Software de Armazenamento e comparação de Poses Capturadas pelo Microsoft Kinect

Natália Ellery Ribeiro Couto, Benjamin Grando Moreira

Universidade do Vale do Itajaí (UNIVALI) – Itajaí – SC – Brasil

nataliaellery@gmail.com, benjamin@univali.br

Abstract. *This paper is about the development of a software capable of communicating with the Microsoft Kinect, capturing poses, storing them and using them for comparison with poses performed at a later time. With the appearance of the Microsoft Kinect comes the opportunity to create applications that involve physical activity, but there is a dependency on the software developer. The software developed in this work allows the storage of a pose, in other words, the user can record postures and provide for himself or another user to perform the poses at a later time, this way the user can create a new pose abstracting all process of software development.*

Resumo. Esse trabalho aborda o desenvolvimento de um software capaz de se comunicar com o Microsoft Kinect, capturando poses, armazenando-as e usando-as para comparação com poses executadas em um momento posterior. Com o surgimento do Microsoft Kinect aparece a oportunidade de criar aplicações que envolvem atividades físicas, mas existe uma dependência com o desenvolvedor de software. O software desenvolvido neste trabalho permite o armazenamento de uma pose, ou seja, o usuário pode gravar posturas e disponibilizar para que outro usuário execute as poses em um momento posterior, desta maneira o usuário poderá criar uma pose nova abstraindo todo o processo de desenvolvimento de software.

1. Introdução

Com a criação do computador foi necessária a criação de uma interface para que humanos pudessem se comunicar com a máquina de alguma forma. Essa interface que realiza a comunicação sempre busca a forma mais natural de interação [Lima 2006].

Quando a tecnologia começou a ser utilizada para entretenimento, foram criados consoles de jogos que necessitavam de uma interface diferenciada para que o jogador controlasse o jogo, sendo essa interface o *joystick*. A forma convencional de interação favorece uma postura ruim e sedentária, onde não se realiza atividade física durante o jogo e provocando esforço repetitivo que, após um período de tempo, torna o uso desconfortável [Abraham e Nath 2004].

Para tornar a interação com o jogo mais dinâmica, foram criados diferentes tipos de *joysticks* que imitavam a realidade, como volante, armas, tapetes de dança. Com a evolução

da tecnologia foi possível criar detectores de movimento como o Wii remote, que funciona através de um acelerômetro de três eixos com sistemas microeletromecânicos [Turner 2007].

Mesmo com detectores de movimento, ainda era necessária a presença de um *joystick* até que a tecnologia de interação evoluiu para a detecção através da webcam, a qual dispensa qualquer periférico, fazendo com que a interação seja natural ao ser humano e atraindo novos públicos para a tecnologia de entretenimento. Utilizando esse conceito surgiu a Touchless SDK, que captura a posição e o movimento de um objeto através de sua cor [Zeid et al. 2011]. Apesar de uma tecnologia inovadora, a detecção através da cor possui algumas irregularidades, pois dependendo da distância, luz e qualidade da câmera, a cor de um mesmo objeto pode variar muito, fazendo com que a captura seja falha em alguns momentos [Wu e Yu 2011].

Em 2010 foi lançado o Microsoft Kinect [Microsoft 2012], que possui duas câmeras e um projetor infravermelho baseado em laser [Konolige e Mihelich 2010]. Através desta nova forma de interação de detecção de movimentos, este trabalho se trata do desenvolvimento de um software que possa adquirir e armazenar poses estáticas representadas por pontos do esqueleto de uma pessoa através do Kinect, para que um jogo possa dinamicamente utilizar novas poses que não foram planejadas durante a criação deste. Este armazenamento permite que o jogador adicione novas poses que posteriormente serão comparadas com as poses de outro jogador através dos pontos de esqueleto adquiridos.

Durante a comparação de poses é necessário fornecer ao jogador um *feedback* relacionado a pose que ele está fazendo, podendo afirmar se ela é similar a uma pose anteriormente armazenada. Vários jogos utilizam diferentes tipos de *feedback* para que o jogador perceba se está realizando a pose corretamente. Este trabalho pretende também propõem algumas formas de *feedback* para comparação de poses, propondo tanto um modelo para a comparação quanto para o *feedback* das poses.

2.Fundamentação teórica

2.1.Microsoft Kinect

Segundo [Paula 2011], o Kinect é formado por um projetor de luz infravermelha, que não pode ser visto ao olho humano, uma câmera infravermelha, uma câmera RGB comum, um conjunto de microfones e um motor. O desenvolvedor pode utilizar os seguintes dados retirados do dispositivo:

- Fluxo de cores imagem: onde cada pixel representa uma cor. A resolução é de 640x480 pixels com 30 FPS, ou 1280x1024 com 15 FPS;
- Fluxo de profundidade: cada pixel indica a distância entre o ponto em questão e o aparelho. É possível perceber objetos posicionados de 1,2 a 3,5 metros à frente do aparelho. Além dos dados de profundidade é possível verificar com exatidão se o pixel faz parte do corpo de um ser humano; e
- Fluxo de áudio: Através do conjunto de quatro microfones, e um sistema de anulação de ruído e eco, o Kinect pode gravar o áudio e reconhecer comandos de voz em inglês.

O Kinect não se limita em apenas informar a distância de um objeto em relação ao sensor, ele obtém informação do esqueleto de até dois jogadores simultâneos, disponibilizando a posição x, y e z de cada articulação.

2.2. SDKs para o desenvolvimento com Kinect

Segundo [Paula 2011], antes da Microsoft disponibilizar sua própria ferramenta foi desenvolvido um aplicativo que criou a interação entre o Kinect e um computador com o uso de um analisador de USB, criando-se assim uma comunidade chamada OpenKinect [OpenKinect 2011] que disponibilizou um conjunto de drivers livres para o Kinect.

Em 2011, a Microsoft disponibilizou sua própria ferramenta gratuita de desenvolvimento para o Kinect, e sem licença comercial, ou seja, limitado a aplicações não comerciais [Microsoft Research 2011].

A OpenNI e a OpenKinect são abordagens abertas e gratuitas, não possuem restrição de uso ou licença com restrições muito amplas, já o Kinect for Windows SDK é gratuito, mas é limitado a aplicações não comerciais. As principais funcionalidades de cada recurso são:

- OpenNI: Profundidade e cor; Manipulação de esqueleto; Rastreamento de mãos; e Geradores de áudio;
- OpenKinect: Rastreamento das mãos; Rastreamento do esqueleto; Processamento de dados de profundidade; Reconstrução 3D; e
- Kinect for Windows: Profundidade e cor; Manipulação de esqueleto; e Reconhecimento de comandos de voz.

Como o trabalho utilizou comandos de voz para acesso a opções da interface, o framework mais adequado para a implementação foi o Kinect for Windows.

2.3. Técnicas de comparação de poses

Podem ser usadas diferentes técnicas para reconhecer poses ou gestos que são capturados pelo Kinect, podendo ser utilizados dados de profundidade, dados de pontos de esqueletos e até mesmo os pixels referentes ao usuário. Duas técnicas destacam-se no objetivo de reconhecimento de poses estáticas, sendo elas:

- Técnica Algorítmica: segundo Sá [2011], a técnica algorítmica realiza a identificação através de um algoritmo que especifica exatamente como a pose é realizada. Um exemplo é reconhecer se um braço levantado, onde o algoritmo deve verificar se a mão está levantada acima da cabeça e se o cotovelo está acima do ombro. A técnica algorítmica só funciona para poses já definidas anteriormente, mas ela possui as vantagens de fácil entendimento, implementação e teste; e

- Comparação de ângulos: segundo [Silveira 2011], a técnica de comparação de ângulos utiliza os pontos de esqueletos. Com os pontos x e y do esqueleto calcula-se o ângulo formado pelo segmento entre dois pontos, e com esse valor, executa-se a ação definida ou não.

3. Trabalhos Similares

Trabalhos sobre a temática apresentada nesse projeto estão surgindo devido ao reduzido preço do dispositivo e a disponibilidade de vários frameworks.

3.1 Kinect SDK Dynamic Time Warping Gesture Recognition

O Kinect SDK Dynamic Time Warping Gesture Recognition é um projeto que utiliza a SDK oficial da Microsoft para gravar gestos realizados por um jogador, armazená-los em um documento de texto e interpretar as ações do jogador de acordo com os gestos armazenados.

Para realizar a gravação de um gesto o jogador deve escolher seu nome em uma lista, e executar o gesto em 32 imagens. O documento de texto gerado possui 12 pontos X e Y do modelo do jogador em 32 imagens. Os pontos utilizados são apenas da parte superior do corpo: cabeça, centro dos ombros, ombros, coluna, centro do quadril, cotovelos, pulsos e mãos.

O reconhecimento dos gestos armazenados é realizado por um algoritmo baseado em vetores e vizinhos próximos. Ou seja, não importa a velocidade que o jogador irá realizar o gesto [Kinect SDK Dynamic Time Warping Gesture Recognition 2011]. O trabalho foi bem sucedido para seu propósito, que é limitado ao 2D.

3.2 Human Detection Using Depth Information by Kinect

O objetivo deste projeto é criar uma abordagem baseada em um modelo que detecta os seres humanos através de um contorno pré-estabelecido 2D e um preenchimento 3D [Xia *et al.* 2012].

Os resultados experimentais mostram que o algoritmo pode efetivamente detectar as pessoas em todas as poses e aparências através dos dados de profundidade, fornecendo também uma estimativa precisa do contorno do corpo inteiro da pessoa.

A limitação é que o algoritmo tem uma alta dependência da precisão de detecção da cabeça, o que implica em se a cabeça não estiver aparecendo ou a pessoa estiver usando um chapéu com um formato estranho, ela provavelmente não será detectada.

4. Projeto

O software desenvolvido nesse trabalho é capaz de armazenar uma postura estática gerada por uma pessoa, capturada através do Kinect. Os pontos do esqueleto capturados são transformados em ângulos representando articulações do corpo. São criados três ângulos para cada uma destas articulações, referentes a visão de lado, cima e frente, conforme ilustra a Figura 1. Como serão utilizados apenas dois eixos para se calcular o ângulo, o terceiro eixo é descartado utilizando a projeção ortográfica.

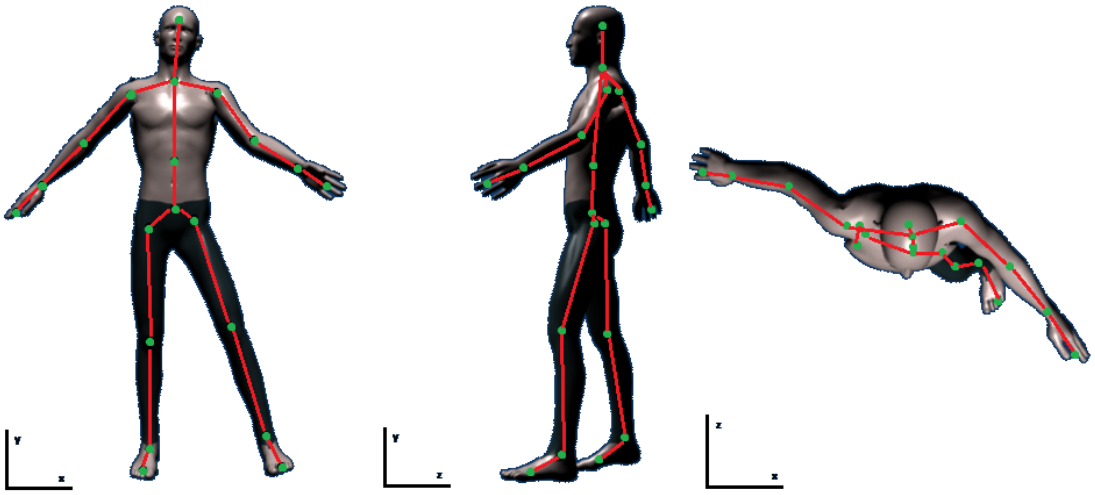

Figura 1: Pontos na visão X e Y, Z e Y, e X e Z

Para calcular o ângulo é necessário obter o ponto de esqueleto anterior e o próximo, a fim de formar um triângulo, conforme a Figura 2.

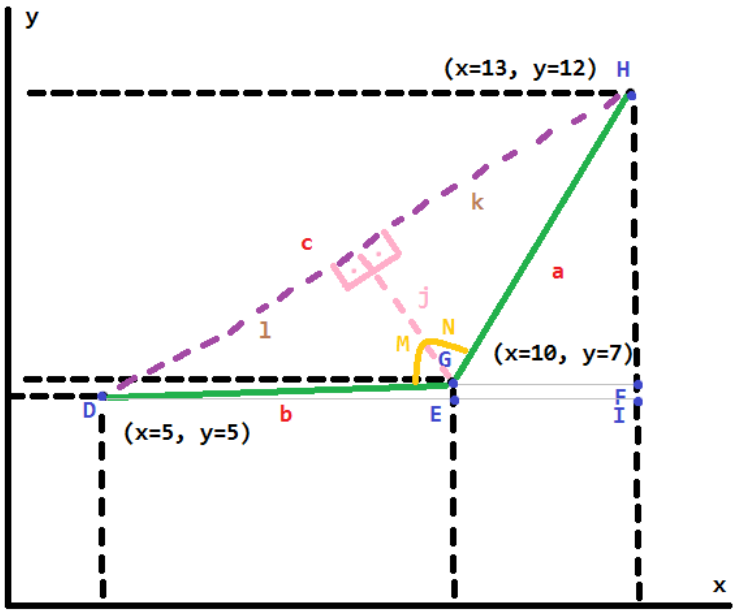

Figura 2: Pontos utilizados para descobrir um ângulo

Depois de identificados os pontos, é necessário calcular o valor das retas formadas entre eles usando o teorema de Pitágoras, a Figura 3 demonstra o cálculo resultante dos pontos da Figura 2.

$b^2 = (DE)^2 + (GE)^2$
$b^2 = (10-5)^2 + (7-5)^2$
$b^2 = 5^2 + 2^2$
$b = \sqrt{29}$
$b = 5,38$

$a^2 = (GF)^2 + (HF)^2$
$a^2 = (13-10)^2 + (12-7)^2$
$a^2 = 3^2 + 5^2$
$a = \sqrt{34}$
$a = 5,83$

$c^2 = (DI)^2 + (HI)^2$
$c^2 = (13-5)^2 + (12-5)^2$
$c^2 = 8^2 + 7^2$
$c = \sqrt{113}$
$c = 10,63$

Figura 3: Teorema de Pitágoras aplicado aos pontos da Figura 2

Quando o valor das retas é encontrado, o triângulo deve ser dividido em dois, para obter dois triângulos retângulos, e em seguida é aplicada uma fórmula para descobrir o valor da altura e das projeções dos catetos, conforme mostra a Figura 4.

$b^2 = c^2 + a^2 - 2ck$
$5,38^2 = 10.63^2 + 5,83^2 - 2*10,63*k$
$-k = (28,94 - 112,99 - 33,98)/21,26$
$-k = -118,03/21,26$
$k = 5,55$

$l = c - k$
$l = 10,63 - 5,55$
$l = 5,08$

$a^2 = j^2 + k^2$
$5,83^2 = j^2 + 5,55^2$
$j = \sqrt{33,98 - 30,80}$
$j = 1,78$

Figura 4: Fórmula para descobrir a altura e as projeções dos catetos

Com os valores da altura e projeções dos catetos do triângulo, é possível descobrir o ângulo que agora está dividido em dois, será necessário calcular cada ângulo separadamente para depois somá-los e obter o resultado. O cálculo do ângulo pode ser obtido através do seno, cosseno ou tangente. Como todos os lados do triângulo já foram encontrados, qualquer um dos três cálculos pode ser usado, então, sem critério de escolha, será usado o seno, conforme mostra a Figura 5.

```
senN = k/a          senM = l/b
senN = 5,55/5,83    senM = 5,08/5,38
senN = 0,9519       senM = 0,9442
N = 72°             M = 71°

        Ângulo 1 = 72° + 71°
        Ângulo 1 = 143°
```

Figura 5: Cálculo do seno dos ângulos

Com o seno do ângulo identificado, ainda é necessário localizar o valor correspondente do seno em uma tabela para identificar o valor do ângulo. Em alguns casos, o ângulo descoberto é replementar do ângulo representante da articulação, por isso deverá ser subtraído do valor 360.

Para armazenar os dados é utilizado um arquivo XML. Cada pose tem a informação de 15 articulações, cada articulação tem a informação de três ângulos, e cada um destes ângulos tem um limite inferior e superior de ângulo, conforme Figura 6.

```
<articulacao nome="pescoco">
  <posicao x="0,002788604" y="0,3026539" z="1,719159" />
  <angulo tipo="x" valor="134">
    <limite superior="60" inferior="60" />
  </angulo>
  <angulo tipo="y" valor="174">
    <limite superior="60" inferior="60" />
  </angulo>
  <angulo tipo="z" valor="86">
    <limite superior="60" inferior="60" />
  </angulo>
  <posicaoTela x="320,8924" y="167,3631" />
</articulacao>
```

Figura 6: Parte do XML representando uma articulação

Após armazenada a postura já pode ser comparada com outras execuções, o software permite que a comparação seja feita em tempo real e ainda utiliza 5 métodos para auxiliar o usuário em realizar a pose:

- O primeiro método de *feedback* exibe um vulto da pose a ser executada, para que o usuário tenha uma referência. Esta referência é criada através dos pontos x,y e z salvos no arquivo XML, e posicionada atrás da representação do esqueleto do usuário que está realizando a pose;
- O segundo método de *feedback* apresenta setas que iniciam na articulação da representação gráfica do esqueleto do usuário que está realizando a pose, e apontam para a direção onde esta articulação deveria estar. A seta desaparece quando o ângulo da articulação está dentro do limite superior e inferior salvos no XML;
- O terceiro método de *feedback* faz com que o esqueleto seja preenchido com cores, estas cores podem ser verde, amarelo ou vermelho. Quando o ângulo está dentro do

limite de aceitação o trecho que inicia na junta anterior do ângulo e termina na junta posterior é preenchido de verde. Se este ângulo estiver apenas 30° de distância do limite superior ou inferior, o mesmo trecho é preenchido de amarelo. Caso o ângulo esteja a mais de 30° de distância de qualquer um dos limites, o trecho é preenchido de vermelho;

- A quarta opção faz com que um marcador vermelho ou amarelo apareça na articulação que está fora do limite angular armazenado. A cor amarela e vermelha seguem as mesmas regras da terceira opção; e
- O quinto método se trata do *feedback* textual. São instruções de como o usuário deve se posicionar para realizar a pose corretamente. As instruções aparecem em forma escrita e tratam uma articulação por vez, para evitar poluição visual.

A Tabela 1 mostra quadro dos métodos de *feedback* implementados nesse trabalho: a primeira célula (primeira linha, e primeira coluna) ilustra o primeiro método; na primeira linha, segunda coluna está o segundo método; o terceiro método é ilustrado na segunda linha, primeira coluna e na célula restante é representado o quarto método. O quinto método simplesmente mostra o esqueleto e um texto logo abaixo indicando o que deve ser corrigido na pose.

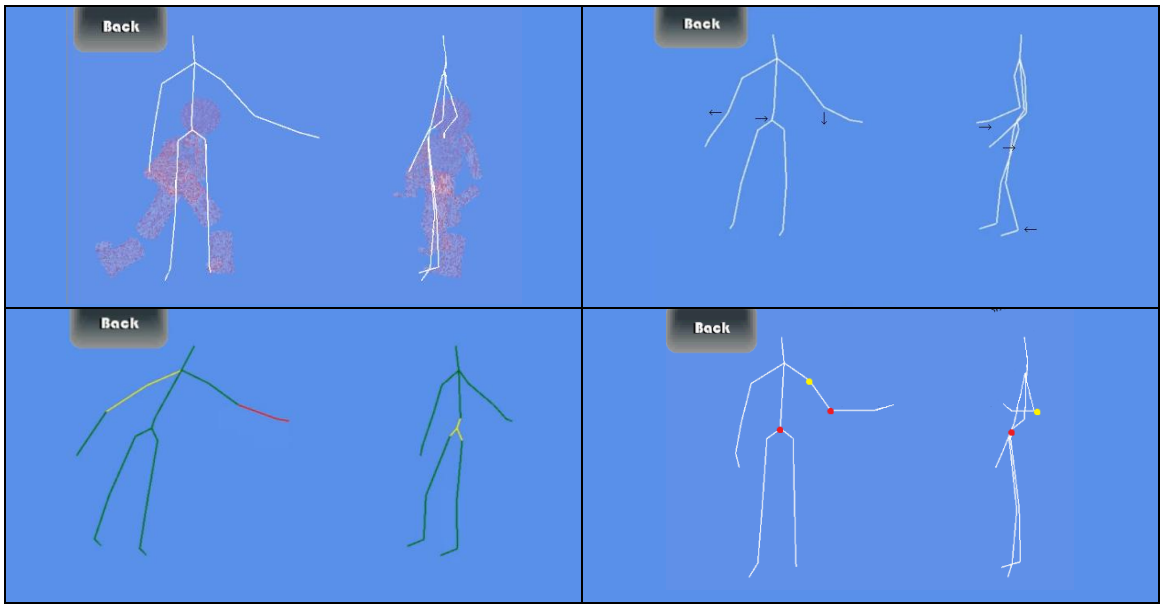

Tabela 1. Alternativas de *feedback*: (1) vulto; (2) setas indicativas; (3) esqueleto preenchido com cores; e (4) *feedback* nas articulações.

5. Conclusões

Com os resultados dos testes, é possível concluir que o cálculo angular permite que uma pose seja comparada com outras, não importando a estrutura corporal do usuário que a executou, e é essencial definir quais os ângulos serão replementares, caso contrário as articulações estarão limitadas a 180°, podendo também ocorrer ângulos iguais em poses diferentes.

Os comandos de voz pré-definidos são muito sensíveis e muitas vezes confundem uma palavra com outra, mesmo palavras cujas pronúncias são tão diferentes, por este motivo foi necessário criar meios alternativos para todas ações que poderiam ser acessados por meio de comandos de voz.

Os *feedbacks* testados em alguns casos funcionam melhor em conjunto, mas podendo também causar uma perturbadora poluição visual quando usados todos simultaneamente. Foi possível perceber que algumas pessoas se identificam mais com *feedbacks* diferentes de outras pessoas. Já que alguns preferem resolver um problema de cada vez, como no caso do *feedback* textual, ou preferem comparar sua representação gráfica com a representação salva.

O software desenvolvido neste projeto pode auxiliar na construção de jogos e aplicativos para o Kinect, tornando mais fácil descobrimento do valor do ângulo das articulações que compõem uma pose. O artigo também apresenta uma alternativa detalhada de como calcular o ângulo, mas essa não é a única forma.

O software atualmente armazena e reconhece apenas poses estáticas, para trabalhos futuros podem ser elaborados métodos de reconhecimento de movimentos, e adicionar opções que permitam armazenar os dados relacionados que são necessários para o reconhecimento do movimento salvo.

Referências

Abraham, Ajay; Nath. Nitendra. Computer Vision For Computer Games 2004 Disponível em: <http://www.ces.clemson.edu/~stb/ece847/fall2004/projects/proj19.doc>. Acesso em: 12 mar. 2012.

Giles, J. Inside the race to hack the Kinect 2010. Disponível em: <http://www.newscientist.com/article/dn19762-inside-the-race-to-hack-the-kinect.html>. Acesso em: 11 abr. 2012.

Kinect SDK Dynamic Time Warping Gesture Recognition 2011. Disponível em: <http://kinectdtw.codeplex.com/>. Acesso em: 21 maio 2012.

Konolige, Kurt; Mihelich, Patrick. Technical description of Kinect calibration 2010. Disponível em: <http://www.ros.org/wiki/kinect_calibration/technical>Acesso em 08 mar. 2012.

Landim, Wikerson. Conheça o Move, o novo controle do Playstation 3 2010. Disponível em: < http://www.tecmundo.com.br/3873-conheca-o-move-o-novo-controle-do-playstation-3.htm> Acesso em 11 mar. 2012.

Lima, Hamilton. IHC: Interação Humano Computador 2006. Disponível em: <http://www.athanazio.com/downloads/ihc/notas-de-aula-athanazio-ihc.pdf >. Acesso em: 07 mar. 2012.

Microsoft. Kinect for Windows 2012. Disponível em:<http://www.microsoft.com/en-us/kinectforwindows/> Acesso em 19 mar. 2012.

Microsoft Research Programming Guide: Getting Started with the Kinect for Windows SDK Beta from Microsoft Research 2011. Disponível em: <http://202.120.53.119/wordpress/wp-content/uploads/2011/06/ProgrammingGuide_KinectSDK.pdf >. Acesso em: 21 maio 2012.

OpenKinect 2011. Disponível em: <http://openkinect.org/>. Acesso em: 21 maio 2012.

Paula, Bruno Campagnolo. Adaptando e desenvolvendo jogos para uso com o Microsoft Kinect, 2011, Salvador. Anais do X Simpósio Brasileiro de Games e Entretenimento Digital. Salvador: Sociedade Brasileira de Computação, 2011.

Sá, Jobert Gomes Prado. Construindo uma DSL para reconhecimento de gestos utilizando o Kinect 2011. Disponível em: <http://www.cin.ufpe.br/~tg/2011-2/jgps.pdf>. Acesso em: 21 maio 2012.

Silveira, Marcus Almeida. Técnica de Navegação em Documentos Utilizando Microsoft Kinect 2012. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/36884/000819161.pdf?sequence=1>. Acesso em: 22 maio 2012.

Turner, Daniel. Hack: The Nintendo Wii 2007. Disponível em: <http://www.technologyreview.com/read_article.aspx?id=18899&ch=infotech> Acesso em 11 mar. 2012

Wu, Yang; Yu, Lu. Touchless Writer: Object Tracking & Neural Network Recognition 2011. Disponível em: <http://www.ces.clemson.edu/~stb/ece847/projects/Touchless_Writer.pdf> Acesso em 07 mar. 2012.

Xia, Lu; Chen, Chia-Chih; Aggarwal, J. K. Human Detection Using Depth Information by Kinect 2012. Disponível em: <http://www.nattee.net/sites/default/files/Human%20Detection%20Using%20Depth%20Information%20by%20Kinect.pdf>. Acesso em: 21 maio 2012.

Zeid, Amir; et al. Towards Economical E-Learning Educational Environments for Physically Challenged Student 2011. Disponível em: <http://cdn.intechopen.com/pdfs/28696/InTech-Towards_economical_e_learning_educational_environments_for_physically_challenged_students.pdf> Acesso em 05 mar. 2012.