UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PARALAXE EM AMBIENTES VIRTUAIS

Nome:	Maria Luíza Santinho Lima Monteiro	RA: 11021012
Email:	marialuiza_mah@hotmail.com	

BAURU
2015

Maria Luíza Santinho Lima Monteiro

PARALAXE EM AMBIENTES VIRTUAIS

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: José Remo Ferreira Brega

BAURU
2015

AGRADECIMENTOS

A Deus, pelo dom da minha vida e todas as bênçãos derramadas.

À Universidade Estadual Paulista, campus de Bauru, pela oportunidade de fazer este curso.

Ao meu orientador, professor José Remo, por todo o auxílio e ensinamentos. A todos os docentes do Departamento de Computação, em especial aos professores Simone e Wilson, que ministraram as disciplinas de Projeto e Implementação de Sistemas 1 e 2 e à professora Márcia que compõe a banca.

Aos colegas, em especial ao Mário, que esteve trabalhando na mesma linha de pesquisa, pelo auxílio.

À minha família, em especial minha mãe Leila, ao meu pai Eduardo e aos meus avós Ieda e Lúcio, por todo o amor e por sempre me incentivarem e torcerem a cada etapa da minha vida pessoal e acadêmica. Aos meus tios e primos, em especial à minha prima Maria Fernanda, pela ajuda e apoio.

Ao meu namorado Vitor, pelo carinho e companheirismo.

Aos meus amigos, que direta ou indiretamente me ajudaram.

A todos que fizeram parte da minha formação.

Muito obrigada.

“Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota.”

(Madre Teresa de Calcutá)

RESUMO

O movimento da paralaxe em um ambiente virtual é relacionado à captura da posição da cabeça do usuário, sendo que o ambiente se move de acordo com a visão do usuário. Fundamentado na literatura estudada, a paralaxe causa melhores resultados em relação à imersão no ambiente e percepção da profundidade, comparado à estereoscopia. Existem aplicações do movimento da paralaxe utilizando outros dispositivos de captação da posição do usuário, mas o objetivo do presente trabalho é aplicar o movimento da paralaxe utilizando o sensor Kinect para tal captura. Uma aplicação exemplo foi desenvolvida com a implementação do movimento da paralaxe e os resultados apontam que é possível gerar o movimento da paralaxe utilizando o sensor Kinect para captura dos dados.

Palavras-chave

Paralaxe. Kinect. Ambiente Virtual.

ABSTRACT

The motion parallax in virtual environment is related to the capture the user's head position, the environment moves according the user viewpoint. Based in the studied literature, parallax causes better results in relation to immersing the environment and depth perception, beside the stereoscopy. There are motion parallax applications using others user position capture devices, but the objective of this study is to apply the motion parallax using the Kinect sensor to data capture. An application was developed using the motion parallax and the results show that is possible to create the motion parallax using Kinect sensor for data capture.

Key-words

Motion Parallax. Kinect. Virtual Enviroment.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Revisão Sistemática

A revisão sistemática, de acordo com Kitchenham et al. (2009), é a identificação, avaliação e interpretação de material científico para uma questão de pesquisa, tópico ou fenômeno de interesse. Segundo estes autores, existem diversos motivos para a execução da revisão sistemática, dentre eles:

a) identificar lacunas em uma pesquisa com a finalidade de sugerir áreas para maiores investigações;
b) sumarizar evidências sobre algum tratamento ou tecnologia, para se obter dados empíricos dos benefícios e limitações de um método ágil específico;
c) fornecer uma base com a finalidade de proporcionar novas atividades de pesquisa apropriadamente.

O trabalho terá como fundamentação teórica uma revisão sistemática aplicada de acordo com Kitchenham et al. (2009), com trabalhos científicos resultados de uma lógica de busca LB1, que segue:

(("virtual reality") OR ("virtual environment")) AND (("graphic engine") OR ("game engine")) AND (("multiple projection") OR ("multiple screen") OR ("multi-projection") OR ("multiprojector displays") OR ("CAVE")) AND (("interaction") OR ("interactive"))

1.2 Objetivos

O principal objetivo deste projeto é aplicar o movimento da paralaxe utilizando o sensor Kinect em ambientes multiprojetados e verificar sua usabilidade.

Por objetivos secundários, temos o desenvolvimento de uma aplicação para implementar o movimento da paralaxe, bem como estudar sobre Unity, aplicações VRPN, ambientes virtuais multiprojetados e realidade virtual.

1.3 Justificativa

O tema para este projeto foi escolhido devido ao avanço da tecnologia, que acarreta maior necessidade de um ambiente imersivo, tanto para jogos, quanto para simuladores; onde o usuário possa ter uma experiência de realidade virtual cada vez melhor. Para isso, dentre outros recursos, pode-se utilizar a paralaxe, que é o movimento do ambiente virtual conforme

o movimento da cabeça do usuário. Já existem alguns experimentos desta natureza, contudo, no presente estudo, será utilizado o sensor Kinect, a fim de verificar sua eficácia, e pontos positivos e negativos da sua utilização para fazer o movimento da paralaxe.

1.4 Organização do Trabalho

Este trabalho está organizado em 5 capítulos, juntamente com este primeiro, divididos da seguinte maneira:

O Capítulo 2, **Realidade Virtual**, apresenta o levantamento científico com definições importantes sobre temas relacionados à realidade virtual.

O Capítulo 3, **Dispositivos e Ferramentas**, apresenta definições e conceitos sobre os dispositivos e ferramentas necessários para a realização do presente trabalho.

O Capítulo 4, **Desenvolvimento**, apresenta todo o desenvolvimento deste trabalho, considerando as conexões e programação de aplicações.

Por fim, o Capítulo 5, **Considerações Finais e Discussão dos Resultados**, apresenta imagens dos resultados e as considerações finais do presente trabalho.

2 REALIDADE VIRTUAL

2.1 Definições

Realidade Virtual (RV) pode ser definida como uma experiência onde o usuário está efetivamente imerso em um mundo virtual e tem o controle dinâmico do ponto de visão neste mundo (BROOKS, 1999).

RV é composta por um mundo virtual interativo, onde o visualizador está imerso e este ambiente sintético tridimensional responde a suas ações (SHERMAN; CRAIG, 2002).

Segundo Ellis (1994), Ambientes Virtuais (AV) são imagens computadorizadas interativas de acordo com o ponto de visão do usuário, promovendo a ilusão do deslocamento do mesmo para outro local.

2.2 Multiprojeção

Em 1992, Cruz-Neira et al. (1993) apresentaram o CAVE™, que consiste em um sistema que usa multiprojeção para aplicações de RV e visualização científica. Criado pelo laboratório *Eletronic Visualization Lab* (EVL), o CAVE™ teve sua exibição no SIGGRAPH'92. Ele apresenta um AV multiprojetado com base no ponto de visão do usuário e imagens estereoscópicas.

Nesta apresentação de 1992, o CAVE™ era composto por quatro projeções, por meio de quatro estações Silicon Graphics, uma para cada tela. Controladores de áudio foram utilizados para o uso de múltiplos autofalantes e foi possível rastrear a posição da cabeça e da mão do usuário por meio dos sensores Polhemus e Ascension (CRUZ-NEIRA et al., 1993). De acordo com estes autores, a visão de câmera é a projeção mais comum em computação gráfica, esta simula o modo no qual a imagem é capturada em um filme. No padrão de projeção por câmera, imagens estereoscópicas são normalmente geradas por duas câmeras, uma para cada olho.

Atualmente, o CAVE™ se encontra na sua terceira geração, desenvolvida por DeFanti et al. (2009), o StarCAVE possui alta resolução alcançada por meio de projetores LCOS LCD e telas dispostas em forma de pentágono, como pode ser visto na Figura 1. Uma novidade desta terceira geração é a utilização de estereoscopia passiva com uso de óculos polarizado, ao contrário das gerações anteriores que utilizavam estereoscopia ativa com uso de óculos *shutter*.

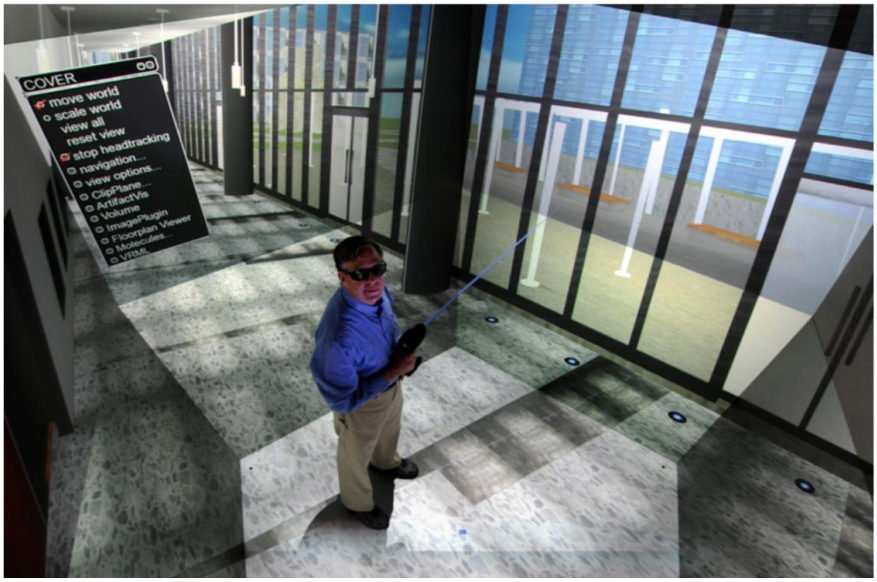

Figura 1 – StarCAVE (DEFANTI et al., 2009).

Os AV multiprojetados são comumente utilizados com o objetivo de prover maior imersão ao usuário. Segundo Staadt et al. (2003), a multiprojeção do ambiente é empregada para as seguintes situações:

a) exibição de imagens com alta resolução, excedendo as disponíveis por monitores de placas gráficas;
b) fornecimento de um grande campo de visão e melhor imersão do cenário.

Sowndararajan et al. (2008) compararam um AV com alto nível de imersão, utilizando duas telas projetadas baseadas no CAVE™ (Figura 2), com um AV de baixo nível de imersão, utilizando uma tela de um *desktop*.

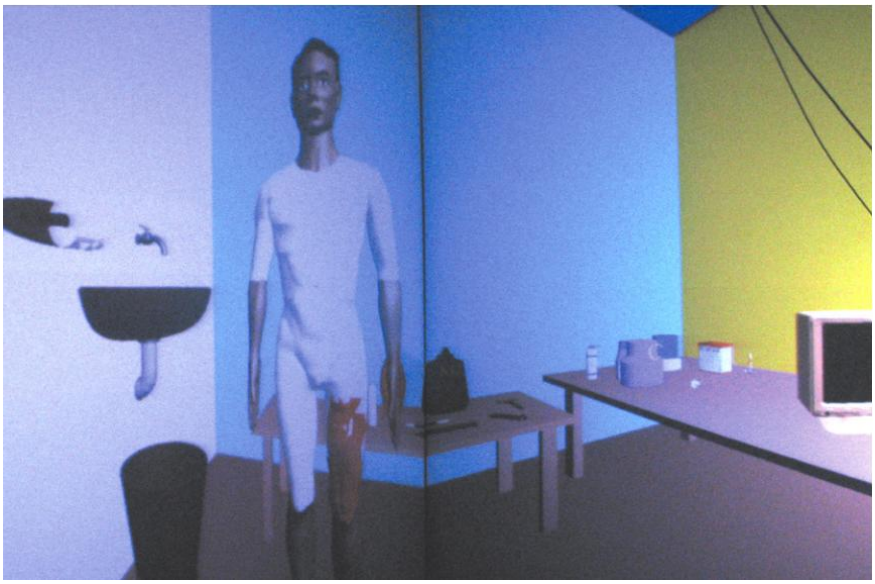

Figura 2 – AV de Alto Nível de Imersão (SOWNDARARAJAN et al., 2008).

Sowndararajan et al. (2008) concluíram em sua pesquisa, que AV de alto nível de imersão são mais efetivos no aprendizado de procedimentos complexos, que requerem a noção de localização espacial (Figura 3).

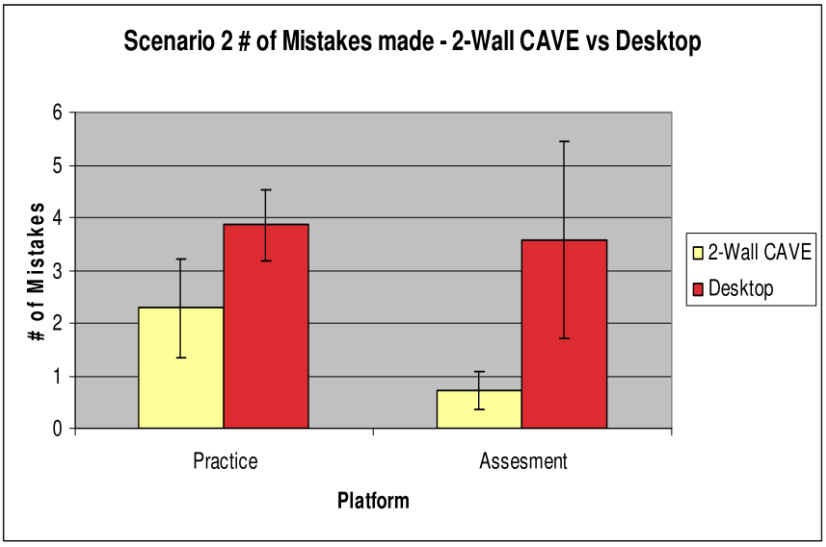

Figura 3 – Treinamento Procedural em AV de Baixo e Alto Nível de Imersão (SOWNDARARAJAN et al., 2008).

2.3 Paralaxe

O movimento de paralaxe verificou-se como sendo um benefício igual ou até maior que a estereoscopia para a percepção de profundidade (LUGRIN et al, 2013). No estudo em questão, participantes com baixa percepção em estéreo apresentaram um desempenho comparado a pessoas com visão normal, ou seja, melhor compreensão do ambiente virtual e da profundidade que apresenta, quando o movimento de paralaxe estava presente. Os autores concluíram que os efeitos de latência e posição de barulho são cruciais para a sensação do participante do movimento de paralaxe. De acordo com Cruz-Neira et al. (1993), a projeção da perspectiva é calculada de acordo com a visão do usuário, gerando o movimento de paralaxe, mostrado na Figura 4.

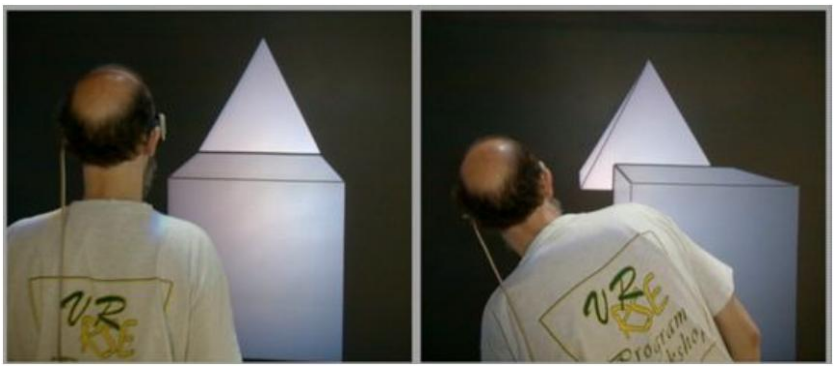

Figura 4 – Movimento de Paralaxe (SHERMAN et al., 2013).

3 DISPOSITIVOS E FERRAMENTAS

3.1 Motor de Jogo

Para o desenvolvimento de jogos digitais, pode-se usar uma ferramenta chamada motor de jogo, que consiste em códigos fonte que podem ser reutilizados em uma família de jogos (LEWIS; JACOBSON, 2002). A Figura 5 representa a estrutura modular de um motor de jogo.

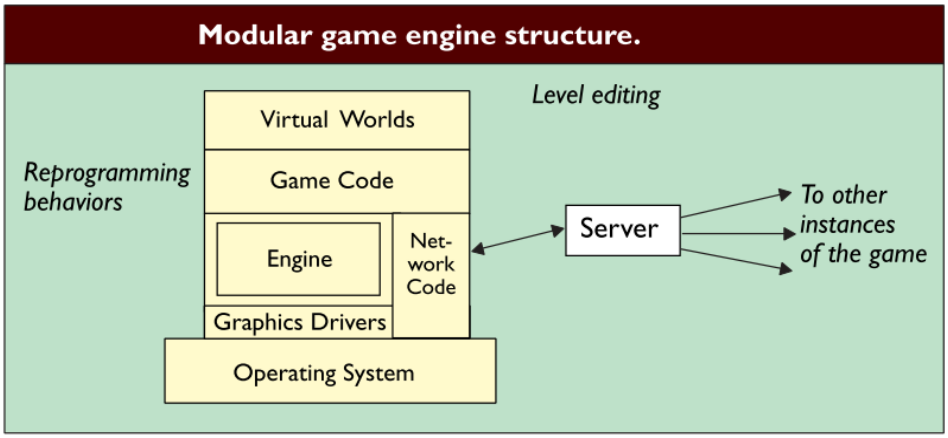

Figura 5 – Estrutura Modular de um Motor de Jogo (LEWIS; JACOBSON, 2002).

De acordo com Szilas et al. (2007), motores de jogo são *frameworks* que viabilizam os elementos básicos para o desenvolvimento de jogos, eles geralmente possuem:

a) bibliotecas gráficas de renderização 3D em tempo real;
b) *interface* de edição;
c) meios para a criação de ações, como o comportamento dos personagens do jogo não controlados pelo jogador, chamada de inteligência artificial;
d) ferramentas para a importação de modelos 3D de softwares de modelagem e animação.

A princípio, motores de jogos eram utilizados apenas pelas companhias de desenvolvimento, mas atualmente são fornecidos juntamente com o jogo, possibilitando ao jogador fazer modificações, criando assim, sua própria versão. O surgimento dos motores de jogo permite que códigos fonte utilizados no desenvolvimento de jogos, possam ser aplicados também em pesquisas científicas (LEWIS; JACOBSON, 2002).

De acordo com Szilas et al. (2007), motores de jogo fornecem algumas das melhores tecnologias 3D. Eles permitem o desenvolvimento em alto nível; permitindo aos desenvolvedores focar em animação, comportamento e narrativa.

A RV pode fazer uso dos novos métodos e recursos de jogos digitais, uma vez que ambas exploram os avanços tecnológicos de diversos campos (BOUVIER et al., 2008).

3.2 Unity

Segundo Filho et al. (2011), Unity 3D é um motor de jogo completo, que oferece a possibilidade de criação de aplicações 2D e 3D em tempo real. Possui sete tipos de licenças que abrangem sua portabilidade: Unity, Unity Pro, Unity iOS, Unity iOS Pro, Unity Android, Unity Android Pro e Unity Wii. É possível criar jogos sem custo na versão *free*. Unity tem suporte para os consoles Microsoft Xbox 360, Sony Playstation 3 e aplicações em Adobe Flash.

Ainda de acordo com estes autores, as principais propriedades do Unity são:

a) **Portabilidade:** O resultado final gerado pelo Unity pode ser exportado para versão *web* ou *desktop*. Na versão *web*, é possível exportar para os seguintes navegadores: Internet Explorer, Mozilla Firefox, Google Chrome, Opera e Safari. A versão *desktop* por sua vez, exporta arquivos executáveis para as plataformas Windows e Mac.

b) **Sistema de Física:** O Unity implementa as propriedades de um sistema de física, como massa, gravidade, inércia, colisão, entre outros, através do motor de física PhysX da Nvidia. O PhysX é considerado um do mais completos motores de física do mercado. Na *interface* do Unity há componentes de física predefinidos que permitem o desenvolvimento rápido e controle eficientes que aumentam a produtividade. No Unity existem diversos tipos de materiais que simulam ferro, borracha, madeira e ainda é possível criar um material específico.

c) ***Interface* do Editor:** A *interface* possui várias janelas que podem ser ordenadas e reagrupadas para facilitar o trabalho. A Figura 6 mostra o ambiente Unity de modo geral. As principais janelas são: *Scene*, *Game*, *Hierarchy*, *Project*, *Inspector*, *Animation* e *Console*.

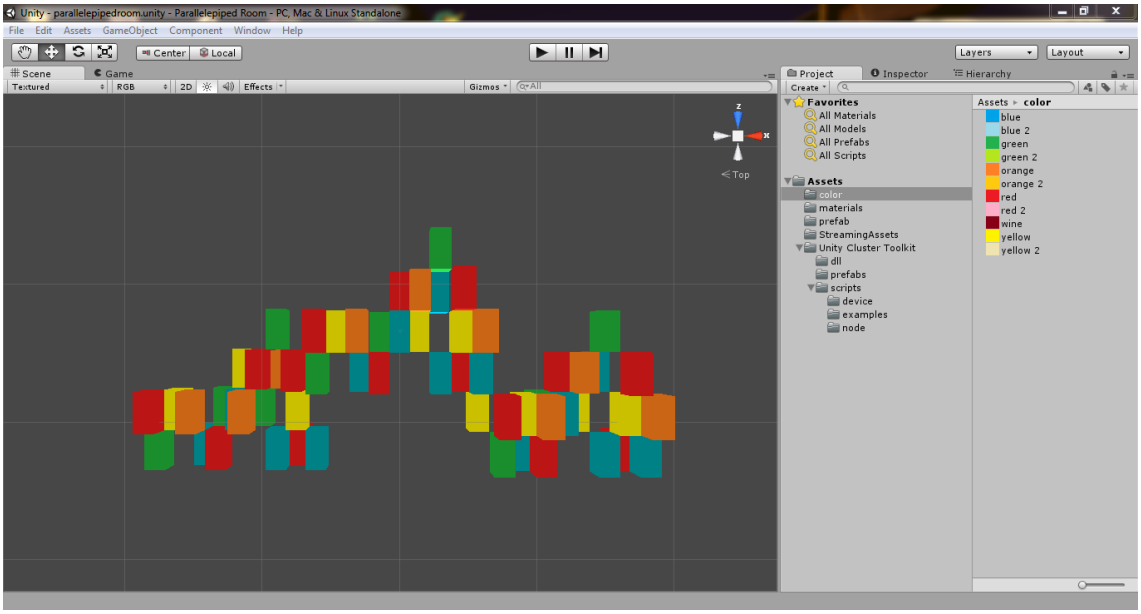

Figura 6 – Ambiente Unity.

– *Scene*: Janela onde os objetos são organizados. Permite uma navegação livre, com mouse e teclado para melhor visualizar o ambiente. Nesta janela é possível manipular graficamente os objetos através das opções de arrastar e soltar o mouse, como as câmeras, cenários, personagens e todos os elementos que compõem a cena. A Figura 7 exibe a janela *Scene*.

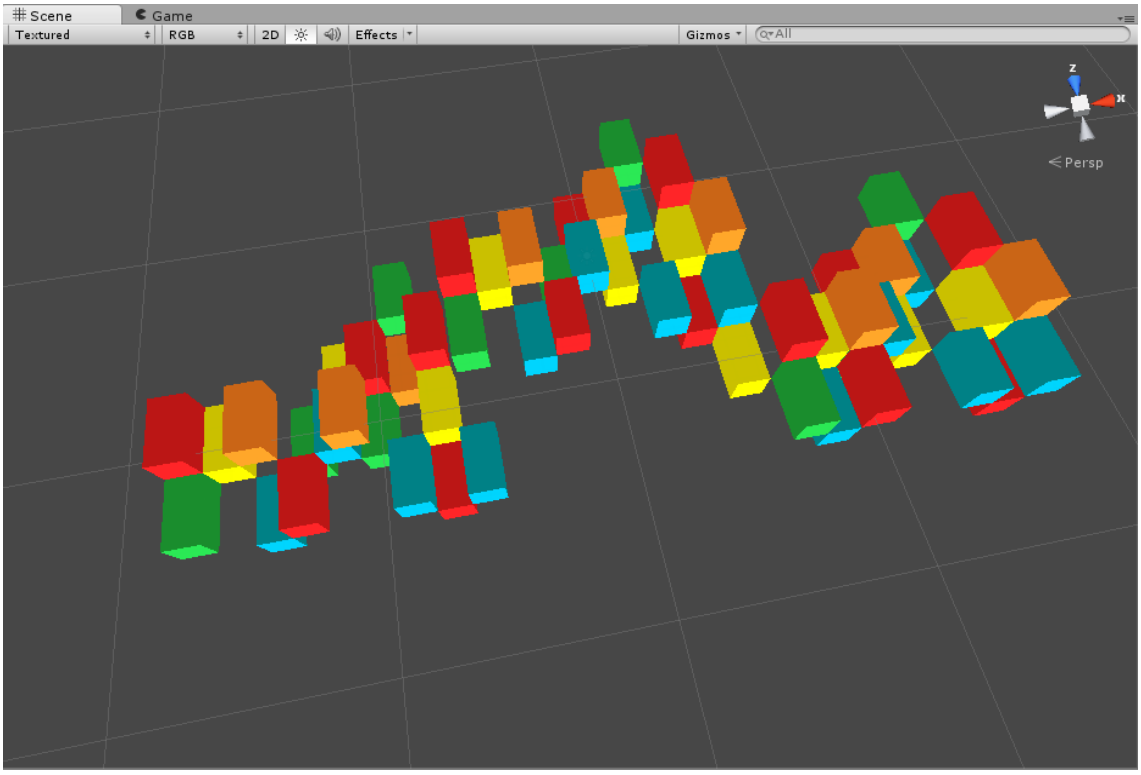

Figura 7 – Janela *Scene*.

– *Game*: Nesta janela é possível executar em tempo real a aplicação que está na *Scene*. Nela existe uma *interface* que mostra as estatísticas gráficas, como, por exemplo consumo de memória, quantidade de triângulos e vértices renderizados e caso a aplicação seja em rede, mostra a quantidade de *players* conectados. Podemos observar a janela *Game* na Figura 8.

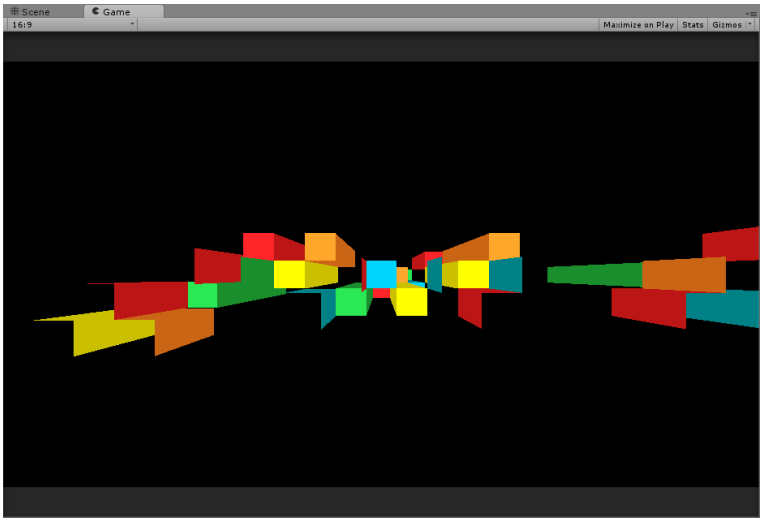

Figura 8 – Janela *Game*.

– *Hierarchy*: Mostra todos os elementos da cena. Esta janela possibilita a seleção rápida de um objeto da cena, bem como a verificação da sua hierarquia, como é mostrado na Figura 9. É possível também duplicar o objeto, renomeá-lo e deletá-lo.

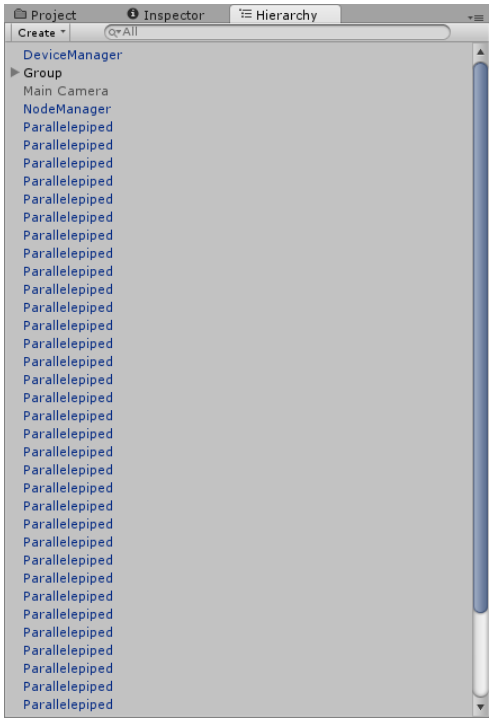

Figura 9 – Janela *Hierarchy*.

– *Project*: Agrupa todos os recursos incluídos no projeto, mesmo os que não estão na cena. Todos os elementos que compõem a aplicação precisam ser carregados nesta janela. A Figura 10 mostra a janela *Project* com uma pasta selecionada.

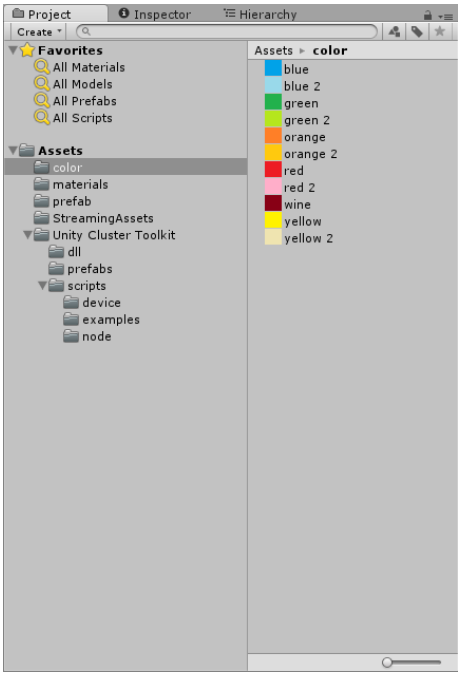

Figura 10 – Janela *Project*.

– *Inspector*: Nesta janela são alteradas as propriedades dos objetos que estão nas janelas *Project* e *Hierarchy*, por isso é uma das janelas mais importantes do Unity. É possível ver todas as propriedades mostradas na Janela *Inspector* na Figura 11.

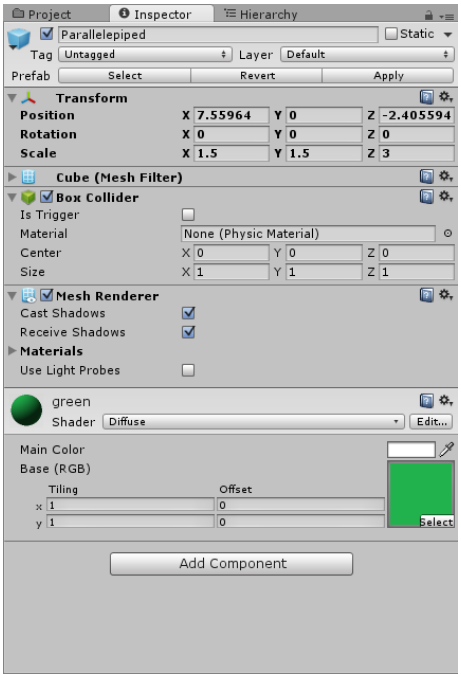

Figura 11 – Janela *Inspector*.

– *Animation*: Permite edição e criação de animações controlando o tempo de execução dos atributos do *Inspector*.

– *Console*: É utilizada para visualizar eventuais erros que podem ocorrer em tempo de execução. É útil também para controle de *logs*.

A *interface* do Unity também possui diversos menus no topo da aplicação, sendo os mais importantes deles: *GameObject*, *Assets* e *Component*.

– *GameObject*: Principal elemento que incorpora as propriedades de diversos componentes. Pode ser criado vazio ou escolhido nas seguintes opções: partícula, câmera, GUI, texto 3D, luz, cubo, esfera, cápsula, cilindro, plano, roupa, árvore, região de vento, entre outras, como é mostrado na Figura 12.

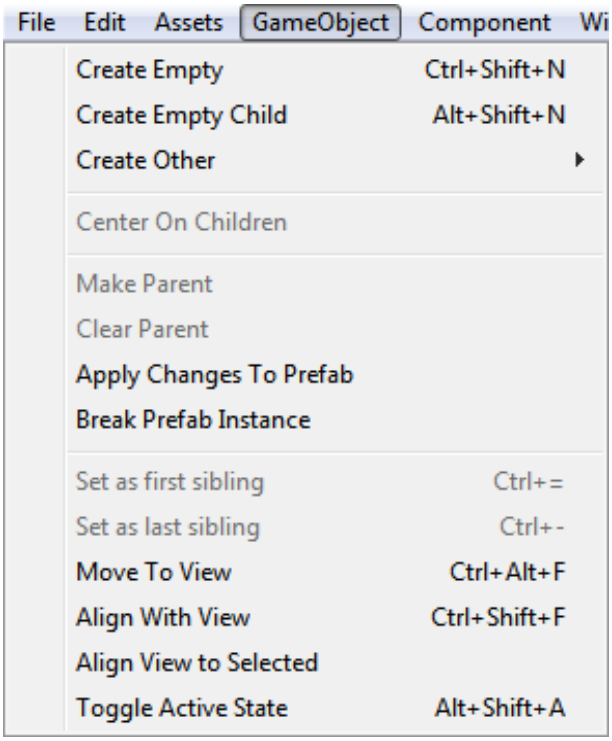

Figura 12 – Menu *GameObject*.

– *Assets*: Este menu permite criar ou adicionar mídias ao projeto, como imagens, sons, criação de pastas. É possível também importar *packages*, que são uma coleção de *Assets*. Estes incluem a criação de animações, *scripts*, *shaders*, materiais comuns e de física, além de GUI's e opção de sincronização com um projeto do Microsoft Visual Studio. A Figura 13 mostra o Menu *Assets*.

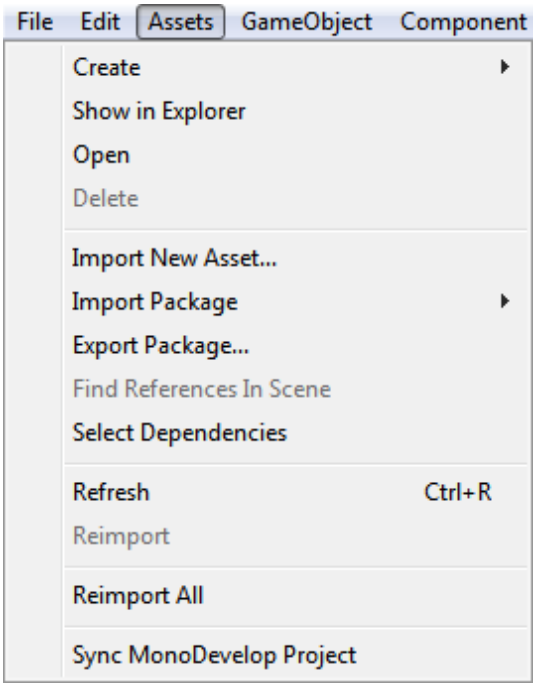

Figura 13 – Menu *Assets*.

– *Component*: Permite ao desenvolvedor atribuir vários elementos ao seu projeto. Estes elementos são agrupados nas opções: *mesh*, *effects*, *physics*, *physics2D*, *navigation*, *audio*, *rendering*, *miscellaneous* e *scripts*, como pode ser visto na Figura 14.

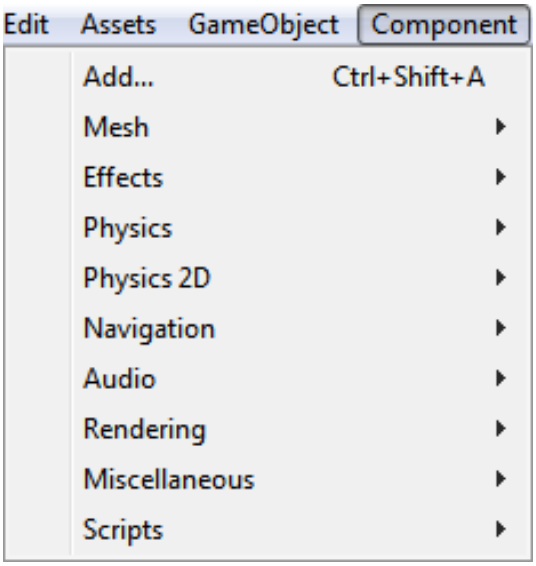

Figura 14 – Menu *Component*.

3.3 Virtual-Reality Peripheral Network (VRPN)

O sistema VRPN é um conjunto de classes dentro de uma biblioteca e um conjunto de servidores que fornecem uma *interface* independente do dispositivo e transparente à rede

entre aplicações e o conjunto de dispositivos físicos usados em sistemas RV. Aplicações VRPN de fatoração por função e de camadas no contexto dos dispositivos produzem uma *interface* nova e poderosa. VRPN também integra uma vasta gama de conhecidas técnicas avançadas em um sistema de publicidade disponível. Essas técnicas beneficiam diretamente os usuários VRPN e aqueles que implementam outras aplicações e fazem uso dos periféricos de RV (TAYLOR II et al., 2001).

De acordo com estes autores, VRPN fornece:

a) Acesso a uma variedade de periféricos RV através de uma *interface* comum e extensível;
b) *Interface* de rede transparente para dispositivos;
c) Marcação de hora e data para todas as mensagens de e para os dispositivos;
d) Sincronização entre clientes e servidores em diferentes máquinas;
e) Conexões múltiplas e simultâneas entre dispositivos;
f) Reconexão automática para servidores remotos que falharam;
g) Armazenamento e repetição de sessões interativas.

3.4 Flexible Action and Articulated Skeleton Toolkit (FAAST)

O FAAST é um *middleware* para facilitar a integração de controle corporal com jogos e aplicações de realidade virtual utilizando OpenNI ou o *software* de rastreamento de esqueleto do Microsoft Kinect para Windows. Ele incorpora um servidor VRPN personalizado para transmitir os dados das juntas do esqueleto do usuário pela rede, permitindo que aplicações de RV leiam as informações das juntas dos esqueletos como rastreadores, usando qualquer VRPN cliente. Adicionalmente, a *toolkit* pode também emular entrada de teclado através da postura corporal e de gestos específicos. Isto permite que o usuário adicione mecanismos baseados em controle corporal personalizado para jogos existentes que não fornecem suporte oficial para sensores de profundidade. (FAAST, 2015)

De acordo com Suma et al. (2011), o FAAST foi inicialmente desenvolvido para prover uma *interface* conveniente e acessível para o PrimeSensor™Reference Design. Esta tecnologia, baseada em luz estruturada infravermelha para computar uma imagem com profundidade no ambiente foi licenciada pela Microsoft para o Kinect. O *software* OpenNI é compatível com ambos os sensores e juntamente com o *middleware* NITE, executa identificação de usuário e reconhecimento básico de gestos usando a imagem de profundidade do sensor. O FAAST interage diretamente com OpenNI/NITE para acessar essas informações

e executa reconhecimento adicional de gestos de alto nível para gerar eventos baseados na ação do usuário.

A informação da posição corporal pode ser usada para movimentação realista de *avatar* virtual ou controle do cursor do *mouse* na tela. As ações e ligações de entrada são configuráveis em tempo de execução, permitindo ao usuário personalizar os controles e a sensibilidade para ajustar os tipos de corpo e preferências individuais (SUMA et al., 2011).

Ainda segundo estes autores, o FAAST considera duas categorias de informação do sensor: ações e articulações do esqueleto. As articulações do esqueleto consistem nas posições e orientação de cada junta em uma figura humana e é útil para aplicações de realidade virtual e videogames. O FAAST recupera essas juntas do esqueleto dos *drivers* OpenNI e as transmite para a aplicação de usuário final usando o VRPN, cada junta equivale a um número, como mostra a Tabela 1.

Tabela 1 – Juntas do Esqueleto (FAAST,2015).

Sensor	Junta	Sensor	Junta
0	Cabeça	12	Cotovelo Direito
1	Pescoço	13	Pulso Direito
2	Torso	14	Mão Direita
3	Cintura	15	Ponta do Dedo Direita
4	Clavícula Esquerda	16	Quadril Esquerdo
5	Ombro Esquerdo	17	Joelho Esquerdo
6	Cotovelo Esquerdo	18	Tornozelo Esquerdo
7	Pulso Esquerdo	19	Pé Esquerdo
8	Mão Esquerda	20	Quadril Direito
9	Ponta do Dedo Esquerda	21	Joelho Direito
10	Clavícula Direita	22	Tornozelo Direito
11	Ombro Direito	23	Pé Direito

3.5 Microsoft Kinect para Xbox 360

O Kinect é um dispositivo que acopla sensor de movimento, rastreamento do esqueleto e reconhecimento facial. Na presente pesquisa é utilizado o Kinect para Xbox 360, utilizando as ferramentas de sensor de movimento e rastreamento do esqueleto. O sensor de movimento do Kinect faz rastreamento do corpo por inteiro, reconhecendo todas as partes do corpo ao serem movimentadas. Baseado nas informações captadas, o Kinect cria um esqueleto digital que se move da mesma maneira que o usuário (XBOX, 2015).

O Microsoft Kinect para Xbox 360 tem vários benefícios que não são compartilhados por abordagens para RV tradicionais de rastreamento de movimento (GREUTER; ROBERTS,

2011). De acordo com Steptoe et al. (2013), o Kinect é capaz de prover dados do esqueleto de alta qualidade em tempo real sem precisar que os participantes sejam equipados com roupa de captura de movimento e calibrados no sistema.

Com os recentes avanços no campo da tecnologia de rastreamento sem marcadores, o Microsoft Kinect conduz para novas possibilidades de rastreamento em tempo real de baixo custo. Enquanto sistemas de rastreamento profissionais provêm precisão superior, com o barato rastreamento de corpo inteiro sem marcadores do Microsoft Kinect, agora é possível ter dados 3D em tempo real do corpo de uma pessoa e sua posição, o que permite aos animadores a captura natural e realista de movimentos 3D (BEIMLER et al., 2013).

3.6 Unity Indie VRPN Adapter (UIVA)

UIVA é um *middleware* baseado em conexão para adaptar VRPN para a versão Indie do motor de jogo Unity3D. Ele consiste de um lado servidor e um lado cliente. O lado cliente do UIVA é um arquivo DLL que reside no motor de jogo Unity3D e expõe alguns dados de funções para *scripts* C# que os empregam. O lado servidor do UIVA contém um cliente VRPN que conversa com o servidor VRPN que está sendo utilizado para adquirir os dados do sensor corrente (UIVA, 2015).

O diagrama representado na Figura 15 explica o conceito de base, bem como a circulação de dados de um quadro de jogo.

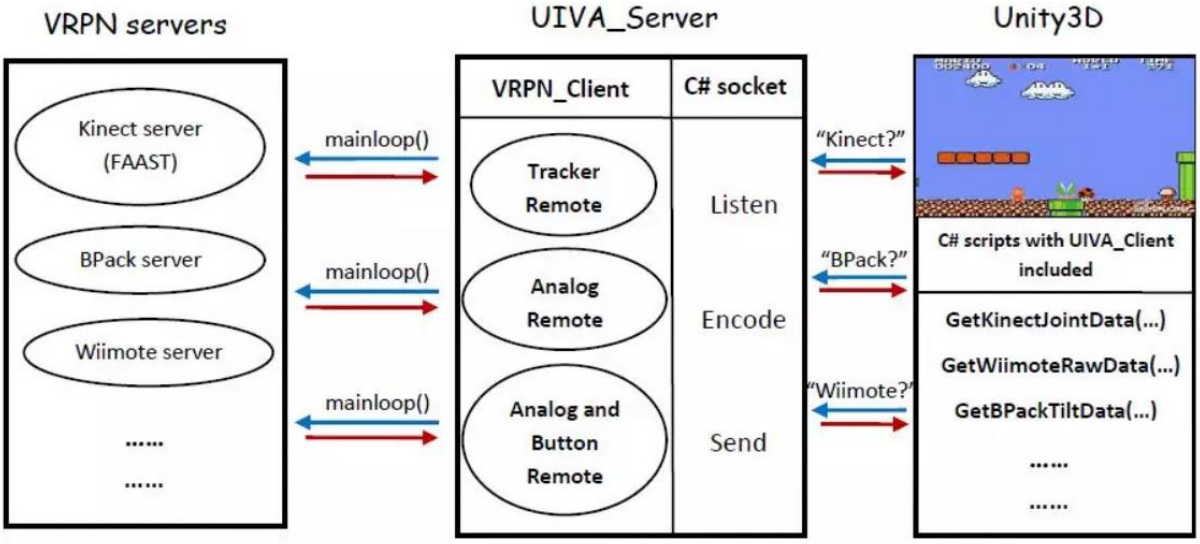

Figura 15 – Representação do Conceito de Circulação de Dados Utilizando Servidor UIVA (UIVA, 2015).

O UIVA é código aberto e é gratuito para uso e distribuição para pesquisa e propósitos não comerciais. Os dispositivos a seguir são suportados pelo UIVA na versão 1.02:

a) Microsoft Kinect
b) Nintendo WiiMote
c) Nintendo Wii Fit *balance board*
d) Wireless-T BPack *accelerometer*
e) *General mouse*
f) SpacePoint Fusion *sensor* (versão 1.01)
g) PhaseSpace *optical motion capture system* (versão 1.02)

4 DESENVOLVIMENTO

Após estudar todas as ferramentas necessárias, o projeto foi desenvolvido da seguinte maneira:

4.1 Conexão entre o computador e o sensor Kinect

A primeira tarefa a ser executada é a conexão entre o computador e o sensor Kinect para Xbox 360, desta forma é possível receber os dados gerados pelo sensor. Esta conexão é feita através do *middleware* UIVA, que se conecta ao servidor VRPN, chamado FAAST. Isso torna a aplicação independente do sensor utilizado, porque o servidor converte todos os dados recebidos em variáveis do mesmo tipo, no caso distância em metros. Caso seja necessário alterar o sensor, basta executar outro servidor VRPN compatível com o sensor, que a rotina funcionará da mesma forma, pois os valores serão recebidos no mesmo formato.

4.2 Desenvolvimento da rotina da paralaxe

Através do FAAST, a posição da cabeça do usuário é recebida e armazenada em um vetor do tipo *double* de três posições, para armazenar as distâncias *x*, *y* e *z*, bem como outros dados não utilizados neste trabalho, que são armazenados em um vetor do tipo *double* de quatro posições como é mostrado no código da Figura 16.

```
public static void UpdateUIVA(){
    double[] position = new double[3];
    double[] quaternion = new double[4];

    ClientUIVA.GetKinectJointData (0, ref position, ref quaternion);

    if(position [0] == 0 && position [1] == 0 && position [2] == 0) {
        headPosition.x = 0.0f;
        headPosition.y = 5.0f;
        headPosition.z = 0.0f;
    } else {
        headPosition.x = Convert.ToSingle(position [0]) * 7.11f;
        headPosition.y = Convert.ToSingle(position [2]) * 2.0f;
        headPosition.z = (Convert.ToSingle(position [1]) *  6.6f) - 5.0f;
    }
}
```

Figura 16 – Código da captura dos dados e do cálculo da posição da cabeça do usuário.

De acordo com as dimensões das telas do MiniCAVE, representado na Figura 17, um cálculo é realizado com cada uma das posições do vetor (Figura 16) e então o valor resultante

é atribuído para a posição da câmera utilizando a função *Lerp* (Figura 18), que realiza a interpolação entre o vetor anterior e o atual, deixando a movimentação mais suave.

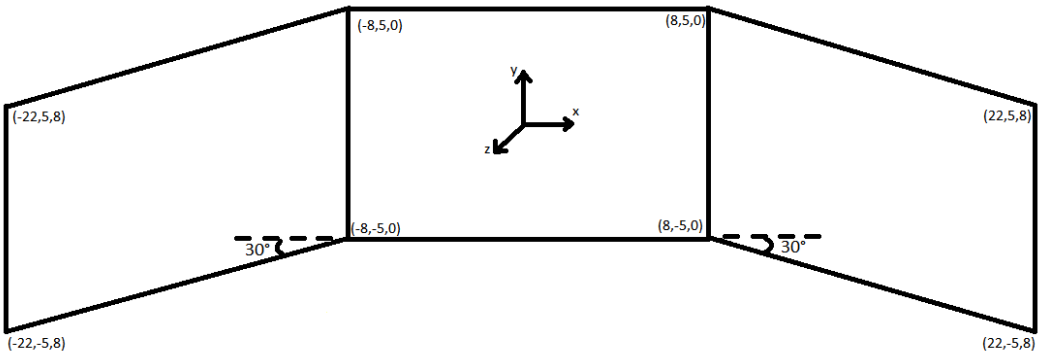

Figura 17 – Representação gráfica do MiniCAVE.

```
void Update(){
    transform.localPosition = Vector3.Lerp(transform.localPosition, KinectUIVA.headPosition, 0.05f);
}
```

Figura 18 – Aplicação da função *Lerp*.

4.3 Desenvolvimento de uma aplicação exemplo no Unity

Para demonstrar e exemplificar o movimento da paralaxe, uma aplicação foi feita através do motor de jogo Unity. Optou-se por realizar uma aplicação simples com vários blocos, ao invés de uma aplicação com paisagens elaboradas, por ser um exemplo mais prático de se perceber o movimento da paralaxe. Além dos blocos observados nas figuras, existe um objeto do tipo Câmera, que recebe a posição da cabeça do usuário de acordo com os cálculos já mostrados. A Figura 19 mostra a vista frontal da aplicação desenvolvida e a Figura 20, mostra a vista superior.

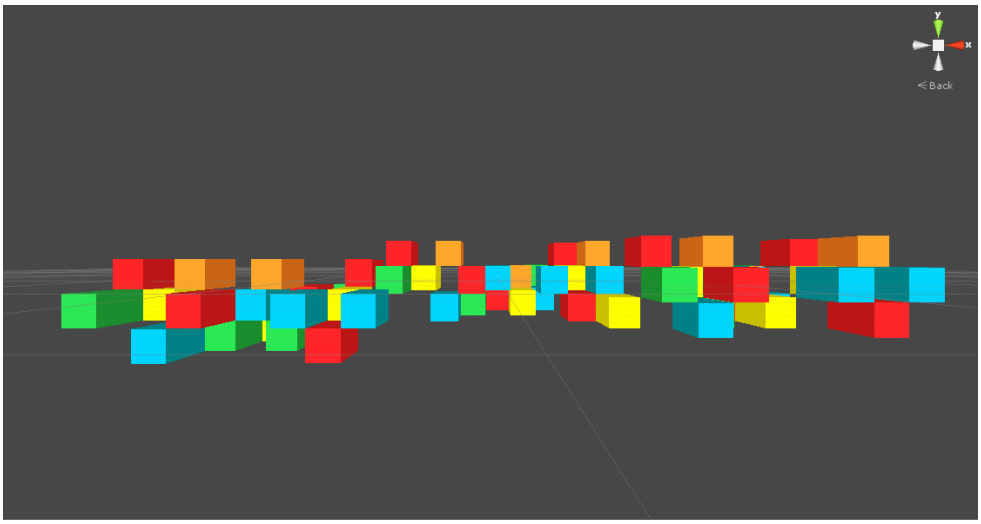

Figura 19 – Vista Frontal da Aplicação Desenvolvida.

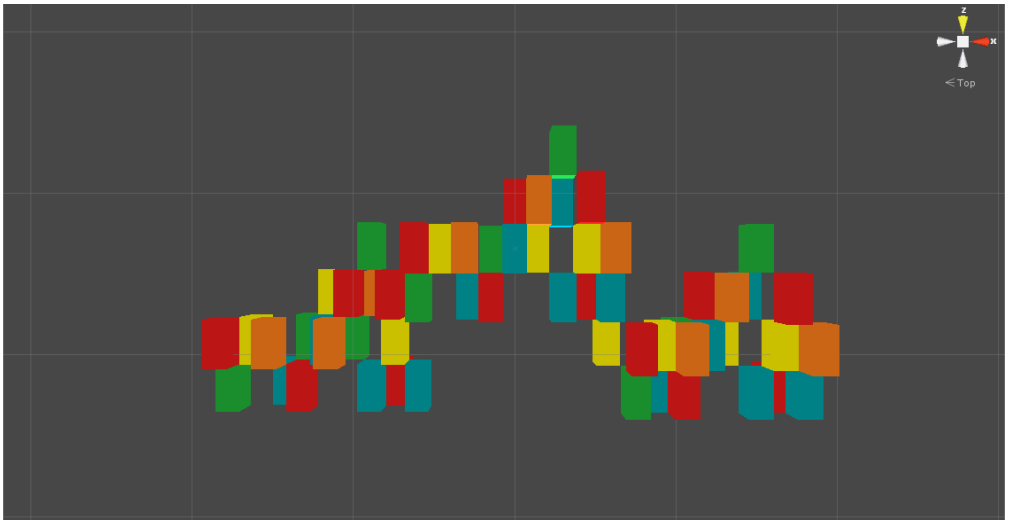

Figura 20 – Vista Superior da Aplicação Desenvolvida.

5 CONSIDERAÇÕES FINAIS E DISCUSSÃO DOS RESULTADOS

Baseado nos resultados obtidos e na literatura revisada e apresentada neste trabalho, foi concluído que é possível implementar o movimento da paralaxe utilizando o sensor Kinect e que o mesmo é capaz de fornecer os dados necessários para a realização deste movimento.

É possível ver o movimento da paralaxe funcionando no ambiente multiprojetado MiniCAVE através da aplicação exemplo criada.

A Figura 21, mostra um usuário observando a aplicação dos blocos de frente com a cena.

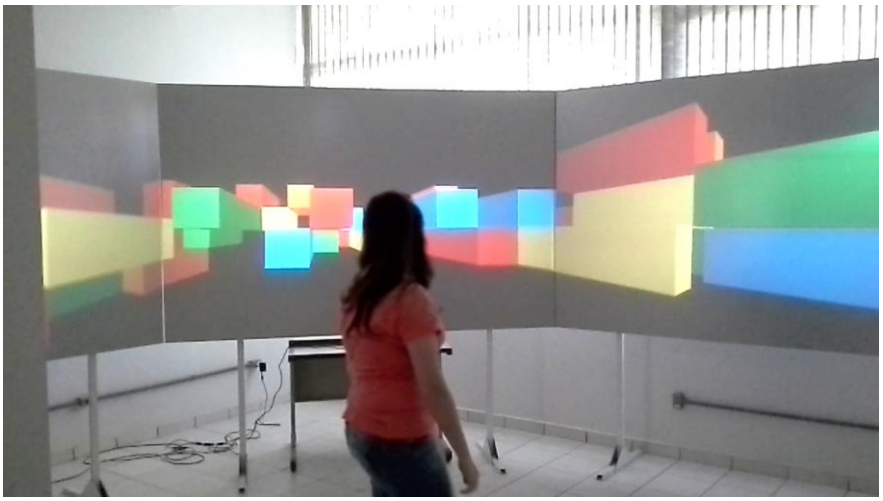

Figura 21 – Visão frontal dos blocos.

Na Figura 22, o usuário está olhando a cena pelo lado direito, tendo mais visão da lateral dos blocos. Na tela do lado direito, alguns blocos tiveram sua face frontal apagada, porque o usuário se aproximou deles e é como se ele estivesse olhando por dentro dos blocos.

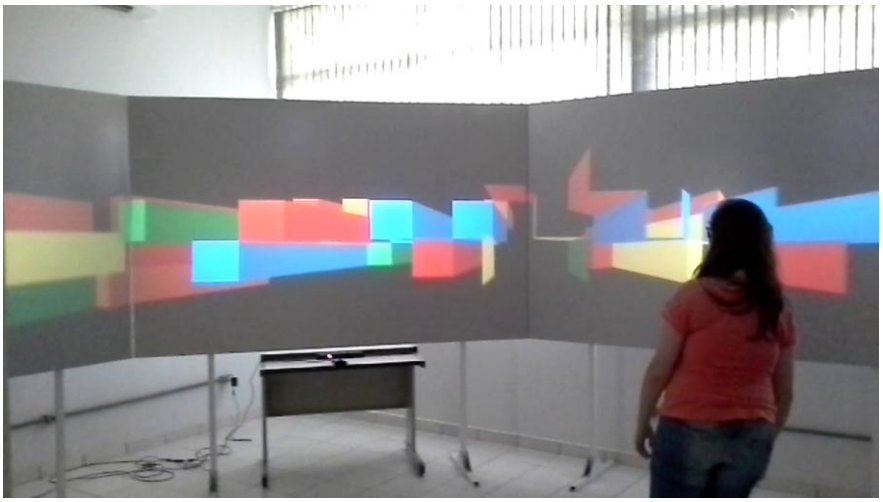

Figura 22 – Visão pela direita dos blocos.

Quando o usuário se abaixa, ele pode ver a face inferior dos blocos, como é demonstrado na Figura 23.

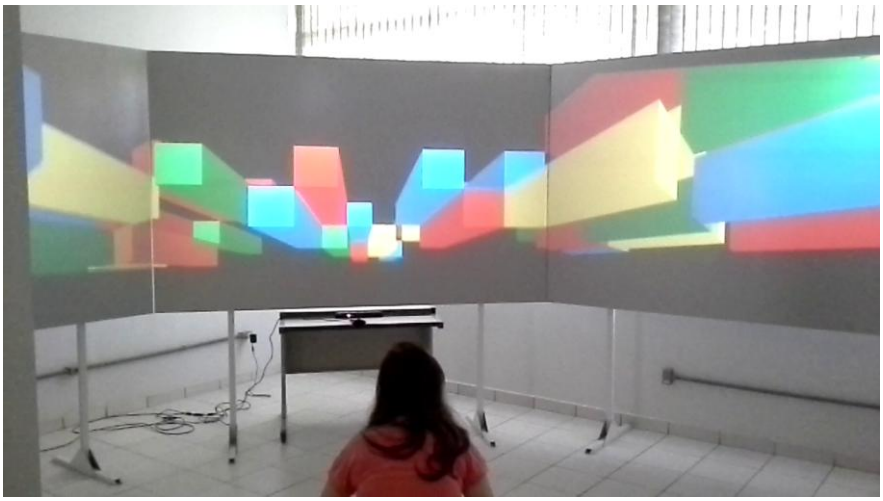

Figura 23 – Visão por baixo dos blocos.

Da mesma forma como foi demonstrado na Figura 22, na Figura 24, o usuário vê o lado esquerdo dos blocos.

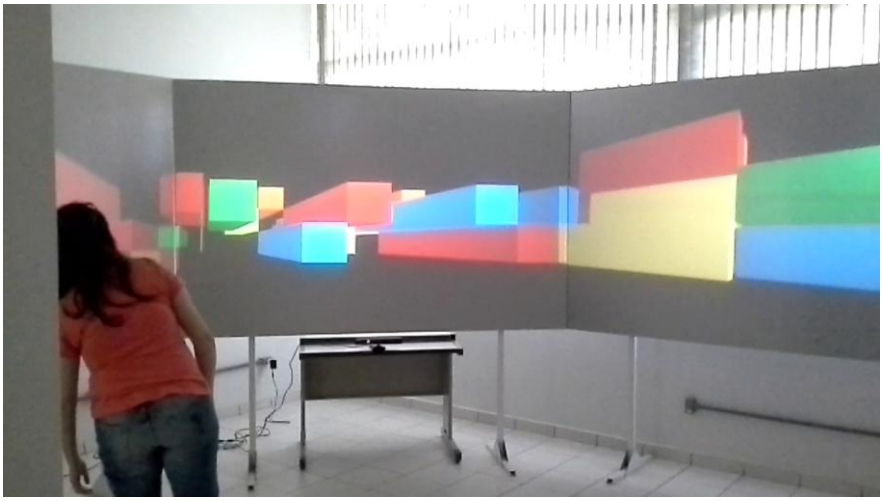

Figura 24 – Visão pela esquerda dos blocos.

O presente trabalho pode ser utilizado tanto para jogos que utilizam ambientes virtuais quanto para simuladores e até mesmo objetos de aprendizagem que possam utilizar ambientes virtuais.

REFERÊNCIAS

BEIMLER, R.; BRUDER, G.; STEINICKE, F. **Smurvebox: A smart multi-user real-time virtual environment for generating character animations.** In: *Proceedings of the Virtual Reality International Conference: Laval Virtual.* New York, NY, USA: ACM, 2013. (VRIC '13), p. 1:1–1:7. ISBN 978-1-4503-1875-4. Disponível em: <http://doi.acm.org/10.1145/2466816.2466818>.

BOUVIER, P.; SORBIER, F.; CHAUDEYRAC, P.; BIRI, V. **Cross benefits between virtual reality and games.** In: *Computer Games and Allied Technology 08, CGAT 08 - Animation, Multimedia, IPTV and Edutainment, Proceedings.* [s.n.], 2008. p. 186–193. Disponível em: <www.scopus.com>.

BROOKS, F. P. J. **What's real about virtual reality?** *Computer Graphics and Applications, IEEE,* v. 19, n. 6, p. 16–27, 1999. ISSN 0272-1716.

CRUZ-NEIRA, C.; SANDIN, D. J.; DEFANTI, T. A. **Surround-screen projection-based virtual reality: the design and implementation of the CAVE.** In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM, 1993. (SIGGRAPH '93), p. 135–142. ISBN 0-89791-601-8. Disponível em: <http://doi.acm.org/10.1145/166117.166134>.

DEFANTI, T. A.; DAWE, G.; SANDIN, D. J.; SCHULZE, J. P.; OTTO, P.; GIRALDO, J.; KUESTER, F.; SMARR, L.; RAO, R. **The StarCAVE, a third-generation CAVE and virtual reality OptIPortal.** *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 25, n. 2, p. 169–178, fev. 2009. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2008.07.015>.

ELLIS, S. **What are virtual environments?** *Computer Graphics and Applications, IEEE,* v. 14, n. 1, p. 17–22, 1994. ISSN 0272-1716.

FAAST. **Flexible Action and Articulated Skeleton Toolkit.** Jan. 2015. http://projects.ict.usc.edu/mxr/faast/

FILHO, M.; NEGRÃO, N.; DAMASCENO, R. **SwImax: A web tool using virtual reality for teaching the WiMAX protocol.** In: *Virtual Reality (SVR), 2011 XIII Symposium on.* [S.l.: s.n.], 2011. p. 217–224.

GREUTER, S.; ROBERTS, D. J. **Controlling Viewpoint from Markerless Head Tracking in an Immersive Ball Game Using a Commodity Depth Based Camera.** In: *2011 15th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications* 2011. p. 64-71.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. **Systematic literature reviews in software engineering - a systematic literature review.** *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 51, n. 1, p. 7–15, jan. 2009. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2008.09.009>.

LEWIS, M.; JACOBSON, J. **Game engines in scientific research.** *Commun.* ACM, ACM, New York, NY, USA, v. 45, n. 1, p. 27–31, jan. 2002. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/502269.502288>.

LUGRIN, J. L; WIEBUSCH, D.; STREHLER, A.; LATOSCHIK, M. E. **Usability Benchmarks for Motion Tracking Systems.** *In: Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology.* Singapore: ACM, 2013. p. 49-58. ISBN 978-1-4503-2379-6.

SHERMAN, W. R.; COMING, D.; SU, S. **FreeVR: honoring the past, looking to the future.** In: . [s.n.], 2013. v. 8649, p. 864906–864906–15. Disponível em: <http://dx.doi.org/10.1117/12.2008578>.

SHERMAN, W. R.; CRAIG, A. B. **Understanding virtual reality: Interface, application, and design.** [S.l.]: Morgan Kaufmann, 2002. 608 p. (The Morgan Kaufmann Series in Computer Graphics).

SOWNDARARAJAN, A.; WANG, R.; BOWMAN, D. A. **Quantifying the benefits of immersion for procedural training.** *In: Proceedings of the 2008 Workshop on Immersive Projection Technologies/Emerging Display Technologies.* New York, NY, USA: ACM, 2008. (IPT/EDT '08), p. 2:1–2:4. ISBN 978-1-60558-212-2. Disponível em: <http://doi.acm.org/10.1145/1394669.1394672>.

STAADT, O. G.; WALKER, J.; NUBER, C.; HAMANN, B. **A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering.** In: *Proceedings of the Workshop on Virtual Environments 2003.* New York, NY, USA: ACM,

2003. (EGVE ’03), p. 261–270. ISBN 1-58113-686-2. Disponível em: <http://doi.acm.org/10.1145/769953.769984>.

STEPTOE, W.; STEED, A.; SLATER, M. **Human Tails: Ownership and Control of Extended Humanoid Avatars.** In: *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 4, abr 2013, p. 583-590.

SUMA, E. A.; LANGE, B.; RIZZO, A. S.; KRUM, D. M.; BOLAS, M. **FAAST: The Flexible Action and Articulated Skeleton Toolkit.** In: *Vitrtual Reality Converence 2011*. Singapore: IEEE 2011, p. 247-248. ISBN 987-1-4577-0039-2.

SZILAS, N.; BARLES, J.; KAVAKLI, M. **An implementation of real-time 3d interactive drama.** *Comput. Entertain.*, ACM, New York, NY, USA, v. 5, n. 1, jan. 2007. ISSN 1544-3574. Disponível em: <http://doi.acm.org/10.1145/1236224.1236233>.

TAYLOR II, R. M.; HUDSON, T. C.; SEEGER, A.; WEBER, H.; JULIANO, J.; HELSER, A.T. **VRPN: A device-independent, network-transparent VR peripheral system.** In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. New York, NY, USA: ACM, 2001. (VRST ’01), p. 55–61. ISBN 1-58113-427-4. Disponível em: <http://doi.acm.org/10.1145/505008.505019>.

UIVA. **Unity Indie VRPN Adapter (UIVA).** Jan. 2015. http://web.cs.wpi.edu/~gogo/hive/UIVA

XBOX. **Kinect Effect.** Jan. 2015. http://www.xbox.com/pt-BR/Kinect/Kinect-Effect