Diogo Luiz da Silva

Central multimídia de entretenimento controlada por gestos

São José – SC

Julho / 2012

Diogo Luiz da Silva

Central multimídia de entretenimento controlada por gestos

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Emerson Ribeiro de Mello, Dr.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Julho / 2012

Resumo

Centrais multimídia de entretenimento consistem de computadores dedicados interconectados a aparelhos de televisão e som, como home theater. Tais computadores, possuem software específico para a reprodução de arquivos de áudio, vídeo, imagens, etc. O XBMC, criado em 2003, consiste de um aplicativo para centrais multimídias, podendo ser executado sobre diferentes sistemas operacionais. A interação do usuário com tais sistemas geralmente se dá através de dispositivos de entrada comuns em computadores pessoais, como teclados e mouses, além de controle remoto por IR. Em 2010 a Microsoft apresentou o Kinect, um dispositivo voltado inicialmente para o videogame Xbox, que permite aos jogadores interagirem com os jogos simplesmente com o movimento de seus corpos, não necessitando mais dos tradicionais joysticks. O presente trabalho tem por objetivo apresentar um aplicativo que permita controlar uma central multimídia XBMC através de gestos capturados por um dispositivo Kinect.

Abstract

Media Centers consist of dedicated computers connected to television and sound system, such as home theater. These computers have specific software for playing audio, video and images files. XBMC, created in 2003, is an application for media centers, which can run on different operational systems. User interaction with such systems usually occurs through input devices common in personal computers, such as keyboards, mouses and IR remote control. In 2010 Microsoft introduced Kinect, a device focused initially for the Xbox video game that allows players to interact with games simply by moving their bodies, not requiring more of the traditional joysticks. This paper aims to present an application to control a media center XBMC through gestures captured by a device Kinect.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Comunicação por gestos está presente em nosso dia a dia, porém a interação com computadores ou aparelhos eletrônicos se dá através de os teclados, mouses e controles remotos. Esses dispositivos atendem perfeitamente na maioria dos casos, mas em algumas situações é desejado que o usuário tenha uma maior liberdade de se comunicar com a máquina, como por exemplo, aumentar o volume da televisão através do gesto de levantar a mão.

Centrais multimídia tem por objetivo apresentar uma interface amigável para organizar e reproduzir conteúdos multimídias. Atualmente centrais multimídia, fazem uso de teclados, mouses ou controles remotos. O que torna pobre a experiência de uso, em alguns casos, até inviabilizando a utilização. Por exemplo, em uma sala, onde um teclado com fio não chega até o sofá, o usuário terá que se locomover para controlar a central. Em outro caso, um teclado sem fio acaba sendo um pouco grande para deixar junto ao sofá e incômodo para o manuseio da aplicação. É neste cenário que a interação através de gestos poderia melhorar a experiência de uso.

1.1 Motivação

A motivação desse trabalho veio com a chegada do Kinect[1], um dispositivo para o vídeo game Xbox 360, que permite ao usuário interagir com o vídeo game através de gestos ou da fala. Inicialmente o aparelho foi usado somente para jogos eletrônicos, mas em pouco tempo demonstrou ser um dispositivo com inúmeras possibilidades, fazendo com que desenvolvedores começassem a criar novas aplicações para o dispositivo. Diante desse grande interesse a Microsoft liberou um kit de desenvolvimento para o Kinect para o Windows 7, incentivando desenvolvedores a criar aplicações para a plataforma. No Linux, algumas bibliotecas acabaram surgindo antes mesmo da Microsoft liberar seu kit de desenvolvimento, sendo a principal delas a OpenNI[2], criada pela fabricante do *chipset* do Kinect.

[1] http://www.xbox.com/pt-BR/Kinect
[2] http://www.openni.org/

Por reconhecer gestos, o Kinect acabou sendo uma solução para o problema de usabilidade das centrais multimídia pois é possível utilizar o Kinect para captar os gestos do usuário e transformar em ações da central, como pausar um vídeo ou música, aumentar ou diminuir o volume, avançar ou retroceder e outras ações possíveis da central.

1.2 Objetivos

Este trabalho tem como objetivo criar uma aplicação que consiga interpretar gestos do usuário e transformá-los em ações conhecidas pela central multimídia, como aumentar ou diminuir o volume de uma música ou vídeo. São objetivos específicos:

- Estudar um meio de interligar os gestos do usuário e as ações da central multimídia, como pausar ou aumentar o volume de uma música;
- Estudo dos melhores gestos para interagir com a central multimídia. Ou seja, para cada ação deve haver um gesto específico;
- Implementação da aplicação para reconhecer gestos e interligá-los com a central multimídia.

1.3 Organização do texto

O texto está organizado da seguinte forma: No Capítulo 2 é apresentado a revisão bibliográfica, no Capítulo 3 é apresentado o resultado do trabalho e no Capítulo 4 as considerações finais.

2 *Revisão Bibliográfica*

2.1 Centrais Multimídia

Central multimídia consiste de uma aplicação que tem como fim o entretenimento. Geralmente possui uma interface simples e permite ao usuário organizar conteúdo como fotos, músicas e vídeos apresentado assim um meio fácil para visualização e reprodução do conteúdo.

A maneira mais comum de utilizar uma central multimídia é em computadores pessoais, onde o usuário simplesmente precisa de um software que reproduza o conteúdo desejado, porém tem-se um declínio na experiência de uso quando o usuário necessita conectar o computador a uma televisão e ter que interagir com o sistema através de um teclado ou mouse. Outra opção é o usuário utilizar um hardware dedicado, feitos para serem utilizados junto a televisões, são opções mais caras, mas podem possuir controles remotos próprios e são esteticamente melhores para uso em ambientes como salas ou quartos. Alguns exemplos de centrais multimídias serão mostrados nas próximas seções.

2.1.1 Apple TV

Apresentada em 2007, o *Apple TV*[1] é um hardware dedicado que permite a transferência de conteúdo multimídia através da internet e o armazenamento do mesmo, para uma reprodução posterior. Trata-se um dispositivo pequeno, que pode interagir com vários outros dispositivos e aplicações da Apple. O Apple TV está na segunda versão de hardware e possui saídas HDMI, USB, Áudio Digital e Ethernet, além de possuir também a possibilidade de conexão por redes sem fio(APPLE, 2011).

O Apple TV torna-se proveitoso para usuários que possuam dispositivos da Apple, pois permite reproduzir conteúdos a partir de qualquer um de seus dispositivos Apple ou dos serviços de nuvem[2], diretamente em seus televisores sem a necessidade de fios. As facilidades das

[1] `http://www.apple.com/appletv/`
[2] `http://pt.wikipedia.org/wiki/Computacao_em_nuvem`

soluções Apple e a qualidade dos produtos são grandes vantagens para o Apple TV, mas também são mais caros.

2.1.2 Windows Media Center

Windows Media center é uma central multimídia desenvolvida pela Microsoft e está presente nos seus sistemas operacionais mais recentes. Uma de suas vantagens é a interação com dispositivos como o Xbox 360 que permite controlar a central multimídia remotamente e ver seus conteúdos pelo console através de uma rede Ethernet ou Wi-Fi, sem que o usuário necessite ter um computador por perto.

Sua interface é simples e sua configuração é bem intuitiva. Por vir instalado nos sistemas da Microsoft ele acaba sendo uma opção interessante para os usuários sem muita experiência que não queiram instalar outros softwares.

Para usuários que desejam realizar funções simples o Windows Media Center pode atender suas necessidades, mas para uma experiência completa, onde o usuário deseja controlar todo seu conteúdo e ações através da central, o Windows Media Center não é a melhor opção, pois uma central multimídia completa deve realizar várias ações em sua própria interface e deixando a disposição do usuário várias opções de configuração, organização e exibição, e são nesses quesitos que o Windows Media Center acaba deixando a desejar.

2.1.3 Moovida

O Moovida Media Player[3] realiza funções básicas de uma central multimídia e não oferece muitas opções para Linux, onde a última versão disponível é a 1.0.9. Para Windows, o Moovida se encontra em uma versão bem mais atual, tem visual atraente, interface 3D para televisores e uma compatibilidade razoável com os formatos de mídias(KARASINSKI, 2010).

O Moovida possui uma interface para utilização no computador e outra interface para melhor visualização em televisores, que preza pela facilidade de navegação e gera imagens tridimensionais com diversos efeitos gráficos, mas que acaba necessitando de mais processamento.

Apesar da qualidade de sua interface, o Moovida acaba penando por uma boa organização e gerenciamento do conteúdo do usuário, que fica um pouco prejudicada com a falta de facilidade na execução de algumas tarefas simples, como adicionar conteúdo a biblioteca e posteriormente encontrá-las na interface.

[3] `http://www.moovida.com/`

2.1.4 Boxee

Boxee[4] surgiu como uma ideia em 2004 de um grupo de amigos que decidiram melhorar a plataforma XBMC, discutida na Seção 2.1.6, adicionando novas funcionalidades e integração com redes sociais e fontes de mídia online como Hulu[5], Netflix[6] e muitos outros (BOXEE, 2011).

O Boxee oferece diversas maneiras para o usuário assistir seus conteúdos, como em seu computador, televisão ou dispositivos portáteis e também oferece a possibilidade de utilizar um hardware dedicado desenvolvido pela D-Link, exclusivo para execução do Boxee, chamado de Boxee Box. O mesmo possui diversas conexões: Saída HDMI, Porta Ethernet, saída de áudio ótico, saída de áudio componente, 2 portas USB e conexão Wireless 802.11n.

Nas versões para computador o Boxee tem suporte a Windows, MAC OS X e Linux e possui uma ótima compatibilidade com diversos formatos de mídia assim como no Boxee Box. Sua interface não possui muitos detalhes e pode ser um pouco confusa inicialmente. O Boxee necessita também que o usuário faça um pequeno cadastro para gerenciar seus serviços, redes sociais e dados. Assim, o usuário não precisa reconfigurar todo o ambiente caso mude de plataforma ou utilize várias delas.

Além de gerenciar o conteúdo local o Boxee também tem uma enorme biblioteca de conteúdo online, lembrando que dependendo do serviço de conteúdo, ele pode ser pago. Outra qualidade do Boxee é a organização do conteúdo local, basta que o usuário especifique os locais onde as mídias se encontram e o Boxee se encarrega de buscar capas e informações sobre a mídia na internet, caso elas não possuam.

2.1.5 LinuxMCE

O LinuxMCE[7] é baseado em Linux e consiste de uma central para controlar toda a casa, desde computadores e televisores até a luz e portas, sendo assim, o LinuxMCE não é exatamente uma central multimídia convencional.

Por ser uma solução de automação completa o LinuxMCE necessita de hardwares específicos para controlar os dispositivos da casa, como aparelhos eletrônicos, iluminação do ambiente, sistema de segurança e o sistema de telefonia. Quanto a parte de central multimídia,

[4] `http://www.boxee.tv`
[5] `http://www.hulu.com/`
[6] `www.netflix.com`
[7] `http://linuxmce.com/`

o LinuxMCE pode armazenar o conteúdo multimídia e o usuário pode acessá-lo de qualquer lugar da casa. Ainda tem a possibilidade de cuidar da parte de recepção de TV via cabo, satélite ou sinal aberto(RIGUES, 2007).

Apesar de ser uma solução completa para uma casa, precisa de vários hardwares adicionais e de um computador exclusivo para ele. Sua interface de configuração ainda é um pouco técnica e necessita de um tempo considerável para sua configuração completa, mesmo para usuários mais experientes. Apesar de permitir o uso de controles infra-vermelhos ou celulares, sua interface não é muito amigável.

2.1.6 XBMC Media Center

XBMC Media Center[8] é uma central multimídia de código aberto e multi-plataforma completa. Lançado em 2003 com o propósito de ser uma central multimídia para o vídeo game Xbox, porém desde 2010 está oficialmente disponível para as plataformas IA-32/x86, x86-64, ARM, ou CPUs baseados em PowerPC e sistemas operacionais Linux, Mac OS X e Microsoft Windows.(WIKIPEDIA, 2011).

O XBMC consegue reproduzir a maioria dos formatos de arquivos de áudio e vídeo do mercado, conteúdos de CDs e DVDs, do disco rígido e arquivos compartilhados na rede, vídeos por streaming de YouTube, Hulu e Netflix, tocar estações de rádio online, previsão do tempo, montar lista de músicas e vídeos, slideshow de imagens, karaokê e ainda existe a opção de habilitar outras funções com a instalação de plugins de terceiros, onde a maioria pode ser obtido dentro da própria interface do XBMC (WIKIPEDIA, 2011).

Os requisitos de hardware para o XBMC podem variar, pois depende do tipo de conteúdo que o usuário pretende executar. A configuração de hardware mínima para o XBMC é: 1GB de memória RAM e Processador Dual Core acima de 2GHz. Em alguns casos pode-se precisar de máquinas com uma boa placa gráfica e um processador melhor para conseguir visualizar o conteúdo com perfeição, como é o caso de vídeos em alta definição (WIKIPEDIA, 2011).

A utilização de hardwares mais específicos podem ser opções melhores em ambientes que precisem de espaço ou que tenham a necessidade de um hardware esteticamente mais agradável, como é o caso do *Mini PC Inspiron Zino HD*[9] que é um computador super compacto da DELL para fins de entretenimento. Além de compacto, é esteticamente bonito, o que permite colocá-lo em qualquer lugar da casa, diferentemente dos computadores convencionais. Ele possui

[8] `http://xbmc.org/`
[9] `http://www.dell.com/br/p/inspiron-zino-hd-410/pd?ref=gzilla`

inicialmente o sistema operacional Windows® 7 Home Premium[10](DELL, 2011).

Figura 2.1: Mini PC Inspiron Zino HD (REALGAGE, 2009)

Existem outras opções de hardwares como *D2 Plug*[11],*DreamPlug*[12], ou o *CuBox*[13] que são hardwares dedicados baseados em ARM que possuem a vantagem de serem pequenos, têm baixo consumo e com um bom poder de processamento para executar as tarefas exigidas pelo XBMC específico para essas plataformas. A figura 2.2a mostra o *D2 Plug* e a figura 2.2b mostra o *CuBox*.

(a) D2 Plug (PINOYTUTORIAL, 2011)

(b) CuBox(SOLIDRUN, 2011)

Figura 2.2: Hardwares dedicados para executar uma central multimídia

2.2 Interações Naturais

Segundo a OpenNI (OPENNI, 2011), o termo Interação Natural(*Natural Interaction*) refere-se a um conceito onde a interação homem-máquina é feita com base nos sentidos humanos, tentando deixar controles remotos, teclados e mouses obsoletos. Alguns exemplos de interações naturais são:

[10] `http://windows.microsoft.com/pt-BR/windows7/products/home`
[11] `http://www.globalscaletechnologies.com/p-43-d2-plug.aspx`
[12] `http://www.globalscaletechnologies.com/c-5-dreamplugs.aspx`
[13] `http://www.solid-run.com/products/cubox`

- Reconhecimento de voz, onde as instruções são dadas pela fala do usuário. Existem diversas aplicações onde o usuário pode, por comodidade ou por alguma deficiência física, utilizar a voz para realizar ações. Neste caso bastaria um microfone e um software para traduzir a fala do usuário em uma linguagem que a máquina entenda. Um exemplo é o Siri[14], onde o usuário consegue realizar diversas ações somente com a voz.

- Gestos com a mão, onde o usuário deve realizar gestos pré-definidos para realizar ações. Permite ao usuário realizar ações sem a utilização de qualquer outro dispositivo, como por exemplo, folhear um livro digital simplesmente movimentando a mão na direção que se deseja folhear, ou controlar a intensidade da luz do ambiente levantando ou abaixando as mãos. A Figura 2.3 mostra um exemplo de controle por gestos com a mão em uma central multimídia.

Figura 2.3: Usuário controlando Windows Media Center por gestos (PINOTTI, 2011)

- Rastreamento do movimento corporal, onde o movimento de corpo inteiro é monitorado, analisado e interpretado. Para diversas aplicações seria interessante conseguir rastrear o movimento do corpo humano, como por exemplo, ajudar na recuperação de pessoas que sofreram acidentes, estimulando ou registrando os movimentos para melhor análise dos médicos, ou simplesmente para entretenimento, como em jogos de dança ou esportes. A Figura 2.4 mostra a utilização do rastreamento do corpo de uma criança para ajudá-la a se recuperar de um acidente, estimulando-a com um jogo, onde ela deve movimentar sua perna para atingir os alvos, e assim melhorar a movimentação da mesma.

[14] `http://www.apple.com/iphone/features/siri.html`

Figura 2.4: Utilização de rastreamento corporal para estimular a recuperação de uma criança (MICROSOFT, 2011)

2.3 Dispositivos de interações naturais

Dispositivos de interações naturais são aqueles que atendem um ou mais conceitos descritos na Seção 2.2 e permitem aos usuários interagir de forma natural e intuitiva com outros dispositivos.

Um exemplo bem simples de um dispositivo de interação natural é o microfone, com ele interagimos com a fala, que é uma interação natural entre os seres humanos. Outro dispositivos de interação natural é a câmera de vídeo, pois com ela conseguimos não só interagir com outras pessoas, mais também com a máquina. Alguns exemplos disso são o reconhecimento facial pra identificar pessoas ou realizar certas ações e reconhecimento de presença para ligar ou desligar equipamentos ou detectar intrusos.

Quando se trata de gestos com a mão ou de rastreamento corporal, é feito um processamento de imagens, onde se analisa a imagem de uma câmera e tenta-se captar o movimento desejado somente com a imagem capturada pela câmera. Mas o resultado em muitos casos não é o mais desejado ou preciso, além de exigir maior poder de processamento e ser mais difícil trabalhar dessa maneira.

Outra maneira de tratar os gestos e o movimento do corpo, e que demostrou ser muito melhor, é com emissores de infra-vermelhos e sensores de profundidade, onde vários feixes de infra-vermelho são lançados pelo ambiente e o sensor de profundidade faz a leitura desses feixes e constrói um mapa tridimensional do ambiente.

A Figura 2.5 ilustra os componentes e interações entre estes para capturar gestos do usuário e como são transformados em ações nos aplicativos. Nas Seções 2.3.1 e 2.3.2 esses dispositivos são apresentados em detalhes.

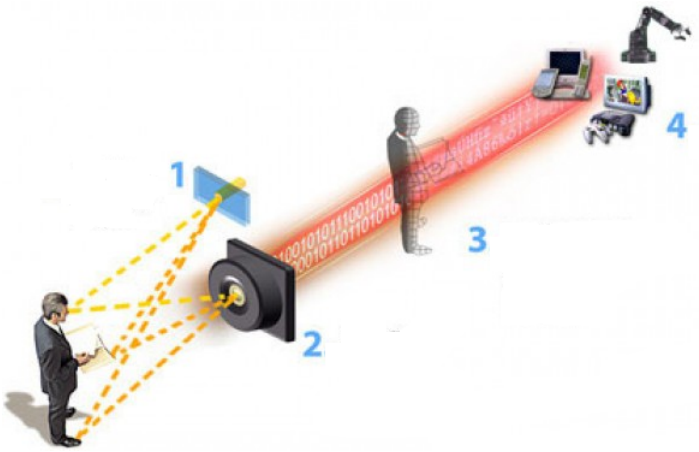


Figura 2.5: Tecnologia de rastreamento corporal com infra-vermelhos(CARMODY, 2010)

1. Uma fonte de luz invisível (emissor de infra-vermelho) iluminando o indivíduo;
2. Um sensor capturando a luz refletida, as distorções da luz gerada pelas superfícies e transformando os dados coletados em um mapa tridimensional;
3. Um *software* usa o mapa criado pelo sensor para identificar os objetos em tempo real, mapear os movimentos feitos pelo indivíduo e transformar essas ações em ações conhecidas pelas aplicações;
4. Os dispositivos finais, que serão controlados pelas ações geradas pelo software.

2.3.1 Kinect

Criado pela Microsoft para o vídeo game Xbox, o Kinect é um dispositivo que permite o usuário interagir com a máquina através de gestos, rastreamento do movimento corporal e da voz. A Figura 2.6 apresenta os principais componentes do Kinect, são estes:

1. Vetor de Microfones: quatro microfones que podem identificar vozes ou sons, filtrando o ruído de fundo;
2. Emissor de infra-vermelho: lança vários feixes infra-vermelhos no ambiente, como mostrado na figura 2.7a ao tocar em uma superfície o ponto fica distorcido, essa distorção é lida pelo sensor de profundidade;
3. Sensor de profundidade: analisa as distorções (distorções por tipo de superfície, distâncias entre os pontos e profundidade dos mesmos) dos feixes infra-vermelhos e constrói um mapa tridimensional do ambiente com todos os objetos e pessoas, conforme a figura 2.7b;

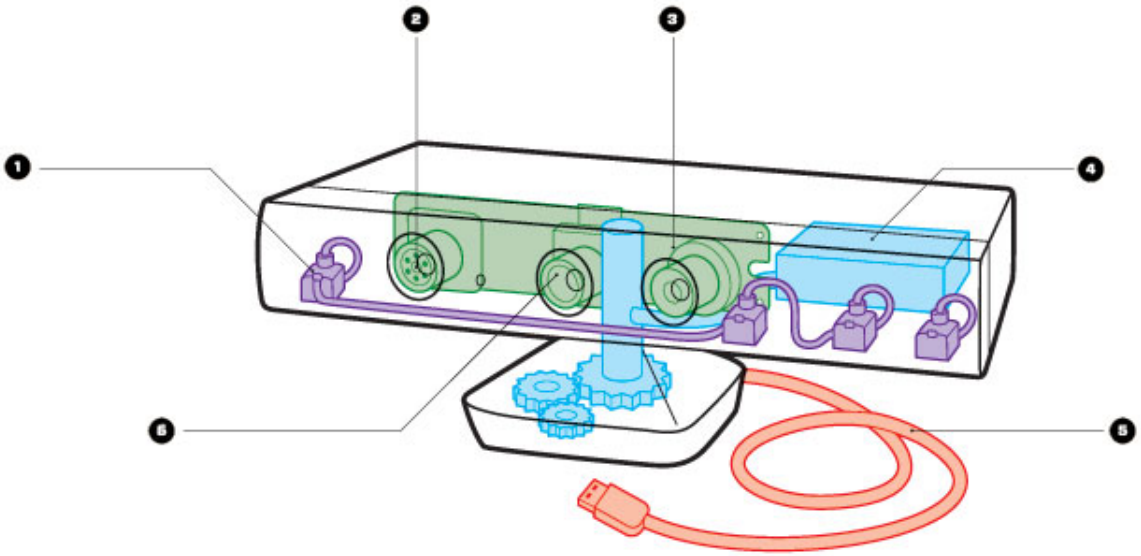
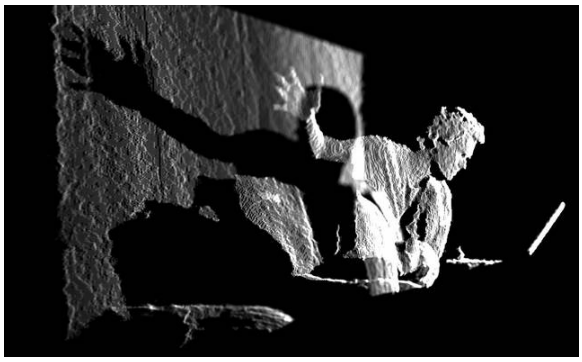


Figura 2.6: Componentes do Kinect (TANZ, 2010)

(a) Feixes infra-vermelhos iluminando um ambiente (HARDMOB, 2011)



(b) Exemplo de imagem gerada pelo sensor de profundidade (KIRN, 2010)

Figura 2.7: Feixes infra-vermelhos e imagem do sensor de profundidade

4. Motor de inclinação: serve para ajustar a inclinação das câmeras conforme a necessidade;
5. Cabo USB: transmite os dados dos microfones e das câmeras para o dispositivo conectado a ele;
6. Câmera RGB: capta uma imagem em vídeo, como uma webcam. Usado para obter detalhes sobre os objetos e pessoas, principalmente para reconhecimento facial dos usuários.

Existem outros componentes no Kinect, como um acelerômetro de três eixos, que deve ajudar na precisão com a movimentação do motor, uma memória SDRAM DDR2 da Hynix de 64MB, uma ventoinha e um LED na frente para mostrar o status do dispositivo.

2.3.2 Asus XTion Pro

Para algumas aplicações o Kinect pode se tornar sub-utilizado, pois não necessita de todas as suas funcionalidades, como por exemplo o rastreamento de movimentos corporais, ele somente necessita do emissor de infra-vermelho e do sensor de profundidade, e não precisa da câmera RGB e dos microfones. Pensando nisso, alguns outros dispositivos foram criados com a finalidade de serem mais baratos e com menos funções que o Kinect. Para aplicações como a captura de movimentos existe um dispositivo da empresa ASUS chamado XTion PRO que possui somente o emissor de infra-vermelho e o sensor de profundidade, existe também o XTion PRO Live, uma variação que possui uma câmera RGB junto, mas por enquanto esses dispositivos são para desenvolvedores(ASUS, 2011). A figura 2.8 demostra o Xtion PRO da ASUS.

Figura 2.8: ASUS XTion PRO (ASUS, 2011)

2.4 Bibliotecas para o Kinect

O SDK (Kit de Desenvolvimento de Software) da Microsoft, exclusivo para o Windows, possui diversas facilidades para o desenvolvimento de aplicações para Windows com o Kinect.

Para plataforma Linux existem poucas aplicações, e as bibliotecas existentes não possuem as facilidades do SDK da Microsoft.

2.4.1 Kinect para Windows SDK

The Kinect for Windows® Software Development Kit (SDK) Beta[15] é um kit de desenvolvimento para desenvolvedores iniciantes, para criar aplicações não comerciais para computadores rodando o sistema operacional Windows 7 da Microsoft. Este SDK possui:

- Drivers, para utilização do Kinect no Windows 7;
- APIs (*Application Programming Interface*), juntamente com a documentação técnica para desenvolvedores;
- Exemplos de códigos fonte.

Para trabalhar com esse SDK o desenvolvedor pode trabalhar com o Visual Studio[16](pacote de programas da Microsoft para desenvolvimento de software) e com as linguagens de programação C# e C++.

2.4.2 Freenect

O *Freenect(libfreenect)*[17] é uma biblioteca mantida pelo grupo OpenKinect[18] que funciona como um driver para ter acesso as funcionalidades do Kinect, pegando seus dados de forma crua. É possível utilizar essa biblioteca juntamente com outras, como por exemplo a OpenNI(Ver Seção 2.4.3). A Freenect está disponível para Windows,Linux, MAC OS X e para várias linguagens de programação. Tem suporte a câmera RGB, motor da base do Kinect, LED encontrado na frente do Kinect, mapa tridimensional e acelerômetro.

2.4.3 OpenNI

A organização OpenNI, liderada por indústrias sem fins lucrativos, foi formada para certificar e promover a compatibilidade e a interoperabilidade dos dispositivos de interação natural, aplicações e middleware. Um dos objetivos da organização OpenNI é acelerar a introdução de

[15] `http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/`
[16] `http://www.microsoft.com/visualstudio/pt-br`
[17] `https://github.com/OpenKinect/libfreenect`
[18] `http://openkinect.org/wiki/Main_Page`

aplicações de Interação Natural no mercado. Segundo (OPENNI, 2011), o principal objetivo do OpenNI é formar uma API padrão que permita a comunicação entre:

- Sensores de áudio e vídeo, os dispositivos que podem "ver" e "ouvir" o usuário.
- Middleware de percepção de áudio e vídeo, componentes de software que analisam o áudio e os dados visuais que são gravado a partir da uma sensor e são capazes de compreendê-las. Por exemplo, um software que recebe dados visuais, como uma imagem e retorna a localização da palma de uma mão detectada na imagem.

A OpenNI fornece um conjunto de APIs para interagir com os dispositivos e com os componentes de middleware, quebrando assim a dependência entre o sensor e o middleware. Permite aos desenvolvedores de middleware, escreverem algoritmos diretamente com o mapa tridimensional, independentemente de qual dispositivo esteja trabalhando e oferece aos fabricantes a capacidade de construir sensores compatíveis com aplicações OpenNI.

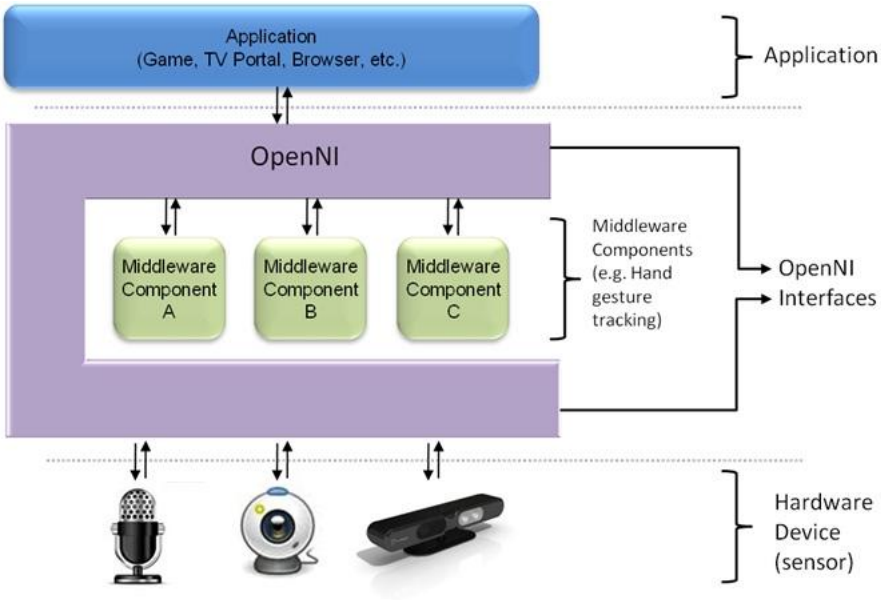


Figura 2.9: Abstração das camadas do OpenNI (OPENNI, 2011)

A Figura 2.9 mostra as três camadas do OpenNI onde o topo representa as implementações com Interações Naturais, o meio representa o OpenNI, com interfaces de comunicação que interagem com os sensores e os componentes de middleware, analisando os dados dos sensores e a última camada representa os dispositivos de hardware que captam os elementos de áudio e vídeo.

A OpenNI é uma camada abstrata que fornece uma interface para ambos os dispositivos físicos e componentes de middleware. A API permite que vários componentes sejam registra-

dos, depois referenciados como módulos e então usados para produzir e processar os dados dos sensores. A OpenNI provê suporte aos seguintes módulos:

- Módulos de sensores:
  - Sensor 3D;
  - Câmera RGB;
  - Câmera IR(infra-vermelho);
  - Dispositivo de áudio(um microfone ou vetor de microfones).
- Módulos de middleware:
  - Análise do corpo inteiro;
  - Análise do ponto da mão;
  - Detecção de gestos.
  - Analisador de cena (Analisa a imagem e consegue separar o fundo e identificar objetos na cena).

2.5 Conclusões

Neste capítulo foram apresentadas algumas centrais multimídia, dispositivos e bibliotecas de interações naturais, bem como as características e vantagens de cada um. Essa apresentação serviu para mostrar as possibilidades existentes para a implementação de uma central multimídia controlada por gestos.

O XBMC mostrou ser uma opção interessante pois é código aberto, possui uma interface simples e não necessita que o usuário tenha conhecimentos avançados ou precise sair da aplicação para realizar certas ações, onde em outras centrais teriam que ser feitas no próprio sistema operacional.

O Kinect, apesar de não ser a melhor escolha para este tipo de aplicação, devido ao seu custo, é um hardware disponível para a realização do trabalho e a biblioteca que demostrou ser a melhor opção para se trabalhar com o mesmo foi a OpenNI, por ser uma biblioteca disponível para Linux, com boa documentação e por possibilitar implementação em alto nível, sem a dependência de um hardware específico.

3 *XBMCMove*

O objetivo deste trabalho é controlar a central multimídia XBMC por gestos utilizando dispositivos de interações naturais como o Kinect. Porém não basta conectar o dispositivo e começar a utilizá-la, é necessário criar uma aplicação que consiga ler os gestos e transformá-los em comandos para a central.

Esta aplicação foi chamada de XBMCMove e sua única função é exatamente ler os gestos e convertê-los em ações para o XBMC. Para isso foram utilizados a biblioteca OpenNI para se comunicar com o dispositivo e a simulação do teclado para enviar comandos a central. Neste capítulo serão apresentados o XBMCMove, os gestos implementados e a avaliação de experiência de uso.

3.1 Arquitetura

O XBMCMove é um *middleware*, que através de dispositivos de interações naturais, como o Kinect, consegue controlar a central multimídia XBMC por gestos. Este funciona como um intermediário entre o dispositivo e o XBMC, transformando os gestos do usuário em comandos que a central consiga entender, como ilustra a Figura 3.1.

Para obter os dados do dispositivo é utilizado a biblioteca OpenNI, onde é possível capturar de forma simples informações do usuário, como posições do corpo e alguns gestos padrões. Com esses dados é possível verificar se o usuário realizou algum dos gestos padrões, próprios da OpenNI ou a partir das posições do usuário verificar se ele fez algum outro gesto personalizado, criado pelo XBMCMove. Os passos 1 e 2 da Figura 3.1 demonstra a interação com a OpenNI, onde ela recebe os dados brutos do dispositivo e os transforma em dados mais fáceis de se trabalhar.

No passo 3 utiliza-se a simulação do teclado para controlar o XBMC. Para isso, utiliza-se o próprio sistema de janelas *Xorg*[1], utilizando as bibliotecas *Xlib.h* e *keysym.h* obtidas através do

[1] `http://www.x.org/`

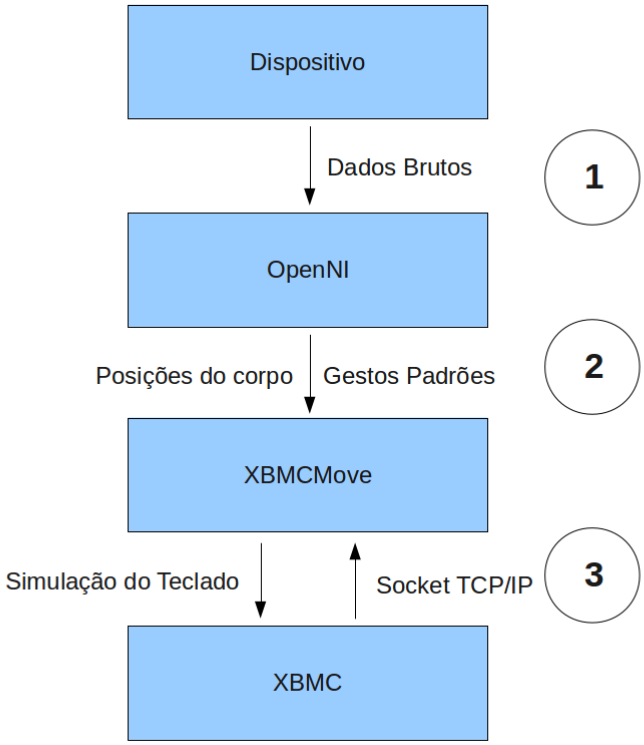


Figura 3.1: XBMCMove

pacote *libx11-dev*. O XBMC mantém aberta uma porta TCP e através dela é possível obter as ações executadas do XBMC.

Na Figura 3.2 é apresentado um fluxograma que representa todo o processo de execução do XBMCMove. No loop principal é somente tratado se o gesto será ou não capturado, dependendo do estado do XBMC.

No bloco "Cria Thread para atualizar o contexto da OpenNI" é abstraído a parte onde se cria uma thread que será responsável por atualizar as informações dos contextos da OpenNI e NITE, pegar as informações das coordenadas do corpo do usuário e verificar se os gestos personalizados criados pelo XBMCMove foram executados. Maiores detalhes sobre o processo podem ser encontrados no Anexo B.

3.2 Gestos Reconhecidos pelo XBMCMove

Determinar qual gesto deva ser feito para cada ação não é uma tarefa trivial. O gesto deve ser fácil e intuitivo, mas não pode ser muito simples ao ponto do usuário fazer um movimento qualquer e a aplicação entender como um gesto que o usuário não queira. A biblioteca OpenNI possui alguns gestos implementados que atende esses requisitos e são de uso genérico, preci-

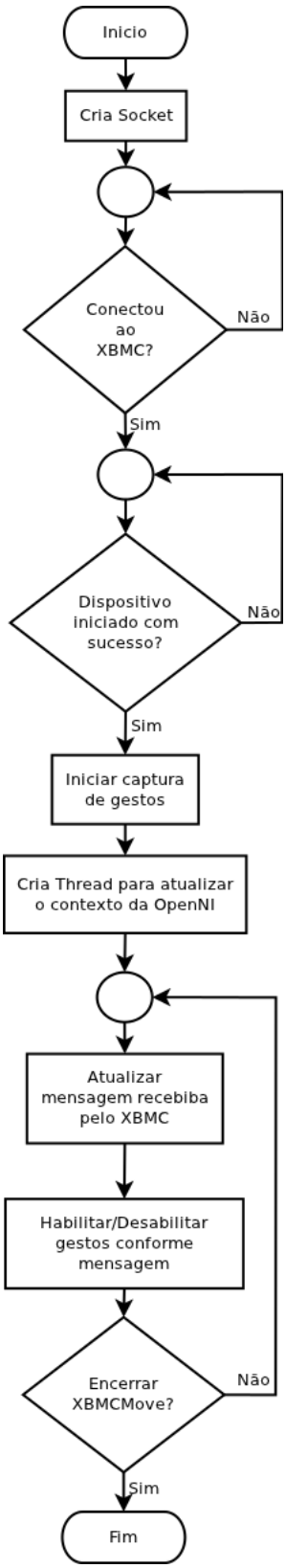


Figura 3.2: Fluxograma do XBMCMove

sando de certos ajustes para a aplicação. No XBMCMove, além destes gestos, foram implementados gestos próprios, sendo estes os gestos de volume e para a fechar a aplicação. Nesta seção serão descritos os gestos usados pela aplicação.

3.2.1 Gestos Direcionais

Os **gestos direcionais** são utilizados nos menus do XBMC, para mover o cursor para direita, esquerda, cima e para baixo. Para realizar os gestos direcionais a mão deve estar acima da barriga e a pouco mais de 50 cm a frente do usuário. A mão deve ir na direção desejada e depois deve retornar imediatamente ao ponto de origem, pois a aplicação pode entender como dois gestos. Por exemplo, o usuário move a mão para direita, deixa a mão parada nesta posição por alguns segundos e ao retornar para o ponto de origem a aplicação irá entender como um gesto para a esquerda. Outro problema ocorre quando o usuário não retorna para o ponto de origem, movimentando a mão para a direção desejada e abaixando a mão logo em seguida, fazendo com que a aplicação entenda como um gesto para baixo.

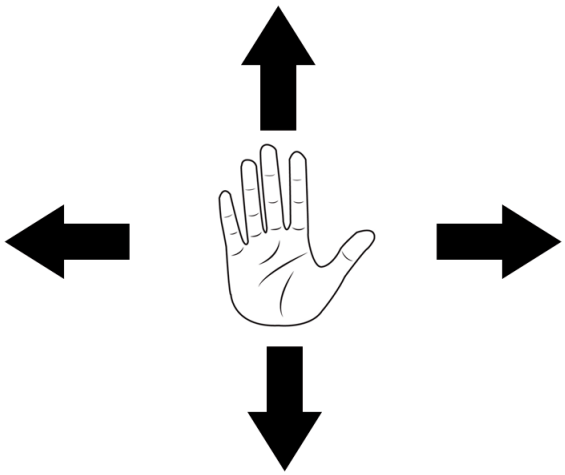


Figura 3.3: Gestos Direcionais

3.2.2 Gesto de Clique e Gesto de Tchau

- O **gesto de clique** é usado para selecionar itens do menu e retornar a reproduzir um vídeo pausado. Para executar o gesto de clique basta, com a mão levantada pelo menos acima da barriga, movimentar a mão para frente em poucos centímetros e depois retornar ao ponto de origem.

- O **gesto de tchau** é usado para pausar um vídeo em reprodução, voltar para o menu quando vídeo estiver pausado e voltar ao menu anterior quando navegando pelos me-

nus do XBMC. Para executar o gesto de tchau basta acenar por alguns segundos para o dispositivo. Fazendo movimentos rápidos de uma lado para o outro.

3.2.3 Gesto de Volume e Gesto Fechar

- O **gesto de volume** é usado para aumentar e diminuir o volume do XBMC e pode ser executado a qualquer momento. Para realizar o gesto basta, levantar as duas mãos, pelo menos acima da barriga, e ao avistar a barra de volume, afastar ou aproximar as mãos horizontalmente para aumentar o volume ou diminuir o volume, respectivamente.

(a) Aumentar Volume (b) Diminuir Volume

Figura 3.4: Gestos para Aumentar e Diminuir o Volume

- O **gesto fechar** permite finalizar o XBMC e pode ser realizado a qualquer momento, basta fazer um "X" com os braços colocando as mãos na altura dos ombros por alguns instantes.

3.2.4 Gesto para Adiantar/Retroceder um Vídeo

Os **gestos para adiantar e retroceder um vídeo** podem ser realizados somente durante a execução de um vídeo. Para avançar um vídeo basta levantar a mão pelo menos acima da barriga e fazer dois ou mais círculos no sentido horário e para retroceder fazer círculos no sentido anti-horário. O movimento se assemelha ao movimento de girar um volante de carro com uma das mãos e com a mão aberta.

3.3 Comunicação com o XBMC

É necessário saber o estado em que se encontra o XBMC, pois algumas teclas assumem diferentes comportamentos nos diferentes estados do XBMC.

O XBMC abre um socket TCP/IP ao se iniciar, por onde são enviados informações sobre o estado do XBMC e também é possível enviar instruções ao XBMC. O protocolo de comunicação usado pelo XBMC é o *JSON-RPC*[2], um protocolo de chamada de procedimento

[2] `http://json-rpc.org/`

remoto. É possível ler as mensagens enviadas pelo XBMC e através de palavras chaves definir o estado do XBMC.

```
1 {"jsonrpc":"2.0","method":"Player.OnPlay","params":{"data":{"item":{"type":"
    movie"},"player":{"playerid":1,"speed":1},"title":""},"sender":"xbmc"}}
```

Figura 3.5: Mensagem enviada pelo XBMC

A figura 3.5 mostra um exemplo de mensagem enviada pelo XBMC. Dentro delas é possível procurar por palavras chaves que mostram o estado do XBMC, nessa mensagem, por exemplo, é possível ver as palavras chaves *"OnPlay"* e *"movie"*, com isso conseguimos saber que o XBMC está reproduzindo um vídeo. A tabela 3.1 mostra as palavras chaves e seus respectivos estados do XBMC 11.0.

Palavra chave	**Descrição**
OnPlay	Reproduzindo
OnPause	Pausado
OnStop	Parado, Voltou para o menu
OnSpeedChanged	Avançando/Retrocedendo
OnQuit	XBMC Finalizado

Tabela 3.1: Palavras chave e estados do XBMC 11.0

3.4 Avaliação da experiência de uso do XBMCMove

Para validar o funcionamento do XBMCMove, bem como sua experiência de uso, foram montados cenários de testes sendo estes: instalação, configuração e uso.

Selecionou-se um grupo de pessoas para realizar todos os cenários. Foram feitos alguns slides mostrando os procedimentos para cada etapa. Na cenário de instalação o avaliado tinha que instalar o XBMC, a OpenNI e o XBMCMove. Para a instalação do XBMC e OpenNI foram utilizados os métodos de instalação indicados na documentação oficial, todos por linha de comando. para criar um padrão a instalação do XBMCMove foi feita com os mesmos padrões da instalação da OpenNI. Para avaliar a execução do XBMCMove o avaliado tinha que executar o XBMCMove após a instalação.

Os procedimentos que os avaliados tiveram que realizar, mostrados em forma de apresentação em slides, estão no Anexo D.

3.4.1 Resultados da Avaliação

Foram realizadas avaliações com onze pessoas, sendo que uma delas foi prejudicada por diversos fechamentos do XBMCMove, não permitindo o usuário avaliar a parte de gestos do XBMCMove. As outras em grande maioria sofreu em algum momento com o problema de fechamento, mas não prejudicou as avaliações.

A grande maioria dos avaliados tinha entre 18 e 25 anos, eram do sexo masculino e tinham um nível intermediário de experiência com sistemas Linux, sendo que apenas dois tinham nível iniciante. Não houve problemas na parte da instalação e somente dois dos avaliados iniciantes tiveram mais dificuldades. A maioria também não tinha até então entrado em contato com o XBMC ou com o Kinect, o que demonstrou que mesmo usuários totalmente leigos podem utilizar a solução do XBMCMove. Os gráficos mostrando os dados de idade, sexo e experiência com Linux, Kinect e XBMC estão na Figura 3.6.

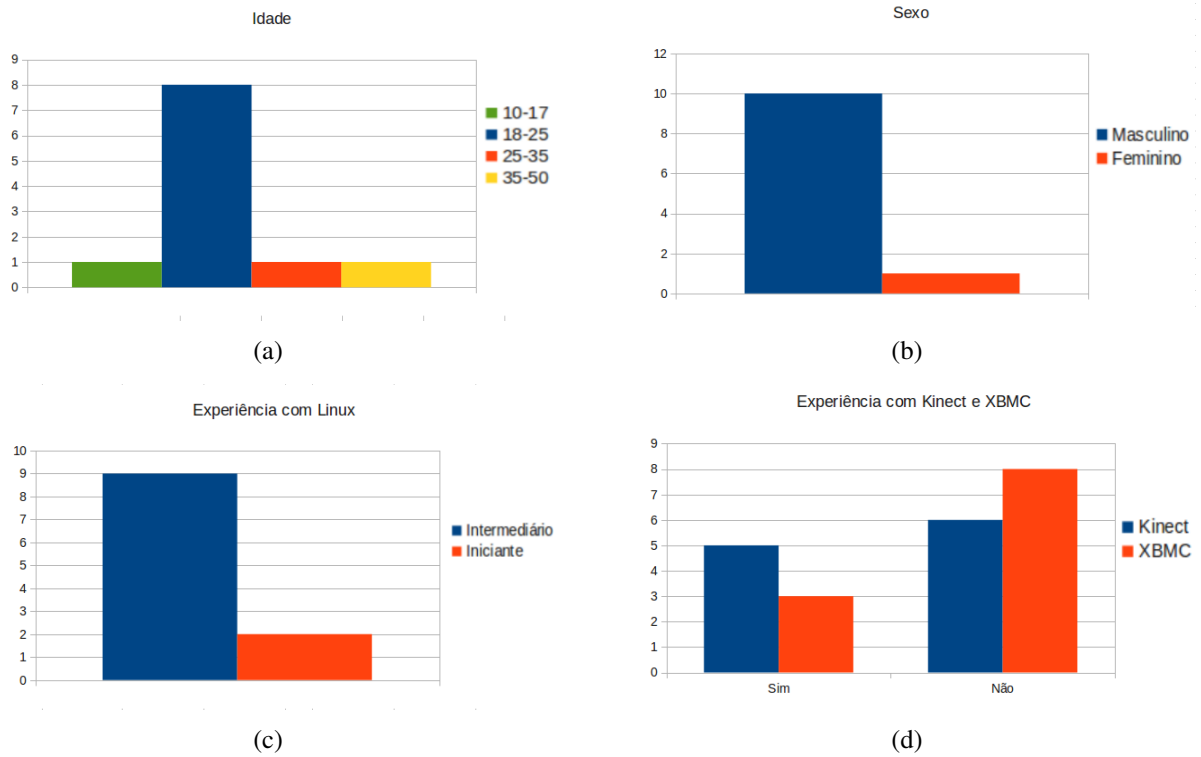


Figura 3.6: Gráficos de Idade,Sexo e Experiência com Linux, Kinect e XBMC

Como mostra o gráfico da Figura 3.7, os avaliados acharam fácil a instalação e principalmente a execução do XBMCMove, apenas os usuários iniciantes em Linux tiveram alguns problemas, mas conseguiram concluir a instalação do XBMC, OpenNI e XBMCMove. A maior dificuldade dos usuários iniciantes é a utilização do terminal do Linux, onde o avaliador teve que ajudar.

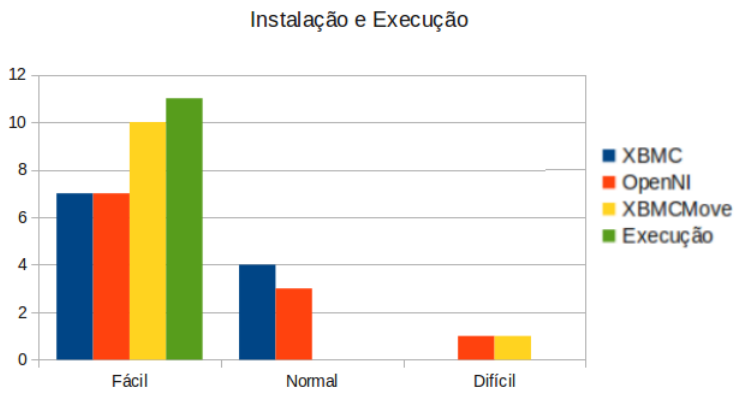


Figura 3.7: Instalação do XBMC, OpenNI e instalação e execução do XBMCMove

Após a instalação, os avaliados realizaram alguns exercícios para avaliar os gestos do XBMCMove. Eles consistiam em fazer o usuário realizar algumas ações no XBMC através dos gestos do middleware e depois avaliar se ele teve alguma dificuldade para realizar os gestos e o nível de dificuldade (Fácil, Normal ou Difícil) dos gestos realizados na opinião do avaliado. Como mostra os gráficos (a) e (b) da Figura 3.8 as avaliações sobre os gestos foram positivas, sendo que a maioria achou fácil a realização dos gestos e não tiveram problemas com eles. Apesar de que os avaliados em geral não acertarão muitos dos gestos na primeira tentativa, a curva de aprendizado foi rápida e conseguiram realizar os gestos em poucas tentativas. Depois de realizar os exercícios foi perguntado se o avaliado achou a utilização do XBMCMove confortável ou não. Como mostra o gráfico (c) da Figura 3.8 somente um dos avaliados não se sentiu confortável utilizando o XBMCMove.

Como último item da avaliação foi pedido aos avaliados que dessem uma nota de 1 a 10 ao XBMCMove. Os avaliados avaliaram o XBMCMove com uma média de 8, sendo a nota mais baixa 5 do usuário que não conseguiu realizar corretamente a avaliação e nota 10 por três dos avaliados.

Durante a avaliação alguns problemas foram percebidos, um dos problemas foi a mal formulação dos slides mostrando todo o procedimento, desde instalação até a reprodução dos gesto, gerando várias dúvidas entre os avaliados, tendo o avaliador que ajuda-los várias vezes durante a avaliação. O outro problema encontrado foi um erro de implementação percebido durante as avaliações, onde o XBMCMove encerra a execução do XBMC por detectar o gesto de fechar sem o avaliado o ter realizado. Este problema não ocorreu durante os testes da aplicação.

Os problemas prejudicaram algumas avaliações, mas foram alertados a avaliar os gestos e não levar em conta o fechamento do XBMCMove como uma dificuldade para realizá-los. As dificuldades com o fechamento do XBMCMove foram abordadas como um problema de

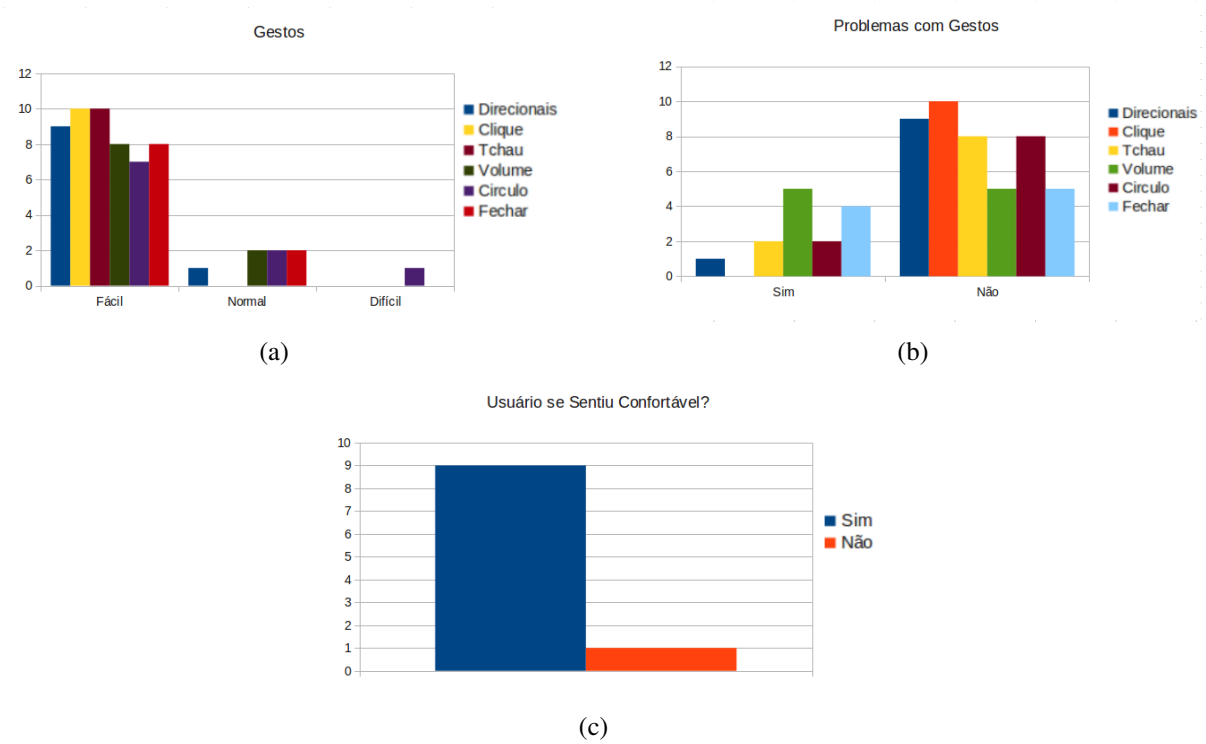


Figura 3.8: Gráficos de dados coletados sobre os gestos do XBMCMove

implementação, como os próprios avaliados perceberam durante o procedimento.

Como mostra os resultados da avaliação, o XBMCMove apresenta problemas que devem ser corrigidos, mas consegue realizar bem as funcionalidades básicas do XBMC, permitindo controlá-lo por gestos.

4 Conclusões

O XBMCMove demonstrou ser uma boa solução para controlar o XBMC por interações naturais. Os gestos escolhidos demonstraram ser fáceis e intuitivos de serem reproduzidos e o XBMC mesmo sem ser feito para ser controlado por gestos, apresenta uma ótima interface para esse tipo de interação.

O XBMCMove também apresentou problemas de implementação que precisam ser melhorados. O principal problema é a detecção errada do gesto de fechar, que acaba causando o fechamento do XBMC sem o usuário querer e os outros problemas são de melhorias nos próprios gestos e a falta de gestos para várias ações do XBMC que precisam ser adicionados.

A avaliação com os usuários apresentou diversas opiniões diferentes, na grande maioria positivas mesmo com os problemas ocorridos, o que demonstrou que o XBMCMove é uma boa solução para controlar o XBMC por interações naturais.

O estudo apresentado nas revisão bibliográfica foi primordial para a implementação da solução, pois foi possível escolher a melhor opção de central multimídia e biblioteca para se trabalhar. Possibilitando a realização de todos os objetivos propostos. Porém ela ainda está incompleta, necessitando de alterações e melhorias que poderão ser feitas em trabalhos futuros.

O resultado final apresentado nesse trabalho, pode ser considerado como um ponto de partida para outros trabalhos com interações naturais com dispositivos como o Kinect e que o conteúdo os ajude a melhorar o XBMCMove ou apresentar outras soluções com interações naturais em geral.

Como sugestão para trabalhos futuros, pode-se citar as seguintes melhorias para o XBMCMove:

- **Melhorar detecção dos gestos atuais:** Os gestos atuais podem ser melhorados para ter uma maior compatibilidade com usuários com tipos físicos diferentes;
- **Adicionar novos gestos as funções do XBMC:** O XBMCMove por enquanto somente realiza as funções básicas do XBMC, ainda pode ser adicionado gestos para outras funções

ou gestos como atalhos para alguma função ou menu do XBMC;

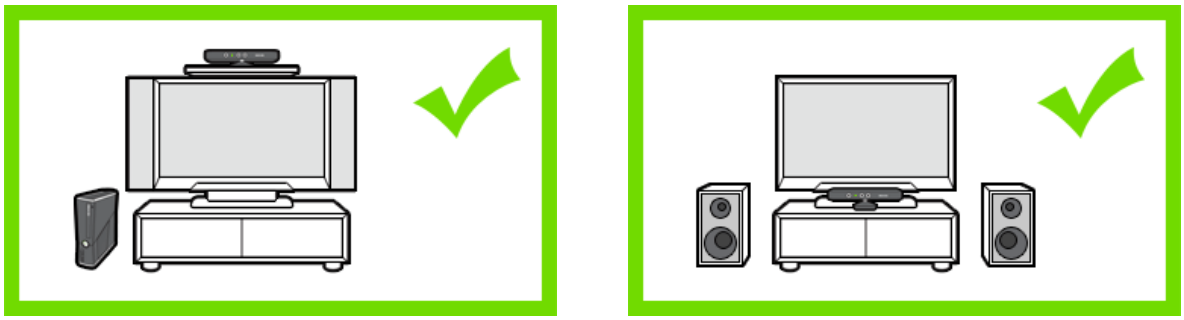
- **Permitir o usuário personalizar os gestos:** Criar um software capaz de ler um gesto personalizado do usuário e depois permitir que ele escolha qual ação o gesto vai fazer.

ANEXO A – Instalação e Execução

Nesta Seção será discutido os passos para a instalação do dispositivo de interação natural, da central multimídia XBMC, da biblioteca OpenNI e instalação e execução do XBMCMove.

Instalação do Dispositivo

É necessário instalar o dispositivo em um lugar adequado, alinhado com a TV ou monitor, pois o usuário deve estar prestando a atenção na tela e não no dispositivo. O usuário irá acabar fazendo os gestos para a TV e se o dispositivo não estiver não posição adequada pode acabar não identificando corretamente o gesto. A figura D.1 mostra a posição mais adequada para a utilização do dispositivo.



(a) (b)

Figura A.1: Posicionamento do Dispositivo (MICROSOFT, 2010)

Os dispositivos de interações naturais são geralmente conectados a porta USB do computador e podem necessitar também de uma fonte de alimentação externa. No caso do Kinect, é necessário a utilização de um adaptador que possui uma conexão USB e uma fonte de alimentação Bivolt. Esse adaptador vem com o Kinect.

Outro fator importante para um bom funcionamento do dispositivo é respeitar a distância de funcionamento do mesmo. No caso do Kinect, o usuário deve ficar a pelo menos 1,5 metros do dispositivo para um bom funcionamento, a distância máxima testada com o XBMCMove foi de

2,5 metros e a aplicação funcionou perfeitamente. Também não deve haver nenhum obstáculo entre o usuário e o dispositivo.

Instalação do XBMC

O XBMC pode ser obtido de várias formas, mas iremos demonstrar somente a opção mais usual e prática de instalação no Ubuntu. Todos os comandos mostrados devem ser executados em um terminal e precisam de senha de *root* (Administrador).

Para começar precisamos adicionar o repositório do XBMC ao nosso sistema e depois atualizar nossa lista de pacotes. Para fazer isso devemos realizar os seguintes comandos no Terminal:

```
1 sudo add-apt-repository ppa:team-xbmc
2 sudo apt-get -y update
```

Figura A.2: Adicionar repositório do XBMC

Depois basta instalar o XBMC e seus dependências com o comando abaixo:

```
1 sudo apt-get -y install python-software-properties pkg-config xbmc xbmc-
    standalone
```

Figura A.3: Instalar o XBMC

Pronto, o XBMC já esta instalado e pronto para ser usado.

Instalação e execução do XBMCMove

Para instalar o XBMCMove é necessário primeiramente instalar a biblioteca OpenNI e para isso é necessário fazer o download dos arquivos *"OpenNI Build for Ubuntu"* e *"PrimeSense NITE Build for Ubuntu"* referentes ao tipo de arquitetura do seu computador (32 Bits ou 64 Bits) no site (`http://www.openni.org/Downloads/OpenNIModules.aspx`). Com os arquivos em mãos basta descompactar o *"OpenNI Build for Ubuntu"*, entrar em um terminal e ir até a pasta descompactada e executar o comando abaixo:

Depois basta descompactar o *"PrimeSense NITE Build for Ubuntu"*, entrar em um terminal e ir até a pasta descompactada e executar o comando acima novamente.

```
sudo ./install.sh
```

Figura A.4: Instalar a biblioteca OpenNI

Após instalar a biblioteca OpenNI e o XBMC, já podemos instalar o XBMCMove. Para instalar o XBMCMove basta extrai-lo, entrar na pasta referente a arquitetura do seu sistema (32 Bits ou 64 Bits) e executar o arquivo *"install.sh"* com o mesmo comando usado para instalar o OpenNI no terminal.

Pronto, o XBMCMove já esta instalado, para executá-lo basta ir no menu Aplicativos e procurar por XBMCMove.

ANEXO B – Implementação do XBMCMove

Existe duas maneiras para implementar gestos utilizando a biblioteca OpenNI, a biblioteca NITE, também fornecida pela OpenNI, que permite usar gestos já implementados, onde somente é necessário fazer pequenos ajustes dependendo da aplicação, ou a própria biblioteca OpenNI, que permite criar gestos somente com as posições do corpo do usuário.

A NITE funciona como um middleware da OpenNI, possibilitando implementações de nível mais alto, onde não é necessário pegar as coordenadas, somente dizer quais gestos serão usados e dizer o que fazer quando esses gestos forem executados.

Para melhor explicar a implementação dos gestos será dividido em duas seções, uma com a implementação com a NITE e outra somente com a OpenNI.

Implementação com OpenNI

Nesta seção iremos discutir os principais elementos para a implementação com a biblioteca OpenNI, para maiores detalhes deve-se olhar a documentação do OpenNI disponibilizada em seu site e o código do XBMCMove.

O Principal objeto da OpenNI é o contexto e é nele que ficam as informações e configurações do dispositivo. Para utilizá-lo é necessário inicializá-lo e para isso existem duas opções: Utilizar um arquivo de configuração XML ou configurar diretamente no código. No XBMCMove foi utilizado o arquivo de configuração mostrado na figura B.1.

Com o arquivo de configuração, basta iniciar o contexto como demostrado na figura B.2.

Após iniciar o contexto devemos dizer com que iremos trabalhar e especificar os retornos de chamada(Callbacks). No nosso caso, iremos trabalhar com as posições do usuário, então iremos usar um gerador de usuário(UserGenerator), que é responsável por identificar os usuários, calibrá-los e enviar as posições do corpo. A criação do UserGenerator é descrita nas linhas 2 e 3 na figura B.3.

```
<OpenNI>
   <Licenses> <!-- Especificando Licenca, no caso, para uso nao comercial e usado
        essa chave -->
      <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
   </Licenses>
   <Log writeToConsole="false" writeToFile="false"><!-- Configuracao do log que
       sera mostrado -->
      <LogLevel value="3"/> <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (
          default) -->
      <Masks>
         <Mask name="ALL" on="false"/>
      </Masks>
      <Dumps>
      </Dumps>
   </Log>
   <ProductionNodes> <!-- Configuracao dos Nos que serao carregados -->
      <Node type="Depth"> <!-- Sensor de Profundidade -->
         <Configuration>
            <Mirror on="true"/> <!-- Espelhar -->
         </Configuration>
      </Node>
   </ProductionNodes>
</OpenNI>
```

Figura B.1: Arquivo de Configuração XML

```
// Arquivo XML de configuracao do OpenNI
#define XML_FILE "/etc/xbmcmove/Sample-Tracking.xml"

XnStatus rc; // Variavel para verificar erros
xn::Context g_Context; //Contexto

rc = g_Context.InitFromXmlFile(XML_FILE, g_ScriptNode, &errors);
if (rc == XN_STATUS_NO_NODE_PRESENT) {
   // Nenhum No presente
} else if (rc != XN_STATUS_OK) {
   // Se Houver algum erro mostra mensagem de erro e retorna o erro
   printf("[ERROR] Open failed: %s\n", xnGetStatusString(rc));
   return rc;
}
```

Figura B.2: Código para iniciar o Contexto

Precisamos agora especificar os Callbacks para o UserGenarator, os Callbacks são funções que são lançadas após um evento, mas precisamos dizer qual evento ira lançar essa função. Dentro dessas funções(Callbacks) dizemos o que queremos fazer quando acontecer esse evento. Por exemplo: Pode-se criar um callback chamado "NewUser" e outro "LostUser" e dizer que o callback "NewUser" sera lançado quando um novo usuário for detectado pelo UserGenerator e o "LostUser" quando o usuário não estiver mais na frente do dispositivo. Dentro do "NewUser" colocamos um código para acender uma lâmpada e em "LostUser" outro código para apagar a luz. A criação dos Callbacks e o registro aos eventos do UserGenarator é demostrado nas linhas 11 à 21 da figura B.3.

Após registrar os Callbacks, precisamos dizer que iremos pegar as posições do corpo. E para isso precisamos verificar se o UserGenerator é capaz de pegar as posições e se ele necessita de uma posição inicial para fazer isso. Essa verificação deve ser feita porque esses parâmetros podem acabar variando em cada versão do OpenNI. Após fazer a verificação temos que dizer quais posições do corpo iremos pegar. O suporte a esqueleto é verificado nas linhas 6 à 9, a verificação da necessidade de pose para calibração é demostrada nas linhas 24 à 32 e um exemplo de quais partes do corpo iremos usar na linha 34 todos da figura B.3.

Com isso podemos dizer ao contexto que ele já pode começar a capturar, como demostrado na linha 3 da figura B.4. Depois precisamos atualizar o contexto em tempos em tempos para ele gerar os eventos(linha 7), pegar as posições do corpo(linha 14 à 18) e verificar se nossos gestos foram realizados(linha 20 e 21). Caso a atualização do contexto não seja feita os eventos não serão gerados e as posições não serão atualizadas.

O gesto de fechar verifica se as mãos estão acima do tronco e se estão atravessadas formando um "X" com os braços, caso o usuário fique assim por alguns instantes o XBMC é finalizado. O gesto volume verifica se as mãos estão acima do tronco e se não estão atravessadas e depois verifica se o usuário afasta ou aproxima as mãos para aumentar ou diminuir o volume a partir da posição inicial das mãos. Caso o usuário abaixe as mãos e levante novamente a distância sera calculada com a nova posição inicial das mãos. A implementação do gesto Fechar e demostrado na figura B.5 e o gesto Volume na figura B.6.

Implementação com a NITE

Nesta seção iremos discutir os principais elementos para a implementação com a biblioteca OpenNI utilizando o pacote NITE, para maiores detalhes deve-se olhar a documentação do OpenNI e da NITE disponibilizada em seu site e o código do XBMCMove.

```
// Diz que iremos trabalhar com usuarios:
xn::UserGenerator g_UserGenerator;
rc = g_UserGenerator.Create(g_Context);

// Verifica se o Gerador de Usuarios suporta esqueleto
if (!g_UserGenerator.IsCapabilitySupported(XN_CAPABILITY_SKELETON)) {
   printf("Supplied user generator doesn't support skeleton\n");
   return 1;
}

// Cria os Callbacks:
XnCallbackHandle hUserCallbacks, hCalibrationStart, hCalibrationComplete,
    hPoseDetected, hCalibrationInProgress, hPoseInProgress;
//Registra os callbacks aos eventos do UserGenerator
rc = g_UserGenerator.RegisterUserCallbacks(User_NewUser, User_LostUser, NULL,
    hUserCallbacks);
rc = g_UserGenerator.GetSkeletonCap().RegisterToCalibrationStart(
    UserCalibration_CalibrationStart, NULL, hCalibrationStart);
rc = g_UserGenerator.GetSkeletonCap().RegisterToCalibrationComplete(
    UserCalibration_CalibrationComplete, NULL, hCalibrationComplete);
if (rc != XN_STATUS_OK) {
   // Se Houver algum erro mostra mensagem de erro e retorna o erro
   printf("[ERROR] Callbacks Register failed: %s\n", xnGetStatusString(rc));
   return rc;
}

// Verifica se o Gerador de Usuarios necessita de uma posicao inicial para
    calibrar o usuario
if (g_UserGenerator.GetSkeletonCap().NeedPoseForCalibration()) {
   g_bNeedPose = TRUE;
   if (!g_UserGenerator.IsCapabilitySupported(XN_CAPABILITY_POSE_DETECTION)) {
       printf("Pose required, but not supported\n");
       return 1;
   }
   rc = g_UserGenerator.GetPoseDetectionCap().RegisterToPoseDetected(
       UserPose_PoseDetected, NULL, hPoseDetected);
   g_UserGenerator.GetSkeletonCap().GetCalibrationPose("Psi"); // Especifica qual
        posicao o usuario deve fazer
}
// Diz que queremos usar todas as partes do corpo:
g_UserGenerator.GetSkeletonCap().SetSkeletonProfile(XN_SKEL_PROFILE_ALL);
```

Figura B.3: Código para iniciar o UserGenerator

```
...
//Comeca a capturar e gerar os eventos
rc = g_Context.StartGeneratingAll();

while(TRUE){
   //Atualiza Contexto
   g_Context2.WaitAnyUpdateAll();
   //Pega posicoes do esqueleto do usuario
   SkeletonCapability skelCap = g_UserGenerator.GetSkeletonCap();
   //Pega numero de usuarios e suas identificacoes
   nUsers = g_UserGenerator.GetNumberOfUsers();
   g_UserGenerator.GetUsers(userIds, nUsers);
   id = userIds[0];
   //Pega posicoes do primeiro usuario
   getJointImgCoordinates(skelCap, id, XN_SKEL_HEAD, head); // Pega posicao da
       cabeca
   getJointImgCoordinates(skelCap, id, XN_SKEL_TORSO, t); // Tronco
   getJointImgCoordinates(skelCap, id, XN_SKEL_RIGHT_HAND, rh); // Mao Direita
   getJointImgCoordinates(skelCap, id, XN_SKEL_LEFT_HAND, lh); // Mao Esquerda
   // Verifica se foi realizado gesto de Volume e Fechar
   gestoVolume();
   gestoFinal();
}
```

Figura B.4: Código para iniciar captura e atualização do Contexto

```
void gestoFinal() {
    if (!gFechar) {
        if (lh[0] != rh[0]) {
            if (((rh[1] < (t[1])) && (lh[1] < (t[1]))) && (rh[0] < lh[0])) {
                fecharCont++;
                if (fecharCont >= 10) {
                    fecharXBMC();
                    gFechar = true;
                }
            } else {fecharCont = 0;}
        }
    }
}
```

Figura B.5: Implementação dos gesto Fechar

```
void gestoVolume() {
 if (lh[0] != rh[0]) {
   if (((rh[1] < (t[1])) && (lh[1] < (t[1]))) && (rh[0] > lh[0])) {
     if (posInit[0] != 0) {
         if (volumeCont >= 20) {
             if ((rh[0] - lh[0]) >= (distInit * 1.9)) {
                 teclar->simular(XK_plus);
                 volumeCont = 20;
             } else {
                 if ((rh[0] - lh[0]) <= (distInit * 0.1)) {
                     teclar->simular(XK_minus);
                     volumeCont = 20;
                 } else {
                     if (((rh[0] - lh[0]) > (distInit * 1.4)) && ((rh[0] - lh[0]) <
                          (distInit * 1.9))) {
                         teclar->simular(XK_plus);
                         volumeCont = 15;
                     } else {
                         if ((rh[0] - lh[0]) < (distInit * 0.6)) {
                             teclar->simular(XK_minus);
                             volumeCont = 15;
                         } else {
                             if (((rh[0] - lh[0]) > (distInit * 0.6)) && ((rh[0] -
                                 lh[0]) < (distInit))) {
                                 teclar->simular(XK_minus);
                                 volumeCont = 5;
                             } else {
                                 teclar->simular(XK_plus);
                                 volumeCont = 5;
                             }
                         }
                     }
                 }
             }
         } else {volumeCont++;}
     } else {
         posInit[0] = lh[0];
         posInit[1] = rh[0];
         distInit = posInit[1] - posInit[0];
         volumeCont = 1;
     }
    } else {volumeCont = distInit = posInit[0] = posInit[1] = 0;}
  }
}
```

Figura B.6: Implementação dos gesto Volume

A NITE é parte da biblioteca OpenNI e possui algumas facilidades e gestos prontos, que permite uma implementação mais rápida, caso os gestos padrões dela forem úteis na aplicação desejada, se for desejável a criação de gestos ou for utilizar gestos prontos de terceiros. Para facilitar a manipulação dos gestos a NITE possui um gerenciador de seção que permite gerenciar os gestos de forma mais rápida. Com o gerenciador de seção é possível dizer quando o gesto poderá ou não ser executado.

A implementação com a NITE é bem parecida com a OpenNI, deve-se criar um contexto e os callbacks igualmente. Diferenciando-se pelo uso do gerenciador de seção e pelas classes dos gestos personalizados.

A criação do contexto é exatamente igual a OpenNI demostrada na figura B.2, mas para usar a NITE devemos utilizar um gerador de gestos e mãos(GestureGenerator e HandGenerator) que permitem gerenciar gestos personalizados e que são usados pela NITE. O código para a criação desses geradores e demostrada na figura B.7.

```
1  // Cria os geradores
2  rc = g_GestureGenerator.Create(g_Context);
3  rc = g_HandsGenerator.Create(g_Context);
4  if (rc != XN_STATUS_OK) {
5     return rc;
6  }
7
8  // Registrar callbacks
9  g_GestureGenerator.RegisterGestureCallbacks(Gesture_Recognized, Gesture_Process,
       NULL, h1);
10 g_HandsGenerator.RegisterHandCallbacks(Hand_Create, Hand_Update, Hand_Destroy,
      NULL, h2);
```

Figura B.7: Criação do GestureGenerator e HandGenerator

Com o geradores de gestos e mãos é possível iniciar o gerenciador de seção(SessionManager) passando o contexto e o gesto que irá disparar o inicio da seção. No caso do XBMCMove o gesto utilizado é o RaiseHand(Levantar as mãos), ou seja, assim que o usuário levantar uma das mãos a seção ira começar. Depois, deve-se registrar os callbacks de inicio e fim da seção, como demostrado na figura B.8.

Com o gerenciador de seção pronto precisamos adicionar os gestos que iremos usar a ele. Iremos utilizar os gestos direcionais, Gesto de Clique, Tchau e Avançar/Retroceder o vídeo. Para os gestos direcionais utilizamos o Detector de direção(SwipeDetector), que possui exatamente essa função.

Para o gesto de clique utilizamos o TrackPad que permite criar menus onde a mão funciona como um mouse. Mas somente utilizamos a parte de reconhecimento do movimento para frente,

```
1  //  Cria o Gerenciador de Secao
2  g_pSessionManager = new XnVSessionManager();
3  rc = g_pSessionManager->Initialize(&g_Context, "RaiseHand", NULL);
4  if (rc != XN_STATUS_OK) {
5     return rc;
6  }
7  // Registra os callbacks de inicio e fim da secao
8  g_pSessionManager->RegisterSession(NULL, &SessionStart, &SessionEnd);
```

Figura B.8: Criação do Gerenciador de Seção

que simula o clique do mouse ou o "ENTER" em nossa aplicação.

Para o gesto de tchau utilizamos o WaveDetector, que somente detecta esse movimento de tchau com as mãos. E para o os gestos de avançar e retroceder usamos o detector de círculos (CircleDetector). O código para adicionar os gestos ao gerenciador de seção é descrito no código da figura B.9.

```
1  // Cria detector de gestos direcionais
2  g_pSwipeD = new XnVSwipeDetector(true);
3  g_pSwipeD->RegisterSwipeLeft(NULL, SwipeLeftCB);
4  g_pSwipeD->RegisterSwipeRight(NULL, SwipeRightCB);
5  g_pSwipeD->RegisterSwipeUp(NULL, SwipeUpCB);
6  g_pSwipeD->RegisterSwipeDown(NULL, SwipeDownCB);
7
8  // Cria TrackPad
9  g_pTrackPad = new XnVSelectableSlider2D(g_TP_XDim, g_TP_YDim);
10
11 // Cria Detector de Tchau
12 wc = new XnVWaveDetector();
13 wc->RegisterWave(NULL, OnWaveCB);
14
15 // Cria o Detector de circulos e Registra os Callbacks
16 g_pCircle = new XnVCircleDetector;
17 g_pCircle->RegisterCircle(NULL, &CircleCB);
18 g_pCircle->RegisterNoCircle(NULL, &NoCircleCB);
19 g_pCircle->RegisterPrimaryPointCreate(NULL, &Circle_PrimaryCreate);
20 g_pCircle->RegisterPrimaryPointDestroy(NULL, &Circle_PrimaryDestroy);
21
22 //Adiciona os gestos ao Gerenciador de secao
23 g_pSessionManager->AddListener(g_pCircle);
24 g_pSessionManager->AddListener(wc);
25 g_pSessionManager->AddListener(g_pSwipeD);
26 g_pSessionManager->AddListener(g_pTrackPad);
```

Figura B.9: Adicionar gestos ao Gerenciador de Seção

Após registrar ao Gerenciador de Seção os gestos que iremos utilizar, adicionamos o gesto "RaiseHand" ao GestureGenerator, linha 3 do código da figura B.10, para o Gerenciador de Seção iniciar a seção assim que o usuário levantar uma das mãos. Com isso iniciamos a captura e

a geração de eventos com o contexto, demostrado na linha 5. Depois de inicializado é necessário ficar atualizando o contexto e o Gerenciador de Seção como no loop da linha 7.

```
1  ...
2  //Adiciona o gesto de levantar as maos
3  g_GestureGenerator.AddGesture("RaiseHand", NULL);
4  //Comeca a capturar e gerar os eventos
5  rc = g_Context.StartGeneratingAll();
6
7  while(TRUE){
8     // Le os proximos dados disponiveis
9          g_Context.WaitAnyUpdateAll();
10         // Processa as informacoes
11         g_pSessionManager->Update(&g_Context);
12 }
```

Figura B.10: Código para iniciar captura e atualização do Contexto e do Gerenciador de Seção

Com essas etapas conseguimos utilizar os gestos descritos e trabalhar com eles utilizando os callbacks e o gerenciador de seção. Todos os callbacks utilizados pela OpenNI e NITE estão no Anexo XX.

Simulação do Teclado

Para simular o teclado o XBMCMove utiliza o próprio sistema de janelas *Xorg*[1], utilizando as bibliotecas *Xlib.h* e *keysym.h* obtidos através do pacote *libx11-dev*. o Código da figura B.11 demonstra como é feito a simulação do teclado.

A função *createKeyEvent* do código cria um evento de teclado, definindo os parâmetros do evento. E dentro do *main* na linha 31 é definido qual tecla sera simulada através do *keycode*, código da tecla que pode ser obtido em *"/usr/include/X11/keysymdef.h"* e nas linhas 46 e 50 são enviados os eventos a janela para o pressionamento e a liberação da tecla.

[1] http://www.x.org/

```
#include <X11/Xlib.h>
#include <X11/keysym.h>

XKeyEvent createKeyEvent(Display *display, Window &win, Window &winRoot, bool
    press, int keycode, int modifiers) {
    XKeyEvent event;

    event.display = display;
    event.window = win;
    event.root = winRoot;
    event.subwindow = None;
    event.time = CurrentTime;
    event.x = 1;
    event.y = 1;
    event.x_root = 1;
    event.y_root = 1;
    event.same_screen = True;
    event.keycode = XKeysymToKeycode(display, keycode);
    event.state = modifiers;

    if (press)
        event.type = KeyPress;
    else
        event.type = KeyRelease;

    return event;
}

int main(){

   //Tecla a ser simulada, no caso a tecla p
   tecla = XK_p;

   // Obtem o display X11.
   Display *display = XOpenDisplay(0);

   // Pega a Janela root do display atual.
   Window winRoot = XDefaultRootWindow(display);

   // Procura a janela com o foco do teclado atual.
   Window winFocus;
   int revert;
   XGetInputFocus(display, &winFocus, &revert);

   // Envia um falso evento de pressionamento de tecla para a janela.
   XKeyEvent event = createKeyEvent(display, winFocus, winRoot, true, tecla, 0);
   XSendEvent(event.display, event.window, True, KeyPressMask, (XEvent *) & event
       );

   // Envia um falso evento de liberacao de tecla para a janela.
   event = createKeyEvent(display, winFocus, winRoot, false, tecla, 0);
   XSendEvent(event.display, event.window, True, KeyPressMask, (XEvent *) & event
       );

   // Fecha o Display.
   XCloseDisplay(display);
}
```

Figura B.11: Código para Simular o teclado

ANEXO C – Avaliação

Nome:

1 - Sexo: ( ) Masculino ( ) Feminino

2 - Idade:

( ) Menor de 10 anos ( ) Entre 25 e 35 anos

( ) Entre 10 e 17 anos ( ) Entre 35 e 50 anos

( ) entre 18 e 25 anos ( ) Maior de 50 anos

3 - Experiência com Linux:

( ) Nenhuma ( ) Iniciante ( ) Intermediário ( ) Avançado

4 - Já utilizou o Kinect? ( ) Sim ( ) Não

5- Já utilizou o XBMC? ( ) Sim ( ) Não

Sobre a instalação do XBMC

6 - Conseguiu concluir a instalação do XBMC sem problemas? ( ) Sim ( ) Não

7 – Você achou a instalação:

( ) Fácil ( ) Normal ( ) Difícil

Sobre a instalação do OpenNI

8 - Conseguiu concluir a instalação do OpenNI sem problemas? ( ) Sim ( ) Não

9 – Você achou a instalação:

( ) Fácil ( ) Normal ( ) Difícil

Sobre a instalação do XBMCMove

10 - Conseguiu concluir a instalação do XBMCMove sem problemas? ( ) Sim ( ) Não

11 – Você achou a instalação:

( ) Fácil ( ) Normal ( ) Difícil

Sobre a forma de execução do XBMCMove

12- Você considera a forma de execução do XBMCMove:

( ) Fácil ( ) Normal ( ) Difícil

Atenção: Não avalie os itens abaixo de forma comparativa a utilização do teclado e mouse!

Sobre os gesto de navegação no menu

13 – Você conseguiu realizar todos os itens do exercício 1? ( ) Sim ( ) Não

14 – Teve dificuldades para realizar algum gesto? ( ) Sim ( ) Não

15 – Como você avalia os gestos utilizados? ( ) Fáceis ( ) Normais ( ) Difíceis

Sobre os gestos realizados no exercício 2

16 – Você conseguiu realizar todos os itens do exercício 2? ( ) Sim ( ) Não

17 – Teve dificuldades para realizar algum gesto? ( ) Sim ( ) Não

18 – Como você avalia os gestos utilizados? ( ) Fáceis ( ) Normais ( ) Difíceis

Sobre os gestos realizados no exercício 3

19 – Você conseguiu realizar todos os itens do exercício 3? ( ) Sim ( ) Não

20 – Teve dificuldades para realizar algum gesto? ( ) Sim ( ) Não

21 – Como você avalia os gestos utilizados? ( ) Fáceis ( ) Normais ( ) Difíceis

Visão geral sobre o XBMCMove

22 – Você teve que repetir algum gesto varias vezes para ele ser reconhecido? Qual?

( ) Sim ( ) Não Gesto:

23 – Você se sentiu confortável utilizando o XBMCMove? Achou algum gesto ruim de ser executado?

( ) Achei ( ) Não, me senti confortável Gesto:

24 – Como você avalia o XBMCMove em uma nota de 1 a 10? Nota:

ANEXO D – Manual da avaliação

Slide 1

Responda as perguntas de 1 a 5 antes de começar!

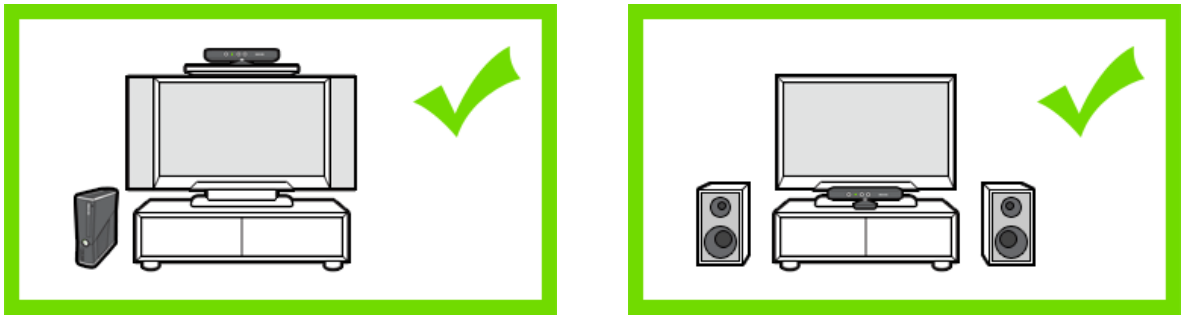
Slide 2

Instalação

Slide 3

Posicionamento do Dispositivo

O dispositivo deve estar alinhado com a TV ou monitor.



(a) (b)

Figura D.1: Posicionamento do Dispositivo (MICROSOFT, 2010)

Slide 4

Posicionamento do Usuário

- O Usuário deve ficar a pelo menos 1,5 metros do dispositivo para um melhor funcionamento.
- A distância máxima testada foi de 2,5 metros e a aplicação funcionou perfeitamente.
- Não deve haver nenhum obstáculo entre o usuário e o dispositivo.

Slide 5

Instalação do XBMC

Para instalar o XBMC:

-Abra o terminal

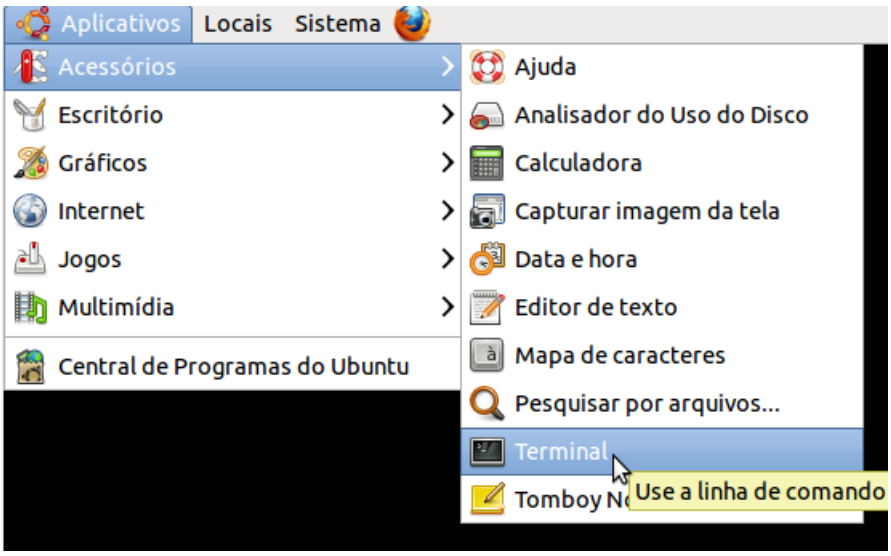


Figura D.2: Procedimento para abrir Terminal no Ubuntu

Slide 6

Instalação do XBMC

Execute os seguintes comandos:

- sudo add-apt-repository ppa:team-xbmc
- sudo apt-get -y update
- sudo apt-get -y install python-software-properties pkg-config xbmc xbmc-standalone

Senha de root para a avaliação: xbmcmove

Pronto, XBMC instalado.

Slide 7

Responda as perguntas 6 e 7 antes de continuar!

Slide 8

Instalação do OpenNI

Entre no site:

http://www.openni.org/Downloads/OpenNIModules.aspx

Baixe os arquivos:

OpenNI Binaries, Unstable, OpenNI for Ubuntu

OpenNI Middleware Binaries, Unstable, OpenNI for Ubuntu

Para a Avaliação os arquivos estão no Pendrive.

Slide 9

Instalação do OpenNI

Descompacte os arquivos;

Abre um terminal novamente e vá até a pasta descompactada e execute o arquivo "install.sh" com o comando:

sudo ./install.sh

Repita o passo anterior para a outra pasta descompactada.

Slide 10

Responda as perguntas 8 e 9 antes de continuar!

Slide 11

Instalação do XBMCMove

Descompacte o arquivo XBMCMove.tar.bz2;

Entre na pasta referente a arquitetura da sua máquina (32 ou 64 Bits).

Abra um terminal e vá até a pasta onde esta o arquivo "install.sh" e execute o comando:

sudo ./install.sh

Slide 12

Responda as perguntas 10 e 11 antes de continuar!

Avise o Avaliador sobre o fim da Instalação!

Slide 13

Execução do XBMCMove

Basta executar o atalho em Aplicativos, Multimídia, XBMCMove

Vejo o exemplo a seguir:

Imagem D.3

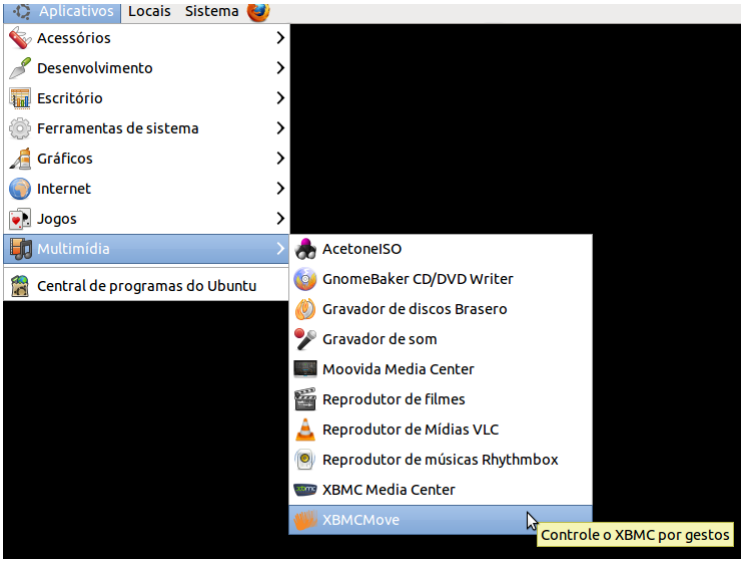


Figura D.3: Procedimento para abrir o XBMCMove

Slide 14

Responda a pergunta 12 antes de continuar!

Slide 15

Gestos Reconhecidos pelo XBMCMove

Slide 16

Gestos Direcionais

Gestos Direcionais são os gestos utilizados nos menus do XBMC;

Eles movem para Direita, Esquerda, Para cima e para baixo.

Slide 17

Gestos Direcionais

Para realizar os gestos Direcionais a mão deve estar, pelo menos acima da barriga e a pouco mais de 50cm á frente do usuário.

Ela deve ir na direção desejada e depois deve retornar ao ponto de origem.

A imagem a seguir mostra como o gesto direcional deve ser feito:

Imagem D.4

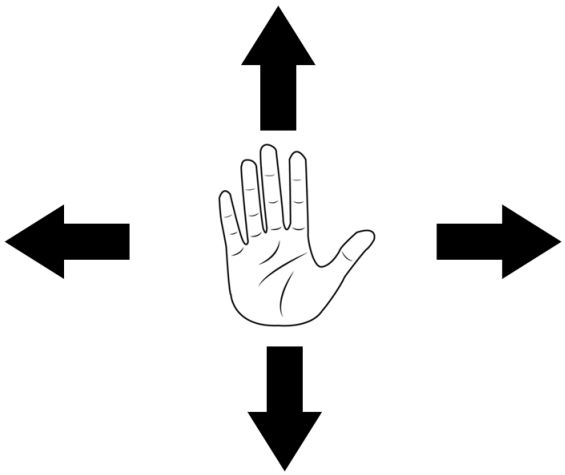


Figura D.4: Gestos Direcionais

Slide 18

Gesto de Clique

O gesto de Clique é usado para:

Selecionar itens do menu; Retornar a reproduzir um vídeo pausado.

Para executar o gesto de Clique basta movimentar a mão para frente em poucos centímetros e depois retornar ao ponto de origem. (Tapinha)

Slide 19

Gesto de Tchau

Gesto de tchau é usado para:

Pausar um vídeo em reprodução; Voltar para o menu quando vídeo estiver pausado; Voltar ao menu principal quando navegando pelo menu.

Para executar o gesto de Tchau basta acenar por alguns segundos para o dispositivo. Fazendo movimentos rápidos de uma lado para o outro.

Slide 20

Exercício 1

Abra a XBMCMove;

Navegue pelo menu usando os gestos direcionais e entre no menu “Músicas” com o gesto de clique;

Saia do menu “Músicas” com o gesto de Tchau;

Entre no menu Vídeos;

Entre no menu Arquivos;

Entre no menu Vídeos;

Responda as perguntas e 13 a 15.

Slide 21

Gesto Aumentar/Diminuir Volume

Com as duas mão levantadas (acima da barriga), ao afastar as mãos o volume ira aumentar e ao aproximá-las o volume ira diminuir.

Imagem D.5

(a) Aumentar Volume (b) Diminuir Volume

Figura D.5: Gestos para Aumentar e Diminuir o Volume

Slide 22

Exercício 2

Dentro do menu Vídeos, Arquivos, Vídeos execute o vídeo que esta na pasta, com o gesto de clique em cima dele;

Diminua completamente o volume e depois aumente-o completamente;

Pause o Vídeo com o gesto de Tchau;

Retorne a reprodução com o gesto de Clique;

Pare de reproduzir o vídeo realizando o gesto de Tchau 2 vezes;

Responda as perguntas 19 a 24.

Slide 23

Gesto Circulo

Avançar ou retroceder o vídeo.

Deve fazer círculos na direção que deseja avançar/retroceder o vídeo.

Slide 24

Gesto Fechar

Para fechar o XBMC a qualquer momento basta realizar o gesto de Fechar;

Para realizar o gesto fechar, levante suas mãos e depois faça um "X" com os braços, cruzando-os com as mãos na altura do ombro.

Slide 25

Exercício 3

Dentro do menu Vídeos ¿ Arquivos ¿ Vídeos execute o vídeo que esta na pasta, com o gesto de clique em cima dele;

Avance o vídeo com o gesto de circulo;

Retroceda o vídeo com o gesto de circulo;

Feche o XBMC com o gesto Fechar.

Responda as perguntas de 16 a 21.

Slide 26

FIM!

Obrigado pela participação!

Referências Bibliográficas

APPLE. *Overview*. 2011. Disponível em: <http://www.apple.com/appletv/>.

ASUS. *Xtion PRO*. 2011. Disponível em: <http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/#overview>.

BOXEE. *About Boxee, Inc.* 2011. Disponível em: <http://www.boxee.tv/press>.

CARMODY, T. *How Motion Detection Works in Xbox Kinect*. 2010. Disponível em: <http://www.wired.com/gadgetlab/2010/11/tonights-release-xbox-kinect-how-does-it-work/all/1>.

DELL. *Mini PC Inspiron Zino HD*. 2011. Disponível em: <http://www.dell.com/br/p/inspiron-zino-hd-410/pd?ref=gzilla>.

HARDMOB. *O Kinect e um novo paradigma para os consoles*. 2011. Disponível em: <http://www.hardmob.com.br/threads/435639-O-Kinect-e-um-novo-paradigma-para-os-consoles>.

KARASINSKI, E. *Moovida Media Player*. 2010. Disponível em: <http://www.baixaki.com.br/download/moovida-media-player.htm>.

KIRN, P. *Kinect Hacking and Art Round Table: Why it Matters, What You Need to Know*. 2010. Disponível em: <http://createdigitalmotion.com/2010/11/kinect-hacking-and-art-round-table-why-it-matters-what-you-need-to-know/>.

MICROSOFT. *Posicionamento do Sensor Kinect*. 2010. Disponível em: <http://support.xbox.com/pt-BR/kinect/setup-and-playspace/sensor-placement>.

MICROSOFT. *The Kinect Effect*. 2011. Disponível em: <http://www.xbox.com/en-US/Kinect/Kinect-Effect>.

OPENNI. *OpenNI User Guide*. [S.l.], 2011. Disponível em: <http://openni.org/Documentation.aspx>.

PINOTTI, G. *Evoluce libera interface baseada no Kinect para Windows 7, você não precisa mais tocar no mouse*. 2011. Disponível em: <http://www.conteudonerd.com/3793/evoluce-libera-interface-baseada-no-kinect-para-windows-7-voce-nao-precisa-mais-tocar-no-mouse>.

PINOYTUTORIAL. *Globalscale D2 Plug: Review, Specs and Price for Mini-PC*. 2011. Disponível em: <http://pinoytutorial.com/techtorial/globalscale-d2-plug-review-specs-and-price-for-mini-pc/>.

REALGAGE. *Dell Inspiron Zino HD Mini PC Hits US*. 2009. Disponível em: <http://www.realgage.com/2009/11/13/dell-inspiron-zino-hd-mini-pc-hits-us/>.

RIGUES, R. *Media center: LinuxMCE serve para a casa do futuro*. 2007. Disponível em: <http://tecnologia.uol.com.br/ultnot/2007/11/07/ult4213u191.jhtm>.

SOLIDRUN. *CuBox Developer Platform*. 2011. Disponível em: <http://www.solid-run.com/products/cubox>.

TANZ, J. *Kinect Hackers Are Changing the Future of Robotics*. 2010. Disponível em: <http://www.wired.com/magazine/2011/06/mf_kinect/all/1>.

WIKIPEDIA. *XBMC*. 2011. Disponível em: <http://en.wikipedia.org/wiki/XBMC>.