Aumento na sensação de tridimensionalidade em mundos virtuais através de janelas virtuais

Leonardo Ronald Perin Rauta[1], Eros Comunello[1], Anita Maria da Rocha Fernandes[1]

[1]Universidade do Vale do Itajaí

São José – SC – Brasil

{leonardorauta, eros.com, anita.fernandes}@univali.br

Abstract. *This paper presents an idea for using virtual windows in order to increase the sense of three-dimensionality in virtual worlds. To do this were used the head-tracking technique to identify the position and movement of the user and move the virtual objects to respond to the change of position of this user. By identifying the movements, the virtual world can change the point of view being presented on the display. The application developed shows the computational efficiency of this type of virtual world control, causing the feeling of three-dimensionality of the virtual objects were increased.*

Resumo. *Este artigo apresenta uma ideia para utilização de janelas virtuais com a finalidade de aumentar a sensação de tridimensionalidade em mundos virtuais. Para isso foi utilizada a técnica de head-tracking para identificar a posição e a movimentação do observador, de modo que os elementos virtuais respondessem à alteração da posição do observador. Através da identificação dos movimentos, é possível alterar a posição do campo de visão do mundo virtual que está sendo apresentado no display. A aplicação de exemplo desenvolvida mostrou a eficácia computacional deste tipo de movimentação, fazendo com que a sensação de tridimensionalidade dos objetos virtuais fosse aumentada.*

1. Introdução

A ideia da utilização de *head-tracking* como uma forma de interação do usuário é uma ideia antiga. Algumas publicações, como a de Yang *et al.* (1998), apresentam técnicas de monitoramento visual para interação humano-computador. Já outras publicações, como a de Toyama (1998), apresentam a aplicação e utilização de *face-tracking* para movimentar o ponteiro do *mouse* através da movimentação da face do usuário.

Algumas propostas mais recentes como a de Lee (2007) e Francone e Nigay (2011), aplicam conceitos de *tracking* para criar janelas virtuais, as quais simulam um ambiente tridimensional por meio de um ambiente bidimensional.

Na proposta de Lee (2007) foi utilizado um Wii-mote para fazer o *tracking* da movimentação do usuário. Já na proposta de Francone e Nigay (2011), foi feita utilizando a câmera de um dispositivo móvel.

Atualmente a maioria dos jogos em primeira pessoa utiliza os movimentos do mouse e teclado para então movimentar, tanto o avatar, quanto seu campo de visão. Por exemplo, para visualizar o que está ao lado do avatar, é necessário movimentar o mouse

para o lado, ou seja, deslocando a posição da arma para a mesma direção. Deste modo, o avatar seria rotacionado para então apresentar aquele campo de visão.

Com a utilização de técnicas de *head-tracking* o usuário pode ser imerso no mundo do game e através da movimentação da sua cabeça, ser possível visualizar elementos que estão ao redor de seu avatar, sem precisar rotacioná-lo, apenas simular o movimento do pescoço do avatar.

Preocupados em fazer a imersão do jogador no mundo virtual, indústrias de consoles e games, como Nintendo, Microsoft e Sony, por exemplo, vêm investindo em tecnologias e recursos para rastrear a movimentação do jogador e, com isso aumentar a imersão e a interatividade do jogo.

Devido à tecnologia envolvida para aumentar a interatividade e a imersão do jogador, os jogos, os consoles e os equipamentos possuem um preço que muitas vezes não é acessível a todos os consumidores de jogos.

Com a utilização desses equipamentos de *tracking* do jogador, é possível identificar sua movimentação e através dela, identificar a posição do jogador e representa-la no display de forma virtual. Isso acaba criando uma espécie de janela virtual, que por meio de uma movimentação no mundo real, os objetos virtuais também são deslocados.

Pensando em um baixo custo para o consumidor, e também em aumentar a sensação de tridimensionalidade em mundos virtuais, esse trabalho apresenta um exemplo da aplicação de *head-tracking* de baixo custo financeiro para alterar o campo de visão de um jogador em primeira pessoa através de reconhecimento facial, utilizando técnicas de realidade virtual e visão computacional.

Este trabalho está organizado da seguinte forma: a primeira seção apresenta uma introdução do que é *head-tracking* e sua aplicação; na segunda seção é apresentada uma técnica utilizada em games para alterar o campo de visão do jogador chamada *Frustum Culling*; a terceira seção apresenta a aplicação de interação com *head-tracking* desenvolvida, e por fim, a última seção apresenta as conclusões do trabalho.

2. Head-Tracking e Face-Tracking

Desde a década de 90 vêm se procurando uma maneira genérica de fazer o reconhecimento facial, ou seja, reconhecer qualquer rosto. Sung e Poggio (1998) e Papageorgiou e Poggio (1998) estudaram diferentes técnicas para detecção e classificação de faces, e as propostas de seus trabalhos consistiram no desenvolvimento de *frameworks* para reconhecimento e detecção genérica de faces. Porém, as respostas eram lentas para o reconhecimento em tempo-real, pois rodavam apenas 15 *frames* por segundo. Isso tornava inviável a utilização de *head-tracking* em jogos, pois atualmente um jogo é considerado estar rodando em tempo real quando a sua taxa de atualização está no mínimo a 20 fps (*frames per second*).

Com base nesses trabalhos, Viola e Jones (2001) propuseram um *framework* com um processo de classificação multi-estágios, o que acabou reduzindo consideravelmente o tempo de processamento, permitindo uma taxa de 30 frames por segundo e, tornando assim um reconhecimento em tempo real, além de conseguir melhorar também a acurácia na classificação da face. Este *framework* ficou conhecido como *Viola-Jones Detector*.

O *framework* desenvolvido foi treinado com algumas faces, as quais atribuíam as características necessárias para que o *framework* pudesse reconhecer o que era uma face. Desse modo, depois dessa etapa de treinamento, o *framework* se torna capaz de reconhecer uma grande gama de faces, mesmo estas sendo diferentes das utilizadas no treinamento.

A Figura 1 apresenta um exemplo do reconhecimento de faces utilizando o *Viola-Jones Detector*. As faces utilizadas nessa figura não foram as faces utilizadas para o treinamento do *framework* e mesmo assim ele foi capaz de reconhecer todas essas faces. Além de detectar todas as faces presentes na imagem, o algoritmo detectou também uma bola de futebol como sendo uma face, indicando assim que o algoritmo é passível de falsas detecções.

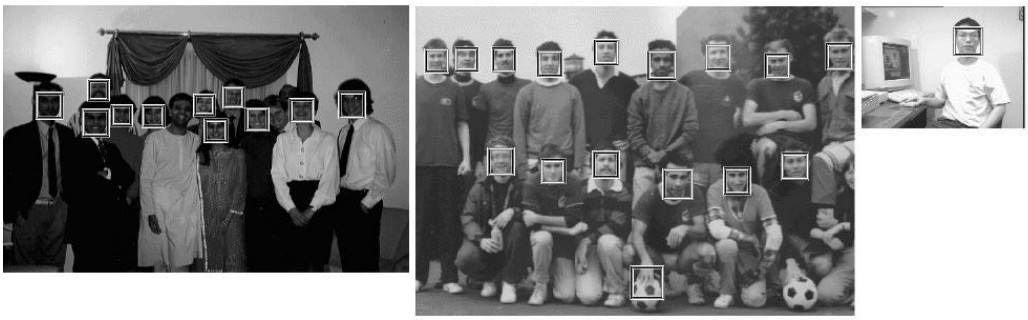

Figura 1. Exemplo do funcionamento do framework Viola-Jones Detector Adaptado de Viola e Jones, 2001

Uma característica do *Viola-Jones Detector* era reconhecer apenas objetos que estivessem na posição horizontal ou vertical. Com base na deficiência em reconhecer faces inclinadas, Lienhart e Maydt (2002) propuseram uma melhoria no *framework,* inserindo um método de classificação de objetos em diagonais, o que acabou melhorando a acurácia na detecção desses objetos.

Diferentes bibliotecas de processamento de imagens digitais e de visão computacional utilizam diferentes técnicas e *frameworks* para realizar o reconhecimento facial. Como este trabalho utilizou a biblioteca OpenCV[1], ele utiliza o algoritmo proposto por Viola e Jones (2001) com a melhoria realizada por Lienhart e Maydt (2002) e a utilização da transformada de Haar, a qual é utilizada para reduzir ainda mais a quantidade de falsas detecções e melhorar a acurácia da técnica de reconhecimento facial [Bradski e Kaehler, 2008].

Os algoritmos, técnicas e *frameworks* listados são apenas a detecção de faces em imagens estáticas. Quando são utilizados vídeos e existe a necessidade de rastrear um objeto em uma cena, é necessário utilizar técnicas de *tracking,* as quais apenas traçam os caminhos feitos pelo objeto de interesse na cena tratada [Gonzalez e Woods, 2008].

Assim, é necessário identificar a cada nova captura de vídeo, a nova posição da face que foi detectada anteriormente. Devido ao fato de estar utilizando o rastreamento de faces, esta técnica é chamada de *head-tracking* ou *face-tracking.*

[1] OpenCV - Open Source Computer Vision - http://opencv.org/

3. Frustum Culling

Determinar quais objetos estão visíveis dentro de uma cena tridimensional é uma área bastante estudada dentro da computação gráfica. Na área de jogos isso é bastante importante, pois o cenário modelado não é inteiramente apresentado ao jogador, apenas parte dele, mais especificamente, apenas o que está em seu campo de visão [Carlos, 2009].

Na geometria, *frustum* é a parte de um sólido que se encontra entre dois planos paralelos. Já na computação gráfica, o *frustum* consiste em seis planos, em que dois são paralelos e os outros são perpendiculares a esses, como ilustrado a Figura 2.

Os seis planos citados consistem em: plano superior, plano inferior, plano lateral direito, plano lateral esquerdo, fundo e um plano de frente. Sendo assim, os objetos dentro desses planos são os objetos que estão sendo observados, já os demais existem no cenário, mas não estão campo de visão do observador [Assarsson e Möller, 1999].

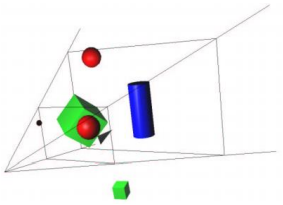

Figura 2. Exemplo de Frustum para a visualização de perspectiva
Adaptado de Assarsson e Möller, 1999

Existem inúmeros algoritmos que tentam determinar a visibilidade de objetos antes que sejam enviados para a placa gráfica. Um deles é o *Culling*, que consiste em um algoritmo que determina apenas o que será enviado para a placa de vídeo renderizar. Isso faz com que a placa de vídeo não gaste seu processamento desnecessariamente renderizando o que não está no campo de visão do jogador [Carlos, 2009].

Considerando que o cenário da Figura 2 seja enviado a uma placa de vídeo, apenas os sólidos que estão dentro do *Frustum* serão renderizados, os demais existem no mundo virtual, mas não são renderizados, não desperdiçando assim tempo de processamento desnecessariamente.

4. Desenvolvimento

Para o desenvolvimento deste trabalho foi utilizada a linguagem de programação C++ juntamente com as bibliotecas OpenCV e OpenGL[2]. A biblioteca OpenCV foi utilizada para o processamento da imagem capturada, detectando e identificando faces. Já a biblioteca OpenGL foi utilizada para a criação do mundo virtual. Como equipamento de hardware foi utilizado um notebook com processador Intel Core i7 - 740 QM (1,73GHz, Cache 6MB), memória 6GB DDR3 1066 MHz, placa de vídeo NVIDIA GeForce GT425M (1GB de memória dedicada) e *webcam* integrada com 1.3 mega pixels.

Devido ao fato de o *head-tracking* utilizar apenas uma webcam, que não é um equipamento intrusivo ao usuário, ele pode ser considerado como uma interface natural. Pois além de não utilizar algum equipamento intrusivo, o usuário pode agir naturalmente.

---

[2] Open Graphics Library - http://www.opengl.org/

Assim, o *head-tracking* pode ser utilizado em cenários virtuais com a finalidade de aumentar a sensação de tridimensionalidade das cenas, uma vez que o monitor poderia ser considerado uma janela virtual, pois conforme o usuário se movimenta no mundo real, o cenário se altera para apresentar os objetos conforme a posição em que o usuário se encontra em relação ao monitor, criando assim uma espécie de janela virtual.

O aplicativo desenvolvido neste trabalho teve o objetivo de simular a ideia de uma janela virtual, para que outros desenvolvedores pudessem aplicar esta ideia no uso em um jogo em primeira pessoa.

A Figura 3 apresenta o fluxo de trabalho do aplicativo desenvolvido. Ele consiste nas etapas de *head-tracking*, cálculos de deslocamentos e aproximações, *frustum culling* e alteração do campo de visão com base no resultado da movimentação no mundo real e o campo de visão atual do mundo virtual.

Os cálculos de movimento facial consistem no cálculo da movimentação realizada pelo usuário, a fim de identificar sua nova posição no mundo real para posteriormente alterar o campo de visão do mundo virtual através de sua posição no mundo real.

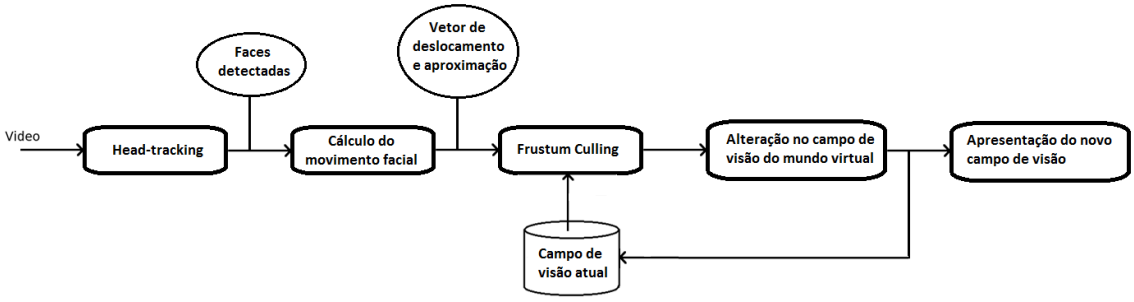

Figura 3. Fluxo de trabalho do aplicativo desenvolvido

Com o objetivo de melhor visualizar as movimentações e a ideia da janela virtual, foi modelado um mundo virtual contendo 900 caixotes dispostos com a mesma distância entre eles. A Figura 4 apresenta a posição inicial do observador e algumas informações sobre o que está sendo processado, como a taxa de amostragem e a quantidade de caixotes que estão sendo renderizados pela placa de vídeo. Nesta posição inicial são apresentados 809 dos 900 caixotes do mundo, isso com uma taxa de 28 *frames* por segundo (sem *head-tracking*) e desvio padrão de 1,45.

Com o mundo modelado, foram desenvolvidos os comandos para navegação dentro desse universo. Inicialmente foram feitos os comandos utilizando entradas padrões, como mouse e teclado, para testar o *frustum* e sua taxa de amostragem. Esse modo de movimentação e navegação no mundo virtual é semelhante ao utilizado pela maioria dos jogos virtuais. A utilização dessa técnica de navegação no mundo virtual não alterou a taxa de *frames* por segundos, permanecendo em 28 *frames* por segundo, com desvio padrão de 1,35.

Posteriormente, com o mundo modelado e validado, foi desenvolvido um módulo de *head-tracking* para poder integrá-lo ao mundo virtual. Para o desenvolvimento desse módulo foi utilizado o *Haar Cascade* presente na biblioteca OpenCV.

A detecção da face foi feita através da *webcam* presente no notebook utilizado. A ideia de utilizar a webcam de um notebook se deu devido ao seu posicionamento. Uma

vez que a webcam de um notebook geralmente se localiza centralizada e no topo da tela, ela facilita a identificação da face do jogador e também a identificação da movimentação de sua face.

O algoritmo de *head-tracking* consistiu nas etapas de detecção da face, a qual foi indicada com um retângulo ao redor de cada uma; calculado o centroide da face; e utilizado esse centroide como ponto de referência para movimentação do usuário, calculado seu deslocamento. Foi utilizado o centroide devido ao fato do usuário poder se aproximar da câmera, o que fazia com que qualquer outro ponto sofresse um grande deslocamento, movendo o mundo virtual de forma incorreta e instável.

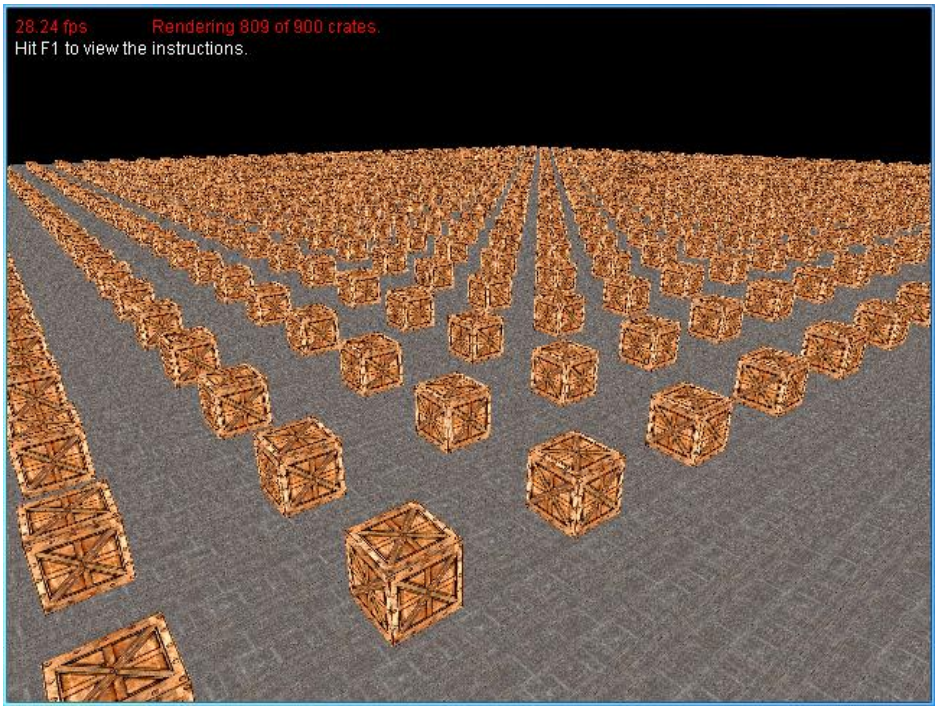

Figura 4. Mundo virtual modelado

Para identificar se o movimento do usuário estava sendo realizado corretamente, foi desenhado um vetor de deslocamento a partir do centroide da face. A direção do vetor indica a direção do movimento e o seu comprimento indica a velocidade desse movimento. A Figura 5 apresenta um exemplo da identificação da face (5.a) e o vetor de deslocamento (5.b).

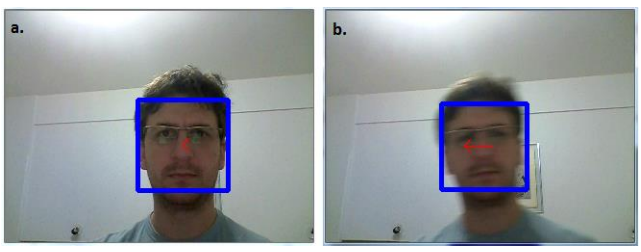

Figura 5. a. Head-Tracking b. Vetor de deslocamento

Com o módulo de reconhecimento facial validado por diferentes usuários, ele foi inserido no mundo virtual, substituindo os comandos anteriormente utilizados, teclado e

mouse. A Figura 6 apresenta a movimentação do usuário e a movimentação do mundo real, partindo do ponto de origem.

Por se tratar de uma janela virtual, a movimentação do usuário para a direita, faz com que seja apresentado o canto esquerdo da imagem virtual. Isso para dar a sensação de estar olhando por meio de uma janela. Assim, a quantidade de caixotes renderizados diminuiu de 809 caixotes do campo de visão inicial, para 772 caixotes do novo campo de visão. Além disso, com a utilização do *head-tracking* para movimentação do mundo virtual no lugar dos tradicionais mouse e teclado, a taxa de amostragem não sofreu grandes alterações, ficou em média 29 *frames* por segundo com desvio padrão de 2,48 para um total de 500 amostras.

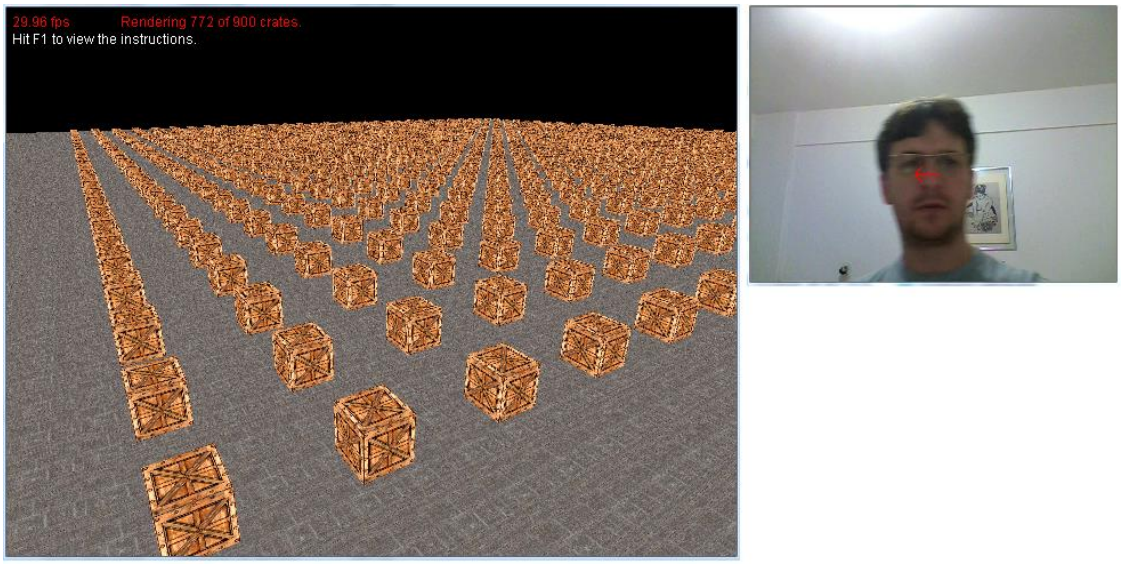

Figura 6. Exemplo de movimentação

Já Figura 7 apresenta algumas das possíveis movimentações do usuário, movendo-se para a direita (Figura 7.b), para a esquerda (Figura 7.c), para cima (Figura 7.d) e para baixo (Figura 7.e), além de apresentar também o ponto inicial (7.a). Além disso, a taxa de amostragem permaneceu em 29 *frames* por segundo. Porém, com um desvio padrão de 2,44 para um total de 1000 amostras.

No mundo virtual, bem como em uma janela, é comum o usuário se aproximar de objetos, os quais dão a sensação de "aumentarem" de tamanho ao chegar mais próximo. Para isso, foi aprimorado o módulo de *head-tracking* para que ele passasse a identificar a aproximação do usuário em relação à câmera, permitindo assim, aproximar o mundo virtual.

Por utilizar um vetor bidimensional de deslocamento, não é possível calcular a aproximação. Para isso foi utilizada uma técnica de aproximação por variação de área. A qual consiste na variação da área do retângulo utilizado para calcular o centroide da face, conforme a alteração dessa área, é calculada a aproximação do usuário.

Ao se aproximar da câmera, a área da face do usuário aumenta, com isso é possível perceber sua aproximação. Já quando se afasta a área de sua face diminui. Essa aproximação também foi feita no mundo virtual com campo de visão, pois ao se

aproximar, o campo de visão também deve se aproximar, e ao se afastar, o campo de visão também deve se afastar.

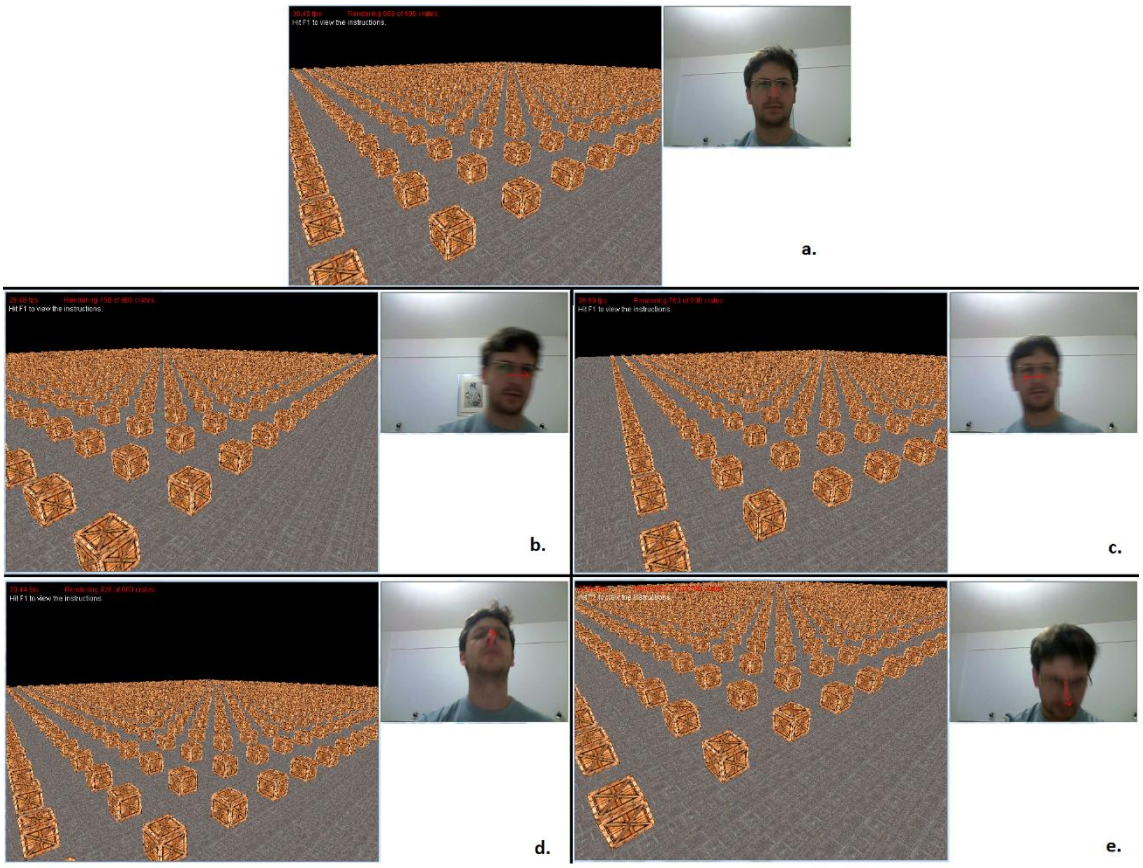

Figura 7. Movimentos. a. Padrão de referência, b. Usuário movendo-se para a direita, c. Usuário movendo-se para a esquerda, d. Usuário movendo-se para cima, e. Usuário movendo-se para baixo

Porém, quando se trabalha com área e aproximação, é necessário definir qual será a posição inicial do observador, para ter uma área de referência para o cálculo da aproximação. Neste caso foi considerada que a posição inicial do observador é a posição do campo de visão inicial e a área da face do observador é levada como referência para as aproximações.

A Figura 8 apresenta um exemplo da aproximação do usuário (Figura 8.b) e quando o usuário se afasta (Figura 8.c) em relação à posição inicial (Figura 8.a). Nesta figura é possível observar que ao se aproximar da câmera a quantidade de caixotes renderizados reduziu de 809 para 684. Já ao se afastar, a quantidade de caixotes aumentou de 809 para 824 caixotes, mantendo a taxa de amostragem em 29 *frames* por segundo com desvio padrão de 2,45 para um total de 500 amostras.

O *head-tracking* proposto foi feito com base na movimentação apenas em três eixos, horizontal, vertical e profundidade (afastamento e aproximação). Isso se deu devido à utilização de um único ponto ser rastreado durante a movimentação.

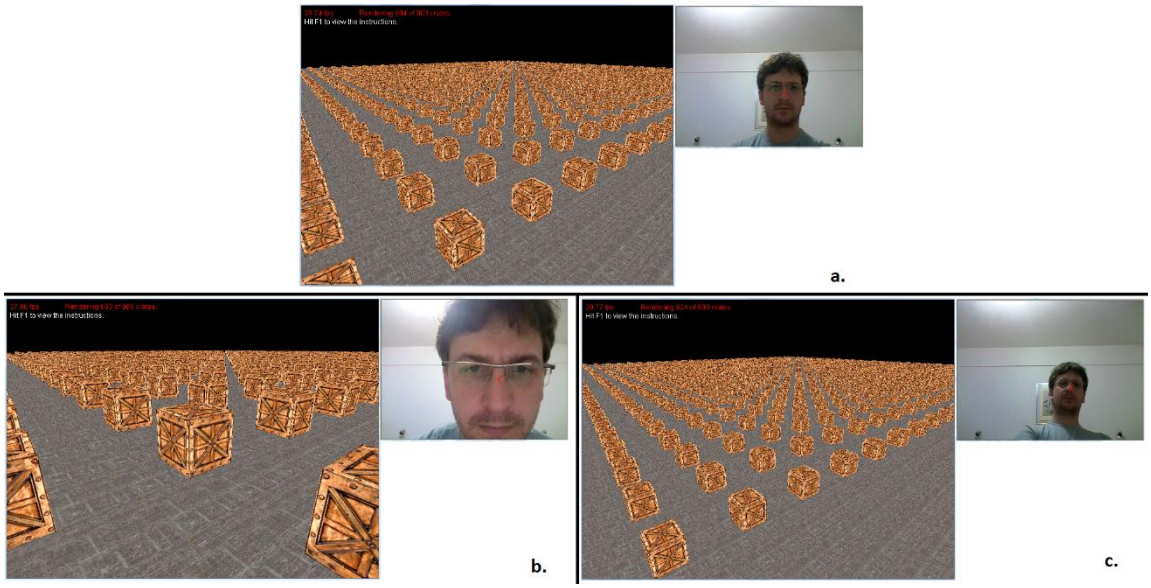

Figura 8. Exemplo de zoom a. Padrão de referência b. Aproximação do usuário c. Usuário se afastando da tela

5. Conclusões

Este trabalho apresentou um conceito que visa aumentar a sensação de tridimensionalidade através de uma janela virtual, na qual a posição e os movimentos do observador mudam o ponto de vista do mundo virtual observado. Para isso utilizou-se os conceitos de *head-tracking* e de *frustum culling.*

Para o *head-tracking* foi utilizada a biblioteca OpenCV. A qual permitiu realizar a identificação de faces na cena. Através do fluxo contínuo de imagem, esta identificação de faces permitiu identificar os movimentos realizados pelo observador. Estes movimentos foram transformados em um vetor de deslocamento e a variação da área da face, a qual permitiu aproximar ou afastar objetos virtuais.

Já o *frustum culling* foi utilizado para que a placa de vídeo renderizasse apenas os objetos que estivem sendo visualizados ou pertencessem à cena, o que permitia a alteração do campo de visão do observador em relação ao mundo virtual.

Com a finalidade de demonstrar a eficácia do uso de *head-tracking* e *frustum culling* para a criação de janelas virtuais, foi desenvolvido um aplicativo. Este aplicativo possuía um cenário com diversos caixotes espaçados igualmente no cenário. Através da movimentação do observador, a cena se adaptava à sua nova posição e alterava o campo de visão. Isso fazia com que a quantidade de caixotes visualizados em cada movimento fosse diferente.

A eficácia do uso dessa técnica se deu avaliando a taxa de atualização do mundo virtual. Para isso, foi utilizada a movimentação padrão na maioria dos jogos (*mouse* e teclado), o que resultou no nosso padrão para comparações futuras. Através desse tipo de movimentação no mundo virtual, foi encontrada uma taxa de atualização de 28 *frames* por segundo e desvio padrão de 1,45. Ao utilizar a movimentação do mundo virtual através dos movimentos do observador real, foi obtida uma taxa de atualização de 29 *frames* por segundo com desvio padrão de 2,45.

Considerando apenas o valor médio de atualização, parece que a utilização do *head-tracking* teve uma resultado melhor do que a utilização de *mouse* e teclado. Porém, ao avaliar o desvio padrão, é possível perceber que a utilização de *mouse* e teclado é mais estável. Esta variação pode ser dada devido à taxa de amostragem utilizada na webcam, 30 *frames* por segundo, e o tempo de processamento do *head-tracking*.

Ainda, através do vetor de deslocamento e da área da face do observador, o *head-tracking* aqui proposto identifica movimentos em apenas três eixos, vertical, horizontal e profundidade (afastamento e aproximação). Isso se dá devido à utilização de um único vetor de deslocamento localizado no centro da área em torno da face. Com a utilização de mais pontos para o cálculo de deslocamento, é possível a movimentação em outros eixos, como a rotação ou inclinação, por exemplo. Esses outros pontos a serem utilizados poderiam ser os olhos e a boca, utilizando assim, três pontos e três vetores de deslocamento.

Com relação ao aumento na sensação de tridimensionalidade, a proposta se mostrou efetiva, pois através da movimentação do usuário, o cenário se move para apresentar uma visão diferenciada. O mundo virtual não é apresentado de forma estática ao usuário, mas sim, é alterado conforme o usuário altera sua posição de visualização, como se estivesse olhando uma janela, mas virtual. Para avaliar isso foram utilizados alguns voluntários, os quais comentaram sobre a eficácia dessa técnica, mas como foram poucos voluntários, não gerou uma representatividade estatística para ser apresentada no trabalho.

Como trabalhos futuros fica a proposta para avaliar a eficácia da técnica de forma estatística testando com diversos voluntários e com um computador com uma arquitetura mais limitada (memória e placa de vídeo); a utilização de mais pontos de rastreamento (3 pontos); e também a utilização desta técnica para criação de uma janela virtual para visualização objetos reais - movimentar uma câmera de vigilância, por exemplo.

Além disso, o uso dessas técnicas poderia ser utilizado no desenvolvimento de novas obras artísticas, onde o usuário pode interagir com a obra, dando a sensação de que a obra se movimenta quando o observador faz algum movimento próximo à ela.

Referências bibliográficas

Assarsson, U., Möller, T. (1999). Optimized View Frustum Culling Algorithms. Technical Report 99-3.

Bradski, G., Kaehler, A. (2008). Learning OpenCV. O'Reilly Media, Inc. Sebastopol, CA. 1st edition.

Carlos, E. T. (2009). Frustum Culling Híbrido Utilizando CPU e GPU. Monografia (Mestrado) Programa de Pós-Graduação em Informática, PUC-RIO, Rio de Janeiro, RJ.

Francone, J., Nigay, L. (2011). Using the User's Point of View for Interaction on Mobile Devices. ACM International Conference of the Association Francophone d'Interaction Homme-Machine, ACM New York, NY, USA.

Gonzalez, R.C., Woods, R. E. (2008). Digital Image Processing, 3. ed. New Jersey:Pearson Prentice Hall.

Lee, J. C. (2007). Head Tracking for Desktop VR Displays using the Wii Remote. Disponível em: http://johnnylee.net/projects/wii/. Acessado em: novembro de 2012.

Lienhart, R., Maydt, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP, Vol. 1, pp. 900-903.

Papageorgiou, C., Oren, M., Poggio, T. (1998). A general framework for object detection. In International Conference on Computer Vision.

Sung, K., Poggio, T. (1998). Example-Based Learning for View-Based Human Face Detection. IEEE Transactions on Pattern analysis and Machine Intelligence, Vol. 20, No. 1.

Toyama, K. (1998). "Look, Ma – No Hands!" Hands-Free Cursor Control with Real-Time 3D Face Tracking. In Proc. PUI'98, pp. 49-54.

Viola, P., Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. Conference on Computer Vision And Pattern Recognition.

Yang, J., Stiefelhagen, R., Meier, U., Waibel, A. (1998). Visual Tracking for Multimodal Human Computer Interaction . Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Pages 140-147.