UNIVERSIDADE ESTADUAL PAULISTA
“Júlio de Mesquita Filho”

Pós-Graduação em Ciência da Computação

Mário Popolin Neto

A Multiprojeção de Ambientes Virtuais Gerados por Motores de Jogo – O Histórico e o Design de uma Solução Genérica Aplicado no Motor de Jogo Unity

UNESP
2015

Mário Popolin Neto

A Multiprojeção de Ambientes Virtuais Gerados por Motores de Jogo – O Histórico e o Design de uma Solução Genérica Aplicado no Motor de Jogo Unity

Orientador: Prof. Dr. José Remo Ferreira Brega

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Computação Aplicada, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UNESP
2015

Popolin Neto, Mário.
A multiprojeção de ambientes virtuais gerados por motores de jogo : o histórico e o design de uma solução genérica aplicado no motor de jogo Unity / Mário Popolin Neto. -- São José do Rio Preto, 2015
77 f. : il., tabs.

Orientador: José Remo Ferreira Brega
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Sistemas de computação virtual. 3. Realidade virtual. 4. Jogos eletrônicos. I. Brega, José Remo Ferreira. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 681.31

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Mário Popolin Neto

A Multiprojeção de Ambientes Virtuais Gerados por Motores de Jogo – O Histórico e o Design de uma Solução Genérica Aplicado no Motor de Jogo Unity

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Computação Aplicada, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Trabalho aprovado. Bauru, 21 de agosto de 2015:

Prof. Adjunto José Remo Ferreira Brega
Faculdade de Ciências
Universidade Estadual Paulista – Bauru
Presidente da Banca

Prof. Associado Fernando Vieira Paulovich
Instituto de Ciências Matemáticas
e de Computação
Universidade de São Paulo – São Carlos

Prof. Adjunto Aparecido Nilceu Marana
Faculdade de Ciências
Universidade Estadual Paulista – Bauru

UNESP
2015

Dedico este trabalho a memória de meu avô Mário Popolin,
do qual tenho imenso orgulho de carregar o nome.

Agradecimentos

Agradeço a Deus, que é único e nós, homens, o interpretamos de várias maneiras.

Aos meus pais Cláudio Popolin e Maria Terezinha Sardinha Popolin, e irmãos Guilherme Sardinha Popolin e Cláudio Popolin Junior, que me apoiaram em mais uma etapa da minha carreira acadêmica, e que foram e sempre serão meu alicerce.

Ao Professor Remo (José Remo Ferreira Brega) por não somente me orientar no desenvolvimento deste trabalho, mas também pelos seis anos de amizade e ensinamentos diversos, que me transformaram como profissional e pessoa.

Aos amigos e colegas do antigo Laboratório de Sistemas de Tempo Real – LSTR e atual Laboratório de Interfaces e Visualização – LIV da UNESP Campus Bauru, de forma especial a Diego Roberto Colombo Dias, Izabele Andreoli Agostinho e Alessandro Campanhã Moraes, que contribuíram e me ajudaram no desenvolvimento deste trabalho.

Ao Professor José Roberto Pereira Lauris da USP Campus Bauru, pela aplicação dos testes estatísticos nos dados qualitativos coletados no experimento realizado neste trabalho.

Aos meus amigos Luiz Fernando Assis Lucita e Daniel Mori Fujita da turma de 2008 do Bacharelado em Ciência da Computação da UNESP Campus Bauru, com que mantive contato utilizado como apoio para o desenvolvimento deste trabalho, de forma especial ao Daniel, colega de quarto e paciente ouvinte dos momentos fáceis e principalmente dos difíceis desta etapa.

Ao canal do YouTube GearRecord [1], que dispõe de vídeo aulas quanto ao motor de jogo Unity, sendo estas utilizadas no aprendizado inicial desta ferramenta.

Ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UNESP, pela oportunidade, e a CAPES pelo apoio financeiro na modalidade bolsa, que sem dúvida foi essencial para o desenvolvimento deste trabalho.

A todos colegas, amigos e familiares não citados diretamente, mas que contribuíram e me apoiaram.

[1] https://www.youtube.com/user/GearRecord

“Eu gosto de computação gráfica, sabe qual o problema ?
Se você pegar uma roda, você demora um dia para fazer ela rodar.”
Sonilóquio do autor.

Resumo

Técnicas e tecnologias são utilizadas em aplicações de Realidade Virtual visando proporcionar uma melhor imersão ao usuário. A multiprojeção de Ambientes Virtuais é a técnica onde múltiplas projeções contínuas e sincronizadas de um Ambiente Virtual são renderizadas em tempo real, geralmente utilizando imagens estereoscópicas e a correção da perspectiva de acordo com o ponto de visão do usuário. As tecnologias apresentam certo grau de dinamismo, uma vez que grande parte é fomentada pelo mercado consumidor que está sempre atento as melhorias e novos recursos. Atualmente, o mercado de Jogos Digitais é um dos mais ativos. Com a disponibilização dos motores de jogo de empresas como Epic Games e Unity Technologies, as tecnologias empregadas na criação de jogos podem ser utilizadas por pesquisadores no desenvolvimento de aplicações com Ambientes Virtuais com propósitos diversos. Neste trabalho, as definições de Realidade Virtual e de motor de jogo são apresentadas, evidenciando as características destas ferramentas para criação de jogos que fazem com que pesquisadores as adotem para o desenvolvimento de suas aplicações e experimentos. Como contribuição principal, apresenta-se as soluções para o uso da multiprojeção em Ambientes Virtuais gerados por motores de jogo, apontando tendências para o design de uma solução genérica aplicada no motor de jogo Unity, dando origem ao pacote Unity Cluster Package. O Unity é um dos mais populares motores de jogo, amplamente utilizado por ser de fácil uso e multiplataforma, permitindo a criação de aplicações para consoles, PCs, navegadores web e dispositivos móveis como tablets e smartphones. O Unity Cluster Package tem como objetivo facilitar o desenvolvimento de aplicações de multiprojeção com o motor de jogo Unity, constituindo-se de componentes *drag-and-drop* bem definidos e altamente configuráveis. Este pacote Unity foi utilizado na criação de aplicações para um determinado sistema de multiprojeção. Dentre tais aplicações, tem-se a exploração imersiva de um Ambiente Virtual rico em detalhes, um jogo multijogador integrando plataformas distintas e a visualização imersiva e interativa de grandes volumes de dados representados como grafos 3D. A avaliação quantitativa das aplicações desenvolvidas com o Unity Cluster Package apresentou bons resultados, enquanto a avaliação qualitativa validou o sistema de multiprojeção utilizado. Com o Unity Cluster Package, aplica-se o design de uma solução genérica para o uso da técnica de multiprojeção com motores de jogo no motor de jogo Unity, uma das tecnologias atuais para o desenvolvimento de jogos.

Palavras-chave: Realidade Virtual. Motor de Jogo. Multiprojeção. Unity.

Abstract

Techniques and technologies are used in Virtual Reality applications aiming to provide a better immersion to the user. The multi-projection of Virtual Environments is a technique where multiple continuous and synchronized projections of a Virtual Environment are rendered in real time, usually using stereoscopic images and perspective correction according to the user's viewpoint. Technologies have certain degree of dynamism, since much of it is supported by the consumer market which is always on the lookout for improvements and new features. Currently, Digital Games market is one of the most active. With game engines availability from companies such as Epic Games and Unity Technologies, the game creation technology can be used by researchers in the development of applications with Virtual Environments for several purposes. In this work, the definitions of Virtual Reality and game engine are presented, pointing out features that make researchers adopt these game creation tools for the development of their applications and experiments. As the main contribution, the solutions to achieve multi-projection Virtual Environments generated by game engines are presented, indicating trends for a generic solution design that was applied in the Unity game engine, originating the Unity Cluster Package. Unity is one of the most popular game engines, widely used featuring user-friendly and multi-platform, allowing the applications creation for consoles, PCs, web browsers, and mobile devices such as tablets and smartphones. The Unity Cluster Package aims to facilitate the multi-projection applications development with the Unity game engine, providing well-defined and highly configurable drag-and-drop components. This Unity package was used in the applications creation for a specific multi-projection system. Among these applications, there are an immersive exploration of a high quality Virtual Environment, a multiplayer game integrating different platforms, and an immersive and interactive visualization application of large datasets represented as 3D graphs. The applications quantitative evaluation developed using the Unity Cluster Package presented good results, while the qualitative evaluation validated the used multi-projection system. With the Unity Cluster Package, the generic solution design to use the multi-projection technique with game engines is applied in the Unity game engine, one of the current technologies for game development.

Keywords: Virtual Reality. Game Engine. Multi-projection. Unity.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

2D	Duas Dimensões
3D	Três Dimensões
AG	Aglomerado Gráfico
API	*Application Programming Interface*
AV	Ambiente Virtual
CAD	*Computer Aided Design*
CAVE	*Cave Automatic Virtual Environment*
DLL	*Dynamic Link Library*
EVL	*Electronic Visualization Lab*
FAAST	*Flexible Action and Articulated Skeleton Toolkit*
FPS	*First Person Shooter*
FPS	*Frames Per Second*
HMD	*Head Mounted Display*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
JDBC	*Java Database Connectivity*
LCD	*Liquid Crystal Display*
NPC	*Non Playing Character*
PC	*Personal Computer*
RPC	*Remote Procedure Call*
RV	Realidade Virtual
SQL	*Structured Query Language*
UIVA	*Unity Indie VRPN Adapter*
UML	*Unified Modeling Language*

URL	*Uniform Resource Locator*
VA	Visualização Aumentada
VC	Visualização Científica
VET	*Virtual Experience Test*
VRPN	*Virtual Reality Peripheral Network*
WYSIWYG	*What You See Is What You Get*
XML	*eXtensible Markup Language*

Sumário

1 Introdução

Este Capítulo contém a introdução desta dissertação, apresentando suas considerações iniciais e organização, bem como os objetivos do trabalho.

1.1 Considerações Iniciais

A técnica de multiprojeção de Ambientes Virtuais foi apresentada por Cruz-Neira, Sandin e DeFanti (1993) em 1992, inovando ao criar um sistema imersivo que dispunha de quatro projeções de um mesmo Ambiente Virtual. O usuário pôde então estar fisicamente dentro do sistema, que o circundava com tais projeções dispostas em forma de cubo, como apresenta a Figura 1. A técnica, até então inovadora, visava proporcionar uma melhor imersão no cenário por meio do aumento do campo de visão, junto a geração de imagens estereoscópicas e a correção da perspectiva de acordo com o ponto de visão do usuário.

Figura 1 – O CAVE™: Quatro projeções sincronizadas e contínuas de um mesmo Ambiente Virtual dispostas em forma de cubo.

Fonte: Pape (2013).

Diferentes *hardwares* e *softwares* oferecem suporte a multiprojeção implementando abordagens distintas. Os *softwares* que suportam tal técnica são geralmente combinados com tecnologias de renderização, como *Application Programming Interface* (API) gráficas (OpenGL ou DirectX), motores gráficos (Ogre3D [1]) e motores de jogo. Os motores de jogo emergiram em 1999 proporcionando uma ótima relação entre interatividade e desempenho gráfico, levando os pesquisadores a explorar estas ferramentas para criação de jogos no desenvolvimento de aplicações para sistemas de multiprojeção (LUGRIN et al., 2012).

Dezesseis anos se passaram com a existência concomitante da técnica de multiprojeção e do motor de jogo. Durante este período, várias pesquisas foram desenvolvidas aplicando a técnica e suas abordagens com motores de jogo atuais, geralmente os presentes no pico de visibilidade da tecnologia da representação gráfica Gartner's Hype Cycle [2]. Tais pesquisas apresentam extensões para motores de jogo específicos e aplicações de multiprojeção em variados domínios, o que possibilita o estudo das tendências do uso da multiprojeção de Ambientes Virtuais gerados por motores de jogo, para se alcançar o design de uma solução genérica e aplicá-lo em um motor de jogo atual, para então explorar este em sistemas de multiprojeção.

Neste trabalho é apresentada uma revisão sistemática da literatura quanto as soluções empregadas na multiprojeção de Ambientes Virtuais gerados por motores de jogo. Por meio desta é possível determinar as abordagens de *hardware* e *software* mais utilizadas na aplicação da técnica de multiprojeção com as tecnologias de desenvolvimento de jogos. Para a abordagem mais difundida tem-se o design de uma solução genérica implementado no motor de jogo Unity, um dos mais populares motores de jogo utilizados na atualidade. Com esta implementação, originou-se o pacote Unity Cluster Package, permitindo o desenvolvimento de aplicações Unity variadas para um sistema de multiprojeção específico. Estas aplicações Unity multiprojetadas oferecem grande campo de visão, imagens estereoscópicas e correção da perspectiva de acordo com o ponto de visão do usuário.

1.2 Objetivos

Esta dissertação é resultante da pesquisa e do desenvolvimento científico com objetivos:

- **Objetivo Geral:** Realizar uma revisão sistemática da literatura sobre o uso da técnica de multiprojeção e suas abordagens com motores de jogo, buscando os domínios de aplicação onde a técnica e as tecnologias são utilizadas, bem como as tendências nos trabalhos científicos que exploram tais recursos.

[1] http://www.ogre3d.org/

[2] A representação gráfica Gartner's Hype Cycle apresenta a relação entre a visibilidade e maturidade de uma tecnologia ou aplicação. Composta por cinco fases, o pico de visibilidade se encontra na fase "Peak of Inflated Expectations", onde a publicidade e propaganda acarreta grande expectativa no mercado.

- **Objetivos Específicos:** i) Propor o design de uma solução genérica para a abordagem da técnica de multiprojeção mais utilizada com motores de jogo. ii) Aplicar o design de uma solução genérica em um motor de jogo atual, sendo então adotado o motor de jogo Unity. iii) Garantir o reuso de *software* da aplicação do design de uma solução genérica no motor de jogo Unity. iv) Desenvolver aplicações Unity para um determinado sistema de multiprojeção. v) Avaliar de forma quantitativa e qualitativa as aplicações multiprojetadas desenvolvidas e o sistema de multiprojeção utilizado.

1.3 Organização

Esta dissertação está organizada em sete capítulos. Além deste Capítulo 1, que apresenta as considerações iniciais e os objetivos geral e específicos do trabalho desenvolvido, tem-se os seis seguintes assim definidos:

- O Capítulo 2, **A Multiprojeção e os Motores de Jogo**, contém as definições de motor de jogo e de Realidade Virtual, sendo esta uma das áreas onde a técnica de multiprojeção é aplicada. Os resultados da revisão sistemática da literatura também estão presentes no Capítulo 2, bem como o design de uma solução genérica para a abordagem da técnica de multiprojeção mais utilizada.

- O Capítulo 3, **O Motor de Jogo Unity e o Unity Cluster Package**, apresenta o motor de jogo Unity, descrevendo as funcionalidades e recursos do Editor Unity e dos módulos Unity. O pacote Unity oriundo da aplicação da solução genérica do Capítulo 2 nesse popular motor de jogo, o Unity Cluster Package, também é apresentado no Capítulo 3.

- O Capítulo 4, **Aplicações Unity Multiprojetadas**, apresenta as aplicações Unity desenvolvidas utilizando o Unity Cluster Package para um sistema de multiprojeção. A utilização dos componentes *drag-and-drop* bem definidos e altamente configuráveis deste pacote Unity é descrita no Capítulo 4, bem como as avaliações quantitativas e qualitativas executadas para com as aplicações Unity multiprojetadas.

- O Capítulo 5, **Conclusão**, apresenta as conclusões com base nos resultados da revisão sistemática da literatura, nas caraterísticas do Unity Cluster Package e nos resultados das avaliações quantitativas e qualitativas das aplicações Unity multiprojetadas.

- O Capítulo 6, **Trabalhos Futuros**, sugere os trabalhos futuros quanto ao design da solução genérica para motores de jogo, ao Unity Cluster Package e as aplicações Unity multiprojetadas.

- O Capítulo 7, **Publicações Realizadas**, cita os textos científicos elaborados e publicados durante o desenvolvimento deste trabalho.

2 A Multiprojeção e os Motores de Jogo

Este Capítulo contém as definições de motor de jogo e de Realidade Virtual, e apresenta o sistema inovador que introduziu a técnica de multiprojeção para dar suporte a duas grandes áreas, sendo uma delas a própria Realidade Virtual. Os resultados da revisão sistemática da literatura, quanto as soluções para o emprego da multiprojeção em Ambientes Virtuais gerados por motores de jogo também são apresentados neste Capítulo, bem como o design de uma solução genérica para a abordagem da técnica de multiprojeção mais utilizada.

2.1 Definições

Segundo Steuer (1992), o termo Realidade Virtual (RV) foi cunhado em 1989 por Jaron Lanier. Porém, a ideia de RV foi apresentada por Ivan Sutherland em 1965, no *The Ultimate Display* (BROOKS, 1999). Vários autores definiram RV e, com isso, a diversificação de termos a ela associados, como Ambiente Virtual (BUCHINGER; JURASZEK; HOUNSELL, 2012).

Brooks (1999) definiu RV como uma experiência onde o usuário está efetivamente imerso em um mundo virtual, possuindo o controle dinâmico do ponto de visão neste mundo. De acordo com Sherman e Craig (2002), RV é composta por um mundo virtual interativo, onde o visualizador está imerso e este ambiente sintético tridimensional responde as suas ações. Para Kirner e Siscoutto (2007), a RV pode ser definida como “interface avançada do usuário”, onde o mesmo visualiza e interage em tempo real em ambientes tridimensionais gerados por computador.

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 um outro local. Por sua vez, Nichols, Haldane e Wilson (2000) definiram AV como modelos gerados por computadores com o qual os participantes interagem em tempo real.

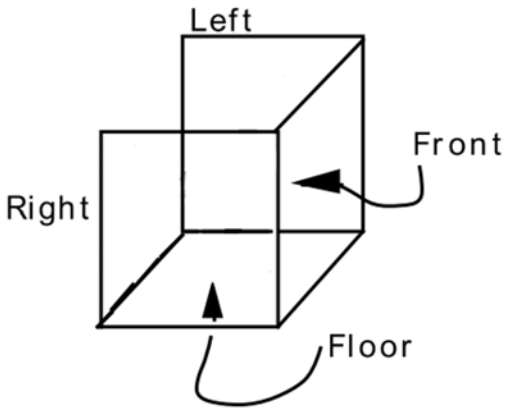
Há mais de duas décadas Cruz-Neira, Sandin e DeFanti (1993) apresentaram o Cave Automatic Virtual Environment (CAVE™), o sistema que introduziu a técnica de multiprojeção para dar suporte a aplicações de RV e Visualização Científica (VC). O CAVE™ apresenta um AV multiprojetado com base no ponto de visão do usuário e imagens estereoscópicas.

2.2 O CAVE™

Criado pelo laboratório *Electronic Visualization Lab* (EVL), o CAVE™ (Figura 1) teve sua primeira exibição no SIGGRAPH’92 e como objetivo não somente aplicações de RV, mas também de VC (CRUZ-NEIRA; SANDIN; DEFANTI, 1993).

O CAVE™ apresentado em 1992 era composto por quatro projeções (Figura 2), utilizando quatro estações Silicon Graphics, uma para cada tela. Controladores de áudio foram utilizados para o uso de múltiplos auto-falantes e por meio dos sensores Polhemus e Ascension, foi possível o rastreamento da posição da cabeça e da mão do usuário (CRUZ-NEIRA; SANDIN; DEFANTI, 1993).

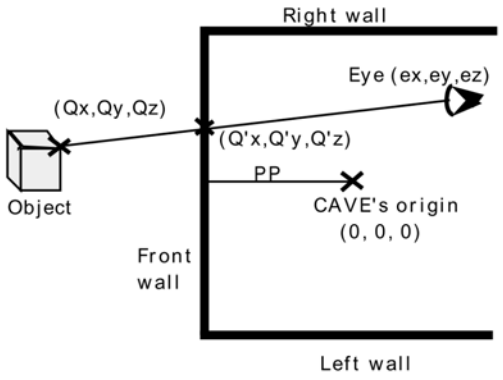
Figura 2 – As quatro telas do CAVE™ apresentado no SIGGRAPH'92.



Fonte: Cruz-Neira, Sandin e DeFanti (1993).

As imagens estereoscópicas no CAVE™ foram obtidas por meio de estereoscopia ativa, onde as estações Silicon Graphics geravam as projeções do olho direito e esquerdo de forma alternada e o usuário fazia uso de *shutter glasses*, a fim de fornecer a cada olho a sua respectiva projeção (CRUZ-NEIRA; SANDIN; DEFANTI, 1993). As projeções utilizadas no CAVE™ têm como especificação um ponto relativo ao plano de projeção, como o ponto Eye (ex, ey, ez) ilustrado na Figura 3, criando as projeções em perspectiva de acordo com o ponto especificado (CRUZ-NEIRA; SANDIN; DEFANTI, 1993).

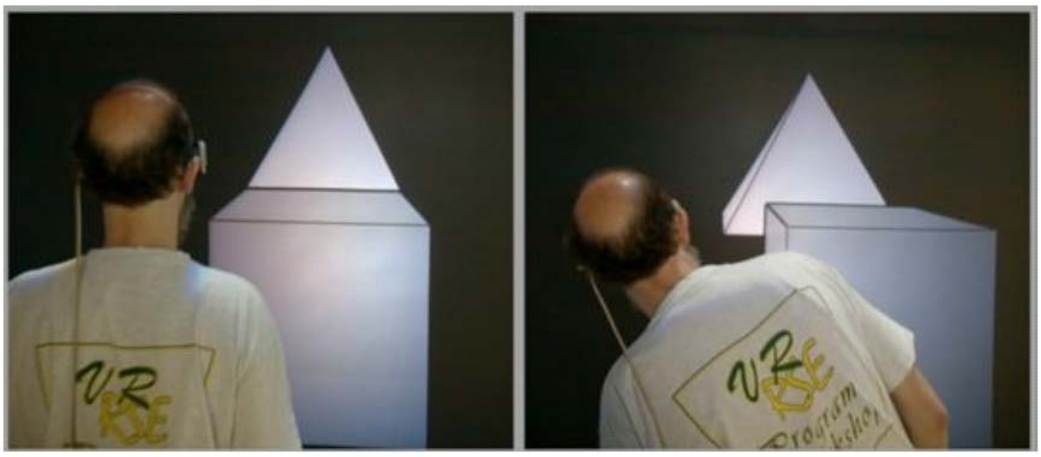
Figura 3 – Especificação do ponto de projeção.



Fonte: Cruz-Neira, Sandin e DeFanti (1993).

Tendo como ponto de especificação a posição da cabeça do usuário, Cruz-Neira, Sandin e DeFanti (1993) aplicam a correção da perspectiva do AV de acordo com o ponto de visão do usuário, implementando o movimento da paralaxe (movimentação da cabeça do visualizador), como apresenta a Figura 4.

Figura 4 – O movimento da paralaxe.



Fonte: Sherman, Coming e Su (2013).

Diversas são as tecnologias de renderização empregadas com a técnica de multiprojeção introduzida com o CAVE™. Segundo Lewis e Jacobson (2002), motores de jogo emergiram proporcionando maior interatividade e desempenho gráfico e, de acordo com Lugrin et al. (2012), estas características levaram pesquisadores a explorar motores de jogo em aplicações de RV, particularmente em sistemas de multiprojeção baseados no CAVE™.

2.3 Motor de Jogo

O termo motor de jogo, do inglês *game engine*, atingiu seu estado de maturidade no lançamento dos jogos Quacke III Arena e Unreal Tournament em 1999, sendo o motor de jogo fornecido com a cópia original do jogo e reutilizado em uma família de jogos, minimizando o alto custo da criação de simulações realísticas no desenvolvimento de jogos digitais (LEWIS; JACOBSON, 2002).

De acordo com Anderson et al. (2013), umas das mais precisas definições para motores de jogo é dada por Lewis e Jacobson (2002), onde os mesmos são códigos que suportam simulações que não especificam diretamente o comportamento (lógica) ou o ambiente do jogo. Para Anderson et al. (2013), motores de jogo fornecem a infraestrutura para a criação de jogos, não sendo somente um conjunto de componentes reutilizáveis mas uma camada que conecta tais componentes.

Segundo Anderson et al. (2013), os motores de jogo podem ser agrupados de acordo com sua arquitetura, dando origem a quatro grupos diferentes, os quais estão descritos na Tabela 1 apresentando suas características e alguns exemplos.

Tabela 1 – Os grupos de motores de jogo.

Grupo	Características	Exemplos
Monolítico	Alto nível de abstração, incluindo ferramentas que reduzem o esforço gasto na criação de jogos. Não fornece controle de baixo nível.	UDK [9]e Crossbow (ANDERSON et al., 2013).
Framework Modular	Possui um número distinto de módulos que fornecem subsistemas ao motor de jogo. Fornece controle de baixo nível (subsistemas).	Source Engine [10], CryEngine [11], Unity [12], id Tech 2 [13], e ZFXCE [14].
Componente Modular	Possui os componentes necessários para o desenvolvimento de jogos, enquanto a lógica do jogo, incluindo o ciclo do jogo, são mantidos fora do motor de jogo.	ZFX Engine [15]e XNA [16].
Compacto	Motores de jogo mais leves, com características limitadas e uma arquitetura de baixa complexidade.	SAGE (PARBERRY et al., 2007) e ACK-3D (ANDERSON et al., 2013)

Fonte: Popolin Neto e Brega (2015b).

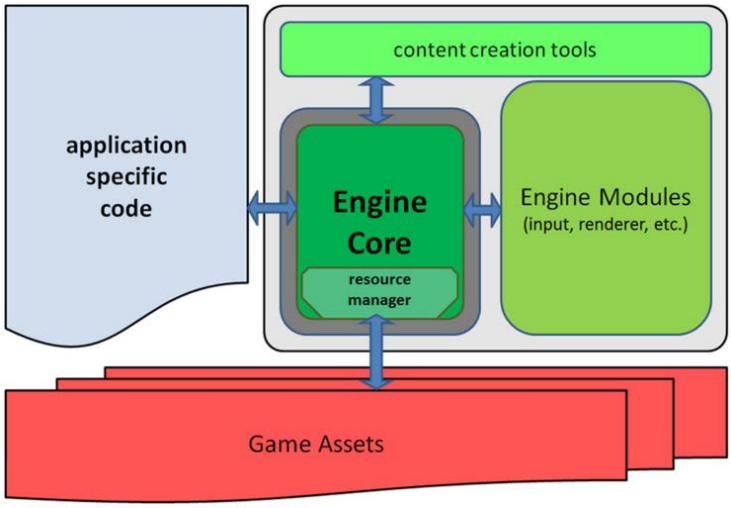
A maioria dos motores de jogo comerciais de alta qualidade se encontram no grupo Framework Modular, onde módulos bem definidos fornecem subsistemas ao núcleo do motor de jogo (ANDERSON et al., 2013). A Figura 5 ilustra a representação típica de um motor de jogo pertencente ao grupo Framework Modular.

Embora exigindo mais esforço para a criação de jogos, quando comparados com os do grupo Monolítico, os motores de jogo que acomodam e gerenciam módulos distintos oferecendo controle de baixo nível fornecem uma boa relação entre customização e complexidade, sendo adequados para o desenvolvimento de aplicações de RV. Segundo Petridis et al. (2010), existem sete módulos comuns entre os motores de jogo atuais, o quais estão contidos na Tabela 2.

A RV pode fazer uso dos métodos e recursos de jogos digitais, uma vez que as duas áreas possuem muitas características em comum. Ambas exploram os avanços tecnológicos de diversos campos, como síntese de imagem e desenvolvimento de dispositivos de interação. Mundos virtuais realísticos, como em um simulador de voo, ou fantasiosos, que não respeitam as leis da física, estão presentes em jogos digitais e RV (BOUVIER et al., 2008).

[9] https://www.unrealengine.com
[10] http://www.valvesoftware.com/
[11] http://www.crytek.com/cryengine
[12] http://unity3d.com/
[13] http://www.idsoftware.com/
[14] http://sourceforge.net/projects/zfxce/
[15] https://github.com/CodeRedd/zfx-engine-zerbst-duvel
[16] https://msxna.codeplex.com/

Figura 5 – A representação típica de um motor de jogo pertencente ao grupo Framework Modular.



Fonte: Anderson et al. (2013).

Tabela 2 – Os módulos comuns entre os motores de jogo atuais.

Módulo	Descrição
Gráfico	Responsável pela geração dos gráficos 2D/3D do ambiente, incluindo bibliotecas para mapeamentos de texturas, sombreamento, iluminação, efeitos de *shader*, etc.
Física	Permite que os objetos se comportem de acordo com as leis da física, tendo como exemplo a queda de objetos sobre a força da gravidade.
Colisão	Habilita que certas ações sejam executadas quando dois objetos colidem.
Entrada/Saída	Responsável pelos dispositivos de entrada e saída que podem ser usados pelo motor de jogo.
Som	Incorpora 2D/3D som a fim de aumentar o realismo da cena.
Inteligência Artificial (IA)	Utilizado para criar objetos ou jogadores não controlados pelo jogador, de modo a interagir de uma maneira inteligente com o mesmo.
Rede	Responsável pela implementação de jogos de multijogadores, permitindo que vários jogadores participem simultaneamente do jogo.

Fonte: Popolin Neto e Brega (2015b).

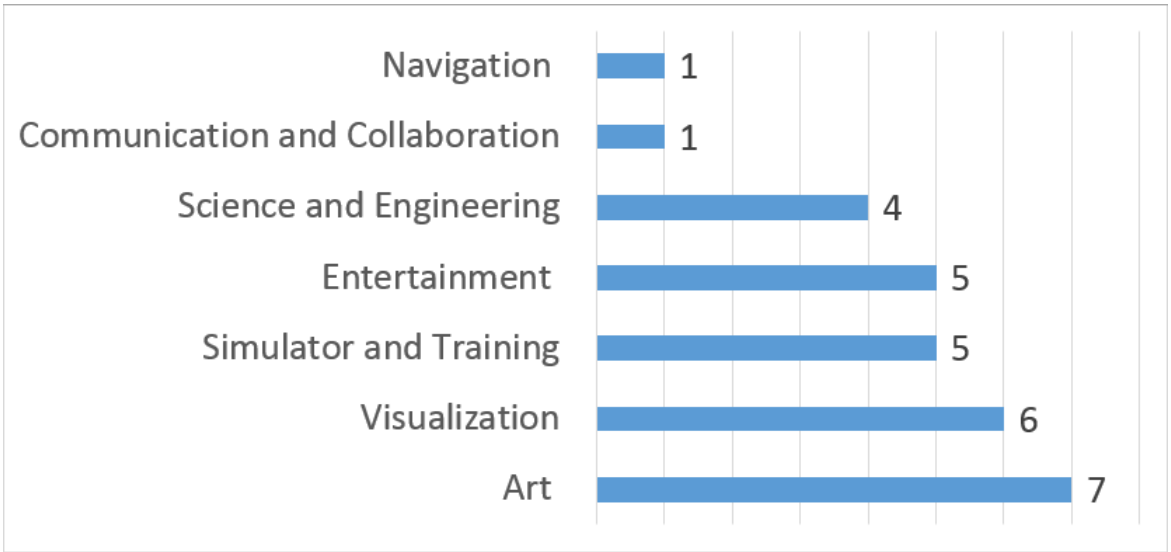
Apesar do emprego de motores de jogo no desenvolvimento de aplicações de RV, estas ferramentas não fornecem originalmente suporte para sistemas de RV, como sistemas de multiprojeção, e dispositivos de interação comumente utilizados em aplicações de RV, como os fornecidos por servidores que utilizam o padrão *Virtual Reality Peripheral Network* (VRPN) desenvolvido por Taylor II et al. (2001).

2.4 Os Motores de Jogo e Sistemas de Multiprojeção

Desenvolvido por Paul Rajlich em 1998, o CAVE Quake II [17] foi provavelmente o primeiro jogo digital em um sistema de multiprojeção (LUGRIN et al., 2012), estendendo o jogo Quake II para um sistema de multiprojeção baseado no CAVE™, onde o jogador no sistema imersivo pôde interagir com jogadores remotos em *Personal Computers* (PC), plataforma de origem do Quake II (JACOBSON; HWANG, 2002).

Os motores de jogo vêm sendo utilizados em sistemas de multiprojeção para propósitos diversos. A revisão sistemática da literatura, com o planejamento e a condução descritas no Apêndice A, resultou em quarenta trabalhos científicos que apresentaram a técnica de multiprojeção em AVs gerados por motores de jogo, dentre os quais vinte e nove utilizam estes AVs imersivos em aplicações para um domínio específico. A Figura 6 apresenta as vinte e nove aplicações classificadas fazendo uso dos domínios de aplicação empregados na pesquisa de Takala, Rauhamaa e Takala (2012).

Figura 6 – As vinte e nove aplicações desenvolvidas com motores de jogo para sistema de multiprojeção dentre os domínios de aplicação.



Fonte: Popolin Neto e Brega (2015b).

A maioria das aplicações multiprojetadas se encontram no domínio Arte, onde qualidade gráfica e implementação das leis da física podem ser utilizadas juntamente a conceitos de arte para alcançar a causalidade em mundos virtuais imersivos (CAVAZZA et al., 2005). As aplicações no domínio Visualização habilitam a visualização imersiva de representações em três dimensões (3D), como projetos *Computer Aided Design* (CAD) (FILHO et al., 2009).

[17] http://www.visbox.com/prajlich/caveQuake/

No domínio Simulador e Treinamento, aplicações utilizam a imersão em cenários realísticos que simulam condições de alto risco, como no simulador para treinamento de bombeiros apresentado na Figura 7. Com relação ao domínio Entretenimento, as aplicações são geralmente jogos oferecendo experiências imersivas de entretenimento.

Figura 7 – Um simulador para treinamento de bombeiros.

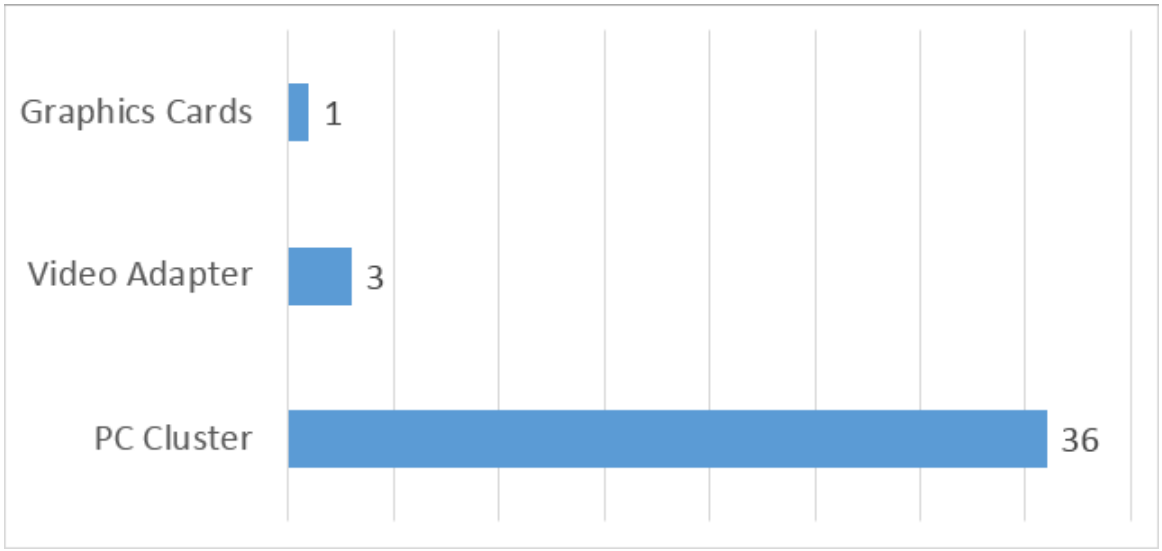
Fonte: Backlund et al. (2007).

O domínio Ciência e Engenharia contém aplicações para o teste de novos métodos em sistemas imersivos. Os domínios Navegação, e Comunicação e Colaboração apresentam uma aplicação. Em geral, as aplicações imersivas classificadas fornecem:

- Projeções sincronizadas e contínuas de AVs;
- Imagens estereoscópicas por meio de estereoscopia passiva ou ativa; e
- Movimento da paralaxe utilizando a correção da perspectiva de acordo com o ponto de visão do usuário.

Três abordagens de *hardware* diferentes são encontradas com relação ao sistema de multiprojeção: adaptador de vídeo, placas gráficas e Aglomerado Gráfico (AG) do inglês *PC Cluster*. A Figura 8 apresenta os quarenta trabalhos científicos que utilizam a técnica de multiprojeção em AVs gerados por motores de jogo em relação as três abordagens de *hardware*.

Figura 8 – Os quarenta trabalhos científicos que utilizam a técnica de multiprojeção em AVs gerados por motores de jogo em relação as três abordagens de hardware.



Fonte: Popolin Neto e Brega (2015b).

2.4.1 Adaptador de Vídeo e Placas Gráficas

Adaptadores que dividem e distribuem o sinal de vídeo, tal como os produtos Matrox Graphics [18] DualHead2Go e TripleHead2Go (Figura 9), habilitam a adição de dois ou três monitores ou projetores a uma única saída de vídeo de uma placa gráfica.

Figura 9 – O adaptador de vídeo Matrox TripleHead2Go.

Fonte: Matrox Graphics Inc (2015).

Sistemas de multiprojeção podem ser construídos utilizando adaptadores de vídeo. Apresentado por Abramyan, Powell e Norris (2012), Stage faz uso do Matrox TripleHead2Go, acomodando três projeções rotacionadas em 90° para alcançar um ambiente de visualização de 270°, onde voluntários interagem entre si em um jogo colaborativo desenvolvido com o motor de jogo Ardor3D.

[18] http://www.matrox.com/graphics/en/products/gxm/

O adaptador Matrox DualHead2Go é utilizado no Virtual Reality Theatre, oriundo do trabalho de Schou e Gardner (2009), que elaboraram um método para o uso de múltiplas barras de sensores, aumentando e melhorando a área de cobertura do controle Wii Remote™, utilizado em um jogo desenvolvido com motor de jogo Source Engine.

Uma vez que placas gráficas podem possuir mais de uma saída de vídeo, também é possível a existência de sistemas de multiprojeção baseados em placas gráficas em um único PC, como o simulador de direção de automóveis desenvolvido por Ni, Zhao e Zhang (2009) utilizando o motor de jogo Delta3D, no qual duas placas gráficas em um único PC operam quatro monitores *Liquid Crystal Display* (LCD), dois monitores para cada placa.

As abordagens adaptador de vídeo e placas gráficas são baseadas em *hardware*, o que não requer o suporte de *software* para a multiprojeção. Entretanto, adaptadores de vídeo aumentam a área ocupada de cada *pixel* resultando em perda de resolução de imagem (STAADT et al., 2003), e em um único PC gerenciando múltiplas placas gráficas e executando o processo de simulação pode haver sobrecarga de *hardware*.

2.4.2 Aglomerado Gráfico

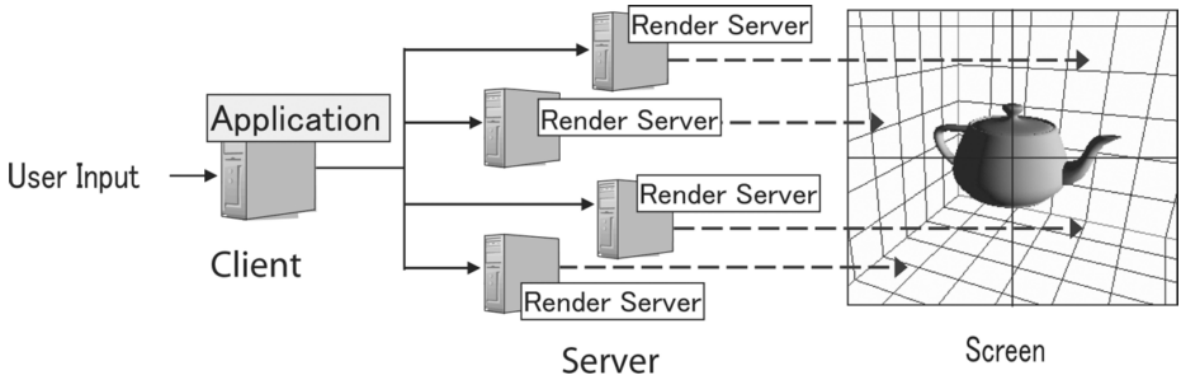
O AG é a abordagem mais utilizada quando se trata de multiprojeção em AVs gerados por motores de jogo, no qual um conjunto de PCs é conectado por uma rede local com o propósito de renderização. Dois modelos emergiram com relação ao processamento de dados compartilhados e distribuição das informações de renderização em AG: Mestre-Escravo e Cliente-Servidor (STAADT et al., 2003).

O modelo Cliente-Servidor (Figura 10) consiste de uma aplicação executada no nó cliente que distribui primitivas de renderização (OpenGL ou DirectX) para cada nó servidor, dependendo de sua atribuição no ambiente de múltiplas telas. A aplicação do nó cliente também é responsável pelas interações de usuário (STAADT et al., 2003). No Mestre-Escravo (Figura 11), todos os nós (mestre e escravos) executam a mesma aplicação com diferentes configurações de câmera virtual, onde a aplicação do nó mestre sincroniza as aplicações dos nós escravos e trata as interações de usuário (STAADT et al., 2003).

O desenvolvimento de aplicações para o modelo de renderização Cliente-Servidor é baseado em uma única execução (nó cliente) sobre uma solução de *software* para distribuição das primitivas gráficas. O Mestre-Escravo requer suporte de *software* para o modelo de comunicação em rede Cliente-Servidor, para que a aplicação do nó mestre (servidor de comunicação) sincronize as aplicações de nó escravo (clientes de comunicação) por meio das técnicas de Framelock e Datalock (RAFFIN et al., 2006).

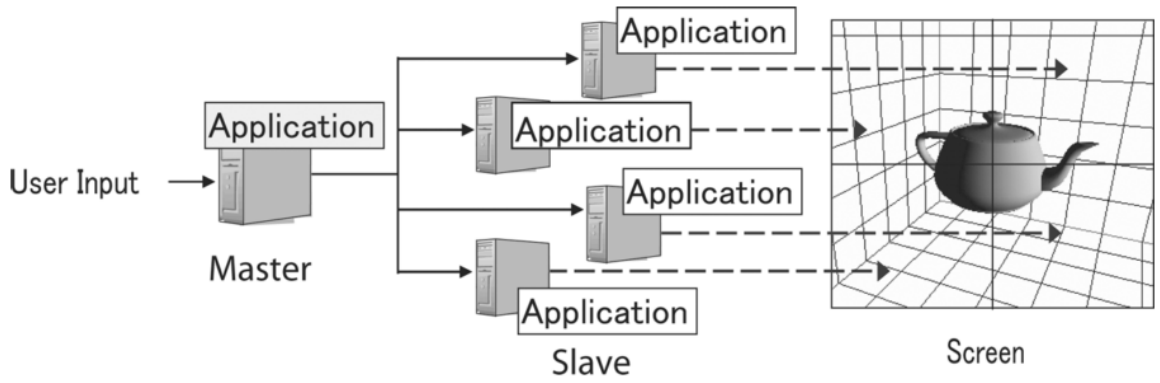
Bibliotecas de RV podem suportar um ou ambos modelos de renderização em AG, requerendo a integração com soluções externas para renderização do AV (SHERMAN; COMING; SU, 2013).

Figura 10 – O modelo de renderização em AG Cliente-Servidor.



Fonte: Hashimoto, Ishida e Sato (2007).

Figura 11 – O modelo de renderização em AG Mestre-Escravo.



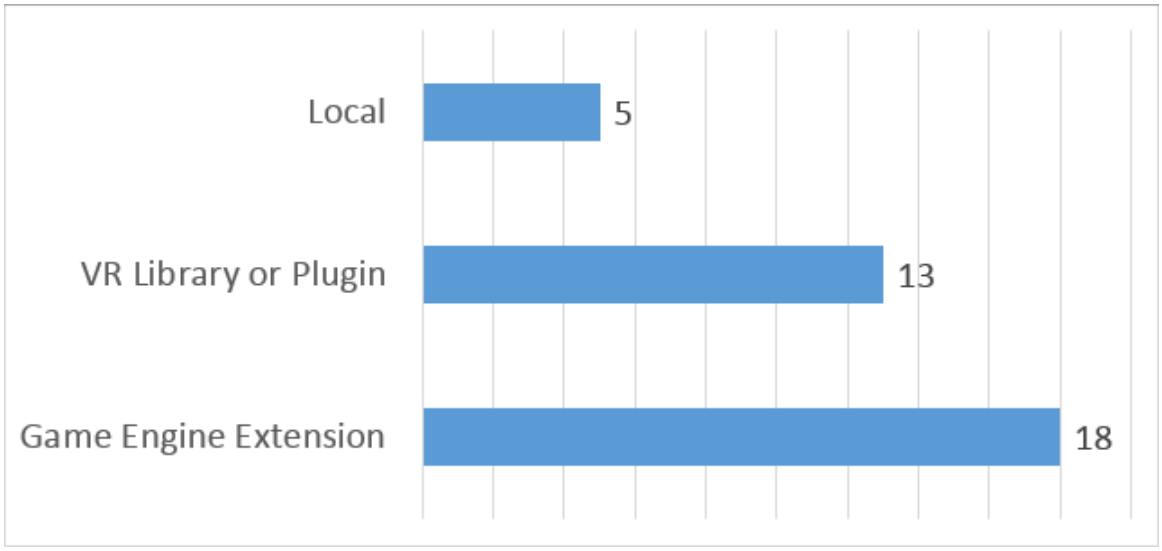
Fonte: Hashimoto, Ishida e Sato (2007).

Alguns autores integram bibliotecas ou *plugins* de RV e motores de jogo com o objetivo de desenvolver aplicações para sistemas de multiprojeção sobre um AG, outros usam extensões dos próprios motores de jogo ou implementam soluções locais. A Figura 12 apresenta os trinta e seis trabalhos científicos contendo a multiprojeção de AVs gerados por motores de jogo utilizando AG em relação as abordagens de suporte de *software*.

Dentre as bibliotecas de RV, a FreeVR desenvolvida por Sherman, Coming e Su (2013) e a D-vision [19] são as mais utilizadas; quanto a *plugins* de RV, o MiddleVR [20] é o mais adotado. As extensões desenvolvidas para motores de jogo oferecem suporte a sistemas de multiprojeção baseados em AGs utilizando o modelo de renderização Mestre-Escravo, o que apresenta melhor média de *Frames Per Second* (FPS) e baixa latência de rede quando comparado com o modelo Cliente-Servidor (STAADT et al., 2003).

[19] http://dvisiontek.com/index.html
[20] http://www.middlevr.com/middlevr-sdk/

Figura 12 – Os trinta e seis trabalhos científicos contendo a multiprojeção de AVs gerados por motores de jogo utilizando AG em relação as abordagens de suporte de software.



Fonte: Popolin Neto e Brega (2015b).

Criado por Jacobson e Hwang (2002), o CaveUT™ foi a primeira extensão para um motor de jogo, habilitando o desenvolvimento de diversas aplicações multiprojetadas com o motor de jogo da Epic Games Unreal® Engine, sendo o clássico motor de jogo por trás do jogo Unreal Tournament. CaveUDK é o sucessor natural do CaveUT™, uma vez que essa extensão desenvolvida por Lugrin et al. (2012) utiliza uma versão mais recente do motor de jogo Unreal® Engine.

CryVE é a extensão para o motor de jogo CryEngine® 2, objetivando um sistema de multiprojeção de baixo custo (JUAREZ; SCHONENBERG; BARTNECK, 2010). RUIS é um pacote Unity para a criação de *interfaces* espaciais de interação para aplicações Unity multiprojetadas (TAKALA, 2014). A Tabela 6 que contém uma comparação entre as extensões existentes para motores de jogo é apresentada na seção 3.3.

O uso do suporte de rede do motor de jogo para implementar o modelo de renderização Mestre-Escravo é uma tendência entre as extensões CaveUT™, CaveUDK e CryVE, e também é encontrada na implementação local de Louloudi e Klugl (2011). Tal tendência se caracteriza na utilização de altas taxas de sincronização de objetos do AV entre as aplicações de nó mestre e as de nó escravo a fim de se alcançar coerência de imagem, em vez de se utilizar as técnicas de Framelock e Datalock comumente implementadas por bibliotecas de RV.

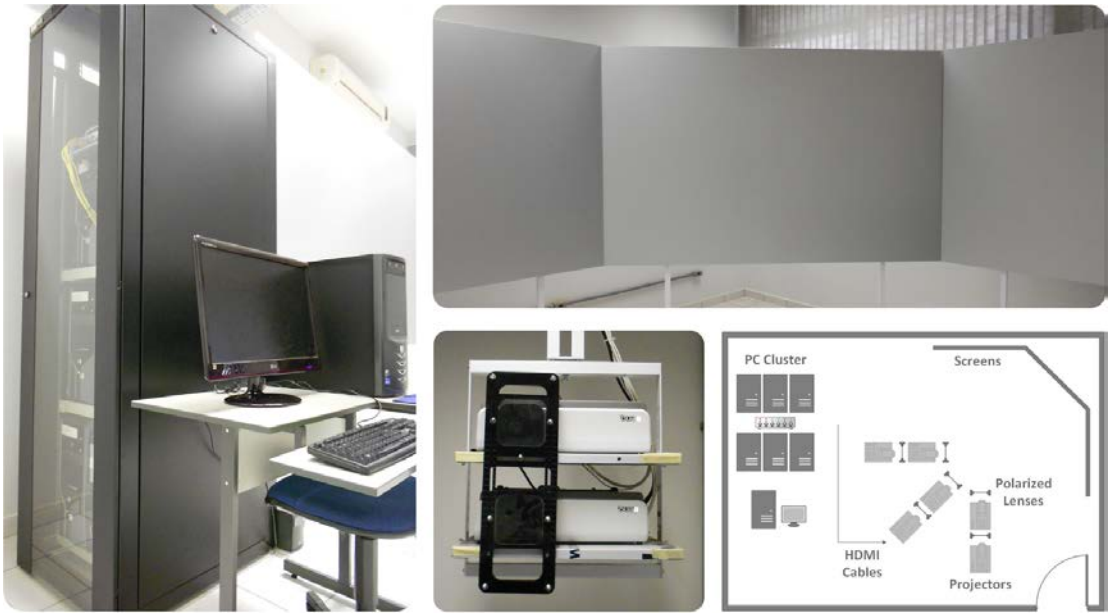
A sincronização de objetos sob altas taxas de atualização por meio do suporte de rede do motor de jogo é o que difere as extensões de bibliotecas de RV, sendo essa a principal característica do uso da técnica de multiprojeção em AVs gerados por motores de jogo.

2.5 O Design de uma Solução Genérica

Baseado nas tendências do emprego da técnica de multiprojeção e suas abordagens em AVs gerados por motores de jogo é apresentado neste trabalho o design de uma solução genérica, que utilizando o modelo de renderização Mestre-Escravo atende sistemas de multiprojeção baseados em AG, como o MiniCAVE. Desenvolvido por Dias et al. (2012) para a visualização de estruturas dentárias, o MiniCAVE também foi utilizado em aplicações de visualização de moléculas e de grande volume de dados (DIAS et al., 2011; MORAES; ELER; BREGA, 2014).

O MiniCAVE é um sistema de multiprojeção de baixo custo com três telas (2.5m x 1.5m cada) dispostas entre si em um ângulo de 30°, como apresenta a Figura 13. Para cada tela, tem-se dois projetores (BenQ W1000 HD) com lentes polarizadas conectados a dois PCs dedicados (Intel Core i7 8GB RAM e NVIDIA FX 1800), dando suporte a estereoscopia passiva. Os seis PCs de renderização (Linux Kubuntu 10.04 64bit) estão conectados a um PC servidor (Windows 7 32bit) por meio de um switch (Gigabit 3Com), que pode ser conectado a um roteador *wireless* (300Mbps TP-Link) de modo a habilitar a conexão de dispositivos móveis.

Figura 13 – O MiniCAVE: AG (sete PCs e um switch), três telas e seis projetores com lentes polarizadas.



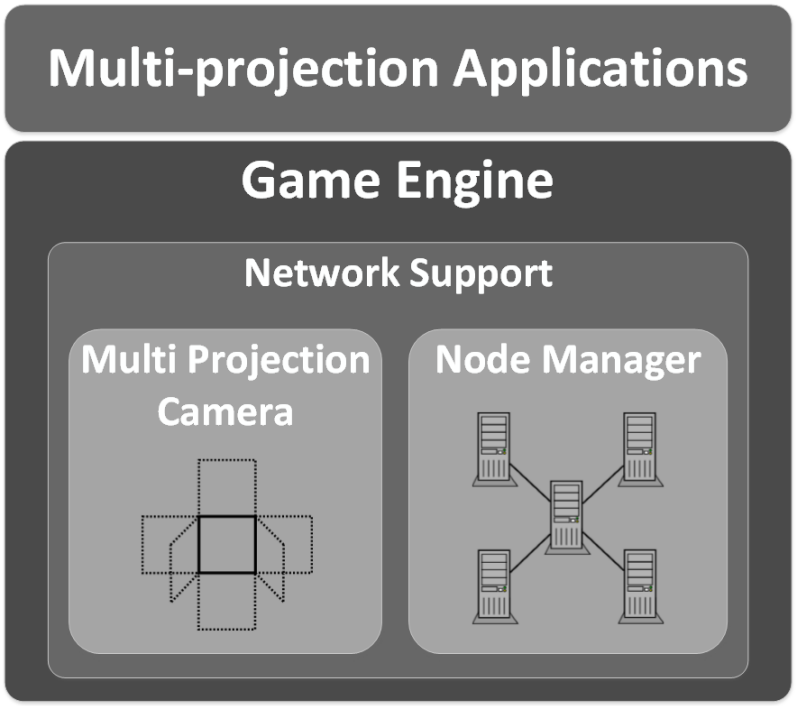
Fonte: Popolin Neto et al. (2015a).

O principal objetivo do design de uma solução genérica elaborado é permitir a criação de extensões para motores de jogo de forma padronizada, baseada no estado atual da literatura quanto ao uso da técnica de multiprojeção com motores de jogo, atendendo a abordagem mais utilizada e viabilizando as características comuns presentes em aplicações multiprojetadas criadas com motores de jogo. Para tal, o design deve atender ao modelo de renderização em AG Mestre-Escravo e suportar:

- Projeções sincronizadas e contínuas de AVs;
- Imagens estereoscópicas por meio de estereoscopia passiva ou ativa; e
- Movimento da paralaxe utilizando a correção da perspectiva de acordo com o ponto de visão do usuário.

Dois componentes bem definidos e altamente configuráveis são necessários para o desenvolvimento de aplicações com motores de jogo para sistemas de multiprojeção: *Multi Projection Camera* e *Node Manager*. Ambos são baseados no suporte de rede do motor de jogo para o modelo de comunicação em rede Cliente-Servidor. A Figura 14 ilustra estes dois componentes.

Figura 14 – Os dois componentes necessários para o desenvolvimento de aplicações com motores de jogo para sistemas de multiprojeção: *Multi Projection Camera* e *Node Manager*.



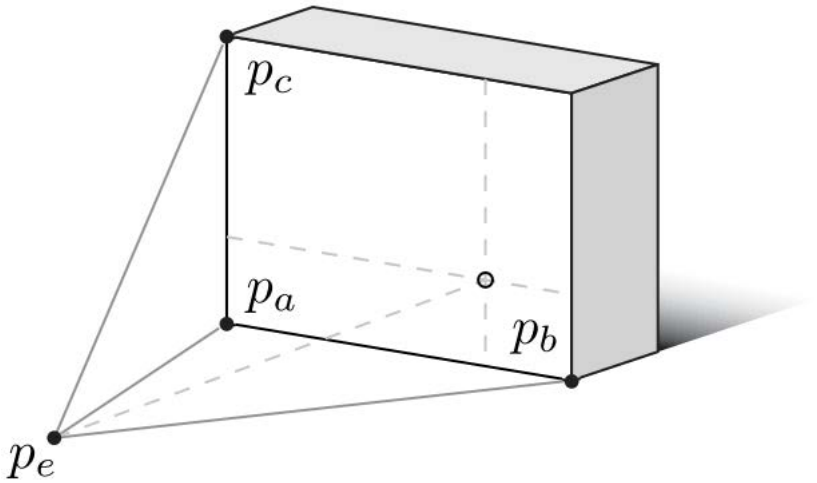
Fonte: Popolin Neto e Brega (2015b).

2.5.1 Multi Projection Camera

O componente *Multi Projection Camera* é uma câmera virtual configurável. Esta câmera virtual especial instanciada na aplicação de nó mestre é replicada em todas aplicações de nó escravo. A posição e orientação do componente *Multi Projection Camera* devem ser alteradas somente na aplicação de nó mestre e sincronizadas para as aplicações de nó escravo, obtendo projeções sincronizadas e contínuas.

Uma matriz de projeção customizada é necessária para a atribuição das aplicações de nó escravo a diferentes telas e para o cálculo da correção das perspectiva dado um ponto qualquer. Kooima (2015) apresenta a matriz de projeção de perspectiva generalizada utilizada em projetos do EVL, como o CAVE™ original e o Varrier™ desenvolvido por Sandin et al. (2005). Com esta matriz, os pontos Pa, Pb e Pc ilustrados na Figura 15 definem tamanho, proporção, posição e orientação da tela. Para o cálculo da perspectiva, tem-se como referência o ponto Pe, permitindo a geração automática e de forma correta de imagens estereoscópicas (projeções para o olho direito e esquerdo).

Figura 15 – Os pontos utilizados para o cálculo da matriz de projeção de perspectiva generalizada: Pa, Pb, Pc e Pe.



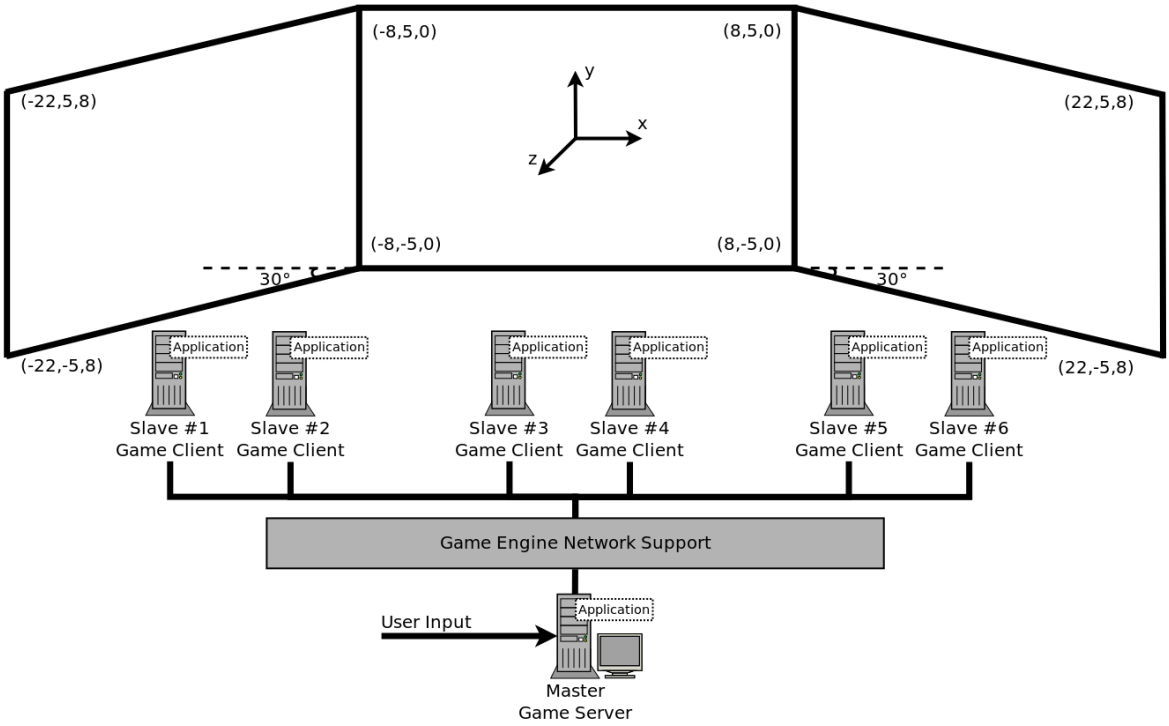
Fonte: Adaptado de Kooima (2015).

O componente *Multi Projection Camera* deve implementar a matriz de projeção de Kooima (2015) em uma função *PreRender()*, ou seja, em um método executado periodicamente antes da renderização de cada *frame*. O componente *Multi Projection Camera* também deve agrupar e tomar um objeto de cena denominado *UserHead* como ponto Pe. Movendo o objeto *UserHead* de acordo com qualquer dispositivo de rastreamento da posição da cabeça do usuário, tem se o movimento da paralaxe.

2.5.2 Node Manager

O componente *Node Manager* inicializa a aplicação de acordo com o tipo do nó, onde cada aplicação de nó escravo (Game Client) é conectada na aplicação de nó mestre (Game Server), como ilustra a Figura 16. Este componente instancia o *Multi Projection Camera* na aplicação de nó mestre utilizando o suporte de rede do motor de jogo, permitindo a replicação desta câmera virtual especial em todas as aplicações de nó escravo. A fim de se obter coerência de imagem, a posição e orientação do componente *Multi Projection Camera* devem ser sincronizadas mantendo altas taxas de sincronização.

Figura 16 – A disposição das aplicações no design de uma solução genérica utilizando o sistema de multiprojeção MiniCAVE.



Fonte: Popolin Neto e Brega (2015b).

O componente *Node Manager* deve obter as especificações do nó do AG por meio de um arquivo de configuração no padrão *eXtensible Markup Language* (XML), contendo os pontos Pa, Pb, Pc e Pe (com origem no centro da tela frontal), tipo do nó (mestre ou escravo) e endereço IP e porta de conexão para o Game Server. A aplicação de nó escravo Slave #1 (Figura 16) deve ser atribuída para a tela esquerda com Pa = (-22,-5,8), Pb = (-8,-5,0) e Pc = (-22,5,8). A fim de se obter imagens estereoscópicas por meio de estereoscopia passiva, os nós escravos Slave #1 e Slave #2 endereçam a projeção para a mesma tela mas para olhos diferentes: o nó Slave #1 para o olho esquerdo (eye = "left") e o nó Slave #2 para o olho direito (eye = "right"). A Figura 17 apresenta o arquivo de configuração para o nó escravo Slave #2.

2.6 Considerações Finais

As definições de motor de jogo e de RV foram apresentadas neste Capítulo, bem como o sistema CAVE™, que introduziu a técnica de multiprojeção para o suporte de aplicações de RV e VC. Interatividade e desempenho gráfico são as características oferecidas por motores de jogo que levaram pesquisadores a explorar estas ferramentas para criação de jogos em sistemas de multiprojeção baseados no CAVE™, utilizando diferentes abordagens de *hardware* e *software* que suportam tal técnica.

Figura 17 – O arquivo de configuração para o nó escravo Slave #2.

```
<node type="slave">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="true" eye="right">
    <pa x="-22" y="-5" z="8"/>
    <pb x="-8"  y="-5" z="0"/>
    <pc x="-22" y="5"  z="8"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

Fonte: Popolin Neto e Brega (2015b).

Os resultados da revisão sistemática da literatura apontaram as tendências do emprego da multiprojeção em AVs gerados por motores de jogo, para o design de uma solução genérica para a abordagem da técnica de multiprojeção mais utilizada, sendo esta o uso de AG com o modelo de renderização Mestre-Escravo. As aplicações multiprojetadas desenvolvidas com motores de jogo se encontram em maioria nos domínios Arte, Visualização, Simulador e Treinamento, e Entretenimento.

O design de uma solução genérica apresentado tem como objetivo permitir a criação de extensões para motores de jogo, para o desenvolvimento de aplicações para sistemas de multiprojeção de forma clara e objetiva, sem necessitar de soluções de *software* externas como bibliotecas ou *plugins* de RV, uma vez que se utiliza o próprio suporte de rede do motor de jogo para o modelo de comunicação em rede Cliente-Servidor.

3 O Motor de Jogo Unity e o Unity Cluster Package

Este Capítulo apresenta o motor de jogo Unity, descrevendo as funcionalidades e os recursos do Editor Unity e dos módulos Unity, que fazem do mesmo um dos mais populares motores de jogo utilizados na atualidade. O Unity Cluster Package também é apresentado neste Capítulo, sendo este oriundo da aplicação do design de uma solução genérica do Capítulo 2 no motor de jogo Unity.

3.1 O Motor de Jogo Unity

Com sua primeira versão lançada em 2005, o Unity é um motor de jogo completo, com motor de renderização customizado (Directx ou OpenGL) e uma ferramenta com *workflows* intuitivos. Sendo por origem um motor de jogo proprietário, desenvolvido e mantido pela Unity Technologies [1], o Unity possui uma versão *free* disponibilizada desde outubro de 2009 (CARTER; RHALIBI; MERABTI, 2010).

A documentação e a comunidade de desenvolvedores são duas características positivas apontadas por vários autores quanto ao motor de jogo Unity. Segundo Craighead, Burke e Murphy (2008), a documentação completa com exemplos de toda API Unity é um dos maiores benefícios deste motor de jogo.

Por meio do Unity é possível desenvolver jogos para mais de dez plataformas, como para Windows, Linux, Mac OS, iOS, Android, consoles e navegadores web (Unity Technologies, 2015). Por ser multiplataforma, Craighead (2008), Koenig et al. (2011), Almeida, Rossetti e Coelho (2013) escolheram o Unity, entre outros motores de jogo, para desenvolver aplicações com AVs interativos.

Pode-se destacar ainda as funcionalidades oferecidas pelo Unity Editor, como a alteração de parâmetros do AV em tempo real de execução e a fácil importação de modelos 3D, texturas, sons entre outros recursos (Unity Technologies, 2015).

Unindo ampla documentação, editor de fácil uso e a possibilidade de se desenvolver não somente jogos, mas também aplicações diversas para múltiplas plataformas, o motor de jogo Unity se tornou um dos mais populares motores de jogo, sendo amplamente utilizado em diversas áreas, como em jogos 3D, RV e Web3D (HU; QU; ZHANG, 2012).

[1] https://unity3d.com/company

3.1.1 O Editor Unity

O motor de jogo Unity permite o desenvolvimento de AVs por meio de um editor *What You See Is What You Get* (WYSIWYG), possibilitando a criação destes dispondo os recursos (modelos 3D, texturas, sons, entre outros) através do *drag-and-drop* (Unity Technologies, 2015). A *interface* do Unity Editor (Figura 18) é composta por painéis altamente configuráveis quanto a tamanho e posição, permitindo a customização e definição de vários *layouts*. A *interface* do Unity Editor pode ser descrita de forma funcional, abordando seus principais painéis, como mostra a Tabela 3.

Figura 18 – O Editor Unity.

Fonte: Unity Technologies (2015).

O Editor Unity oferece pacotes que podem ser facilmente importados e utilizados para o desenvolvimento de AVs realistas. A Tabela 4 apresenta os pacotes Unity padrão disponíveis para a versão *free*. O Editor Unity permite também a criação e exportação de pacotes customizados (extensão de arquivo .unitypackage). Depois de importados, os recursos dos pacotes ficam disponíveis no painel Project para serem usados na construção do AV no painel Scene.

A lógica dos objetos de cena no AV é construída por meio de *scripts*, podendo ser codificados em três linguagens diferentes: UnityScript (semelhante a JavaScript), C# ou Boo.

Tabela 3 – Os principais painéis da *interface* do Editor Unity.

Painel	Descrição
Project	Por meio deste painel, pode-se acessar e gerenciar os recursos pertencentes ao projeto Unity, localizados no diretório escolhido ao se criar o projeto, apresentando tais recursos e as árvores de diretórios nos quais estes estão armazenados.
Scene	Onde são criadas os AVs, dispondo os recursos disponíveis no painel Project por meio do "arrastar e soltar". Pode-se navegar neste painel de forma livre fazendo uso de teclado e mouse, visualizando e editando o posicionamento dos objetos dentro do AV.
Hierarchy	Apresenta todos os objetos contidos no AV, ou seja, os dispostos no painel Scene. Por meio deste, é possível o agrupamento de objetos, criando um sistema de coordenadas local para os objetos filhos com origem na posição global do objeto pai.
Game	Permite a visualização e interação com a execução do que vem sendo criado no painel Scene.
Toolbar	Consiste em controles básicos, como opções de navegação e posicionamento dos objetos para o painel Scene, e botões para execução, pausa e quadro por quadro para o painel Game.
Inspector	Ao selecionar qualquer objeto no painel Project, Scene ou Hierarchy, é apresentado no painel Inspector os componentes (malhas, textura, sons, animações, *scripts*, entre outros) do objeto selecionado. Pode-se alterar as propriedades dos componentes apresentados, visualizando tais alterações na edição do AV no painel Scene, ou em tempo execução no painel Game.

Fonte: Popolin Neto et al. (2015).

O Unity faz uso do Mono, uma implementação *open source* da plataforma .NET e os *scripts* são editados na *Integrated Development Environment* (IDE) Mono Develop. Porém, pode-se utilizar editores de código fonte externos, como por exemplo, o Microsoft Visual Studio [2]. A Figura 19 apresenta a estrutura básica de um *script* Unity em C#.

Figura 19 – Um exemplo básico de um *script* Unity em C#.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {

    void Start () {

    }

    void Update () {

    }
}
```

Fonte: Popolin Neto et al. (2015).

[2] http://www.visualstudio.com/

Tabela 4 – Os pacotes disponíveis no Editor Unity para a versão *free*.

Pacote	Descrição
Character Controller	Fornece controladores para personagens em primeira e terceira pessoa.
Light Cookies	Habilita efeitos para emissores de luz.
Light Flares	Disponibiliza efeitos de refração da luz dentro da lente da câmera.
Particles	Contém efeitos de partícula como fogo, fumaça, quedas d'água, faíscas, fogos de artifícios e flocos de neve.
Physic Materials	Permite adicionar fricção e robustez aos colisores dos objetos.
Projectors	Habilita a projeção de determinado material sobre objetos, como sombras e marcas de tiros de armas de fogo.
Scripts	Contém *scripts* como para os movimentos de câmera virtual.
Skyboxes	Dispõem de um conjunto de texturas panorâmicas para serem dispostas atrás de todos os objetos do AV a fim de se representar o céu.
Standard Assests (Mobile)	Contém *scripts* para o uso de telas sensíveis ao toque disponíveis em dispositivos com o sistema iOS ou Android.
Terrain Assets	Fornece recursos para a criação de terrenos, como texturas, árvores, arbustos e gramas.
Tree Creator	Permite a criação (modelagem) de árvores dentro do Unity Editor.
Water (Basic)	Habilita dois tipos de objeto água, diurno e noturno, para a criação de rios, lagos e oceanos.

Fonte: Popolin Neto et al. (2015).

Todo *script* Unity é derivado da classe base *MonoBehaviour*, fornecendo por herança acesso à variáveis e funções executados em momentos específicos, utilizados a fim de se inicializar e alterar tais variáveis, que representam as propriedades e componentes do objeto no qual o *script* está aplicado.

O *script* apresentado na Figura 19 contém as funções *Start()*, executado quando o objeto é criado no AV e *Update()*, executado a cada *frame* renderizado, onde são comumente implementados os comportamentos (lógica) dos objetos. Por meio do mecanismo prefab, o Unity Editor permite armazenar objetos e seus componentes (como por exemplos *scripts*), de modo a serem reutilizados por meio do *drag-and-drop* na criação do AV ou por referência em tempo de execução utilizando *scripts*.

Ainda por meio do Unity Editor, pode-se requisitar o acesso a documentação da API Unity em um navegador web e obter outros recursos de livre uso ou mediante pagamento no Unity Asset Store [3], como modelos 3D, animações, projetos completos (demos Unity), tutoriais e extensões.

[3] https://www.assetstore.unity3d.com/en/

3.1.2 Os Módulos do Motor de Jogo Unity

Anderson et al. (2013) classificam o motor de jogo Unity, com seu ambiente de desenvolvimento integrado, como pertencente ao grupo Framework Modular, onde se tem um número distinto de módulos que fornecem subsistemas ao motor de jogo, apresentando uma ótima relação entre customização e complexidade. A Tabela 5 apresenta as características dos módulos do motor de jogo Unity.

Tabela 5 – Os módulos do motor de jogo Unity.

Módulo	Descrição
Gráfico	Oferece um motor de renderização customizado, DirectX ou OpenGL, dependendo da plataforma selecionada. Também são oferecidos quarenta tipos de efeitos *shader*, bem como a opção da criação de novos efeitos.
Física e Detecção de colisão	Implementa propriedades da física como massa, gravidade, inércia e colisão por meio do motor de física PhysX da NVIDIA [6].
Entrada/Saída	Não somente oferece suporte a dispositivos convencionais usados em jogos (teclado, mouse, joystick e gamepad), mas também a telas sensíveis ao toque, acelerômetro e posição geográfica de dispositivos móveis.
Som	Utiliza o do motor de som FMOD [7], oferecendo funcionalidades como modelos customizados de atenuação por distância, filtros básicos e efeitos Doppler.
Inteligência Artificial	Fornece o sistema 'NavMesh', utilizado na implementação da movimentação dos *Non Playing Characters* (NPC) dentro do AV.
Rede	Implemtenta o modelo de comunicação em rede Cliente-Servidor, onde Clientes Unity conectados a um Servidor Unity podem se comunicar de acordo com a dinâmica do jogo.

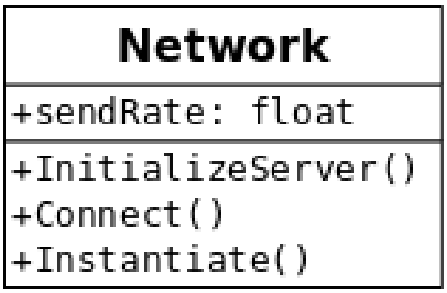
Fonte: Popolin Neto et al. (2015).

Por meio do módulo de rede do motor de jogo Unity, é relativamente simples criar jogos que permitem a interação entre jogadores remotos. Tem-se como ideia fundamental o relacionamento entre Cliente (aplicação que requer informações) e Servidor (aplicação que envia as informações). Utilizando-se do padrão *Unified Modeling Language* (UML), a Figura 20 contém uma representação simplificada da classe *Network*, sendo esta a classe principal do módulo de rede do motor de jogo Unity.

A classe *Network* permite inicializar o servidor ou se conectar a um existente por meio das funções *InitializeServer()* e *Connect()*. Para criar objetos de cena, em todas as instâncias do jogo (clientes) conectadas no servidor, tem-se a função *Instantiate()*. Os objetos da cena que precisam enviar dados e serem sincronizados, dentre as instâncias do jogo (clientes) conectadas no servidor, devem conter o componente NetworkView. A Figura 21 apresenta as funcionalidades do componente NetworkView.

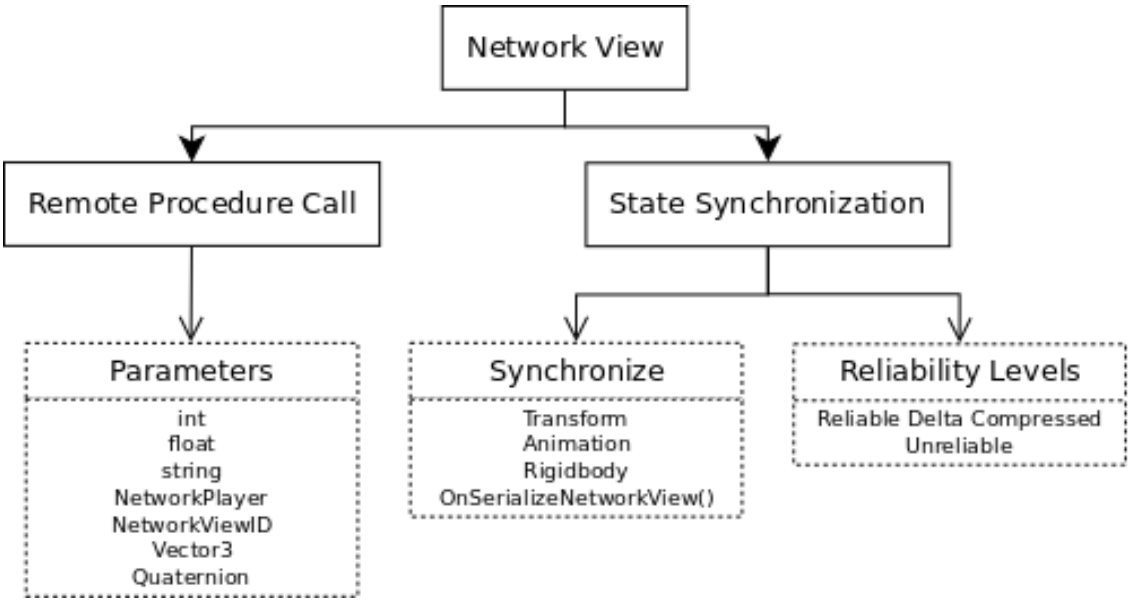
[6] http://www.nvidia.com/
[7] http://www.fmod.org/

Figura 20 – A classe *Network* do módulo de rede no motor de jogo Unity.



Fonte: Produzida pelo autor.

Figura 21 – O componente NetworkView do módulo de rede no motor de jogo Unity.



Fonte: Popolin Neto et al. (2015b).

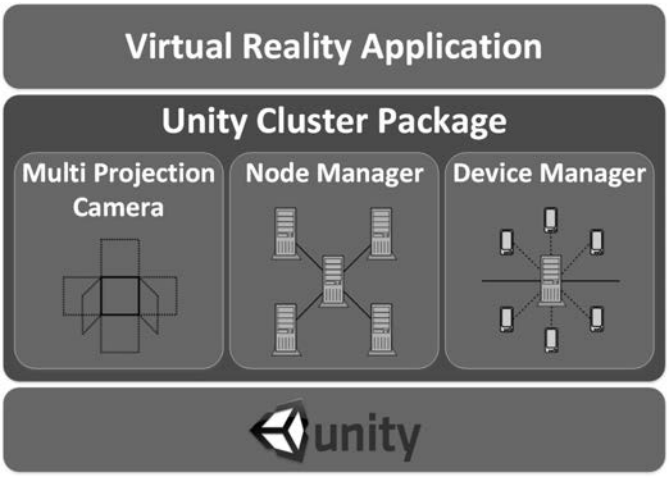
Uma determinada instância do jogo pode executar funções em outras instâncias de maneira remota através de *Remote Procedure Call* (RPC). Tais funções podem conter parâmetros enviados da instância que requisita a execução e são executadas na ordem em que são recebidas as requisições.

As propriedades (Transform, Animation e Rigidbody) dos objetos da cena podem ser sincronizadas, entre as instâncias do jogo, de maneira a verificar se todas as instâncias receberam o estado alterado (Reliable Delta Compressed) ou sem a verificação de recebimento de todo o estado (Unreliable). Pode-se ainda customizar a sincronização de outras propriedades fazendo uso da função *OnSerializeNetworkView()*. O atributo *sendRate* da classe *Network* contém o número de vezes que o estado dos objetos são sincronizados por segundo.

3.2 Unity Cluster Package

O Unity Cluster Package é a implementação do design de uma solução genérica apresentado no Capítulo 2 para o motor de jogo Unity. Este pacote Unity customizado (extensão de arquivo .unitypackage) tem como objetivo facilitar o desenvolvimento de aplicações de RV por meio de componentes *drag-and-drop*. A Figura 22 ilustra o Unity Cluster Package.

Figura 22 – O Unity Cluster Package.



Fonte: Popolin Neto et al. (2015a).

O Unity Cluster Package é facilmente importado no Editor Unity e permite o desenvolvimento de aplicações de RV imersivas e interativas por meio de seus componentes, que juntos oferecem suporte ao modelo de renderização em AG Mestre-Escravo, à imagens estereoscópicas via estereoscopia passiva e a servidores de dispositivos no padrão VRPN. Cada componente é um prefab Unity que possui objetos e *scripts* C# de acordo com sua funcionalidade. O Unity Cluster Package possui três componentes: *Multi Projection Camera*, *Node Manager* e *Device Manager*.

3.2.1 Multi Projection Camera

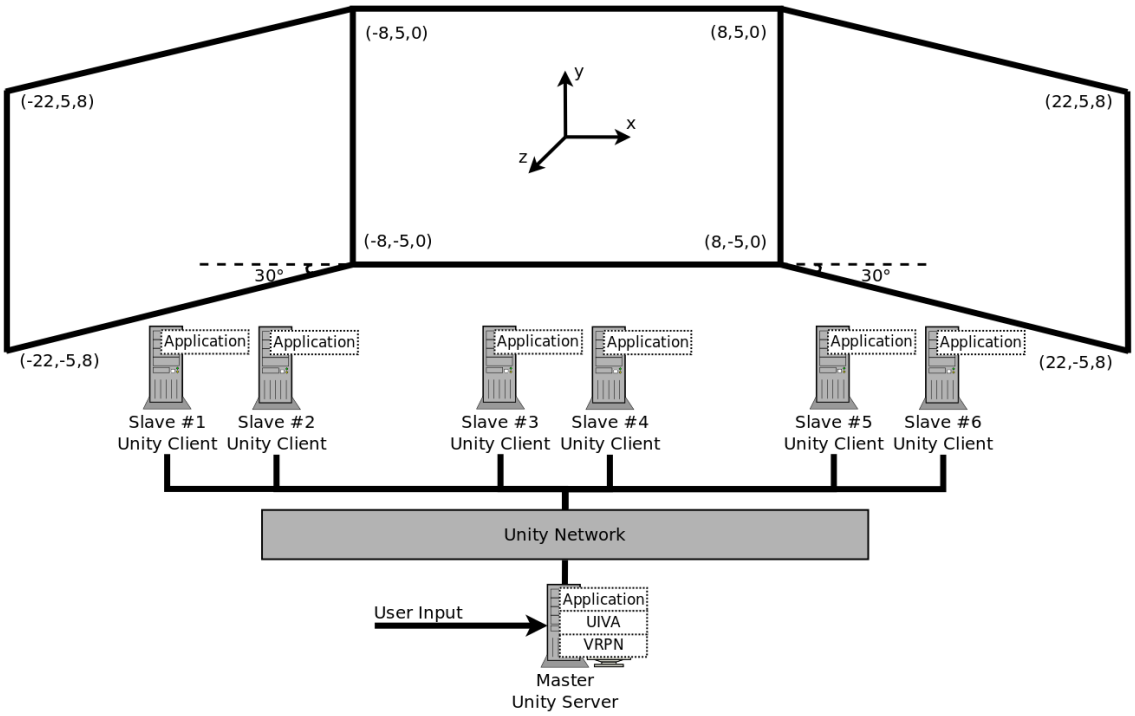
O componente *Multi Projection Camera* é uma câmera virtual Unity customizada que implementa na função *PreRender()* herdada da classe base *MonoBehaviour*, a matriz de projeção de perspectiva generalizada de Kooima (2015), que para tal utiliza os pontos Pa, Pb, Pc e Pe ilustrados na Figura 15.

Este componente também agrupa os objetos *ProjectionPlane* (PlaneMesh Unity – objeto primitivo Unity para plano) e *UserHead* (SphereMesh Unity – objeto primitivo Unity para esfera). Os pontos Pa, Pb e Pc são obtidos no AV por meio do objeto *ProjectionPlane*. A posição do objeto *UserHead* é tomada como ponto Pe, o que permite a implementação do movimento da paralaxe atribuindo a este objeto o movimento da cabeça do usuário.

3.2.2 Node Manager

O componente *Node Manager* utiliza o módulo de rede do motor de jogo Unity para implementar o modelo de renderização em AG Mestre-Escravo. Este componente inicializa a aplicação de acordo com o tipo do nó do AG, onde cada nó escravo (Unity Client) deve estar conectado ao nó mestre (Unity Server), como apresenta a Figura 23.

Figura 23 – A disposição das aplicações desenvolvidas com o Unity Cluster Package para o sistema de multiprojeção MiniCAVE.



Fonte: Popolin Neto et al. (2015a).

O componente *Node Manager* também é responsável por instanciar o componente *Multi Projection Camera* utilizando a função *Instantiate()* fornecida pelo módulo de rede do motor de jogo Unity. Desta maneira, toda aplicação de nó escravo possui uma instância desta câmera virtual customizada. A posição e rotação do *Multi Projection Camera* são alteradas somente na aplicação de nó mestre e sincronizadas para as aplicações de nó escravo por meio da funcionalidade StateSynchronization do componente NetworkView, com alta taxa de atualização (sendRate = 100).

As aplicações do AG são configuradas através do arquivo de configuração no padrão XML. A Figura 24 apresenta o arquivo de configuração para o nó escravo Slave #3, que possui aplicação atribuída a tela central com pontos Pa = (-8,-5,0), Pb = (8,-5,0) e Pc = (-8,5,0). O arquivo de configuração para o nó escravo Slave #4 difere quanto ao olho para o qual a aplicação endereça a projeção, sendo este o direito (eye = “right”) para se obter imagens estereoscópicas via estereoscopia passiva.

Figura 24 – O arquivo de configuração do nó escravo Slave #3.

```
<node type="slave">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="true" eye="left">
    <pa x="-8" y="-5" z="0"/>
    <pb x="8"  y="-5" z="0"/>
    <pc x="-8" y="5"  z="0"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

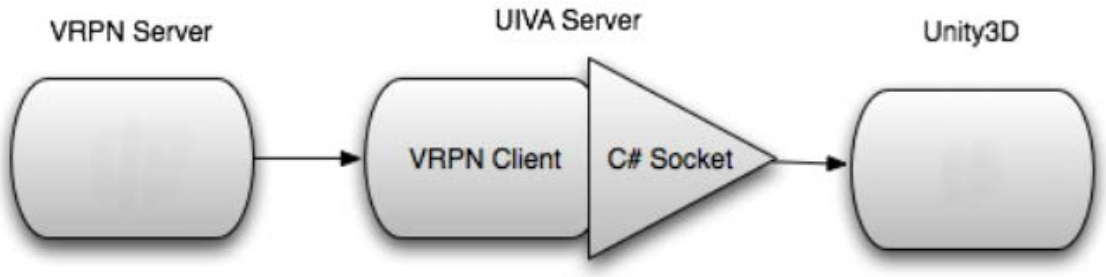
Fonte: Popolin Neto et al. (2015a).

3.2.3 Device Manager

O módulo de Entrada/Saída do motor de jogo Unity fornece a classe *Input* para o acesso de dados dos dispositivos de iteração suportados (*mouse*, teclado, *joystick* e dispositivos móveis). As variáveis estáticas da classe *Input* que representam tais dados são atualizadas a cada *frame* renderizado.

O componente *Device Manager* habilita o uso de dispositivos de interação de servidores VRPN utilizando o adaptador *Unity Indie VRPN Adapter* (UIVA). Desenvolvido por Wang e Lindeman (2015), UIVA é um adaptador para conexão a servidores de dispositivos VRPN implementados em C++ na versão *free* do motor de jogo Unity, uma vez que nessa versão [8] não é permitido o uso de bibliotecas de código nativo C/C++. A Figura 25 ilustra a relação entre um servidor VRPN, o adaptador UIVA e o motor de jogo Unity.

Figura 25 – A relação entre um servidor VRPN, o adaptador UIVA e o motor de jogo Unity.



Fonte: Adaptado de Noyes (2013).

O componente *Device Manager* permite o uso do Wii Remote™, como dispositivo de interação, e o Microsoft Kinect™, como dispositivo de rastreamento da posição da cabeça do usuário, através de *scripts* baseados na classe *Input*. Estes *scripts* possuem variáveis estáticas, atualizadas a cada *frame* renderizado, contendo dados desses populares controles para consoles.

[8] A versão do Unity ao qual o texto se refere é a 4.6.0f3.

Além de suportar outros dispositivos, como o sensor SpacePoint Fusion e o sistema de captura de movimento PhaseSpace, o adaptador UIVA é *open source*, o que permite a implementação do suporte a novos dispositivos. Utilizando o componente *Device Manager* é possível atribuir ao objeto *UserHead* do componente *Multi Projection Camera* o movimento da cabeça do usuário, obtendo assim, a implementação do movimento da paralaxe.

3.3 Considerações Finais

O Unity é tido como um dos motores de jogo mais populares por possuir um editor de fácil uso, motor de renderização customizado (Directx ou OpenGL), documentação completa e por ser multiplataforma. Tais características tornam o motor de jogo Unity um boa escolha para o desenvolvimento de AVs interativos.

A implementação do design de uma solução genérica do Capítulo 2 para o motor de motor de jogo Unity deu origem ao Unity Cluster Package, sendo possível a comparação contida na Tabela 6 deste pacote Unity a outras extensões para motores de jogo, como também ao *plugin* de RV mais utilizado com motores de jogo na implementação da técnica de multiprojeção, a solução comercial MiddleVR.

Tabela 6 – A comparação entre o Unity Cluster Package a as extensões existentes para motores de jogo e o *plugin* de RV MiddleVR.

Solução	Motor de Jogo	Movimento da Paralaxe	Estereoscopia	VRPN	Reusabilidade
Unity Cluster Package	Unity	Sim	Passiva	Sim	Componentes *drag-and-drop*
CaveUT™	Unreal® Engine	Sim	Ativa	Sim	DLL
CaveUDK	Unreal® Engine	Sim	Ativa	Sim	DLL
CryVE	CryEngine® 2	Sim	–	Não	DLL
RUIS	Unity	Sim	Ativa	Não	Componentes *drag-and-drop*
MiddleVR	Unity	Sim	Ativa/Passiva	Sim	DLL

Fonte: Produzida pelo autor.

O suporte ao movimento da paralaxe e a estereoscopia tem como objetivo melhorar a imersão no AV. Com o uso da estereoscopia passiva habilitada pelo Unity Cluster Package utilizando lentes polarizadas, como as presentes no sistema de multiprojeção MiniCAVE, imagens estereoscópicas podem ser geradas com placas gráficas e projetores comuns; por outro lado, a estereoscopia ativa requer *hardware* específico, como a placa gráfica NVIDIA Quadro 5000 e os *shutter glasses* utilizados no CaveUDK (LUGRIN et al., 2012).

O suporte a servidores VRPN permite o uso de um grande número de dispositivos de forma padronizada, habilitando o desenvolvimento de aplicações independentes de dispositivos.

A reusabilidade do Unity Cluster Package está sobre seus componentes *drag-and-drop*, que podem ser utilizados no painel Scene na criação do AV no Editor Unity. As extensões CaveUT™, CaveUDK e CryVE são códigos *open source* compilados em *Dynamic Link Library* (DLL), requerendo conhecimento de programação específico para o uso das mesmas.

A extensão para o motor de jogo Unity RUIS disponibiliza componentes *drag-and-drop* que podem ser utilizados para o desenvolvimento de aplicações de multiprojeção com este popular motor de jogo. Entretanto, além de não oferecer suporte a servidores VRPN, em seu trabalho Takala (2014) não apresenta de forma explícita como RUIS implementa a técnica de multiprojeção, apresentado como foco desta extensão a criação de *interfaces* espaciais de interação.

O MiddleVR pode ser utilizado com o motor de jogo Unity a fim de se desenvolver aplicações para sistemas de RV, como para sistemas de multiprojeção baseados no CAVE™ (STEPTOE; STEED; SLATER, 2013). Esta solução fornece suporte à AGs, ambas abordagens de estereoscopia (ativa e passiva) e a servidores VRPN, mas também consiste de códigos fonte compilados em DLLs além de ser uma solução comercial fechada. O Unity Cluster Package é *open source* e está disponível para *download* no projeto Sourceforge Unity Cluster Package [9].

[9] https://sourceforge.net/projects/unityclusterpackage/

4 Aplicações Unity Multiprojetadas

Este Capítulo apresenta as aplicações Unity desenvolvidas utilizando o Unity Cluster Package para o sistema de multiprojeção MiniCAVE. A utilização dos componentes *drag-and-drop* bem definidos e altamente configuráveis deste pacote Unity também é apresentada neste Capítulo, bem como as avaliações quantitativas e qualitativas executadas para com essas aplicações Unity multiprojetadas.

4.1 Demos Unity no MiniCAVE

Os demos Unity StarTrooper e Bootcamp disponíveis no Unity Asset Store foram adaptados para o sistema de multiprojeção MiniCAVE. Os componentes do Unity Cluster Package foram utilizados na adaptação de tais demos, com os nós do AG dispostos como apresentado na Figura 23. O processo de adaptação pode ser descrito em três passos básicos:

1. Adicionar no painel Scene o componente *Node Manager*, que inicializa a aplicação do nó do AG e instancia o componente *Multi Projection Camera* de acordo com o tipo do mesmo.

2. Adicionar no painel Scene o componente *Device Manager*, que permite acesso aos dados dos dispositivos Wii Remote™ e Microsoft Kinect™ atualizados a cada *frame* renderizado.

3. Definir os objetos de cena que necessitam ser sincronizados entre os nós do AG utilizando o componente NetworkView do módulo de rede do motor jogo Unity, onde somente a aplicação de nó mestre deve realizar mudanças de estado (posição e orientação) em tais objetos.

O StarTrooper no MiniCAVE (Figura 26-a) consiste em uma arena circular onde o voo de uma aeronave e o lançamento de mísseis são controlados por meio de um Wii Remote™. Com relação ao terceiro passo, os objetos de cena aeronave, míssil e anel são sincronizados utilizando o componente NetworkView do módulo de rede do motor de jogo Unity. O demo Unity StarTrooper é apresentado com maiores detalhes na seção 4.2.

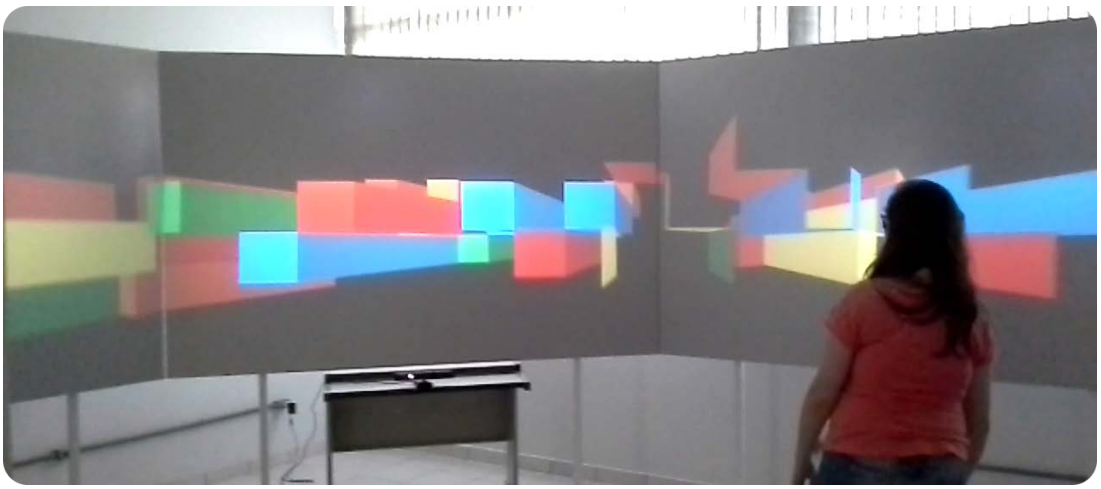
O Bootcamp Explore (Figura 26-b) habilita a navegação em primeira pessoa no rico cenário do demo Unity Bootcamp no MiniCAVE utilizando o Wii Remote™ e seu acessório Nunchuk™. Os *scripts* Unity do pacote Character Controller para o controle de câmera em primeira pessoa foram adaptados e aplicados no componente *Multi Projection Camera*, permitindo o uso do Wii Remote™ e Nunchuk™ como dispositivos de controle, utilizando o componente NetworkView para sincronização (terceiro passo).

Figura 26 – Os demos Unity StarTrooper e Bootcamp adaptados para o MiniCAVE.

Fonte: Popolin Neto et al. (2015a).

O movimento da paralaxe foi implementado no Bootcamp Explore, associando a posição do *UserHead* do componente *Multi Projection Camera* a posição da cabeça do usuário obtida por meio do Microsoft Kinect™. Abordagem também utilizada na aplicação Unity multiprojetada "A Sala dos Paralelepípedos", apresentada na Figura 27.

Figura 27 – A Sala dos Paralelepípedos: Aplicação Unity desenvolvida utilizando o Unity Cluster Package para o MiniCAVE.



Fonte: Popolin Neto e Brega (2015b).

Para o uso dos dispositivos Wii Remote™ e Microsoft Kinect™, o nó mestre do AG é responsável pela execução do adaptador UIVA e de dois servidores VRPN, o disponibilizado por Wang e Lindeman (2015) junto com o adaptador UIVA para o Wii Remote™, e o FAAST desenvolvido por Suma et al. (2011) para o Microsoft Kinect™. Algumas alterações foram feitas no adaptador UIVA para que este pudesse suportar o acessório Nunchuk™. Os demos Unity adaptados podem ser vistos no vídeo [1] disponibilizado no YouTube.

A performance dos demos Unity adaptados para o MiniCAVE é apresentada na Tabela 7, tendo como relação a taxa de FPS e a média do número de triângulos renderizados. O Unity Cluster Package apresenta boa relação entre a taxa de FPS e a média do número de triângulos renderizados, uma vez que aplicações de RV devem apresentar taxa maior de que 30 FPS. A extensão CaveUDK possui taxa de 50 FPS para o intervalo de 200 K e 600 K triângulos renderizados (LUGRIN et al., 2012), e CryVE apresenta taxas inferiores a 20 FPS (JUAREZ; SCHONENBERG; BARTNECK, 2010).

Tabela 7 – A performance dos demos Unity StarTrooper e Bootcamp adaptados para o MiniCAVE.

Adaptação	FPS	Média de Triângulos Renderizados
StarTrooper	566	12K
Bootcamp Explore	58	237K

Fonte: Popolin Neto et al. (2015a).

4.2 StarTrooper Multijogador – MiniCAVE vs. Tablet

Jogos imersivos são empregados em diversos propósitos, como no jogo sério desenvolvido por Backlund et al. (2007), dando origem ao simulador para treinamento de bombeiros apresentado na Figura 7, como também para entretenimento, onde jogos *First Person Shooter* (FPS) é um dos gêneros mais explorados em sistemas imersivos (JACOBSON; HWANG, 2002; BECKHAUS; BLOM; HARINGER, 2005; TEDJOKUSUMO; ZHOU; WINKLER, 2010; LUGRIN et al., 2013).

O CAVE Quake II permite a interação entre o jogador no sistema imersivo e jogadores remotos em PCs. Este pioneiro jogo imersivo foi seguido pelo uso da extensão CaveUT™ no porte do jogo Unreal Tournament para um determinado sistema de multiprojeção (JACOBSON; HWANG, 2002), que em uma versão posterior habilita múltiplos jogadores em diferentes sistemas de multiprojeção (JACOBSON; LEWIS, 2005). Por sua vez, a extensão CaveUDK foi utilizada no porte de jogos para um sistema imersivo onde o jogador interage com *Non Playing Character* (NPC) (LUGRIN et al., 2013).

[1] https://www.youtube.com/watch?v=nvVDAvdw_k8

O StarTrooper Multijogador apresentado nesta seção é uma adaptação do demo Unity StarTrooper integrando plataformas distintas em um jogo multijogador, sendo estes dispositivos móveis e o sistema de multiprojeção MiniCAVE. Jogos imersivos multijogador também podem ser encontrados para diferentes sistemas de RV, como os jogos FPS desenvolvidos por Tedjokusumo, Zhou e Winkler (2010), onde dois jogadores interagem utilizando *Head Mounted Display* (HMD). O StarTrooper Multijogador suporta e integra *hardware* heterogêneos, sendo estes o AG do sistema de multiprojeção MiniCAVE e um dispositivo móvel utilizando o sistema operacional Android.

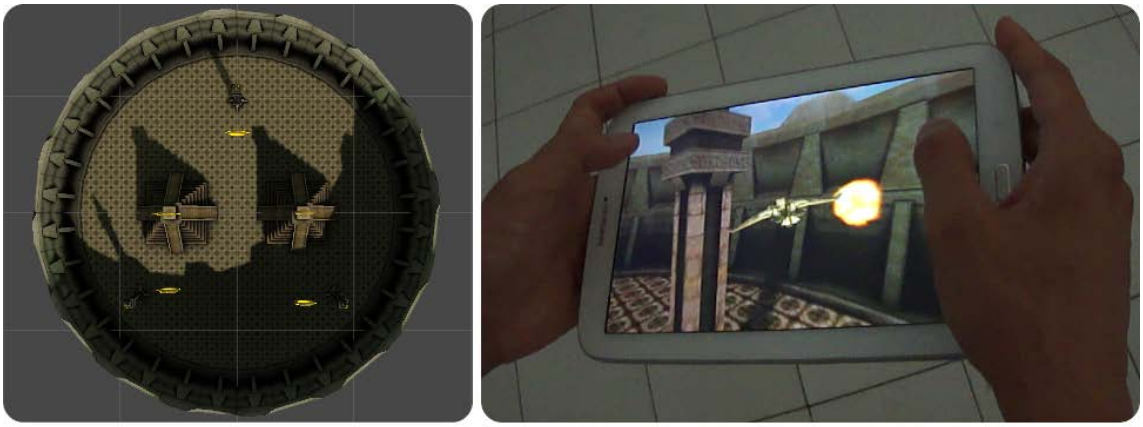
Os jogos que utilizam os dispositivos ChairIO e Game Gun desenvolvidos por Beckhaus, Blom e Haringer (2005) tem como objetivo fornecer uma nova abordagem de interação por meio de dispositivos customizados. No StarTrooper Multijogador o acelerômetro e a tela sensível ao toque do dispositivo móvel são utilizados para interação, enquanto que no sistema de multiprojeção MiniCAVE o Wii Remote™ é utilizado.

O StarTrooper Multijogador possibilita a interação entre jogadores submetidos a experiências diferentes, integrando dispositivo móvel e sistema de RV, utilizando o módulo de rede do motor de jogo Unity e o Unity Cluster Package.

4.2.1 O Demo Unity StarTrooper

O demo Unity StarTrooper dispõe de duas pirâmides, três anéis em constante rotação e três obeliscos dispostos como obstáculos em uma arena circular (Figura 28-a), onde o usuário controla o voo de uma aeronave e lança mísseis (Figura 28-b). O projeto StarTrooper disponível no Unity Asset Store é originalmente direcionado a dispositivos móveis, utilizando o acelerômetro para controlar o voo da aeronave, *roll* para direita/esquerda e *pitch* para cima/baixo, e um toque único na tela para o lançamento de míssil.

Figura 28 – A arena StarTrooper (a). A execução do StarTrooper em um dispositivo móvel (b).



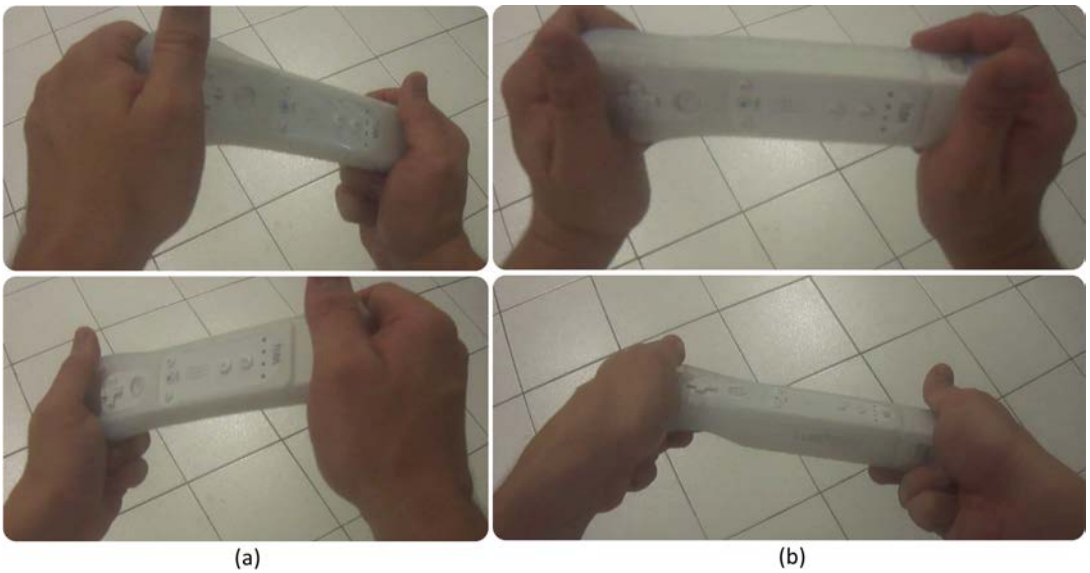
(a) (b)

Fonte: Produzida pelo autor.

4.2.2 O StarTrooper Multijogador

O StarTrooper no MiniCAVE fornece ao jogador um maior campo de visão por meio das três telas e um Wii Remote™ como dispositivo de interação. O acelerômetro do Wii Remote™ é utilizado no controle do voo da aeronave, *pitch* para direita/esquerda (Figura 29-a) e *roll* para cima/baixo (Figura 29-b), sendo o oposto do dispositivo móvel uma vez que estes possuem direções de eixos diferentes. O botão 'B' do Wii Remote™ localizado logo abaixo do dedo indicador esquerdo na Figura 29 é utilizado para o lançamento de mísseis.

Figura 29 – O uso do Wii Remote™ no StrarTrooper no MiniCAVE.



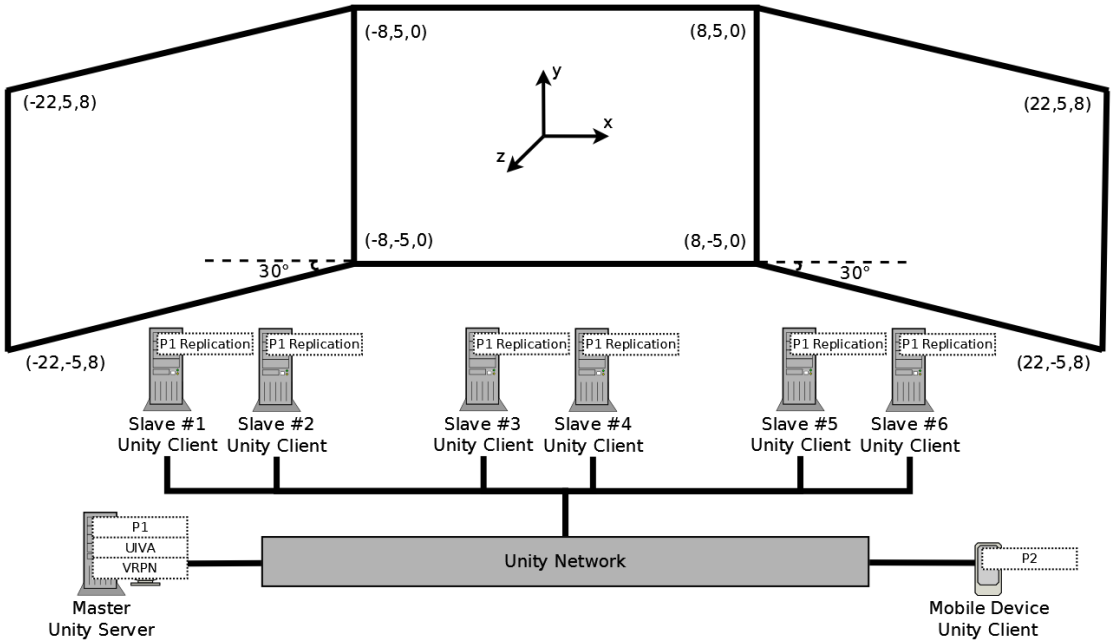
Fonte: Produzida pelo autor.

A integração entre MiniCAVE e dispositivo móvel depende diretamente da conexão entre ambas plataformas para se alcançar a comunicação e sincronização necessária. Uma vez que o Unity Cluster Package implementa o modelo de renderização em AG Mestre-Escravo, o nó mestre é responsável pela execução de uma instância do jogo, P1 na Figura 30, e cada nó escravo executa uma replicação da instância P1 com diferentes configurações de câmera virtual, enquanto que no dispositivo móvel tem-se a segunda instância do jogo, P2 na Figura 30.

Além dos três passos básicos e das configurações e responsabilidades dos nós do AG, apresentados na seção 4.1, no StarTrooper Multijogador o componente *NodeManager* também compreende a inicialização da aplicação (segunda instância do jogo) no dispositivo móvel. A Figura 31 apresenta os arquivos de configuração para o nó escravo Slave #3 e do dispositivo móvel.

Com relação a dinâmica do jogo, tem-se a destruição da nave inimiga após o acerto de três mísseis na mesma. O objeto de cena aeronave possui um contador de acertos sincronizado entre as instâncias do jogo utilizando a função *OnSerializeNetworkView()* do componente Networkview.

Figura 30 – A disposição das aplicações na integração entre MiniCAVE e dispositivo móvel utilizando o StarTrooper Multijogador.



Fonte: Produzida pelo autor.

Figura 31 – O arquivo de configuração para o nó escravo Slave #3 (a). O arquivo de configuração do dispositivo móvel file (b).

```
<node type="slave">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="true" eye="left">
    <pa x="-8" y="-5" z="0"/>
    <pb x="8"  y="-5" z="0"/>
    <pc x="-8" y="5"  z="0"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

(a)

```
<node type="mobile">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="false" eye="">
    <pa x="-8" y="-5" z="0"/>
    <pb x="8"  y="-5" z="0"/>
    <pc x="-8" y="5"  z="0"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

(b)

Fonte: Produzida pelo autor.

Os efeitos de fumaça e faíscas do pacote Unity Particles são utilizados na colisão do míssil com a aeronave, onde a emissão dos efeitos depende de uma variável boleana também sincronizada utilizando a função *OnSerializeNetworkView()*. Efeitos sonoros também foram adicionados no projeto StarTrooper, utilizando áudios [2] editados no *software* Audacity® [3] para o motor do aeronave e para o voo e explosão do míssil.

[2] https://www.freesound.org/
[3] http://audacity.sourceforge.net/

4.2.3 MiniCAVE vs. Tablet

Em pares, voluntários jogaram o StarTrooper Multijogador duas vezes (Figura 32), uma vez no MiniCAVE e uma vez no tablet (dispositivo móvel). Uma única partida teve duração máxima de seis minutos. O primeiro minuto foi destinado ao aprendizado da plataforma, onde o jogador pôde aprender como controlar o voo da aeronave e lançar mísseis. Os cinco minutos restantes foram utilizados para o combate.

Figura 32 – Dois voluntários jogando o StarTrooper Multijogador.

Fonte: Produzida pelo autor.

Um total de quarenta jogadores participaram do experimento 'MiniCAVE vs. Tablet', vinte e oito do sexo masculino e doze do feminino, entre dezoito e trinta e dois anos de idade. Cada jogador pontuou seu conhecimento prévio em jogos digitais de 1 (pouco conhecimento) a 5 (excelente conhecimento). A Tabela 8 contém as frequências e os percentuais da pontuação do conhecimento prévio em jogos digitais. Obteve-se como maior percentual 37.5, onde os jogadores pontuaram em 5 o conhecimento prévio em jogos digitais.

Tabela 8 – A pontuação do conhecimento dos jogadores quanto a jogos digitais.

Pontuação	Frequência	Percentual
1 - Pouco	6	15
2 - Razoável	3	7.5
3 - Mediano	6	15
4 - Bom	10	25
5 - Excelente	15	37.5
Total	40	100

Fonte: Produzida pelo autor.

Após jogar em ambas plataformas, foi aplicado nos jogadores uma adaptação (Tabela 9) do questionário criado por Chertoff, Goldiez e LaViola (2010) *Virtual Experience Test* (VET), a fim de qualificar a experiência obtida no StarTrooper no MiniCAVE em três das cinco dimensões de Chertoff et al. (2008): afetiva, relacional e sensorial. As questões selecionadas foram pontuadas de 1 (descordo totalmente) a 5 (concordo totalmente).

Tabela 9 – As questões selecionadas do questionário VET.

Questão	Dimensão
1) Na minha opinião, o equipamento de visualização (tela) é de alta qualidade.	Sensorial
3) Na minha opinião, o conteúdo visual do ambiente virtual é de alta qualidade.	Sensorial
4) O ambiente foi capaz de suportar múltiplos usuários ao mesmo tempo.	Relacional
5) Minhas reações emocionais foram de acordo com os eventos ocorridos no ambiente virtual.	Afetiva
7) Senti uma variedade de emoções enquanto trabalhava nas tarefas do ambiente virtual.	Afetiva
8) Um alto grau de interação com outros usuários foi necessário para cumprir as tarefas no ambiente virtual.	Relacional
17) Na minha opinião, o equipamento de áudio é de alta qualidade.	Sensorial
19) Na minha opinião, o áudio do ambiente virtual é de alta qualidade.	Sensorial
24) A informação sensorial do ambiente foi consistente. Por exemplo, o som de dois objetos metálicos colidindo soou de modo metálico.	Sensorial

Fonte: Adaptado de Chertoff, Goldiez e LaViola (2010).

A dimensão afetiva no StarTrooper no MiniCAVE está relacionada as emoções sentidas pelos jogadores na visualização e interação nesta plataforma imersiva. A Tabela 10 contém as frequências da pontuação das questões associadas a dimensão afetiva e a Tabela 11 apresenta os percentuais destas frequências. Obteve-se como maior percentual global 43.75, onde os jogadores pontuaram em 4 a dimensão afetiva.

Tabela 10 – As frequências da pontuação das questões associadas a dimensão afetiva.

	1	2	3	4	5	Total
Questão 5	0	1	6	18	15	40
Questão 7	0	2	13	17	8	40
Total	0	3	19	35	23	80

Fonte: Produzida pelo autor.

A dimensão relacional pode ser associada a característica multijogador oferecida por meio da integração entre MiniCAVE e dispositivo móvel. A Tabela 12 contém as frequências da pontuação das questões associadas a dimensão relacional.

Tabela 11 – Os percentuais correspondentes as frequências da pontuação das questões associadas a dimensão afetiva.

	1	2	3	4	5	Total
Questão 5	0	2.5	15	45	37.5	100
Questão 7	0	5	32.5	42.5	20	100
Global	0	3.75	23.75	43.75	28.75	100

Fonte: Produzida pelo autor.

A Tabela 13 o apresenta os percentuais das frequências da pontuação das questões associadas a dimensão relacional. Obteve-se como maior percentual global 47.5, onde os jogadores pontuaram em 5 a dimensão relacional.

Tabela 12 – As frequências da pontuação das questões associadas a dimensão relacional.

	1	2	3	4	5	Total
Questão 4	1	0	3	10	26	40
Questão 8	1	3	13	11	12	40
Total	2	3	16	21	38	80

Fonte: Produzida pelo autor.

Tabela 13 – Os percentuais correspondentes as frequências da pontuação das questões associadas a dimensão relacional.

	1	2	3	4	5	Total
Questão 4	2.5	0	7.5	25	65	100
Questão 8	2.5	7.5	32.5	27.5	30	100
Global	2.5	3.75	20	26.25	47.5	100

Fonte: Produzida pelo autor.

A dimensão sensorial pode ser atribuída ao conteúdo visual do demo Unity StarTrooper, aos efeitos visuais e sonoros adicionados e ao *hardware* contido no sistema de multiprojeção MiniCAVE. A Tabela 14 contém as frequências da pontuação das questões associadas a dimensão sensorial e a Tabela 15 o apresenta os percentuais destas frequências. Obteve-se como maior percentual global 40.5, onde os jogadores pontuaram em 5 a dimensão sensorial.

Para a verificação da existência da diferença na pontuação de cada questão dentro de cada dimensão, foi aplicado o teste de Wilcoxon para a comparação entre as duas questões das dimensões afetiva e relacional, e o teste de Friedman para a comparação entre as cinco questões da dimensão sensorial. A Tabela 16 contém os resultados dos teses executados onde se foi adotado nível de significância $p < 0.05$.

Tabela 14 – As frequências da pontuação das questões associadas a dimensão sensorial.

	1	2	3	4	5	Total
Questão 1	0	0	9	16	15	40
Questão 3	0	0	9	19	12	40
Questão 17	0	3	7	12	18	40
Questão 19	0	3	9	12	16	40
Questão 24	1	0	12	7	20	40
Total	1	6	46	66	81	200

Fonte: Produzida pelo autor.

Tabela 15 – Os percentuais correspondentes as frequências da pontuação das questões associadas a dimensão sensorial.

	1	2	3	4	5	Total
Questão 1	0	0	22.5	40	37.5	100
Questão 3	0	0	22.5	47.5	30	100
Questão 17	0	7.5	17.5	30	45	100
Questão 19	0	7.5	22.5	30	40	100
Questão 24	2.5	0	30	17.5	50	100
Global	0.5	3	23	33	40.5	100

Fonte: Produzida pelo autor.

O teste de Wilcoxon apontou a existência de diferença estatisticamente significante entre as duas questões da dimensão afetiva e relacional, enquanto o teste de Friedman não apresentou diferença entre as cinco questões da dimensão sensorial.

Para a verificação da correlação entre o conhecimento prévio em jogos digitais (Tabela 8) e a pontuação das questões associadas as dimensões foi utilizado o coeficiente de correlação de Sperman, apresentado na Tabela 17. O resultado não mostrou haver correlação entre o conhecimento prévio e a pontuação dada as questões selecionadas do VET.

4.2.4 Análise dos Resultados e Comparação dos Sistemas Imersivos

A diferença estatisticamente significante entre as questões da dimensão afetiva e a média das mesmas apontam que os jogadores sentiram reações emocionais (Questão 5), mas que estas não são variadas (Questão 7).

Quanto a diferença e média das questões da dimensão relacional, os jogadores alegam que o ambiente é capaz de suportar vários usuários ao mesmo tempo (Questão 4) mais do que a necessidade de um alto grau de interação com o outro jogador para completar as tarefas no AV (Questão 8).

Tabela 16 – Os resultados dos testes de Wilcoxon e Friedman.

Dimensão	Questão	Média	Erro Padrão	p
Afetiva	5	4.18	0.12	0.015
	7	3.78	0.13	
	Total	3.98	0.10	-
Relacional	4	4.50	0.13	0.002
	8	3.75	0.17	
	Total	4.13	0.12	-
Sensorial	1	4.15	0.12	0.847
	3	4.08	0.12	
	17	4.13	0.15	
	19	4.03	0.15	
	24	4.13	0.16	
	Total	4.10	0.06	-

Fonte: Produzida por José Roberto Pereira Lauris.

Tabela 17 – Os coeficientes de correlação de Sperman entre conhecimento prévio em jogos digitais e as dimensões afetiva, relacional e sensorial.

Dimensão	r	p
Afetiva	0.29	0.066
Relacional	0.07	0.670
Sensorial	-0.04	0.800

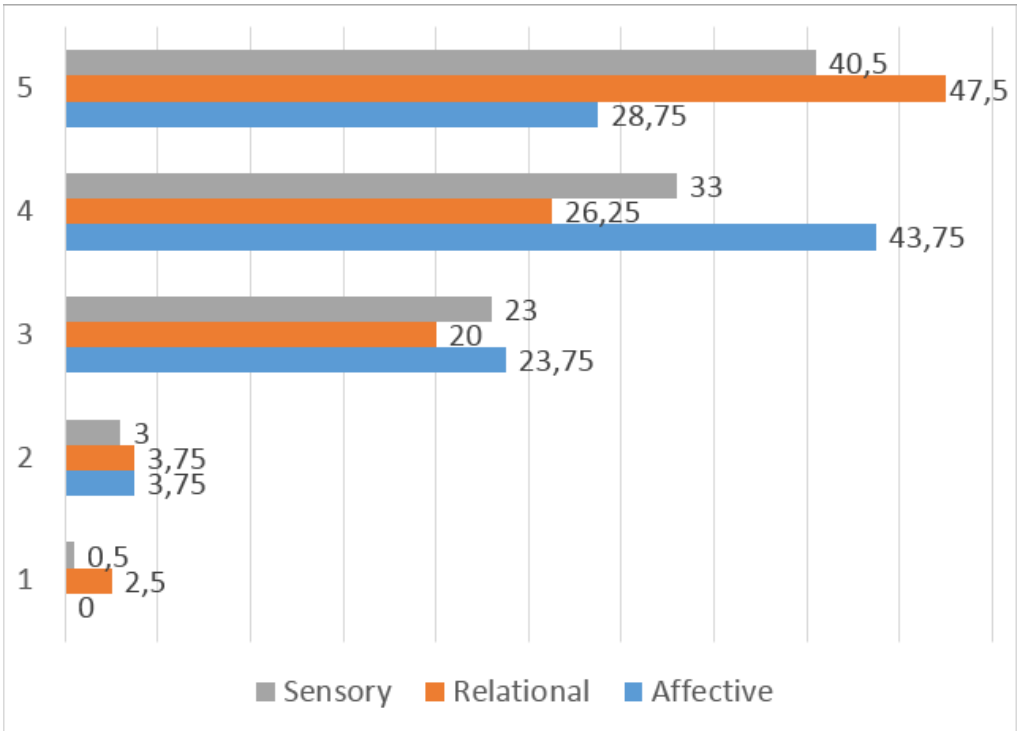
Fonte: Produzida por José Roberto Pereira Lauris.

A ausência de diferença entre as questões associadas a dimensão relacional junto aos percentuais e médias positivas validam o conteúdo visual do demo Unity StarTrooper, os efeitos visuais e sonoros adicionados e o *hardware* contido no sistema de multiprojeção MiniCAVE.

Apesar dos percentuais positivos das Tabelas 11, 13 e 15 apresentados na Figura 33, as diferenças constadas nas dimensões afetiva e relacional, apresentadas na Tabela 16, indicam a demanda do StarTrooper Multijogador por um número variado de tarefas no AV a fim de proporcionar emoções variadas e aumentar a interação entre os jogadores.

Algumas considerações podem ser feitas na comparação entre o sistema empregado no StarTrooper Multijogador e outros sistemas imersivos utilizados como plataforma para jogos digitais (Tabela 18). O StrarTrooper Multijogador integra MiniCAVE e dispositivo móvel oferecendo as características especificas de cada plataforma. O Wii Remote™ é utilizado como dispositivo de interação no MiniCAVE, utilizando seu acelerômetro e o botão gatilho 'B'. Além de apresentar o jogo, o próprio dispositivo móvel é utilizado na interação, por meio do acelerômetro e a tela sensível ao toque. O sistema utilizado no StarTrooper Multijogador permite jogadores co-localizados em telas separadas, fornecendo dispositivos de interação sem fio não restringindo os movimentos dos jogadores.

Figura 33 – Os percentuais globais correspondentes as frequências da pontuação das questões agrupadas por dimensão.



Fonte: Produzida pelo autor.

4.3 Uma Aplicação de Visualização Aumentada

Estudos realizados no CaveDataView apontam uma melhor performance de usuário em AV imersivo com relação a visualização da informação em gráficos de dispersão 3D (RAJA et al., 2004). Desenvolvido por Wijayasekara, Linda e Manic (2011), o sistema CAVE-SOM fornece ao usuário interação intuitiva e natural em AVs imersivos, enquanto preserva informações com relação a redução de dimensionalidade para a construção de representações em duas dimensões (2D).

Ambos sistemas de visualização (CaveDataView e CAVE-SOM) utilizam um bastão como dispositivo de interação e não permitem múltiplos usuários simultâneos devido a implementação do movimento da paralaxe, uma vez que a correção da perspectiva do AV requer o ponto de visão de somente um usuário (RAJA et al., 2004; WIJAYASEKARA; LINDA; MANIC, 2011).

A seleção de parte da estrutura 3D de maneira a obter informações adicionais pode ser trabalhosa com bastões, e a perda da visão geral na navegação e na requisição por informações pode ser um problema ainda maior, uma vez que desorientação e navegação livre pode fazer com que os usuários se percam dentro do AV (HENRY; POLYS, 2010).

Tabela 18 – A comparação entre os sistemas imersivos utilizados como plataforma para jogos digitais.

	Motor de Jogo	Exibição	Dispositivos de Interação	Plataforma 2
StarTrooper Multijogador	Unity	Sistema de multiprojeção	Wii Remote™	Dispositivo móvel
CAVE Quake II	Quake II Engine	Sistema de multiprojeção	Bastão	PC
CaveUT™	Unreal® Engine	Sistema de multiprojeção	Bastão	Sistema de multiprojeção
CaveUDK™	Unreal® Engine	Sistema de multiprojeção	Bastão	–
ChairIO e Game Gun	Unreal® Engine	Projeção única	ChairIO e Game Gun	–
HMD FPS	Cube Engine	HMD	Bastão	HMD

Fonte: Produzida pelo autor.

Baseado na Visualização Aumentada (VA), que é processo onde hipóteses sobre os dados são formadas e refinadas em um ambiente interativo que permite a manipulação, exploração e navegação dos dados (WARE, 2012), o grande campo de visão fornecido pelos sistemas de multiprojeção pode ser utilizado a fim de se explorar grandes volumes de dados, por meio de uma visão geral nas múltiplas telas e visões específicas em dispositivos móveis (POPOLIN NETO et al., 2015b).

A aplicação Unity desenvolvida com o Unity Cluster Package apresentada nesta seção integra plataformas distintas em um sistema de visualização imersivo e interativo. Este sistema visa fornecer uma experiência imersiva por meio do aumento do campo de visão e da estereoscopia, permitindo a visão geral da representação 3D dos dados a múltiplos usuários simultaneamente no sistema de multiprojeção MiniCAVE. A visão geral pode ser utilizada como orientação na navegação dos usuários realizada em dispositivos móveis, onde estes podem navegar em uma área de trabalho individual e acessar informações sob demanda.

O sistema de visualização XIM utiliza um sistema de multiprojeção de três telas com diferentes visualizações dos dados (BETELLA et al., 2012). Neste sistema, grafos 3D criados com o motor de jogo Unity são apresentados utilizando o movimento da paralaxe na tela central, uma visão geral dos grafos 3D é apresentada na tela direita, e diagramas dinâmicos são exibidos na tela esquerda.

Além de não permitir múltiplos usuários, como os sistemas CaveDataView e CAVE-SOM, o sistema XIM de Betella et al. (2012) não permite a requisição de informações adicionais, o usuário navega pelos dados utilizando o Wii Remote™ e o Microsoft Kinect™ com a uma visão geral na tela direita para a orientação na navegação na representação.

A aplicação de visualização desenvolvida neste trabalho também utiliza grafos 3D para representações de dados. As três telas do MiniCAVE são utilizadas para orientação na navegação, bem como para identificar agrupamentos, padrões, tendências e anomalias na representação 3D, enquanto que em dispositivos móveis usuários podem navegar e requerer informações adicionais utilizando a tela sensível ao toque.

Diferente dos sistemas mencionados acima (CaveDataView, CAVE-SOM e XIM), a aplicação desenvolvida com o motor de jogo Unity e o Unity Cluster Package suporta e integra *hardware* heterogêneos, sendo estes o AG do sistema de multiprojeção MiniCAVE e dispositivos móveis utilizando o sistema operacional Android.

4.3.1 Um Sistema de Visualização Imersivo e Interativo

Os PCs do AG (nós mestre e escravos) e os dispositivos móveis executam a mesma aplicação de visualização, onde a aplicação de nó mestre (Unity Server) conecta e sincroniza as demais (Unity Clients), como apresenta a Figura 34.

O Unity Cluster Package fornece projeções continuas e sincronizadas no MiniCAVE por meio das replicações do componente *Multi Projection Camera* nas aplicações de nó escravo. O *Multi Projection Camera* é instanciado na aplicação de nó mestre e faz uso do componente Networkview. Deste modo, qualquer alteração de estado (posição e orientação) do *Multi Projection Camera* na aplicação de nó mestre é sincronizada para as aplicações de nó escravo.

O componente *Node Manager* na aplicação de visualização executada em um dispositivo móvel cria uma nova instância do *Multi Projection Camera*, não estando esta vinculada e sincronizada com a aplicação de visualização de nó mestre e com as de nó escravo.

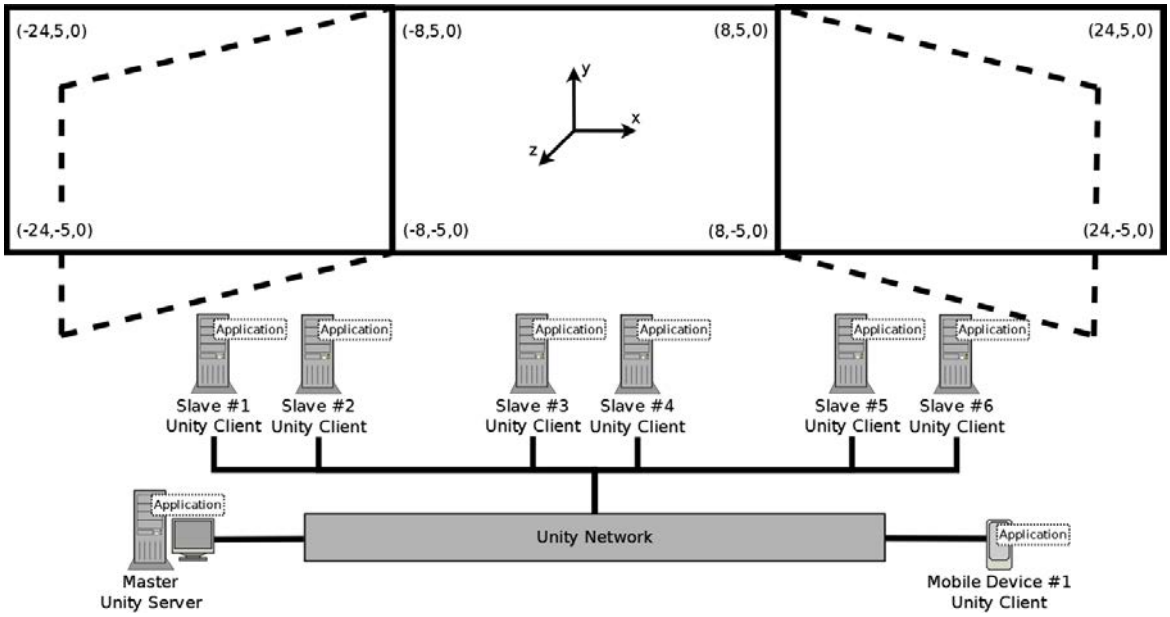
O ângulo de 30° das telas laterais do MiniCAVE não foi considerado a fim de se obter uma melhor visão geral nas três telas para múltiplos usuários. A Figura 35 apresenta o arquivo de configuração para o nó escravo Slave #4.

4.3.2 Extração de Dados

O módulo *desktop* do sistema de visualização desenvolvido por Moraes, Eler e Brega (2014) DBVis é utilizado na extração de dados. Por meio da *interface* de usuário (UI) do módulo *desktop* do DBVis é possível a execução de comandos *Structured Query Language* (SQL) sobre determinado banco de dados relacional e, baseado no conjunto de dados resultante, pode-se mapear as colunas em atributos visuais como tamanho, cor, formato e conexões.

A UI do DBVis foi desenvolvida em Java, que suporta a conexão com vários sistemas de banco de dados relacionais por meio da especificação *Java Database Connectivity* (JDBC), e utiliza a biblioteca JUNG, que fornece representações do conjunto de dados resultantes de consultas SQL em grafos 2D.

Figura 34 – A disposição das aplicações na integração entre MiniCAVE e dispositivo móvel utilizando a aplicação de visualização.



Fonte: Popolin Neto et al. (2015b).

Figura 35 – O arquivo de configuração para o nó escravo Slave #4.

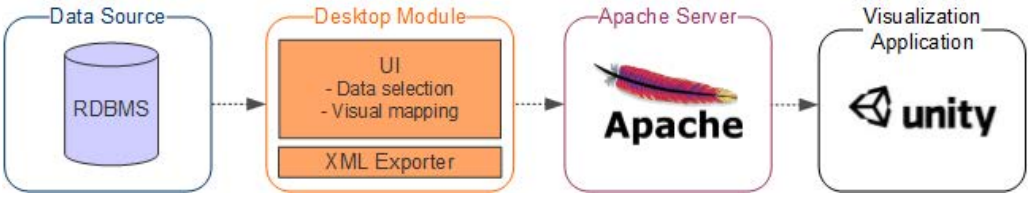
```
<node type="slave">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="true" eye="right">
    <pa x="-8" y="-5" z="0"/>
    <pb x="8"  y="-5" z="0"/>
    <pc x="-8" y="5"  z="0"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

Fonte: Popolin Neto et al. (2015b).

Os dados não inicialmente mapeados em propriedades visuais podem ser utilizados como terceira dimensão (coordenada z para grafos 3D) ou informações adicionais em formato de texto. Após o mapeamento visual é possível exportar um arquivo no padrão XML contendo a representação dos dados e atributos (MORAES; ELER; BREGA, 2014).

A UI do DBVis é utilizada na criação e exportação de representações de dados contidos em banco de dados relacionais em grafos 3D. O arquivo XML exportado é obtido pela aplicação de visualização de um servidor Apache. O fluxo dos dados é ilustrado na Figura 36.

Figura 36 – O fluxo dos dados.



Fonte: Popolin Neto et al. (2015b).

4.3.3 A Criação de Grafos 3D

A aplicação de visualização obtém o arquivo XML exportado de um servidor Apache utilizando a classe Unity *WWW*, que permite acessar conteúdos de *Uniform Resource Locator* (URL). Todos os nós do grafo 3D são criados como novas instâncias de um prefab Unity denominado GraphNode, que possui três recursos Unity: SphereMesh, TextMesh e LineRenderer.

De acordo com a propriedades obrigatórias *x*, *y*, *z*, *id*, *label*, *size* e *color*, para cada nó do grafo 3D contido no arquivo XML exportado tem-se uma nova instância do prefab GraphNode, onde o SphereMesh (objeto primitivo Unity para esfera) tem posição (*x*, *y*, *z*), *id* e *label* como identificadores, escala como *size*, e cor do material de renderização como *color*.

Os atributos remanescentes de cada nó do grafo 3D contidos no arquivo XML exportado são inseridos no TextMesh (texto passível de renderização) do respectivo GraphNode, no tocante a possibilidade de requerer informações adicionais de um determinado nó. O recurso Unity TextMesh permite o uso de um conjunto de *tags* para incorporar múltiplos estilos de fonte e tamanho. O TextMesh no prefab GraphNode é oculto por padrão caracterizando a exploração dessas informações adicionais sob demanda.

As arestas do grafo 3D são representadas pelo recurso LineRenderer presente no prefab GraphNode, que toma um conjunto de dois ou mais pontos tridimensionais e desenha uma linha reta passando pelos mesmos. O LineRenderer possui a cor do GraphNode origem para representar um grafo 3D direcionado e em caso de um grafo 3D não direcionado, o mesmo deve possuir a cor de ambos GraphNodes (origem e destino). O uso de LineRenderers para representar as arestas do grafo 3D otimiza a complexidade de renderização da cena, uma vez que este recurso Unity não é renderizado como um objeto tridimensional, e sim como linhas planas sem espessura que possuem comprimento, textura e cor.

O prefab GraphNode possui *scripts* C# não somente para a construção de grafos 3D utilizando os recursos Unity SphereMesh, TextMesh e LineRenderer, mas também para implementar as dinâmicas de visualização e interação.

4.3.4 As Dinâmicas de Visualização e Interação

As dinâmicas de visualização e interação são baseadas no processo da VA e no experimento realizado por Henry e Polys (2010), que aponta tendências da imersão e navegação na aquisição de conhecimento espacial em representações de dados em grafos 3D. Henry e Polys (2010) submeteram voluntários a diferentes opções de imersão e navegação e obtiveram por meio de um questionário o conhecimento dos participantes a respeito do grafo 3D apresentado.

No experimento de Henry e Polys (2010), a interação com o grafo 3D foi realizada em múltiplas telas, e foi observado uma forte tendência entre os usuários de ter como foco a tela central e como distração as telas laterais, bem como a de navegar de forma distante da representação para se obter uma visão geral da mesma.

No sistema de visualização desenvolvido neste trabalho, uma visão geral imersiva do grafo 3D é fornecida no MiniCAVE (grade campo de visão e estereoscopia), enquanto a interação em dispositivos móveis oferece uma área de trabalho individual no qual usuários podem navegar sem perder o contexto da visão geral. A separação da visão geral e interação entre MiniCAVE e dispositivos móveis tem como objetivo fornecer interação intuitiva e um melhor uso das múltiplas telas.

Henry e Polys (2010) constaram, quanto a navegação em tela única em grafos 3D, que usuários obtiveram uma melhor performance utilizando a câmera virtual em modo egocêntrico.

Nos dispositivos móveis, usuários podem navegar em três graus de liberdade utilizando a tela sensível ao toque para movimentar a câmera virtual para cima/baixo (deslizando um dedo para cima/baixo), esquerda/direita (deslizando um dedo para esquerda/direita) e frente/trás (deslizando dois dedos separando/unindo).

Tendo como base a VA, dispositivos móveis são também utilizados na exploração e exibição de informações adicionais de um nó específico do grafo 3D tocando o mesmo na tela sensível, tornando visível o TextMesh do GraphNode em questão com os atributos adicionais.

Sendo o *feedback* visual da seleção dos nós do grafo 3D utilizado como ponto de referência na navegação (HENRY; POLYS, 2010), o prefab GraphNode possui o recurso Unity PacticleEmitter com o intuito de renderizar um efeito de seleção nos nós no MiniCAVE quando o usuário os selecionar no dispositivo móvel. O procedimento do efeito de seleção é uma função Unity RPC [4] executada nas aplicações de nó escravo requisitada por uma aplicação de dispositivo móvel. Tocando o nó pela segunda vez os recurso TextMesh e PacticleEmitter são ocultados, desabilitando a visualização dos atributos adicionais no dispositivo móvel e o efeito de seleção no MiniCAVE.

Com o toque duplo em nó do grafo 3D no dispositivo móvel, uma visão geral secundária é realizada no MiniCAVE, tendo como foco o nó selecionado.

[4] Funções RPC (*Remote Procedure Call*) são funções executadas remotamente em outras instâncias de uma aplicação Unity (seção 3.1.2).

A função de seleção em foco é uma função Unity RPC executada na aplicação de nó mestre requisitada por uma aplicação de dispositivo móvel, que move o componente *Multi Projection Camera* de modo a focar o nó selecionado. Com um toque duplo no *background* da representação tem-se o retorno para visão geral original no MiniCAVE.

4.3.5 Caso de Uso

Um conjunto de dados contendo informações estruturais a respeito de um banco de dados relacional de uma universidade foi tomado como caso de uso. Este banco de dados relacional utiliza PostgreSQL 9.0 e possui 700 tabelas, 1100 chaves estrangeiras e 77 milhões de linhas.

Todos os sistemas de *software* da universidade armazenam dados neste banco de dados e muitos destes sistemas ainda estão em desenvolvimento, sendo necessária uma atenção especial quanto a estrutura do banco de dados. Deste modo, visualizações que possam auxiliar na identificação de possível problemas estruturais são de grande utilidade aos desenvolvedores de tais sistemas.

Utilizando a UI do DBVis, o conjunto resultante de uma consulta SQL para obter os dados estruturais é mapeado com tabelas como nós do grafo 3D e chaves estrangeiras como arestas. A cor do nó representa o esquema a qual tabela é pertencente e o tamanho está associado ao número de relações que tabela possui. O grafo 3D foi então exportado para um arquivo XML, que além dos atributos mapeados contém para cada nó do grafo 3D atributos adicionais (Tabela 19), onde o "Número Estimado de Linhas" é utilizado como a profundidade do nó (terceira dimensão – coordenada z).

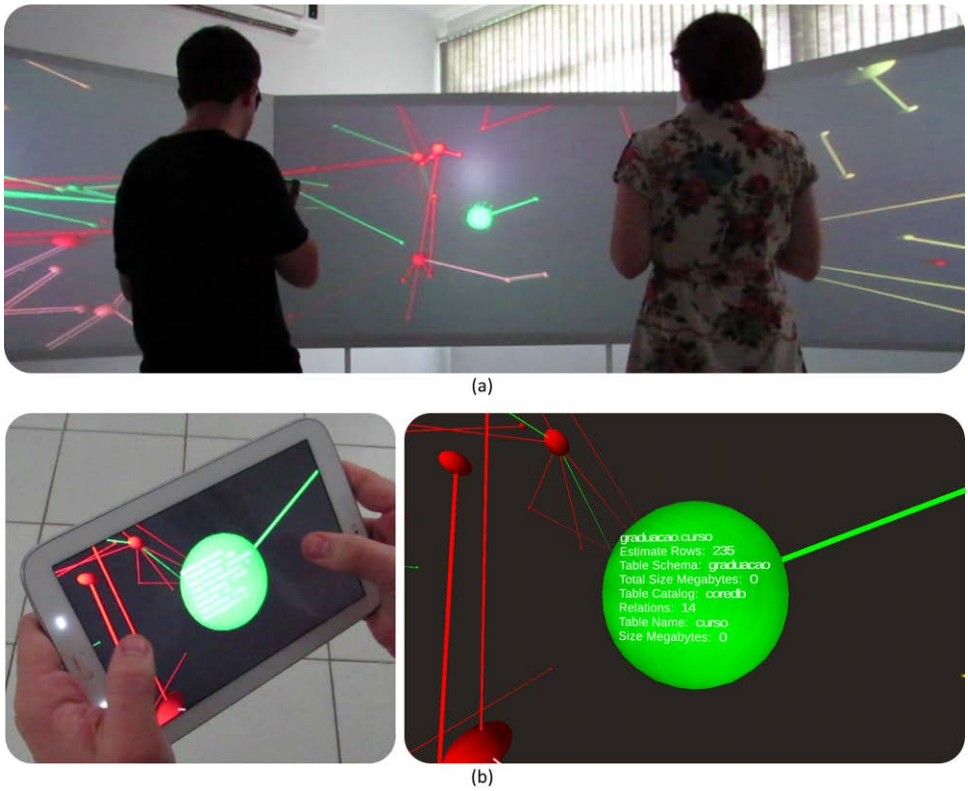
Tabela 19 – Os atributos adicionais.

Atributo	Descrição
Número Estimado de Linhas	Número estimado de linhas de dados da tabela.
Esquema da Tabela	Esquema onde a tabela está localizada.
Tamanho Total em Megabytes	Tamanho da tabela com dados e indexes em megabytes.
Catalogo da Tabela	Banco de dados onde a tabela está localizada.
Relações	Número de relações com outras tabelas.
Nome da Tabela	Nome da tabela.
Tamanho em Megabytes	Tamanho da tabela em megabytes.

Fonte: Popolin Neto et al. (2015b).

O grafo 3D exportado visualizado no sistema desenvolvido auxilia os desenvolvedores na identificação de tabelas importantes, em termos de relacionamentos e alto uso, por meio da visualização geral dos nós e das arestas e pela sensação de profundidade no MiniCAVE (Figura 37-a), uma vez que as arestas representam chaves estrangeiras e o número de linhas é a profundidade do nó.

Figura 37 – Visão geral imersiva no MiniCAVE (múltiplos usuários) (a). Navegação e informação adicional no dispositivo móvel (área de trabalho individual) (b).



Fonte: Popolin Neto et al. (2015b).

As informações adicionais quanto as tabelas de interesse são obtidas pelo desenvolvedor em sua área de trabalho individual no dispositivo móvel, tocando os respectivos nós na tela sensível ao toque (Figura 37-b), onde os atributos adicionais (Tabela 19) são apresentados gerando um efeito de seleção nos nós no MiniCAVE (Figura 37-a).

4.3.6 Comparação entre Sistemas de Visualização Imersivos

Algumas considerações podem ser feitas na comparação entre o sistema de visualização imersivo e interativo desenvolvido e os outros sistemas de visualização citados (Tabela 20). O sistema desenvolvido habilita múltiplos usuários simultaneamente, uma vez que o movimento da paralaxe não foi utilizado e dispositivos móveis proporcionam áreas de trabalho individuais. Com a separação da visão geral e interação entre MiniCAVE e dispositivos móveis, os usuários podem navegar e requerer informações adicionais sem perder o contexto da visão geral. Por meio de telas sensíveis ao toque, dispositivos móveis aparentam ser intuitivos no quesito interação para visualização, talvez mais do que bastões e controles de consoles (Wii Remote™ / Microsoft Kinect™).

Tabela 20 – A comparação entre sistemas de visualização imersivos.

Sistema	Telas de Projeção	Estereoscopia	Movimento da Paralaxe	Representação dos Dados	Dispositivo de Interação	Usuários Simultâneos
MiniCAVE / Dispositivos Móveis	3-Telas	Passiva	Não	Grafo 3D	Dispositivo Móvel	1-6
CaveDataView	4-Telas	Ativa	Sim	Diagramas de Dispersão 3D	Bastão	1
CAVE-SOM	4-Telas	Ativa	Sim	Cubos 3D	Bastão	1
XIM	3-Telas	–	Sim	Grafo 3D	Controles de Consoles	1

Fonte: Adaptado de Popolin Neto et al. (2015b).

4.4 Considerações Finais

O Unity Cluster Package é uma solução flexível para o desenvolvimento de aplicações imersivas e interativa com o motor de jogo Unity. Este pacote foi utilizado em projetos Unity compilados para plataforma Linux (seis PCs de renderização do AG do MiniCAVE), Windows (o PC servidor do AG do MiniCAVE) e Android (tablet), tornando possível a integração entre sistema de multiprojeção e dispositivos móveis.

Os testes quantitativos, que relacionam a taxa de FPS e a média do número de triângulos renderizados, apresentaram bons resultados para AVs com baixo (StarTrooper) e alto (Bootcamp) número de polígonos.

Os testes qualitativos, sendo aplicado uma adaptação do VET, validaram o sistema de multiprojeção utilizado (MiniCAVE) e apontaram a necessidade de um número variado de tarefas no StarTrooper Multijogador, a fim de proporcionar uma interação mais intensa entre os jogadores e variadas reações emocionais.

O Unity Cluster Package foi utilizado no desenvolvimento de aplicações Unity multiprojetadas para o sistema de multiprojeção MiniCAVE. Contudo, este pacote Unity suporta diferentes sistemas de multiprojeção de acordo com o tamanho, proporção, posição e orientação da telas, visto a variação das configurações para o próprio MiniCAVE na aplicação de visualização, onde não foi considerado o angulo de 30° das telas laterais.

A aplicação de visualização desenvolvida habilita a exploração de grandes volumes de dados de banco de dados relacionais, permitindo o mapeamento de quaisquer dados armazenados utilizando a UI do DBVis, em um sistema de visualização imersivo e interativo integrando plataformas distintas.

5 Conclusão

Este Capítulo apresenta as conclusões com base nos resultados da revisão sistemática da literatura, nas caraterísticas do Unity Cluster Package, e nos resultados das avaliações quantitativas e qualitativas da aplicações Unity multiprojetadas.

A técnica de multiprojeção de AVs pode ser empregada com sucesso nas tecnologias para o desenvolvimento de jogos digitais. A multiprojeção de AVs gerados por motores de jogo é utilizada com maior frequência nos domínios de aplicação Arte, Visualização, Simulação e Treinamento, e Entretenimento. A abordagem de *hardware* mais utilizada em AVs criados com motores de jogo é o AG sob o modelo de renderização Mestre-Escravo, que requer suporte de *software* para alcançar imagens em alta resolução e processamento distribuído.

As extensões para motores de jogo específicos, que implementam o modelo de renderização em AG Mestre-Escravo, permitem o desenvolvimento em alto nível de aplicações com AVs imersivos. As extensões diferem das bibliotecas de RV ao utilizar altas taxas de sincronização sobre o suporte de rede do motor de jogo, em vez de utilizar as técnicas Framelock e Datalock, para obter coerência de imagem nas múltiplas telas.

Baseado nessas tendências, o design de uma solução genérica proposto suporta diferentes sistemas de multiprojeção de acordo com o tamanho, proporção, posição e orientação da telas, e o modelo de renderização em AG Mestre-Escravo utilizando o suporte de rede do motor de jogo, além de estereoscopia passiva e o movimento da paralaxe.

O Unity Cluster Package é a implementação do design de uma solução genérica para o motor de jogo Unity, um dos mais populares motores de jogo utilizados na atualidade. Além de validar o design de uma solução genérica, este pacote Unity facilita o desenvolvimento de aplicações de RV imersivas e interativas por meio de componentes *drag-and-drop*. O Unity Cluster Package difere das extensões para motores de jogo existentes dentre os quesitos motor de jogo (Unity), estereoscopia (abordagem passiva), acesso a dispositivos de interação (servidores no padrão VRPN) e reusabilidade (componentes *drag-and-drop*).

O Unity Cluster Package foi utilizado na adaptação e desenvolvimento de aplicações para o sistema de multiprojeção MiniCAVE. Os demos Unity StarTrooper e Bootcamp foram adaptados para este sistema de multiprojeção de baixo custo. Os testes quantitativos realizados para com estes demos Unity adaptados apresentou bons resultados.

Para o demo Unity StarTropper foi desenvolvido uma segunda adaptação, integrando o sistema de multiprojeção MiniCAVE e dispositivo móvel no jogo StarTrooper Multijogador. No experimento ‘MiniCAVE vs. Tablet’, quarenta voluntários jogaram o StarTrooper Multijogador e foram submetidos aos testes qualitativos (adaptação do VET) para o StarTrooper no MiniCAVE.

Os resultados dos testes qualitativos validaram o sistema de multiprojeção utilizado e apontaram a necessidade de um número variado de tarefas no StarTrooper Multijogador.

A integração entre MiniCAVE e dispositivo móvel ainda foi utilizada em uma aplicação de visualização, que permite a exploração de grandes volumes de dados em um sistema de visualização imersivo e interativo. A aplicação desenvolvida visou integrar MiniCAVE e dispositivos móveis de maneira complementar, fornecendo um visão geral imersiva no MiniCAVE, e visões específicas por meio da navegação e requisição de informações adicionais nos dispositivos móveis.

Contribuições do Trabalho

- A constatação do uso de altas taxas de atualização dos objetos sobre o suporte de rede do motor de jogo para se alcançar coerência de imagem em AVs multiprojetados, em vez de se utilizar as técnicas de Framelock e Datalock encontradas em bibliotecas de RV.
- O design de uma solução genérica que permite a criação de extensões para motores de jogo, suportando diferentes sistemas de multiprojeção e atendendo as tendências do uso da técnica de multiprojeção em AVs gerados por motores de jogo.
- O Unity Cluster Package que facilita o desenvolvimento de aplicações de RV por meio de componentes *drag-and-drop*, habilitando diferentes sistemas de multiprojeção, estereoscopia passiva, movimento da paralaxe e acesso a servidores no padrão VRPN.
- A integração entre o sistema de multiprojeção MiniCAVE e dispositivo móvel, por meio do StarTrooper Multijogador, que permite a interação entre jogares em plataformas distintas, e em uma aplicação de VA, que oferece a visualização imersiva e interativa de grafos 3D a múltiplos usuários simultaneamente.

6 Trabalhos Futuros

Este Capítulo sugere os trabalhos futuros quanto ao design de uma solução genérica para motores de jogo, ao Unity Cluster Package e as aplicações Unity multiprojetadas desenvolvidas para o MiniCAVE.

De maneira geral, pretende-se continuar a explorar o motor de jogo Unity em sistemas de multiprojeção, particularmente no MiniCAVE integrando dispositivos móveis, e ao passo do surgimento de novos motores do jogo, aplicar o design de uma solução genérica nestes, de modo a permitir e facilitar o uso das novas tecnologias para a criação de jogos digitais no desenvolvimento de aplicações com AVs imersivos.

Além de implementar um número variado de tarefas no StarTrooper Multijogador, planeja-se desenvolver jogos e aplicações para avaliação do uso do Microsoft Kinect™ na implementação do movimento da paralaxe, para interação de forma colaborativa entre os usuários da integração MiniCAVE e dispositivos móveis, e para a integração de dois sistemas de multiprojeção MiniCAVE.

Pretende-se também explorar grandes volumes de dados de banco de dados relacionais no sistema de visualização imersivo e interativo desenvolvido, tal como realizar a avaliação deste sistema. Propõe-se investigar outras representações 3D e a visualização de dados na web, uma vez que o motor de jogo Unity habilita o acesso a URLs por meio da classe *WWW*.

7 Publicações Realizadas

Este Capítulo cita os textos científicos elaborados e publicados durante o desenvolvimento deste trabalho.

POPOLIN NETO, M.; BREGA, J. R. F. Ambientes virtuais multiprojetados gerados pelo motor de jogo unity. In: *IV Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP*. Presidente Prudente, SP, Brasil: WPPGCC, 2014.

POPOLIN NETO, M.; AGOSTINHO, I. A.; DIAS, D. R. C.; RODELLO, I. A.; BREGA, J. R. F. A realidade virtual e o motor de jogo unity. In: *Tendências e Técnicas em Realidade Virtual e Aumentada*. São Paulo, SP, Brasil: SBC, 2015. p. 9–23. ISSN 2177-6776.

POPOLIN NETO, M.; BREGA, J. R. F. A survey of solutions for game engines in the development of immersive applications for multi-projection systems as base for a generic solution design. In: *Virtual and Augmented Reality (SVR), 2015 XVII Symposium on*. [S.l.: s.n.], 2015. p. 61–70.

POPOLIN NETO, M.; COLOMBO, D. R.; GUIMARÃES, M. d. P.; BREGA, J. R. F. Unity cluster package – dragging and dropping components for multi-projection virtual reality applications based on pc clusters. In: GERVASI, O.; MURGANTE, B.; MISRA, S.; GAVRILOVA, M. L.; ROCHA, A. M. A. C.; TORRE, C.; TANIAR, D.; APDUHAN, B. O. (Ed.). *Computational Science and Its Applications – ICCSA 2015*. Springer International Publishing, 2015. (Lecture Notes in Computer Science, v. 9159), p. 261–272. ISBN 978-3-319-21412-2. Disponível em: <http://dx.doi.org/10.1007/978-3-319-21413-9_19>. **Prêmio de melhor artigo no Virtual Reality and Applications Workshop da 15th International Conference on Computational Science and Its Applications**.

POPOLIN NETO, M.; BREGA, J. R. F. O design de uma solução genérica aplicada no motor de jogo unity para o desenvolvimento de aplicações de multiprojeção. In: *V Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP*. Bauru, SP, Brasil: WPPGCC, 2015. **Menção honrosa pelo segundo melhor trabalho da linha de pesquisa Processamento de Imagens e Visão Computacional**.

POPOLIN NETO, M.; MORAES, A. C.; ELER, D. M.; BREGA, J. R. An immersive and interactive visualization system by integrating distinct platforms. In: *Information Visualisation (IV), 2015 19th International Conference on*. [S.l.: s.n.], 2015.

Referências

ABRAMYAN, L.; POWELL, M.; NORRIS, J. Stage: Controlling space robots from a cave on earth. In: *Aerospace Conference, 2012 IEEE*. [S.l.: s.n.], 2012. p. 1–6. ISSN 1095-323X. Citado na página 26.

ALMEIDA, J.; ROSSETTI, R.; COELHO, A. L. Mapping 3d character location for tracking players' behaviour. In: *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*. [S.l.: s.n.], 2013. p. 1–5. Citado na página 35.

ANDERSON, E. F.; MCLOUGHLIN, L.; WATSON, J.; HOLMES, S.; JONES, P.; PALLETT, H.; SMITH, B. Choosing the infrastructure for entertainment and serious computer games - a whiteroom benchmark for game engine selection. In: *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2013 5th International Conference on*. [S.l.: s.n.], 2013. p. 1–8. Citado 4 vezes nas páginas 21, 22, 23 e 39.

BACKLUND, P.; ENGSTROM, H.; HAMMAR, C.; JOHANNESSON, M.; LEBRAM, M. Sidh - a game based firefighter training simulation. In: *Information Visualization, 2007. IV '07. 11th International Conference*. [S.l.: s.n.], 2007. p. 899–907. ISSN 1550-6037. Citado 2 vezes nas páginas 25 e 48.

BECKHAUS, S.; BLOM, K. J.; HARINGER, M. A new gaming device and interaction method for a first-person-shooter. *Proceedings of the Computer Science and Magic*, Leipzig, Germany, 2005. Citado 2 vezes nas páginas 48 e 49.

BETELLA, A.; CARVALHO, R.; SANCHEZ-PALENCIA, J.; BERNARDET, U.; VERSCHURE, P. F. M. J. Embodied interaction with complex neuronal data in mixed-reality. In: *Proceedings of the 2012 Virtual Reality International Conference*. New York, NY, USA: ACM, 2012. (VRIC '12), p. 3:1–3:8. ISBN 978-1-4503-1243-1. Disponível em: <http://doi.acm.org/10.1145/2331714.2331718>. Citado na página 58.

BOUVIER, P.; SORBIER, F. D.; 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>. Citado na página 22.

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. Citado na página 19.

BUCHINGER, D.; JURASZEK, G. D.; HOUNSELL, M. d. S. Estudo bibliométrico do crescimento da área de realidade virtual. In: *Workshop de Realidade Virtual e Aumentada*. [S.l.: s.n.], 2012. p. 1–4. Citado na página 19.

CARTER, C.; RHALIBI, A.; MERABTI, M. Development and deployment of cross-platform 3d web-based games. In: *Developments in E-systems Engineering (DESE), 2010*. [S.l.: s.n.], 2010. p. 149–154. Citado na página 35.

CAVAZZA, M.; LUGRIN, J.-L.; CROOKS, S.; NANDI, A.; PALMER, M.; RENARD, M. L. Causality and virtual reality art. In: *Proceedings of the 5th Conference on Creativity &Amp;*

Cognition. New York, NY, USA: ACM, 2005. (C&C '05), p. 4–12. ISBN 1-59593-025-6. Disponível em: <http://doi.acm.org/10.1145/1056224.1056228>. Citado na página 24.

CHERTOFF, D.; GOLDIEZ, B.; LAVIOLA, J. Virtual experience test: A virtual environment evaluation questionnaire. In: *Virtual Reality Conference (VR), 2010 IEEE*. [S.l.: s.n.], 2010. p. 103–110. ISSN 1087-8270. Citado na página 53.

CHERTOFF, D. B.; SCHATZ, S. L.; MCDANIEL, R.; BOWERS, C. A. Improving presence theory through experiential design. *Presence*, v. 17, n. 4, p. 405–413, Aug 2008. ISSN 1054-7460. Citado na página 53.

CRAIGHEAD, J. Distributed, game-based, intelligent tutoring systems - the next step in computer based training? In: *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*. [S.l.: s.n.], 2008. p. 247–256. Citado na página 35.

CRAIGHEAD, J.; BURKE, J.; MURPHY, R. Using the unity game engine to develop sarge: a case study. In: . [S.l.: s.n.], 2008. (IROS '08), p. 366–372. Citado na página 35.

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>. Citado 4 vezes nas páginas 16, 19, 20 e 21.

DIAS, D. R. C.; BREGA, J. R. F.; LAMARCA, A. F.; NETO, M. P.; SUGUIMOTO, D. J.; AGOSTINHO, I.; GOUVEIA, A. F. Chemcave3d: Sistema de visualização imersivo e interativo de moléculas 3d. In: *Workshop de Realidade Virtual e Aumentada*. Uberaba, MG, Brasil: WRVA, 2011. Citado na página 30.

DIAS, D. R. C.; BREGA, J. R. F.; TREVELIN, L. C.; NETO, M. P.; GNECCO, B. B.; GUIMARAES, M. de P. Design and evaluation of an advanced virtual reality system for visualization of dentistry structures. In: *Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on*. [S.l.: s.n.], 2012. p. 429–435. Citado na página 30.

ELLIS, S. What are virtual environments? *Computer Graphics and Applications, IEEE*, v. 14, n. 1, p. 17–22, 1994. ISSN 0272-1716. Citado na página 19.

FILHO, N.; BOTELHO, S.; CARVALHO, J.; MARCOS, P. de B.; MAFFEI, R. de Q.; OLIVEIRA, R.; HAX, V. An automated platform for immersive and collaborative visualization of industrial models. In: *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*. [S.l.: s.n.], 2009. p. 258–264. Citado na página 24.

HASHIMOTO, N.; ISHIDA, Y.; SATO, M. A self-distributing software environment for immersive multiprojector displays. *Systems and Computers in Japan*, v. 38, n. 2, p. 1–9, 2007. Cited By (since 1996)0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-33847238082&partnerID=40&md5=b84d3dbd381062c39cdeee4ee9f36743>. Citado na página 28.

HENRY, J. A.; POLYS, N. F. The effects of immersion and navigation on the acquisition of spatial knowledge of abstract data networks. *Procedia Computer Science*, v. 1, n. 1, p. 1737 – 1746, 2010. ISSN 1877-0509. ICCS 2010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050910001961>. Citado 2 vezes nas páginas 57 e 62.

HU, W.; QU, Z.; ZHANG, X. A new approach of mechanics simulation based on game engine. In: *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*. [S.l.: s.n.], 2012. p. 619–622. Citado na página 35.

JACOBSON, J.; HWANG, Z. Unreal tournament for immersive interactive theater. *Commun. ACM*, ACM, New York, NY, USA, v. 45, n. 1, p. 39–42, jan. 2002. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/502269.502292>. Citado 3 vezes nas páginas 24, 29 e 48.

JACOBSON, J.; LEWIS, M. Game engine virtual reality with caveut. *Computer*, v. 38, n. 4, p. 79–82, April 2005. ISSN 0018-9162. Citado na página 48.

JUAREZ, A.; SCHONENBERG, W.; BARTNECK, C. Implementing a low-cost cave system using the cryengine2. *Entertainment Computing*, v. 1, n. 3–4, p. 157 – 164, 2010. ISSN 1875-9521. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1875952110000108>. Citado 2 vezes nas páginas 29 e 48.

KIRNER, C.; SISCOUTTO, R. Fundamentos de realidade virtual e aumentada. In: *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações*. [S.l.]: Editora SBC - Socidade Brasileira de Computação, Porto alegre, 2007. cap. 1, p. 2–21. ISBN 85-7669-108-6. Citado na página 19.

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>. Citado 2 vezes nas páginas 76 e 77.

KOENIG, S. T.; DUNSER, A.; BARTNECK, C.; DALRYMPLE-ALFORD, J. C.; CRUCIAN, G. P. Development of virtual environments for patient-centered rehabilitation. In: *Virtual Rehabilitation (ICVR), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 1–7. Citado na página 35.

KOOIMA, R. *Generalized Perspective Projection*. 2015. Acesso em: 8 de Março 2015. Disponível em: <http://aoeu.snth.net/static/gen-perspective.pdf>. Citado 2 vezes nas páginas 32 e 41.

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>. Citado na página 21.

LOULOUDI, A.; KLUGL, F. Visualizing agent-based simulation dynamics in a cave - issues and architectures. In: *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*. [S.l.: s.n.], 2011. p. 651–658. Citado na página 29.

LUGRIN, J.-L.; CAVAZZA, M.; CHARLES, F.; RENARD, M. L.; FREEMAN, J.; LESSITER, J. Immersive fps games: User experience and performance. In: *Proceedings of the 2013 ACM International Workshop on Immersive Media Experiences*. New York, NY, USA: ACM, 2013. (ImmersiveMe '13), p. 7–12. ISBN 978-1-4503-2402-1. Disponível em: <http://doi.acm.org/10.1145/2512142.2512146>. Citado na página 48.

LUGRIN, J.-L.; CHARLES, F.; CAVAZZA, M.; RENARD, M. L.; FREEMAN, J.; LESSITER, J. Caveudk: A vr game engine middleware. In: *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*. New York, NY, USA: ACM, 2012. (VRST '12), p. 137–144.

ISBN 978-1-4503-1469-5. Disponível em: <http://doi.acm.org/10.1145/2407336.2407363>. Citado 6 vezes nas páginas 17, 21, 24, 29, 44 e 48.

Matrox Graphics Inc. *Graphics eXpansion Modules*. 2015. Acesso em: 8 de Março 2015. Disponível em: <http://www.matrox.com/graphics/en/products/gxm/>. Citado na página 26.

MORAES, A. C.; ELER, D. M.; BREGA, J. R. F. Collaborative information visualization using a multi-projection system and mobile devices. In: *Information Visualisation (IV), 2014 18th International Conference on*. [S.l.: s.n.], 2014. p. 71–77. ISSN 1550-6037. Citado 3 vezes nas páginas 30, 59 e 60.

NI, T.; ZHAO, D.; ZHANG, H. Realistic vehicle driving simulator with dynamic terrain deformation. In: *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*. [S.l.: s.n.], 2009. p. 4795–4800. Citado na página 27.

NICHOLS, S.; HALDANE, C.; WILSON, J. R. Measurement of presence and its consequences in virtual environments. *International Journal of Human-Computer Studies*, v. 52, n. 3, p. 471 – 491, 2000. ISSN 1071-5819. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1071581999903439>. Citado na página 19.

NOYES, M. A. Implementation of headtracking and 3d stereo with unity and vrpn for computer simulations. NTRS, 2013. Citado na página 43.

PAPE, D. *Mirage 5000 CAVE - Sneak Preview*. 2013. Acesso em: 1 de Dezembro 2013. Disponível em: <http://www.evl.uic.edu/pape/CAVE/DLP/>. Citado na página 16.

PARBERRY, I.; NUNN, J.; SCHEINBERG, J.; CARSON, E.; COLE, J. Sage: a simple academic game engine. *Proceedings of the Second Annual Microsoft Academic Days on Game Development in Computer Science Education*, Citeseer, p. 90–94, 2007. Citado na página 22.

PETRIDIS, P.; DUNWELL, I.; FREITAS, S. de; PANZOLI, D. An engine selection methodology for high fidelity serious games. In: *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*. [S.l.: s.n.], 2010. p. 27–34. Citado na página 22.

POPOLIN NETO, M.; AGOSTINHO, I. A.; DIAS, D. R. C.; RODELLO, I. A.; BREGA, J. R. F. A realidade virtual e o motor de jogo unity. In: *Tendências e Técnicas em Realidade Virtual e Aumentada*. São Paulo, SP, Brasil: SBC, 2015. p. 9–23. ISSN 2177-6776. Citado 4 vezes nas páginas 37, 38, 39 e 69.

POPOLIN NETO, M.; BREGA, J. R. F. Ambientes virtuais multiprojetados gerados pelo motor de jogo unity. In: *IV Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP*. Presidente Prudente, SP, Brasil: WPPGCC, 2014. Citado na página 69.

POPOLIN NETO, M.; BREGA, J. R. F. O design de uma solução genérica aplicada no motor de jogo unity para o desenvolvimento de aplicações de multiprojeção. In: *V Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP*. Bauru, SP, Brasil: WPPGCC, 2015. Citado na página 69.

POPOLIN NETO, M.; BREGA, J. R. F. A survey of solutions for game engines in the development of immersive applications for multi-projection systems as base for a generic solution design. In: *Virtual and Augmented Reality (SVR), 2015 XVII Symposium on*. [S.l.: s.n.], 2015. p. 61–70. Citado 11 vezes nas páginas 22, 23, 24, 26, 29, 31, 33, 34, 47, 69 e 77.

POPOLIN NETO, M.; COLOMBO, D. R.; GUIMARÃES, M. d. P.; BREGA, J. R. F. Unity cluster package – dragging and dropping components for multi-projection virtual reality applications based on pc clusters. In: GERVASI, O.; MURGANTE, B.; MISRA, S.; GAVRILOVA, M. L.; ROCHA, A. M. A. C.; TORRE, C.; TANIAR, D.; APDUHAN, B. O. (Ed.). *Computational Science and Its Applications – ICCSA 2015*. Springer International Publishing, 2015. (Lecture Notes in Computer Science, v. 9159), p. 261–272. ISBN 978-3-319-21412-2. Disponível em: <http://dx.doi.org/10.1007/978-3-319-21413-9_19>. Citado 7 vezes nas páginas 30, 41, 42, 43, 47, 48 e 69.

POPOLIN NETO, M.; MORAES, A. C.; ELER, D. M.; BREGA, J. R. An immersive and interactive visualization system by integrating distinct platforms. In: *Information Visualisation (IV), 2015 19th International Conference on*. [S.l.: s.n.], 2015. Citado 8 vezes nas páginas 40, 58, 60, 61, 63, 64, 65 e 69.

RAFFIN, B.; SOARES, L.; NI, T.; BALL, R.; SCHMIDT, G.; LIVINGSTON, M.; STAADT, O.; MAY, R. Pc clusters for virtual reality. In: *Virtual Reality Conference, 2006*. [S.l.: s.n.], 2006. p. 215–222. ISSN 1087-8270. Citado na página 27.

RAJA, D.; BOWMAN, D. A.; LUCAS, J.; NORTH, C. Exploring the Benefits of Immersion in Abstract Information Visualization. In: *In proceedings of Immersive Projection Technology Workshop*. [S.l.: s.n.], 2004. Citado na página 57.

SANDIN, D. J.; MARGOLIS, T.; GE, J.; GIRADO, J.; PETERKA, T.; DEFANTI, T. A. The varriertm autostereoscopic virtual reality display. In: *ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005. (SIGGRAPH '05), p. 894–903. Disponível em: <http://doi.acm.org/10.1145/1186822.1073279>. Citado na página 32.

SCHOU, T.; GARDNER, H. J. The wiimote with multiple sensor bars: Creating an affordable, virtual reality controller. In: *Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*. New York, NY, USA: ACM, 2009. (CHINZ '09), p. 41–44. ISBN 978-1-60558-574-1. Disponível em: <http://doi.acm.org/10.1145/1577782.1577790>. Citado na página 27.

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>. Citado 3 vezes nas páginas 21, 27 e 28.

SHERMAN, W. R.; CRAIG, A. B. *Understanding virtual reality: Interface, application, and design*. first. [S.l.]: Morgan Kaufmann, 2002. 608 p. (The Morgan Kaufmann Series in Computer Graphics). Citado na página 19.

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>. Citado 2 vezes nas páginas 27 e 28.

STEPTOE, W.; STEED, A.; SLATER, M. Human tails: Ownership and control of extended humanoid avatars. *Visualization and Computer Graphics, IEEE Transactions on*, v. 19, n. 4, p. 583–590, 2013. ISSN 1077-2626. Citado na página 45.

STEUER, J. Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, v. 42, p. 73–93, 1992. Citado na página 19.

SUMA, E.; LANGE, B.; RIZZO, A.; KRUM, D.; BOLAS, M. Faast: The flexible action and articulated skeleton toolkit. In: *Virtual Reality Conference (VR), 2011 IEEE*. [S.l.: s.n.], 2011. p. 247–248. ISSN 1087-8270. Citado na página 48.

TAKALA, T.; RAUHAMAA, P.; TAKALA, T. Survey of 3dui applications and development challenges. In: *3D User Interfaces (3DUI), 2012 IEEE Symposium on*. [S.l.: s.n.], 2012. p. 89–96. Citado na página 24.

TAKALA, T. M. Ruis: A toolkit for developing virtual reality applications with spatial interaction. In: *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*. New York, NY, USA: ACM, 2014. (SUI '14), p. 94–103. ISBN 978-1-4503-2820-3. Disponível em: <http://doi.acm.org/10.1145/2659766.2659774>. Citado 2 vezes nas páginas 29 e 45.

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>. Citado na página 23.

TEDJOKUSUMO, J.; ZHOU, S.; WINKLER, S. Immersive multiplayer games with tangible and physical interaction. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, v. 40, n. 1, p. 147–157, Jan 2010. ISSN 1083-4427. Citado 2 vezes nas páginas 48 e 49.

Unity Technologies. *Unity Manual*. 2015. Acesso em: 8 de Março 2015. Disponível em: <http://docs.unity3d.com/Manual/>. Citado 2 vezes nas páginas 35 e 36.

WANG, J.; LINDEMAN, R. W. *Unity Indie VRPN Adapter – UIVA*. 2015. Acesso em: 18 de Julho 2015. Disponível em: <http://web.cs.wpi.edu/~gogo/hive/UIVA/>. Citado 2 vezes nas páginas 43 e 48.

WARE, C. *Information Visualization: Perception for Design*. 3. ed. [S.l.]: Elsevier Science & Technology, 2012. (Interactive Technologies). ISBN 9780123814647. Citado na página 58.

WIJAYASEKARA, D.; LINDA, O.; MANIC, M. Cave-som: Immersive visual data mining using 3d self-organizing maps. In: *Neural Networks (IJCNN), The 2011 International Joint Conference on*. [S.l.: s.n.], 2011. p. 2471–2478. ISSN 2161-4393. Citado na página 57.

APÊNDICE A – Revisão Sistemática

Este Apêndice apresenta o planejamento e a condução da revisão sistemática da literatura executada, que obteve os materiais científicos para fundamento teórico e técnico deste trabalho.

Segundo Kitchenham et al. (2009), a revisão sistemática da literatura é a identificação, avaliação e interpretação de material científico para uma questão de pesquisa, tópico ou fenômeno de interesse. Existem muitas razões para a execução da revisão sistemática, as mais comuns são:

- Sumarizar evidências sobre algum tratamento ou tecnologia, para se obter dados empíricos dos benefícios e limitações de um método ágil específico;
- Identificar lacunas em uma pesquisa, a fim de sugerir áreas para maiores investigações; e
- Fornecer um *framework* ou base, a fim de posicionar novas atividades de pesquisa apropriadamente.

A revisão sistemática da literatura executada para este trabalho teve como diretrizes as definidas por Kitchenham et al. (2009), que apresenta e descreve os princípios básicos do planejamento e da condução de uma revisão sistemática da literatura.

A.1 Planejamento

De acordo com Kitchenham et al. (2009), a especificação das questões de pesquisa é a parte mais importante de qualquer revisão sistemática da literatura, uma vez que estas questões guiam por completo a metodologia da revisão. Após serem aprimoradas, teve-se como questões de pesquisa:

- Como é implementada a multiprojeção em ambientes virtuais gerados por motores de jogo?
- Onde são aplicados os ambientes virtuais multiprojetados gerados por motores de jogo?

A.2 Condução

Kitchenham et al. (2009) recomendam a construção de uma lógica de busca sofisticada por meio de termos, oriundos das questões de pesquisa, e operadores lógicos (ANDs e ORs), para buscar por materiais científicos que possivelmente atendam as questões de pesquisa. Após ser aperfeiçoada, teve-se como lógica de busca a apresentada na Figura 38.

Figura 38 – A lógica de busca utilizada.

```
( ("virtual reality") OR ("virtual environment") ) AND ("game engine") AND
( ("multiple projection") OR ("multiple screen") OR ("multi-projection")
OR ("multi-projector displays") OR ("CAVE") )
```

Fonte: Popolin Neto e Brega (2015b).

Kitchenham et al. (2009) intitulam estudos primários as produções científicas que respondem de forma parcial ou total as questões de pequisa. A Tabela 21 contém os resultado da aplicação da lógica de busca da Figura 38 nas bases científicas IEEE Xplore [1], ACM Digital Library [2], Scopus [3] e Science Direct [4].

Tabela 21 – Os resultados da aplicação da lógica de busca em quatro bases científicas.

Biblioteca Científica	Número de Estudos
IEEE Xplore	76
ACM Digital Library	70
Scopus	80
Science Direct	35
Total	261
Candidatos	44
Estudos Primários	40

Fonte: Popolin Neto e Brega (2015b).

Dentre o total de duzentos e sessenta e um estudos encontrados, quarenta e quatro foram candidatos a estudos primários, onde destes quarenta apresentaram a multiprojeção de AVs gerados por motores de jogo, e vinte e nove utilizam estes AVs imersivos em domínios de aplicação específicos, atendendo as questões de pesquisa. A aplicação da lógica de busca nas bases científicas foi realizada em novembro de 2013 e em fevereiro de 2015.

[1] http://ieeexplore.ieee.org/Xplore/home.jsp
[2] http://dl.acm.org/dl.cfm
[3] http://www.scopus.com/
[4] http://www.sciencedirect.com/

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 21 / 08 / 15

Assinatura