, numa rotina do dia-a-dia. Os apontamentos passaram a ser armazenados em formato digital, em vez do caderno, permitindo uma maior partilha de recursos e ideias entre alunos e professores. Em 2001, os países da União Europeia (UE) definiram, como objectivo, aumentar a qualidade e a eficácia dos sistemas de educação e formação na UE, do qual resultaram duas questões-chave [6]:

- "*Assegurar uma gama adequada de equipamento e de software pedagógico a fim de optimizar a aplicação das TIC e dos processos de aprendizagem electrónica (e-Learning) nas práticas de ensino e de formação*";
- "*Incentivar a melhor utilização possível das técnicas inovadoras de ensino e aprendizagem baseadas nas TIC*".

Segundo este objectivo, os países da UE devem garantir que tanto os estabelecimentos de ensino, como os centros de aprendizagem, disponham de suficientes equipamentos, recursos multimédia e programas informáticos educativos e formativos de elevada qualidade. Devem também ter ligações de alta velocidade de modo a que os estudantes possam tirar verdadeiro proveito dos recursos disponíveis e das possibilidades interactivas da Internet [7].

Para atingir os objectivos propostos pelos países da UE em 2001, em Portugal foi criado, no ano de 2007, o Plano Tecnológico da Educação, do qual consta o Projecto Kit Tecnológico Escola [8]. O Projecto Kit Tecnológico Escola tem como objectivo promover a utilização de tecnologia no processo de ensino, dotando todas as escolas de um número adequado de computadores, de impressoras, de vídeo projectores e de quadros interactivos, de modo a alcançar os seguintes objectivos:

- Atingir o rácio de dois alunos por computador em 2010;
- Assegurar que nenhuma escola apresenta um rácio de alunos por computador superior a cinco;
- Assegurar um vídeo projector em todas as salas de aula;
- Assegurar um quadro interactivo em cada três salas de aula;
- Assegurar a renovação dos equipamentos, garantindo que a proporção de equipamentos com antiguidade superior a três anos não ultrapasse os 20%;
- Assegurar a disponibilização de computadores e de impressoras para utilização livre na escola, atingindo um rácio de cinco alunos por cada computador de acesso livre e de três professores por cada computador de acesso livre.

Para atingir estes objectivos, o Estado Português definiu como principais medidas o fornecimento às escolas com 2.º e 3.º ciclos do ensino básico ou com ensino secundário de 310 000 computadores, 9000 quadros interactivos e 25 000 vídeo projectores até 2010. Para isso, disponibilizou 9.000.000€ [9] com vista à aquisição dos serviços e bens necessários ao fornecimento, instalação e manutenção de quadros interactivos para essas escolas, de modo a obter um rácio de 1 quadro interactivo por cada 3 salas de aula em 2010.

1.1 Objectivo da tese

O objectivo desta tese foi projectar e implementar um sistema de suporte às aulas que permita ter projecções interactivas, criando um sistema de quadro interactivo de baixo custo, tendo como base o trabalho desenvolvido por Johnny Lee. O sistema deveria ter aplicações que permitissem ao docente desenhar e/ou anotar directamente sobre o

que é projectado, bem como, controlar o computador sem estar dependente do rato e do teclado deste.

1.2 Estrutura da dissertação

Para além deste capitulo, esta dissertação está estruturada do seguinte modo:

- No capitulo 2 é feita uma descrição das tecnologias existentes, bem como, apresentados alguns trabalhos relacionados;
- No capitulo 3 é descrito o sistema de projecções interactivas implementado, bem como, as aplicações de suporte ao sistema;
- No capitulo 4 são apresentadas as conclusões e o trabalho futuro.

Capítulo 2

Trabalhos relacionados

2.1 Impacto das TIC no ensino

As TIC, em particular os quadros interactivos, estão cada vez mais integradas nos sistemas de ensino dos países modernos. Em 2006, um estudo realizado em Inglaterra [10] apontou qual o impacto e as barreiras que surgiram no uso das TIC, mais especificamente dos quadros interactivos, no ensino. Ao nível dos estudantes:

- As TIC têm um impacto positivo no desempenho educacional nas escolas primárias, em particular no Inglês, mas menos nas Ciências, com a excepção da Matemática;

- O uso das TIC aumenta o desempenho dos alunos em Inglês (como língua mãe), Ciências, Desenho e Tecnologia entre as idades dos 7 e 16 anos, especialmente nas escolas primárias;

- Nos países da OCDE, existe uma associação positiva entre o tempo de uso das TIC e o desempenho dos alunos nos testes de Matemática do PISA (Program for International Student Assessment);

- Escolas com bons recursos de TIC alcançam melhores resultados que aquelas mal equipadas;

- A introdução de quadros interactivos levou a um maior aumento do desempenho dos estudantes nos exames nacionais de Inglês, Matemática e Ciências, do que as escolas sem quadros interactivos.

Além disto, foram também observados benefícios nos estudantes ao nível da motivação, competências, aprendizagem independente e trabalho de equipa.

Ao nível dos professores, o estudo mostrou um considerável número de provas do impacto das TIC:

- Um alto entusiasmo;
- Eficiência e colaboração elevada;
- Os quadros interactivos fazem a diferença nos aspectos de interacção na sala de aulas;
- Aumento das competências dos professores nas TIC.

Contudo, o estudo também identificou barreiras na adopção das TIC nas escolas:

- **Ao nível do professor:** A fraca competência de TIC dos professores, baixa motivação e falta de confiança para usar novas tecnologias no ensino são determinantes nos seus níveis de compromisso nas TIC.
- **Ao nível da escola:** O acesso limitado às TIC (devido à falta ou pobre organização de recursos TIC), fraca qualidade e manutenção inadequada de hardware bem como software educacional inadequado irá também definir o nível de utilização das TIC pelos professores. Ainda por mais, a falta de uma dimensão de TIC nas estratégias das escolas e as suas experiências limitadas com actividades/projectos suportadas pelas TIC, também é decisivo para determinar os níveis de utilização destas pelos professores.
- **Ao nível do sistema:** Em alguns países é o próprio sistema educacional e as suas estruturas rígidas que impedem a integração das TIC nas actividades de ensino do dia-a-dia.

2.2 Quadros Interactivos

Um quadro interactivo é um ecrã sensível ao toque que funciona em conjunto com um computador e um projector [11]. O primeiro quadro interactivo foi construído pela SMART Technologies Inc. em 1991. Existem três tipos de quadros interactivos: quadros de projecção frontal, quadros de projecção traseira [12], e ecrãs planos [13].

Os quadros interactivos de projecção frontal (ver Figura 2.1) têm um vídeo projector em frente ao quadro. Este tipo de quadro é muito mais barato que os quadros interactivos de projecção traseira. As desvantagens de quadros deste tipo são [5]:

1. o facto de quem está a apresentar ter de permanecer em frente ao quadro faz com que o seu corpo projecte sobras sobre o quadro;
2. se o apresentador estiver a falar para a audiência está sujeito a ser encadeado pelo feixe do projector;
3. normalmente não trazem o projector incluído.

No entanto, alguns quadros interactivos já vêm com o projector incluído colocado numa posição superior com um ângulo de projecção de aproximadamente 45º de modo a anular a primeira desvantagem.

Os quadros interactivos de projecção traseira (ver Figura 2.2) têm o projector colocado atrás da superfície do quadro de modo a que não ocorram sombras nem que o apresentador seja encadeado com a luz do projector. No entanto, este tipo de quadros interactivos é mais caro e muito volumoso, o que impossibilita a sua montagem numa parede. Entretanto, é possível embuti-los numa parede de modo a não ocupar muito espaço.

Os painéis planos [13] (ver Figura 2.3) são quadros interactivos em que a área de interacção é um ecrã LCD ou plasma.

Recentemente, tem havido um grande interesse na educação para inserir os quadros interactivos nos métodos de ensino, tendo surgido aspectos positivos em relação à sua utilização [5]:

- Facilitam a colaboração com colegas e parceiros;
- Recorre-se a desenhos para que a turma possa visualizar em conjunto, pois a informação visual é partilhada e entendida mais facilmente;
- Maior motivação dos alunos, pois estes gostam de interagir fisicamente com o quadro, manipulando texto e imagens, fornecendo por isso, mais oportunidades para interacção e discussão;
- São notadas também vantagens psicológicas como o aumento do planeamento e preparação, na marcação e avaliação, em guardar e editar lições, no estilo de

Figura 2.1: Quadro interactivo de projecção frontal (retirado de [1]).

Figura 2.2: Quadro interactivo de projecção traseira (retirado de [2]).

Figura 2.3: Painel plano (retirado de [3]).

ensino, na sensibilização de estilos de ensino, no planear para o desenvolvimento cognitivo, na clara representação visual de conceitos e nas actividades que encorajam uma abordagem de pensamento activa;

- Compromisso: Há um aumento na motivação, credibilidade, validade e focalização da turma;

- Aspectos socioculturais: Contribuem para uma melhor interacção social e um melhor trabalho de equipa;

- Tecnologia: O recurso a *drag-and-drop*, *esconder-e-revelar*, *faça-a-correspondência* e a utilização de movimento são boas maneiras de interacção entre os alunos e o quadro interactivo.

No entanto, devido às grandes limitações dos quadros interactivos, tais como o alto preço (ver Tabela 2.1) e a mobilidade, existe uma necessidade urgente de novas tecnologias para encontrar soluções que tenham um nível de desempenho similar, mas com um custo muito mais baixo. Uma solução existente é o sistema eBeam [14] (ver Figura 2.4). O dispositivo receptor do sistema eBeam é um dispositivo compacto, portátil e fácil de utilizar que torna qualquer superfície lisa, ou quadro branco, num quadro interactivo. O sistema interactivo eBeam pesa menos de 200g, é instalado em minutos, é amovível e tem um preço de 665€ sem o projector. Uma solução idêntica ao sistema eBeam que apareceu recentemente no mercado é o sistema mimio Interactive (ver figura 2.5 [15]) que custa cerca de 595€ sem o projector [16]. Apesar destes dispositivos já terem um preço em conta, comparado com um quadro interactivo

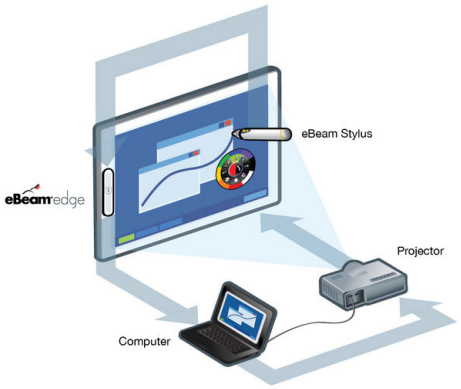


Figura 2.4: Sistema eBeam.

tradicional, ainda têm um preço considerável, pois quando se pensa em várias salas de aula o preço ainda é elevado (i.e., nº salas×600€).

Modelo	Reconhecimento de escrita	Projecção	Projector incluído	Preço
eBeam Integral 65	Não	Frontal	Não	790€
InterWrite 1071	Sim	Frontal	Não	1142€
Activboard 95 studio	Sim	Frontal	Não	1890€
SMARTBoard ESP680-N	Sim	Frontal	Sim	3390€
SMARTBoard 2000i	Sim	Traseira	Sim	7090€

Tabela 2.1: Comparação de características de cinco quadros interactivos (retirado de [5]).

2.3 Wiimote Whiteboard

Para fazer face às limitações dos quadros interactivos apresentadas no capitulo anterior (i.e., preço alto e mobilidade) surgiu recentemente, uma solução baseada numa caneta com um emissor de infravermelhos (IV) e uma câmara de IV. Neste caso a caneta de IV que feita com os componentes mais básicos fica com um custo a rondar os 5€ (ver figura 2.6) e, para a câmara, podemos usar o comando Wiimote, que custa cerca de 40€. Este sistema de nome Wiimote Whiteboard foi desenvolvido por Johnny Lee e teve uma grande divulgação pela Internet [17].

O Wiimote é o comando da consola Nintendo Wii e pode ser ligado a qualquer

Figura 2.5: Sistema mimio Interactive.

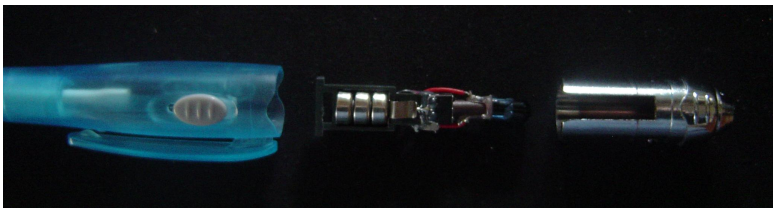


Figura 2.6: Caneta com emissor de IV.

computador através de Bluetooth, tal como um telemóvel. Entre várias características, o comando tem uma câmara de IV incorporada, capaz de detectar até quatro pontos emissores de IV, com uma resolução de 1024x768 pixeis, uma taxa de actualização de 100Hz e um ângulo de visão de 45º na horizontal [5, 17, 18]. Considerando a Figura 2.7 e usando a equação 2.1 é possível calcular a distância a que o Wiimote deve estar da projecção.

$$\tan 22.5^{\circ} = \frac{0.5l}{x} \iff x = 1.2l \tag{2.1}$$

Na equação 2.1, x é a distância do Wiimote à superfície da projecção e l a largura da área de projecção. Assim, podemos concluir que o Wiimote deve ser colocado a uma distância igual a 1.2× a largura da projecção. Entretanto, a câmara sofre do mesmo problema dos projectores, isto é, se o apresentador estiver entre a câmara e o emissor de IV, poderá detectar a posição do emissor incorrectamente, ou mesmo não detectá-la. Se a projecção for frontal, e o projector for colocado numa posição alta, colocar o Wiimote junto do projector (ver Figura 2.8) é uma boa solução.

O sistema desenvolvido por Johnny Lee permite ao apresentador calibrar a área de projecção sempre que o entender, usando quatro pontos de referência (ver figura 2.9). O sistema indica ainda alguma informação sobre o comando (ver figura 2.10) tal como, o estado da bateria, o total de fontes IV detectadas e a utilização de rastreamento.

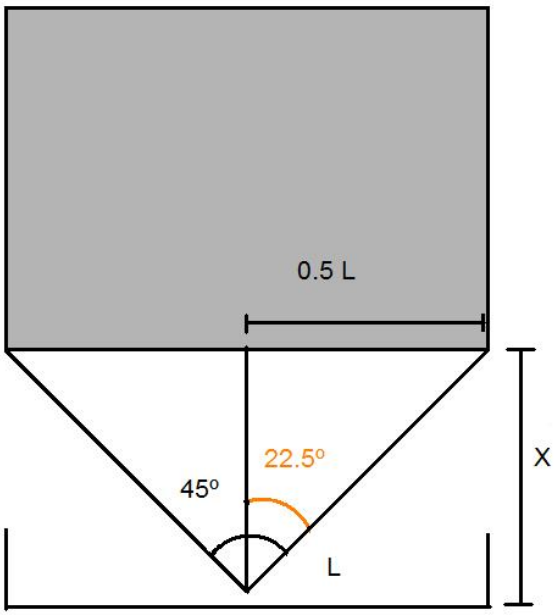


Figura 2.7: Distância do Wiimote em relação à projecção.

Figura 2.8: Posicionamento do Wiimote junto ao projector (retirado de [4]).

Figura 2.9: 1º ponto de calibração da área de projecção.

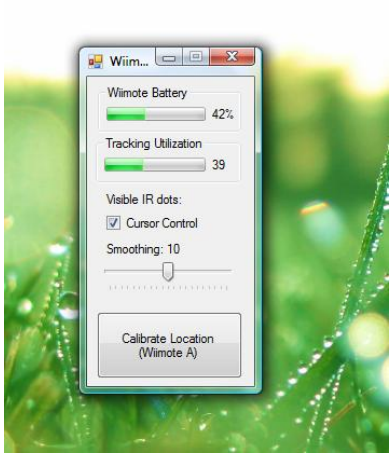


Figura 2.10: Janela da aplicação Wiimote Whtiteboard.

Também permite ao apresentador configurar a suavidade da detecção do emissor de IV, bem como activar ou desactivar o controlo do cursor. No entanto, este sistema tem uma grande limitação: só permite emular o evento de clique do botão esquerdo do rato. Assim um dos objectivos deste trabalho foi também tentar eliminar esta limitação.

Capítulo 3

Trabalho desenvolvido

O trabalho desenvolvido foi a implementação de um sistema de projecções interactivas, usando como base o trabalho de Johnny Lee. Além disso, foi desenvolvida uma aplicação que permite controlar o computador à distância, contribuindo assim para tornar as projecções mais interactivas.

Foram assim desenvolvidas duas aplicações para tornar as projecções interactivas (ou seja, a aplicação iiNote e a aplicação eeNote), bem como, uma aplicação de suporte às aulas de Introdução à Programação (a aplicação iiProgramming). A aplicação iiNote permite tirar apontamentos sobre o que é apresentado enquanto a aplicação eeNote permite editar as anotações capturadas pela aplicação iiNote. A aplicação iiProgramming é uma aplicação visual que permite gerar pseudo-código e código fonte com base no fluxograma de um algoritmo.

3.1 Implementação do sistema de projecções interactivas

Para anular a limitação do sistema desenvolvido por Johnny Lee, que apenas emula o clique do botão esquerdo do rato, e para que quem esteja a fazer uma apresentação possa estar fisicamente próxima da superfície de projecção (à semelhança do tradicional quadro), bem como não estar dependente do teclado e do rato do computador, foi desenvolvida uma aplicação em C# que visa anular estas limitações do sistema.

A aplicação permite ao apresentador, através de um segundo Wiimote, ter o controlo sobre algumas teclas chaves do teclado, bem como dos botões esquerdo e direito do rato (ver Figura 3.1). Esta aplicação usa a API WiimoteLib [19] que permite a uma

Figura 3.1: Mapeamento de teclas no Wiimote.

aplicação .NET (C# e VB.NET) ligar-se e comunicar com um ou vários Wiimotes. Para poder usar os eventos do teclado e rato foi necessário importar as funções da biblioteca user32 contidas no ficheiro user32.dll:

- Para os eventos do teclado:

```
[DllImport("user32.dll")]
public static extern void keybd_event(byte bVk, byte
    bScan, long dwFlags, long dwExtraInfo);
```

  Onde [20]:

  - bVk representa o código da tecla virtual (ver tabela 3.1 para os códigos utilizados [21]);
  - bScan representa o código do hardware de entrada para a tecla, neste caso 0x45;
  - dwFlags especifica vários aspectos da operação da função, podendo sendo um ou mais dos seguintes valores: (1) KEYEVENTF_EXTENDEDKEY, o código bScan foi precedido por um prefixo de byte com o valor 0xE0, ou (2) KEYEVENTF_KEYUP (0x02), a tecla está a ser solta.

- Para os eventos do rato:

```
[DllImport("user32.dll")]
```

```
static extern bool GetCursorPos(ref System.Drawing.Point
    lpPoint);
[DllImport("user32.dll")]
private static extern void mouse_event(long dwFlags,
   long dx, long dy, long dwData, long dwExtraInfo);
```

Onde:

- GetCursorPos é a função que permite obter as coordenadas X e Y do ecrã do cursor do rato e armazená-las numa variável do tipo System.Drawing.Point;
- mouse_event é a função que permite invocar os eventos do rato, onde [22]:
  - dwFlags representa o tipo de evento do rato (ver tabela 3.2 para os eventos utilizados [23]);
  - dx e dy representam as coordenadas X e Y do ecrã do cursor do rato;
  - dwData representa o movimento do scroll do rato;
  - dwExtraInfo especifica valor adicional associado ao evento do rato.

Nome simbólico da constante	Valor (hexadecimal)	Equivalentes do teclado
VK_TAB	0x09	Tecla TAB
VK_RETURN	0x0D	Tecla ENTER
VK_LEFT	0x25	Tecla direccional Esquerda
VK_UP	0x26	Tecla direccional Cima
VK_RIGHT	0x27	Tecla direccional Direita
VK_DOWN	0x28	Tecla direccional Baixo
VK_LWIN	0x5B	Tecla Windows
VK_ESC	0x1B	Tecla ESCAPE
VK_APPS	0x5D	Tecla Aplicações

Tabela 3.1: Códigos virtuais das teclas

Estando definidos os códigos das teclas virtuais e dos eventos do rato, estabelece-se a ligação ao Wiimote através da WiimoteLib e consulta-se o seu estado para, então, chamar as funções *mouse_event* e *keybd_event*. Para fazer a ligação ao Wiimote usando a WiimoteLib:

Nome simbólico da constante	Valor (hexadecimal)	Equivalentes do rato
MOUSEEVENTF_LEFTDOWN	0x02	Botão esquerdo pressionado
MOUSEEVENTF_LEFTUP	0x04	Botão esquerdo solto
MOUSEEVENTF_RIGHTDOWN	0x08	Botão direito pressionado
MOUSEEVENTF_RIGHTUP	0x10	Botão direito solto

Tabela 3.2: Códigos virtuais dos eventos do rato

- É inicializado um objecto do tipo *Wiimote*;

```
Wiimote wm = new Wiimote();
```

- Estabelece-se a ligação;

```
wm.Connect();
```

- É atribuído o evento *wm_WiimoteChanged* que é chamado sempre que o estado do Wiimote é actualizado;

```
wm.WiimoteChanged += wm_WiimoteChanged;
```

- Por fim, é definido qual o tipo de relatório que o Wiimote vai gerar, neste caso o tipo *Buttons*. Assim, o relatório gerado devolve os dados apenas acerca dos botões [24]. Também é definido, por *true* ou *false*, se o relatório é constantemente gerado, ou apenas quando o estado do Wiimote sofre alterações.

```
wm.SetReportType(InputReport.Buttons, false);
```

Estando o Wiimote devidamente ligado ao computador, é no evento *wm_WiimoteChanged* que é invocada a função *UpdateWiimoteChanged* onde são chamadas as funções *mouse_event* e *keybd_event*, dependo do botão do Wiimote que é pressionado:

```
private void wm_WiimoteChanged(object sender,
    WiimoteChangedEventArgs args)
{
BeginInvoke(new UpdateWiimoteStateDelegate(
    UpdateWiimoteChanged), args);
}
```

Na função *UpdateWiimoteStateDelegate* é comparada a informação do estado actual do Wiimote com o estado anterior, sendo que se forem diferentes, significa que o utilizador pressionou um botão do Wiimote. Caso isto aconteça, é então chamada a função *mouse_event* ou *keybd_event*, dependendo do botão do Wiimote que é pressionado. Por exemplo, se o utilizador pressiona o botão 1 do Wiimote, queremos que o sistema entenda como o utilizador ter pressionado a tecla ENTER do teclado:

```
if (!lastWiiState.ButtonState.One && ws.ButtonState.One)
keybd_event(VK_RETURN, 0x45, 0, 0);
if (lastWiiState.ButtonState.One && !ws.ButtonState.One)
keybd_event(VK_RETURN, 0x45, KEYEVENTF_KEYUP, 0);
lastWiiState.ButtonState.One = ws.ButtonState.One;
```

Com esta aplicação a complementar o Wiimote Whiteboard, o apresentador pode controlar o computador à distância e tornar, deste modo, o quadro mais interactivo (ver Figura 3.2). Temos assim implementado, ao nível do hardware, um sistema de projecções interactivas. No entanto, para que o sistema funcione do mesmo modo que um quadro interactivo normal, é necessário ter software idêntico ao que vem incluído nos quadros interactivos. Por exemplo, uma aplicação que permita a quem esteja a utilizar o quadro interactivo possa desenhar e tirar apontamentos sobre o que é apresentado, de modo a que o quadro interactivo seja usado de forma similar a um quadro normal. Foram então desenvolvidas três aplicações para suporte do sistema de projecções interactivas, aplicações essas que serão descritas nas próximas secções.

3.2 Aplicação iiNote

Para que um quadro interactivo possa ser usado como um quadro normal que permite escrever sobre o que é apresentado, é normalmente disponibilizado software que torna isso possível. Uma aplicação disponível para utilização com o Wiimote Whiteboard que permita tirar apontamentos sobre o que é apresentado, bem como utilizar algumas funcionalidades do computador, tal como aceder a um teclado virtual, é a aplicação Smoothboard [25]. Sendo assim, foi desenvolvida uma aplicação similar (a aplicação iiNote) que contém um diverso leque de funcionalidades que o apresentador (utilizador) pode usar, tais como:

- permite escrever sobre o que está a ser apresentado (Figura 3.3a);

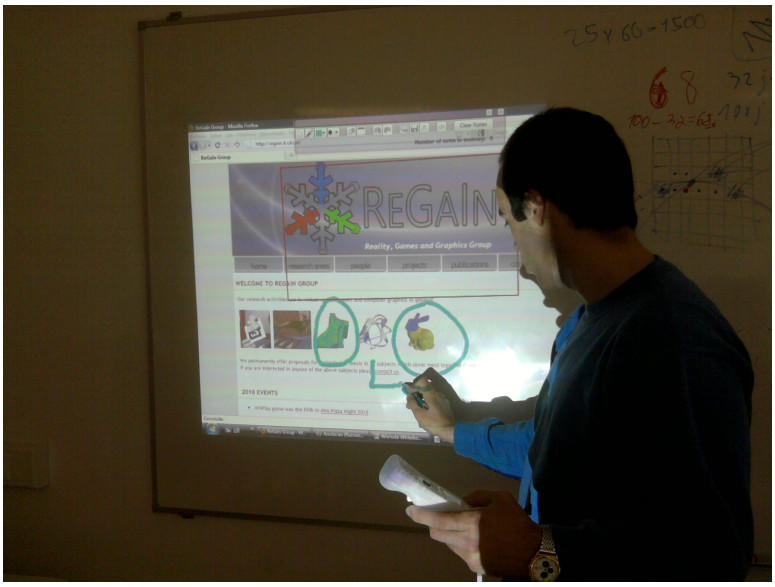
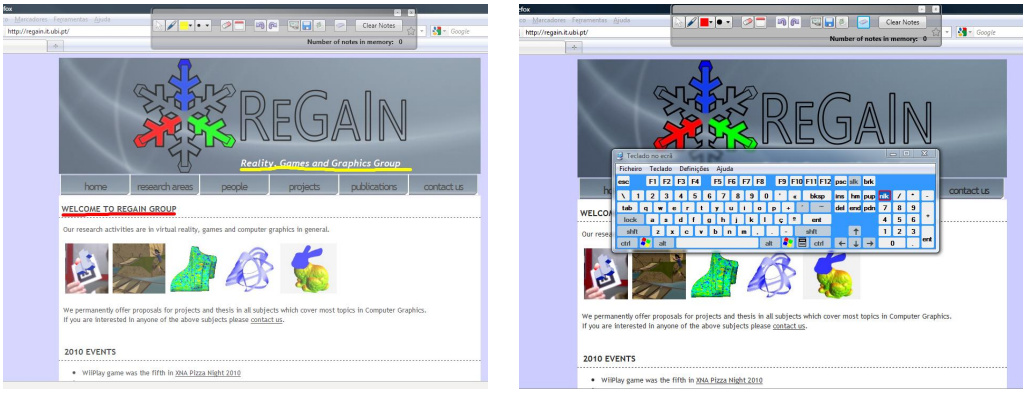


Figura 3.2: Exemplo de utilização do sistema.

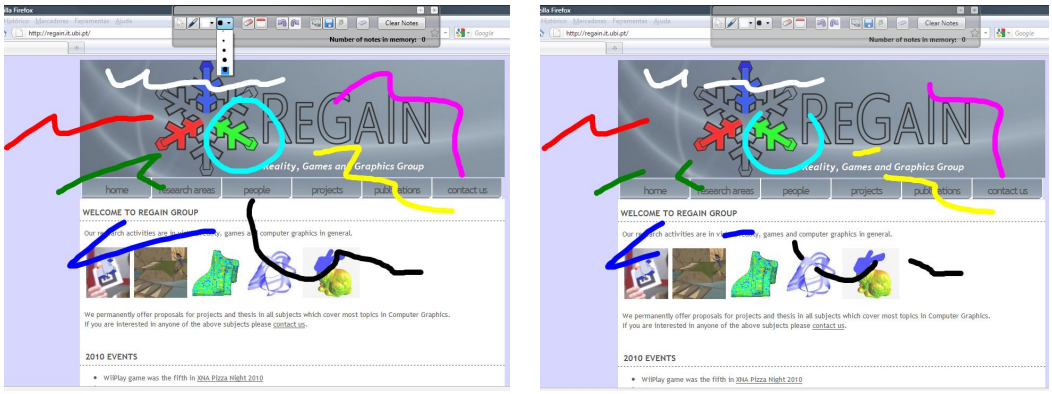
- permite ter acesso a um teclado virtual (Figura 3.3b);
- permite definir as cores e tamanhos para as anotações (Figura 3.4a);
- permite usar uma borracha para apagar anotações (Figura 3.4b), bem como um botão para limpar todas as anotações;
- Histórico de acções, de modo a que o utilizador possa desfazer/refazer acções;
- Capturar as anotações feitas durante a apresentação, armazenando-as em memória, informando o apresentador do número de notas em memória (Figura 3.5a), bem como limpar as notas em memória;
- Exportar as notas em memória para um documento XPS (XML Paper Specification) [26] (Figura 3.5b e 3.6) e/ou guardá-las num ficheiro com a extensão *.nota* para posterior edição (Figura 3.7) com a aplicação eeNote também desenvolvida.

Esta aplicação foi desenvolvida em WPF (Windows Presentation Foundation). O WPF é um subsistema gráfico para renderizar interfaces do utilizador (IU) em aplicações Windows, tendo sido disponibilizado pela primeira vez como parte da .NET Framework 3.0 [27]. O WPF é construído sobre DirectX, o que disponibiliza aceleração de hardware e permite interfaces modernas para o utilizador tal como transparência, gradientes e transformações. O WPF tem como objectivo unir um número de serviços de



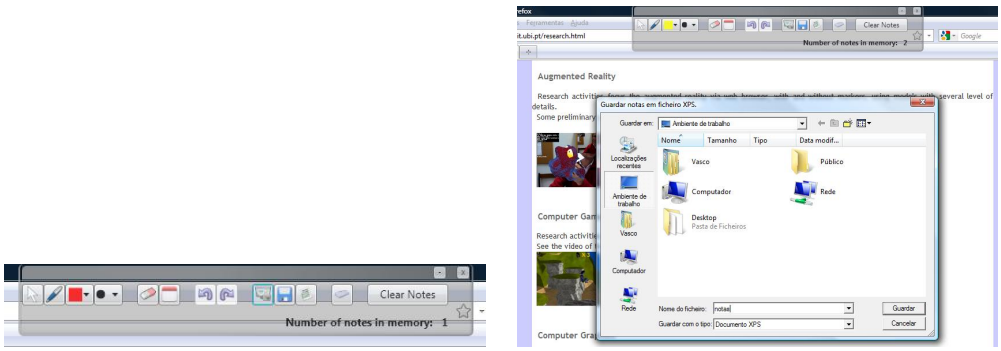
(a) (b)

Figura 3.3: *(a)* aplicação iiNote e *(b)* teclado virtual



(a) (b)

Figura 3.4: *(a)* cores e tamanhos das anotações e *(b)* uso da borracha



(a) (b)

Figura 3.5: *(a)* número de notas em memória e *(b)* exportar notas para um documento XPS

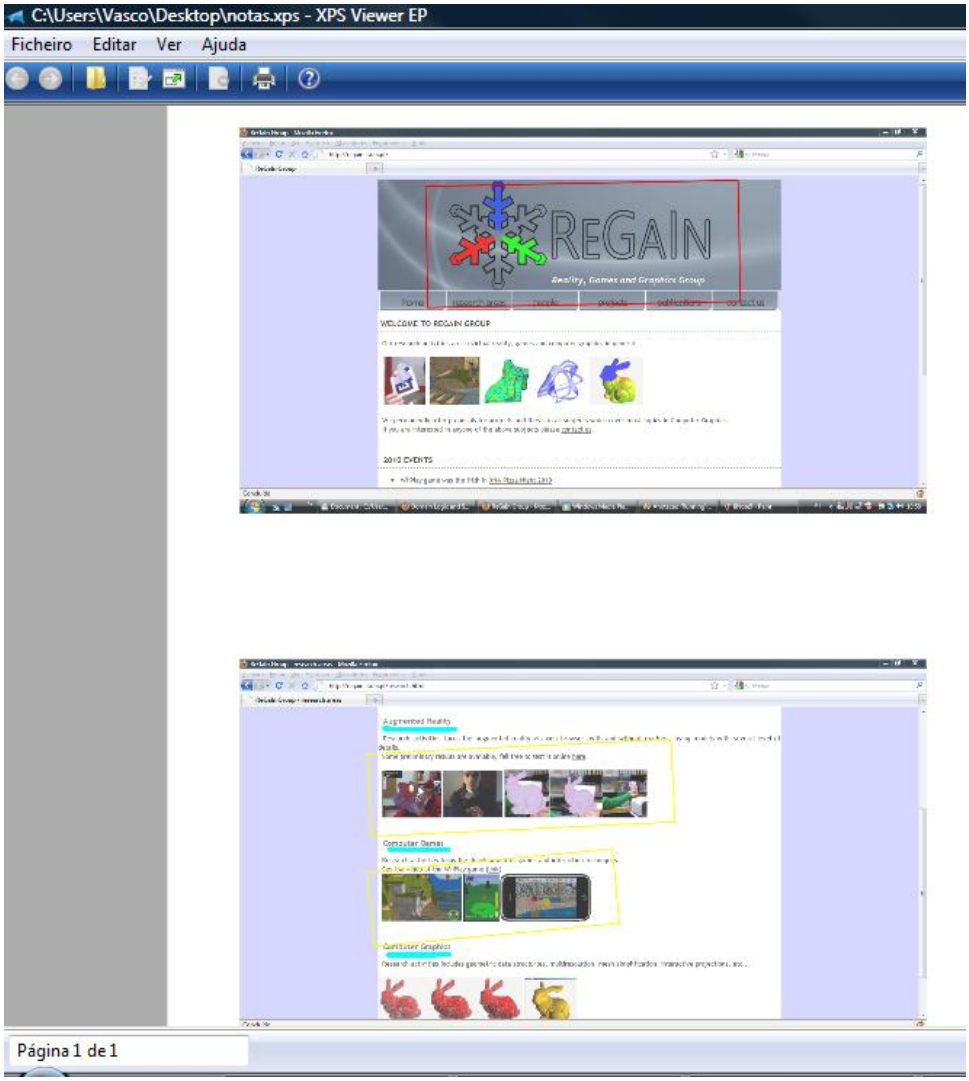


Figura 3.6: Documento XPS.

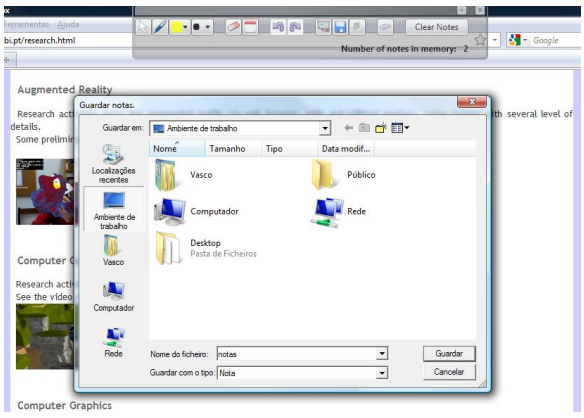


Figura 3.7: Guardar as notas em ficheiro.

aplicações: IU, desenho 2D e 3D, documentos fixos e adaptáveis, tipografia avançada, imagens vectoriais, imagens bitmap, animações, ligação de dados, áudio e vídeo. O WPF fornece um modelo de programação consistente para criar aplicações e fornece um clara separação entre a IU e o domínio lógico, sendo que a IU é definida usando uma nova linguagem, conhecida como XAML, e o domínio lógico é definido em C#. Das várias funcionalidades disponibilizadas pelo WPF, destaca-se a classe InkCanvas, a possibilidade de ter janelas no topo e a ocupar toda a área do ecrã, bem como utilizar a API InkAnalysis para Tablet PCs.

A aplicação iiNote é constituída pelos seguintes ficheiros:

- **App.xaml e App.xaml.cs:** Onde são definidas algumas propriedades da aplicação;
- **Quadro.xaml:** Onde é definida a IU do quadro onde o utilizador irá anotar;
- **Quadro.xaml.cs:** Onde são definidos alguns métodos que interagem com os elementos definidos na IU especificada no ficheiro Quadro.xaml;
- **Comandos.xaml:** Onde é definida a IU da janela principal da aplicação, onde estão os vários botões, tais como o marcador, a borracha e o teclado virtual;
- **Comandos.xaml.cs:** Onde são definidos alguns métodos e os eventos dos elementos definidos na IU especificada no ficheiro Comandos.xaml;
- **Notas.cs e Nota.cs:** Onde estão definidos alguns objectos que são utilizados pela aplicação.

3.2.1 Utilização do elemento InkCanvas

O InkCanvas é um elemento que pode ser usado para receber e mostrar tinta digital, usando uma caneta digital ou o rato e, através de um digitalizador, produzir desenhos [28]. Os desenhos criados são representados como objectos do tipo **Stroke** e podem ser manipulados, tanto através de um input do utilizador, como programando na própria aplicação. O InkCanvas permite aos utilizadores modificar ou apagar um desenho existente. Para permitir ao utilizador fazer as anotações sobre o que é apresentado, recorreu-se à utilização de uma janela contendo um elemento do tipo InkCanvas, janela essa onde é definida transparência para permitir ao utilizador ver o que está a ser apresentado enquanto faz a anotação, bem como colocar a janela no topo de todas as

outras e de modo a ocupar toda a área do ecrã. Utilização do InkCanvas no ficheiro Quadro.xaml:

```
<Window ... AllowsTransparency="True" Background="Transparent"
    WindowState="Maximized" ShowInTaskbar="False">
    <Grid x:Name="LayoutRoot">
        <! Para dar um aspecto esbranquiçado ao quadro >
        <Rectangle Fill="White" Opacity="0.195"/>
        <InkCanvas x:Name="whiteboard" Background="{x:Null}"/>
    </Grid>
</Window>
```

Esta janela é colocada visível no evento correspondente ao botão *marcador* sempre que este é pressionado pelo utilizador. Evento do botão *marcador* no ficheiro Comandos.xaml.cs:

```
public Quadro q = null;
private void marcador_Click(object sender, RoutedEventArgs e)
{
        //se o quadro ainda não tinha sido inicializado, é
            inicializado um novo
    if (q == null)
    {
        //define a cor e tamanho do marcador conforme os valor
            seleccionados
        string s = cor.SelectedValue.ToString();
        System.Windows.Media.ColorConverter cc = new System.
            Windows.Media.ColorConverter();
        System.Windows.Media.Color color = (System.Windows.Media.
            Color)cc.ConvertFrom(s);
        q = new Quadro(color, (tamanho.SelectedIndex + 1) * 2);

        //atribui um evento para quando o utilizador soltar o
            botão esquerdo do rato
        q.whiteboard.MouseLeftButtonUp += new
            MouseButtonEventHandler(whiteboard_MouseLeftButtonUp);

        //mostra o quadro
```

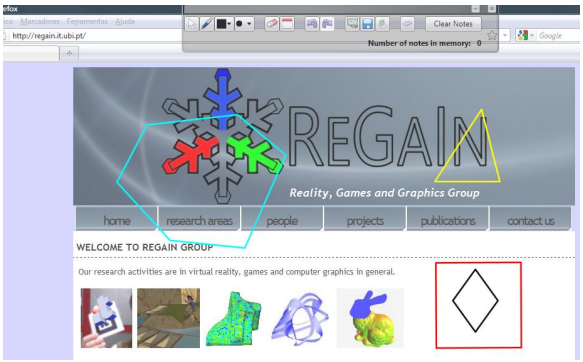


Figura 3.8: Formas geométricas reconhecidas.

```
            q.Show();
        }
        //senão, mostra a tela e desliga o modo Apagar
        else
            {
                    q.Show();
            q.setModeEraserOff();
        }
}
```

Cada desenho (ou anotação) feito no InkCanvas pelo utilizador é armazenado num objecto do tipo **Stroke**, sendo depois esses desenhos armazenados num objecto do tipo **StrokeCollection** no InkCanvas.

3.2.2 Reconhecimento de formas geométricas

A aplicação tem implementada um reconhecedor de formas geométricas de modo a que, no caso de o utilizador desenhar um rectângulo, por exemplo, esse mesmo rectângulo seja desenhado na perfeição (Figura 3.8). Este reconhecedor de formas geométricas foi implementado usando a API InkAnalysis. Esta API está disponível como parte do SDK Windows® que contém os componentes WinFX® [29]. A API InkAnalysis combina o reconhecimento de tinta, também conhecido como reconhecimento de escrita, análise de formas e classificação, que permite fazer o reconhecimento semanticamente de partes significativas tais como parágrafos, palavras ou desenhos. Sempre que o utilizador deixa de desenhar (i.e., solta o botão esquerdo do rato), é invocado o evento *whiteboard_MouseLeftButtonUp* (apresenta-se de seguida o seu código), no

qual será adicionada a nova acção ao histórico e será feito o reconhecimento de formas geométricas, ou seja, é feito o reconhecimento de todas as formas reconhecíveis pela API InkAnalysis, com a excepção de círculos, elipses e formas desconhecidas, de modo a não causar problemas quando o utilizador escreve texto. Evento whiteboard_MouseLeftButtonUp no ficheiro Comandos.xaml.cs:

```
private void whiteboard_MouseLeftButtonUp(object sender,
    MouseButtonEventArgs e)
{...
        //inicializa o analisador para formas geométricas
        InkAnalyzer analyzer = new InkAnalyzer(q.Dispatcher);
        analyzer.AddStrokes(q.whiteboard.Strokes);
        analyzer.SetStrokesType(q.whiteboard.Strokes, StrokeType.
            Drawing);
        analyzer.Analyze();

        //procura formas geométricas
        ContextNodeCollection paras = analyzer.FindNodesOfType(
            ContextNodeType.InkDrawing);

        //para cada forma geométrica encontrada
        foreach (InkDrawingNode drawingNode in paras)
        {
                //obter o nome da forma geométrica
                string name = drawingNode.GetShapeName();

                //se a forma não for nem Outro nem Circulo nem
                    Elipse
                if (name != "Other" && name != "Circle" && name !=
                     "Ellipse")
                {

                        //Obter os pontos de referência
                        PointCollection pontosquentes =
                            drawingNode.HotPoints;
```

```
//Colecção de pontos onde são armazenados
    os pontos de referência
StylusPointCollection pontosshape = new
    StylusPointCollection();
StylusPoint ponto = new StylusPoint();

//Copia os pontos de referência e colocar
    numa colecção de pontos
for (int i = 0; i < pontosquentes.Count; i
    ++)
{
        ponto.X = pontosquentes[i].X;
        ponto.Y = pontosquentes[i].Y;
        pontosshape.Add(ponto);
}

ponto.X = pontosquentes[0].X;
ponto.Y = pontosquentes[0].Y;
pontosshape.Add(ponto);

//cria uma Stroke com os pontos de
    referência obtidos
Stroke stroke = new Stroke(pontosshape);

//Define a cor e tamanho da Stroke
stroke.DrawingAttributes.Color =
    drawingNode.Strokes[0].
    DrawingAttributes.Color;
stroke.DrawingAttributes.Height =
    drawingNode.Strokes[0].
    DrawingAttributes.Height;
stroke.DrawingAttributes.Width =
    drawingNode.Strokes[0].
    DrawingAttributes.Width;

//substitui a forma antiga pela nova
```

```
                        q.whiteboard.Strokes.Remove(drawingNode.
                            Strokes[drawingNode.Strokes.Count - 1])
                            ;
                        q.whiteboard.Strokes.Add(stroke);
                    }
                }
                //é adicionada a acção ao histórico
...
}
```

3.2.3 Captura das anotações

À medida que o utilizador faz a sua apresentação, muitas vezes, quer seja por dúvidas postas pela assistência, quer seja pelo o que é apresentado não estar completo ou se pretender fazer um esquema, tem de fazer anotações sobre o que é apresentado. Sempre que o utilizador pede à aplicação para capturar uma anotação, a aplicação simplesmente tira um screenshot ao ecrã, guardando esse screenshot como imagem, como acontece, por exemplo, na aplicação Smoothboard. Isto implica que o apresentador, caso queira partilhar essas anotações rapidamente no final da apresentação, tenha de perder tempo a, por exemplo, juntar essas imagens, num ficheiro comprimido *.zip* ou *.rar*, ou perder tempo a juntar essas imagens num único documento (e.g., Word).

A aplicação iiNote, para poder optimizar o tempo da apresentação, permite ao apresentador capturar as anotações com um simples clique sobre o botão *capturar*, ficando as anotações em memória. Quando o utilizador o pretender, poderá eliminar essas anotações em memória, guardá-las num ficheiro *.nota* ou exportá-las para um documento XPS.

Sempre que o utilizador pressiona o botão *capturar*, a aplicação iiNote esconde temporariamente a janela dos comandos para, então, tirar um screenshot ao ecrã (incluindo as anotações), tirar um screenshot ao que é apresentado (sem as anotações), bem como guardar as anotações (Strokes). Os screenshots e as anotações são guardados num objecto *Nota*. Definição do objecto *Nota*:

```
[Serializable()]
public class Nota
{
```

```
        public Bitmap fundo { get; set; }
        public Bitmap fundo_completo { get; set; }
        public System.Windows.Point[][] notas { set; get; }
        public List<string> cor { get; set; }
        public List<double> tamanho { get; set; }

        public Nota(int i)
        {
                this.fundo = null;
                this.notas = new System.Windows.Point[i][];
                this.cor = new List<string>();
                this.tamanho = new List<double>();
                this.fundo_completo = null;
        }
}
```

onde:

- **fundo** é a variável onde é armazenado o screenshot do que é apresentado (sem as anotações);
- **fundo_completo** é a variável onde é armazenado screenshot do ecrã (com as anotações);
- **notas** é a variável onde são armazenadas as anotações (StrokeColletion). Como a classe Stroke não é serializável, foi necessário converter as Strokes para o tipo Point[][];
- **cor** é a variável onde é guardada a informação referente à cor de cada Stroke;
- **tamanho** é a variável onde é guardada a informação referente ao tamanho de cada Stroke;

A captura das anotações é feita no evento printscreen_Click do botão *capturar* no ficheiro Comandos.xaml.cs:

```
public List<Nota> notas = new List<Nota>(10);
public StrokeCollection inkcanvas = null;
private void printscreen_Click(object sender, RoutedEventArgs e)
```

```
{...
        //esconde a janela dos comandos
        this.janela.Hide();

        //Tira screenshot à nota completa e guarda no Bitmap bmp
        System.Drawing.Size sz = Screen.PrimaryScreen.Bounds.Size;
        IntPtr hDesk = GetDesktopWindow();
        IntPtr hSrce = GetWindowDC(hDesk);
        IntPtr hDest = CreateCompatibleDC(hSrce);
        IntPtr hBmp = CreateCompatibleBitmap(hSrce, sz.Width, sz.
           Height);
        IntPtr hOldBmp = SelectObject(hDest, hBmp);
        bool b = BitBlt(hDest, 0, 0, sz.Width, sz.Height, hSrce,
           0, 0, CopyPixelOperation.SourceCopy |
           CopyPixelOperation.CaptureBlt);
        Bitmap bmp = Bitmap.FromHbitmap(hBmp);
        SelectObject(hDest, hOldBmp);
        DeleteObject(hBmp);
        DeleteDC(hDest);
        ReleaseDC(hDesk, hSrce);

        //esconde a tela
        this.q.Hide();

        //Tira screenshot ao fundo e guarda no Bitmap bmp2
        System.Drawing.Size sz2 = Screen.PrimaryScreen.Bounds.Size
           ;
        IntPtr hDesk2 = GetDesktopWindow();
        IntPtr hSrce2 = GetWindowDC(hDesk2);
        IntPtr hDest2 = CreateCompatibleDC(hSrce2);
        IntPtr hBmp2 = CreateCompatibleBitmap(hSrce2, sz2.Width,
           sz2.Height);
        IntPtr hOldBmp2 = SelectObject(hDest2, hBmp2);
        bool b2 = BitBlt(hDest2, 0, 0, sz2.Width, sz2.Height,
           hSrce2, 0, 0, CopyPixelOperation.SourceCopy |
           CopyPixelOperation.CaptureBlt);
```

```
Bitmap bmp2 = Bitmap.FromHbitmap(hBmp2);
SelectObject(hDest2, hOldBmp2);
DeleteObject(hBmp2);
DeleteDC(hDest2);
ReleaseDC(hDesk2, hSrce2);

//copia a StrokeCollection da tela
inkcanvas = q.whiteboard.Strokes.Clone();
//Inicializa uma Nota e armazena as imagens capturadas
Nota nn = new Nota(inkcanvas.Count);
nn.fundo = bmp2;
nn.fundo_completo = bmp;
//Converte a StrokeColletion em Point[][] e guarda na Nota
for (int i = 0; i < inkcanvas.Count; i++)
{
        nn.notas[i] = new System.Windows.Point[inkcanvas[i
            ].StylusPoints.Count];
        nn.cor.Add(inkcanvas[i].DrawingAttributes.Color.
            ToString());
        nn.tamanho.Add(inkcanvas[i].DrawingAttributes.
            Width);

        for (int j = 0; j < inkcanvas[i].StylusPoints.
            Count; j++)
        {
                nn.notas[i][j] = new System.Windows.Point
                    ();
                nn.notas[i][j].X = inkcanvas[i].
                    StylusPoints[j].X;
                nn.notas[i][j].Y = inkcanvas[i].
                    StylusPoints[j].Y;
        }
}
//adiciona a Nota a uma lista de Notas
notas.Add(nn);
//actualiza o número de notas em memória
```

```
            n_notas.Content = notas.Count.ToString();
            //informa o utilizador que a nota foi armazenada
            System.Windows.MessageBox.Show("Nota armazenada!");

            //mostra a janela principal
            janela.Show();
...
}
```

3.2.4 Exportação das anotações para um documento XPS

Um documento XPS (XML Paper Specification) é um formato criado pela Microsoft para concorrer com o PDF (Portable Document Format) da Adobe [30]. Um documento XPS tem algumas vantagens sobre um documento PDF:

- porque este formato é baseado em XML e aderiu ao Open Packaging Conventions [31], integra-se facilmente com outras tecnologias, tais como a tecnologia de gestão dos direitos digitais da Microsoft, e a tecnologia de compressão ZIP;
- os documentos XPS são independentes de software e hardware e, por isso, não precisão de um leitor em separado (como o Adobe Acrobat) para abrir e ler documentos XPS (basta usar o Internet Explorer);
- XPS integra mais facilmente documentos em aplicações de fluxo (workflow applications);
- a maior capacidade de renderizar com mais fidelidade conteúdo no ecrã, ou seja, por exemplo, imagens são armazenadas no documento sem perda de resolução;
- Todos os documentos são capazes de imprimir para XPS sem um arquivo intermediário conversor, como o Acrobat PDF. O XPS visa integrar esta funcionalidade em todas as aplicações Windows e não apenas em aplicações profissionais ou aplicações high-end.

Estas razões levaram a escolha deste formato para a exportação das anotações. Ao exportar para XPS, a aplicação iiNote usa a API InkAnalysis para reconhecer texto nas várias notas. Assim, se o utilizador escrever apontamentos nas notas, a aplicação

Figura 3.9: Documento XPS com texto reconhecido.

reconhece o texto e adiciona-o ao documento XPS (ver Figura 3.9). A exportação das anotações é feita no evento xps_Click do botão *xps* no ficheiro Comandos.xaml.cs:

```
private void xps_Click(object sender, RoutedEventArgs e)
{
        //declaração das variáveis
        FlowDocument documento = new FlowDocument();
        BlockUIContainer bloco = null;
        Figure fi = null;
        StringBuilder text = null;
        InkAnalyzer analyzer2 = null;
        InkAnalyzer analyzer = null;
        ContextNodeCollection paras = null;
        Paragraph p = null;
        List<Paragraph> paragrafos = new List<Paragraph>();

        //para cada Nota armazenada, passar a nota_completa para o
            documento e texto reconhecido
        for (int i = 0; i < notas.Count; i++)
        {
```

```
text = new StringBuilder();
//inicialização dos analisadores do InkCanvas
analyzer2 = new InkAnalyzer(this.Dispatcher);
analyzer = new InkAnalyzer(this.Dispatcher);

//análise dos Strokes
analyzer.AddStrokes(toStrokes(notas[i]));
analyzer.Analyze();
//procurar por parágrafos de texto.
paras = analyzer.FindNodesOfType(ContextNodeType.
   Paragraph);
//por cada parágrafo são analisadas as linhas
foreach (ContextNode paragraph in paras)
{
        p = new Paragraph();
        p.FontSize = 12;
        p.FontWeight = FontWeights.Bold;
        //por cada linha é feito o reconhecimento
           de texto
        foreach (ContextNode line in paragraph.
           SubNodes)
        {
                analyzer2.AddStrokes(line.Strokes)
                   ;
                analyzer2.Analyze();
                text.AppendLine(analyzer2.
                   GetRecognizedString());
                p.Inlines.Add(new Run(analyzer2.
                   GetRecognizedString()));
        }
        //é juntado o texto reconhecido ao já
           existente
        paragrafos.Add(p);
        text.AppendLine();
}
```

```
                    //criação de um bloco de um FlowDocument
                    bloco = new BlockUIContainer();
                    //cria figura com o conteúdo do whiteboard
                    System.Windows.Controls.Image im = new System.
                        Windows.Controls.Image();
                    im.Source = System.Windows.Interop.Imaging.
                        CreateBitmapSourceFromHBitmap(notas[i].
                        fundo_completo.GetHbitmap(), IntPtr.Zero,
                        Int32Rect.Empty,
                    System.Windows.Media.Imaging.BitmapSizeOptions.
                        FromEmptyOptions());
                    //adiciona a imagem ao bloco
                    bloco.Child = im;
                    //é criada uma figura para o FlowDocument
                    fi = new Figure();
                    //adiciona o bloco que contem a imagem à figura
                    fi.Blocks.Add(bloco);
                    //criação de um paragrafo
                    Paragraph para = new Paragraph();
                    //adiciona a figura ao paragrafo
                    para.Inlines.Add(fi);
                    //adiciona o parágrafo ao documento
                    documento.Blocks.Add(para);
                    //adiciona o texto reconhecido ao FlowDocument
                    if (paragrafos.Count > 0)
                            documento.Blocks.Add(paragrafos[paragrafos
                                .Count    1]);

            }
            //guardar o documento
            Save(documento);
}
```

Método Save para guardar um FlowDocument em XPS:

```
public static void Save(FlowDocument flowDocument)
{
```

```
Microsoft.Win32.SaveFileDialog saveFileDialog1 = new
    Microsoft.Win32.SaveFileDialog();
saveFileDialog1.Filter = "Documento XPS|*.xps";
saveFileDialog1.Title = "Guardar notas em ficheiro XPS.";
if (saveFileDialog1.ShowDialog()==true)
{
        if (saveFileDialog1.FileName != "")
        {
                //apaga ficheiro existente
                DeleteOldFile(saveFileDialog1.FileName);

                SerializerProvider serializerProvider =
                    new SerializerProvider();
                SerializerDescriptor selectedPlugIn = null
                    ;
                foreach (SerializerDescriptor
                    serializerDescriptor in
                    serializerProvider.InstalledSerializers
                    )
                {
                        if (!serializerDescriptor.
                            IsLoadable || !saveFileDialog1.
                            FileName.EndsWith(
                            serializerDescriptor.
                            DefaultFileExtension))
                        continue;
                        selectedPlugIn =
                            serializerDescriptor;
                        break;
                }
                if (selectedPlugIn != null)
                {
                        using (Stream package = File.
                            Create(saveFileDialog1.FileName
                            ))
                        {
```

```
                                        SerializerWriter
                                            serializerWriter =
                                            serializerProvider.
                                            CreateSerializerWriter(
                                            selectedPlugIn, package
                                            );
                                        IDocumentPaginatorSource
                                            idoc = flowDocument;
                                        if (idoc != null)
                                                serializerWriter.
                                                    Write(idoc.
                                                    DocumentPaginator
                                                    , null);
                                        package.Close();
                                        System.Windows.MessageBox.
                                            Show("Ficheiro criado
                                            com sucesso.");
                                    }
                                }
                                else
                                throw new Exception("Não foi encontrado um
                                     Serializer para o formato escolhido.")
                                    ;
                            }
                    }
            }
            //apaga ficheiro existente
            private static void DeleteOldFile(string fileName)
            {
                    if (File.Exists(fileName))
                    {
                            File.Delete(fileName);
                    }
            }
```

A funcionalidade de exportação das anotações para um documento XPS permite assim, que o utilizador possa gerar rápida e facilmente um documento de texto com

as anotações tiradas durante uma apresentação, podendo partilhar o documento mais facilmente e mais rápido com a assistência. Deste modo, a audiência não precisa de se preocupar com as anotações feitas durante a apresentação pelo apresentador, concentrando-se totalmente na apresentação.

3.3 Aplicação eeNote

De modo a permitir ao apresentador completar, editar, ou apagar as anotações efectuadas pela aplicação iiNote, foi criada a aplicação eeNote. A aplicação eeNote (ver Figura 3.10) tem incluída algumas das características da aplicação iiNote, bem como algumas novas:

- abrir/guardar ficheiros com a extensão *.nota*;
- usar o marcador para fazer novas anotações;
- reconhecimento de formas geométricas;
- seleccionar cores e tamanhos para o marcador;
- usar uma borracha para apagar anotações, bem como um botão limpar todas as anotações;
- histórico de acções, de modo a que o apresentador possa desfazer/refazer acções;
- seleccionar anotações e apagar/redimensionar (Ver Figura 3.11);
- apagar uma nota completa;
- exportar as anotações para um documento XPS, fazendo o reconhecimento de escrita.

Deste modo, o apresentador pode fazer uma apresentação sem se preocupar se as anotações ficam incompletas ou imperfeitas, dado que tem um modo de as editar à posterior.

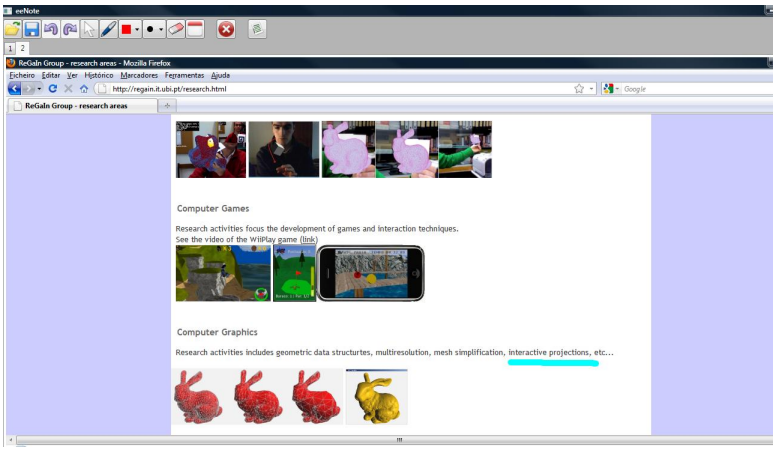


Figura 3.10: Aplicação eeNote

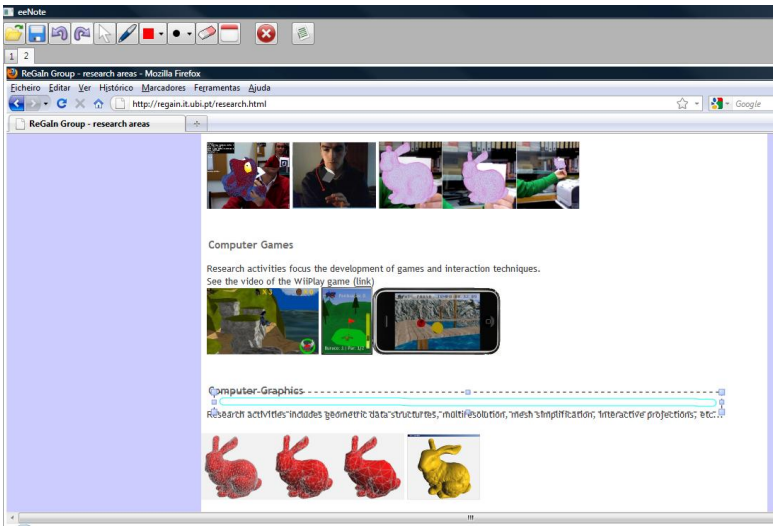


Figura 3.11: Seleccionar uma anotação e redimensioná-la.

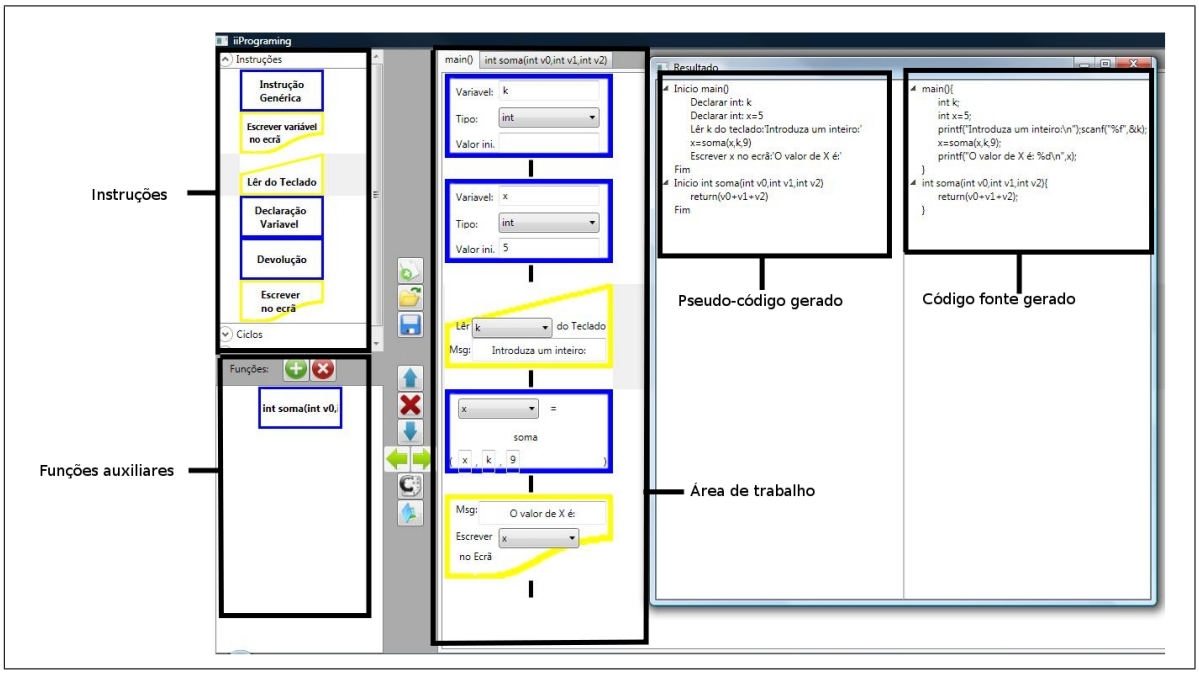


Figura 3.12: Aplicação iiProgramming.

3.4 Aplicação iiProgramming

A grande parte dos sistemas de quadros interactivos tem incluído algum software educativo tal como, por exemplo, uma aplicação que mostre o sistema solar para ensinar astronomia aos alunos. Neste caso, foi criada uma aplicação de suporte às aulas de Introdução à Programação: a aplicação iiProgramming.

A aplicação iiProgramming é uma aplicação visual que permite criar fluxogramas e gerar pseudo-código e código fonte em Linguagem C (ver Figura 3.12). O fluxograma é a representação gráfica de um algoritmo, descrevendo o fluxo dum algoritmo através de um conjunto de figuras geométricas padronizadas (ver Figura 3.13) ligadas por setas de fluxo. Em meados da década de 60, alguns matemáticos provaram que qualquer programa podia ser construído através da combinação de 3 estruturas básicas: sequência (um bloco de instruções), selecção (IF-THEN-ELSE, IF-THEN e SWITCH-CASE) e repetição (FOR, WHILE e DO-WHILE) [32]. Uma sequência tanto pode ser um bloco de instruções da função principal (main()) como de uma função criada pelo utilizador, o bloco de instruções de um teste IF-THEN, IF-THEN-ELSE, o bloco do SWITCH-CASE, ou o bloco de instruções das repetições FOR, WHILE E DO-WHILE. O utilizador da aplicação pode escolher a instrução desejada de uma lista e arrastá-

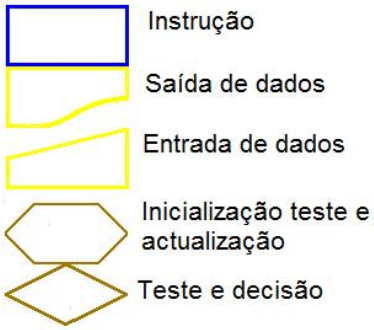


Figura 3.13: Figuras geométricas utilizadas num fluxograma.

la (através de drag-and-drop) para a posição pretendida para uma ListBox (isto é, para a uma sequência de instruções). As instruções disponibilizadas ao utilizador, por categoria, são:

- Instruções:
  - instrução genérica;
  - escrever variável no ecrã (printf());
  - ler variável do teclado (scanf());
  - declarar uma variável do tipo int, float, char ou char[];
  - devolução (return());
  - escrever mensagem no ecrã (printf());
- Ciclos:
  - while;
  - do while;
  - for;
- Testes:
  - if-then;
  - if-then-else;
  - switch-case;

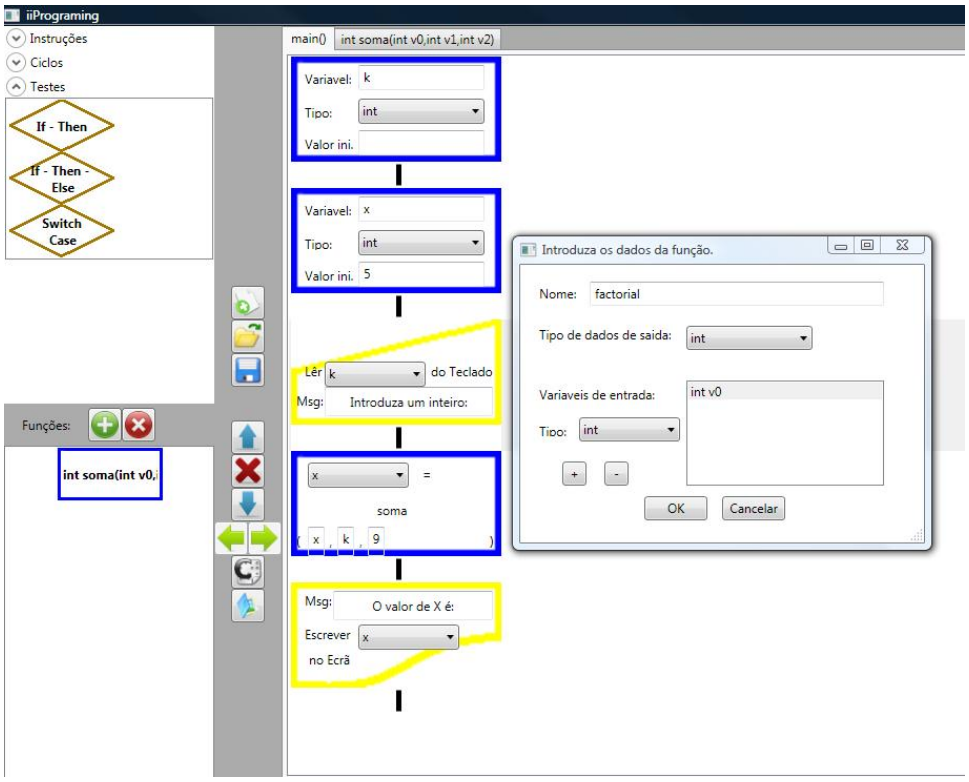


Figura 3.14: Declaração da função *factorial*.

A aplicação também permite declarar e editar funções auxiliares (ver Figura 3.14 e 3.15), guardar/abrir os fluxogramas em ficheiros com a extensão *.flux* e guardar o fluxograma como imagem (ver Figura 3.16).

A aplicação iiProgramming é constituída pelos seguintes ficheiros:

- **App.xaml e App.xaml.cs:** Onde são definidas algumas propriedades da aplicação;
- **Sequencia.xaml:** Onde é definida a IU da janela principal da aplicação;
- **Sequencia.xaml.cs:** Onde são definidos os métodos que interagem com os elementos definidos na IU especificada no ficheiro Sequencia.xaml;
- **Resultado.xaml:** Onde é definida a janela onde é mostrado o código fonte e pseudo-código resultante do fluxograma;
- **Resultado.xaml.cs:** Onde são definidos alguns métodos e os eventos dos elementos definidos na IU especificada no ficheiro Resultado.xaml;
- **Imagem.xaml:** Onde é definida a janela onde é mostrado o fluxograma simplificado e a janela de dialogo para guardá-lo como imagem;

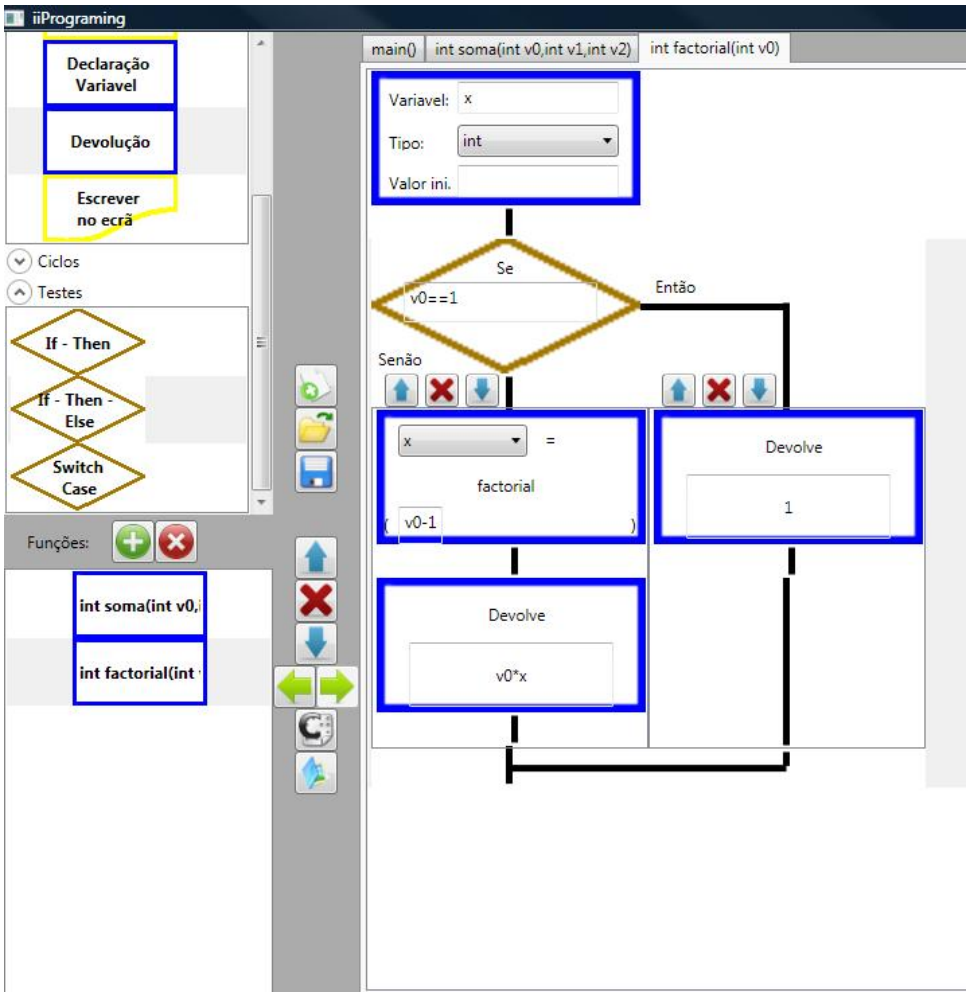


Figura 3.15: Algoritmo da função recursiva *factorial*.

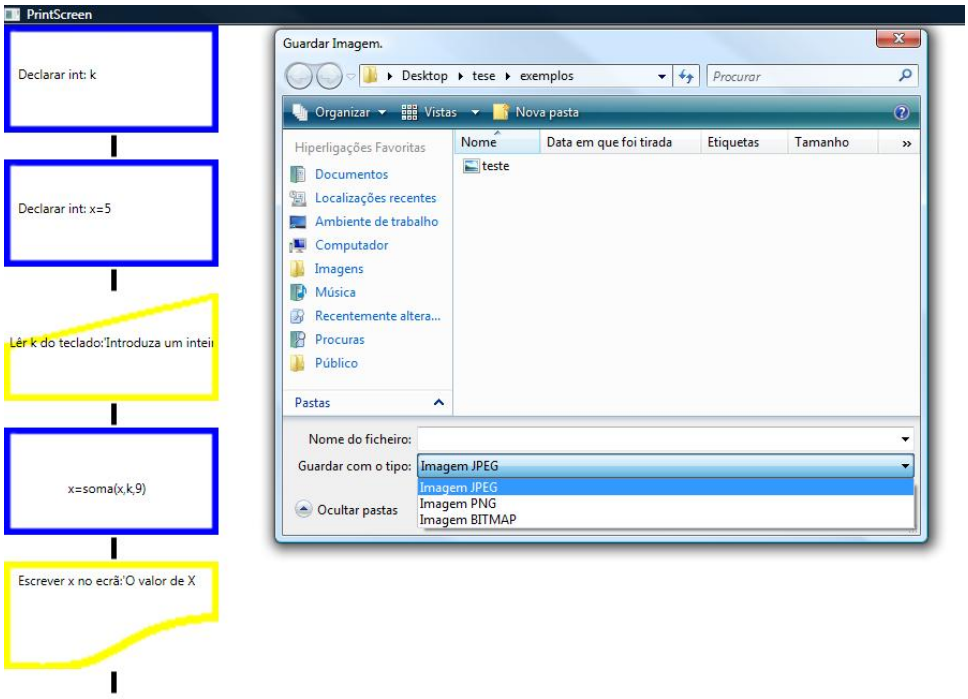


Figura 3.16: Guardar fluxograma como imagem.

- **Imagem.xaml.cs:** Onde são definidos alguns métodos e os eventos dos elementos definidos na IU especificada no ficheiro Imagem.xaml;
- **DialogFuncao.xaml:** Onde é definida a janela onde é declarada uma função;
- **DialogFuncao.xaml.cs:** Onde são definidos alguns métodos e os eventos dos elementos definidos na IU especificada no ficheiro DialogFuncao.xaml;
- **Pasta classes:** Onde estão diversos ficheiros *.cs* onde são definidos os objectos que irão armazenar os dados do algoritmo em ficheiro;
- **Pasta itemImagens:** Onde estão diversos ficheiros *.xaml* e respectivos ficheiros *.cs* onde são definidas (em classes do tipo UserControl) as instruções disponibilizadas ao utilizador para criar o algoritmo;
- **Pasta intrucoes:** Onde estão diversos ficheiros *.xaml* onde são definidas (em classes do tipo UserControl) as instruções inseridas no algoritmo pelo utilizador;
- **Pasta intrucoesSimples:** Onde estão diversos ficheiros *.xaml* onde são (em classes do tipo UserControl) definidas as versões simplificadas das instruções inseridas no algoritmo pelo utilizador;

Figura 3.17: UserControl UCInstrucao.xaml.

3.4.1 Utilização da classe UserControl

A classe UserControl permite criar controlos que podem ser usados em vários locais dentro de uma aplicação [33]. É possível incluir todo o código necessário para validar os dados comuns que são pedidos ao utilizador como, por exemplo, eventos. Um uso eficiente do UserControl é simplesmente carregar uma ListBox com itens estáticos usados com frequência em várias aplicações. Na aplicação iiProgramming foram definidos três tipos de UserControl:

- **ItemsImagem:** São os UserControl correspondentes às instruções pré-definidas disponibilizadas ao utilizador, contidos na pasta itemImagens. São constituídos por uma imagem da forma geométrica correspondente ao tipo de instrução e por uma Label que contém a descrição da instrução. Por exemplo, a definição do UserControl correspondente a uma instrução genérica (Figura 3.17), é feita no ficheiro UCInstrucao.xaml do seguinte modo:

```
<UserControl ... >
        <Canvas x:Name="LayoutRoot" Width="100" Height="50">
        <Image x:Name="imagem" Width="100" Height="50" Source=
            "/PseudoFluxCode;Component/imagens/instrucao.png"/>
        <Label x:Name="texto" Content="Instrução&#xd;&#xa;
            Genérica" Width="100" Height="50" .../>
        </Canvas>
</UserControl>
```

- **Instruções:** São os UserControl correspondentes às instruções que fazem parte do fluxograma criado pelo utilizador, contidos na pasta *instrucoes*. Quando o utilizador arrasta um UserControl do tipo ItemsImagem para a ListBox *sequencia*, é inicializado o UserControl correspondente.

- **InstruçõesSimples:** São os UserControl simplificados dos UserControl do tipo Instruções, usados quando o utilizador pretende guardar o fluxograma como imagem. São constituídos por uma imagem da forma geométrica correspondente ao tipo de instrução e por uma Label que contém a instrução definida pelo utilizador.

3.4.2 Implementação do drag-and-drop

O drag-and-drop é implementado no WPF em seis passos:

1. detectar o *drag* como a combinação dos eventos MouseMove e MouseLeftButtonDown;
2. encontrar os dados a serem arrastados e criar um DataObject que contém o formato, os dados e os efeitos permitidos;
3. inicializar o drag chamando o método DoDragDrop();
4. Definir para *true* a propriedade AllowDrop no elemento que vai receber os dados arrastados;
5. Verificar o formato e os dados chamando o método GetDataPresent no argumentos do evento Drop;
6. Obter os dados através do método GetData().

Drag

O primeiro passo da implementação do drag-and-drop é detectar o movimento do rato enquanto o botão esquerdo é pressionado. Por isso, foram utilizados os eventos *MouseMove* e *MouseLeftButtonDown* na origem dos objectos que se querem arrastar, ou seja, nas ListBox que contêm os UserControl do tipo ItemsImagem. Caso o botão esquerdo do rato seja solto, então é cancelado o drag-and-drop. Em seguida é verificada a origem dos dados e criado o DataObject. Por fim, dá-se inicio ao Drag invocando o método *DoDragDrop*. É também usada a variável *dragStarted* do tipo *bool* para auxiliar a operação. Eventos referentes ao Drag:

```
public bool dragStarted;
private void ListBox_PreviewMouseDown(object sender,
    MouseButtonEventArgs e)
```

```
{
        dragStarted = true;
        base.OnPreviewMouseDown(e);
}
private void ListBox_MouseLeftButtonUp(object sender,
   MouseButtonEventArgs e)
{
        dragStarted = false;
}
private void ListBox_PreviewMouseMove(object sender,
   MouseEventArgs e)
{
        //verifica se o botão esquerdo está a ser pressionado e se
            quem invoca o evento é do tipo ListBox
        if (dragStarted && sender is ListBox)
        {
                //cria o DataObject
                ListBox theList = sender as ListBox;
                DataObject data = CreateDataObjectFromList(theList
                   );

                //captura os eventos do rato para a ListBox e
                   realiza o drag/drop
                Mouse.Capture(sender as UIElement);
                System.Windows.DragDrop.DoDragDrop(theList, data,
                   DragDropEffects.Copy);
                Mouse.Capture(null);

                dragStarted = false;
                base.OnPreviewMouseMove(e);
        }
}
```

Onde *CreateDataObjectFromList* é o método que detecta qual o tipo de UserControl ItemsImagem seleccionado pelo utilizador e devolve o DataObject com o UserControl do tipo Instruções correspondente. Método CreateDataObjectFromList:

```
private DataObject CreateDataObjectFromList(ListBox list)
```

```
{
        DataObject data = new DataObject();

        UserControl uc = list.SelectedItem as UserControl;
        //verifica se o objecto a largar é do tipo UCInstrucao
        UCInstrucao i = uc as UCInstrucao;
        if(i!=null)
        {
                InstrucaoUC ins = new InstrucaoUC();
                data.SetData(DataFormats.FileDrop, ins);
                return data;
        }
        //verifica para os restantes tipos
...
}
```

Drop

Para ser possível largar (drop) o objecto arrastado numa ListBox, foi necessário definir a propriedade AllowDrop a *true* e associar os eventos *Drop* e *MouseMove*. Definição de uma ListBox no código XAML que vai conter uma sequência de instruções:

```
<ListBox x:Name="sequencia"  AllowDrop="True" Drop="panel_Drop"
    MouseMove="panel_MouseMove"/>
```

Definição de uma ListBox no código C# que vai conter uma sequência de instruções:

```
ListBox lb = new ListBox();
lb.AllowDrop = true;
lb.Drop += new DragEventHandler(panel_Drop);
lb.MouseMove += new MouseEventHandler(panel_MouseMove);
```

Quando o utilizador larga o objecto arrastado sobre a ListBox de destino, é invocado o evento *Drop*, onde será introduzido os dados provenientes do DataObject na ListBox. Evento Drop:

```
public void panel_Drop(object sender, DragEventArgs e)
{
        //Verificar o formato e os dados chamando o método
            GetDataPresent
        if (e.Data.GetDataPresent(DataFormats.FileDrop))
```

```
        {
                //Obter os dados
                UserControl controlo = e.Data.GetData(DataFormats.
                    FileDrop, true) as UserControl;
                if (controlo != null)
                {
                        ListBox s = sender as ListBox;
                        ...
                        //insere na posição pretendida
                        s.Items.Insert(index, controlo);
                        //coloca o objecto UserControl inserido à
                            vista do utilizador
                        s.ScrollIntoView(s.Items.GetItemAt(index))
                            ;
                        ...
                }
        }
        e.Handled = true;
}
```

Para que o objecto arrastado possa ser inserido na posição da sequência onde é largado, é invocado o evento *MouseMove* na ListBox de destino do objecto, sendo a posição guardada na variável *index*. Evento MouseMove:

```
public int index=0;
private void panel_MouseMove(object sender, MouseEventArgs e)
{
        ListBox list1 = sender as ListBox;
        if (list1.Items.Count > 0)
        {
                object item = null;
                //obtém a posição do objecto onde o ponteiro do
                    rato está sobre
                item = GetElementFromPoint(list1, e.GetPosition(
                    list1));
                index = list1.Items.IndexOf(item);
        }
}
```

Capítulo 4

Conclusões e trabalho futuro

Durante algumas demonstrações feitas a várias turmas de escolas secundárias, notou-se que, mesmo aquelas que já têm acesso a quadros interactivos, ambos os alunos e professores mostraram-se entusiasmados com o sistema implementado, bem como as aplicações desenvolvidas. Também concluímos que, pelo facto da aplicação iiNote guardar as notas em memória de um modo simples e rápido, o tempo que o utilizador (professor/apresentador) perde a guardar essas notas seja bastante pequeno, maximizando, por isso, o tempo de duração da apresentação. Além disso, a possibilidade de editar as notas posteriormente leva a que o utilizador não esteja preocupado se as notas estão perfeitas, permitindo ao utilizador corrigir as notas posteriormente. No entanto, no futuro pretende-se que a aplicação eeNote permita a introdução de texto através do teclado para ser adicionado às notas, de modo a poder complementá-las ainda mais.

Em relação à aplicação iiProgramming, ainda apresenta algumas limitações, pois está bastante dependente dos dados introduzidos pelo utilizador de modo a produzir código sem erros. Entretanto, no futuro pretende-se implementar um analisador de instruções de modo a assistir o utilizador a gerar código correcto.

O uso de um segundo Wiimote, em conjunto com o Wiimote Whiteboard para permitir controlar o computador à distância, cancela quase completamente a dependência que o utilizador tem do rato e do teclado. Isto permite ao utilizador estar perto do quadro onde, além de o poder controlar sem limitações, pode também controlar à distância o que quer mostrar no quadro (e.g., PowerPoint, imagens, vídeos, etc.). No entanto, devido ao número limitado de botões do Wiimote, não foi possível implementar algumas

funcionalidades tais como o scroll do rato ou as teclas Backspace, Delete, Page Up e Page Down. Por isso, no futuro, existe a possibilidade em implementar um dispositivo próprio para controlar o computador à distância e, então, eliminar as limitações atrás referidas.

Resumindo, desenvolveu-se um sistema de projecções interactivas de baixo custo e mostrou-se que é fácil desenvolver aplicações educativas para o sistema. Deste modo é possível ter um quadro interactivo em qualquer sala de aulas.

De referir ainda que este trabalho já deu origem a um artigo [34] aceite na Computers and Advanced Technology in Education (CATE) 2010 da International Association of Science and Technology for Development (IASTED).

Referências

[1] Lucy Gray. Interactive Whiteboard Explorations. Website, Consultado em Abril 2010. http://isenet.ning.com/groups/group/show?groupUrl=interactivewhiteboardexplorations&xg_source=activity.

[2] SMART Technologies Inc. SMART Board™ rear projection systems. Website, Consultado em Abril 2010. http://www.smartboard.ie/product-rear-projection-interactive-whiteboard.php.

[3] SMART Technologies Inc. SMART Board™ for flat panel displays. Website, Consultado em Abril 2010. http://www.smartboard.ie/product-smart-board-for-flat-panel-displays.php.

[4] Smoothboard Wiki. Mount and position the Wiimote. Website, Consultado em Março 2010. http://www.boonjin.com/smoothboard/index.php?title=Mount_and_position_the_Wiimote.

[5] Marco Silva, Luís Paulo Reis, Armando Sousa, Brígida Mónica Faria, and A. Pedro Costa. iiBOARD, Development of a Low-Cost Interactive Whiteboard using the Wiimote Controller. *International Conference on Computer Graphics Theory and Applications*, pages 337–344, 2009.

[6] Conselho da União Europeia. Jornal oficial das comunidades europeias C 142/1. Website, 2002. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:C:2002:142:0001:0022:PT:PDF.

[7] Conselho de Educação da União Europeia. Relatório do Conselho (Educação) para o Conselho Europeu "Os objectivos futuros concretos dos sistemas de educação e formação". Website, 2001. http://ec.europa.eu/education/policies/2010/doc/rep_fut_obj_pt.pdf.

[8] Conselho de Ministros. Resolução do Conselho de Ministros n.º 137/2007 de 18 de Setembro. Website, 2007. http://www.escola.gov.pt/idc/idcplg?IdcService=GET_FILE&dID=11496&dDocName=001952.

[9] Conselho de Ministros. Resolução do Conselho de Ministros n.º 132/2007 de 13 de Setembro. Website, 2007. http://www.escola.gov.pt/idc/idcplg?IdcService=GET_FILE&dID=11499&dDocName=001953.

[10] Anja Balanskat, Roger Blamire, and Stella Kefala. *The ICT Impact Report, A review of studies of ICT impact on schools in Europe*. European Schoolnet, 2007. http://ec.europa.eu/education/doc/reports/doc/ictimpact.pdf.

[11] SMART Technologies Inc. Interactive Whiteboards and Learning. Improving student learning outcomes and streamlining lesson planning. Março 2006. http://www2.smarttech.com/NR/rdonlyres/2C729F6E-0A8D-42B8-9B32-F90BE0A746D8/0/Int_Whiteboard_Research_Whitepaper_Update.pdf.

[12] Wikipedia. Interactive Whiteboard. Website, Consultado em Março de 2010. http://en.wikipedia.org/wiki/Interactive_whiteboard.

[13] Wikipedia. SMART Board Interactive Whiteboard. Website, Consultado em Março de 2010. http://en.wikipedia.org/wiki/SMART_Board_interactive_whiteboard.

[14] Luidia Inc. Interactive Whiteboard - Enhance Classroom Communications. Website, Consultado em Abril 2010. http://www.luidia.com/products/ebeam-edge-for-education-page.html.

[15] mimio Interactive Teaching Technologies. mimio Interactive Xi Bar and Stylus Overview. Website, Consultado em Abril 2010. http://www.mimio.com/products/mimio_interactive/index.asp.

[16] Dilpesh V. Laxmidas. *Exame Informática*, chapter Aulas 2.0, page 57. Number 117. Março 2010.

[17] Johnny C. Lee. Wii Projects. Website, Consultado em Setembro 2009. http://johnnylee.net/projects/wii/.

[18] Filipe Lino, Paulo Dias, Arnaldo Oliveira, and Beatriz S. Santos. Comparação de Dispositivos de Interacção em Ambientes de Realidade Virtual: Desenvolvimento de um Setup Experimental e Estudos com Utilizadores. *Actas do 17º Encontro Português de Computação Gráfica*, pages 175–183, 2009.

[19] Brian Peek. WiimoteLib - .NET Managed Library for the Nintendo Wii Remote. Website, Consultado em Janeiro 2010. http://www.brianpeek.com/blog/pages/wiimotelib.aspx.

[20] MSDN. keybd_event function (Windows). Website, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/library/ms646304%28VS.85%29.aspx.

[21] MSDN. Virtual-Key Codes (Windows). Website, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/library/dd375731%28v=VS.85%29.aspx.

[22] MSDN. mouse_event function (Windows). Website, Consultado em Janeiro 2010. http://msdn.microsoft.com/en-us/library/ms646260%28VS.85%29.aspx.

[23] PINVOKE.NET. mouse_event (user32). Website, Consultado em Janeiro 2010. http://www.pinvoke.net/default.aspx/user32.mouse_event.

[24] FYP-Wii. Wiimote and Wiimotelib. Website, Consultado em Janeiro 2010. http://fypwii.blogspot.com/2008/11/wiimote-and-wiimotelib.html.

[25] Smoothboard.net. Smoothboard - The Wiimote Whiteboard. Website, Consultado em Setembro 2009. http://www.smoothboard.net/.

[26] Microsoft.com. Explore the features: Xps documents. Website, Consultado em Março 2010. http://www.microsoft.com/windows/windows-vista/features/xps.aspx.

[27] Wikipedia. Windows Presentation Foundation. Website, Consultado em Setembro 2009. http://en.wikipedia.org/wiki/Windows_Presentation_Foundation.

[28] MSDN. InkCanvas Class. Website, Consultado em Setembro 2009. http://msdn.microsoft.com/en-us/library/system.windows.controls.inkcanvas.aspx.

[29] Markus Egger. *MSDN Magazine*, chapter Find New Meaning In Your Ink With Tablet PC APIs In Windows Vista. Maio 2006. http://msdn.microsoft.com/en-us/magazine/cc300793.aspx.

[30] Alex Woodie. Support for XPS, Microsoft's PDF-Killer, Gaining Steam. Website, Consultado em Dezembro 2009. http://www.itjungle.com/two/two011806-story01.html.

[31] Wikipedia. Open Packaging Conventions. Website, Consultado em Dezembro 2009. http://en.wikipedia.org/wiki/Open_Packaging_Conventions.

[32] Abel Gomes. Cap.4:Design de Algoritmos e Programação Estruturada. Website, Consultado em Setembro 2009. http://www.di.ubi.pt/~agomes/programacao/teoricas/04-algoritmos.pdf.

[33] MSDN. UserControl Class. Website, Consultado em Setembro 2009. http://msdn.microsoft.com/en-us/library/system.windows.forms.usercontrol.aspx.

[34] Vasco Santos and Frutuoso Silva. Interactive Projections in Classrom. In *30th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2010)*, to appear.