UNIVERSIDADE FEDERAL DO PARANÁ

FABIANO PAULICO STANGE

PROTÓTIPO DE AMBIENTE VIRTUAL EDUCACIONAL PARA ATIVIDADE TÍPICA DA CONSTRUÇÃO CIVIL BRASILEIRA

CURITIBA

2012

FABIANO PAULICO STANGE

PROTÓTIPO DE AMBIENTE VIRTUAL EDUCACIONAL PARA ATIVIDADE TÍPICA DA CONSTRUÇÃO CIVIL BRASILEIRA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciências, Curso de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Mecânica Computacional, Setores de Tecnologia e de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Sergio Scheer

CURITIBA

2012

TERMO DE APROVAÇÃO

FABIANO PAULICO STANGE

PROTÓTIPO DE AMBIENTE VIRTUAL EDUCACIONAL PARA ATIVIDADE TÍPICA DA CONSTRUÇÃO CIVIL BRASILEIRA

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Ciências no Curso de Pós-Graduação em Métodos Numéricos em Engenharia, área de Concentração em Mecânica Computacional, Setor de Tecnologia e Setor de Ciências Exatas, Universidade Federal do Paraná, pela seguinte banca:

Orientador: _____

Prof. Dr. Sérgio Scheer
Programa de Pós-Graduação em Métodos Numéricos em Engenharia, UFPR.

_____

Prof. Dr. Klaus de Geus
Programa de Pós-Graduação em Métodos Numéricos em Engenharia, UFPR.

_____

Profa. Dra. Cinthia Obladen de Almendra Freitas
Programa de Pós-Graduação em Informática Aplicada, PUCPR.

CURITIBA, 24 de outubro de 2012.

À minha querida esposa Cris e à minha filhinha Nina.

Aos meus pais.

E a Deus.

AGRADECIMENTOS

Ao meu Deus por minha vida.
À minha esposa Cris e à minha filha Nina pelo apoio, amor e compreensão.
Aos meus pais Antônio Stange e Suelí Stange por tudo.
Ao professor Sérgio Scheer pela orientação e paciência.
Aos professores Klaus de Geus, Mildred Hecke e José de Almendra Freitas Junior por dividirem comigo parte de seus conhecimentos.
Aos funcionários do CESEC, em especial à Maristela.
À CAPES pelo apoio financeiro.
Aos colegas do PPGMNE pela força e paciência e que contribuíram, de certa forma, para a realização deste trabalho. Em especial: ao Lucas, Jean Michael, Marcos Argenta, Tiago Buriol, Abimael Jr., Amanda Jarek, Juliano Scremim, Sandro Rodrigues, Fabio Balbo, Rudinei Bogo.
Aos amigos Everton, Mauricio, Bruno, Claudio, Fernando (Dindo) pelas horas de diversão e estudo.

You thought the leaden winter would
bring you down forever,
But you rode upon the steamer to the
violence of the sun.
And the colors of the sea blind your
eyes with trembling mermaids,
And you touch the distant beaches with
the tales of brave Ulysses.

Trecho da música *Tales of Brave Ulysses* da banda Cream (Eric Clapton, Jack Bruce e Ginger Baker).

O começo de todas as ciências é o espanto de as coisas serem o que são.

Aristóteles.

RESUMO

Atualmente, a indústria brasileira de construção está passando ao processo econômico que resulta em um polo atrativo para os jovens. Eles estão se esforçando para aprender profissões como Arquitetura, Engenharia E Construção. Eles pertencem a uma geração onde o computador está presente em suas vidas para acessar informações e conhecimento. Esta geração não é mais um receptor passivo de conteúdos transmitidos, mas um produtor do seu conhecimento através de mídias interativas nos dias de hoje. Mas, a educação no Brasil nas áreas da Arquitetura, Engenharia e Construção ainda estão baseadas em um paradigma onde o conhecimento está centrado na figura do educador. O educador brasileiro não explora os recursos que são familiares a esta nova geração. Isso ressalta a necessidade de mudar este paradigma, utilizando recursos da aprendizagem medida por tecnologias da informação e comunicação. Pesquisando contextos de aprendizagem em Arquitetura, Engenharia e Construção pode-se notar que a Realidade Virtual se apresenta como uma poderosa ferramenta educacional. O objetivo da dissertação de mestrado é o desenvolvimento de uma ferramenta para analisar e complementar o processo de ensino e aprendizagem. É apresentado um protótipo de ambiente virtual educacional para simular atividades típicas da construção civil brasileira que foi testado por alunos e docente do curso de graduação em Engenharia Civil da Universidade Federal do Paraná, demonstrando sua usabilidade e eficácia como ferramenta de treinamento e aprendizagem. Conclui-se que através da experiência com o ambiente virtual protótipo um aprendiz possa entender a tarefa simulada de maneira mais intuitiva e menos opressora do que as metodologias usuais em sala de aula, ao mesmo tempo em que visa incentivar os educadores a buscar pedagogias mais inovadoras e motivadoras.

Palavras-chaves: Realidade virtual, ambientes virtuais, construção, educação, treinamento.

ABSTRACT

Currently, Brazilian construction industry is going through to economic process which results in an attractive polo to young people. They are trying hard to learn professions such as Architecture, Engineering and Construction. They belong to a generation where the computer is present in their lives to access information and knowledge. This generation is no longer a passive recipient of contents transmitted, but a producer of his knowledge through interactive media nowadays. But, in Brazil, the education in the fields of Architecture, Engineering and Construction are still based on a paradigm where knowledge is centered on the figure of the educator. The Brazilian educator does not exploit the resources that are familiar to this new generation. This underscores the need to change this paradigm, by using learning resources as measured by information and communication technologies. Searching contexts of learning in Architecture, Engineering and Construction may be noted that Virtual Reality presents itself as a powerful educational tool. The aim of the master thesis is development of a tool for analyze and complement the teaching and learning process. It is presented a prototype of an educational virtual environment to simulate typical activities of the Brazilian civil construction which has been tested by students and a professor of the Civil Engineering course from Federal University of Parana, demonstrating its usability and efficacy as a tool for task training and learning. We conclude that through experience with the virtual environment prototype an apprentice can understand the simulated task in a more intuitive and less oppressive than the usual methodologies in the classroom at the same time as it seeks to encourage educators to search more innovative and motivating pedagogies.

KEYWORDS: Virtual reality, virtual environments, construction, education, training.

LISTA DE FIGURAS

LISTA DE SIGLAS

1D – Unidimensional (relativo a uma dimensão)

2D – Bidimensional (relativo a duas dimensões)

3D – Tridimensional (relativo a três dimensões)

API – *Application Programming Interface*

AV – Ambiente virtual

AVC – Ambiente virtual colaborativo

B-rep – *Boundary representation*

BUS – Barramento

CAD – *Computer Aided Design*

CAE – *Computer Aided Engineering*

CAVE – *Cave Automatic Virtual Environment*

CFD - *Computational Fluid Dynamics*

CG – Computação gráfica

CGS – *Constructive Solid Geometry*

CPU – *Center Processing Unity*

CRT – *Catode Ray Tube*

DLP – *Digital Light Processing*

EVL – *Electronic Visualization Lab*

HMD – *Head Mounted Display*

HTML – *Hyper Text Markup Language*

GIF – *Graphic Interchange Format*

LCD – *Liquid Crystal Display*

LCOS – *Liquid Crystal on Silicone*

LED – *Light Emitting Diode*

NASA – *National Aeronautics and Space Administration*

OLED – *Organic Light-Emitting Diode*

RA – Realidade Aumentada

RGB – *Red*, *Green*, *Blue* ou Vermelho, Verde, Azul

RPG – *Role Playing Game*

RV – Realidade Virtual

VC – Visualização científica

VCASS – *Visually Coupled Airborne simulator system*

VCS4D – *Virtual Construction Simulator 4D*

VRML – *Virtual Reality Modeling Language*

VTK – *Visualization Toolkit*

SUMÁRIO

1 INTRODUÇÃO

O cenário atual da indústria da construção civil brasileira apresenta um crescimento econômico graças às ações do Governo Federal, através de estratégias adotadas de interesse social, tais como o Programa de Aceleração do Crescimento (PAC) e do Programa Minha Casa, Minha Vida (PMCMV). Um dos efeitos percebidos por tal crescimento econômico está no aumento da procura pelas profissões relacionadas à Arquitetura, Engenharia e Construção (AEC), principalmente a profissão de engenheiro civil. Tais profissões se tornaram alvo de jovens que aspiram a entrar no mercado de trabalho brasileiro. Isso pode ser confirmado, por exemplo, quando se observa o aumento da demanda para o curso de Engenharia Civil a ponto de ser o curso mais competitivo em certas instituições de ensino superior no Brasil (CEF, 2012; CORREIO BRAZILIENSE; 2012, TÉCHNE; 2012; TRIBUNA DO NORTE, 2012; VEJA(b), 2012; VEJA(a), 2009).

Esses jovens aprendizes pertencem a uma geração chamada nativos digitais (PRENSKY, 2001) onde o computador está presente em suas vidas, facilitando o acesso à informação e ao conhecimento. O nativo digital não é mais um receptor passivo do conteúdo transmitido, mas sim um construtor do seu conhecimento através de mídias interativas que antes eram de acesso exclusivo e agora estão se tornando comuns. Eles estão mais dispostos a ser influenciados pelas novas tecnologias, porque a cada dia eles recebem uma quantidade considerável de novos dispositivos eletrônicos e aplicativos.

A educação brasileira em AEC ainda continua fundamentada em um paradigma onde o conhecimento está centrado exclusivamente na figura do instrutor/educador/professor e, geralmente, este não explora os recursos tecnológicos existentes para aumentar a motivação dos alunos nativos digitais em buscar o conhecimento além daquele oferecido pelo instrutor/educador/professor na sala de aula. Ainda, os campos de AEC já não são mais os mesmos quando comparados aos de 30 anos atrás, destacando a necessidade de um aprendizado mais interativo, a construção do pensamento crítico, a motivação de criatividade para resolver problemas, em contrapartida de um aprendizado com uma perspectiva

passiva. E tais requisitos ou habilidades nem sempre são adquiridos na universidade, mas quando o aluno já está inserido no mercado de trabalho (CARVALHO, 2003; HUTCHINSON et al, 2003;POMPEU, 1999).

Assim, há uma necessidade de mudança do paradigma utilizado para o ensino e aprendizagem em AEC. Uma possível abordagem está na utilização do computador como um apoio à aprendizagem. O processo educacional deve ser atualizado para resolver a resistência entre o ensino e a aprendizagem, além de ser cauteloso com as diferenças entre a interação entre indivíduo/ indivíduo e indivíduo /máquina, assim como o diálogo entre instrutor/educador/professor e o aprendiz através do uso adequado da tecnologia, visando maior qualidade para o próprio profissional, a instituição e o curso (CARVALHO, 2003, HUTCHINSON et al, 2003, PASQUALOTTI, 2000; POMPEU, 1999).

Paralelo a este contexto, os Ambientes Virtuais (AV) que usam os recursos da Realidade Virtual (RV) estão começando se tornar mais acessíveis graças aos avanços na indústria de computadores e desta forma, possibilitando sua utilização no ensino destinado a AEC. Os AV possuem tudo o que as outras ferramentas computacionais gráficas podem oferecer, mas com mais motivação, pois busca proporcionar o estímulo e a motivação ao aprendiz, proporcionando a este um exame mais detalhado do objeto em estudo contido em um AV. Proporciona ainda, a participação das pessoas com deficiência, agrega uma maior liberdade de tempo para o aluno em troca de horas fixas em salas de aulas, motiva a participação ativa em vez da passiva e permite uma experiência de aprendizagem multiperspectival. É importante lembrar que o RV não vai resolver todos os problemas relacionados à educação, mas é possível mostrar ao educador/ instrutor/professor a existência de opções além da sala de aula (SEABRA; SANTOS, 2005; CARVALHO, 2003; PASQUALOTTI, 2000).

O objetivo do uso de RV é aumentar a performance humana (MCLELLAN, 1996). E esta performance se relaciona com o processo cognitivo humano. Cognição é o ato ou processo de conhecer envolvendo atenção, percepção, memória, raciocínio, juízo, imaginação, pensamento e linguagem. É um processo complexo que é predicado da interação senso-motor dos indivíduos e seus sistemas neurológicos. Isto é possível desde que a RV seja usada junto com um educador/instrutor/professor em guiar os aprendizes tal como um *dugeon master* em um jogo de RPG guia os players (KATHELEEN; FLANNERY; KRAUCHUNAS, 2000;

MCLELLAN, 1996). Atualmente se busca no conceito de *serious games* simular, treinar e educar com o uso de consoles e pc transferindo experiencias positivas na construção do conhecimento.

A justificativa deste estudo está na necessidade de compreender novos paradigmas de ensino e aprendizagem em AEC. A procura por tais paradigmas motivou o desenvolvimento de uma ferramenta para ser utilizada como complemento da metodologia aplicada ao ensino e aprendizagem de atividades típicas em construção civil, harmonizando estímulo e motivação ao processo de construção do conhecimento realizado pelo próprio aprendiz sobre a supervisão da figura do educador/instrutor/professor. Conclui-se que esta ferramenta proporciona uma aprendizagem mais intuitiva e menos opressora que as antigas metodologias abordadas em sala de aula, e também estimular os educadores em buscar pedagogias mais motivadoras e inovadoras.

1.1 OBJETIVO DA PESQUISA

O objetivo desta pesquisa é desenvolver um protótipo de um ambiente virtual através de ferramentas computacionais disponíveis para AEC como um instrumento complementar para a aprendizagem de atividades típicas de construção civil, além de ser adaptado para a nova geração que almeja conhecimento relacionados a AEC.

O ambiente virtual protótipo foi desenvolvido para simular uma atividade: a montagem de uma fôrma de madeira para moldar um pilar em concreto armado situado em um canteiro de obras virtual.

1.2 OBJETIVOS ESPECÍFICOS

A implementação do ambiente virtual protótipo voltado para o ensino e aprendizagem em AEC será possível quando forem explorados os seguintes objetivos específicos:

a) Investigar os conceitos e técnicas de realidade virtual para auxiliar no desenvolvimento do ambiente virtual protótipo voltado ao ensino de atividades típicas da construção civil.
b) Estudar a teoria do construtivismo para fundamentar a aprendizagem e o conhecimento através do uso de ambientes virtuais.
c) Estudar atividades típicas de construção civil brasileira para serem simuladas pelo ambiente virtual protótipo.
d) Investigar ferramentas computacionais que possam servir em conjunto para o desenvolvimento de um protótipo de um ambiente virtual que simule uma atividade típica da construção civil brasileira.
e) Desenvolver um ambiente virtual que possa ser acessível economicamente, através do conjunto de ferramentas computacionais investigadas para fornecer certa facilidade ao educador/instrutor/professor quando desenvolver ou aplicar a simulação ao invés de gastar horas em programação e sincronização de ferramentas computacionais e de dispositivos avançado para a simulação.
f) Realizar testes com um grupo de alunos e professores para validar o ambiente virtual protótipo como ferramenta de aprendizagem.

1.3 ESTRUTURA DA DISSERTAÇÃO

A dissertação está organizada em seis capítulos, sendo que neste primeiro capítulo foi apresentada a justificativa da pesquisa e seu objetivo.

O segundo capítulo apresentada conceitos pertinentes à visualização científica e a computação gráfica, abordando uma breve retrospectiva histórica, uma descrição do processo de visualização por computação e os modelos utilizados. Por último é apresentado os componentes de visualização por computação.

O terceiro capítulo descreve os conceitos relacionados à realidade virtual, apresentando uma classificação e uma descrição dos dispositivos usados em uma RV e uma retrospectiva histórica. Ainda, apresenta uma descrição da teoria construtivista e seu relacionamento com ambientes virtuais e o processo de aprendizagem, apresentando também o emprego destes ambientes virtuais no ensino e aprendizagem em AEC.

O quarto capítulo apresenta os aspectos adotados para o desenvolvimento do ambiente virtual protótipo, descrevendo as ferramentas computacionais utilizadas, a metodologia de implementação e um estudo sobre a atividade típica da construção civil adotada na simulação.

O quinto capítulo apresenta a metodologia utilizada para a realização dos testes com o ambiente virtual e os resultados.

O sexto capítulo apresenta a conclusão da pesquisa, as considerações finais e sugestões para trabalhos futuros.

1.4 LIMITAÇÃO DA PESQUISA

Para o experimento do ambiente virtual protótipo foi considerado um grupo limitado de alunos dos primeiros anos do curso de graduação de Engenharia Civil da Universidade Federal do Paraná e uma entrevista com um professor especialista sobre a Construção Civil da mesma instituição.

Foi adotada apenas a atividade de montagem de fôrma para moldar u um pilar em concreto armado para ser simulada, pois, se a atividade simulada conseguir ser validada, isto é um indício que o mesmo pode ser considerado para outras atividades semelhantes.

2 VISUALIZAÇÃO CIENTÍFICA E COMPUTAÇÃO GRÁFICA

Desde o século X os cientistas utilizavam gráficos e imagens como apoio à suas pesquisas científicas, principalmente quando era necessária a representação de dados. Descobertas na matemática, astronomia, mecânica, termodinâmica, biologia, medicina entre outras áreas do conhecimento científico ganharam uma maior compreensão graças à visualização. A citar: estudo de órbitas planetárias por representação de elipses, descrição vetorial para estudar movimentos, o uso da geometria descritiva no auxilio de projetos de arquitetura e engenharia são exemplos da aplicação da visualização para melhorar o entendimento sobre a informação (BRODLIE *et al*, 1995). Assim, o capítulo aborda o conceito de visualização e sua relação com a ciência através da computação gráfica, apresentando uma breve retrospectiva histórica e os processos e componentes de visualização por computação.

2.1 CONCEITOS

Essa seção apresenta os conceitos pertinentes sobre Computação Gráfica e Visualização Científica. Ainda, são apresentadas a taxonomia da Computação Gráfica e as áreas que usam ela como base.

2.1.1 Computação gráfica

Em Azevedo e Conci (2003) a computação gráfica (CG) está conceituada como uma matemática artística. Sob a perspectiva da arte a CG é uma ferramenta não convencional para o artista trabalhar suas técnicas de desenho. Na perspectiva matemática a CG pode ser entendida como uma linguagem entre o homem e a

natureza pela simulação numérica computacional ao invés de um aglomerado de números nas equações. Ainda, no trabalho dos autores citados está apresentada uma definição formalizada pela *International Organizations for Standardization*: *"conjunto de ferramentas e técnicas par converter dados para ou de um dispositivo gráfico através do computador"* (AZEVEDO; CONCI, 2003, p. 3).

Computação gráfica (CG) está definida segundo Velho e Gomes (2001, p.1) como a *"área que estuda os processos computacionais envolvendo modelos geométricos e imagens digitais."* Complementando esta definição, a CG pode ser representada por uma taxonomia onde modelos geométricos podem ser transformados em imagens digitais pelos processos de síntese de imagens. O processo inverso é chamado de análise de imagem. Os modelos geométricos e imagens digitais são tipos distintos e podem ser criados ou modificados computacionalmente. Se por um lado temos a modelagem geométrica com o foco de criar dados a serem apresentados sob a forma de imagens e o processamento de imagens para manipular e modificar estas imagens, a visão computacional traça um caminho inverso ao gerar um modelo geométrico quando se visualiza um dado computacional sob o formato de imagem. A figura 2.1 apresenta a taxonomia da CG.

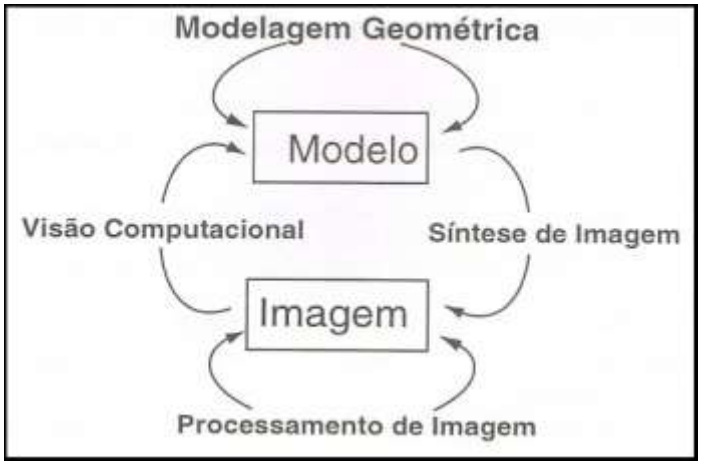

FIGURA 2.1 - TAXONOMIA DA COMPUTAÇÃO GRÁFICA
FONTE: Velho e Gomes, 2001.

Quando analisada a taxonomia é possível relacionar a CG com as áreas do entretenimento, planejamento e projetos de arquitetura e engenharia, interfaces gráficas e visualização científica. No entretenimento a CG promove suporte para a criação de efeitos especiais em cinema assim como na indústria de jogos eletrônicos. Na engenharia e arquitetura as ferramentas como CAD (*Computer Aided*

Design) e CAE (*Computer Aided Engineering*) utilizam os conceitos de CG. Na aplicação em interfaces gráficas são estudadas formas que a CG pode auxiliar na interface homem-máquina. A visualização científica usa os recursos de CG para visualizar dados oriundos de simulações científicas. Assim, a computação gráfica serve como base para quatro grandes áreas de estudo: sistemas interativos 3D, fotorrealismo, mundo virtual e simulação de iluminação (figura 2.2).

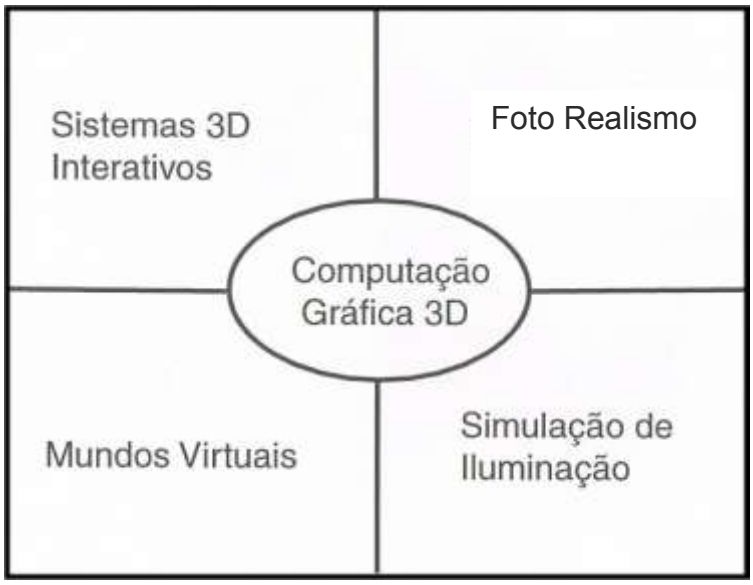

FIGURA 2.2 - ÁREAS QUE A COMPUTAÇÃO GRÁFICA SERVE COMO BASE
FONTE:Velho e Gomes,2001.

2.1.2 Visualização científica

A visualização científica (VC) pode ser entendida como uma área da computação cujo foco está no estudo e na pesquisa de técnicas de computação gráfica para tratar visualmente dados científicos procedentes de medições físicas e simulações numéricas. Estuda ferramentas de visualização para auxiliar o desenvolvimento de programas e sistemas dando suporte ao seu desenvolvimento, depuração, verificação em conjunto com formas para visualizar informações complexas, multidimensionais, volumosas sem geometria própria ou natureza gráfica (OLIVEIRA; MINGHIM, 1997). É uma ferramenta de descoberta e compreensão para

entender a estrutura global do campo das variáveis e suas interconexões, analisando qualitativamente a natureza das soluções numéricas através da computação gráfica (McCORMICK *et al,* 1987). E tais dados são oriundos das áreas de conhecimento como medicina (tomografia), astronomia (visualização de campos gravitacionais), engenharias (visualização de campos de tensões), química (visualização de cadeias de moléculas) e demais ciências (ADAIME, 2005).

Visualização científica também está definida como a transformação de informações em imagem para excitar o principal sentido humano, a visão, e classificando a tecnologia de visualização conforme o contexto e natureza dos dados (SCHROEDER *et al*, 2004). Esta classificação é necessária para avaliar o tratamento da imagem sem comprometer sua eficácia na transmissão visual ou omitir informações relevantes, ou seja, não basta representar os dados por imagens com apelo estético, mas também com a capacidade de transmitir conteúdo científico (GLOBUS; RAIBLE, 1994).

Brodlie *et al* (1992) define que VC está na relação entre exploração e percepção de dados cujo objetivo é prover maior entendimento na investigação da informação ao melhorar a percepção destes dados confiando na habilidade visual dos humanos. Ao dispor de técnicas de visualização, grandes volumes de dados multidimensionais como campos escalares (valores escalares atribuídos aos pontos em um domínio 2D ou 3D), campos vetoriais (para cada ponto pertencente a um domínio 2D ou 3D está atribuído um vetor) e campos tensoriais (um tensor de ordem igual ou maior que 2 está atribuído a um ponto pertencente a um domínio 2D ou 3D) podem ser analisados melhor. Dentre estas técnicas é pertinente citar as seguintes: extração de contorno, *rendering* de volumes, superfícies elevadas e os planos de corte para visualização de campos escalares; campos de setas, glifos, linhas de fluxos, trajetória de partículas e textura para representar campos vetoriais; *hiperstreamlines* e glifos para representar os campos tensoriais.

2.2 BREVE RETROSPECTIVA HISTÓRICA

As técnicas de visualização se baseiam na geometria euclidiana, que orientou o desenvolvimento do mundo até o século XVIII (AZEVEDO; CONCI, 2003). No século XII a China já usava sistemas de coordenadas ortogonais para mapeamento (BURIOL, 2006). A perspectiva da imagem visualizada foi desenvolvida pelo arquiteto Fellipo Brunelleschi em 1425, século XV.

Em 1594, Pierre Bruinsz usou pontos para demarcar posições de mesma profundidade em um rio através de linhas. Unindo os pontos que possuíam mesmo valor através de uma linha, esta técnica hoje é conhecida como isolinhas.

Johann Beyer, em 1603 desenvolveu o sistema de coordenadas esféricas. Rene Descartes, em 1637 formulou a geometria analítica e reintroduziu o sistema de coordenadas ortogonais na matemática (AZEVEDO; CONCI, 2003). Em 1663, Christopher Wren construiu o primeiro artefato de medição de temperatura e nível da chuva que desenhava um gráfico automaticamente. Athanasius Kicher apresentou um trabalho, em 1665, sobre os fluxos das correntes marítimas através de linhas de fluxo, que hoje são conhecidas como *streamlines*. Em 1689, Edmund Haley usava sistema ortogonal de coordenadas para relacionar de pressão atmosférica e altitude (BURIOL, 2006).

Gaspard Monge, no século XVIII desenvolveu a geometria descritiva que se tornou um ramo da geometria. Ainda, Euler desenvolveu uma relação entre o número de vértices, arestas e faces de poliedros que hoje é conhecida com a fórmula de Euler-Poincaré (AZEVEDO; CONCI, 2003; MÄNTYLÄ, 1988). Gottfried Hessel, em 1741 utilizou mapas coloridos para realçar dados e diferenciar informações para estudar regiões onde populações falavam diferentes línguas. Em 1779, Johann Heinrich Lambert usava traçados de linhas para mostrar a variação de temperatura abaixo da superfície terrestre. Em 1782, August Crome usou ícones para demonstrar a distribuição de produtos na Europa. E em 1785, Wiliam Playfair demonstrava dados de importação e exportação através de linhas e gráficos coloridos (BURIOL, 2006).

No século XIX Joseph Sylverster desenvolveu o conceito de matriz e a notação matricial que é usada até hoje na computação gráfica (AZEVEDO; CONCI,

2003). A representação de informações no espaço tridimensional cujo desenho era uma elevação de dados para coordenada bidimensional foi feita por Luigi Perozzo, em meados de 1875, e batizado de *stereogram*. Osbourne Reynolds, em 1883 demonstrou visualmente o escoamento laminar para o turbulento através de traços de partículas. A primeira representação de volumes através de dados foi feito por Elihu Thompson, em 1896, ao representar um rato por um par de fotografias de raios-X estereoscópicas (BURIOL, 2006).

O século XX foi marcado por um grande desenvolvimento da computação e com isto, a própria computação gráfica. Segue como o surgimento do padrão de 24 imagens por segundos da indústria cinematográfica em 1927, o primeiro computador foi construído em 1930 e em 1956 surgia o computador com transistores. O termo computação gráfica foi designado por Hudson da empresa Boeing em 1959. Em 1974 foi realizada a primeira conferência do grupo SIGGRAPH (*Special Interest Group on Graphics*) (AZEVEDO; CONCI, 2003). Em 1987 o painel *Visualization in Scientific Computing*, SIGGRAPH foi considerado com o marco inicial da visualização científica (BURIOL, 2006).

2.3 PROCESSO DE VISUALIZAÇÃO

O processo de visualização por computação abrange a exploração de dados através da construção de um modelo empírico, a transformação desse modelo em objetos gráficos e a exposição dos objetos gráficos através de dispositivos (OLIVEIRA e MINGHIM, 1997). Objetos gráficos são entendidos como um subconjunto de suporte geométrico, ou seja, existe uma relação geométrica e topológica[1] e uma função de atributo para especificar propriedades em um dado ponto pertencente ao objeto gráfico (VELHO e GOMES, 2001).

[1] A topologia estuda os conceitos das relações entre objetos investigando sua conexidade, separabilidade (estudo de vizinhança entre pontos) e compacidade (tamanho de um conjunto finito) e a teoria dos nós, tratando de objetos e suas relações sem haver preocupação com sua dimensão ou resistência elástica (DAMBRASCO, 1977). A geometria é o estudo do espaço e como as formas podem ocupá-lo e utiliza números e símbolos para descrever as formas e a relação com o espaço. (LANDAVERDE, 1970).

Os dados podem pertencer a um domínio unidimensional ou multidimensional e geralmente são obtidos por medidas experimentais ou simulações numéricas e apresentados de forma discreta. Como não se conhecem as informações em todos os pontos do domínio, procede-se na organização destes dados com o uso de uma grade de células chamada de malha. Quando as relações topológicas e geométricas da malha formam um tramo bem definido (existência de um padrão de conectividade) então a malha é dita estruturada, caso contrário, a malha é dita não estruturada. Algoritmos que usam funções de interpolação ou aproximação podem ser empregados na determinação de valores não conhecidos em pontos arbitrários da malha, deste modo, evitam introduzir erros significativos (ADAIME, 2005).

Na etapa de transformação, ou processamento são usados algoritmos de visualização (filtros e mapeadores) para modelagem geométrica e/ou processamento de imagem digital, resultando num formato conveniente de apresentação dos dados do domínio.

Quando os dados são transformados em objetos gráficos, estes são representados por sistemas de visualização através de técnicas de renderização disponíveis em bibliotecas, APIs e *toolkits*.

As bibliotecas gráficas ou *Application Programming Interfaces* (API) facilitam as representações de dados por computação gráfica, pois geram desempenho e economia em códigos computacionais de interface com o hardware gráfico. As APIs de destaque são OpenGL (atualmente da Khronos Group) e DirectX (Microsoft).

As ferramentas (*toolkits*) provém ao usuário e/ou programador a facilidade na manipulação de visualização através de rotinas já implementadas. Cabe destacar os seguintes *toolkits*: AVS *Data Visualization* (*Advanced Visual System*); AmiraVR (*Visage Imaging Company*) especializado em ambientes imersivos de realidade virtual; OpenDX que é uma versão livre do IBM *VIsualization Data Explorer*; e o *Visualization Toolkit* (VTK) da Kitware Inc. que é portável, estável e bem documentado e vem se destacando no meio científico (BURIOL, 2006).

2.3.1 Modelo de fluxo de dados

Um paradigma de processo de visualização adotado em diversos sistemas é o modelo de fluxo de dados (*daflow*) apresentado em 1989 por Craig Upson (BRODLIE, 1993; UPSON, 1989). Este modelo consiste em um conjunto de módulos servindo tanto para entrada de dados para serem transformados, bem como para saída de dados, já transformados. Quando estes módulos são combinados formam um *pipeline* de visualização, ou seja, uma sequência lógica para entrada, transformação e saída de dados até ser representado por uma imagem a ser apresentada em um dispositivo gráfico.

Este modelo orientado a dados possui os seguintes módulos: filtro, mapeador e *rendering*. O filtro é um módulo que extraí as informações relevantes dos dados nativos através de operações típicas como interpolação, suavização, extração de contorno entre outras. No módulo de mapeamento as informações extraídas dos dados nativos são transformadas em geometria e utilizam operações para criar volume, cor e contorno, por exemplo. O *rendering* é um módulo que transforma a geometria em imagem para ser mostrada num dispositivo gráfico por meio de cálculo de superfícies visíveis e iluminação. A figura 2.3 apresenta o pipeline do modelo de fluxo de dados.

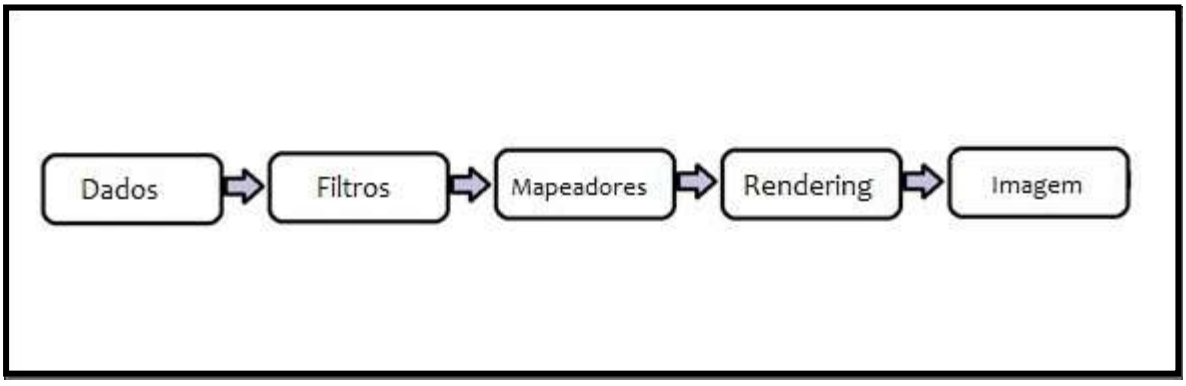

FIGURA 2.3 - PIPELINE DO MODELO DE VISUALIZAÇÃO
FONTE: Adaptado de Brodlie (1993).

Nos dispositivos gráficos serão apresentadas as imagens para serem visualizadas conforme o *pipeline* do modelo de visualização. Esses possuem um número fixo de *pixels* disposto em forma de uma matriz. Uma vez que são chamadas

as instruções no CPU (*Center Processing Unit*) relacionadas aos processos de visualização ocorre a transferência de informação para o *frame buffer* através do barramento (BUS). O *frame buffer* consiste em uma região da memória que possui tamanho suficiente para armazenar os valores atribuídos para cada pixel, por exemplo, informações sobre core e brilho. Pode ser uma memória física que pertence ao próprio dispositivo gráfico ou pertencer ao computador cujo dispositivo gráfico está conectado. O gerenciamento das informações que são armazenadas no *frame buffer* para uma superfície de visualização (*display)* ocorre no controlador de vídeo (*scan controler*) que avalia o endereço de cada informação relativa a cada *pixel* e assim controla a intensidade de cor no circuito de conversão do *display* (HILL, 2000). A figura 2.4 apresenta uma arquitetura simples de um sistema gráfico.

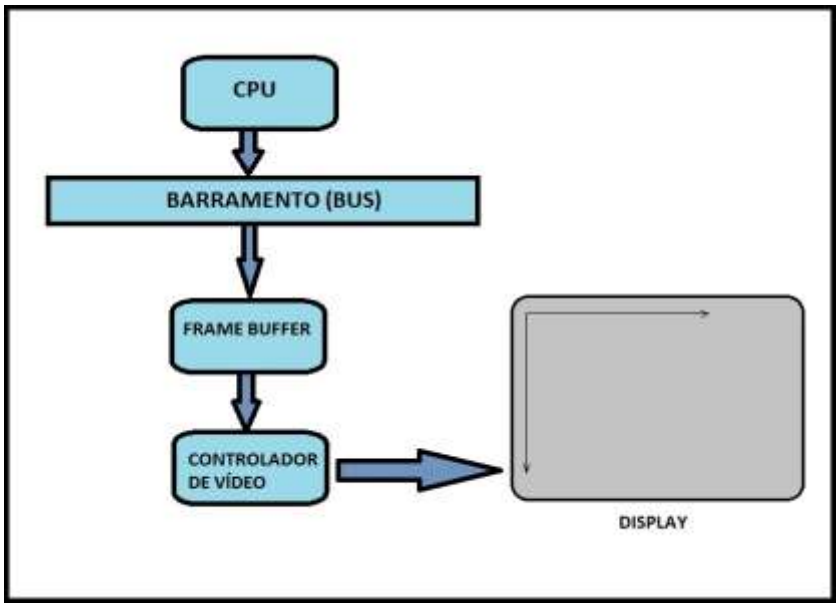

FIGURA 2.4 – ARQUITETURA SIMPLES DE UM SISTEMA GRÁFICO
FONTE: Adaptado de Hill (2000).

2.3.2 Modelo orientado a objetos na computação gráfica

Conforme Mizrahi (2006) a orientação a objetos é uma metodologia de análise e programação para sistematizar e estruturar dados computacionais em

novos tipos de dados pela combinação de dados, funções e mesmo outros objetos. Essa metodologia é baseada em herança, polimorfismo e sobrecarga.

A herança é a possibilidade de formar novos objetos por base de um objeto preexistente. O polimorfismo, por sua vez, é o uso do mesmo nome de um objeto, mas para outra estruturação de dados. E a sobrecarga é a utilização de um mesmo símbolo para executar tarefas distintas.

Bibliotecas e algoritmos de visualização podem ser implementados por esse modelo para estruturar dados (SCHOROEDER, 2004), e com a vantagem de se reaproveitar códigos usando herança e polimorfismo (BATTAIOLA; SOARES[2], 1998 *apud* BURIOL, 2006).

2.4 COMPONENTES DE VISUALIZAÇÃO POR COMPUTAÇÃO

Mäntylä (1988) define modelo como uma construção artificial de um objeto para facilitar a observação daquilo ao qual foi sujeito a modelagem. O emprego de um modelo oferece facilidade quando se analisa sua contraparte real. Assim como existem os modelos matemáticos que são empregados em diversos campos da ciência para representar o comportamento de algum fenômeno natural por meio de equações existem também os modelos que representam objetos reais através da sua aparência. E mesmo os desenhos técnicos de engenharia podem ser vistos como modelos. Na computação um modelo geométrico é um conjunto de dados alocados na memória do computador para representar a geometria de um objeto no mundo real, ou seja, gera um conjunto de componentes com geometria e aparência bem definidas existindo uma conectividade entre estes componentes tais como encontrados em estruturas arquitetônicas e de engenharia, moléculas e outras estruturas químicas, estruturas geográficas (FOLEY *et al,* 1996).

Ao transformar dados em modelos geométricos é necessário usar a modelagem geométrica para designar sua forma visual e assim, pelo estímulo da visão melhorar a compreensão de sua estrutura ou comportamento, além de

[2] BATTAIOLA, A. L.; SOARES, L. P. Estudo e Uso Exploratório de Ferramentas de Visualização Científica. In: SEMANA DE INFORMÁTICA DA UFBA, 7., 1998, Salvador. Anais... Salvador: UFBA, 1998. p. 16-30.

proporcionar manipulação e alteração dos dados. Velho e Gomes (2001) analisam a modelagem geométrica em quatro níveis. O primeiro nível é caracterizado por universo ou mundo real onde se encontra o objeto sujeito à modelagem. No segundo nível é adotada uma geometria para representar o objeto ao qual se está modelando. No terceiro nível é desenvolvido o conjunto de parâmetros para a geometria do modelo. O último nível ocorre uma implementação do modelo por uma estrutura de dados, por exemplo, *half-edge* que interliga faces, arestas, vértices e *loops* de um sólido por meio de uma lista duplamente encadeada (MÄNTYLÄ, 1988).

Os modelos geométricos usam os fundamentos da geometria projetiva para a descrição das formas visuais que ocupam o espaço tridimensional e seguem a geometria Euclidiana, incluindo as operações de transformações lineares como escalamento (alteração da dimensão), reflexão, cisalhamento (distorção da forma) e translação. E ainda, a possibilidade de projeções visuais ortogonais e perspectivas de uma maneira unificada, ou seja, uma matriz engloba essas transformações e projeções. Os modelos geométricos destes objetos gráficos podem ser descritos por funções matemáticas tanto na forma paramétrica como na forma não-paramétrica ou ainda, por meio de estruturas de dados para representar tais modelos através de malhas discretas.

2.4.1 Geometria e representação

Adotado um modelo matemático para a representação geométrica dos objetos gráficos a próxima etapa consiste em representá-los através de primitivas geométricas, ou simplesmente primitivas. Primitivas são entidades geométricas simples e fáceis de representar tais como o ponto coordenado e a reta que une estes pontos, e descrevem formas poligonais ou poliédricas. As demais formas bidimensionais e tridimensionais são obtidas por conjuntos de primitivas conectadas. Na computação gráfica podem ser consideradas como primitivas as seguintes formas tridimensionais: esfera, cone, cilindro, cubo, toro (VELHO; GOMES, 2001).

Mäntylä (1988) apresenta uma classificação para a representação de modelagem: modelos de decomposição, modelos construtivos e modelos de

contorno. Modelo de decomposição representa formas geométricas por uma operação de “colagem” entre entidades geométricas simples. A figura 2.5 demonstra as possíveis representações por decomposição de um objeto gráfico. A enumeração da ocupação espacial (figura 2.5.a) representa um objeto geométrico por entidades denominadas *voxels*[3]. A subdivisão do espaço ocupado representa o objeto geométrico através de uma relação hierárquica na forma célula-mãe e célula-filha usando quadtree/octree[4] (figuras 2.5.b e 2.5.c respectivamente) ou árvores binárias[5]. Representação por decomposição em células usa entidades poligonais ou poliédricas para dar forma ao objeto geométrico (figura 2.5.d).

O modelo construtivo também é representado por conjuntos de entidades geométricas simples, mas as operações que combinam tais entidades são mais sofisticadas que no modelo de decomposição. Essas operações, conhecidas por operações booleanas envolvem união, subtração e intersecção. A representação por CSG (*constructive solid geometry*) ou geometria sólida construtiva gera objetos geométricos complexos por combinações de simples geometrias tridimensionais como esferas, cubos, cilindros, cones (figura 2.6).

No modelo de contorno, também denominado de *B-rep* (*Boundary representation*), a representação acontece quando uma superfície orientada molda o contorno do objeto geométrico tridimensional distinguindo exterior do interior. Essa superfície é dividida por um número conveniente de faces poligonais. As faces são limitadas por arestas que por sua vez, são delimitadas por vértices. A representação incide na descrição topológica e geométrica. A descrição topológica define as relações entre os vértices, arestas e faces. A descrição geométrica refere-se ao

---

[3] *Voxel*: *Volumetric element* é um elemento de volume análogo ao *pixel*. *Pixel* é uma palavra criada da combinação, em inglês, do termo *picture element*. O *pixel* pode ser entendido como uma unidade lógica básica da imagem, que regula informações sobre cor e brilho. Sua dimensão física de um pixel depende da configuração da resolução da tela. Quando a resolução de uma tela está em máxima, a dimensão física do pixel equivale à dimensão unitária da tela. Outras configurações de resolução englobam mais pontos físicos da tela ao pixel (AZEVEDO; CONCI, 2003).

[4] Quadtree e octree são formas de subdivisão recursiva em células de mesma dimensão em 2D e 3D, respectivamente. Quando inspeção na célula-mãe não atinge o critério de preenchimento (cheio ou vazio) ocorre a divisão desta em quatro células-filhas (no caso 2D) ou oito células-filhas (no caso 3D). Isto se sucede até atingir um critério de parada (MÄNTYLÄ, 1988).

[5] Árvore binária é uma forma de divisão recursiva em que uma célula-mãe é divida em duas células-filhas. Sua vantagem em relação às quadtree e octree está no fato de seu formato ser menor (MÄNTYLÄ, 1988).

posicionamento no espaço das entidades que formam o objeto geométrico. A figura 2.7 mostra um exemplo de representação do modelo de contorno.

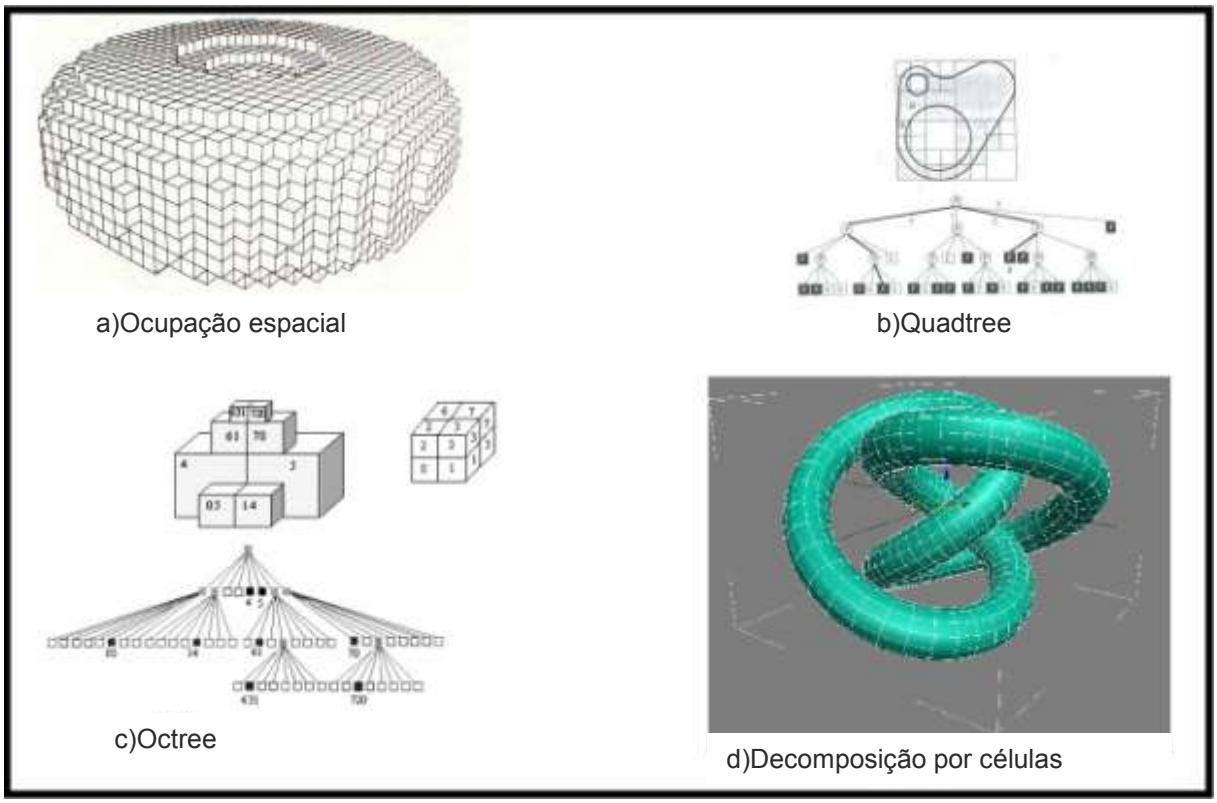

FIGURA 2.5 - MODELOS DE DECOMPOSIÇÃO
FONTE: Adaptado de Mäntylä (1988).

Os modelos podem ser combinados para representar a geometria dos objetos gráficos formando um modelo hibrido de representação. Recomenda-se a leitura do livro *An introduction to solid modeling* do autor Martti Mäntylä (1988) para maiores esclarecimentos sobre os sólidos realizáveis, modelos de representação e como combiná-los e sua implementação em linguagem de programação.

2.4.2 Cor e transparência

A cor é entendida como a sensação percebida pelo olho humano em resposta à radiação eletromagnética através do sentido da visão humana, que consegue captar uma faixa de comprimento de onda entre 400 a 700 nanômetros (HILL, 2000). Isto envolve a percepção do matiz, saturação e brilho.

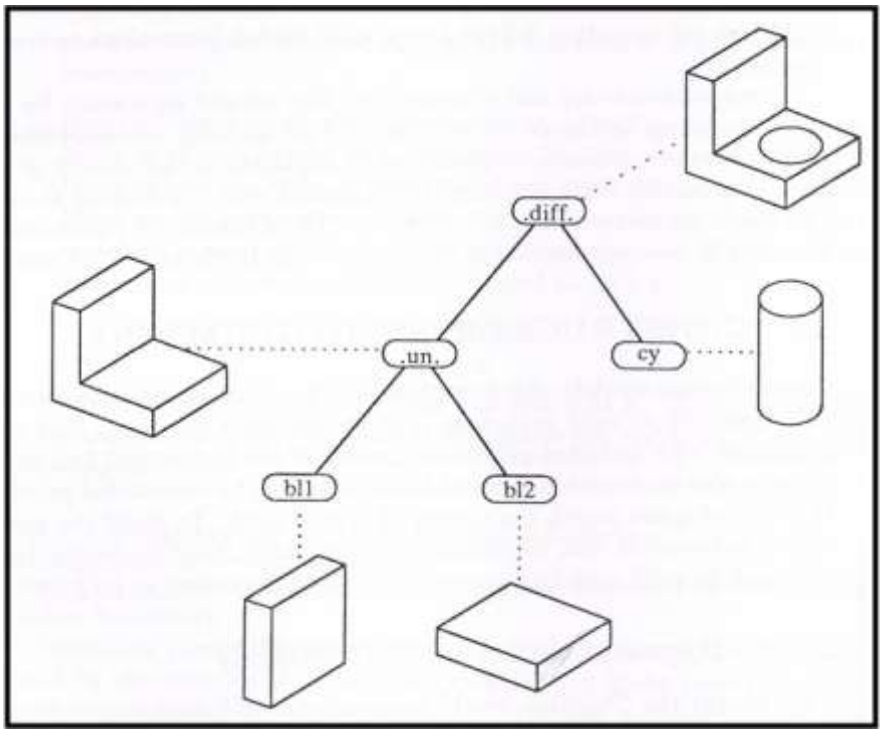

FIGURA 2.6 - MODELO CSG
FONTE: Adaptado de Mäntylä (1988).

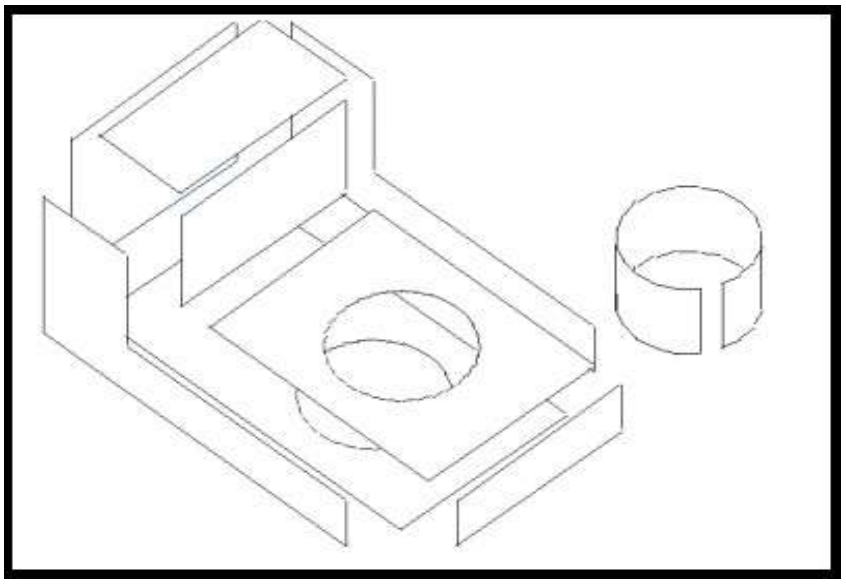

FIGURA 2.7 - MODELO DE CONTORNO
FONTE: http://www.cad-tutor.com/corsi/mod/resource/view.php?id=1735

O matiz é a forma que se pode classificar uma cor em termos de três tipos básicos de cores visíveis pelo homem: vermelho, verde e azul. A saturação é o grau

de pureza de uma cor quando esta se encontra diluída no branco. O brilho diz respeito ao nível de luminância que uma cor possui (AGOSTON, 2005).

O sistema mais usual de cor é modelo RGB (*Red, Green, Blue,* ou vermelho, verde, azul) que formam uma base para informar a cor para um *pixel*, conforme a combinação delas. Esta combinação pode ser representada por um vetor onde cada coordenada está associada uma intensidade para o vermelho, para o verde e para o azul, em um intervalo entre 0 e 1 ou 0 e 255. Outros sistemas podem ser usados, por exemplo, o sistema CMY (*Cyan, Magenta, Yellow*, ou ciano, magenta, amarelo) também baseado em uma combinação linear de cores; o sistema HSV (*Hue, Stauration, Value*, ou matiz, saturação, brilho); o modelo XYZ que está fundamentado nas medidas de refletância espectral (FOLEY *et al*, 1990). O grau da qualidade da cor também depende da quantidade de sensores e emissores do dispositivo gráfico e ainda, como a função de distribuição espectral foi reconstruída (VELHO; GOMES, 2001).

Uma superfície é considerada transparente quando possui a capacidade, de um modo geral, para refletir e transmitir a luz, considerando não somente o meio onde a luz se propaga, mas também as propriedades de reflexão e refração do material da superfície, ou seja, qual será a direção da luz rebatida pela superfície e a magnitude da luz transmitida pelo material transparente, respectivamente (BAKER; HEARN, 1997). Na computação gráfica a transparência é tratada da seguinte maneira: adotado um sistema de cor, este é divido em canais conforme o sistema operacional. Por exemplo, no sistema RGB (*Red, Green, Blue,* ou vermelho, verde, azul) em um sistema operacional de 32 bits serão separados quatro canais de 8bits onde cada um representa uma cor. Para cada canal a imagem fica representada somente com as tonalidades da cor referente ao canal. Para o canal alfa a imagem fica representada em preto, branco e tons de cinza. O preto indica transparência total e o branco indica opacidade total e os tons de cinza são a escala de transparência. A imagem final é o resultado da sobreposição de todos os canais. O canal alfa funciona como uma máscara para mesclar *pixels*, realçando as regiões da imagem que estão sobrepostas por partes transparentes (AZEVEDO; CONCI, 2003).

2.4.3 Textura

Na computação gráfica, a descrição realista de um objeto gráfico está no grau de acurácia da modelagem geométrica, na correta atribuição de cores e nas características da textura relacionada à superfície do modelo. Conforme Adaime (2005) a textura pode ser entendida como a variação de cor na superfície do objeto em função da rugosidade, transparência e posição do observador. A simulação da textura tem como objetivo evocar uma sensação similar à superfície real do objeto físico modelado geometricamente, melhorando a percepção do espaço, a forma e aparência dos objetos.

Em termos computacionais a textura pode ser obtida pelo mapeamento de uma imagem ou de forma procedural. O mapeamento de imagem consiste em relacionar o espaço paramétrico da imagem ao espaço preenchido pelo objeto gráfico por funções lineares ou não lineares. Esse mapeamento é gerado por algoritmos que podem, ainda, simular a rugosidade na textura mapeada pela perturbação dos vetores normais da superfície e simular a reflexão por algoritmos que alteram os parâmetros dos vetores de reflexão e luminosidade associados à textura. Os mapas procedurais consistem em um conjunto de pixels devidamente coloridos para simular a textura aplicada em um objeto gráfico, evitando cálculos para transformar uma imagem no seu espaço paramétrico para as faces daquele objeto gráfico. Uma forma para criar mapas procedurais consiste em utilizar funções harmônicas como as curvas senóides para representar padrões, por exemplo, de gramas e granitos (BAKER; HEARN, 1997; VELHO; GOMES, 2001).

2.4.4 Cena virtual e câmera virtual

Cena virtual ou cena é a descrição gráfica e computacional de um mundo virtual para representar aspectos relevantes para a visualização de objetos pertinentes. Fazem parte da cena, além dos objetos virtuais, as fontes de luz, a

câmera virtual, as relações e os vínculos entre os objetos e configuração. A cena deve ser descrita em um padrão que ofereça especificidade, generalidade e flexibilidade (VELHO; GOMES, 2001).

Câmera virtual ou câmera sintética é uma forma computacional de visualização de objetos ou uma cena. Deve ser implementada para permitir operações relacionadas à visualização como projeção ortogonal e perspectiva, calcular faces visíveis e movimentar conforme os comandos do usuário. Os parâmetros necessários para orientação de uma câmera virtual são: sua posição no mundo virtual, o ponto focal que orienta a direção perpendicular de projeção entre a câmera e o plano de imagem que a cena será projetada, o vetor que indica o lado de cima da cena 3D (*view-up*), os planos que delimitam a profundidade da cena (*clipping planes* – *near plane* e *far plane*), tipo de projeção: ortogonal (paralela) ou perspectiva (HILL, 2000). A figura 2.8 apresenta alguns parâmetros de uma câmera virtual.

2.4.5 Rendering

Rendering é o processo de criação de uma imagem em um dispositivo de visualização conforme uma descrição computacional (OLIVEIRA; MINGHIM, 1997). Também pode ser compreendido como o processo de sintetizar um objeto ou cena gerada por computador até obter uma aparência real conforme as seguintes etapas: criação da aparência tridimensional dos objetos virtuais por suas projeções ortogonais e perspectivas; eliminação das faces escondidas devido à posição entre objeto e observador (*culling back-faces*); corte de objetos da cena cuja visualização não está no campo de visão (*clipping*); conversão da representação tridimensional em pixels (rasterização); oclusão das faces por causa da obstrução ocasionada pela posição relativa entre os objetos; colorir cada pixel em função da simulação da iluminação, brilho, transparência e reflexão (BAKER; HEARN, 2003).

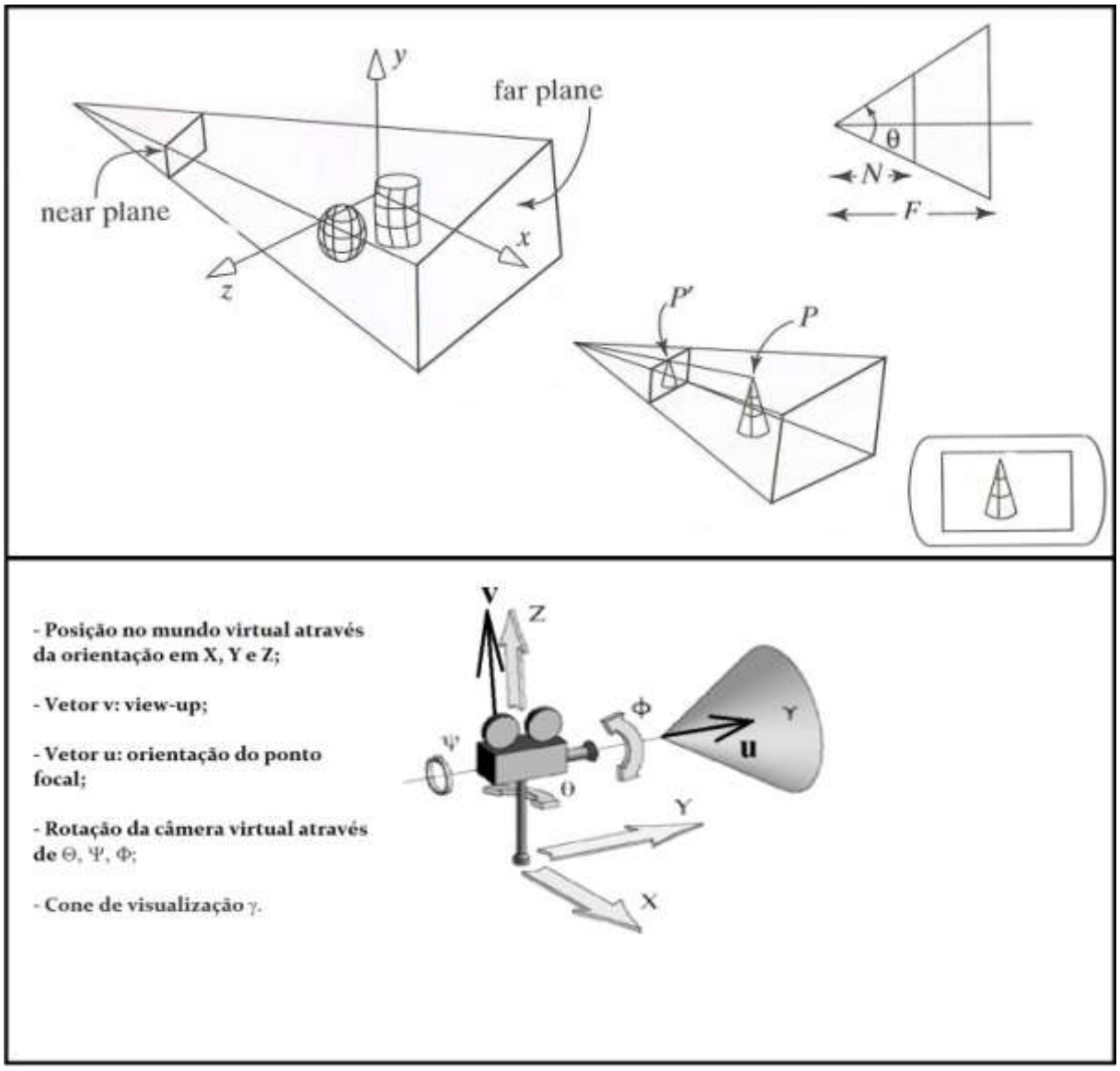

FIGURA 2.8: PARÂMETROS DE UMA CÂMERA VIRTUAL.
FONTE: Adaptado de Hill (2000).

O recorte de superfícies ou faces determina o conjunto de pontos do objeto gráfico dentro do campo de visão. Evita que certas regiões do objeto gráfico sejam projetadas na imagem aumentando a eficiência no processamento da imagem. Dentre os algoritmos para calcular as faces visíveis vale citar o algoritmo do pintor (*z-sort*), eliminação das faces pelo cálculo da normal e o *z-buffer*. Dentro de cada particularidade de implementação, esses algoritmos armazenam as coordenadas do objeto gráfico na cena e executam cálculos para descobrir a magnitude do ângulo entre o vetor normal de uma face do objeto gráfico e a linha de visualização da câmera virtual. A superfície será visível se a magnitude estiver num intervalo entre -90º e +90º (AGOSTON, 2005).

Rasterização é o processo de discretização da cena em coordenadas de pixels. Quando a cena é projetada no plano de imagem, visualizada pela câmera virtual, são atribuídos valores para cor e iluminação para cada pixel em função dos cálculos realizados de superfícies visíveis e iluminação. Esses cálculos podem ser iterativos (mais rápidos) ou recursivos (mais adaptativos). Nessa etapa ocorre também o processo de *anti-aliasing* cuja função é suavizar contornos "serrilhados" quando apresentados pelo dispositivo visual formado por pixels. Como exemplo pode-se citar o algoritmo de Bresenham e o algoritmo *Scanline* (AGOSTON, 2005).

No mundo real a iluminação está relacionada com a interação entre luz e matéria. A luz se comporta como uma onda eletromagnética e também como um deslocamento de partículas chamadas fótons. Normalmente, os modelos de iluminação na computação gráfica modelam a luz usando seu comportamento corpuscular para calcular as trajetórias de incidência dos feixes de luz entre os objetos da cena (VELHO; GOMES, 2001). As fontes de luz numa cena podem ser representadas por três formas: direcional, pontual e focalizada. Fonte de luz direcional simula feixes paralelos em uma única direção, por exemplo, o sol. A fonte de luz pontual emite os feixes de uma única origem para todas as direções, como uma chama de uma vela. Fonte de luz focalizada concentra em uma pequena região a origem dos feixes e possuem um intervalo de direção da emissão como a luz de uma lanterna (BAKER; HEARN, 1997).

A simulação da incidência da luz segue a lei de Snell que é função do material da superfície: material translúcido que permite a passagem dos feixes de luz, material metálico que reflete de forma concentrada os feixes de luz, e material composto que reflete a luz e/ou permite sua passagem. Modelos propostos por Gouraud, Phong, Blinn e Torrance simulam a iluminação local, ou seja, atuam na configuração da reflexão em uma dada superfície (BAKER; HEARN, 1997; VELHO; GOMES, 2001; COHEN; MANSSOUR, 2006). Técnicas como o traçado de raios (*Ray Tracing*) e a radiosidade simulam a iluminação global da cena, ou seja, a trajetória dos feixes de luz em relação à interatividade dos objetos e fontes de luz. A radiosidade funciona melhor para as iluminações difusas e o traçado de raios funcionam melhor para iluminações especulares. Ainda, existe a técnica de *rendering* de volumes para objetos gráficos que representam fumaças e nuvens e adotam algoritmos de reconstrução de volumes como o *marching cubes* até

algoritmos para simulação de luz com técnicas de *Ray Tracing* também chamado de *Ray Casting* (AGOSTON, 2005).

2.4.6 Animação

Os objetos gráficos podem ter suas propriedades alteradas conforme uma cadência de tempo, assegurando uma sensação de continuidade como o movimento de um ator em uma cena de cinema. Normalmente os objetos gráficos são animados com o uso das seguintes técnicas: *keyframe*, *script* ou procedural. No *keyframe* os objetos são posicionados em quadros ditos críticos e o efeito de animação acontece quando se interpolam esses quadros. Ao usar o *script* o sistema reconhece essa linguagem interpretada que possui uma sequência de instruções relativas aos objetos que serão animados. Na procedural ocorre a implementação do movimento por modelos matemáticos que serão compilados (BAKER; HEARN, 1997; AZEVEDO; CONCI, 2003).

2.5 CONSIDERAÇÕES FINAIS

O capítulo 2 apresentou os conceitos fundamentados sobre Computação Gráfica e Visualização Científica, além de situar as duas áreas de conhecimento em um contexto cronológico. Ainda, foi apresentado o processo de visualização e seus respectivos modelos e componentes.

Dessa forma, conclui-se o capítulo e assim, a terminologia pertinente à Computação Gráfica e Visualização Científica foi adotada para nos próximos capítulos.

3 AMBIENTES VIRTUAIS

Conforme surgem as evoluções na ciência, as pesquisas resultam em dados mais complexos e, por consequência, existe a dificuldade em analisar tais dados. Para ressaltar tais dados as técnicas de visualização científica exploram o sentido mais aguçado do ser humano: a visão. Porém, ao usar a Realidade Virtual (RV) pode ser proporcionada uma experiência maior, pois possibilita explorar os demais sentidos humanos por meio de interfaces inovadoras (ERICKSON[6], 1993; BUXTON[7], 1992 *apud* McLELLAN, 1996).

Os Ambientes Virtuais (AV), ou Ambientes de Realidade Virtual, como recurso didático - dentro do contexto da informática aplicada ao ensino e treinamento - podem despertar no aprendiz o seu interesse e pensamento crítico para construir seu conhecimento, pois não exige do aprendiz que abstraia os conceitos dos dados simulados, mas em contrapartida, as informações pertinentes à aprendizagem são apresentadas buscando transcrever como aqueles indivíduos iriam interagir para aprender no mundo real. Pesquisas como do matemático sul-africano Seymour Papert, criador da linguagem LOGO para ensino de informática aplicado às crianças, e do filósofo francês Pierre Lévy apontam o uso do computador como forma positiva para o processo de aprendizagem (PASQUALOTTI, 2000).

Este capítulo apresenta as definições, as classificações, os dispositivos típicos e uma retrospectiva histórica sobre RV assim como a relação entre os AV de educação e treinamento e teoria construtivista e sua aplicação na Arquitetura, Engenharia e Construção.

---

[6] Erickson, T. (1993). Artificial realities as data visualization environments.In Alan Wexelblat (Ed.), Virtual reality: Applications and explorations (pp. 1–22). Boston: Academic Press Professional.

[7] Buxton, B. (1992). Snow's two cultures revisited: Perspectives on human-computer interface design. In Linda Jacobson (Ed.). CyberArts: Exploring art and technology (pp. 24–38).

3.1 CONCEITOS

O termo Realidade Virtual (RV) começou ser usado pelo cientista norte-americano Jaron Lanier para distinguir simulações computacionais tradicionais de simulações envolvendo múltiplos usuários em um ambiente compartilhado (NETTO *et al*, 2002). Lee Adams (1999) *apud* Pompeu(1999) define RV como sendo uma simulação do espaço-tempo controlada por interfaces de Realidade Virtual, sendo formada pelos seguintes elementos: universo ou cena, que inclui o ambiente tridimensional virtual e as entidades a ele pertencentes; iluminação; sensores de resposta aos comandos enviados à cena; gerenciador de simulação para controle das regras do universo; ações e respostas das entidades. Não se deve confundir RV com CAD (*Computer Aided Design*), animação e multimídia. Comparada a esses sistemas, a RV oferece uma sensação maior de interação homem-máquina, além de formas mais intuitivas para manipular o comportamento de entidades de uma cena (NETTO *et al*, 2002).

Nota-se que as palavras realidade e virtual apresentam sentidos opostos, mas quando unidas formam um conceito moderno sobre atuar em um mundo sem estar efetivamente nele (MIRANDA[8], 1999 *apud* RIBEIRO JUNIOR, 2004). No mesmo trabalho encontra-se, ainda, outra definição para RV:

> *Em termos computacionais, a realidade virtual é considerada a forma mais avançada de interface com o computador. Permite ao usuário obter a imersão, navegação e interação em um ambiente sintético tridimensional gerado por computador, utilizando canais multissensoriais (VINCE[9], 1998* apud *RIBEIRO JUNIOR, 2004, p. 1).*

Howard Rheingold define RV como a experiência sentida pelo usuário quando está cercado por uma representação tridimensional em tempo real computacional (RHEINGOLD[10], 1991 *apud* DeFANTI *et al*, 1993).

O conceito RV fica bem claro quando se unem as seguintes explicações:

---

[8] MIRANDA, José Carlos. Urbanismo e Espaços Virtuais – Divulgação e Discussão na Comunidade. Tese de Mestrado, Universidade do Porto, 1999

[9] VINCE, J. Essential virtual reality fast: how to understand the techniques and potential or virtual reality. Berlin: Springer, 1998. 174 p.

[10] RHEINGOLD, H. Virtual reality. New York: Touchstone, 1991.

> *Mundo virtual é o nome dado ao mundo digital criado a partir de técnicas de computação gráfica. Uma vez que se possa interagir e explorar esse mundo através de dispositivos de entrada e dispositivos de saída, ele se transforma em um ambiente virtual ou ambiente de Realidade Virtual (VINCE[11], 1995* apud *MACHADO, 1997, p. 7).*
>
> *Dois fatores tornam-se então bastante importantes em sistemas de RV são eles: imersão e interatividade. A imersão pelo seu poder de prender a atenção do usuário, e a interatividade no que diz respeito à comunicação usuário-sistema (PIMENTEL & TEIXEIRA[12], 1995* apud *MACHADO,1997, p. 7).*

Na literatura é comum encontrar termos relacionados à RV tais como: Ambiente Virtual, Mundo Virtual, Realidade Artificial, Ciberespaço. O Ambiente Virtual (AV) diz respeito a um espaço tridimensional gerado por computação gráfica com o qual um ou mais usuários podem interagir. Conforme Pompeu (1999) o termo Ambiente Virtual tem maior uso na comunidade científica, pois um sistema de RV não precisa, necessariamente, retratar uma realidade. Mundo Virtual remete maior amplitude dimensional para o ambiente virtual. Realidade Artificial foi o termo usado por Myron Krueger (1985, 1991) para definir interação entre humano e computador por meio de imagens e objetos que não existiam no ambiente físico. Ciberespaço é um termo que foi usado no romance de ficção científica *Neuromancer* do escritor norte-americano William Gibson para descrever um mundo de comunicação aberto mediante tecnologias de interconexão mundial de computadores que ditam estruturas e princípios sociais da civilização (NETTO *et al,* 2002).

Para este trabalho a Realidade Virtual (RV) será conceituada como um conjunto das definições e termos já mencionados. A finalidade está em compreender a RV como um sistema, que compreende uma perspectiva científica e uma perspectiva tecnológica - cujos requisitos necessários são: interface homem máquina de alta qualidade; resposta adequada e rápida aos comandos do usuário; prover o usuário de um grau adequado de envolvimento; ser análogo ao mundo real/físico ou poder se misturar com ele. Vale ressaltar que os requisitos não precisam ser necessariamente explorados ao máximo, porém a falta de um deles pode comprometer o sistema de realidade virtual (NETTO *et al*, 2002). Já os

---

[11] VINCE, J. Virtual reality systems. Cambridge: Addison-Wesley, 1995.

[12] PIMENTEL, K.; TEIXEIRA, K. Virtual reality - through the new looking glass. 2.ed. New York: McGraw-Hill, 1995.

Ambientes Virtuais (AV), neste trabalho, remetem aos ambientes desenvolvidos mediante conceitos de RV, cuja simulação procura reproduzir um cenário real para um propósito ou finalidade, por exemplo, treinamento e aprendizagem.

3.1.1 Classificação

Na bibliografia são apresentadas diversas formas de classificar a realidade virtual. Mesmo havendo particularidades nessas classificações, elas acabam tendo o mesmo foco: a questão da imersão. Pode-se, portanto, classifica-las em Realidade Virtual Imersiva e Realidade Virtual Não Imersiva (RIBEIRO JUNIOR, 2004). Os conceitos de imersão e não imersão estão relacionados ao uso de dispositivos que permitem ao usuário desconectar-se do mundo real. Neste contexto, RV imersiva usa dispositivos não convencionais para imersão e interatividade, sofisticados aplicativos de softwares de modelagem geométrica e hardwares de alto desempenho. Já a RV não imersiva tende a não necessitar de hardwares de alto desempenho ou dispositivos sofisticados. A RV não imersiva não possui um alto grau de precisão, mas o suficiente aceitável para não acarretar custos computacionais (FRANCIS; TAN, 1999).

Uma classificação mais abrangente foi abordada no artigo de Hilary McLellan (1996), classificando RV em dez tipos: imersiva em primeira pessoa, aumentada, desktop ou *through window*, projetada, *waldo world*, câmara, *cab simulator*, ciberespaço, *visiodome* e *experience learning system*.

Realidade Virtual Imersiva em Primeira Pessoa, também conhecida como *Visually Coupled Display* (PIMENTEL; TEIXEIRA[13],1995 *apud* MACHADO, 1997) proporciona ao usuário a sensação de estar inserido na cena de RV. Procura-se nela estimular a percepção visual, auditiva, tato e olfato, e requer o uso de dispositivos de interface mais sofisticados para imersão e interatividade.

Realidade Aumentada, ou Realidade Realçada, ou Realidade Misturada é o aumento da percepção causada ao usuário por meio de dispositivos de visualização

[13] PIMENTEL, K.; TEIXEIRA, K. Virtual reality - through the new looking glass. 2.ed. New York: McGraw-Hill, 1995.

para misturar objetos virtuais ao mundo real. O potencial para o uso de Realidade Aumentada está no treinamento de atletas, exploração de dados em objetos e locais reais, guia para operações em manufatura industrial e cirurgias. A figura 3.1 apresenta uma escala criada por Paul Milgram (FISHER, 2001) para conferir o grau de realidade aumentada.

FIGURA 3.1-ESCALA DA REALIDADE AUMENTADA
FONTE: Adaptado de Fisher, 2001.

A modalidade *Through Window*, ou *Desktop*, ou *World on Window* pode ser entendida como a RV cuja visualização, navegação e interatividade podem ser realizadas por equipamentos menos sofisticados, convencionais e de baixo custo como monitores, mouse, teclado entre outros. Um exemplo do uso de T*hrough Window* está no campo da ciência forense: reconstrução de cenas de crimes para estudar sequência de eventos (BAIRD[14], 1992; HAMILTON[15], 1993 *apud* McLELLAN, 1996).

Realidade projetada, ou *mirror world*, ou realidade artificial proporciona a criação de um personagem ou ator virtual (também conhecido por avatar) que representa o usuário virtualmente, de forma que a interatividade é realizada em terceira pessoa. O *videoplace* (KRUEGER *et al*, 1985) é um exemplo de realidade projetada.

Waldo World é o termo usado para captura de movimentos e/ou expressões faciais de um usuário e reproduzidos (em tempo real) por uma marionete virtual.

[14] BAIRD, J. B. New from the computer: 'Cartoons' for the courtroom. New York: New York Times, 1992.

[15] HAMILTON, J. Virtual reality: How a computer generated world could change the world. Businessweek, 1992, f. 96–105.

Esse tipo de RV foi aplicado no sistema *Virtual Actor*™ da *SimGraphic Engineering* (TICE; JACOBSON[16], 1992 *apud* McLELLAN, 1996).

A câmara é um tipo de RV cujo envolvimento é alcançado por meio de um ambiente real (uma câmara) que permite movimentação com maior liberdade a um ou mais usuários. São usados diversos computadores para geração e monitoramento da RV que é projetada nas paredes da câmara (essas paredes são formadas, geralmente por telas de projeção traseira para evitar que a iluminação de uma tela e a sombra do participante atrapalhe as demais telas). Como exemplo, cita-se o sistema CAVE desenvolvido pela equipe de Thomas DeFanti (DeFANTI *et al*, 1993).

Cab simulator, ou realidade virtual de simulação são cabines físicas onde os usuários ficam envolvidos na simulação. É uma extensão da realidade virtual de câmara. Simuladores de avião são exemplos dessa classificação.

O termo ciberespaço foi criado por William Gibson em seu livro *Neuromancer* (GIBSON[17], 1985 *apud* McLELLAN, 1996) e define um mundo dominado por redes de computadores e dados para simular uma realidade artificial. Esse tipo de RV está ligado aos games de *Role-playing game* (*RPG*).

O v*isiodome* é um ambiente físico de RV imersiva projetado pela *Elumes Corporation,* e consiste em uma estrutura hemisférica que proporciona imersão de 360º num plano paralelo ao piso e imersão de 180º num plano perpendicular ao piso. Vários usuários podem experimentar a imersão sem usar dispositivos como HMD (*Head Mounted Display*) *das* câmaras. A Universidade da Carolina do Norte, nos EUA, foi a primeira instituição a obter esse sistema para pesquisas. Pode ser aplicada, por exemplo, no estudo de planejamento urbano, exploração geográfica e treinamento.

O *Institute for Creative Technlogies* (ICT) da Universidade de Carolina do Sul, nos EUA, pesquisa uma RV altamente realística pelo uso das mais modernas tecnologias e inteligência artificial para treinamento do exército americano. Chamado e*xperience learning system*, tal sistema procura simular emoções tais como pânico, tensão, medo e ansiedade, com base nas ações e decisões tomadas pelo usuário.

---

[16] TICE, S.; JACOBSON, L. VR in visualization, animation, and entertainment. In: Jacobson, L. (Ed.), CyberArts: Exploring art andtechnology. San Francisco, CA: Miller Freeman, 1992.
[17] GIBSON, W. Neuromancer. New York: Bantam Books, 1886.

Completando esta classificação existem os Ambientes Virtuais Colaborativos (AVC), tele presença e tele operação. Os AVC proporcionam interação em tempo real de multiusuários com acesso simultâneo a RV, sendo que os usuários não precisam estar no mesmo espaço físico, mas conectados em uma rede ligada à RV. Pesquisas sobre o uso da banda larga de internet e computação de alto desempenho são alguns dos campos do conhecimento que fazem parte deste tipo de RV. Como exemplo, cita-se o projeto I-WAY da *Electronic Visualization Laboratory* (DeFANTI *et al*, 1996). Para Scott Fischer (FISHER, 1995) tele existência é um termo para definir a experiência do usuário em sentir como se estivesse em outro local. Em outras palavras, tele existência, também denominado de tele presença, é o estímulo que o sistema de RV provoca no usuário de estar num local sem sua presença física. Tele operação, por sua vez, é o meio pelo qual o usuário controla autômatos a distância (TACHI, 1990).

3.1.2 Imersão, interatividade e envolvimento

Imersão é a sensação de estar “dentro” de uma cena simulada por um AV. O grau de imersão do usuário está relacionado com a sua percepção em relação ao espaço físico que o cerca. Ou seja, quanto maior for a imersão proporcionada por um AV ao usuário, menor será sua percepção do mundo real, ou ainda, o indivíduo não distinguir o mundo virtual do real quando ambos estão misturados.

O conceito de interatividade diz respeito à capacidade de um computador detectar os comandos e as ações de um ou mais usuários por meio de dispositivos e responder tais ações pela modificação e/ou atualização da cena virtual (BOWMAN, 1999).

O envolvimento está ligado com o grau da motivação do usuário ao interagir, ativa ou passivamente, com um sistema de RV (NETTO *et al*, 2002).

3.1.3 Tempo real e latência

A resposta entre movimento do usuário capturado e evento que ocorre em um AV pode ser classificada por duas formas: tempo real e latência. O tempo real é o sentimento de instantaneidade percebido pelo usuário do sistema em resposta a um evento. Latência é o atraso ou a não sincronia entre ação e resposta num AV.

3.1.4 Percepção visual e estereoscopia

Aproximadamente 70% dos receptores de estímulo ligados aos cinco sentidos estão concentrados no olho humano (JACOBSON, 1994). É natural esperar que pesquisas e projetos de tecnologias em RV foquem seus esforços nas interfaces de visualização.

O ser humano, por meio da visão, tem a capacidade de distinguir forma, cor, textura, posição alvos dentro do seu campo de visão. Essa capacidade é chamada de percepção visual. São três as categorias de estímulos visuais: informações monoculares, informações óculo-motoras e informações estereoscópicas (AZEVEDO; CONCI, 2003).

Informações monoculares são informações contidas na retina e estão relacionadas à perspectiva linear, oclusão, textura, reflexão da luz, sombra e conhecimento prévio do alvo. A perspectiva linear é a alteração aparente da dimensão do alvo em relação ao observador. A oclusão é a obstrução parcial ou total de um alvo ocasionado por outro alvo. A textura está relacionada aos padrões específicos da aparência de um objeto. A reflexão da luz , assim como a sombra, informa a curvatura de superfícies de um alvo. O conhecimento prévio do alvo auxilia na determinação da distância absoluta entre o observador e um alvo e a distância relativa referente aos alvos observados (AZEVEDO; CONCI, 2003).

Informações óculo-motoras estão relacionadas com a movimentação dos músculos dos olhos. O movimento de contração da lente ocular (cristalino) é denominado de acomodação e diz respeito ao foco dos objetos. O movimento de

rotação dos olhos é chamado de convergência e informa tanto a posição como a distância dos alvos (AZEVEDO; CONCI, 2003).

Informações estereoscópicas estão relacionadas pelo fato do olho esquerdo captar uma imagem diferente do olho direito enquanto o ser humano visualiza um alvo. Ao formarem um par de imagens de um alvo, denominado de par estéreo, ocorre o realce das distâncias relativas do próprio alvo para montar sua aparência tridimensional. Devido a essa disparidade, o cérebro usa tal informação para montar a distância relativa a um alvo focalizado pela visão. A profundidade da imagem relativa a um alvo também esta relacionada com a magnitude de paralaxe, ou seja, a distância entre a imagem esquerda e a imagem direita (FOLEY *et al*, 1996).

Dispositivos computacionais podem simular a percepção visual para aumentar a imersão mediante a técnica de estereoscopia. A imagem pode ser representada em por dois dispositivos e cada um projetando uma imagem respectiva para cada olho. Alternativamente, pode se representada por um dispositivo que projete alternadamente as imagens respectivas a cada olho. Inicialmente, os sistemas de estereoscopia proporcionavam a visão somente a um observador; porém, atualmente é possível que diversos usuários possam experimentar a cena estereoscópica (FOLEY *et al*, 1996). Estima-se que 10% a 20% dos humanos não possuem capacidade para visualizar imagens estereoscópicas (JACOBSON, 1994).

Em Machado (1997) foi apresentada uma classificação para estereoscopia: voluntária, anaglifo, polarização da luz, luz intermitente, holografia. A estereoscopia voluntária faz o uso de um instrumento chamado estereoscópio que serve para examinar pares de fotografias com paralaxe através de lentes, espelhos e prismas. Na estereoscopia em anaglifo o relevo de uma imagem bidimensional pode ser realçado por meio de cores complementares que são captadas por filtros em lentes, um para cada olho. A polarização da luz utiliza filtros polarizadores para separar o par de imagens em planos ortogonais, um para cada olho. A luz intermitente trabalha com a projeção alternada de imagens numa frequência entre 1/20 s a 1/60s. A holografia é uma técnica de captura de informações do alvo tridimensional e o representa em um filme com as suas dimensões espaciais.

Mesmo que a percepção visual represente a maior porcentagem dentre os sentidos, as percepções táteis e auditivas também podem ser exploradas. Estudos propostos por Zimmerman *et al* (1987) e Blanchard *et al* (1990) faziam o uso de luvas com sensores flexíveis para simular o tato. Krueger e Gildren (1997)

apresentaram um dispositivo para guiar cegos denominado *KnowWhere*. Weimer e Ganapathy (1989) apresentavam um estudo sobre a importância em combinar gestos e comandos de voz para interatividade em RV, além do estímulo visual.

3.1.5 Navegação

A navegação é o resultado da combinação entre o movimento do ponto de vista (da câmera virtual que visualiza o cenário sintético) com o processo cognitivo para se traçar e/ou escolher um caminho (BOWMAN, 1999). É a forma mais usual de interatividade entre usuário e um AV. Portanto, a navegação diz respeito à forma pela qual o usuário explora um cenário sintético, seja ela passiva ou pré-programada, exploratória ou interativa, sem alteração na cena ou com alteração na cena.

Como exemplo, podem ser citada as seguintes navegações: *fly* (*fly-through*), *point-and-fly*, *eyeball-in-hand*, *scene-in-hand* (NETTO *et al*, 2002). A navegação *fly* permite ao usuário navegar com a sensação de estar voando em uma cena virtual. A *point-and-fly* é uma navegação onde o usuário explora a cena quando aponta para algum local de uma cena mediante um dispositivo de interação. *Eyeball-in-hand* é uma forma de navegar em uma cena por algum sistema de sensores eletromagnéticos de interação com seis graus de liberdade. Quando se usa a navegação *scene-in-hand*, o ponto de vista do usuário está fixado e a cena é movimentada (NETTO *et al*, 2002). Ainda, para completar as formas de navegação existe a *walk*, que simula uma pessoa caminhando em uma cena.

A figura 3.2 apresenta uma taxonomia de navegação conforme Bowman (1999). A direção/seleção do alvo se refere ao método de controle e movimentação contínua ou não do ponto de vista. A velocidade e aceleração estão relacionadas à taxa de movimento ao longo da cena. A condição de entrada diz respeito ao tipo de controle conforme as especificações do sistema.

Na próxima seção são apresentados os dispositivos relativos ao sistema de RV que abordam os conceitos referentes à imersão, interatividade e envolvimento, a

definição de tempo de latência e tempo real, os conceitos pertinentes à percepção visual e navegação que foram explorados nesta sessão.

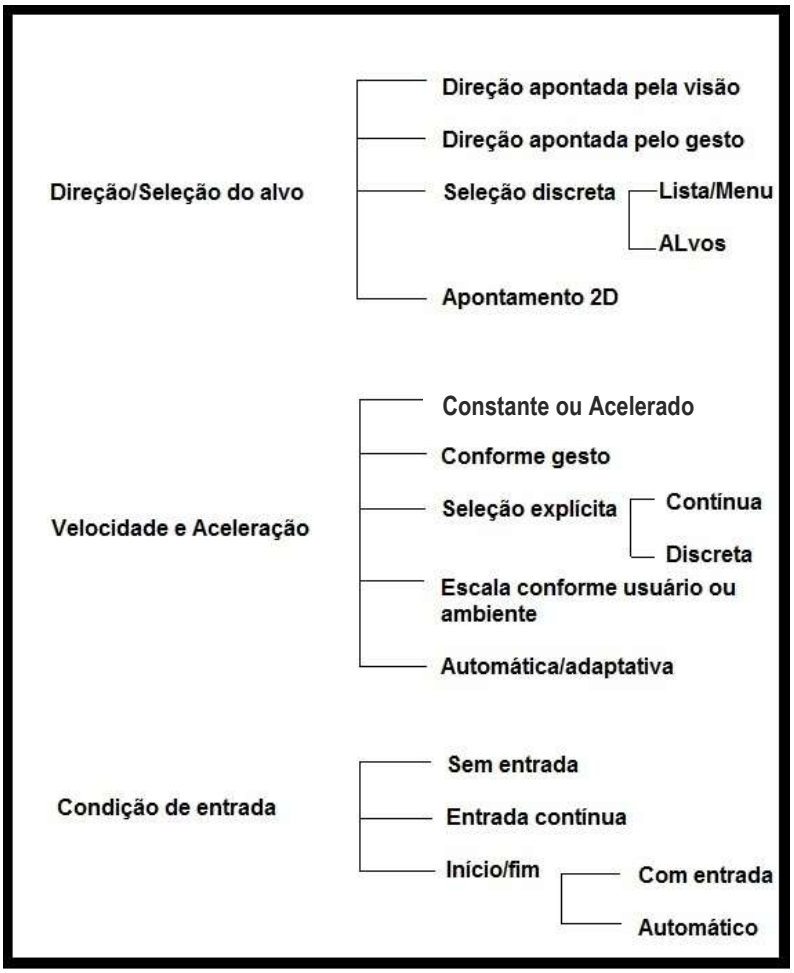

FIGURA 3.2-TAXONOMIA DA NAVEGAÇÃO EM AV
FONTE: Bowman, 1999.

3.2 DISPOSITIVOS APLICADOS AOS AMBIENTES VIRTUAIS

A comunicação entre o usuário e um AV, ou seja, a imersão, a interatividade e o envolvimento dependem de aplicativos de softwares para a visualização e criação de mundos virtuais e ainda, dispositivos físicos que permitem fluxo de dados tais como captura de gestos, forma de navegação, captura do movimento e visualização. Esses dispositivos podem ser classificados conforme a direção do fluxo de dados: entrada de dados, ou saída de dados, ou seja, dispositivos de captura de

comandos ou informações gerados pelo usuário e dispositivos que transmitem informações ao usuário, respectivamente.

Neste trabalho optou-se por uma classificação conforme a finalidade do dispositivo: visualização, caminhada e interatividade, mas realçando se o dispositivo é de entrada ou saída de dados.

3.2.1 Dispositivos de visualização da simulação em um ambiente virtual

CAVE (*Cave Automatic Virtual Environment*) é um sistema de imersão para AV de câmara onde são projetadas imagens estereoscópicas em suas paredes (DeFANTI *et al*, 1993). Normalmente requerem grandes instalações e tecnologias não convencionais. Funciona com dispositivos de entrada e saída de dados e consiste em um espaço físico real para rodar a cena de um AV cercado por paredes formadas, geralmente, por telas de projeção traseira para evitar que a iluminação de uma tela e a sombra do participante atrapalhe as demais telas e no piso. Os movimentos do usuário são captados por sensores e a estereoscopia é realizada por óculos estereoscópicos. O sistema incorpora fontes remotas de dados e supercomputadores de alto-desempenho. As telas recebem tubos de fósforo verdes especiais para melhorar a estereoscopia. Dispositivos de correção de distorções e sistema de memória reflexiva para corrigir erros de sincronia da imagem são alguns dos artifícios usados para criar a sensação de imersão. Além disso, o sistema de som é composto por caixas posicionadas ao redor do ambiente físico, proporcionando a imersão além da imagem. A figura 3.3 apresenta a 3ª geração da CAVE operando com resolução de 68 milhões de pixels e sistemas wireless para posicionamento (DEFANTI *et al*, 2009).

Monitores são dispositivos de saída de dados gráficos baseados na projeção da luz sobre uma superfície para visualizar uma imagem. Comumente são empregadas as seguintes tecnologias: CRT (*Catode Ray Tube*), LCD (*Liquid Crystal Display*), DLP (*Digital Light Processing*), LCOS (*Liquid Crystal on Silicone),* LED (*Light Emitting Diode*), OLED (*Organic Light-Emitting Diode*).

FIGURA 3.1-VISTA INTERNA (ESQUERDA) E EXTERNA (DIREITA) DA 3ª GERAÇÃO DA CAVE
FONTE: DeFanti *et al*, 2009.

A tecnologia de CRT usa um canhão de feixes de elétrons de um cátodo quando este é aquecido. Os elétrons são atraídos por anodos próximos à tela de fósforo. Então ocorre o processo de varredura que é a movimentação dos elétrons na tela variando a taxas de 60Hz a 75Hz. A cor é obtida pela sensibilização do fósforo que pode ser monocromático, ou colorido com base no sistema RGB de cores (FOLEY *et al*, 1996).

Os monitores de LCD usam luz fluorescente ao invés do canhão de feixes permitindo ao dispositivo uma dimensão física menor em relação ao CRT. As telas usam um conjunto de células de cristais (cada célula forma um *pixel* monocromático) entre dois filtros polarizados. Quando os raios luminosos passam pelo primeiro filtro e encontram as células de cristais cujo arranjo permite a passagem da luz para o segundo filtro, então a imagem é gerada. A configuração do arranjo das células, ou seja, se permite ou não a passagem de luz depende do campo elétrico a que está submetido. Existem os monitores denominados *see-through* compostos por LCD transparentes.

Os dispositivos com LED possuem uma configuração similar ao LCD. A diferença está na fonte luminosa; enquanto no dispositivo com LCD existe uma única fonte de luz, nos dispositivos com LED existem milhares de pequenas fontes de luz usando o LED como emissor e trabalhando de forma independente entre eles, geando imagens com qualidade melhor que a LCD.

Os dispositivos que usam DLP[18] são formados por milhares de espelhos reflexivos em escala microscópica para o controle dos pixels da imagem. São arranjados em forma de matriz sobre um chip semicondutor chamado DMD (*Digital Micromirror Device*), o qual controla o brilho dos *pixels* conforme a movimentação angular dos espelhos.

Os dispositivos LCOS[19] utilizam uma matriz de cristal líquido posicionada sobre uma camada de espelhos reflexivos para representar os pixels da imagem. Trabalham de forma similar aos dispositivos DLP e podem gerar imagens grandes sem perda de qualidade.

Telas com a tecnologia OLED[20] são compostas por moléculas de carbono que emitem luz ao receber uma carga elétrica pelos filamentos metálicos que conduzem os impulsos elétricos acoplados na tela. As telas possuem luz própria e são apropriadas para uso de notebooks e computadores de mão.

Atualmente novas tecnologias estão sendo empregadas nesses dispositivos, tais como as superfícies retrorrefletivas, cujo material projeta a luz refletida numa direção muito próxima da fonte luminosa (BOLAS; KRUM, 2010).

Existem duas maneiras de apresentar conteúdos em 3D em dispositivos como monitores por meio das configurações passiva e ativa. A estereoscopia passiva consiste na exibição de uma imagem com pontos de vista ligeiramente diferentes (paralaxe) em que os óculos fazem a função de separar a imagem por filtros de cores ou lentes polarizadas. Já a estereoscopia ativa faz exibição de uma imagem na tela alternando-a em altas velocidades, e os óculos alternam entre transparente e opaco na medida em que as imagens são alternadas na tela.

Head mounted display (HMD) ou videocapacete (figura 3.4) é um dispositivo de saída de dados que consiste em duas telas, normalmente de LCD com resolução entre 360x280 pixels chegando até 1280x1024 pixels funcionando a 24 fps ou *frames per seconds* (taxa de quadro) e com um campo de visão entre 80º a 140º (Jacobson, 1994). Também funciona como um dispositivo de entrada de dados quando possuem sensores que captam o movimento e posição do usuário. *Head Coupled Display* (figura 3.5) é um display de saída de dados montado sobre um

[18] http://www.infowester.com/projetores.php
[19] http://www.infowester.com/projetores.php
[20] http://www.tecmundo.com.br/oled

braço mecânico com um contrapeso e sensores ligados para demarcação da posição do usuário (JACOBSON, 1994).

FIGURA 3.2-DISPOSITIVO HMD
FONTE: http://www.5dt.com/products/ihmd03.html

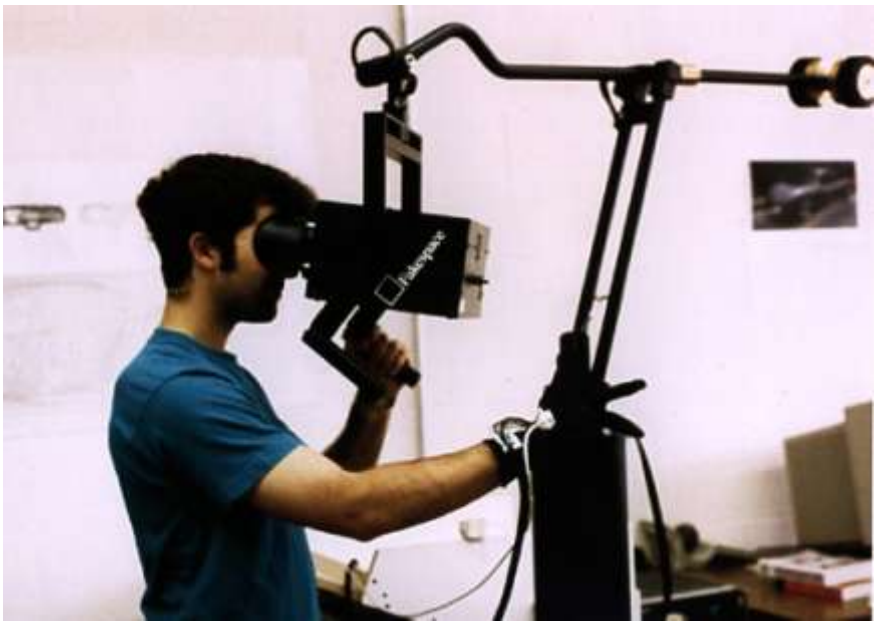

FIGURA 3.3-DISPOSITIVO HEAD COUPLED DISPLAY
FONTE: http://www-vrl.umich.edu/intro/

3.2.2 Dispositivos para simulação de caminhada

Para melhorar a imersão, a interatividade e o envolvimento em RV foram criados dispositivos de entrada para simular a caminhada. São eles: *string walker,* v*irtusphere e* esteira multidirecional ou o*mnidirectional treadmill*.

O *stringwalker*[21] é um dispositivo que simula a caminhada conforme a figura 3.6. Seu funcionamento consiste em cabos tracionados ligados a sapatos utilizados para fazer a leitura do movimento e repassar esses dados para que seja simulada uma caminhada na realidade virtual. Os cabos aplicam força no sapato em uma direção arbitrada. O mecanismo de roldanas automatizadas (figura 3.7.b) gera tensão no cabo, para assim medir a posição e a orientação. Um sensor de toque (figura 3.7.a) detecta o momento em que o pé se mantém apoiado ao chão para que a tensão seja aplicada ao pé que está em balanço. Quando o usuário muda a direção da caminhada, um mecanismo de engrenagens gira a plataforma para que siga a direção escolhida pelo usuário.

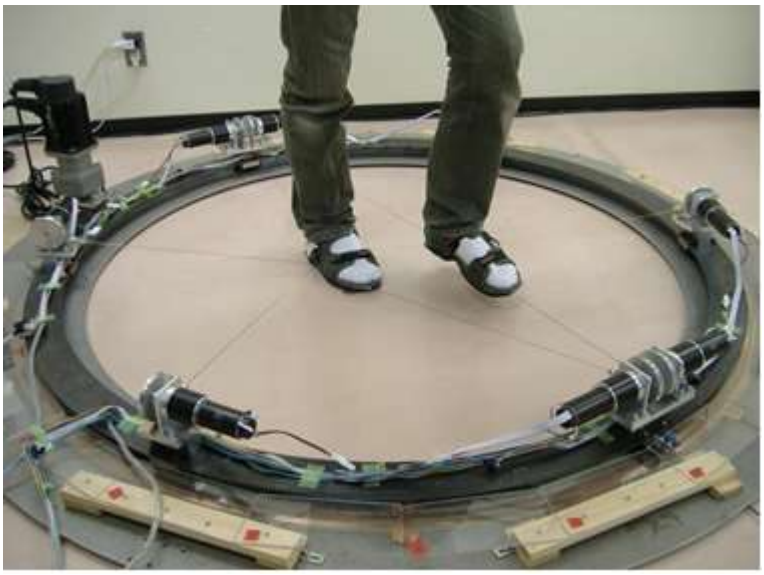

FIGURA 3.4-STRINGWALKER
FONTE: http://intron.kz.tsukuba.ac.jp/stringwalker/stringwalker.html

[21] http://intron.kz.tsukuba.ac.jp/stringwalker/stringwalker.html

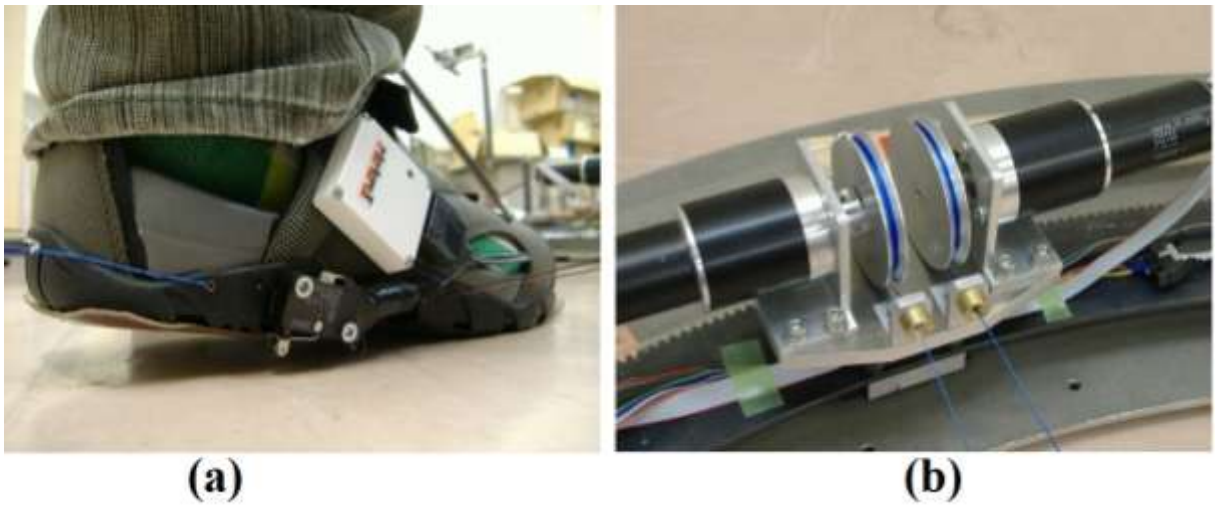

FIGURA 3.5- (a) SENSOR DE TOQUE E (b) ROLDANAS AUTOMATIZADAS.
FONTE: http://intron.kz.tsukuba.ac.jp/stringwalker/stringwalker.html

O *Virtusphere* [22] (figura 3.8) é um dispositivo de entrada composto por uma esfera que permite ao usuário simular a caminhada com maior liberdade. Sensores instalados na base fazem a leitura do movimento do usuário e passam os dados para um computador. Acoplado a um HMD, o usuário é capaz de experimentar uma grande imersão em uma realidade virtual.

FIGURA 3.6-VIRTUSPHERE
FONTE: http://www.virtusphere.com/

A esteira multidirecional (figura 3.9) ou *Omnidirectional treadmill* é um dispositivo de entrada para simulação de caminhada que pode ser acoplado a uma CAVE. O sistema é composto por duas esteiras. Uma esteira gira em um eixo e a outra gira ao redor da primeira esteira.

[22] http://www.virtusphere.com

FIGURA 3.7 – *OMNIDIRECTIONAL TREADMILL*
FONTE: http:// http://www.edailypost.com/omnidirectional-treadmill/

3.2.3 Dispositivos de interatividade com o ambiente virtual

Os dispositivos para interação com um AV são aqueles necessários para que o usuário interaja com a cena simulada. Como exemplos: telas sensíveis, mouses, luvas, dispositivos hápticos, rastreadores de posição, sensores que captam os movimentos e interfaces tangíveis.

As telas sensíveis ao toque servem como dispositivo de entrada assim como de saída. Dentre seus tipos, podem-se destacar três: resistivas, telas capacitivas e sistemas de microcâmaras. As telas resistivas funcionam pela pressão aplicada na tela. O sistema usa duas camadas de material condutor de eletricidade que ao se aproximarem entre si dividem a tensão da corrente aplicada aos materiais condutores e mapeiam o local onde ocorreu a pressão. As telas capacitivas utilizam uma camada carregada (capacitância) de eletricidade com uma pequena tensão, de tal forma que ao pressionar a tela com os dedos ocorre uma alteração na tensão e o sistema mapeia os pontos através desta perda de elétrons. Dispositivos como Ipad da Apple™ usam esse tipo de tela. O sistema de microcâmaras foi desenvolvido

pela Microsoft e é utilizado no dispositivo chamado *Surface*, e consiste em uma grande onde microcâmaras de infravermelho fazem a leitura da posição da mão do usuário. A figura 3.10.a demonstra um exemplo do Microsoft *Surface* e a figura 3.10.b demonstra um exemplo do Ipad.

Os dispositivos do tipo mouse 3D são dispositivos de entrada de dados para rastrear posições (*tracking)* que utilizam ponteiros e permitem a movimentação com 6 graus de liberdade. Alguns destes dispositivos podem ser usados totalmente apoiados numa superfície (figura 3.11). Existem outros dispositivos de entrada que oferecem 6 graus de liberdade como o *torque ball* e controles (joystick).

FIGURA 3.8-: EXEMPLO DO MICROSOFT SURFACE (A) E APPLE IPAD (B).
FONTE: http://www.vivaolinux.com.br e http://rockntech.com.br

A criação de roupas para realidade virtual (figura 3.12.a) se deu pela necessidade de um usuário conseguir interagir com o ambiente virtual de modo intuitivo. Tendo em vista que a utilização de roupas completas tem pouca ergonomia, foram adaptadas partes dessas roupas, tais como as luvas (*dataglove*, figura 3.12.b e 3.12.c), dispositivos de entrada de dados que têm a função de manipular de forma interativa no ambiente virtual. Atualmente esses dispositivos de entrada estão em evolução com vistas a promover sensação tátil (saída de dados) mediante interação de sistemas eletromecânicos (figura 3.12.e) ou exoesqueletos (figura 3.12.d) que simulam rugosidade, temperatura e atrito. Esses dispositivos são chamados de hápticos (NETTO *et al*, 2002).

FIGURA 3.9 - MOUSE 3D
FONTE: http://www.3dconnexion.com

Os rastreadores de posição são dispositivos de entrada para captar a posição do usuário, podendo ser magnéticos, óticos e inerciais. Os rastreadores utilizam sensores magnéticos e são rápidos, precisos e sensíveis a variações no campo eletromagnético que os rodeia, podendo causar erros na posição e na orientação. Os rastreadores óticos usam um conjunto de pequenas esferas reflexivas em um objeto para ser rastreado por câmeras infravermelho. Rastreadores inerciais são autônomos, ou seja, não há necessidade de um referencial externo para aquisição de dados e o cálculo de distâncias é obtido de forma discreta.

Quadro interativo[23] (figura 3.13) é uma superfície que pode reconhecer a escrita eletronicamente. Esses dispositivos são usados para capturar apontamentos escritos na superfície do quadro, utilizando canetas próprias para tal que utilizam tinta eletrônica, e para controlar (selecionar e arrastar) ou marcar notas ou apontamentos numa imagem gerada por computador e projetada no quadro por um projetor digital. Existem três tipos diferentes de quadros interativos: os eletromagnéticos, os sensíveis ao toque e os infravermelhos.

O Wii *remote*[24] (figura 3.14) é um dispositivo de entrada para detectar posição, fabricado pela Nintendo. Possui três acelerômetros (um para cada eixo) para captura de movimento, sensor ótico para determinar a direção que o dispositivo aponta e autofalante.

Microsoft Kinect[25] (figura 3.15) é um sensor de cores e movimento utilizado para interagir com o console Xbox360. Seu funcionamento consiste em dois

[23] http://smarttech.com/smartboard

[24] http://www.nintendo.com/wii

[25] http://www.xbox.com/pt-BR/Kinect

sensores infravermelhos para mapear o ambiente e usuários mesmo com pouca luminosidade, um sensor RGB para detectar cores e um software que consegue ler até 48 articulações de cada usuário.

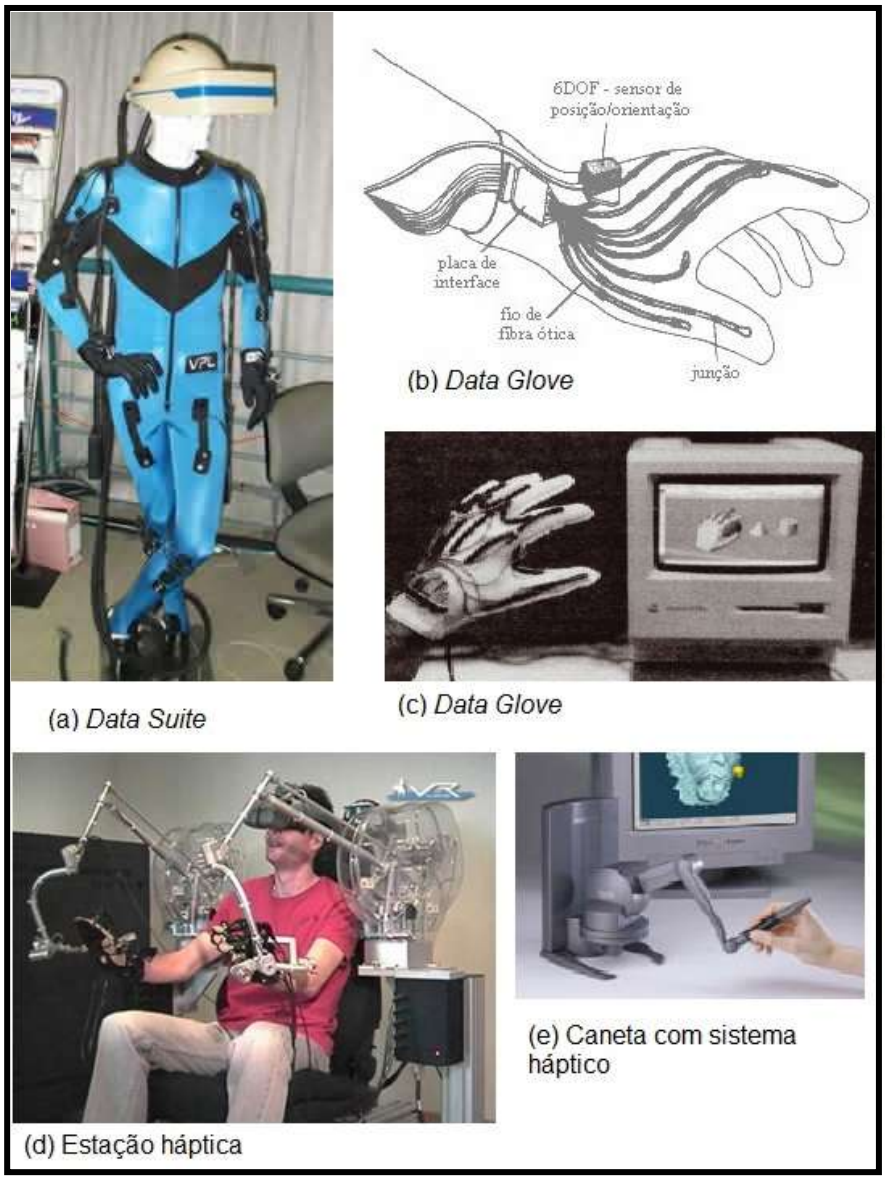

FIGURA 3.10 -DISPOSITIVOS DE INTERATIVIDADE

FIGURA 3.11- QUADRO INTERATIVO
FONTE: http://smarttech.com/smartboard

FIGURA 3.12 - WIIMOTE
FONTE: http://www.nintendo.com/wii

FIGURA 3.13 - KINECT
FONTE: http://www.techguru.com.br/wp-content/uploads/
2010/11/Kinect-Open-Source-Released-1.jpg

As Interfaces Tangíveis (TUIs - *Tangible User Interfaces*) buscam utilizar formas físicas com a finalidade de aproveitar as habilidades hápticas do usuário e assim comunicar-se com o sistema. Esses dispositivos tornam a informação digital diretamente manipulável e perceptível por meio dos nossos sentidos periféricos, oferecendo o acesso simultâneo a múltiplos dispositivos de entrada especializados. Assim, o sistema computacional pode explorar os atributos de forma, dimensão e orientação dos objetos que compõem a interface, diminuindo a complexidade da interatividade. A figura 3.16 ilustra um exemplo desses dispositivos.

Na seção subsequente está apresentada uma retrospectiva histórica que apresenta a evolução cronológica dos dispositivos empregados em RV.

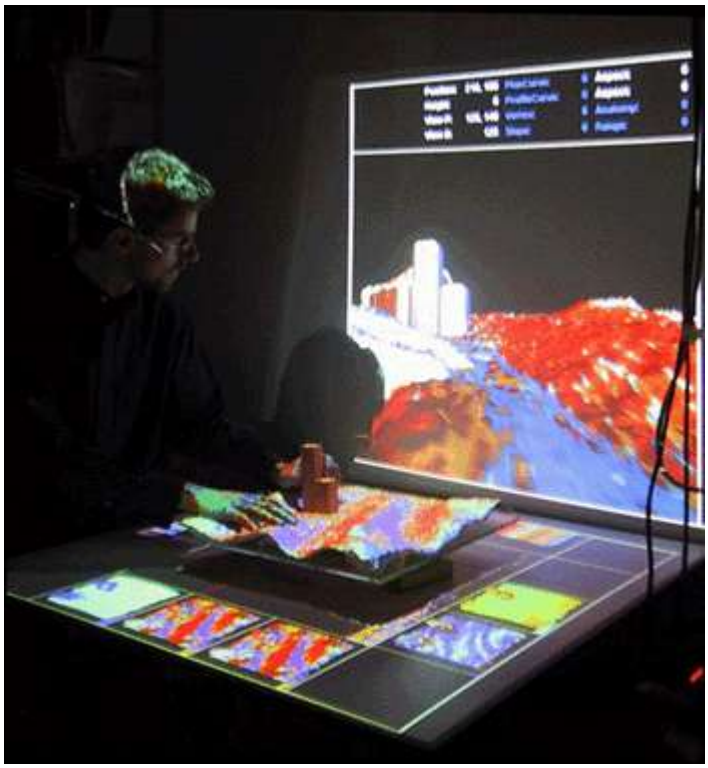

FIGURA 3.16: EXEMPLO DE UMA INTERFACE TANGÍVEL.
FONTE: http://telecomessentials.typepad.com/.a/6a00e3933364dc8834011278fb 351128a4-800wi

3.3 RETROSPECTIVA DA REALIDADE VIRTUAL

Esta seção explora a cronologia das pesquisas científicas relacionadas aos conceitos e dispositivos pertinentes ao sistema de RV, situando desde 1960 até a década de 2000.

3.3.1 Décadas de 60 e 70

Ivan Sutherland, um dos grandes pesquisadores na área da computação gráfica, em sua tese de doutorado defendida em 1963 no MIT (SUTHERLAND,1963) apresentou um conceito inovador para interatividade entre indivíduo e máquina: uma caneta que permitia o usuário podia desenhar na tela do computador. O dispositivo

ficou conhecido como *Sketchpad*. Ampliando o conceito de diálogo direto entre usuário e computador, Sutherland (SUTHERLAND, 1965) escreveu outro artigo mostrando a necessidade de interatividade por meio de joysticks, teclados e demais dispositivos disponíveis naquela época na busca por uma melhor interface homem-máquina. Ainda, em parceria com a Universidade de Harvard, o *Advanced Research Projects Agency* do Departamento de Defesa e Bell *Telephone Laboratories*, Sutherland (SUTHERLAND,1968) demonstrava um dispositivo de interatividade de exibição tridimensional que alterava a perspectiva do alvo conforme o movimento do usuário, denominado *Head-Mounted Display* (HMD). Concluiu, então, que a experiência da interatividade era mais importante que a sensação de estereoscopia.

Na universidade de Wisconsin, nos EUA, Myron Kruger (1978) pesquisou uma sala desenvolvida para estudar a captura de movimentos de alunos de diversas áreas da ciência usando sensores de pressão acoplados ao piso e dispositivos de áudio e vídeo. O experimento consistia em fazer um símbolo se movimentar em uma tela de 8'x10'.

3.3.2 Década de 80

O termo realidade virtual começou a ser popularizado na comunidade científica graças a Jaron Lanier, chefe pesquisador da VPL *Research Inc*, para diferenciar simulação computacional e simulação envolvendo múltiplos usuários em um ambiente compartilhado (RIBEIRO JUNIOR, 2004). Em 1987, Lanier apresentava um trabalho junto com Thomas Zimmerman (ZIMMERMAN *et al*, 1987) sobre a criação de uma luva que podia se comunicar com uma mão virtual e interagir com objetos virtuais, chamada de *DataGlove. S*imulava o tato através de sensores, captava os movimentos das articulações dos dedos além da posição e orientação.

Em 1982 o projeto *Visually Coupled Airborne Systems Simulator* de Thomas Furnes apresentava para a força área americana um equipamento chamado *SuperCockpit* . O sistema simulava o voo de avião numa cena tridimensional através de vídeo capacetes com áudio e um realismo e rendering em tempo real (RIBEIRO JUNIOR, 2004).

Já em 1984 a NASA iniciava o projeto *Virtual Visual Environment Display* para imagens estereoscópicas visualizadas por um dispositivo acoplado a uma máscara de mergulho. Depois de um ano, Scott Fisher se juntou ao projeto desenvolvendo dispositivos de *feedback* tátil e áudio. Passado mais um ano, a NASA possuía um ambiente virtual que permitia ao usuário uma interatividade por voz e manipulação de objetos virtuais (RIBEIRO JUNIOR, 2004).

Myron Krueger (KRUGER *et al*, 1985) publicou um estudo englobando educação, ciência da computação e ciência social resultando num sistema de interface usuário e máquina chamado *VIDEOPLACE*. O usuário podia ver sua silhueta em uma tela e interagia com uma criatura virtual por meio de movimentos capturados e processados por três *workstations* da *Silicon Graphics*. A arquitetura era dividida em dois componentes: sistema cognitivo rodado em um VAX11/80 para o monitoramento do ambiente virtual e um sistema de gerenciamento instantâneo de decisões por meio de um *Reflex System* com processadores dedicados. A pesquisa explorava aplicações de RV em atividades físicas voltadas a aprendizagem por computadores.

Motivado pelo desafio proposto por Ivan Sutherland em 1965 no congresso IFIP, Frederick Brooks (BROOKS, 1986) do Departamento de Ciência da Computação da Universidade da Carolina do Norte, nos EUA, pesquisou o conceito de *walkthrough* para auxiliar arquitetos na concepção e estudo de espaços em edificações. Usava joysticks, tablets para a navegação; sistema CAD para modelagem do edifício virtual; bibliotecas com algoritmos e rotinas para o gerenciamento do sistema.

Pesquisas envolvendo a RV e a teleoperação começaram a ser exploradas na década de 80. Como exemplo, pode ser citado o trabalho de Weimer e Ganapathy (1989) que estudava uma forma de sincronizar gestos com fala para melhorar a interatividade entre usuário e sistema, usando display estéreo, *dataglove*, dispositivos de reconhecimento de voz.

3.3.3 Década de 90

Com os avanços da RV nos anos 80 motivados principalmente para a arte e entretenimento, pesquisadores como Brooks (1990) começaram a questionar o caminho na qual a RV estava se desenvolvendo. Brooks apresentou então outra tendência ao uso da RV: a visualização científica e sua aplicação em áreas como química e medicina. Segundo ele, com a RV, os modelos científicos poderiam ser mais explorados e "sentidos" pelos cientistas.

Jaron Lanier ainda contribuiu com pesquisas voltadas a teleimersão: tecnologia de RV para multiusuários dispostos em diferentes locais, usando internet como ferramenta networking, captura de imagem dos usuários e do espaço físico que o usuário se encontra em tempo real, gerando avatares[26] foto-realísticos (LANIER, 1998). A tele-existência começava a ser explorada como interface remota para robôs, garantindo segurança ao usuário quando usava equipamentos e exoesqueletos (TACHI, 1990, 1998).

Scott Fisher publicou um artigo sobre desenvolvimento e pesquisa na área de *workstations* para RV (FISHER *et al*, 1991), no qual relatava uma experiência sobre a construção de um ambiente virtual no *Aerospace Human Factors Division* da NASA para experimentos em AV. Os tópicos da pesquisa foram telepresença para operações de controle de robôs; gerenciamento e monitoramento de informações portáveis e de grande escala; visualização de dados em ambientes tridimensionais como edifícios, *Computational Fluid Dynamics* (CFD)[27] e cirurgias médicas.

Uma grande novidade foi o trabalho desenvolvido por Thomas DeFanti e Daniel Sandin (DeFANTI *et al*, 1993) para um sistema de realidade virtual cuja imersão era proporcionada por conjuntos de display dispostos a formar paredes, limitando um espaço para formar uma câmara. Este sistema foi batizado CAVE. Pesquisando velocidade de banda em redes de computador, Thomas DeFanti apresentou o projeto I-WAY: múltiplos supercomputadores conectados em diversos locais do EUA compartilhando AV provendo aplicação em grande escala de testes

[26] Avatar uma figura gráfica de complexidade variada que empresta sua vida simulada para o transporte identificatório

[27] CFD é o ramo da mecânica dos fluidos que usa métodos numéricos para resolução de problemas de escoamentos.

IP/ATM[28], exploração de ferramentas comuns aos computadores, entre outros, motivando diversas pesquisas e aplicações em RV (DeFANTI *et al*, 1996) .

Uma aplicação de informações geográficas para cegos batizada de KnowWhere, que explorava a interatividade pela percepção auditiva em um sistema de *datatablet* (KRUEGER; GILDREN, 1997, 2002). Ainda na área de usuários com deficiência física, Pausch, Vogtle e Conway (1992) apresentou um trabalho sobre dispositivos de entrada para interatividade considerando o conforto do usuário quando seus movimentos são capturados.

Outro estudo motivado pela *National Academy of Sciences* (RANDY *et al*, 1997) onde foi realizado um ensaio com o intuito de prever quando deveria ser usado RV ao invés de displays convencionais. Ao testar tanto em ambientes com interface RV como em telas convencionais a procura de um dado objeto virtual, os usuários apresentavam melhor desempenho quando usavam RV imersiva.

3.3.4 Década de 2000

Em um artigo publicado nessa década, Scott Fisher (FISHER *et al*, 2002), junto com uma equipe de pesquisadores da Universidade de Keiko, no Japão, apresenta o conceito de micropresença que por meio da tecnologia de RV era possível interação de um micro mundo (termo usado pelo pesquisador para descrever um mundo físico que não pode ser visto a olho nu). Desse trabalho, cabe destacar que os modelos virtuais eram gerados por reconstrução da imagem através de informações de profundidade.

A equipe do *Electronic Visualization Laboratory* (EVL), da Universidade de Illinois, nos EUA, que possuía entre os pesquisadores Thomas DeFanti, desenvolveu uma arquitetura de computação para uso de RV em rede através de

[28] ATM Asynchronous Transfer Mode (ATM) é uma técnica de mudança de padrão concebido para unificar telecomunicações e redes de computadores. Ele usa multiplexação por divisão de tempo assíncrona.

computadores fotônicos[29] para criar um pipeline de computação distribuída na exploração interativa de dados e ferramentas de visualização (LEIGH *et al*, 2003).

Daniel Sandin e Thomas DeFanti junto com demais pesquisadores da EVL apresentaram em um artigo o sistema Varrier™ (SANDIN *et al*, 2005), que estuda autoestereoscopia (sem a necessidade de dispositivos calçados na face como óculos e HMD) composto por telas de barreira de paralaxe passiva.

Em 2009, os dois pesquisadores ora citados publicaram um artigo sobre a 3ª geração da *CAVE*, chamada de *StarCAVE* (DeFANTI *et al*, 2009)**.** Operando com a resolução de 68 milhões de pixels distribuídos em 15 telas de projeção traseira em uma câmara com cinco paredes e piso que projetam RV estérea entre 96 a 160 quadros por segundos, proporcionando uma imersão de 360º.

A tecnologia *smart touch* faz a combinação entre displays de visualização, realidade aumentada, e dispositivos táteis. O protótipo desenvolvido consiste em uma placa com três camadas sendo a primeira composta de eletrodos em uma placa fina, a segunda um filme de sensores de força e a terceira camada é composta por sensores óticos. Um caso estudado foi o usuário passar o dedo sobre uma imagem como fitas em relevo, de forma que o dispositivo provocava a sensação de relevo (KAJIMOTO *et al*, 2004).

Neste capítulo, até a presente seção, foi dedicada a exploração de conceitos relacionados à RV e seus componentes e dispositivos, além de uma retrospectiva histórica. A próxima seção é reservada para a exploração dos conceitos de RV e sua relação com os AV para educação e treinamento na aprendizagem em AEC.

[29] Computadores fotônicos utilizam cabos óticos para tráfego de fótons, aumentando a velocidade de transferência de dados e largura de banda.

3.4 AMBIENTES VIRTUAIS APLICADOS AO ENSINO E TREINAMENTO

3.4.1 Perspectiva construtivista aplicada em ambientes virtuais

O construtivismo tem sua base fundamentada nas pesquisas do biólogo suíço Jean W. F. Piaget e faz parte do conjunto de correntes teóricas comprometidas em ilustrar como a inteligência humana se desenvolve. A teoria parte do princípio que o incremento da inteligência humana é um resultado das ações recíprocas entre o indivíduo e o meio. Considera a hipótese do indivíduo não nascer inteligente, e também não é um indivíduo passivo diante da influência que o meio pode proporcionar, ou seja, ele responde aos estímulos externos que atuam sobre ele para construir o seu próprio conhecimento, cada vez mais organizado e elaborado (LOPES, 1996). No construtivismo o instrutor/educador/professor não é visto como um condutor do aprendizado, mas um facilitador que pode provocar situações que irão contribuir para o enriquecimento do meio ao qual o aprendiz irá interagir para desenvolver o seu conhecimento (WEISS; CRUZ, 2001).

Conforme essa teoria o indivíduo pode construir seu conhecimento por meio das suas ações físicas ou mentais sobre os objetos (estímulo ou conhecimento, por exemplo) que estão contidos no meio e assim provocam um desequilíbrio ou conflito cognitivo[30] no indivíduo. Como resultado acontece o processo de assimilação dessas ações ou então o processo de acomodação e depois a assimilação para o indivíduo construir suas estruturas mentais ou cognitivas, denominadas de esquemas. Os esquemas são resultados dos sucessivos processos de construção de lógicas intelectuais cada vez mais complexas. Em outras palavras, uma vez que o indivíduo cognitivamente não consegue captar e organizar um objeto do meio para ampliar seus esquemas, ou seja, proceder com a assimilação, ele tenta fazer uma modificação ou acomodação de um esquema em função das peculiaridades do

[30] Cognição é o ato ou processo de conhecer, que envolve atenção, percepção, memória, raciocínio, juízo, imaginação, pensamento e linguagem, a palavra tem origem nos escritos de Platão e Aristóteles.

objeto para depois proceder com o processo de assimilação e finalmente, atingir o equilíbrio. O indivíduo constrói e reconstrói sucessivamente seus esquemas cognitivos para chegar cada vez mais ao equilíbrio (LOPES, 1996; MORAES, 2003).

McLellan (1996) destacou em seu trabalho que os AV são uma poderosa ferramenta para o aprendizado quando se utiliza a perspectiva construtivista, pois os dados não são percebidos como uma lista numérica oriunda de uma simulação, mas sim como um ambiente que envolve o indivíduo ou aprendiz de tal forma a contribuir no processo da construção do seu conhecimento pela exploração intuitiva. Num ambiente virtual o indivíduo pode fazer o mundo virtual e cuidar dele, pode interagir e aprender por meio de tentativas e de forma segura e ainda experimentar as possíveis consequências para então construir seu conhecimento. Este conhecimento pode ganhar maior potencialidade quando os próprios aprendizes se empenham em criar ambientes virtuais, pois conforme Flannery e Krauchunas (2000) os aprendizes provavelmente criam novas ideias quando estão comprometidos ativamente em criar novos projetos e compartilhar com outros, sendo uma importante experiência no sentido de poderem confirmar ou não suas hipóteses.

Atualmente, uma característica marcante das novas gerações é a computação estar tão integrada ao cotidiano e costume dos indivíduos por se apresentar praticamente de forma oculta (ou ubíqua[31]). E os indivíduos das novas gerações estão muito mais propícios à aprendizagem por meios visuais que inclusive podem ser combinados com entretenimento por meio de um AV. Assim, os recursos tradicionais como livros impressos, por exemplo, podem não resultar num conhecimento almejado pelo instrutor/educador/professor (BURIOL, 2011).

Pasqualotti (2000) comenta que os ambientes de ensino que utilizam tecnologias educativas como AV podem oferecer motivação ao aprendizado e ainda a experiência com uma interface de AV requer reflexão antes da resposta, ou seja, um pensamento mais consciente mesmo que algumas ações presentes nesse contexto já estejam dominadas de forma automática, como o uso do mouse e teclado. O autor apresenta tecnologias educativas em cinco grandes categorias: tutorial, exploratória, aplicativa, comunicativa. A categoria tutorial adota sequências

[31] Computação ubíqua diz respeito à integração dos computadores num ambiente onde os indivíduos utilizam sem percebê-los. A origem do termo foi apresentada primeiramente por Mark Weiser (BURIOL, 2011).

pré-definidas de informações no auxilio à aprendizagem como os sistemas multimídias contidos em um CD-ROM, por exemplo. Na categoria exploratória a informação e por sua vez o conhecimento são requeridos pelo aprendiz (usuário) como uma pesquisa na internet. A categoria aplicativa utiliza recursos de edição de texto e figuras como, por exemplo, as planilhas eletrônicas. A categoria comunicativa utiliza o conjunto hardware e software para gerar intercomunicação e pode ser exemplificada pelo correio eletrônico. E uma quinta categoria pode ser destacada que são as tecnologias com o foco construtivista e fornecem suporte à construção de ambientes virtuais (PASQUALOTTI, 2000). O termo tecnologia educativa surgiu no século XX e estava relacionado primeiramente com a forma de modernização do ensino em sala de aula. Depois esse conceito sofreu uma evolução remetendo a ideia de como o ensino poderia ser melhorado (otimizado) e por último o conceito está focalizado nas necessidades de mudança do ensino (BLANCO; SILVA, 1993).

3.4.2 Ambientes virtuais educacionais e ambientes virtuais de treinamento

Hounsell, Silva e Miranda (2008) apresentam conceitos para distinguir os AV com ênfase em educação e os AV com ênfase em treinamento. Para esses autores a diferença entre treinar e educar reside no aspecto mais específico do primeiro e no aspecto mais abrangente do segundo. Ou seja, o treinamento busca compreender determinados procedimentos relativos a uma tarefa e desse modo melhorar as habilidades relacionadas àquela tarefa. Já a educação proporciona incrementos intelectuais sobre um tema pela reflexão e análise, ponderando menos a preocupação em relação ao tempo e execução, como acontece num treinamento.

Em um AV para treinamento, o conteúdo da aprendizagem é obtido ao se observar e executar procedimentos simulados, mas, em contrapartida, um AV voltado para educação possibilita aprender o conteúdo mediante comparações e reflexões com base teórica.

No contexto pedagógico a ênfase em um AV voltado para a educação está em atividades não exaustivas por meio de explicações e visualizações. A pedagogia

em um treinamento foca a repetição dos procedimentos, relembrando-os por comandos e ordens.

A comunicação do conteúdo em um treinamento proporcionado por um AV tem uma característica mais direta, pois geralmente é voltada para um indivíduo que está usando o AV para melhorar suas capacidades relativas à tarefa simulada pelo treinamento. Isto pode ser realçado quando se avalia um aprendiz, pois o resultado final é considerado como sucesso se a tarefa for bem executada. Na abordagem educativa o enfoque recai sobre um ambiente colaborativo, onde a comunicação também ocorre na interação do instrutor/educador/professor com seus aprendizes, avaliando-os num processo mais contínuo.

Em outra pesquisa desenvolvida por esses autores é possível identificar, mesmo que quantitativamente, se um ambiente virtual tem uma ênfase no ensino ou no treinamento, e ainda é possível existir um ambiente virtual cujo enfoque seja voltado tanto para a educação como para o treinamento (HOUNSELL; SILVA; MIRANDA, 2008). Mas em ambos os casos o objetivo ainda é o mesmo: o usuário está experimentando uma simulação em um AV para aprender. Assim, a perspectiva construtivista pode ser abordada para se desenvolver um AV com enfoque educacional ou com enfoque em treinamento.

3.4.3 Considerações sobre o uso de ambiente virtual para ensino e treinamento

3.4.3.1 Ambiente virtual

Quando for adotado o uso de ambientes virtuais para ser usado no uso de treinamento ou educação vale lembrar que algumas considerações devem estar definidas no que diz respeito ao seu desenvolvimento.

A primeira consideração a ser feita é que o AV deve ser desenvolvido para ser uma interface entre indivíduo e o computador e ser capaz de fornecer um cenário tridimensional sintético para prover o usuário de formas de imersão e

interatividade sobre uma cena ou simulação mediante dispositivos de entrada e saída (STUART, 1996). Além da cena referente ao AV, o espaço físico ocupado também deve ser ponderado para considerar fatores como acesso, segurança, área de ocupação, iluminação e acústica. Por exemplo, uma *CAVE* requer uma área física considerável para ser implementada, sendo difícil seu emprego em uma sala de aula.

Goméz e Trefftz (2011) explicam que um AV é uma ferramenta poderosa para suporte no processo de aprendizagem, a ponto de existir a possibilidade de incluir conceitos abstratos para originar aumento do desempenho do indivíduo. E o poder dessa ferramenta é aumentada quando se usam tecnologias de RV imersiva tais como *cab simulator* e *CAVE*. Mas por causa de problemas como limitação de hardware, das tecnologias de RV que nem sempre estão popularizadas ou acessíveis, o custo de aquisição de dispositivos e a rápida desatualização são algumas das dificuldades que fazem um AV ainda não estar incorporado e difundido no suporte ao processo de aprendizagem. Isso é mais realçado, principalmente, em países ditos emergentes ou em desenvolvimento, como o caso do Brasil, cuja economia se encontra em processo de crescimento e dificultando o acesso financeiro para adotar novas tecnologias no processo de aprendizagem.

Veronica Pantelidis (1998) sugere o emprego de um ambiente virtual para educação e treinamento quando ocorrem as seguintes situações:

a) Quando o ensino ou o treinamento no mundo real oferecem um perigo tanto para o aprendiz como para o meio ambiente, ou quando existem inconveniências de ensinar ou treinar no mundo real, ou mesmo quando é impossível pôr em prática. Por exemplo, nem sempre é possível levar os alunos de um curso de Arquitetura ou Engenharia Civil em uma obra para que seja ofertado algum ensinamento na área da construção. Algumas obras podem ser até perigosas como, por exemplo, escavações para construção de um túnel.

b) Um modelo irá treinar ou ensinar com a mesma eficiência quando se dispõe de uma aprendizagem no mundo real. Os alunos de engenharia podem aprender a execução de uma atividade típica de construção ao interagir em um ambiente virtual assim como visitar uma obra e verificar a execução dessa mesma atividade.

c) A interatividade proporcionada pelo ambiente virtual pode ser considerada motivadora tanto quanto a interação com algo real ou ainda promover maior motivação quando se usam técnicas de *games*, por exemplo. Então, algumas tarefas podem usar movimentos físicos para tornar mais interessante o aprendizado.

d) O desenvolvimento de um ambiente virtual ou de um modelo para ser simulado é objeto de importância para a construção da aprendizagem.

e) Vencer algumas barreiras como introspeção para que se consiga atingir a aprendizagem em grupo ou dar oportunidade aos aprendizes que tenham alguma deficiência para fazerem experimentos e tarefas dos quais não poderiam, de outro modo,participar.

3.4.3.2 Usuário de um ambiente virtual

O perfil do usuário é outro fator a ser considerado no desenvolvimento de um AV. Quando o AV tem sua aplicabilidade voltada para aprendizagem e treinamento, o aprendiz é o usuário do sistema de RV. Um erro que não deve ser cometido quando se define um ambiente virtual é pressupor que o usuário é especialista do sistema. Deve ser levado em consideração que certos usuários podem requerer algum tempo para adquirir destreza na interatividade com os dispositivos do ambiente virtual. Assim, para a construção de um ambiente virtual deve ser avaliada não somente a quantidade de usuários que podem interagir ao mesmo tempo, mas também suas diferenças, tais como deficiência física, por exemplo, para que seu desempenho no uso do AV não seja afetado (STUART, 1996).

Não pode ser esquecido que o número de dispositivos também contribui para a delimitação do ambiente virtual no que diz respeito ao número de usuários interagindo ao mesmo tempo durante uma mesma sessão da simulação. Outra questão a ser considerada quando se desenvolve um AV é se os usuários estarão

presentes no local do mesmo ou se existirá a necessidade de ser instalada alguma rede de conexão remota (STUART, 1996).

3.4.3.3 Tarefas relativas a um ambiente virtual

A definição do conjunto de tarefas é outro fator de relevância a ser considerado quando se especifica a aplicação do AV. Para isso, existe a necessidade de identificação das metas a serem atingidas para assim definir quais tarefas serão requeridas e de que forma serão executadas num AV. Destacam-se algumas tarefas que são comumente encontradas em AV: a navegação, a busca, a manipulação, a percepção e o aprendizado. Assim, uma aplicação como uma simulação de um passeio por um modelo virtual de uma edificação pode usar as tarefas supracitadas (STUART, 1996).

Em Pompeu (1999) pode ser encontrada uma classificação para as tarefas que podem ser designadas em um AV, a saber:

a) Execução *online* que usa o AV como interface para executar tarefas no mundo real tendo como exemplo a teleoperação.

b) Treinamento *offline* que usa o AV para a prática de atividades que depois serão executadas no mundo real como as simulações de guerra, por exemplo.

c) Compreensão *online* para ganhar conhecimento e confiança ao interagir com o ambiente virtual como o planejamento de radioterapia no tratamento do câncer.

d) Aquisição do conhecimento e aprendizagem *offline* onde o usuário é conduzido a adquirir conhecimento para depois sintetizá-lo em mais conhecimentos abstratos como acontece nos laboratórios virtuais.

e) Projeto *online* onde o AV é usado como interface para projetar objetos a serem mostrados pelo sistema como uma visita a um edifício virtual.

f) Entretenimento que busca trazer a diversão para os AV e assim propor um prazer experimental aos usuários como no caso de jogos.

g) Comunicação e ferramentas para pesquisa das capacidades perceptivo-motoras humanas para estudo das capacidades do ser humano.

3.4.3.4 Qualidade de um ambiente virtual de treinamento e educação

As características relevantes para a qualidade de um AV de treinamento e educação estão estreitamente relacionadas àquelas que são relevantes para a qualidade de software voltado às mesmas aplicações. Assim, a primeira consideração recai sobre as características pedagógicas e estas englobam um conjunto de atributos que demonstram a convivência e a viabilidade de utilização de um AV em condições de aprendizagem. Compreende as seguintes: identificação do ambiente e o modelo de aprendizagem, clareza dos conteúdos, a relação com o programa curricular, facilidade de uso, motivação, densidade informacional e tratamento de erros (GAMA, 2007).

A segunda consideração sobre a qualidade visa às características ergonômicas ou de usabilidade: conjunto de atributos que demonstram a usabilidade do software, tais como: facilidade no processo de aprendizagem e de memorização, os meios disponíveis de condução do usuário na interação com o AV, afetividade, consistência da interface, a avaliação da adequação entre o objeto e sua referência, controle de erros e os mecanismos para a sua correção.

A terceira consideração é a adaptabilidade, ou seja, a capacidade de um AV de se adaptar às necessidades e preferências do usuário e ao ambiente de aprendizagem, assim como se adequar ao modelo e aos objetivos almejados para a aprendizagem. E por último, a consistência da documentação para a instalação e uso do AV.

3.4.3.5 Desempenho do sistema

Uma consideração de muita relevância no desempenho do sistema de AV está no controle da taxa de quadros para visualizar uma cena em um ambiente virtual.

Quanto mais a cena virtual se aproxima do mundo real, isto é, quanto maior o realismo conforme a percepção humana, maior é a quantidade de detalhes encontrados na cena que implica um maior número de polígonos para representar os objetos e consequentemente, o uso mais intenso dos dispositivos gráficos. Espera-se que uma cena virtual tenha sua taxa de quadros com pelo menos 24 *fps* (*frames per second*, ou quadros por segundo) assim a visualização do movimento ou animação em uma cena virtual não fica comprometida.

Os demais dispositivos também influenciam no desempenho do sistema como a velocidade de processamento, a memória necessária para armazenar as informações que o ambiente virtual utiliza, a durabilidade destes dispositivos em relação ao custo, o software e a forma com que gerencia a interatividade para não gerar uma latência entre o comando do usuário e a resposta do sistema (STUART, 1996).

3.4.4 Ambientes virtuais de aprendizagem na engenharia, arquitetura e construção

Haque (2001) da Universidade de Texas A&M, nos EUA, desenvolveu um ambiente virtual não imersivo para o ensino de estruturas de concreto armado a ser aplicado aos alunos do curso de engenharia civil usando os recursos da internet combinados com a linguagem de programação JAVA e os formatos HTML e VRML. O AV desenvolvido foi estruturado e modulado conforme conteúdo e nível de conhecimento do usuário sobre o conteúdo e dividido em cinco conceitos de visualização, sendo eles: apresentação em HTML, visualização e animação tridimensional, imagens manipuladas, simulação interativa e navegação. Na apresentação em HTML o usuário é guiado ao material didático sobre conceitos de

análise de projetos de estruturas de concreto por slides de PowerPoint e links para outros sites sobre o assunto. O módulo de animação e visualização 3D utiliza animações em formato GIF[32] para representar como a estrutura se comporta sobre uma determinada ação, mostrando o comportamento da estrutura e como ela entra em colapso. Imagens de estruturas de concreto são manipuladas por animações e setas semitransparentes para indicar informações que auxiliam os usuários a compreender problemas mais complexos. Para reforçar a aprendizagem o módulo de simulação interativa faz com que o usuário possa alterar a intensidade da ação e dimensões da peça estrutural. Por último, a navegação auxilia na compreensão das peças estruturais. O autor conclui que o ambiente virtual realmente contribui na aprendizagem pela visualização e interatividade.

Santos e Duarte (2005) pesquisaram o impacto de tecnologias de RV, em particular a Realidade Aumentada e interatividade com *datagloves* e estereoscopia, para então possibilitar o desenvolvimento de um AV onde alunos dos primeiros anos de cursos como engenharia e arquitetura, em especial a disciplina de Geometria Descritiva, pudessem desenvolver a capacidade de visualizar elementos tridimensionais e assim não se desestimulassem e abandonassem tais cursos.

Na Universidade da Califórnia, nos EUA, foi desenvolvida uma sala de aula conceitual chamada de *vizclass* que utilizava algumas tecnologias de RV para serem aplicadas ao processo de aprendizagem da disciplina de método dos elementos finitos[33] aos alunos de graduação e pós-graduação de engenharia civil, num período entre 2003 e 2005. O ambiente de aprendizagem contava com um software de simulação de terremotos em sistemas de engenharia chamado *OpenSees – Open System for Earthquake Engineering System* (Pacific Earthquake Engineering Research Center), um software de visualização para os resultados do método dos elementos finitos chamado de *SketchFEA*, três telas *touch screen whiteboard* 2D de 72 polegadas, um projetor estereoscópico, um *cluster* de 8 núcleos, um *trackball*, um teclado *wireless* e um sistema de som *surround*. Foi adotada a seguinte metodologia para avaliar a pedagogia: os alunos e o professor foram submetidos a um questionário avaliando o processo de aprendizagem. Os resultados obtidos apontam

[32] GIF (Graphics Interchange Format) é um formato de imagem bitmap introduzido pela CompuServe em 1987 cuja característica é a portabilidade.

[33] Método dos elementos finitos é método numérico que subdivide um domínio 2D ou 3D em um número finito de elementos cuja formulação matemática e física é conhecida, resultando a princípio, na resolução de um sistema linear de equações para estudar um dado fenômeno.

um melhor desempenho por parte dos alunos, gerando melhores trabalhos, mais interesse e comprometimento com os deveres propostos pelo professor da disciplina. Ainda, o professor da disciplina relatou que houve uma melhora na forma de abordar a disciplina, aumentando a capacidade de expor mais conteúdo, além de acréscimo do estímulo às discussões dentro e fora da sala de aula e entusiasmo ao usar o *whiteboard* (GRIMES *et al*, 2006).

A Escola de Construção da Universidade do Sul do Mississipi, nos EUA, usou ferramentas de RV e games 3D para auxílio pedagógico no ensino de arquitetura ao introduzir aos alunos de graduação e pós-graduação duas disciplinas: modelagem 3D e animação e RV aplicada. O objetivo era aperfeiçoar as habilidades dos alunos na visualização tridimensional com o uso de fotorrealismo em navegação em tempo real por meio do ensino de ferramentas como *engine* C4Engine (Theraton Software), Google *SketchUp* (Google) para modelagem geométrica, e Artlantis (Abvent Group) para fotorrealismo e animação e uma ferramenta para criação de ambientes virtuais desenvolvida sobre a *engine C4Engine* (Theraton Software) chamada BuildIT4.O estudo teve como foco o *feedback* dos alunos depois de cursarem as disciplinas já citadas, tendo como resposta a motivação e capacidade dos alunos em usar RV em exemplos de arquitetura, além da necessidade de melhor *hardware* e certo desconforto no uso das ferramentas por estarem acostumados com aplicativos CAD (SHIRATUDDIN; FLETCHER, 2007).

Sampaio, Henriques e Martins (2008) apontam a necessidade de novas formas de ensino para os cursos de engenharia civil, visando proporcionar aos alunos uma melhor forma de entendimento sobre a relação entre etapas, tarefas, prazos e processos construtivos envolvidos numa construção real. Por meio de dois modelos virtuais, sendo eles uma parede típica de um edifício e uma construção de uma ponte, especialistas nessas áreas foram consultados para auxiliar no desenvolvimento dos modelos buscando a acurácia bem como a eficiência para a didática. A pesquisa aplicava aplicativos de RV (EON Studio) e modelagem tridimensional por computador (AutoCad) para explicar a evolução dos modelos reais usando RV. Alunos dos cursos de desenho técnico, processos construtivos e pontes podiam interagir com os modelos selecionando quantidades, alterando dimensões, controlando cronogramas e prazos e forma de construção.

A concessionária de energia paranaense COPEL, em parceria com a Universidade Federal do Paraná e o Instituto de Tecnologia para o Desenvolvimento

(LACTEC), desenvolveu um AV cujo objetivo era o desenvolvimento e modelagem de um treinamento de atividades de manutenção em linha viva utilizando técnicas de RV, estabelecendo um novo modelo de treinamento ao complementar o processo existente. O AV foi desenvolvido usando o *toolkit* de código aberto chamado *OpenSceneGraph. A* simulação da física foi implementada usando *Bullet Physics,* duas TV Mitsubishi de 73 polegadas, modelagem geométrica usando o Autodesk 3DS Max e dispositivos de interatividade pelo *Wiimote* (Nintendo) (BURIOL *et al*, 2009).

Na Universidade Pennsylvania State, nos EUA, foi desenvolvida experiência com AV em torno de um estudo de caso, o hotel de MGM Grand em Las Vegas, Nevada. Por meio do desenvolvimento de um software chamado *Virtual Construction Simulator* 4D (VCS4D) com base na *engine Deep Creator* (conhecida atualmente como *Experient Creator*), dez grupos de estudantes de engenharia foram solicitados para desenvolver um cronograma com o caminho crítico da produção para um piso do hotel e seus componentes, incluindo paredes, assoalhos, peças e conexões estruturais. Como conclusão a interatividade adicionada no VCS4D ajudou estudantes a desenvolver cronogramas com mais qualidade e forneceu uma experiência de aprendizagem agradável. A execução de VCS4D provou também incentivar o trabalho colaborativo em grupo e promover maior geração de soluções com a melhor visualização de processos da construção (NIKOLIC; LEE; MESSNER; ANUMBA, 2010).

O curso de arquitetura e o curso de ciência da computação da Universidade Federal do Ceará estão desenvolvendo um projeto de ensino chamado de imagem espaço – imagem objeto. Através de um AV imersivo procura-se envolver o usuário no exercício de explorar soluções para projetar espaços arquitetônicos. O projeto está configurado para pesquisas de novas estratégias para percepção de espaços imersivos, o uso de modeladores tridimensionais e metodologia BIM[34] e oficina para prototipagem rápida. O projeto foi explicado no artigo de D. Cardoso (CARDOSO *et al*, 2010).

[34] BIM (Building Information Modeling) é o processo de gestão de informações referentes a um edifício durante o seu ciclo de vida, representado por um modelo (formato de dado) que carrega a sua geometria, propriedades, relações e atributos.

Um sistema está em desenvolvimento como um recurso de aprendizagem e de ensino com um grande grupo de estudantes de gerência da construção da Universidade de New South Wales, Austrália. A análise é usada para investigar um dado projeto específico e as opções da construção. Pela modificação da *engine CryENGINE* 2, os alunos exploram a possibilidade de diversas ferramentas para a construção de residências (NEWTON; LOWE, 2011).

Lin *et al* (2011) propõem um AV detalhado para treinamento de segurança em canteiros de obra onde estudantes de AEC assumem o papel de inspetores de segurança e navegam na cena do jogo para identificar potenciais perigos. O jogo foi projetado com as características tais como realismo, *self-learning*, não linearidade, interatividade através do uso da *engine* do Torque 3D para executar o sistema do jogo e modelagem geométrica usando 3DS MAX (Autodesk), *MilkShape* 3D (Chumbalum Soft) foi usado para criar os objetos 3D não disponíveis. Os resultados de testes com alunos indicaram que o uso jogo aumentou o interesses de aprendizagem, apreciaram a aprendizagem em si e foram motivados a atualizar seus conhecimentos em segurança.

3.5 CONSIDERAÇÕES FINAIS

O capítulo 3 tratou a educação e treinamento em AEC por meio de AV que utilizam como base a RV. Assim, foram explorados os temas que definem o conceito sobre RV, os seus componentes e dispositivos que formam e classificam um AV educacional e de treinamento. Também foram explorados a relação dos AV com o processo de aprendizagem em AEC, relatando alguns trabalhos de cunho científicos realizado no mundo.

O próximo capítulo da dissertação trata sobre o desenvolvimento de um protótipo de ambiente virtual educacional que utiliza os conceitos tratados neste capítulo.

4 AMBIENTE VIRTUAL PROTÓTIPO PARA TREINAMENTO NA CONSTRUÇÃO CIVIL

Este capítulo descreve a implementação de um ambiente virtual protótipo para ser utilizado em treinamento de atividades típicas da construção civil. As ferramentas, códigos e sequências e atividades também são explicadas.

4.1 CONSIDERAÇÕES INICIAIS

A questão de qual técnica de construção será mais eficiente ou resultará num produto com maior qualidade depende do perfil da obra assim como da equipe responsável pelo empreendimento. Mesmo quando se busca padronizar as atividades de construção conforme normas e processos construtivos conceituados cientificamente existem dificuldades na escolha de uma tarefa que possua a melhor sequência de passos ou uma única forma de ser abordada. Até mesmo a escolha da metodologia de execução de uma atividade em uma obra de construção civil depende de fatores que podem ultrapassam os conhecimentos adquiridos durante a formação acadêmica do engenheiro.

O protótipo aqui descrito foi desenvolvido como um ambiente virtual não imersivo e pode ser classificado como uma RV *world-on-window* ou *desktop*. Tal escolha se deu pela relativa facilidade na realização das implantações da simulação, ao invés de se gastar tempo em sincronizar e programar dispositivos tecnológicos típicos de RV classificadas como imersivas. Ainda, as RV não imersivas oferecem a vantagem do custo reduzido em relação aos AV imersivos os quais exigem dispositivos que nem sempre estão acessíveis financeiramente. As RV não imersivas também não exigem espaço físico exclusivo, o que constitui outra vantagem, dada a baixa disponibilidade de espaço em instituições de ensino brasileira. O protótipo pode ser usado em um único computador numa sala de aula como uma forma de visualizar a atividade simulada ou mesmo ser instalado em um

laboratório com diversos computadores, por exemplo, aqueles que servem para ensinar ferramentas computacionais aplicadas a AEC como CAD.

A proposta deste trabalho foi simular a atividade de montagem de uma fôrma para moldar um pilar de concreto armado. Conforme Rezende (2010) as estruturas em concreto armado são amplamente utilizadas no Brasil e no mundo. Fôrmas são estruturas temporárias com o objetivo de sustentar o concreto dito fresco, ou seja, enquanto não atinge um estado sólido com resistência suficiente para ser autoportante. Mas mesmo sendo uma tarefa muito comum na construção civil brasileira e independente do tamanho da obra, existem mais de um modo para abordá-la, em parte por causa das diversas técnicas construtivas e em parte por causa das escolhas adotadas pelos engenheiros e encarregados em executar a obra quando existe a possibilidade de combinar essas técnicas. Em Calil Junior *et al.* (2000) foram encontrados pelo menos seis metodologias de execução de fôrmas para moldar peças de concreto armado conforme as recomendações da Associação Brasileira de Cimento Portland (ABCP). Mesmo nos cursos de Engenharia Civil no Brasil não são contempladas todas as variações das técnicas existentes de construção. É por isso que o ambiente virtual protótipo aqui proposto adotado serve como complemento ao instrutor e cabe a ele averiguar quando é possível adotar ou modificar o treinamento conforme seu plano de ensino.

Para a implementação do ambiente virtual protótipo foi usado o pacote de visualização (*toolkit*) VTK versão 5.4 para auxiliar na implementação da interatividade durante a simulação.

A modelagem geométrica da cena foi desenvolvida pelo software Autodesk 3DS Max, pois permite a criação com certa facilidade da cena virtual além de oferecer ferramentas para exportar a malha geométrica dos modelos num formato compatível com o VTK.

Um notebook equipado com o processador Intel Core i7-2630 2 GHz, memória RAM de 6GB, usando o sistema operacional Microsoft Windows 7 64 bits, com uma placa de vídeo NVIDIA GeForce GT 540M foi utilizado para servir como hardware. Assim, o ambiente virtual protótipo foi concebido a princípio para ser uma realidade virtual *world on window* ou *desktop* e a interatividade ocorre por meio de comandos realizado por um mouse. Esta escolha reside no fato de ser um atrativo em relação às caras plataformas utilizadas em ambientes AV imersivos, além de

proporcionar o desenvolvimento de uma ferramenta que não tenha um custo elevado e possa ser fácil de usar tanto em uma sala de aula como em um laboratório.

Para desenvolver as rotinas pertinentes à funcionalidade da simulação da atividade foi adotada a linguagem C++ e o compilador Microsoft Visual C++ 2008 *Express Edition*.

4.1.1 Descrição do ambiente virtual e da cena

A cena virtual consiste de um canteiro de obras situado num ambiente urbano virtual, cercado por edificações e ruas. Optou-se por não carregar a cena com muitos objetos para não haver perda no desempenho do ambiente virtual protótipo. O canteiro de obras é delimitado por tapumes e possui um alojamento para funcionários e um abrigo que serve como almoxarifado, abrigando ferramentas, materiais construtivos e algumas das peças da fôrma. No canteiro de obras existe uma estrutura de concreto armado que se encontra com a fundação, blocos e vigas de baldrames já executadas. Existem oito pilares de concreto armado sendo sete deles já estão concretados. O cenário foi modelado no Autodesk 3DS Max e exportado no formato adequado para ser importado pelas classes do VTK que serve como núcleo para a programação da interatividade e sequência da tarefa.

O ambiente protótipo apresenta uma tela inicial e as demais telas surgem na sequência informando ao usuário sobre as características da atividade pelas ilustrações e texto informativo (figuras 4.1 e 4.2). Finalmente abre-se a tela para cena. No canto superior direito da janela de visualização da cena existe uma legenda que informa ao usuário o que ele deve fazer durante a atividade, indicando qual a peça na sequência e sua localização no canteiro de obras virtual.

A navegação pela cena é fornecida ao usuário pelo ambiente virtual protótipo pela câmera virtual implementada pelas rotinas do VTK. Durante a sincronização entre os objetos modelados e as rotinas de interface e interatividade foram testados os recursos oferecidos pelo VTK na manipulação da câmera virtual. Optou-se por aquela que simula uma navegação *fly* para prover ao usuário maior liberdade tanto para explorar a cena como para executar a atividade e facilita a

movimentação das peças para as possíveis posições que o usuário pode aproveitar para o seu campo de visão, conforme mostra a figura 4.3. Apesar desta navegação não representar como um sujeito executa a montagem da fôrma no mundo real, a ideia é transmitir conhecimento aos usuários sobre as técnicas construtivas usando RV como suporte ao instrutor pelo estímulo que a RV provê ao aprendiz como interface intuitiva ao aprendizado.

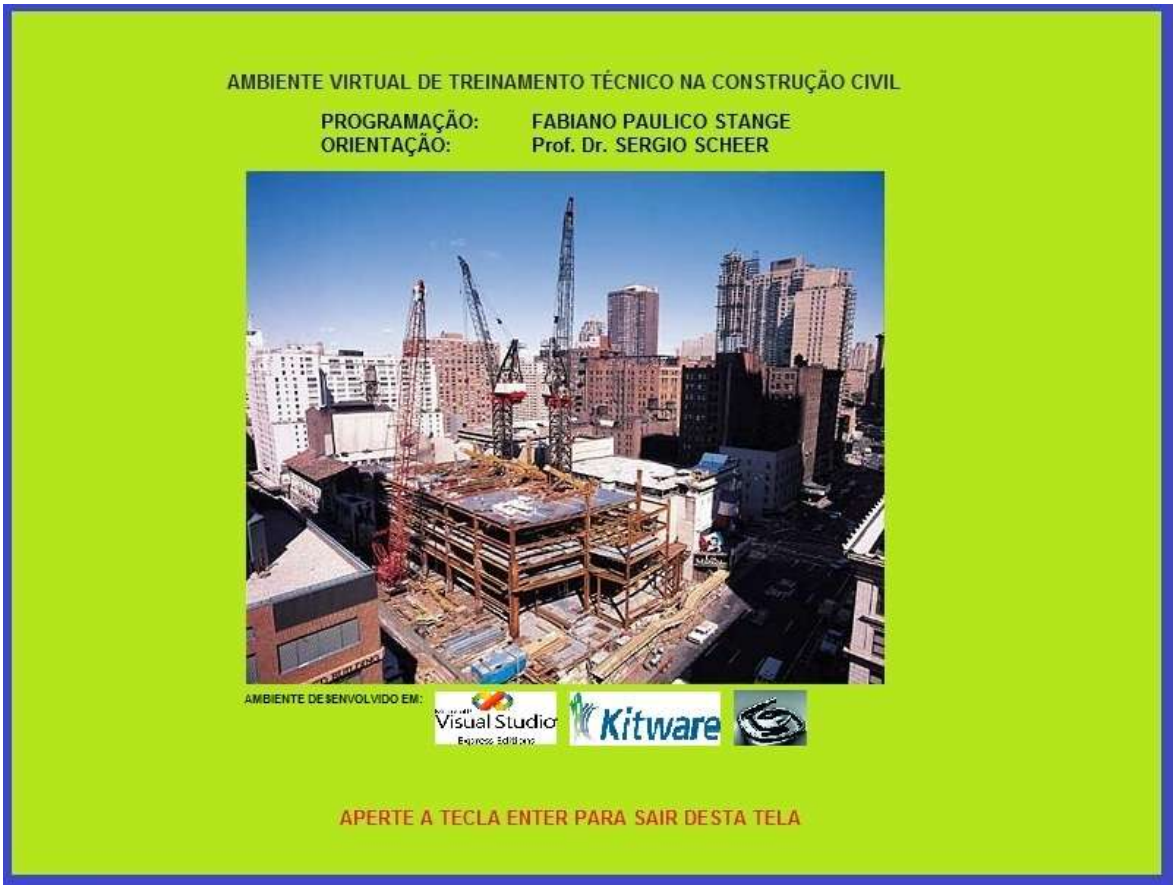

FIGURA 4.1: Apresentação da tela inicial do ambiente virtual protótipo.
FONTE: Autor(2012).

FIGURA 4.2: Tela informativa.
FONTE: Autor(2012).

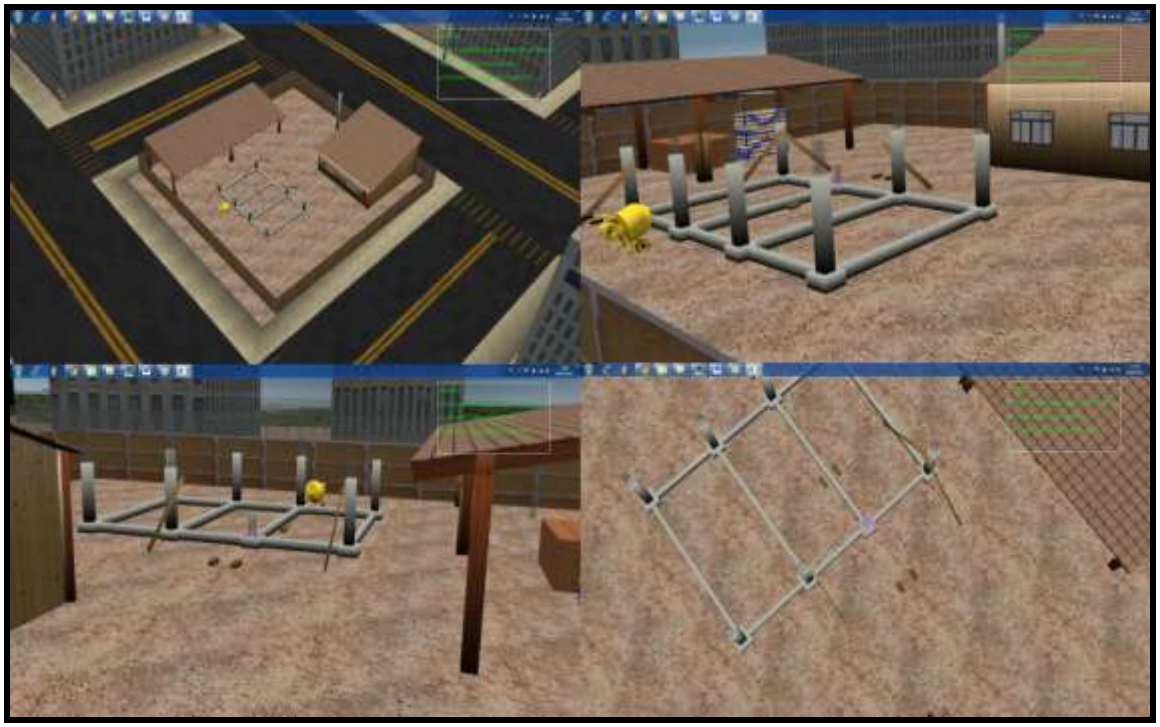

FIGURA 4.3: Diversas posições do campo de visão da cena do ambiente virtual.
FONTE: Autor(2012).

4.1.2 Descrição da atividade simulada

A atividade a ser simulada será executar o processo de montagem de uma fôrma para moldar o pilar em concreto armado que falta ser executado. As peças já estão produzidas e se encontram no canteiro de obra virtual.

Durante a simulação de montagem, quando se escolhe uma peça relacionada à sequência de montagem, existe outra peça idêntica na cena, na a cor magenta, e indica ao usuário a posição que a peça escolhida deve ser locada. Assim ocorre com todas as peças até o usuário completar a atividade de montagem da fôrma. As figuras 4.4 e 4.5 ilustram uma parte da atividade simulada.

Como mencionado anteriormente, existem pelo menos seis técnicas para se executar uma fôrma de madeira para moldar um pilar de concreto. Essas técnicas se baseiam no sistema tradicional descrito no boletim técnico n. 50 de 1943 da ABCP e servindo de base para as demais técnicas (CALIL JUNIOR *et al.,* 2000). As recomendações desse sistema são:

a) Execução das fôrmas conforme as dimensões das peças de concreto descritas no projeto estrutural;

b) A fôrma deve possuir rigidez suficiente para não ocasionar defeitos na moldagem das peças de concreto;

c) Evitar perda de cimento durante o molde da peça com concreto (concretagem) observando a estanqueidade da fôrma;

d) As fôrmas devem estar projetadas para permitir a retirada de suas partes com facilidade, devem ser executadas permitindo o maior número de reutilizações.

Os painéis que compõem a fôrma são de madeira de lei, formadas por tábuas de seção transversal 1”x4” ou por caibros de seção transversal 3”x3” . Eles são encaixados conforme as dimensões e forma da seção transversal do pilar a ser concretado. Elementos de travamento dos painéis chamados de gravatas são posicionados para evitar que a fôrma abra durante o lançamento do concreto na

forma fluida. Montantes feitos de caibros de 3”x3” são posicionados para reforçar as gravatas e os painéis e ligados por ferros de seção circular ou tirantes. A figura 4.6(a) mostra um exemplo da fôrma neste sistema de execução.

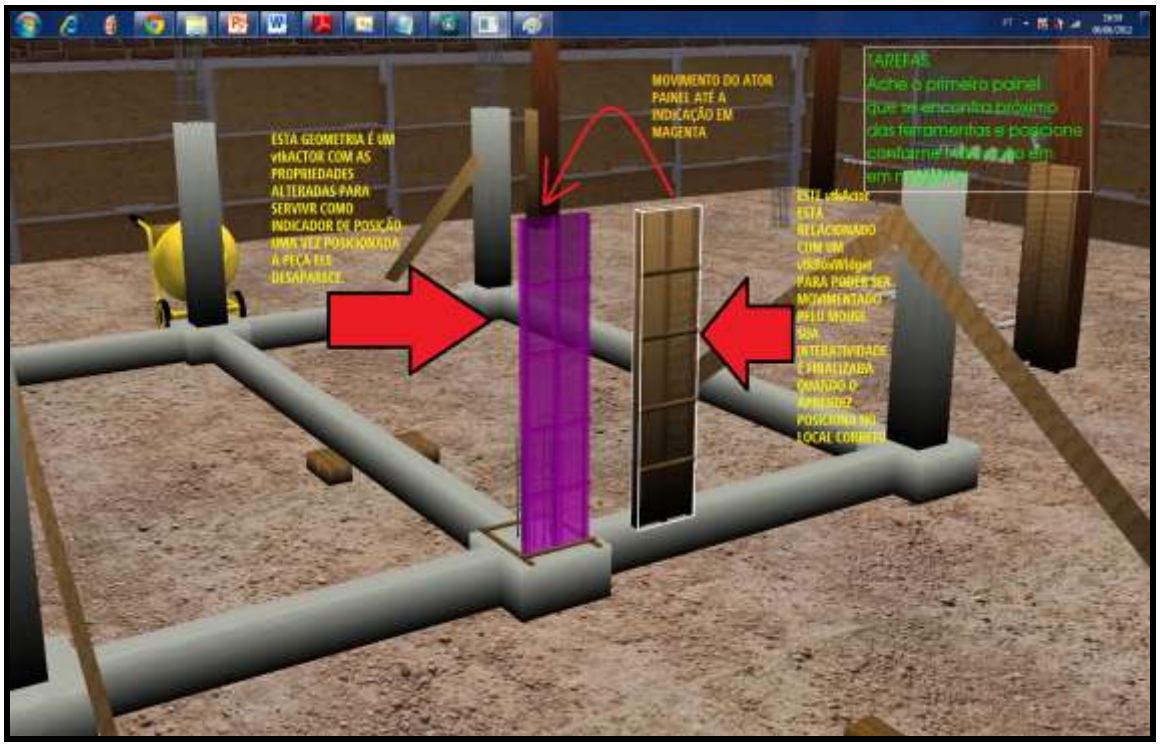

FIGURA 4.4: Movimentação do primeiro painel conforme indicação em magenta.
FONTE: Autor(2012).

Em 1960 foi desenvolvido o sistema de fôrmas Ueno que apresentava duas novidades: a elaboração de projeto para fabricação de fôrmas e a substituição de peças de madeira serrada por chapas de madeira compensadas[35] na confecção de painéis conforme três elementos típicos: lajes, vigas e pilares que está mostrado na figura 4.6(b). A confecção das fôrmas era executada em obra e, ao invés de usar gravatas para enrijecer os painéis das fôrmas para os pilares, são usados sarrafos posicionados verticalmente e travados ao longo do comprimento por barras de ferro de seção circular (JUNIOR *et al.,* 2000).

O sistema de fôrmas FORMAPRÉ, criado em 1980, aperfeiçoou o sistema Ueno fazendo com que fosse mais versátil, principalmente em relação às obras com pouco espaço, pelo fato de ser possível confeccionar as fôrmas fora da obra, reduzindo espaço para estoque.

[35] Chapas de madeira compensadas são constituídas de um número ímpar de lâminas e estão dispostas de tal modo que as fibras das lâminas alternadas sejam paralelas e a direção das fibras adjacentes forme um ângulo de 90º (CALIL JUNIOR *et al*, 2000).

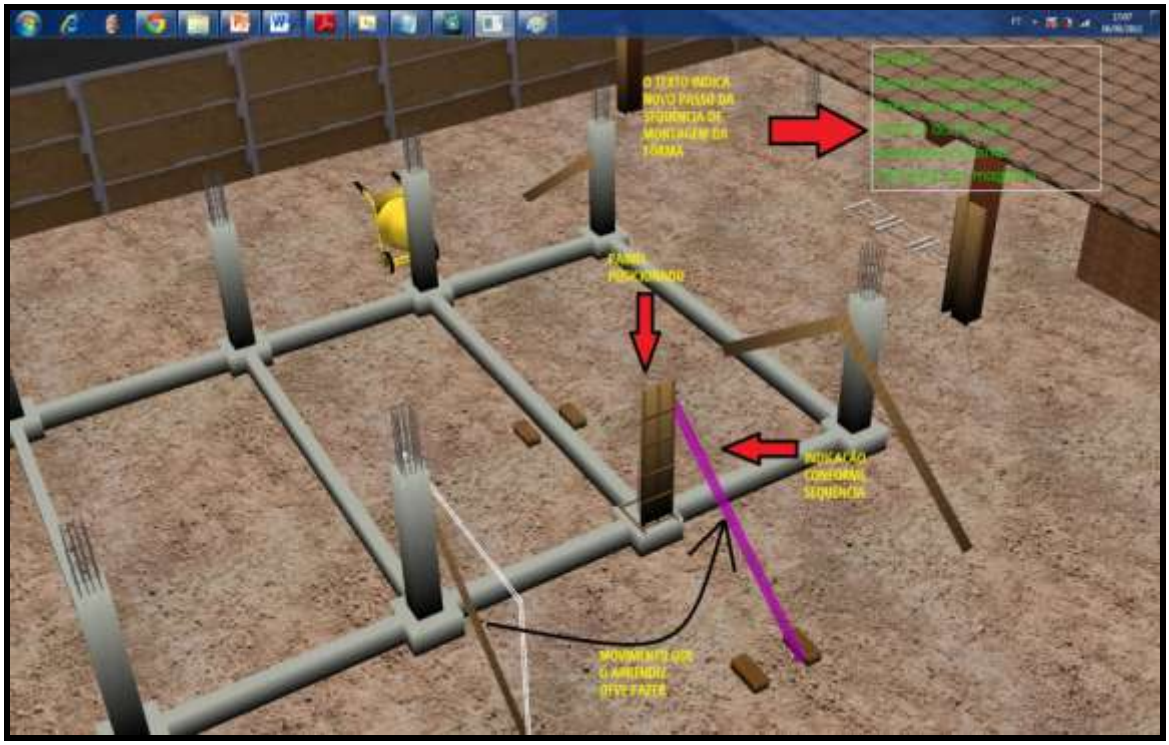

FIGURA 4.5: Posição confirmada do primeiro painel e nova indicação em magenta.
FONTE: Autor(2012).

Os painéis são confeccionados por chapas de madeira compensada com espessura de 14 mm e enrijecidas por tensores[36]. A figura 4.7(a) apresenta a fôrma de pilar que usa este sistema (CALIL JUNIOR *et al.,* 2000).

O sistema FORMAPRONTA é um sistema que industrializou o sistema Ueno agregando velocidade e mais precisão. Os painéis são feitos de chapas de madeira compensada e enrijecidos por tensores. O prumo e a estabilidade da fôrma são garantidos por sarrafos de prumo (CALIL JUNIOR *et al.,* 2000).

Baseado nos sistemas Ueno e FORMAPRÉ, o sistema PRÁTIKA, ilustrado pela figura 4.7(b) tem a característica do grande reaproveitamento das peças ao usar equipamentos metálicos além da possibilidade de adaptar o conjunto de fôrma tanto para andares típicos como para andares atípicos (CALIL JUNIOR *et al.,* 2000).

O sistema GETHAL também utiliza chapas de madeira compensada, mas com espessuras de 18 mm para elementos retos e 12 mm para elementos curvos e o arranjo das escoras conforme sistema GETHAL de escoramento. A figura 4.8 apresenta alguns casos para o sistema GETHAL.

[36] Tensores são barras de ferros que conectam peças que enrijecem os painéis da fôrma. No Brasil são usados ferros com seção circular cujo raio varia entre 3/16” a 5/8” (Junior et al, 2000).

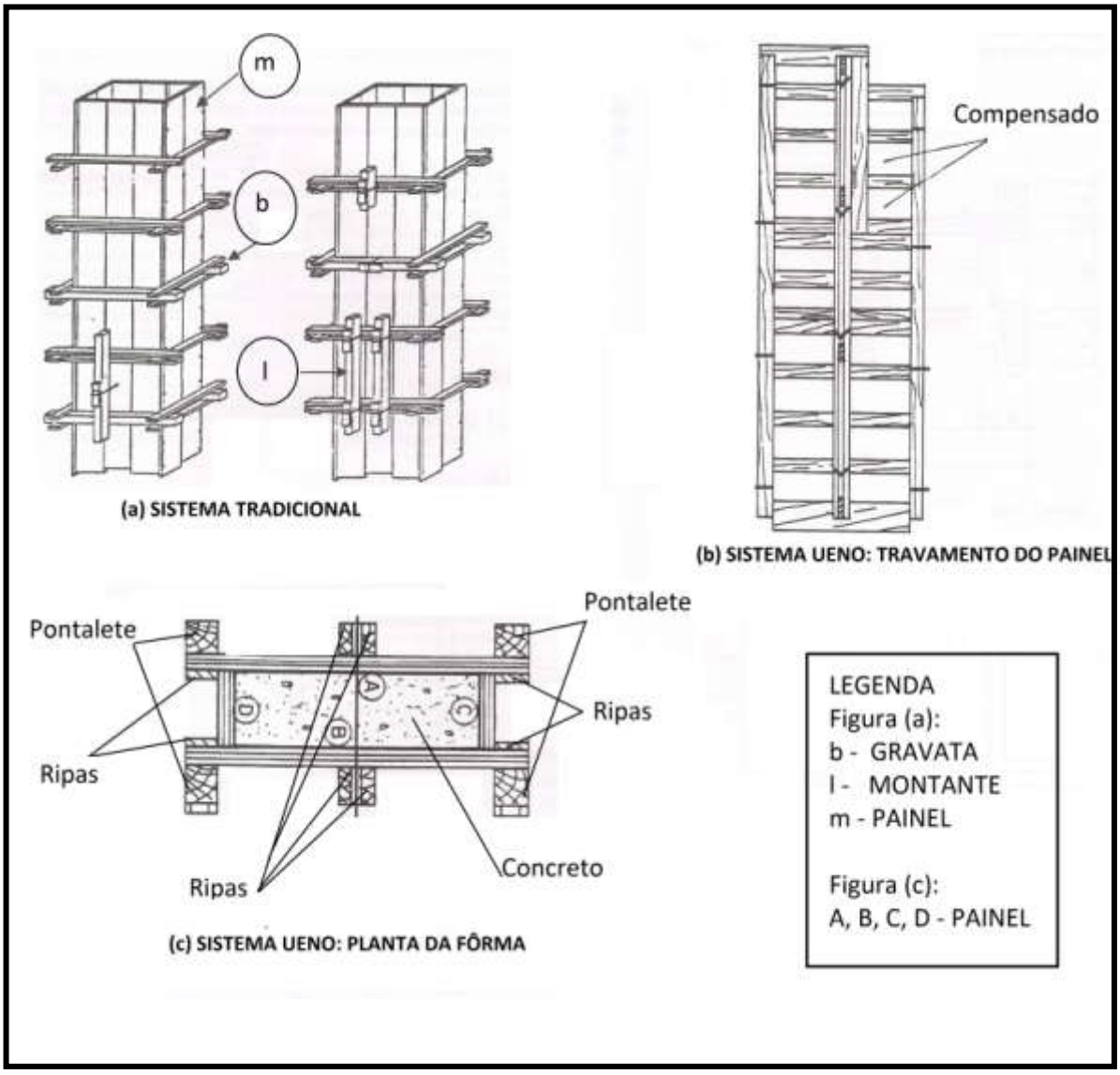

FIGURA 4.6: Sistema de fôrma tradicional e Ueno.
FONTE: Adaptado de Calil Junior *et al.*, 2000.

A metodologia adotada para simular o procedimento construtivo foi baseada em Yazigi (2004) e Calil Junior *et al*. (2000) para o sistema GETHAL e consiste no uso das seguintes peças: quatro painéis da fôrma compostas por chapas de madeira já preparadas e enrijecidas por caibros, a armadura do pilar composta por barras de aço longitudinal e estribos que já se encontra montada, o gastalho feito de madeira para servir como guia e travamento dos pés dos painéis que também já está pronto, as tábuas de madeira para os travar à fôrma e três barras para travamento dos painéis com as respectivas gravatas de aço e tensores para enrijecerem os painéis, evitando que se deformem devido à pressão exercida do concreto ao ser lançado dentro da fôrma.

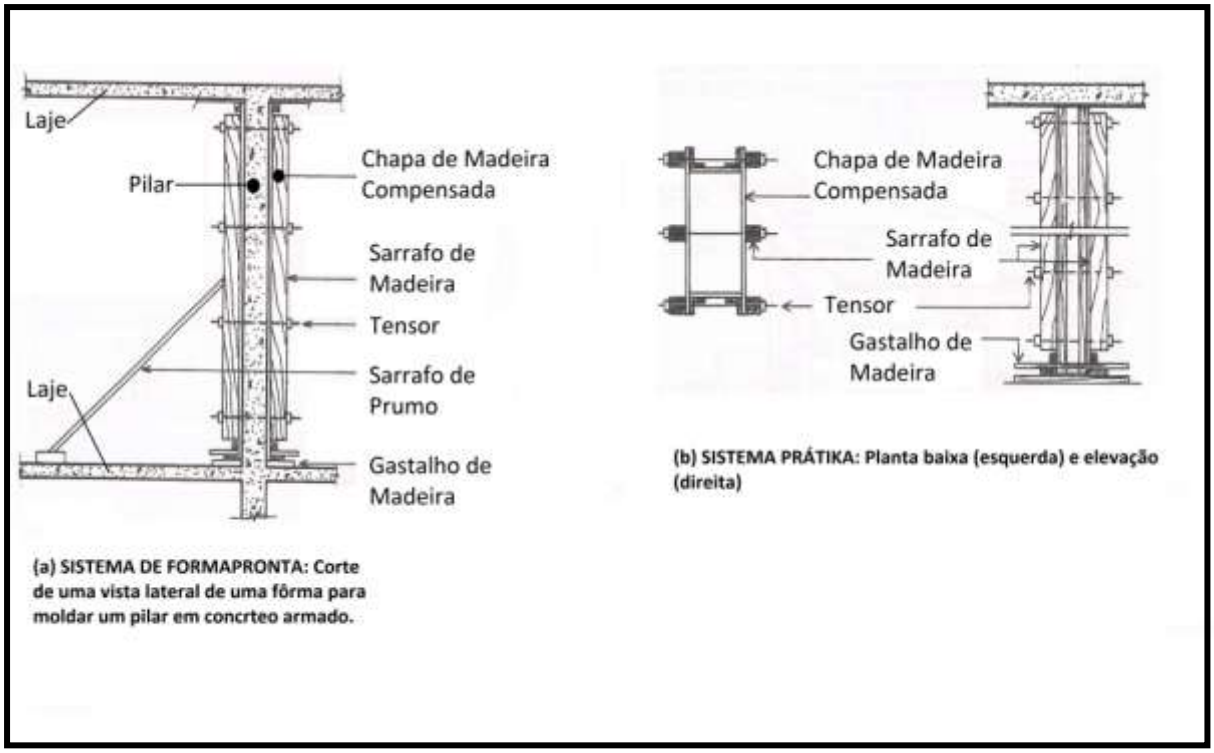

FIGURA 4.7: Sistema FORMAPRONTA e PRÁTIKA.
FONTE: Adaptado de Calil Junior *et al.*, 2000.

Então se procede com a montagem da fôrma: o usuário deve achar o gastalho e posicioná-lo conforme a indicação na cena; quando posicionado o gastalho procede-se com a identificação do painel a ser posicionado; uma vez posicionado o painel, suas travas devem ser posicionadas como indicado; procede-se com o encaixe dos outros painéis formando um esquadro de 90º em relação ao primeiro painel; antes de fechar a fôrma com o último painel a armadura deve ser colocada dentro da fôrma; estando a armadura dentro da fôrma o último painel deve ser encaixado e as travas devem ser posicionadas; as barras devem ser encaixadas nos painéis conforme a indicação; as gravatas de aço devem ser posicionadas encerrando o processo de montagem.

Nesta seção foram descritas a cena virtual do ambiente virtual protótipo e também foi apresentado um estudo sobre a atividade simulada. Na próxima seção está abordado como que a cena virtual foi modelada.

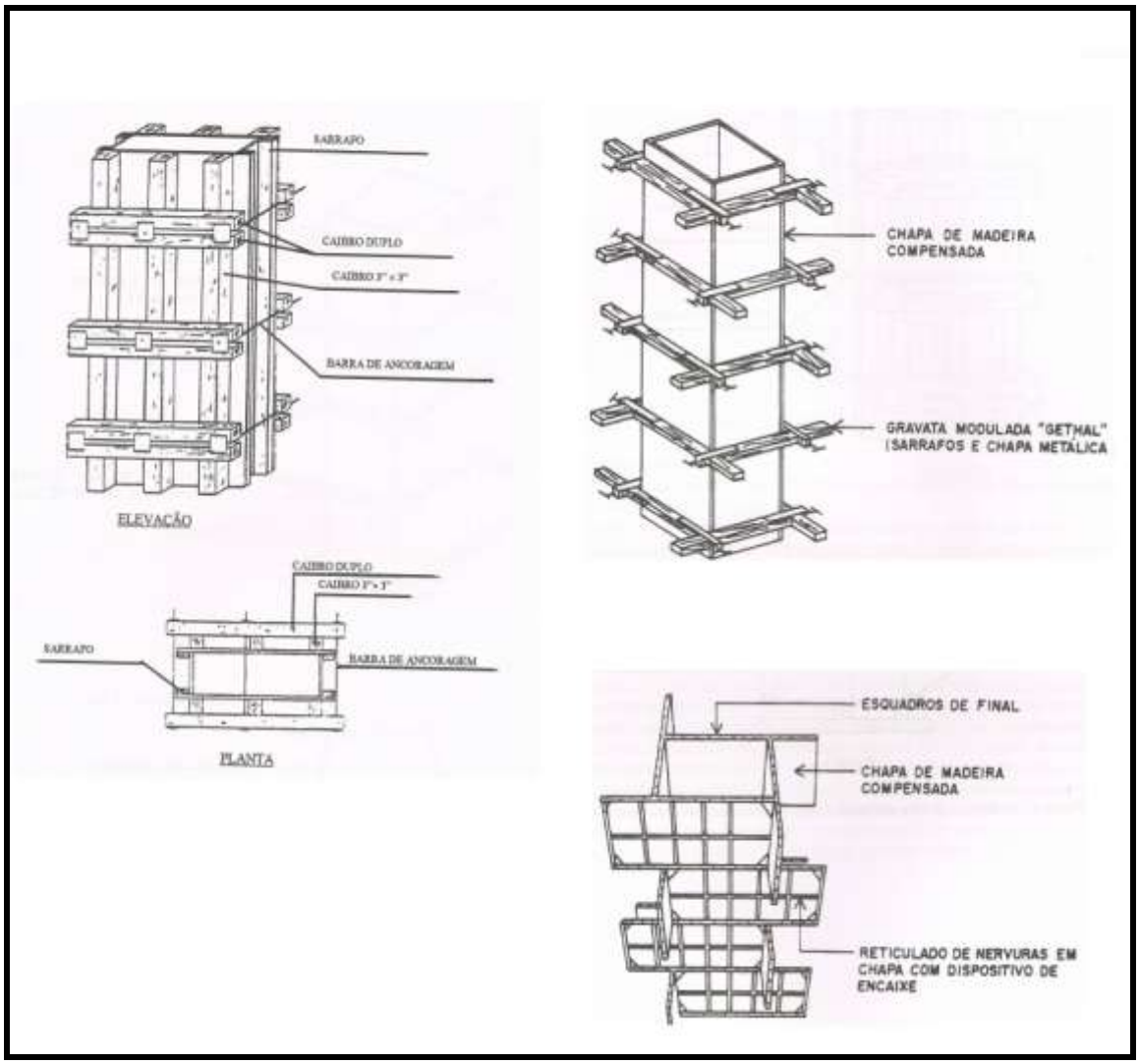

FIGURA 4.8: SISTEMA GETHAL.
FONTE: Adaptado de Calil Junior *et al.*, 2000.

4.2 MODELAGEM GEOMÉTRICA DA CENA E OBJETOS

A modelagem dos objetos gráficos da cena foi desenvolvida através do software 3DS Max (Autodesk), permitindo facilidade para criar os modelos geométricos dos componentes pertinentes à cena do ambiente virtual protótipo. Assim, as malhas geométricas dos modelos podem ser exportadas pelas ferramentas desse modelador para um formato conveniente para depois serem

importadas ao ambiente virtual protótipo do *toolkit* de visualização *Visualization Toolkit*, da Kitware.

O formato adotado para exportar as malhas geométricas foi o OBJ, comum aos aplicativos de modelagem gráfica 3D. Esse formato foi desenvolvido originalmente pela empresa Wavefront Technologies para ser usado em seu aplicativo *Advanced Visualizer* e aceito pela maioria dos aplicativos de computação gráfica pelo fato de ser um formato aberto.

Os dados da modelagem geométrica de um objeto podem ser escritos em um editor de texto e representam somente uma geometria tridimensional por vez, pela posição de cada vértice, a posição de cada coordenada parametrizada da textura em relação às coordenadas dos respectivos vértices, os vetores normais às faces e as faces poligonais definidas por uma lista de vértices. Os vértices são armazenados em um sentido anti-horário por padrão. O formato OBJ encontra-se na versão 3.0.

O formato possui os seguintes elementos em sua forma escrita em formato OBJ: ponto (p), linha (l), face (f), curva (Curv), curva bidimensional (Curv2) e superfície (surf). Ainda, é possível descrever curvas e superfícies de forma livre como, Bezier, B-spline, Taylor e Cardinal por meio dos seguintes atributos e sua forma escrita em formato OBJ: graus (deg), matriz base (bmat), passos (step) e tipo de curva (cstype).

Existe a possibilidade de criar agrupamentos de modelo geométricos (o) escritos em OBJ por meio das seguintes declarações: nome do grupo (g), suavidade do grupo (s), *merging* do grupo (mg). Os atributos para *render* e sua forma escrita em formato OBJ: interpolação *Bevel* (bevel), interpolação de cor (c_interp), interpolação *dissolve* (d_interp), nível de detalhe (lod), nome do material (usemtl), biblioteca de material (mtllib), sombras (shadow_obj), *ray tracing* (trace_obj), tipos de curva com *Bezier*, Cardinal, *Taylor*, *B-spline* (ctech) e tipos de superfícies (stech).

As coordenadas dos vértices são definidas por coordenadas no espaço ou por pontos no espaço paramétrico de uma curva ou superfície. O parâmetro "u" só é necessário para os pontos de uma curva. Os parâmetros "u" e "v" são necessários para os pontos de superfície e para os pontos de controle de curvas não racionais. Os parâmetros "u", "v" e "w" (peso) são necessários para os pontos de controle de uma curva racional. Dados de vértice: vértices geométricos (v), vértices de textura (vt), vértices normais (vn), vértices parâmetros de espaço (vp). A seguir são

apresentados exemplos de vértices escritos em formato OBJ precedidos por linhas de comentários que usam o símbolo “#”:

```
# Lista de vértices
 v 0,123 0,234 0,345 1,0
 v ...
# Coordenadas de texturas
vt 0,500 -1,352 [0,234]
vt ...

# Normais
  vn 0.707 0.000 0.707
  vn ...
```

As faces dos modelos geométricos são determinadas por listas dos índices de vértices, de textura e normais. As faces podem ser representadas na forma poligonal e definidas usando pelo menos três índices de vértices. O índice de um vértice é escrito por números inteiros e estão relacionados segundo uma lista previamente definida. A seguir são exibidos exemplos de faces escritas em formato OBJ precedidos por linhas de comentários que usam o símbolo “#”:

```
 #Face definida por vértices geométricos.
f v1 v2 v3 v4 ...
 # Face definida por vértices geométricos e vértices de texturas.
f v1/vt1 v2/vt2 v3/vt3 ...
# Face definida por vértices geométricos, texturas e normais.
f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 ...
```

A figura 4.9 ilustra um modelo geométrico de um painel de uma fôrma para moldar um pilar em concreto armado e sucede o seu arquivo no formato OBJ. No desenvolvimento dos modelos geométricos pertinentes à cena do ambiente virtual foram optados por faces quadrilaterais quando o modelo apresenta geometria mais retangular e faces compostas por triângulos quando os modelos geométricos apresentam formas curvas.

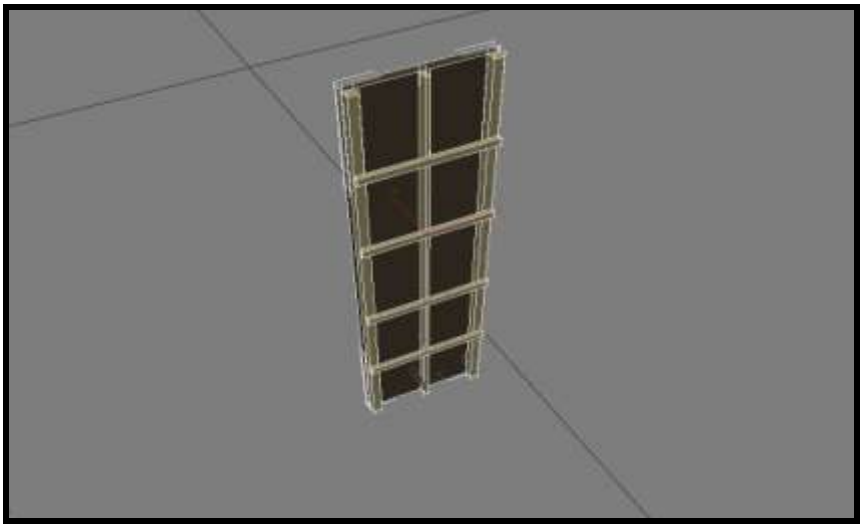

FIGURA 4.9: Modelo geométrico de um painel de fôrma para moldar um pilar .
FONTE: Autor(2012).

A próxima seção trata sobre a implementação da simulação dos objetos virtuais modelados conforme a abordagem descrita nesta seção.

4.3 *VISUALIZATION TOOLKIT*

O *Visualization Toolkit* (VTK) é um *toolkit* de visualização científica para computação gráfica 3D, processamento de imagens e visualização e o seu código é aberto, ou seja, gratuito (SCHROEDER, MARTIN e LORENSEN, 2004). Começou a ser desenvolvido em 1993 por William J. Schroeder, Bill Lorensen e Ken J. Martin e fazia parte do livro *The Visualization Toolkit an Object-Oriented Approach to 3D Graphics* e o seu objetivo consiste em ser de fácil usabilidade na programação voltada ao modelo orientado a objetos. Isto é possível, pois os algoritmos do VTK possuem as características de herança e polimorfismo.

O VTK consiste de um pacote de bibliotecas com classes escritas em linguagem de programação C++ e várias camadas de interfaces interpretadas como o Tcl/Tk, Java, e Python, trabalhando de forma portável, isto é, a instalação independe do sistema operacional, hardware ou compilador. O VTK pode ser utilizado em plataformas como Windows, Linux e Apple OSX, e também em

máquinas como Intel e Apple. Independe da linguagem gráfica, como XGL da *Sun Graphics*, *Star* da HP ou OpenGL da *Silicon Graphics* (que hoje é um padrão adotado pela indústria da computação gráfica). Apesar de não possuir *Graphic User Interface* (GUI, ou interface gráfica), pode ser integrado aos componentes de interfaces existentes em camadas como o Tk ou X/Motif (SCHROEDER, AVILA e HOFFMAN, 2000).

O VTK é um sistema grande e complexo, possuindo mais de 700 classes escritas em mais de 350.000 linhas de códigos e organizado para oferecer processamento de dados, gerenciamento do *pipeline* de visualização, leitura e gravador de dados, suporte para importar e exportar dados oriundos de diferentes aplicativos, processamento de imagens, processamento paralelo, classes patenteadas para o uso comercial e suporte para linguagens interpretadas como Tcl/Tk, Python e Java. Encontra-se num nível de abstração mais elevado comparado ao OpenGL, facilitando desenvolver aplicativos e visualizações. Ainda, suporta diversos tipos de interatividade, inclusive *trackball*, *joystick* e configuração para novos estilos de interatividade.

4.3.1 Arquitetura

O VTK é formado por um núcleo de classes compiladas e escritas em linguagem de programação C++, e uma camada interpretada que suporta as linguagens Tcl/Tk, Java e Python. Essa arquitetura permite eficiência tanto em processamento como em memória ao núcleo em C++ onde estão implementadas as estruturas de dados, algoritmos e funções de sistema. Proporciona flexibilidade e extensibilidade à camada interpretada permitindo rapidez na criação de aplicações utilizando ferramentas de GUI tais como Tcl/Tk, Python/Tk, Java AWT, QT e Motif. A figura 4.10 mostra o esquema da arquitetura do VTK (SCHROEDER, MARTIN e LORENSEN, 2004).

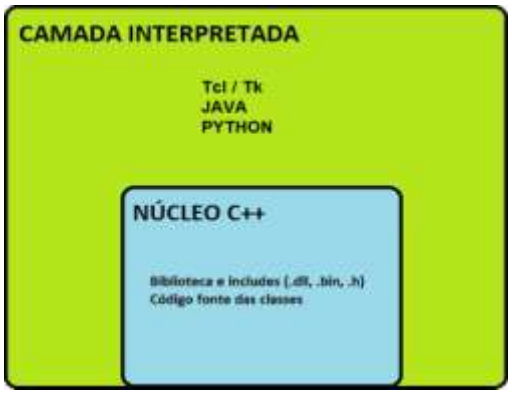

FIGURA 4.10: Arquitetura do VTK.
FONTE: Adaptado de Kitware, 2004.

Ao adotar o modelo de fluxo de dados, o *pipeline* do VTK é composto basicamente por uma fonte de dados a ser visualizado, um subsistema (ou módulo) chamado modelo de visualização para transformar as informações dos dados em objetos gráficos e outro subsistema (ou módulo) chamado modelo gráfico para a criação da cena conforme mostra a figura 4.11.

FIGURA 4.11: Modelo de fluxo de dados usado pelo VTK.
FONTE: Adaptado de Kitware, 2004.

4.3.1.1 Modelo de visualização

O modelo de visualização é responsável pela representação geométrica dos dados ao transformá-los em primitivas gráficas, por exemplo, visualizar um conjunto de dados ou pontos (*dataset*) por meio de uma malha poligonal via triangulação de Delaunay. Esse modelo decompõe a transformação em diversas operações sobre os dados e estas formam conexões constituindo um fluxo de dados. A figura 4.12(a) apresenta o pipeline do modelo de visualização, onde é possível destacar dois tipos distintos de objetos: o objeto dado cuja denominação é *vtkDataObject* e objeto

processo, denominado por v*tkProcessObject*. Quando os objetos dados e os objetos processo estão conectados formam o *pipeline* do modelo de visualização conforme figura 4.12(b) (SCHROEDER, MARTIN e LORENSEN, 2004).

O objeto dado ao receber *input* é responsável pela estrutura topológica, pela estrutura geométrica e pelos atributos relacionados a um *dataset*. A topologia determina a configuração ou forma do objeto dado representado, por exemplo, um cubo ou uma malha composta por um conjunto de células, conforme a figura 4.13. A estrutura geométrica relaciona o conjunto de pontos ou vértices pertencentes às células por meio de coordenadas e instâncias. Os atributos são informações que podem ser associadas aos vértices e às células, como exemplo a densidade em cada célula (SCHROEDER, MARTIN e LORENSEN, 2004).

O objeto processo é formado por algoritmos chamados de filtros e operam nos objetos dados gerando novos objetos dados através de três tipos: *source* (fonte), *filter* (filtro) e *mapper* (mapeador) como ilustra a figura 4.12(c). O objeto processo *source* inicia o *pipeline* gerando pelo menos uma saída (*output*) e pode ser, por exemplo, um leitor de um tipo específico de dado. Os dados se tornam um *input* quando sofrem operações dos objetos processo *filter* que irão gerar novas saídas, tais como a extração do contorno de um dado. O *mapper* é um objeto processo que recebe todos os objetos dados (*input*) para serem transformados em gráficos a serem renderizados. Ele é a interface entre o modelo de visualização e o modelo gráfico.

4.3.1.2 Modelo gráfico

O Modelo Gráfico do VTK é responsável por transformar os dados oriundos do modelo de visualização em imagens para serem representadas no ecrã, criando uma camada abstrata sobre a linguagem gráfica como o OpenGL e assim, garantindo a portabilidade entre plataformas (SCHROEDER, AVILA e HOFFMAN, 2000).

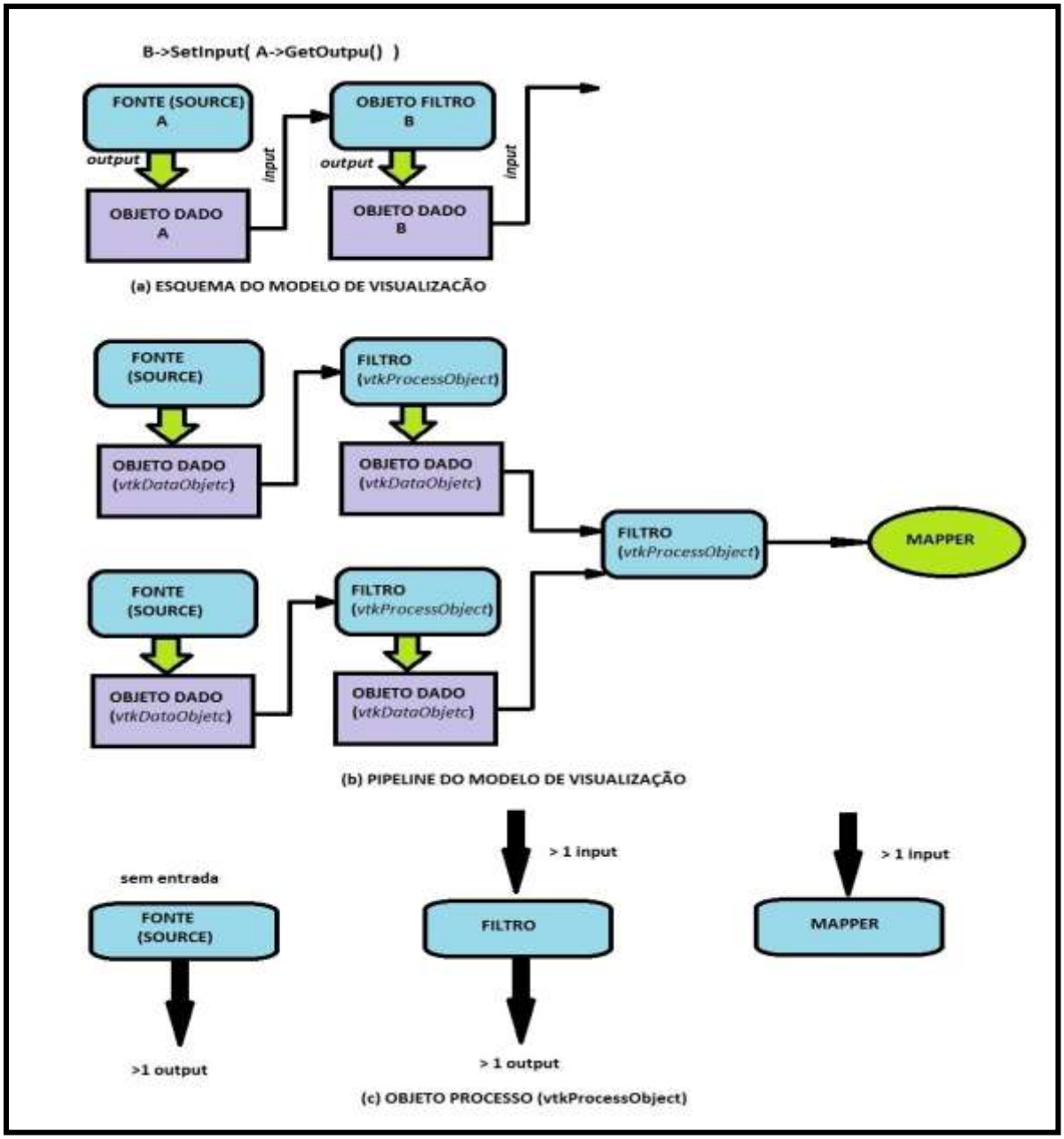

FIGURA 4.12: Pipeline do modelo de visualização do VTK>
FONTE: Adaptado de Kitware, 2004.

No processo de renderização de uma cena, o VTK utiliza os seguintes objetos (apesar da possibilidade da existência de mais objetos durante o processo):

a) *vtkRenderWindow*: É responsável pelo controle de uma janela para visualização dos dados no dispositivo de vídeo conforme a plataforma, ou seja, suas instâncias são independentes do dispositivo. É uma classe que engloba os objetos que estão contidos nos objetos render;

b) *vtkRenderer*: Coordena o processo de renderização, envolvendo luz, câmeras e atores. Há a necessidade de ser definido pelo menos um ator na

cena. Gerencia a interface entre hardware gráfico e a janela do sistema operacional pela associação entre ele e objetos da classe *vtkRenderWindow*;

c) *vtkLight*: São as fontes de luz para iluminação da cena cujas instâncias podem controlar a intensidade, posição, direção e tipo de luz (spot ou direcional);

d) *vtkCamera*: Esta classe gerencia a câmera virtual da cena que projeta a cena 3D numa imagem bidimensional durante o processo de renderização. Controla parâmetros como projeção ortogonal ou perspectiva e a definição dos planos de corte;

e) *vtkActor*, *vtkActor2D*, *vtkVolume*: São classes que herdam a classe *vtkProps* que é responsável por informar posição, orientação e visibilidade durante o processo de render dos objetos na cena. Estas classes dão suporte à manipulação de propriedades como cor, tipo de sombra, transparência, definição geométrica, e sua posição e orientação no sistema de coordenadas global pelas instâncias do *vtkProperty* e *vtkTransform* e suas subclasses;

f) *vtkProperty*: Define as propriedades de aparência de um ator, tais como cor e transparência;

g) *vtkTransform*: Corresponde a uma matriz 4x4 que promove transformações geométricas como translação, rotação, escalamento;

h) *vtkMapper*: Define a representação geométrica para um ator. É a interface entre o *pipeline* do modelo de visualização e o modelo gráfico;

i) *vtkRenderWindowInteractor*: gerencia os diversos estilos de interatividade uma vez que a cena está renderizada por meio do padrão comando/observador.

4.3.1.3 *Pipeline* de execução

O *pipeline* de execução do VTK adota uma abordagem chamada *lazy evaluation*, ou seja, são executados os processos de visualização somente quando algum dado relacionado é requisitado. O método *Render()* inicia a requisição dos dados que serão enviados conforme a direção do fluxo. Quando ocorre uma nova requisição, as partes do *pipeline* que estão desatualizadas são novamente executadas pelos filtros. Eles atualizam os dados para enviá-los ao final do processo do fluxo podendo ser renderizados por um ator da cena. A figura 4.13 mostra a abstração do *pipeline* de execução e a figura 4.14 mostra um exemplo do *pipeline* por meio de um código escrito em C++ (SCHROEDER, MARTIN e LORENSEN, 2004).

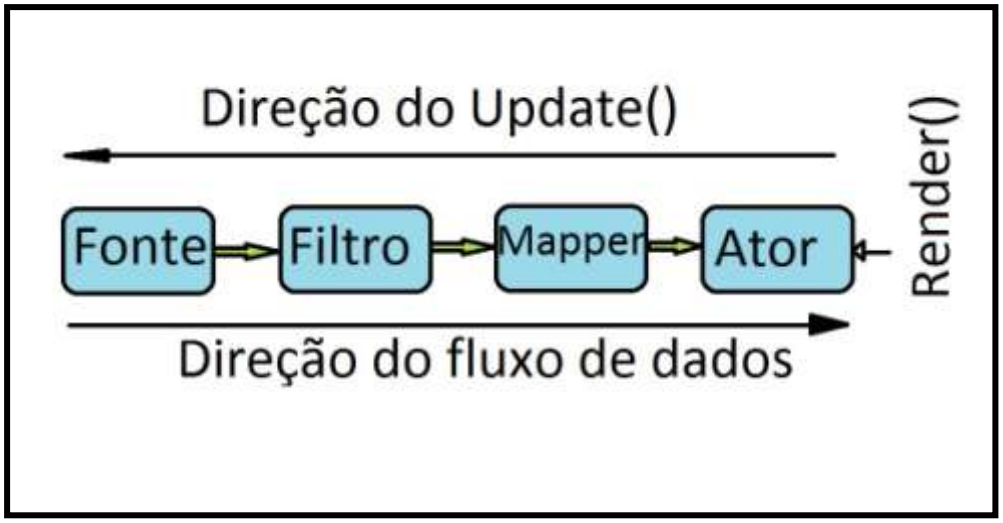

FIGURA 4.13: Pipeline de execução.
FONTE: Adaptado de Kitware, 2004.

4.3.2 Instalação do VTK versão 5.4

O VTK pode ser compilado em diversas combinações entre sistemas operacionais, configurações de *hardware* e compiladores. Pela diversidade das possíveis combinações faz a distribuição de binários (cadeia de bits associadas a

um símbolo conforme o sistema numérico binário, ou seja, 0 e 1) não ser viável. Assim, para se proceder com a instalação do VTK é utilizado seu código fonte (*source code*) que será compilado e conectado (*link*) para gerar as bibliotecas das classes e os executáveis. Como o AV protótipo não foi desenvolvido em Tcl ou Python, não existe a possibilidade de desenvolver aplicativos em VTK usando os binários pré-compilados. Para isto é necessário um utilitário que realize o processo de construção do VTK chamado CMake (Kitware, 2004).

O CMake é uma ferramenta de código aberto (*open source*) que serve para configurar e construir processos como códigos nativos para um determinado compilador e sistema operacional. Através do CMake é possível construir as bibliotecas da versão do VTK em quaisquer computadores utilizando seu *source tree* e trabalhar com ferramentas como editores, *debbugers*, compiladores entre outros (Kitware, 2004).

Para iniciar o processo de construção das bibliotecas da versão do VTK via CMake é necessário saber as seguintes informações: qual compilador será utilizado para desenvolver aplicativos com o VTK, o diretório onde está o *source code* da versão do VTK e o diretório para locar as bibliotecas e binários que serão criados durante o processo de instalação da versão VTK. Uma vez fornecidos os três dados necessários via interface gráfica (GUI) do CMake são lidos conforme as variáveis e as entradas de cache selecionadas pelo usuário. Ao terminar o passo de configuração procede-se com o passo de geração para produzir os *workspaces* e *makefiles*.

Um roteiro de instalação está descrito no apêndice 2.

4.3.3 Manipulação de arquivos via VTK

O VTK possui classes para gerenciar a leitura e gravação dos seus próprios modelos de arquivo, que são os formatos *legacy* baseado em XML.

O formato *legacy* é mais simples e consiste numa sequência de informações que podem ser escritas tanto manualmente como computacionalmente.

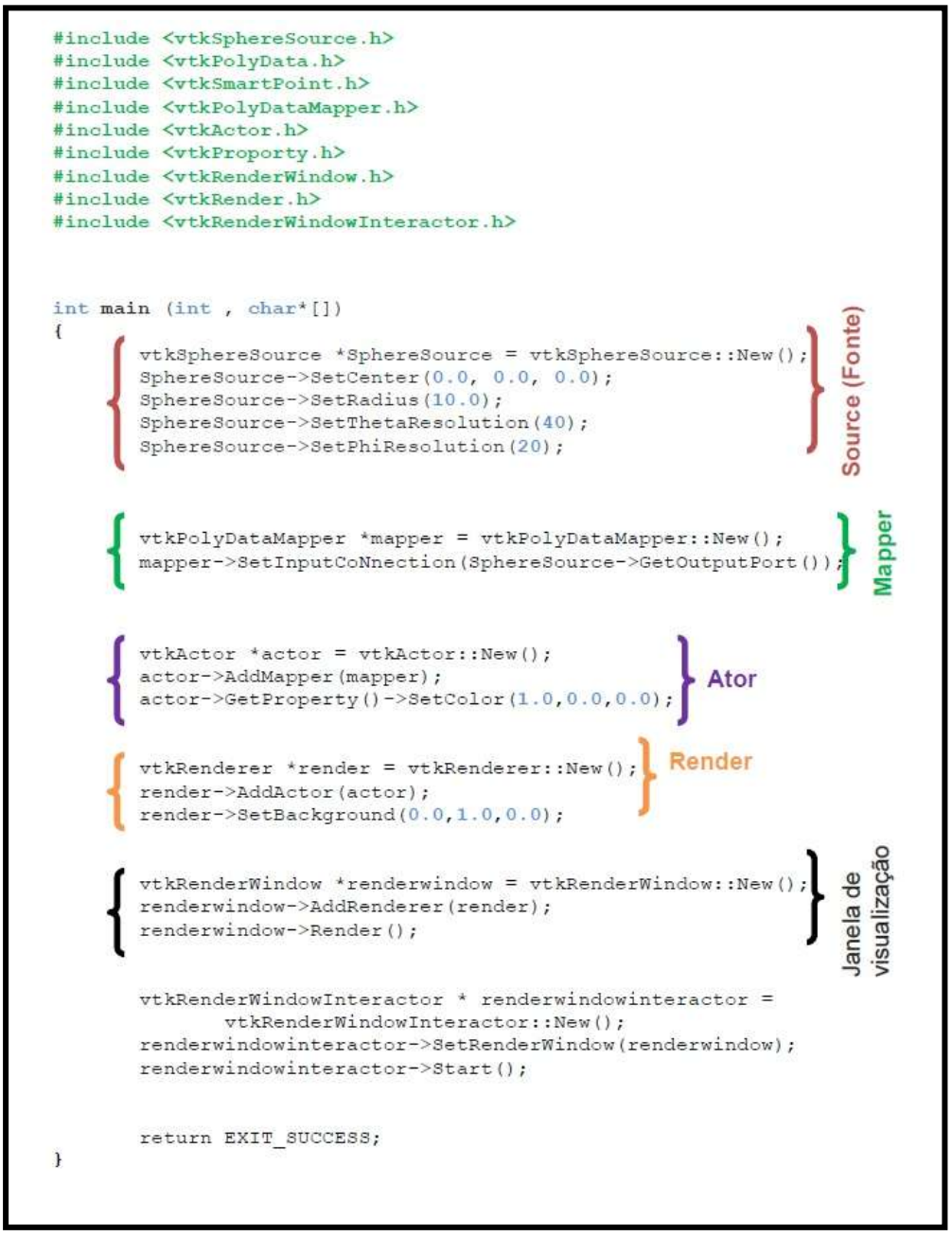

FIGURA 4.14: Pipeline do VTK ilustrado por um código escrito em C++.
FONTE: Autor(2012).

A figura 4.15 ilustra um exemplo do formato, que é composto por cinco partes básicas:

1) Identificador do arquivo e da versão do VTK. Sua sintaxe consiste na primeira linha e deve ser escrita exatamente desta forma com exceção do "x.x" que será substituído pela versão do VTK: #vtk DataFile Version x.x;

2) O cabeçalho que deve ser escrito na segunda linha e conter no máximo 256 caracteres e sucedido por "\n". Ele é uma *string* que pode conter informações pertinentes aos dados;

3) O formato do arquivo: binário ou ASCII. Deve ser escrito na terceira linha e sintaxe ASCII ou *BINARY*. No formato binário podem ocorre problemas de portabilidade devido às diferentes representações para alguns tipos como inteiros, por exemplo, que dependem do formato uso em sistemas operacionais;

4) Conjunto de dados estruturados. Aqui é descrito a geometria e a topologia dos dados. Deve ser começado na quarta linha pela palavra *DATASET* sucedida pela palavra que define o tipo da estrutura. O VTK suporta cinco tipos de estruturas de dados: pontos estruturados, malhas (*grid)* estruturadas, retilíneas, não-estruturadas e dados poligonais. Dados que utilizam funções implícitas (estruturados) variam conforme acréscimo em x, depois em y e por último em z;

5) A parte final é o atributo, que segue em uma nova linha do arquivo logo que terminam as informações do *DATASET*. Começa com a palavra *POINT_DATA* ou *CELL_DATA* e sucedida por um número inteiro que informa a quantidade de pontos ou células, respectivamente. As demais palavras definem os valores dos atributos como escalares, vetoriais, tensoriais entre outros.

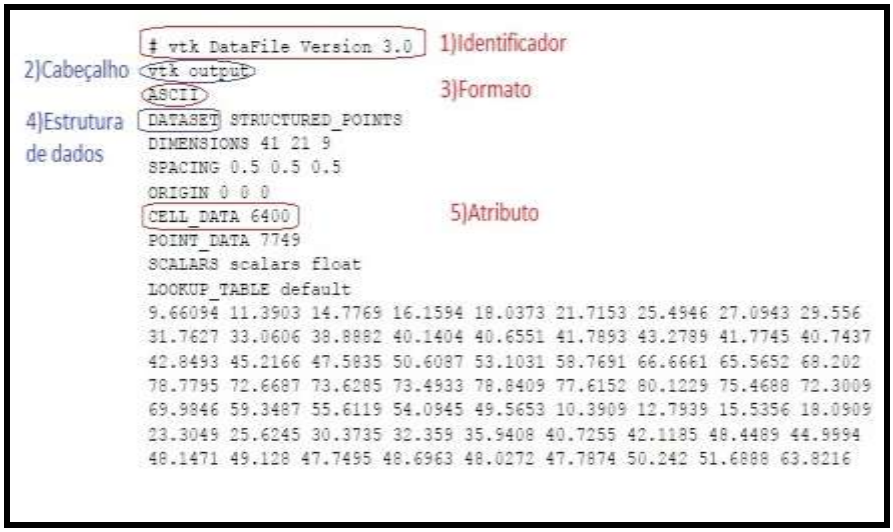

FIGURA 4.15: Exemplo de um arquivo do VTK no formato *legacy.*
FONTE: Adaptado de Buriol, 2005.

Outro formato usado no VTK é baseado em XML. É mais complexo, porém permite mais flexibilidade além de suportar acesso aleatório, entrada e saída de sistemas paralelos e permitir maior compressão.

Mesmo possuindo seus formatos nativos, o VTK tem o poder de importar e exportar formatos de arquivos originados de outros aplicativos de computação gráfica através das classes de importação e exportação. Os seguintes formatos são suportados pelo VTK: BYU, PDB, PLY, STL, XML, BMP, DICOM, GES, JPEG, PNM, PNG, SLC, TIFF, PLOT3D, POP, AVS, CGM e PNG assim como formatos 3D *Studio*, VRML, *PostScript*, *Inventor*, *Wavefront* e *GeomView*.

4.3.4 Classes utilizadas do VTK

O ambiente virtual protótipo utiliza as classes do VTK e a linguagem de programação C++ para importar a geometria e as texturas pelos formatos OBJ e JPEG respectivamente. Apesar de o VTK suportar o formato VRML[37] por meio da classe *vtkVRMLImporter*, não há suporte para todos os *nodes* que o formato oferece, especialmente o *node ImageTexture*, responsável por dar mais realismo aos objetos por mapear imagens aos objetos geométricos. O VTK oferece ainda a classe *vtk3DSImporter* que herda os métodos da classe *vtkImporter* para gerenciar importação de arquivos em formatos típicos do 3DS Max (geometria e material). Mas como é uma instância da classe *vtkRender*, ele direciona o *dataset* direto para *renderer* o que dificulta o uso das classes para gerenciar e sincronizar a interatividade, servindo para o uso de objetos gráficos da cena que não irão interagir com o usuário.

Os modelos geométricos da cena foram exportados no formato OBJ e podem ser importados pela classe *vtkOBJReader*, porém não foi achada uma classe ou método para importar material usando esse formato, ou seja, o formato *material template library*, também desenvolvido pela Wavefront Technologies. Esta classe é um *source object* cuja saída é um *polygonal data* e quando mapeado através do

[37] *Virtual Reality Modeling Language* (VRML) é um formato típico usado na Realidade Virtual.

vtkPolyDataMapper transforma os dados em primitivas gráficas. Cabe à classe *vtkActor* representar as primitivas gráficas na cena renderizada.

As texturas são representadas por imagens no formato JPEG e importadas pela classe *vtkJPEGReader* que é um *source object*. O mapeamento das coordenadas da imagem é feito pela classe *vtkTexture*. As propriedades da textura podem ser ajustadas conforme os métodos da classe tais como *SetInterpolateOn*( ) para interpolação da textura durante o render, adaptação da imagem quando ultrapassa o intervalo paramétrico [0,1] por meio do *EdgeClampOn*( ) e forçar a qualidade da textura pelo método *SetQualityTo32Bit*( ). Atualmente o VTK suporta apenas texturas 2D. As instâncias do *vtkTexture* estão associadas com os atores pelo método *SetTexture*( ). Isto significa que a geometria recebe o mapeamento de texturas já no modelo gráfico do *pipeline* do VTK.

A classe *vtkCamera* possui métodos para manipular a câmera virtual. Os métodos *SetPosition*( ), *Azimuth*( ), *Roll*( ) permitem o controle da posição e orientação da projeção da câmera. Uma vez que a câmera virtual está especificada ela pode ser usada pelo *vtkRenderer* por meio do método *SetActiveCamera* ( ) que recebe como argumento um *vtkCamera*. A navegação pela cena ocorre pela translação, rotação e controle do zoom da câmera virtual, que calcula as superfícies visíveis e a oclusão dos objetos.

As luzes são ativadas na cena pela classe *vtkLight* que simula uma luz virtual durante o rendering. A classe possui métodos para posicionar e alterar cor e brilho. Pode simular tipos como *spot light* (cone de luz ou fontes de luz pontuais) pelo método *PositionalOn*( ) e *directional light* (luz cuja posição está no infinito e os raios são paralelos, simulando fontes de luz como o sol) sendo um tipo *default*. Ainda possui métodos para posicionar a fonte de luz conforme a posição da câmera usando os métodos *LightTypeIsHeadlight*( ) ou *LightTypeIsCameralight*( ) (que se movimenta conforme a câmera mas não precisa estar coincidente com ela).

Objetos gráficos virtuais que necessitam de interatividade para serem posicionados na cena durante o treinamento usam as classes do *3DWidget*. Estas classes são subclasses de *vtkInteractor*, uma classe abstrata responsável por observar os eventos invocados na janela da classe *vtkRenderWindow*. Os *widgets* possuem representação própria na cena renderizada e oferecem tipos de interatividade ao respectivo ator da cena, sincronizados pelo método *SetProp3D*( ). Cada *widget* deve usar o método *SetInteractor*( ) para especificar o *renderer* ao qual

se deseja adicionar o *widget*. A lista dos principais *widgets* a seguir explica suas aplicabilidades:

a) *vtkScalarBarWidget*: classe para gerenciar o *vtkScalarBar* onde se define a posição e a orientação;

b) *vtkPointWidget*: adiciona na cena um ponto no espaço da cena e produz uma saída poligonal;

c) *vtkLineWidget*: adiciona na cena uma linha reta e produz uma saída poligonal;

d) *vtkPlaneWidget*: adiciona na cena um plano orientado e finito. Produz uma saída poligonal ou implícita;

e) *vtkImplicitPlaneWidget*: adiciona na cena um plano orientado e não limitado. Produz uma saída poligonal ou implícita;

f) *vtkBoxWidget*: cria um cubo orientado e produz uma função implícita e uma matriz de transformação geométrica. É usado para transformar *vtkProp3D*;

g) *vtkImagePlaneWidget*: manipula três planos ortogonais inseridos num conjunto volumétrico de dados e serve para amostragem de dados;

h) *vtkSphereWidget*: manipula uma esfera com resolução variável e usada para controle de câmeras e luz;

i) *vtkSplineWidget*: manipula uma curva de interpolação e produz uma representação gráfica.

No ambiente protótipo foi usada uma classe que herda *vtkBoxWidget* chamada *vtkBoxWidget1* como forma de manipular as peças para montar a fôrma do pilar de concreto. A figura 4.16 apresenta as características do *boxwidget.* Suas caraterísticas são: não serem visíveis na cena para não atrapalhar a identificação das peças pelos métodos *HexActor→GetProperty()→SetOpacity, HexFace→VisibilityOff, HexOutline→VisibilityOff*; não são visíveis e não estão ativados os *handles* e os *outlinecursores*; a classe já inicializa a posição do *widget* na cena. O quadro 1 apresenta a classe escrita em C++.

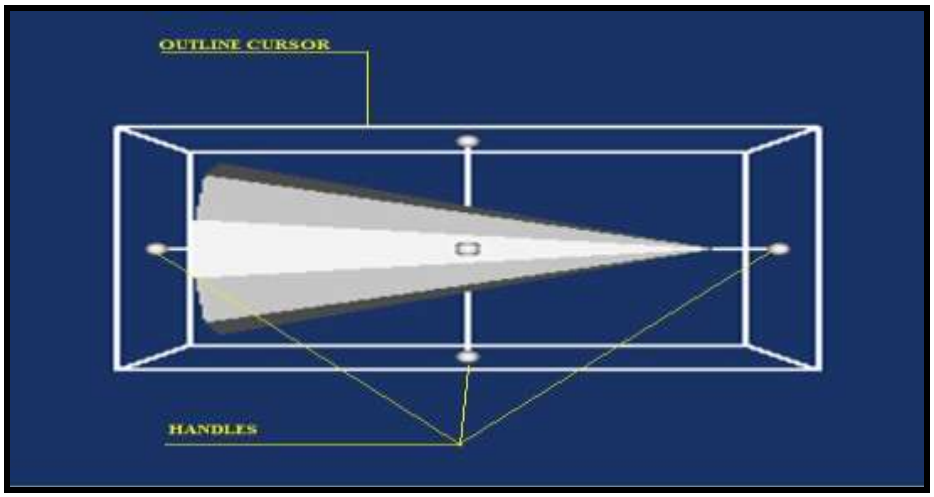

FIGURA 4.16: Representação gráfica de um *BoxWidget*.
FONTE:Autor(2012).

```
class vtkBoxWidget1:public vtkBoxWidget
{
public:
static vtkBoxWidget1* New();
virtual void corbox()
{
this->HexActor->GetProperty()->SetRepresentationToSurface();
this->HexActor->GetProperty()->SetOpacity(0.001);
this->HexFace->VisibilityOff();
this->HexOutline-> VisibilityOff();
this->PlaceWidget();
this->OutlineCursorWiresOff();
this->HandlesOff();
          }
};
```

Quadro 1: Classe *vtkBoxWidget1.*

A movimentação das peças (translação e rotação) acontece pela interatividade provida por *vtkInteractor* e *vtkCommand*, além de *3Dwidgets* . Um *vtkObject* (um ator, por exemplo) pode observar um evento qualquer por meio de uma das suas instâncias. Quando observa um evento ao qual está relacionado invoca um comando associado. Isto faz o usuário estar envolvido dentro do loop do *pipeline* do VTK quando, por exemplo, executa um click com o mouse na janela do sistema que o VTK está usando. Vale lembrar que o *vtkCommand* possui um ponteiro que se encontra nulo na maior parte do tempo do processo de visualização e quando ocorre um evento ela passa o *vtkObject* que está invocando o evento e

não o evento em si. O mesmo acontece quando se criam novas classes que herdam as características de *vtkCommand* para definir outros modos de interatividade.

O *vtkInteractor,* ao contrário dos *3DWidgets*, não possui representação gráfica. Esta classe é a base para o suporte de eventos de controle de atores e câmeras virtuais numa cena, possuindo a maioria das rotinas de controle, tais como interatividade com *joystick, trackball.* Possui os seguintes estilos: *vtkInteractorStyle Flyght* (rotinas para movimento de vôo), *vtkInteractorStyleTerrain* (manipulação da câmera com view up natural), *vtkInteractorStyle Image* (manipulação de câmera especializada em imagens), *vtkInteractorStyleJoystickActor* e *vtkInteractorStyleTrackBallActor* (manipulação de atores independentes uns dos outros), *vtkInteractorStyle JoystickCamera* e *vtkInteractorStyleTrackBallCamera* (manipulação da câmera), *vtkRubberBandZoom* (aplica-se o zoom desenhando um retângulo sobro o objeto), *vtkInteractorStyleUser* (programar formas de interatividade com a câmera).

Neste trabalho optou-se pelo uso de *vtkInteractorStyleTrackBallCamera* pois é uma classe que provê interatividade para se visualizar uma determinada área da cena de vários pontos de vista, auxiliando o usuário na visualização durante a manipulação de uma peça da fôrma.

A classe criada para interatividade no ambiente protótipo chama-se *vtkMyCallback* e herda a classe *vtkCommand* . Esta classe possui ponteiros para os atores e *widgets* que representam as peças da fôrma; para os atores que indicam onde a peça deve estar posicionada na cena, chamados de ator indicador; e para a legenda, atualizando as informações. Estes objetos apontam para os respectivos atores, *widgets*, e legenda que estão sendo renderizados, ou seja, funcionam para associar um evento invocado por um ator e associar um comando conforme a sequência que aparece na legenda. Quando uma peça está sendo selecionada ocorre uma comparação entre o *widget* da cena com o ponteiro da sequência de montagem. Se a comparação resulta em verdadeiro a interatividade é iniciada e a movimentação da peça acontece. Caso retorne falso, não existe interatividade com a peça. As coordenadas são salvas por um *array* conforme a matriz linear de transformação do *widget* selecionado. Ao posicionar a peça próximo do ator indicador, o ajuste da posição final ocorre automaticamente através de uma transformação. A interatividade é encerrada e um novo indicador na cena é liberado e também é liberada a interatividade da próxima peça da sequência de montagem.

No apêndice 3 pode ser observada esta classe *vtkMyCallback* na versão completa dos códigos escritos em C++ para o desenvolvimento do ambiente protótipo.

A legenda, por meio do *vtkLegendBoxActor* fornece as informações sobre quais peças devem ser selecionadas e posicionadas conforme o ator indicador. As informações são atualizadas conforme o usuário vai posicionando as peças. Isto ocorre porque a classe *vtkMyCallback* possui um ponteiro para a legenda. A figura 4.17 ilustra a legenda.

FIGURA 4.17: Zoom da legenda que informar a sequência e a respectiva peça.
FONTE: Autor(2012).

4.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi descrito o desenvolvimento do protótipo do ambiente virtual educacional e as ferramentas computacionais que serviram para sua implementação. A implementação do protótipo pode ser descrita em quatro núcleos conforme as ferramentas computacionais e sua finalidade.

O primeiro núcleo consiste no uso de uma ferramenta de modelagem geométrica para a criação da cena virtual, no caso o canteiro de obras; os objetos

virtuais que compõem a cena. Tal ferramenta deve permitir a exportação dos modelos geométricos em um formato compatível com as ferramentas computacionais de visualização. Assim, como já descrito anteriormente, a ferramenta escolhida para este núcleo foi o Autodesk 3DS Max.

O segundo núcleo consiste na escolha de um pacote computacional de visualização para auxiliar na programação da interface e interatividade do protótipo e ainda, ser capaz de importar os modelos geométricos desenvolvidos no primeiro núcleo. Assim, foi adotado o *Visualization ToolKit* (VTK) da Kitware.

O terceiro núcleo consiste em uma linguagem de programação e um compilador (IDE) para então programar, pelo pacote computacional de visualização do segundo núcleo, a visualização da cena virtual e a sequência da simulação conforme a atividade previamente escolhida. Para este núcleo foram adotados a linguagem de programação C++ e o compilador Microsoft Visual Studio C++.

O quarto núcleo consiste no sistema operacional que será adotado para rodar o ambiente virtual protótipo, que irá usar as janelas, o gerenciamento de dispositivos gráficos e de interatividade conforme a programação do terceiro núcleo junto com o segundo núcleo. O sistema operacional escolhido foi o Microsoft Windows 7 64 bits.

A figura 4.18 demonstra graficamente os quatro núcleos de implementação do ambiente virtual protótipo.

FIGURA 4.18: Núcleos do sistema implementado para o ambiente virtual protótipo.
FONTE: O Autor (2012).

5 TESTES E RESULTADOS

Este capítulo descreve uma metodologia adotada para realização dos testes do ambiente virtual protótipo, esclarecendo quem são os participantes dos testes, os resultados obtidos através de uma avaliação dos participantes e uma análise dos resultados das avaliações.

5.1 ESTRATÉGIA DE APLICAÇÃO E VALIDAÇÃO DO AMBIENTE VIRTUAL PROTÓTIPO

5.1.1 Elaboração da metodologia

Ao aplicar o ambiente virtual protótipo como instrumento e ferramenta de ensino e aprendizagem foi necessário realizar testes com um grupo de alunos do curso de graduação em Engenharia Civil da Universidade Federal do Paraná, buscando reproduzir uma aula em laboratório sobre Construção Civil, em específico, o tema de montagem de fôrmas para moldar pilares em concreto armado. O grupo ficou limitado a sete alunos. Visando uma forma mais próxima da didática adotada pelo curso de Engenharia Civil, um professor da disciplina de Construção Civil foi entrevistado para avaliar a usabilidade do ambiente virtual protótipo.

Como o ambiente virtual protótipo foi desenvolvido para interagir com apenas um usuário, a estratégia adotada para avaliar o desempenho da aprendizagem se deu por meio de três aulas-testes realizadas no laboratório de visualização científica do Programa de Pós-Graduação em Métodos Numéricos em Engenharia da mesma universidade. O grupo de sete alunos foi locado da seguinte forma: na primeira aula-teste compareceram três alunos para sessão e nas outras aulas-testes realizadas compareceram dois alunos por aula. Não houve repetição das aulas para os participantes.

Cada aula-teste teve a duração de uma hora e foram realizadas em diferentes dias. Elas contemplavam a explicação sobre o porquê que estavam participando da aula-teste, a instrução (treinamento) da atividade de montar uma fôrma pela simulação e um debate sobre o tema, conforme a necessidade e curiosidade dos participantes. Ao final de cada aula-teste os participantes da simulação eram convidados a responder um questionário para avaliar o ambiente virtual protótipo.

Para que as sessões de simulação das aulas-testes pudessem ser realizadas com pelo menos dois participantes, foi disposto o notebook usado para a implementação do ambiente virtual protótipo e um monitor de 46 polegadas como demonstra a figura 5.1. Demais alunos foram convidados a participar das aulas-testes e mesmo que não participassem da simulação eles poderiam participar do debate.

FIGURA 5.1 – AULA-TESTE NO LABORATÓRIO DE VISUALIZAÇÃO CIENTÍFICA DA UFPR.
FONTE: O Autor (2012).

O questionário desenvolvido para avaliar o ambiente virtual protótipo foi baseado e adaptado conforme o estudo realizado pela pesquisadora Rosália

Gusmão de Lima sobre validação de um ambiente virtual de ensino a distância sobre estruturas de aço (LIMA, 2002). Na opinião do autor deste trabalho, alguns dos aspectos abordados no questionário desenvolvido pela pesquisadora poderiam ser adaptados para a avaliação do ambiente virtual protótipo proposto. Assim, o questionário usado neste trabalho foi adaptado para abordar os seguintes aspectos: aceitação do ambiente virtual, requisitos de desempenho e qualidade visual. Uma versão do questionário está disponível no apêndice 1.

A aceitação do ambiente virtual serve para avaliar se o usuário compreende que está usando uma didática de aprendizagem pela simulação, considerando a facilidade do uso dos dispositivos que permitem a interatividade com a simulação, sua posição em relação aos ambientes virtuais como instrumento de construção para a sua aprendizagem e a qualidade de instrução que se deseja ensinar ao usuário.

Os requisitos de desempenho estão relacionados com o tempo de resposta e a qualidade da interatividade entre usuário e o ambiente virtual protótipo, ou seja, relacionam a usabilidade do protótipo. O critério de avaliação foi de acordo com a sensibilidade de cada participante que avaliou o ambiente virtual protótipo. Assim, quando um participante arrasta uma peça selecionada da simulação, o comando deve estar sincronizado à velocidade que o participante executa o movimento com o mouse, sem perceber uma latência no tempo de execução de tal comando.

Por último a qualidade visual diz respeito ao ambiente virtual protótipo como uma interface gráfica de ensino cuja característica é suficiente para desempenhar um processo de aprendizagem pelo estímulo visual.

Após a coleta da avaliação feita pelos participantes, os dados foram avaliados para retroalimentar o ambiente virtual protótipo e assim validar sua aplicabilidade como um instrumento de ensino e aprendizagem em construção civil.

5.2 RESULTADOS DOS TESTES REALIZADOS COM O AMBIENTE VIRTUAL PROTÓTIPO

5.2.1 Perfil dos participantes

Os sete indivíduos participantes dos testes com o ambiente virtual protótipo são alunos e alunas do primeiro ano do curso de Engenharia Civil da Universidade Federal do Paraná. Os participantes afirmam nunca ter usado um ambiente virtual voltado à simulação de atividades da construção civil. A idade dos participantes está em uma faixa entre 17 e 21 anos.

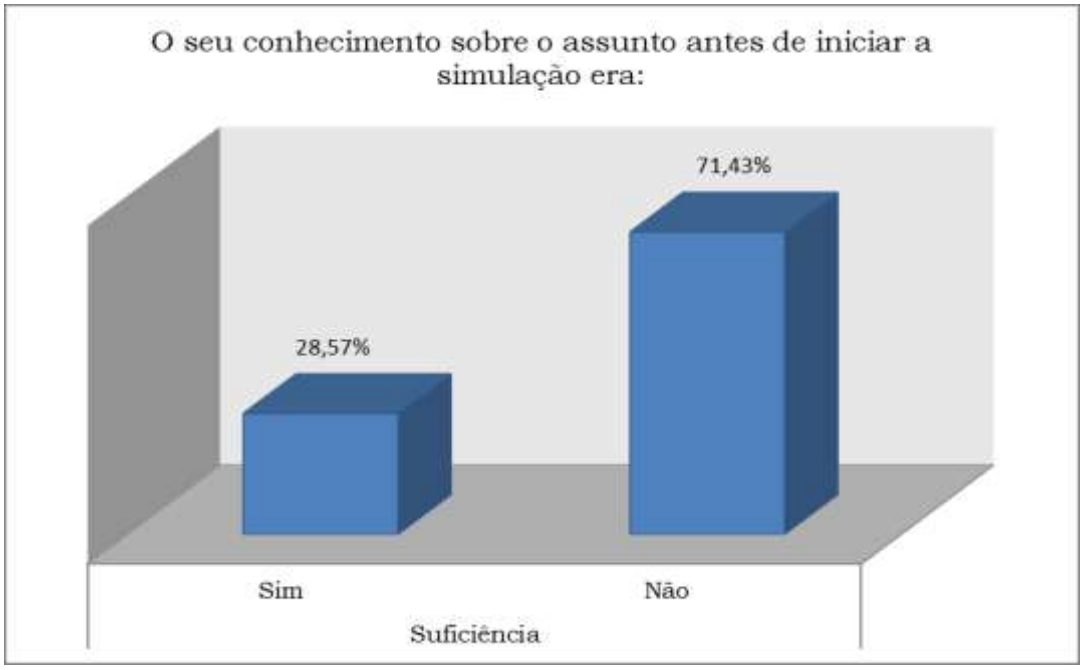

FIGURA 5.2 – NÍVEL DE CONHECIMENTO DOS PARTICIPANTES SOBRE CONSTRUÇÃO CIVIL.
FONTE: O Autor (2012).

Durante as sessões de testes os participantes foram questionados sobre o grau de relação e conhecimento com a construção civil. Somente 28,57% participantes afirmaram ter certo conhecimento sobre construção civil, como mostra o gráfico da figura 5.2. O nível de conhecimento sobre construção civil dos demais participantes estava limitado no máximo em uma visita a um canteiro de obras. Tal constatação foi realçada quando um dos participantes afirmou, pela simulação da

montagem da fôrma de um pilar de concreto armado, ser o primeiro contato com esse tipo de atividade da construção civil.

Em relação às ferramentas computacionais aplicadas a AEC os participantes afirmaram ter conhecimento somente sobre o Autodesk AutoCAD e, mesmo assim, com pouco domínio.

5.2.2 Aceitação do ambiente virtual

Conforme as respostas dos participantes em relação às informações dispostas nas telas de abertura apresentadas no gráfico da figura 5.3, seis participantes responderam ser muito importante e um participante respondeu ser importante. Do conjunto total de participantes, seis participantes responderam que as informações nas telas iniciais eram suficientes enquanto um participante respondeu que não eram suficientes, como demonstra o gráfico da figura 5.4. Ainda, todos os participantes responderam que a sequência é boa e sua compreensão é fácil, conforme o gráfico da figura 5.5.

A avaliação da apresentação do canteiro de obras, disposta no gráfico da figura 5.6, obteve as seguintes avaliações: seis participantes responderam ser muito importante e um participante respondeu ser importante. Todos os participantes avaliaram que a apresentação do canteiro de obra era suficiente e a sequência também, conforme os gráficos das figuras 5.7 e 5.8. Em relação à compreensão do canteiro de obras e objetos típicos como almoxarife e betoneira, todos responderam ser fácil de compreender, como demonstra o gráfico da figura 5.8.

FIGURA 5.3 – GRÁFICO DA AVALIAÇÃO DA RELEVÂNCIA DAS INFORMAÇÕES DISPOSTAS NAS TELAS DE ABERTURA.
FONTE: O Autor (2012).

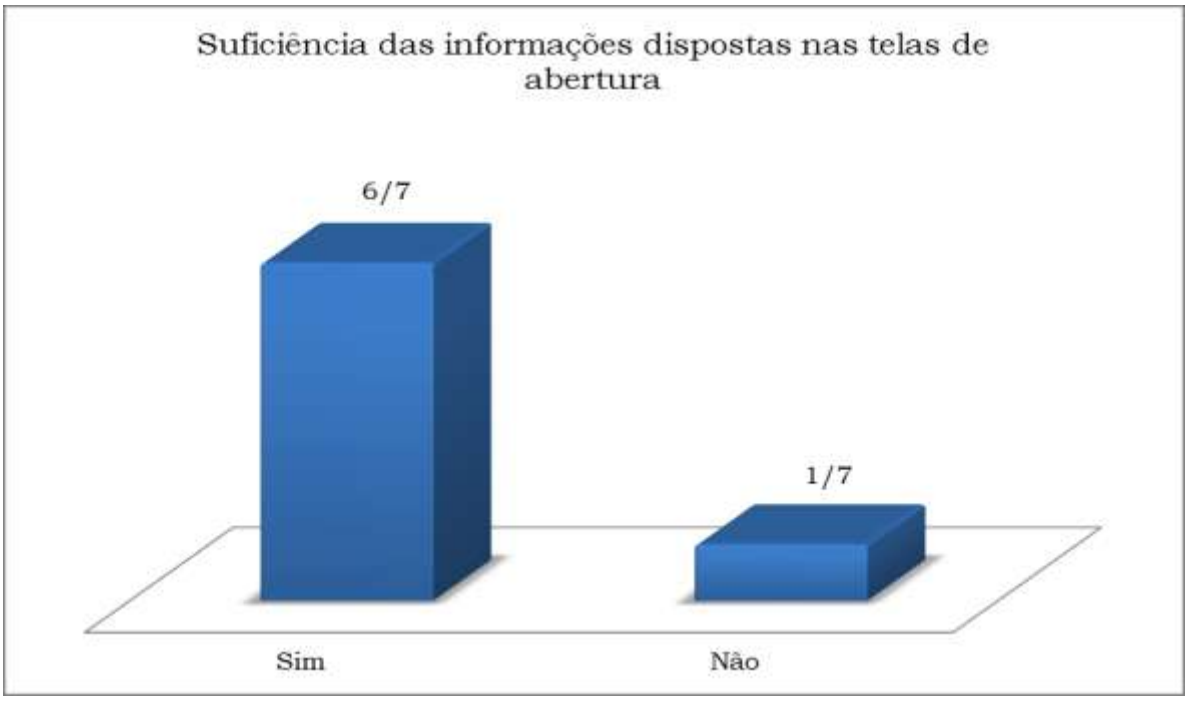

FIGURA 5.4 – GRÁFICO DA AVALIAÇÃO DA SUFICIÊNCIA DAS INFORMAÇÕES DISPOSTAS NAS TELAS DE ABERTURA.
FONTE: O Autor (2012).

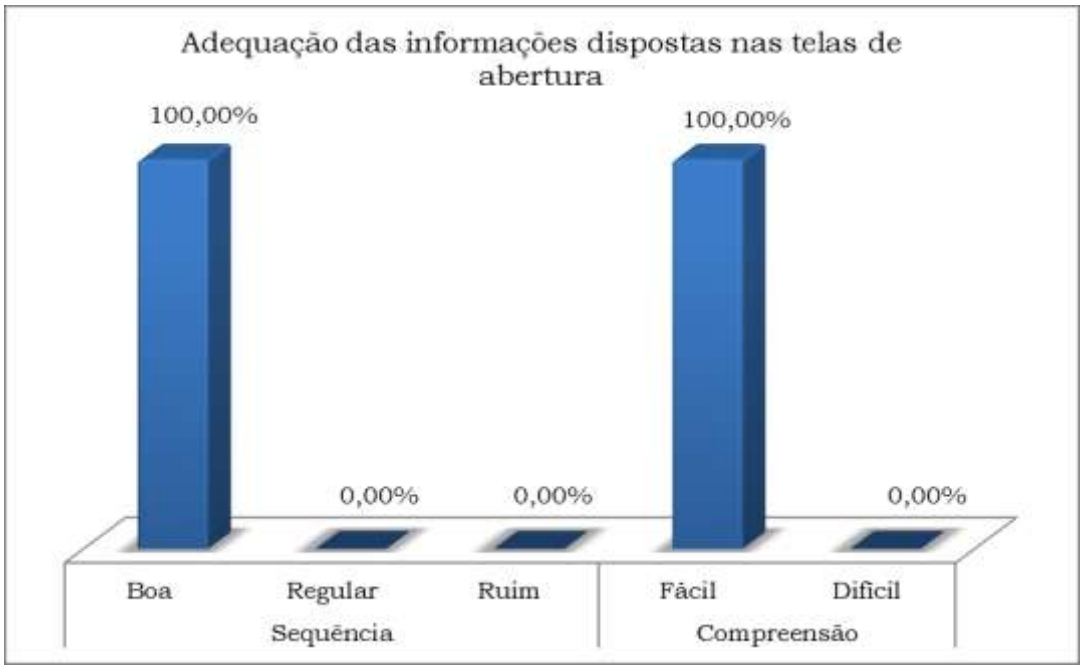

FIGURA 5.5 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DAS INFORMAÇÕES DISPOSTAS NAS TELAS DE ABERTURA.
FONTE: O Autor (2012).

FIGURA 5.6 – GRÁFICO DA AVALIAÇÃO DA RELEVÂNCIA DA APRESENTAÇÃO DO CANTEIRO DE OBRAS VIRTUAL.
FONTE: O Autor (2012).

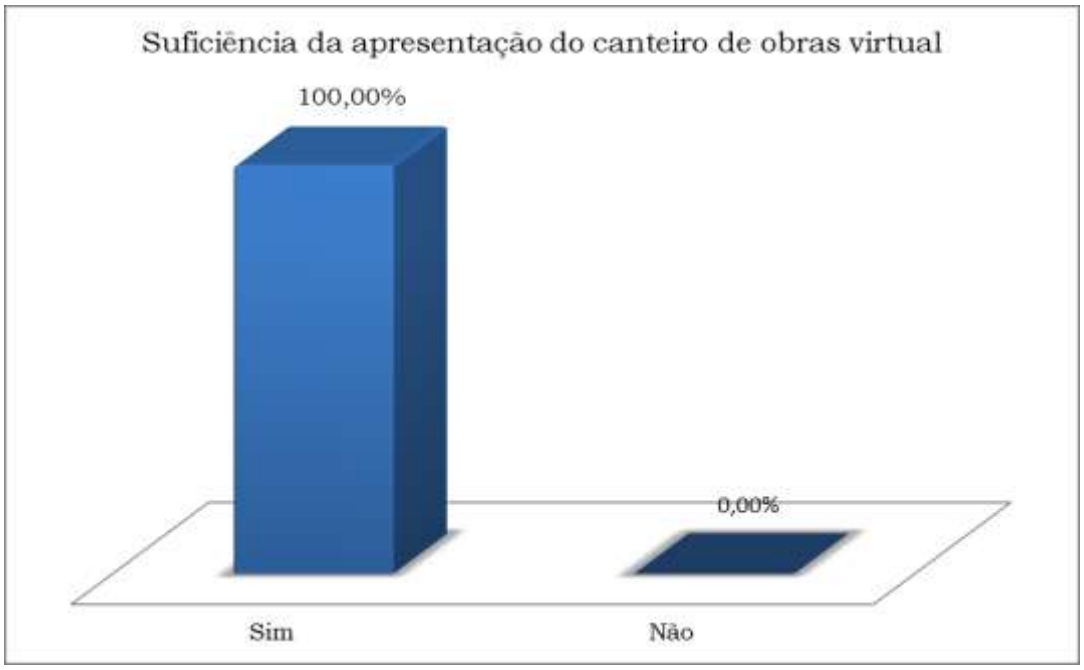

FIGURA 5.7 – GRÁFICO DA AVALIAÇÃO DA SUFICIÊNCIA DA APRESENTAÇÃO DO CANTEIRO DE OBRAS VIRTUAL.
FONTE: O Autor (2012).

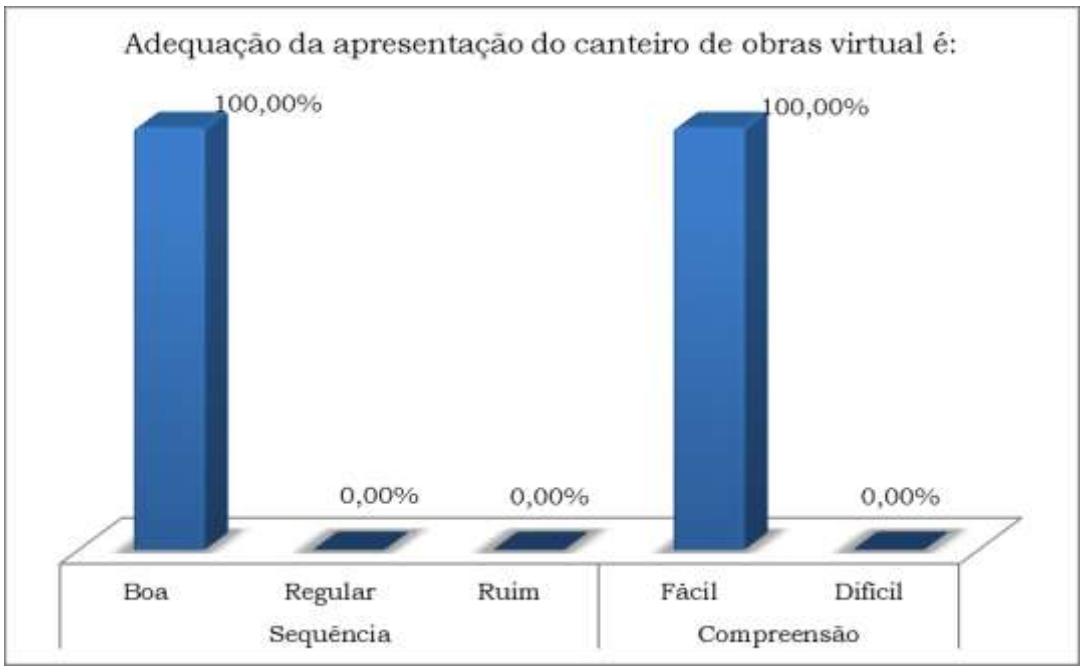

FIGURA 5.8 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DA APRESENTAÇÃO DO CANTEIRO DE OBRAS VIRTUAL.
FONTE: O Autor (2012).

A navegação oferecida pelo ambiente virtual foi avaliada da seguinte forma: seis participantes responderam ser suficiente e um participante respondeu não ser suficiente, como mostra o gráfico da figura 5.9. Conforme o gráfico da figura 5.10, quatro dos participantes responderam que a sequência é boa enquanto três participantes responderam que a sequência é regular. De todos os participantes, seis responderam ser fácil compreender os comandos para a navegação enquanto um participante respondeu ser difícil.

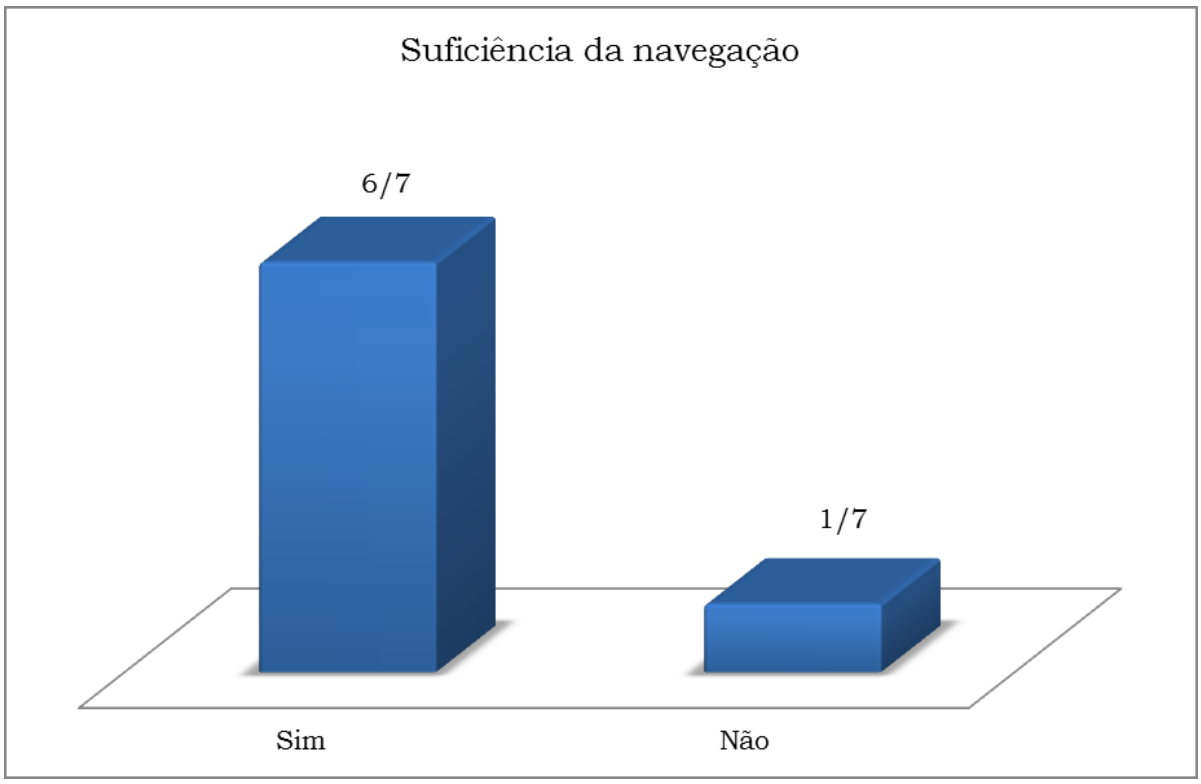

FIGURA 5.9 – GRÁFICO DA AVALIAÇÃO DA SUFICIÊNCIA DA NAVEGAÇÃO NO CANTEIRO DE OBRAS VIRTUAL.
FONTE: O Autor (2012).

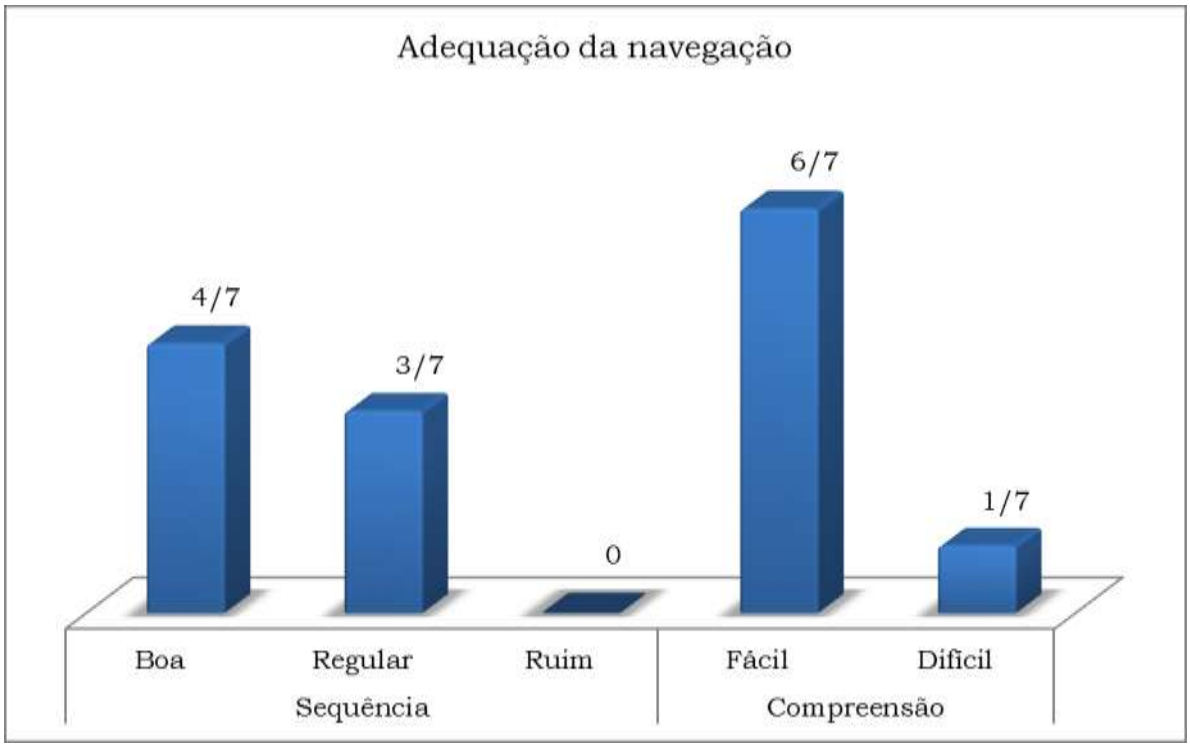

FIGURA 5.10 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DA NAVEGAÇÃO NO CANTEIRO DE OBRAS VIRTUAL.
FONTE: O Autor (2012).

Todos participantes avaliaram que não sentiram dificuldades em aprender por meio da simulação e ainda se sentira motivados e abertos a participar de novas metodologias de ensino e treinamento como mostra o gráfico da figura 5.11.

A qualidade da instrução do ambiente virtual protótipo recebeu a seguinte avaliação, como mostra a figura 5.12: cinco participantes classificaram como muito importante o aprendizado através de um ambiente virtual, conforme o gráfico da figura 5.12. Ainda, conforme o gráfico da figura 5.13, a avaliação da adequação clareza do conteúdo do ambiente virtual protótipo foi avaliada por todos os participantes como suficiente e destes, seis participantes avaliaram que a sequência está clara (boa), além de ser compreendido de forma fácil. Seis participantes avaliaram que o conteúdo do ambiente virtual protótipo é suficientemente objetivo, como mostra o gráfico da figura 5.14.

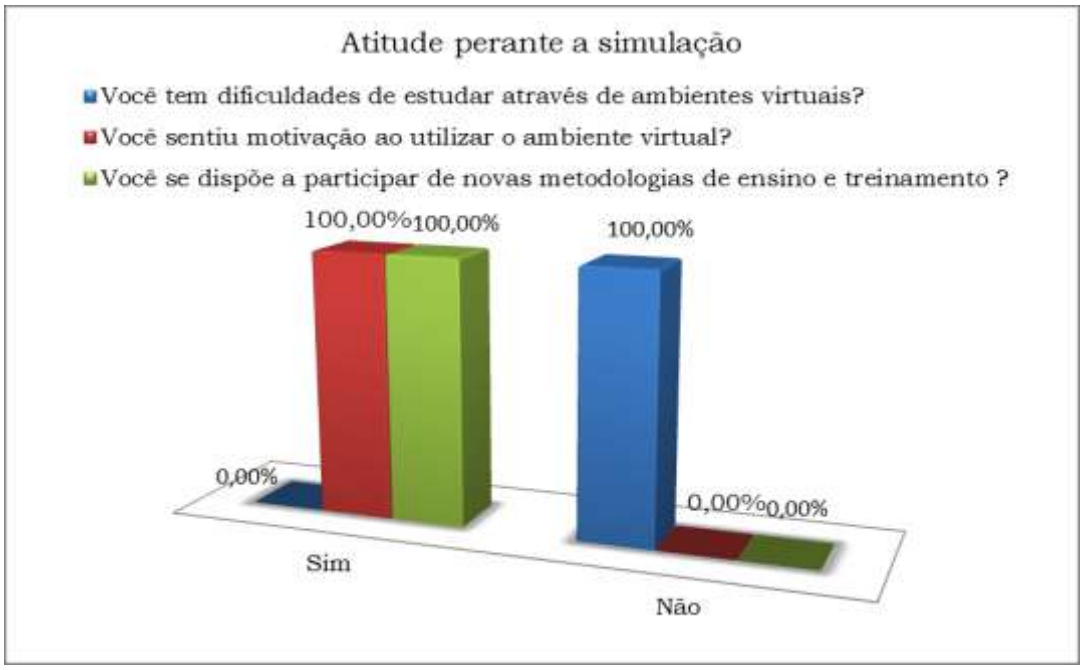

FIGURA 5.11 – GRÁFICO DA AVALIAÇÃO DO QUESITO
DA ATITUDE PERANTE A SIMULAÇÃO.
FONTE: O Autor (2012).

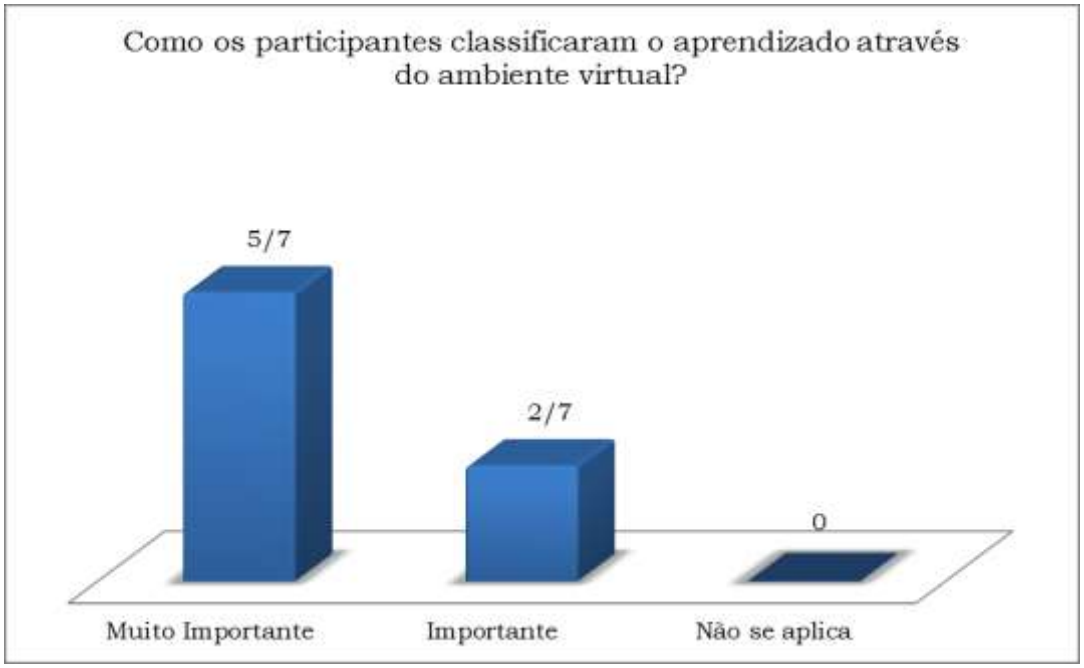

FIGURA 5.12 – GRÁFICO DA AVALIAÇÃO APRENDIZAGEM
EM UM AMBIENTE VIRTUAL.
FONTE: O Autor (2012).

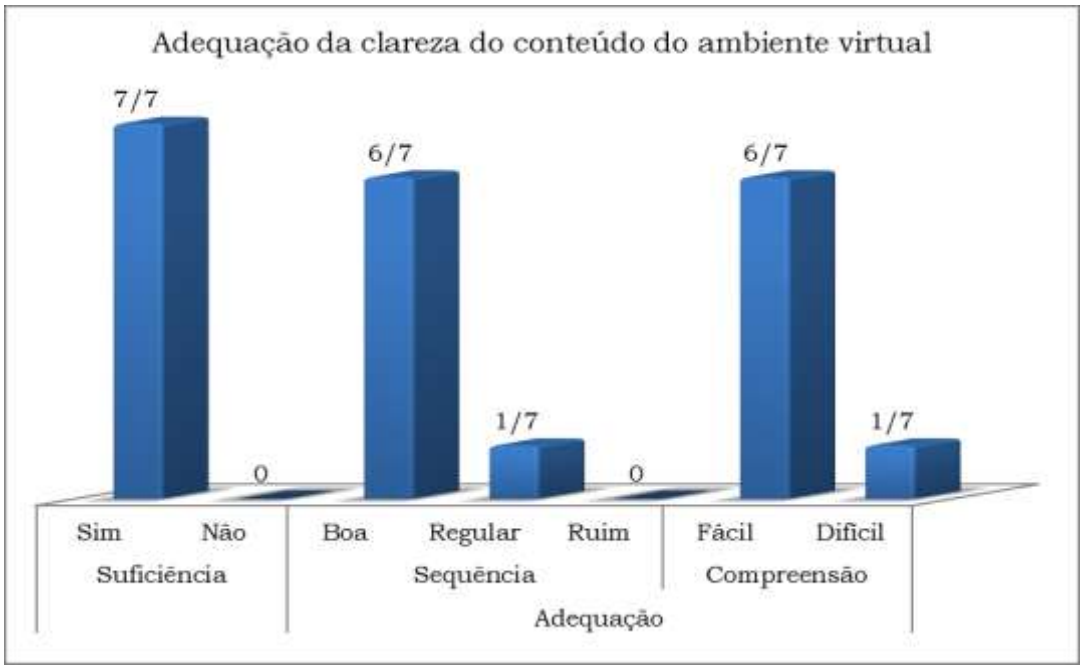

FIGURA 5.13 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DA CLAREZA DO CONTEÚDO APRESENTADO PELO AMBIENTE VIRTUAL.
FONTE: O Autor (2012).

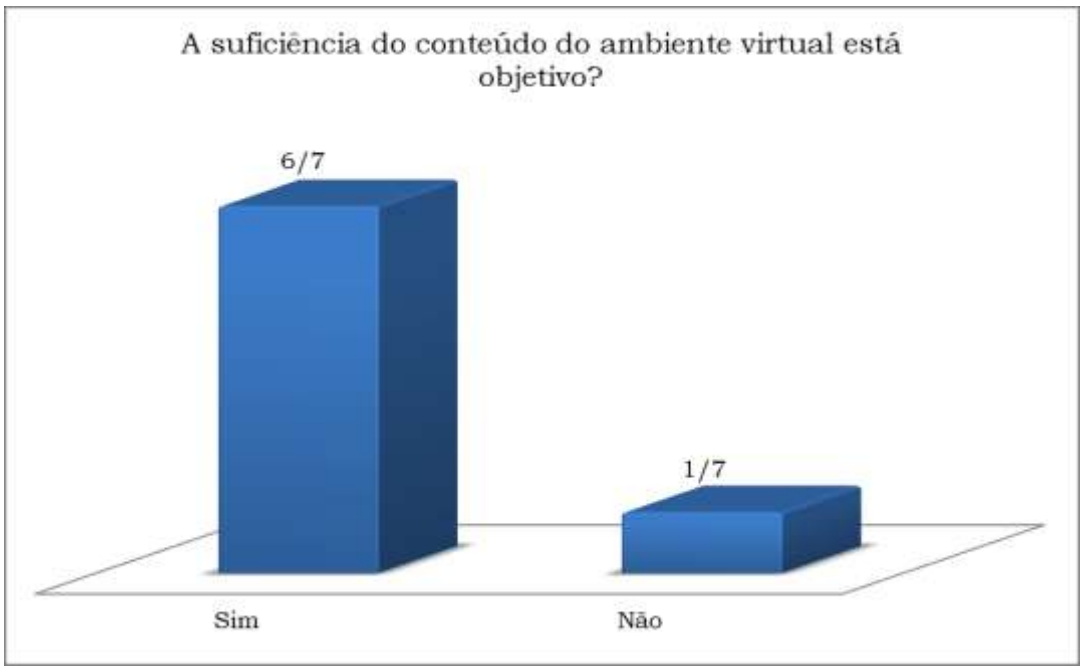

FIGURA 5.14 – GRÁFICO DA AVALIAÇÃO DA CONTEÚDO DO AMBIENTE VIRTUAL ESTÁ SUFICIENTEMENTE OBJETIVO.
FONTE: O Autor (2012).

5.2.3 Requisitos de desempenho

A avaliação sobre o tempo de resposta entre o ambiente virtual protótipo e o usuário ficou classificada como suficiente para seis dos participantes como mostra o gráfico da figura 5.15.

A interatividade proporcionada pelo ambiente virtual protótipo foi classificada como suficiente por todos os participantes. Conforme o gráfico da figura 5.16, seis participantes responderam como boa a sequência de interatividade assim como sua compreensão é fácil enquanto um participante classificou como regular a interatividade e consideraram como difícil sua compreensão. Ainda, todos os participantes responderam que o ambiente virtual protótipo é suficientemente motivador.

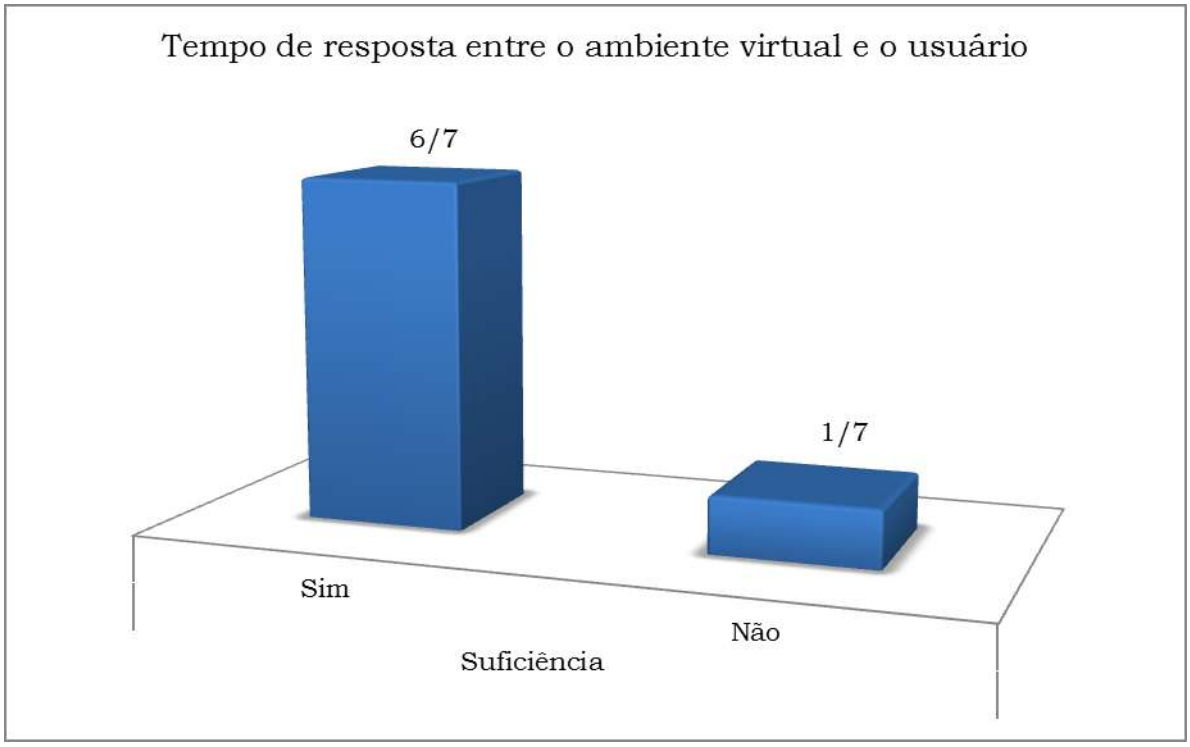

FIGURA 5.15 – GRÁFICO DA AVALIAÇÃO DO TEMPO DE RESPOSTA DA INTERATIVIDADE.
FONTE: O Autor (2012).

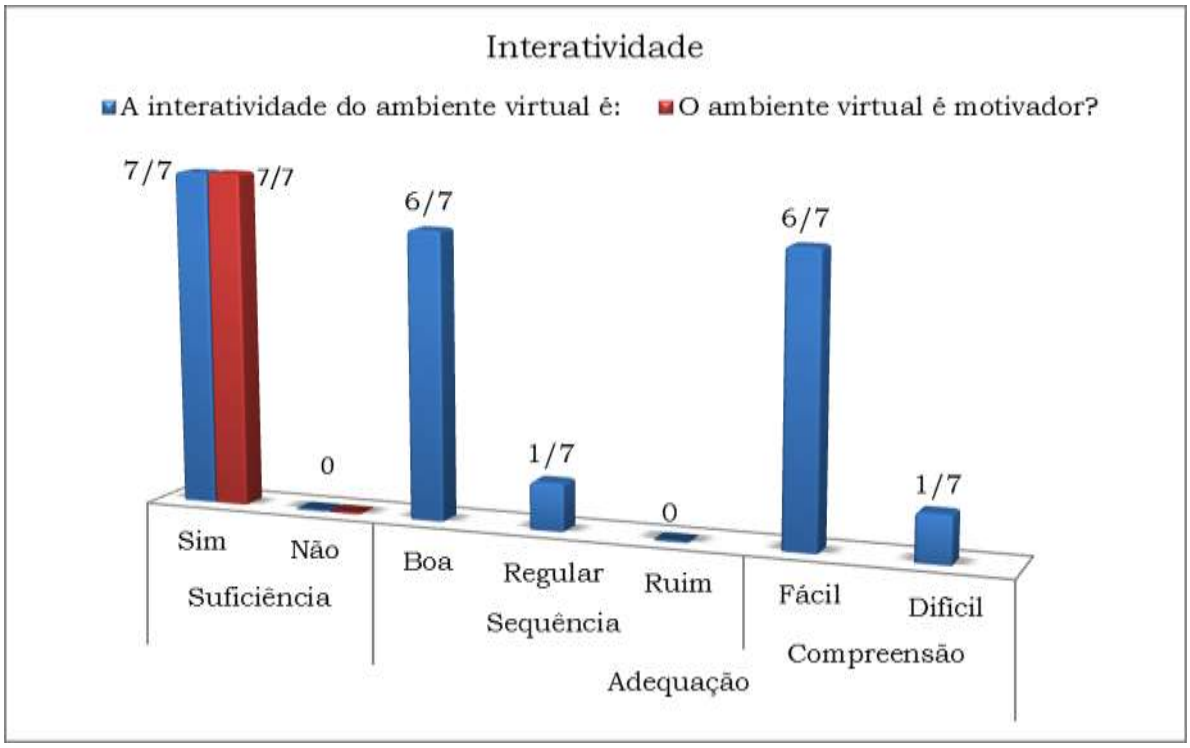

FIGURA 5.16 – GRÁFICO DA AVALIAÇÃO DA INTERATIVIDADE.
FONTE: O Autor (2012).

5.2.4 Qualidade visual

Conforme mostra o gráfico da figura 5.17 os participantes avaliaram o projeto visual onde quatro participantes avaliaram como muito importante enquanto três participantes avaliaram como importante. Todos os participantes responderam como suficiente o projeto visual do ambiente virtual protótipo assim como sua compreensão é fácil. Entre todos os participantes que avaliaram a adequação do projeto visual, seis participantes avaliaram como boa a sequência visual enquanto um participante avaliou como regular e todos os participantes avaliaram o projeto visual como fácil de compreender, como mostra o gráfico da figura 5.18.

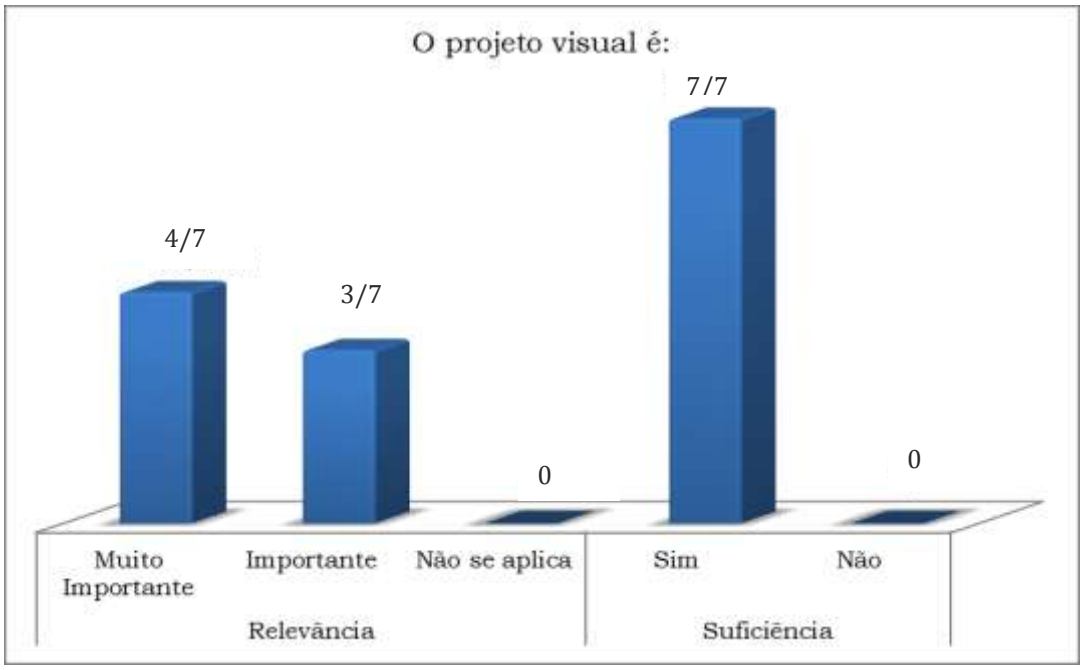

FIGURA 5.17 – GRÁFICO DA AVALIAÇÃO DO PROJETO VISUAL.
FONTE: O Autor (2012).

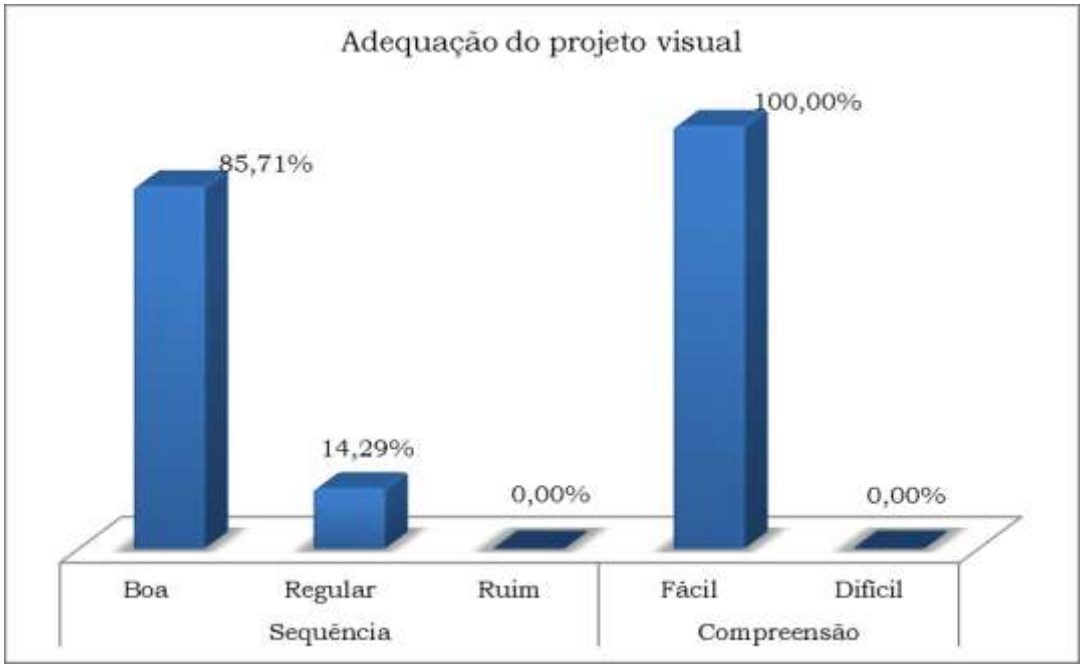

FIGURA 5.18 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DO PROJETO VISUAL.
FONTE: O Autor (2012).

O gráfico da figura 5.19 demonstra a avaliação dos participantes em relação à quantidade de informações que são expostas na tela de apresentação e três participantes responderam que tais informações são muito importantes enquanto quatro participantes responderam como importantes. Todos os participantes responderam que as informações são suficientes para a simulação e sua sequência é boa, além de serem facilmente compreendidas, como demonstrado no gráfico da figura 5.20.

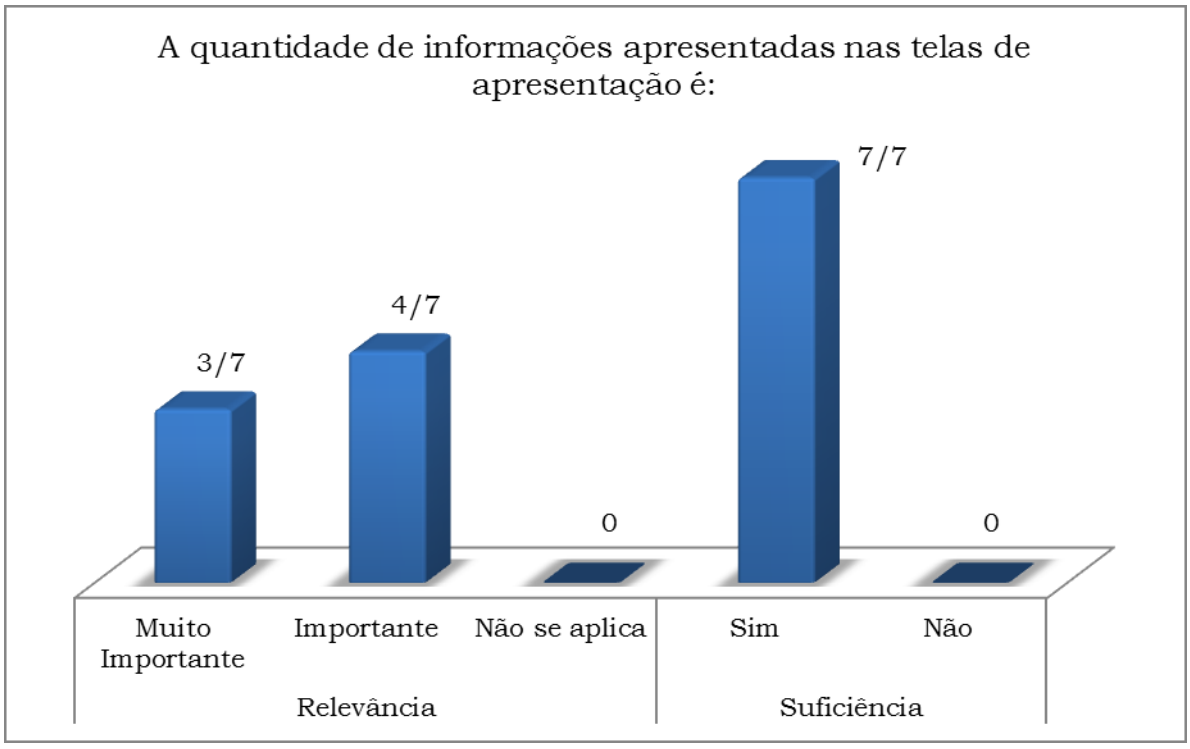

FIGURA 5.19 – GRÁFICO DA AVALIAÇÃO DA QUANTIDADE DE INFORMAÇÕES EXPOSTAS NAS TELAS DE APRESENTAÇÃO.
FONTE: O Autor (2012).

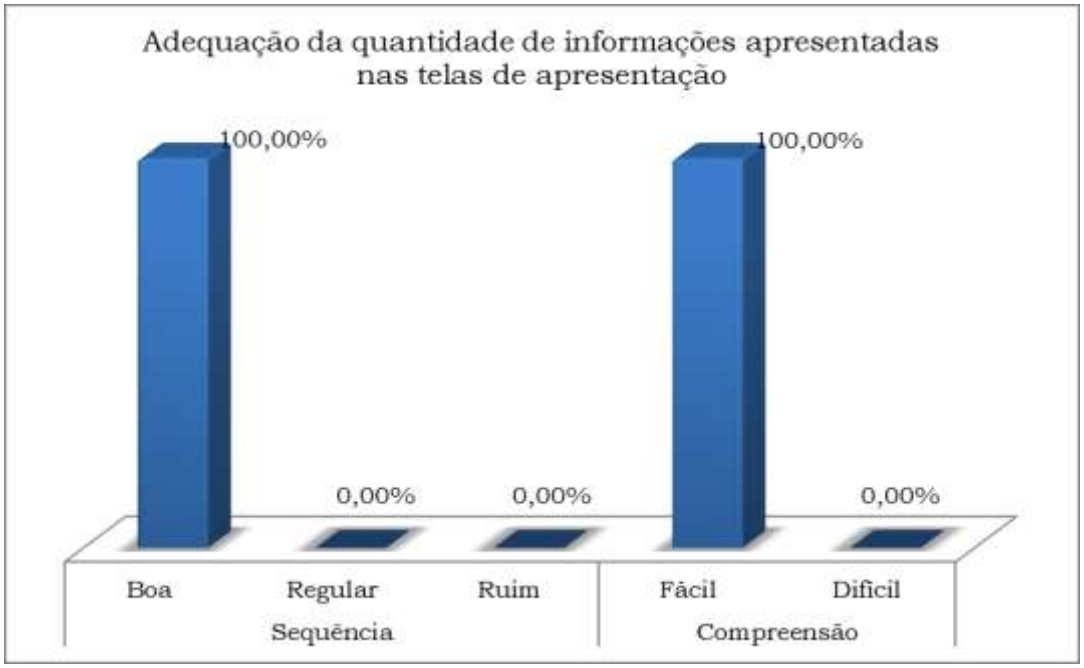

FIGURA 5.20 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DA QUANTIDADE DE INFORMAÇÕES EXPOSTAS NAS TELAS DE APRESENTAÇÃO.
FONTE: O Autor (2012).

O vocabulário usado para informar o conteúdo necessário à aprendizagem foi avaliado por todos os participantes como suficiente e está em uma sequência boa e de fácil compreensão, como mostra o gráfico da figura 5.21. Conforme o gráfico da figura 5.22 no quesito de relevância do vocabulário, seis participantes avaliaram o vocabulário como muito importante enquanto um participante avaliou como importante.

O gráfico da figura 5.23 mostra a avaliação do tamanho do texto usado para as informações sendo que seis participantes avaliaram como suficiente, com uma sequência boa e de fácil compreensão. Na avaliação sobre a adequação da compreensão, um dos participantes não respondeu a esta pergunta.

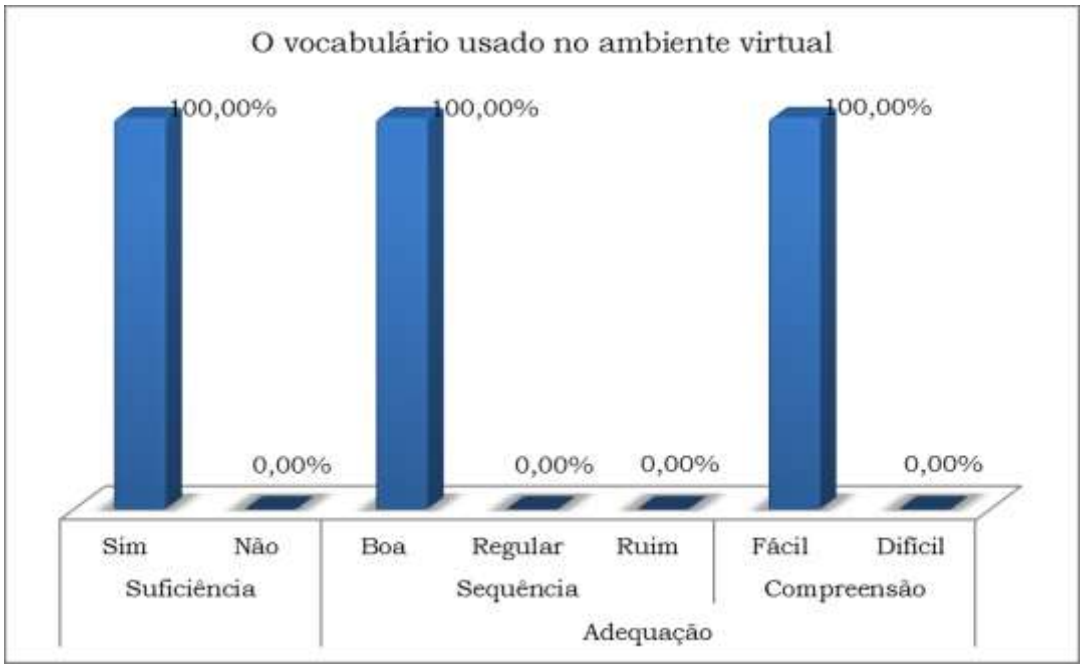

FIGURA 5.21 – GRÁFICO DA AVALIAÇÃO DO VOCABULÁRIO.
FONTE: O Autor (2012).

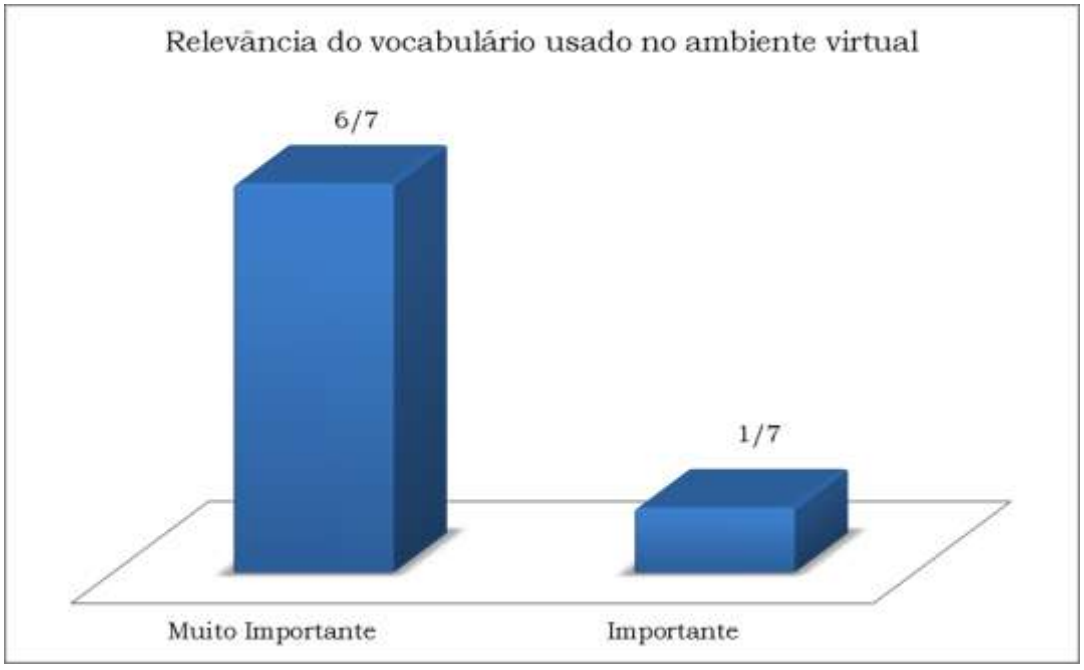

FIGURA 5.22 – GRÁFICO DA AVALIAÇÃO DA RELEVÂNCIA DO VOCABULÁRIO.
FONTE: O Autor (2012).

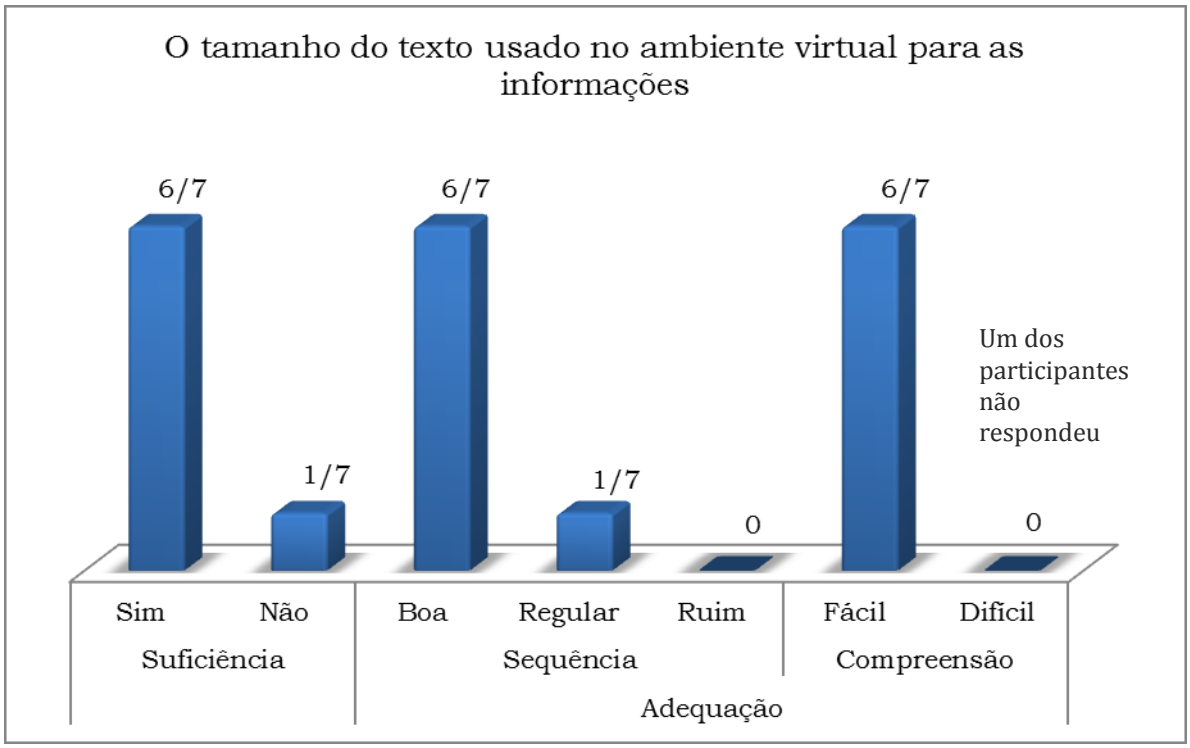

FIGURA 5.23 – GRÁFICO AVALIAÇÃO DO TAMANHO DA FONTE DO TEXTO.
FONTE: O Autor (2012).

Em relação às imagens apresentadas pelo ambiente virtual protótipo, todos os participantes avaliaram como fácil de compreender, onde seis participantes avaliaram a sequência de apresentação como boa, conforme o gráfico da figura 5.25. Ainda, quatro participantes avaliaram como suficiente enquanto dois participantes avaliaram como não sendo suficiente e um dos participante não respondeu a esta pergunta, como mostra o gráfico da figura 5.24.

A avaliação da quantidade de informações apresentadas pela legenda durante a simulação apresentou os seguintes resultados: três participantes avaliaram como muito importante enquanto quatro participantes avaliaram como importante e ainda, seis participantes avaliaram que as informações da legenda são suficientes, conforme o gráfico da figura 5.26. Sete participantes avaliaram a sequência da legenda como boa e todos os participantes avaliaram como fácil de compreender, como mostra o gráfico da figura 5.27.

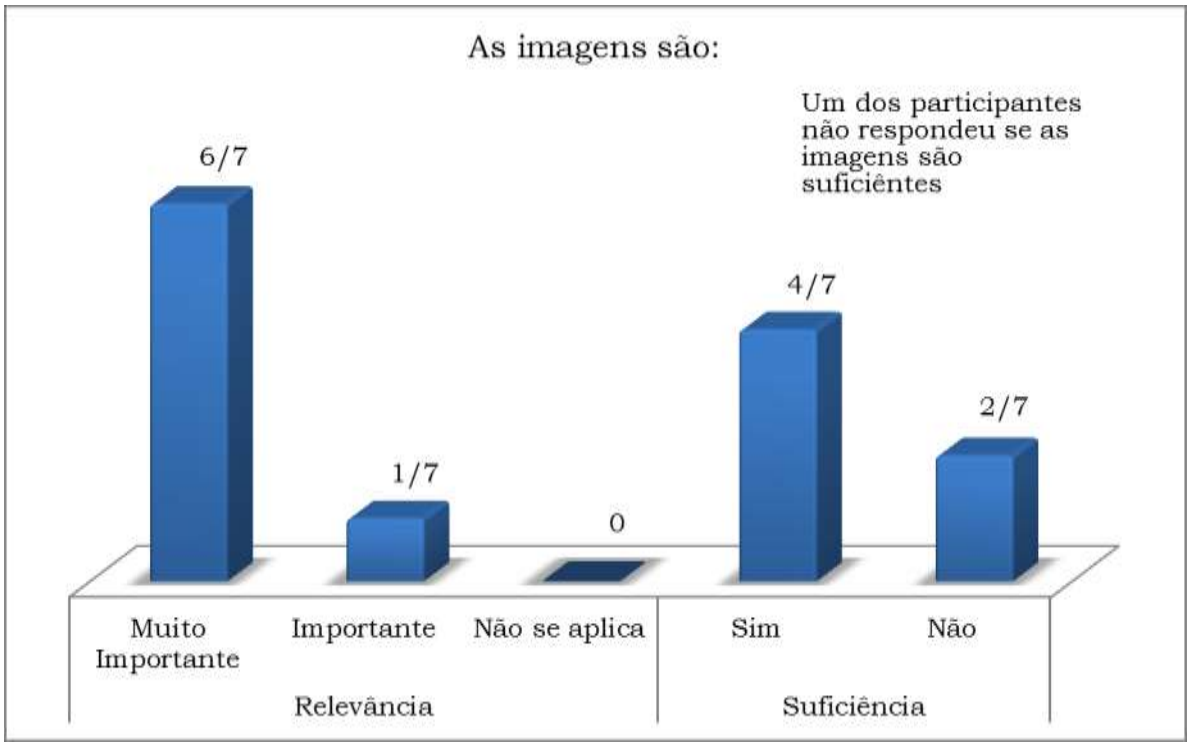

FIGURA 5.24 – GRÁFICO DA AVALIAÇÃO DAS IMAGENS USADAS.
FONTE: O Autor (2012).

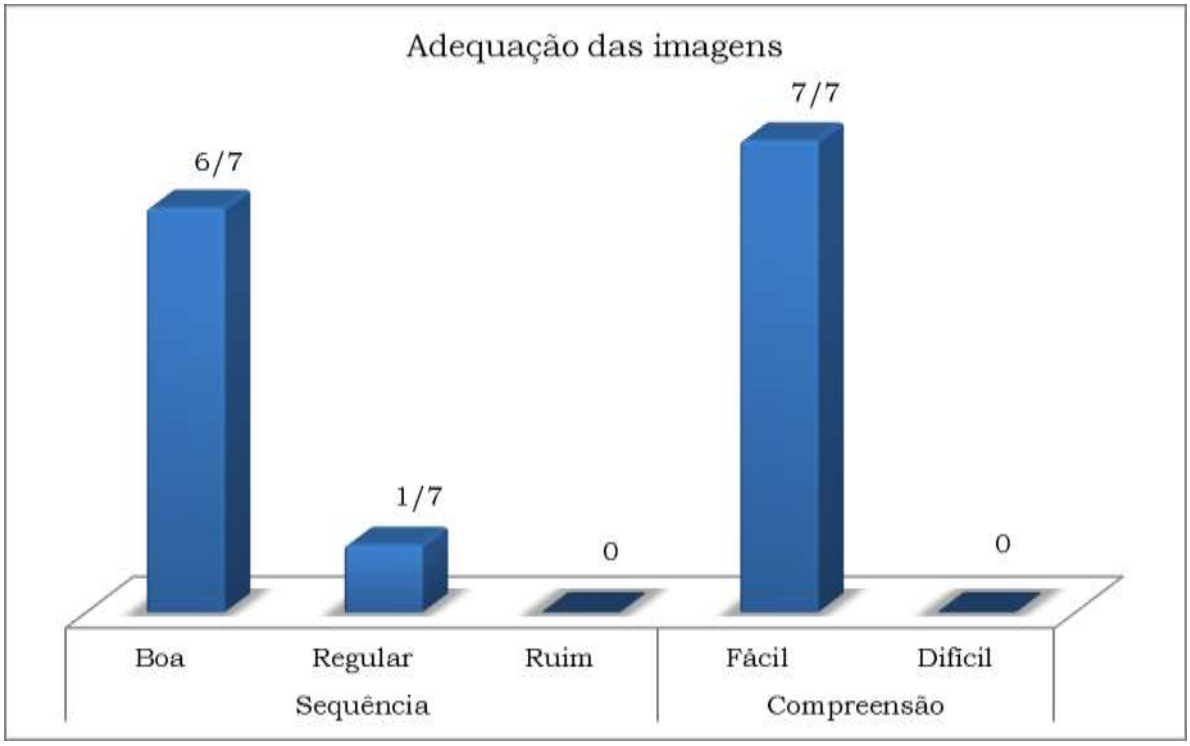

FIGURA 5.25 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DAS IMAGENS USADAS.
FONTE: O Autor (2012).

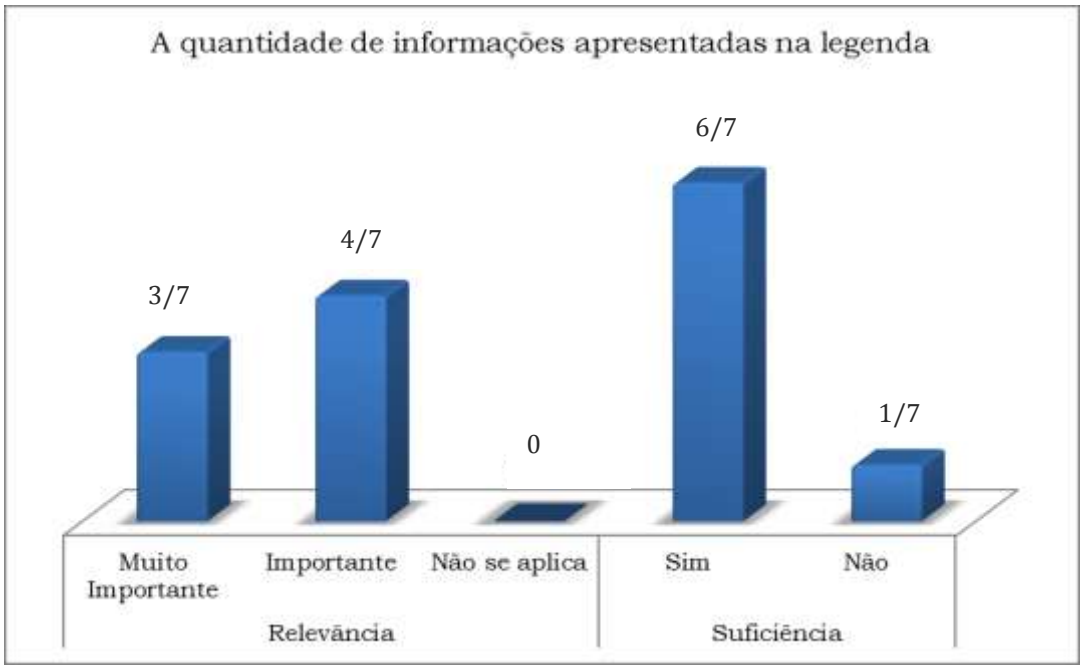

FIGURA 5.26 – GRÁFICO DA AVALIAÇÃO DA QUANTIDADE DE INFORMAÇÕES NA LEGENDA.
FONTE: O Autor (2012).

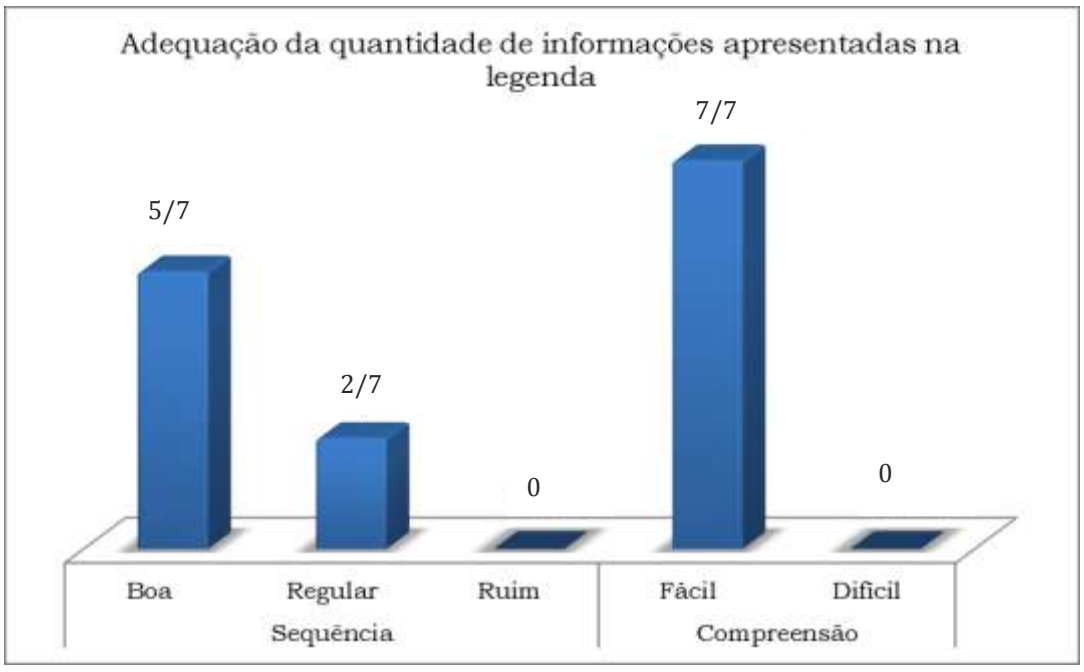

FIGURA 5.27 – GRÁFICO DA AVALIAÇÃO DA ADEQUAÇÃO DA QUANTIDADE DE INFORMAÇÕES NA LEGENDA.
FONTE: O Autor (2012).

5.3 VALIDAÇÃO POR ESPECIALISTAS NA ÁREA DA CONSTRUÇÃO CIVIL

Após os testes realizados com protótipo pelos alunos que participaram da simulação, foi realizada uma entrevista com o professor José de Almendra Freitas Junior, que ministra a disciplina de construção civil do curso de engenharia civil da UFPR. Ao demonstrar o ambiente virtual protótipo e a simulação, o professor afirmou que o protótipo tem potencial para ser utilizado como material aplicado na disciplina, ressaltando sua interface gráfica como suficientemente boa para estimular o aprendizado. Ainda, o professor explicou que para buscar uma melhor forma de ensinar os conteúdos em suas aulas, ele prepara vídeos de até cinco minutos sobre as temas relacionados à construção civil para estimular os alunos a apreciarem melhor o conhecimento sobre o assunto trabalhado durante a aula. Esses vídeos normalmente estão disponíveis gratuitamente na internet ou são produzidos pelo próprio professor. Porém, ele salientou a dificuldade da disponibilidade de vídeos com conteúdo especifico sobre construção civil. E ainda, nem sempre é possível filmar as atividades de construção civil, quer pela falta de motivação por parte das construtoras em autorizar uma filmagem de suas obras, quer pela disponibilidade de horário e recursos para filmar as atividades quando permitidas. Na opinião do professor, o protótipo pode ser uma solução, pois por meio de aplicativos de softwares livres seria possível gravar diversas simulações no próprio computador e transmitir em sala de aula para os alunos, buscando uma forma de estimular ainda mais os alunos sobre os conhecimentos da construção civil brasileira.

5.4 CONSIDERAÇÕES SOBRE O TESTE

As avaliações realizadas sobre a simulação mostram que o ambiente virtual protótipo oferece potencial para ser aplicado como complemento para instrução e ensino atingindo o seu objetivo principal: motivar a aprendizagem provendo também o entretenimento. Sua aplicabilidade pode ser usada tanto em aulas de laboratórios, podendo ser utilizados diversos computadores para abrigar uma turma completa ou dividir a turma em grupos com menor quantidade de alunos, atingindo um caráter menos formal e mais descontraído, mas motivador. É interessante ressaltar que o instrutor deve atuar como um facilitador para a construção do conhecimento aspirado pelos aprendizes. Ainda, o próprio instrutor pode usar a simulação para apresentar em uma sala de aula a visualização dos conceitos sobre a construção civil. Mas, apesar dos parâmetros avaliados atingirem bons resultados, algumas melhorias devem ser adotadas para aumentar sua aceitabilidade entre os aprendizes em AEC, sendo elas as informações dispostas sobre a atividade simulada e a navegação.

Dentre todos os participantes que avaliaram a quantidade de informações nas telas de abertura como suficiente para a simulação, um dos participantes considerou que as informações não são suficientes (figura 5.4) para aprendizagem. Isto ainda é confirmado quando se analisa a avaliação se o conteúdo do ambiente virtual está claro, onde um dos participantes considerou que a sequência é regular (figura 5.13) e ainda, a qualidade das imagens apresentadas foi avaliada como não sendo suficiente por dois participantes. Mas esta insuficiência pode ser contornada ao detalhar mais as informações sobre o processo de montagem, apresentando mais imagens sobre o assunto, detalhando melhor cada parte do processo de montagem e, ainda, demonstrar um esquema gráfico da sequência de montagem simulada e da sequência real, para que o instrutor possa explorar melhor o conteúdo quando sua metodologia de ensino adotar o ambiente virtual protótipo.

A navegação foi o maior alvo de críticas, onde três participantes não avaliaram como regular a adequação da navegação (figura 5.10), um participante não avaliou como suficiente (figura 5.9). Esta avaliação pode ser comprovada durante as aulas-teste, ondes os participantes apresentaram certa dificuldade para

navegar no canteiro de obras virtual. Parte dessa dificuldade se deu pelo fato da falta de familiaridade da configuração dos dispositivos usados para a navegação (mouse e teclado) e em parte foi notado que os participantes estão apenas acostumados a navegar em interfaces bidimensionais como aquelas presentes nas redes sociais da web, por exemplo. Outra reclamação apresentada durante as aulas-teste apontavam que a falta de colisão entre as peças e o piso da cena dificultava um pouco a navegação. Isso pode ser contornado restringindo o movimento das peças na direção perpendicular ao piso do canteiro de obra virtual. Os participantes sugeriram que a falta de familiaridade com a movimentação tridimensional da cena poderia ser resolvida adicionar eixos coordenados ortogonais na cena. Porém, o autor deste trabalho critica tal sugestão pelo fato do propósito do ambiente virtual é proporcionar uma aprendizagem motivadora em AEC e não para treinar a destreza do usuário para navegação em ambientes virtuais tridimensionais.

Os participantes conseguiram responder ao quesito do questionário “Descreva de forma simplificada a sequência da atividade simulada” com segurança e de forma correta, apresentando uma suficiência na aprendizagem sobre a atividade simulada. Ainda, dos participantes, aqueles que tinham certo conhecimento sobre construção civil apresentaram maior interesse em saber se existem mais procedimentos da atividade simulada.

5.5 CONSIDERAÇÕES FINAIS SOBRE A VALIDAÇÃO DO AMBIENTE VIRTUAL PROTÓTIPO

Na avaliação do ambiente virtual protótipo, tanto os discentes como o docente evidenciam que o protótipo atinge a característica desejada: ser um ambiente virtual educacional motivador. Mesmo com as críticas sobre a navegação e a necessidade de mais informações, por anseio dos participantes discentes, é possível estimular a construção do conhecimento relativo ao tema de Construção Civil por meio desta tecnologia de informação e comunicação.

Vale lembrar que os participantes pertencem à geração dos nativos digitais e mesmo assim é esperada uma dificuldade por parte deles em operar os comandos

relativos ao ambiente virtual protótipo. Isso ocorreu pelo fato dos participantes experimentarem a simulação uma única vez, o que acarretou um aumento na carga cognitiva na interatividade entre usuário e ambiente virtual protótipo.

6 CONCLUSÃO

Esta pesquisa estudou como abordar o ensino e aprendizagem sobre uma atividade relativa à Arquitetura, Engenharia e Construção através de um Ambiente Virtual. A motivação levou em conta que, uma vez detectado que os jovens da geração conhecida como nativos digitais almejam ingressar no mercado de trabalho brasileiro. Em específico, no mercado de AEC onde os cursos de graduação estão cada vez mais disputados por tais jovens. Sabendo que hoje existe certa resistência entre a aprendizagem focalizada somente na figura do instrutor. Fica evidente que existe a necessidade de mudar tal paradigma.

Assim, esta pesquisa procurou conhecimentos nas áreas de Computação Gráfica, Visualização Científica e Realidade Virtual para promover o desenvolvimento de uma ferramenta computacional que possua as características desejadas para motivar e contribuir para os alunos no seu processo de aprendizagem. Ainda, pela teoria de Piaget sobre o processo de construção do conhecimentofp, a Realidade Virtual se mostrou uma grande ferramenta para tal processo.

Uma vez determinado quais ferramentas são necessárias para o desenvolvimento de um protótipo de um ambiente virtual cuja característica é motivar e contribuir com a aprendizagem dos alunos de graduação em AEC, foi desenvolvido o ambiente de aprendizagem e um método de validação constituído de testes realizados com um pequeno grupo de alunos de engenharia civil da Universidade Federal do Paraná.

Os participantes que experimentaram a simulação conseguiram compreender que o ambiente virtual protótipo é uma forma didática ou mesmo uma metodologia de ensino que permite ao usuário motivar-se para produzir seu conhecimento. Isto ficou evidente quando os participantes avaliaram a suficiência das informações contidas no protótipo. Os resultados dos testes sugerem que os participantes, em sua visão de aluno, necessitam de mais informações tanto para praticar a simulação, mas também para conhecer mais sobre a atividade simulada. Ou seja, a motivação ficou evidenciada e comprova a eficácia do ambiente virtual como ferramenta que estimula e motiva os usuários a aumentar seu conhecimento.

Os participantes discentes conseguiram responder com sucesso como acontece o processo da atividade simulada, entendendo quais são as peças, o cenário (no caso, um canteiro de obras virtual) mesmo não tendo experiência sobre construção civil.

A proposta do protótipo responde ao objetivo de treinar e ensinar e consiste em uma solução para o uso de didáticas mais motivadoras sem comprometer o ensino. Sua aplicabilidade oferece potencial tanto em sala de aula como em laboratórios para os alunos participarem das simulações.

Como sugestão para trabalhos futuros sobre o assunto pesquisado: podem-se listar mais atividades vinculadas à Construção Civil, relacionando-as com as disciplinas da grade curricular dos cursos voltados a AEC e assim desenvolver simulações mais reais; revisar as bibliotecas gráficas computacionais que permitam maior facilidade para programar a simulação; pesquisar formas de interatividade mais intuitivas como dispositivos de interfaces tangíveis, melhorando a navegação no cenário virtual; pesquisar formas para implementar um ambiente colaborativo com mais deu um usuário interagindo durante a simulação.

7 REFERÊNCIAS

AMES, Andrea L.; NADEAU, David R.; MORELAND, John L. VRML 2.0 Sourcebook. 2nd ed. New York: John Wiley, 1997. 654 p.

ADAIME, L. M. Aplicação do *Visualization Toolkit* para pós-processamento de análises pelo método dos elementos. 141 f. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 2005.

AGOSTON, M.K. Computer Graphics and Geometric Modeling. London: Springer-Verlag, 2005.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6118/2007: Projetos de estruturas de concreto – Procedimentos. Rio de Janeiro, 2007.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 7190/1997: Projetos de estruturas de concreto – Procedimentos. Rio de Janeiro, 1997.

AVS Home Page. Disponível em: <http://www.avs.com>. Acesso em: 10 jan. 2011.

AZEVEDO, E.; CONCI, A. Computação Gráfica: teoria e prática. Rio de Janeiro: Elsevier, 2003.

BAI, X.; WHITE D.; SUNDERMAN D. Contextual adaptave visualization environment. The Eletronic Journal of Knowledge Management, Reino Unido, Vol. 10 n. 1, p. 01-14, 2011.

BAIRD, J. B. New from the computer: 'Cartoons' for the courtroom. New York: New York Times, 1992.

BAKER, M. P.; HEARN, D. Computer Graphics, C version. 2nd ed. New Jersey: Prentice Hall, 1997.

BATTAIOLA, A. L.; SOARES, L. P. Estudo e Uso Exploratório de Ferramentas de Visualização Científica. In: SEMANA DE INFORMÁTICA DA UFBA, 7., 1998, Salvador. Anais... Salvador: UFBA, 1998. p. 16-30.

BLANCHARD, C.; BURGESS, S.; HARVIL, Y.; LANIER, J.; LASKO, A.; OBERMAN, M.; TEITEL, M. Reality built for two: a virtual reality tool. In: ACM SIGGRAPH Special Issue on Symposium on Interactive 3D Graphics vol. 24, n. 2,1990, Snowbird, Utah. Simpósio… Snowbird, Utah, 1990. p. 35-36.

BLANCO, E.; SILVA, B. Tecnologia educativa em Portugal: conceito, origens, evolução, áreas de intervenção e investigação. Revista Portuguesa de Educação. Braga, Vol. 6, n. 3, p. 37-55, 1993.

BLYTHE, D.; McREYNOLDS, T. Advanced Graphics Programming Techniques Using OpenGL. Tutorial apresentado no SIGGRAPH 1999. Disponível em: < http://swarm.cs.pub.ro/~mihai/OpenGL_books/Advanced%20OpenGL%20Programming.pdf >. Acesso em: 16 jan. 2010.

BOLAS , M.; KRUM, D.M. Augmented reality applications and user interfaces using head-coupled near-axis personal projectors with novel retroreflective props and surfaces. In: UBIPROJECTION WORKSHOP, PERVASIVE '10, 2010, Helsinki. Proceedings… Helsinki, 2010. pp 5–8.

BOLAS, M. PAIR, J. HAYNES K. MCDOWALL, I. Display Research at the University of Southern California. IEEE Emerging Displays Workshop, Alexandria, VA. 2006.

BROOKS, F. P. Virtual reality at work In: HUMAN MACHINE INTERFACES FOR TELEOPERATORS AND VIRTUAL ENVIRONMENTS, 1990, Santa Barbara. Conferência…Santa Barbara: NASA Conference Publication 10071, 1990. p. 67-68.

BROOKS,F.P. Walkthrough—a dynamic graphics system for simulating virtual buildings. In: ACM WORKSHOP ON INTERACTIVE 3D GRAPHICS, 1986, New York. Proceedings… New York: I3D '86 Proceedings of the 1986 workshop on Interactive 3D graphics, 1987, p. 9-21.

BOWMAN, D. A. Interaction techniques for common tasks in immersive virtual environment – design, evaluation and application. 132 p. Tese (Doctor in Philosophy in Computer Science) – Georgia Institute of Technology, Georgia, 1999.

BRODLIE, **K. et al. Scientific Visualization, techniques and applications. Springer-Verlag, 1992.**

BRODLIE, **K. A classification scheme for scientific visualization. In: EARNSHAW**, **R. A.; WATSON**, **D. (Eds.) Animation and Scientific Visualization: tools & applications. Academic Press, 1993. p. 125-140.**

BRODLIE, **K. Scientific Visualization: past, present and future. Nuclear Instruments & Methods in Physics Research, p. 104-111. Elsevier Science, 1995.**

BURIOL, T. M.; ROSENDO, M.; SCHEER, S.; de GEUS, K.; FELSKY, C.; GOULART, J. C. Proposta de plataforma baseada em realidade virtual para treinamento de atividades em linha viva. In: XXX CILAMCE - Congresso Ibero Latino Americano de Métodos Computacionais em Engenharia, 2009, Armação dos Búzios. Congresso... Armação dos Búzios, 2009. p. 1–13.

BURIOL, T. M. Convergência de Games e Realidade Virtual para Treinamento de Manutenção em Redes de Energia em Linha Viva. 135 p. Tese (Doutorado em Métodos Numéricos em Engenharia) – Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 2011.

BUXTON, B. Snow's two cultures revisited: Perspectives on human-computer interface design**. In Linda Jacobson (Ed.). CyberArts: Exploring art and technology, 1992, p. 24–38.**

CALIL JUNIOR, C.; OKIMOTO, F.; STAMATO, G. C.; PFISTER, G. SET 613 - Formas de madeira para concreto armado. Universidade de São Paulo, Escola de Engenharia de São Carlos. São Carlos, 2000.

CAREY, R.; BELL, G. The Annotated VRML 2.0 reference manual. Reading, Mass.: Addison-Wesley Developers Press, 1997.

CARVALHO, R. G. Mudanças promovidas pela aprendizagem colaborativa e tecnologia da informação em sala de aula na disciplina Sistemas Estruturais. 112 p. Dissertação (Mestrado em construção civil) – Universidade Federal do Paraná, Curitiba, 2003.

CEF Homepage. Disponível em: <http://www1.caixa.gov.br/gov/gov_social /municipal/programas_habitação/ pcmcmv/saiba_mais.asp>. Acesso em: 15/01/2012.

COHEN, M.; MANSSOUR, I. H. OpenGL – Uma abordagem prática e objetiva. São Paulo: Novatec, 2006.

COLLINS, B. M. Data visualization: has it all been seen before? In: EARNSHAW, R. A.; WATSON, D. (Eds.) Animation and Scientific Visualization: tools & applications. Academic Press, 1993. p. 3-28.

CORSEUIL, E. RAPOSO, A. SILVA, R. PINTO,M. WAGNER, G. GATTASS, M. A VR Tool for the Visualization of CAD Models. In: SYMPOSIUM ON VIRTUAL REALITY, 7, 2004, São Paulo. Anais…Porto Alegre, SBC, 2004. p. 327-338.

CORREIO BRAZILIENSE Homepage. Disponível em: < http://www.correiobraziliense .com.br/app/noticia /cidades/2011/08/14/interna_cidadesdf,265443/maior-parte-dos-cursos-procurados-no-df-sao-para-areas-saturadas.shtml>. Acesso em: 15/01/2012.

CUPERSCHIMID, A. R. M. RUSCHEL, R. C. MARTINS, F. A. Uso de realidade aumentada para visualização do modelo da edificação. V TIC 2011. Bahia, 2011

DAMBRASCO, U. Métodos da topologia, introdução e aplicações. Rio de Janeiro: Livros Técnicos e Científicos, 1977.

DeFANTI,T.A. BROWN,M.D. STEVENS,R. Virtual reality over high-speed networks. IEEE Comput. Graphics Appl. USA.1996

DeFANTI, T. A. DAWE, G. SANDIN, D. J. SCHULZE, J. P. OTTO, P. GIRADO, J. KUESTER, F. SMARR, L. RAO, R. The StarCAVE, a third-generation CAVE and virtual reality OptIPortal. Future Generation Computer Systems, USA, vol.25, no.2, pp.169-178, 2009.

DeFANTI,T.A. FOSTER,I. PAPKA,M.E. STEVENS,R. KUHFUSS,T. Overview of the I-way wide area visual supercomputing, International Journal of Supercomputer Applications, USA, vol. 10, n.2, 1996.

DeFANTI,T.A. SANDIN,D.J. CRUZ-NEIRA,C. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In: SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques ACM. 1993, New York. Proceedings…New York, 1993.

ERICKSON, T. Artificial realities as data visualization environments.In Alan Wexelblat (Ed.), Virtual reality: Applications and explorations Boston: Academic Press Professional, 1993, P. 1 – 22.

FERREIRA, G. LOUREIRO, E. NOGUEIRA, W. FERREIRA, A. ALMEIDA, H. FERRY, A. Uma Abordagem Baseada em Componentes para a Construção de Edifícios Virtuais. Universidade Federal de Campina Grande – Campina Grande, PB – 2004.

FILE FORMAT homepage. Disponível em: <http://www.fileformat.info/format/wave frontobj/egff.htm#WAVEOBJ-DMYID.3>. Acesso em janeiro de 2012.

FISHER, S. et al Micro-Archiving and Interactive Vritual Insect Exhibit. Disponível em: <http://www.itofisher.com/PEOPLE/sfisher/MAP_SPIE-Jan02.PDF >. Acesso em: Janeiro de 2011.

FISHER, S. Environmental Media: Accessing Virtual Representations of Real-Time Sensor Data and Site-specific Annotations Embedded in Physical Environments.In: WSMM '01 SEVENTH INTERNATIONAL CONFERENCE ON VIRTUAL SYSTEMS AND MULTIMEDIA, 2001, Berkley. Proceedings…Thwaites and Addison, eds. Los Alamitos: IEEE Comuter Soc., 2001. p. 25-27.

FISHER, S. Recent developments in virtual experience design and production. Stereoscopic Displays and Virtual Reality Systems SPIE 2409,San Jose, CA, USA,1995. Disponível em: < http://www.itofisher.com/PEOPLE/sfisher/Virtual ExperienceDesign-SPIE1995.pdf>. Acesso em: 01/01/2011.

FISHER, S. JACOBY, R. BRYSON, S. STONE, P. MCDOWALL, I. BOLAS, M.DASARO, D. WENZEL, E. COLER, C KERR, D. The Ames Virtual Environment Workstation: Implementation issues and requirements. In: HUMAN MACHINE INTERFACES FOR TELEOPERATORS AND VIRTUAL ENVIRONMENTS, 1990, New York. Proceedings…New York, 1990. p 70-73.

FLANNERY K. A.; KRAUCHUNAS S.; FLINN, P. Virtual cognition lab: a constructivist approach toward learning. In: 3rd WORKSHOP ON VIRTUAL REALITY IN RGS, 2000, Gramado. Workshop… Gramado, 2000.

FOLEY, J. D. et al. Computer graphics: principles and practice. 2nd ed. New York: Addison Wesley, 1996.

FRANCIS, G. A.; TAN, H.S. Virtual Reality as a Training Instrument. The Temasek Journal, vol. 7. pp. 4 –15, 1999.

GAMA, C. L. G. Método de construção de objetos de aprendizagem com aplicação em métodos numéricos. 196p. Tese (Doutorado em Métodos Numéricos em Engenharia) – Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 1999.

GIBSON, W. Neuromancer**. New York: Bantam Books, 1886.**

GIRADO, J. SANDIN, D. DEFANTI, T. WOLF, L. *Real- time Camera-based Face Detection using a Modified LAMSTAR Neural Network System*. Proceedings of IS&T/SPIE's 15th Annual Symposium Electronic Imaging 2003, Applications of Artificial Neural Networks in Image Processing VIII. **Proceedings…**CA, USA. 2003

GLOBUS, A.; RAIBLE, E. Fourteen ways to say nothing with Scientific Visualization. IEEE Computer, v. 27, p. 86-88, 1994.

GOMÉZ, R.; TREFFTZ, H. Design and implementation of a low cost projected virtual reality system to support learning process. In: HCI INTERNATIONAL 2011 - 14th INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 2011, Orlando.Anais… Communication In Computer and Information Science, 2011, vol. 174, p. 107-111.

GRIMES, D.; WARSCHAUER, M.; HUTCHINSON, T. C.; KUESTER, F. Civil engineering education in a visualization environment: Experiences with vizclass. Journal Engineering Education, vol. 95, n. 3, p. 249-254, 2006.

HAQUE, M.E. 3-D Visualization and Animation Techniques in Structural Design Education. In: CIBW78 CONFERENCE IT IN CONSTRUCTION, South Africa, 2001.

HAMILTON, J. Virtual reality: How a computer generated world could change the world**. Businessweek, 1992, f. 96–105.**

HARTMAN, J.; WERNECKE, J. The VRML 2.0 handbook: building moving worlds on the web. Reading, Mass.: Addison-Wesley,1996.

HILL, F. S. Computer graphics using OpenGL. 2nd ed. New Jersey: Prenticie-Hall, 2000.

HINCKLEY, K. PAUSCH, R. PROFFITT, D. KASSELL, N. Attention and visual feedback: The bimanual frame-of-reference. Proceedings of the ACM/SIGGRAPH Symposium on Interactive 3D Graphics1997, NY, USA. Proceedings…1997

HOUNSELL, M. da S.; PIMENTEL, A. On the Use of Virtual Reality to Teach Robotics, 3rd International Conference on Engineering and Computer Education, São Paulo: 2003.

HOUNSELL, M. da S.; SILVA, E. L. da; MIRANDA, J. J. de. Detalhando aspectos de educação e treinamento em ambientes virtuais 3d. In: INTERTECH – 10th INTERNATIONAL CONFERENCE ON ENGINERING AND TECHNOLOGY EDUCATION. 2008. p. 641- 645.

HUTCHINSON, T.; KUESTER, F.; KIM, S.; LU, R. Developing an advanced it-based visualization classroom for enhanced engineering learning. In: INTERNATIONAL CONFERENCE ON ENGINEERING EDUCATION, Valencia, 2003.

IBM. IBM Research Visualization Data Explorer. Disponível em: <http://www. research.ibm.com/dx>. Acesso em: 02 fev. 2005.

JACOBSON, L. Realidade Virtual em casa. Rio de Janeiro: Berkley, 1994.

KAJIMOTO, H. KAWAKAMI, N. TACHI, S. INAMI, M. *Smarttouch: electricskin to touch the untouchable*, IEEE Computer Graphics and Applications 24. Pg 36–43.2006.

KHRONOS GROUP homepage. Disponível em: <http://www.khronos.org/>. Acesso em 16 jan. 2010.

KITWARE. The VTK user's guide: install, use and extend The Visualization Toolkit. 2004.

KITWARE. CMake: cross platform make. Disponível em: <www.cmake.org>. Acesso em: 10 jul. 2005.

KITWARE. VTK home page. Disponível em: <http://www.vtk.org>. Acesso em: 19/01/2010.

KRUEGER,M. Responsive environments. AFIPS '77, n. 46, p.423-429, 1977.

KRUEGER, M. Real-time laboratory for interdisciplinary computer projects. SIGCSE '78 Papers of the SIGCSE/CSA technical symposium on Computer science education, ACM, New York, USA.1978

KRUEGER, M. Artificial Reality II. Reading, Mass: Addison-Wesley.1991.

KRUEGER, M. Automating virtual reality. IEEE Computer Graphics and Applications, USA.1995.

KRUEGER, M. GILDEN, D. Going places with "KnowWare": Virtual reality maps for blind people, Lecture Notes Computer Science, vol. 2398, p.565, 2002.

KRUEGER, M. GILDEN, D. KnowWhere: an Audio/Spatial Interface for Blind People. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY (ICAD), 1997, PaloAlto. Proceedings… Palo Alto: Xerox PARC, 1997, p. 1-4.

KRUEGER, M. GIONFRIDDO, T. HINRICHSEN, K. *Videoplace—an artificial reality*. CHI '85 Proceedings of the SIGCHI conference on Human factors in computing systems. ACM. New York,NY,USA. 1985.

KRUEGER, M.; ROLLAND, J. P.; GOON, A. Multifocal planes head-mounted displays. Appl. Opt., vol. 39, n.19, p. 3209–3215, 2000.

KU, K.; MAHABALESHWARKAR, P. Building interactive modeling for construction education in virtual worlds. Journal of Information Technology in Construction – Itcon, vol. 16, p. 189-208, 2011.

LANDAVERDE, F. J. Curso de geometria para secundario y preparatório, 6ª ed. México: Progresso, 1970.

LANIER, J. Tele-Immersion: The Ultimate QoS-Critical Application, In: FIRST INTERNET2 JOINT APPLICATIONS/ ENGINEERING QOS WORKSHOP, Santa Clara, 1998.

LANIER, J. ZIMMERMAN, T.G. BLANCHARD, C. BRYSON,S. HARVILL, Y. A hand gesture interface device. CHI '87 Proceedings of the SIGCHI/GI

conference on Human factors in computing systems and graphics interface. New York, NY, USA 1987.

LANIER, J. BLANCHARD, C. BURGESS, S. LASKO, A. OBERMAN,M. TEITEL,M. Reality built for two: a virtual reality tool. N: I3D '90 PROCEEDINGS OF THE 1990 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS,1990, New York, USA. Proceedings… ACM, New York, NY, USA.1990.

LEIGH, J. RENAMBOT, L. DeFANTI, T. BROWN, M. HE, E. KRISHNAPRASAD, N. MEERASA, J. NAYAK, A. PARK, K. SINGH, R. VENKATARAMAN, S. ZHANG, C. LIVINGSTON, D. MCLAUGHLIN, M. An experimental OptIPuter architecture for data-intensive collaborative visualization. In: PROCEEDINGS OF THE WORKSHOP ON ADVANCED COLLABORATION ENVIRONMENTS, 2003, Seattle. Proceedings… Seattle, 2003, p. 22–24.

LIMA, R. G. Ferramentas computacionais para o ensino e aprendizagem a distância de estruturas de aço. 142 p. Dissertação (Mestrado em Engenharia de Estruturas) –Universidade Federal de Minas Gerais, Belo Horizonte, 2002.

LIN, K.; SON, J.; ROJAS, E. A pilot study of a 3d game environment for construction safety education. Journal of Information Technology in Construction – ITCon, vol. 16, pp. 69 - 84, 2011.

LOPES, J. Jean Piaget. Revista Nova Escola, ano XI, n. 95, 1996.

MACHADO, L. S. A realidade virtual em aplicações científicas. Dissertação(Mestrado). Ministério da Ciência e Tecnologia - Instituto Nacional de Pesquisas Espaciais. São José dos Campos, 1997

MÄNTYLÄ, M. An Introduction to Solid Modeling. Rockville: Computer Science Press, 1988.

MCCORMICK, B. H.; DEFANTI, T. A.; BROWN, M. D. Visualization in Scientific Computing. Computer Graphics (edição especial), v. 21, n. 6, nov. 1987.

MCLELLAN, H. Virtual realities. Handbook of research for educational communications and technology, Boston, Kluwer-Nijhoff Publishing, p. 457–487, 1996.

MENDES, N. P. R. SANTOS, E. T. Maquete virtual interativa: proposta de uma ferramenta de vendas para o mercado imobiliário brasileiro. V TIC 2011. Bahia, 2011. CD-ROM.

MIRANDA, J. C. Urbanismo e Espaços Virtuais – Divulgação e Discussão na Comunidade. Dissertação (Mestrado) - Universidade do Porto, 1999.

MIZRAHI, V. V. Treinamento em linguagem C++-Módulo 1 e 2. 2ª edição. São Paulo: Ed. Pretince Hall. 2006.

MORAES, R. É possível ser construtivista no ensino de ciências? Construtivismo e ensino de ciências – reflexões epistemológicas e metodológicas, Rio Grande do Sul, 2ª edição, p. 103 – 130, 2003.

NETTO, A. V.; MACHADO, L. S. M.; OLIVEIRA, M. C. F. Realidade virtual - definições, dispositivos e aplicações. Disponível em: www.di.ufpb.br/liliane/publicacoes/reic2002.pdf, Acesso em: 24.out..2011.

NEWTON, S.; LOWE, R. Using an analytics engine to understand the design and construction of domestic buildings, In: PROCEEDINGS OF RICS - CONSTRUCTION AND PROPERTY CONFERENCE - COBRA 2011, 2011, Manchester. Proceedings… Manchester, 2011,p. .410-419.

NIKOLIC, D.; JARUHAR, S.; MESSNER, J. I. An educational simulation in construction: the virtual construction simulator, In: PROCEEDINGS OF THE 2009 ASCE INTERNATIONAL WORKSHOP ON COMPUTING IN CIVIL ENGINEERING, 2009, Austin. Proceedings…, Austin, 2009, p. 633 – 642.

NIKOLIC, D.; LEE, S.; MESSNER, J. I.; ANUMBA, C. The virtual construction simulator: evaluating an educational simulation application for teaching construction management concepts, In: PROCEEDINGS OF THE CIB W78 2010-27TH INTERNATIONAL CONFERENCE , 2010, Cairo. Proceedings… Cairo, 2010, p. 1-8.

OLIVEIRA, M. C. F. de; MINGHIM, R. Uma introdução à Visualização Computacional. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA (JAI), 16., 1997, Brasília. Anais... Brasília: SBC, 1997. p. 85-131.

PAN, Z.; CHEOK, A.; YANG, H.; ZHU, J.; SHI, J. Virtual reality and mixed reality for virtual learning environments. Computer and Graphics, vol. 30, n.1, p. 20-28, 2006.

PANTELIDIS, V. Reasons to use virtual reality in education. Disponível em: <http://eastnet.educ.ecu.edu/ vr/reas.html>, Acesso: 2011.

PAUSCH, R.; PROFFIT, D.; WILLIAMS, G. Quantifying immersion in virtual reality, In: SIGGRAPH 97- COMPUTER GRAPHICS PROCEEDINGS - ANNUAL CONFERENCE SERIES, 1997, USA. Proceedings… USA, 1997. p.13–18.

PAUSCH, R.; VOGTLE, L.; CONWAY, M. One dimensional motion tailoring for the disabled: A user study. In: PROCEEDINGS OF ACM CHI 92 CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 1992, USA. Proceedings…USA, 1992. p.405-411.

PAUSCH, R.; CONWAY, M.; GOSSWEILER, R.; BURNETTE, T. Alice: A rapid prototyping system for building virtual environments. In: ADJUNCT PROCEEDINGS OF ACM CHI'94 HUMANFACTORS IN COMPUTING SYSTEMS CONFERENCE. 1994, New York. Proceedings…ACM, New York, 1994. p.295–296.

PIMENTEL, K.; TEIXEIRA, K. Virtual reality - through the new looking glass. 2.ed. New York, McGraw-Hill, 1995.

POMPEU R. C., Um estudo sobre ambientes virtuais de apoio ao ensino e aprendizagem de resistência dos materiais. 90 p. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 1999.

PRENSKY, M. Digital Natives, Nigital immigrants. On the Horizon, vol. 9, n. 5, pp.1- 6, 2001.

RAPOSO, A.; SOARES, A.; CARVALHO, F.; LOAIZA, M.; GATTASS, M. Virtual Reality Group at Tecgraf/PUC-Rio, SBC Journal on 3D Interactive Systems, vol. 2, n. 2, pp. 75 - 78, 2011.

REVISTA VEJA (a) Homepage. Disponível em: <http://veja.abril.com.br/111109/ reconstrucao-uma-carreira-p-174.shtml>. Acesso em: 15/01/2010.

REVISTA VEJA (b) Homepage. Disponível em: <http://veja.abril.com.br/noticia /economia/brasil-produz-19-milionarios-por-dia- aponta -revista -forbes>. Acesso em: 15/01/2012.

REZENDE, R. B. Uma visão sobre o uso de fôrmas e escoramentos utilizados em cidades de grande, médio e pequeno porte do Brasil e novas diretrizes normativas. 164 p., Dissertação (Mestrado) - Universidade Federal de Uberlândia, 2010.

RHEINGOLD, H. Virtual reality. New York: Touchstone, 1991.

RIBEIRO JUNIOR, V. J. Técnicas de otimização de ambientes virtuais extensos e suas aplicações. Monografia (Ciência da Computação). Universidade do Estado de Santa Catarina. Joinville, 2004.

ROMANEL, F. B. Jogo "Desafiando a produção": uma estratégia para disseminação dos conceitos da construção enxuta entre operários da construção civil. 155 p. Dissertação (Mestrado em Construção Civil) – Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 2009.

SAMPAIO, A.Z.; HENRIQUES, P.G.; MARTINS, O.P. Virtual Reality Technology Used in Civil Engineering Education, The Open Virtual Reality Journal,n. 2, p. 18-25, 2008.

SAMPAIO, A.Z.; CRUZ, C.O.; MARTINS, O. P. Didactic models in civil engineering education: virtual simulation of construction works. Disponível em: <http://cdn. intechweb.org/pdfs/12788.pdf> Acesso em 2012

SANDIN D., MARGOLIS T., GE J., GIRADO J., PETERKA T., DEFANTI T.: The varrier TM autostereoscopic virtual reality display. ACM Transactions on Graphics, vol. 24, n.3, 894–903, 2005.

SAWHNEY, A.; MUND, A.; KOCZENASZ, J. Internet-based interactive construction management learning system. Journal of Construction Education, Vol. 6, No. 3, p. 124-138, 2001.

SCHROEDER W. J.; AVILA, L. S.; HOFFMAN, W. Visualizing with VTK: a tutorial. IEEE Computer Graphics and Applications. IEEE, 2000.

SCHROEDER, W.; MARTIN, K.; LORENSEN, B. The Visualization Toolkit: an object-oriented approach to 3D graphics. 3rd. ed. Kitware, Inc, 2004.

SIGGRAPH home page. Disponível em: <http://www.siggraph.org>. Acesso em: 01 jan. de 2012.

SHERIF, A.; MEKKAWI, H. Developing a computer aided learning tool foor teaching construction engineering decision making. In: INTERNATIONAL CONFENCE ON COMPUTING AND DECISION MAKING IN CIVIL AND BUILDING ENGINEERING, 2006, Montreal. Proceedings… Montreal, Canada, 2006, p. 3986 – 3995.

SHIRATUDDIN, M. F.; FLETCHER, D. Utilizing 3d games development tool for architectural design in a virtual environment, In: 7TH INTERNATIONAL CONFERENCE ON CONSTRUCTION APPLICATIONS OF VIRTUAL REALITY – CONVR2007, 2007, Pensilvania, p. 20 – 27.

SOUZA, P. C.; WAZLAWICK, R. S.; HOFFMANN, A. B. Um ambiente construtivista em realidade virtual para aprendizagem em engenharia civil, Revista de ensino em engenharia, São Paulo, n.18, p. 24-30,1997.

SUTHERLAND, I. Sketchpad, A Man-Machine Graphical Communication System. Tese - Massachusetts Institute of Technology, New York, 1963.

SUTHERLAND, I.E. The Ultimate Display. In: Proceedings IFIP 65, 2, p. 506-508, p. 582-583, 1965.

SUTHERLAND, I. A head-mounted three dimensional display. In: Proceedings FJCC, 1968. Proceedings…Thompson Books, 1968. p. 757–764.

STUART, R. The design of virtual environments, McGraw-Hill, New York, 1996.

TACHI, S. Real-time remote robotics-toward networked telexistence. IEEE Computer Graphics and Applications, vol. 8, n.6, p. 6-9, 1998.

TACHI, S. Tele-existence and/or cybernetic interface studies in Japan. In: NASA CONFERENCE PUBLICATION, 1990, Santa Barbara. Proceedings… Santa Barbara: Engineering Foundation - NASA Ames Research Center, ONR, IBM Corp., 1990. p. 67-68

TACHI, S.; KAWAKAMI, N.; INAMI, M.; ZAITSU, Y. Mutual Telexistence System using Retro-Reflective Projection Technology, International Journal of Humanoid Robotics, vol.1, n.1, p.45-64, 2004.

TACHI, S.; KAWAKAMI, N.; NII, H.; WATANABE, K.; MINAMIZAWA, K. TELEsarPHONE: Mutual telexistence master-slave comunication system based

on retroreflective projection technology, Journal of Control, Measurement and System Integration, vol.1, n.5, p. 335-344, 2008.

TÉCHNE Homepage. Disponível em: <http://www.revistatechne.com.br/engenharia-civil/134/artigo89320-1.asp>. Acesso em: 25/03/2012.

TICE, S.; JACOBSON, L. VR in visualization, animation, and entertainment. In: Jacobson, L. (Ed.),CyberArts: Exploring art andtechnology. San Francisco, CA: Miller Freeman, 1992.

TRIBUNA DO NORTE Homepage. Disponível em: <http://tribunadonorte.com.br /noticia/engenharia-civil-e-a-mais-procurada-por-alunos-de-ct/215237>. Acesso em: 19/03/2012.

UNISANTA Homepage. Disponível em: <http://www.online.unisanta.br/2012/03-24/campus-4.htm>. Acesso em: 25/03/2012.

UPSON, C. et al. The Application Visualization System: a computational environment for Scientific Visualization. IEEE Computer Graphics & Applications, v. 9, n. 4, p. 30-42, 1989.

VELHO, L.; GOMES, J. Sistemas gráficos 3D. São Paulo: IMPA, 2001.

VIANNA, A. S. G.; MACHADO, L. S. Controle e Gerenciamento de Ambientes Reais Educacionais Através de Ambientes Virtuais, In: PROCEEDINGS INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION (ICECE2009), 2009, Buenos Aires. Proceedings..., Argentina. CD-ROM.

VINCE, J. Virtual reality systems. Cambridge, Addison-Wesley, 1995.

VINCE, J. Essential virtual reality fast: how to understand the techniques and potential of virtual reality. Berlin; Springer, 1998.

VISAD Home Page. Disponível em: <http://www.ssec.wisc.edu/~billh /visad.html>. Acesso em: 01 jan. 2012.

VTK homepage. Disponível em: <http://www.vtk.org.> Acesso em janeiro de 2012.

WEIMER, D.; GANAPATHY, S.K. A synthetic visual environment with hand gesturing and voice input. In: CHI '89 PROCEEDINGS OF THE SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS: WINGS FOR THE MIND, 1989, Austin, USA.Proceedings…1989.

WEISS, A. M. L.; CRUZ, M. L. R. M. A informática e os problemas escolares de aprendizagem. 3ª edição. DP&A editora, Rio de Janeiro, 2001.

WINN, W. D. A conceptual basis for educational applications of virtual reality. Human Interface Technology Laboratory Technical Report. Seattle, WA: Human Interface Technology Laboratory, 1993.

WIKIPEDIA homepage. Disponível em: <http://www.wikipedia.org>. Acesso em: 01 jan. de 2012.

YAZIG W, A técnica de edificar - revisada e atualizada. 10ªed. Editora Pini, 2009.

ZHAO, Q. 10 scientific problems in virtual reality. Communications of the ACM, vol. 54, no. 2, p.116 - 118, 2011.

ZIMMERMAN, T.G. LANIER, J. BLANCHARD, C. BRYSON, S. HARVIL, Y. A hand gesture interface device. In: CHI '87 PROCEEDINGS OF THE SIGCHI/GI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS AND GRAPHICS INTERFACE, 1987,New York, USA. Proceedings… New York, USA,. p.189-192.

APÊNDICE 1

AVALIAÇÃO DO AMBIENTE VIRTUAL PROTÓTIPO										
As informações serão mantidas em sigilo e utilizadas para melhorar o ambiente virtual										
idade:										
escolaridade:										
curso:										
	Relevância			Suficiência		Adequação				
						Sequência			Compreensão	
	Muito Importante	Importante	Não se aplica	Sim	Não	Boa	Regular	Ruim	Fácil	Difícil
Aceitação do ambiente virtual										
1. Facilidade de uso										
As informações dispostas nas telas de abertura são apresentadas de forma:										
A apresentação do canteiro de obras virtual é:										
A navegação é fácil?										
2. Atitude perante a simulação										
Você tem dificuldades de estudar através de ambientes virtuais? ( )Sim ( )Não Caso afirmativo, quais são as dificuldades?										

Você sentiu motivação ao utilizar o ambiente virtual? ( )Sim ( )Não Caso afirmativo, quanto?										
Você se dispõe a participar de novas metodologias de ensino e treinamento ?										
Qual sua posição quanto a utilização de Ambientes virtuais como estratégia de ensino? ( )Favorável ( )Desfavorável										
Observações										
	Relevância			Suficiência		Adequação				
	Muito Importante	Importante	Não se aplica	Sim	Não	Sequência			Compreensão	
						Boa	Regular	Ruim	Fácil	Difícil
3.Qualidade da instrução										
O seu conhecimento sobre o assunto antes de iniciar a simulação era:										
Como você classifica o aprendizado através do ambiente virtual?										
O conteúdo do ambiente virtual está claro?										

O conteúdo do ambiente virtual está objetivo?										
Você concorda com a ordem da sequência da simulação?										
Descreva de forma simplificada a sequência da atividade simulada:										

Requisitos de desempenho									
1. Tempo de resposta									
O tempo de resposta entre o ambiente virtual e o usuário é:									
2. Interatividade									
A interatividade do ambiente virtual é:									
O ambiente virtual é motivador?									
Qualidade visual									
O projeto visual é:									
A quantidade de informações apresentadas nas telas de apresentação é:									
A quantidade de informações apresentadas na legenda é:									
O vocabulário é:									
O tamanho do texto usado no ambiente virtual para as informações é:									
As imagens são:									
Você já experimentou alguma vez uma simulação sobre construção civil?									

APÊNDICE 2

Instalação do VTK versão 5.4

1. Proceder com o download e instalação do CMake através do endereço eletrônico: http://www.cmake.org/cmake/resources/software.html, conforme figura 1;

2. Proceder com o download do código fonte da versão do VTK em um diretório através do endereço eletrônico: http://www.vtk.org/VTK/resources/software.html;

3. Proceder com a execução do CMake conforme figura 2;

4. Proceder com a localização do diretório do código fonte e o diretório de saída CMake conforme figura 3. Definir *CMAKE_PREFIX_INSTALL* para um diretório;

5. Pressionar o botão configurar e esperar conforme figura 4. Depois pressionar o botão OK;

6. O diretório de saída CMake estará criado. Neste diretório está o arquivo de solução (VTK.sln). Duplo clique neste arquivo para abrir a solução no MS Visual Studio C++ (ver a figura 5);

7. Escolha a configuração release e selecione *ALL_BUILD* para o projeto (ver a figura 6). Esperar o término da compilação;

8. Selecionar *INSTALL* e build. Todos os arquivos do tipo header e os arquivos da biblioteca estarão localizados no *CMAKE_PREFIX_INSTALL* definido anteriormente (Ver a figura 4.7);

9. VTK será construído no caminho *PREFIX_INSTALL_CMAKE*;

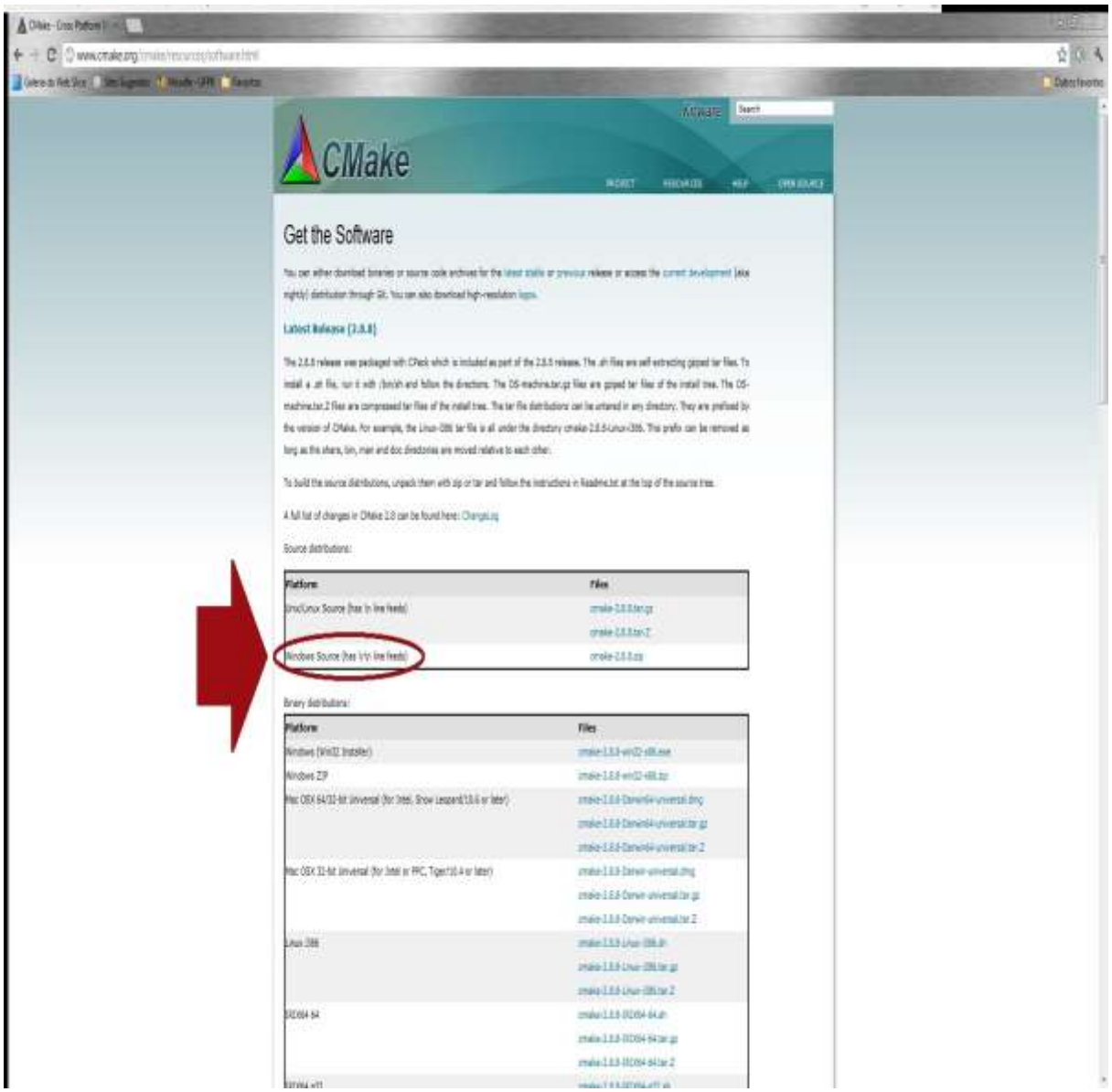

FIGURA 1: Página para fazer o download do CMake.
FONTE: Autor(2012).

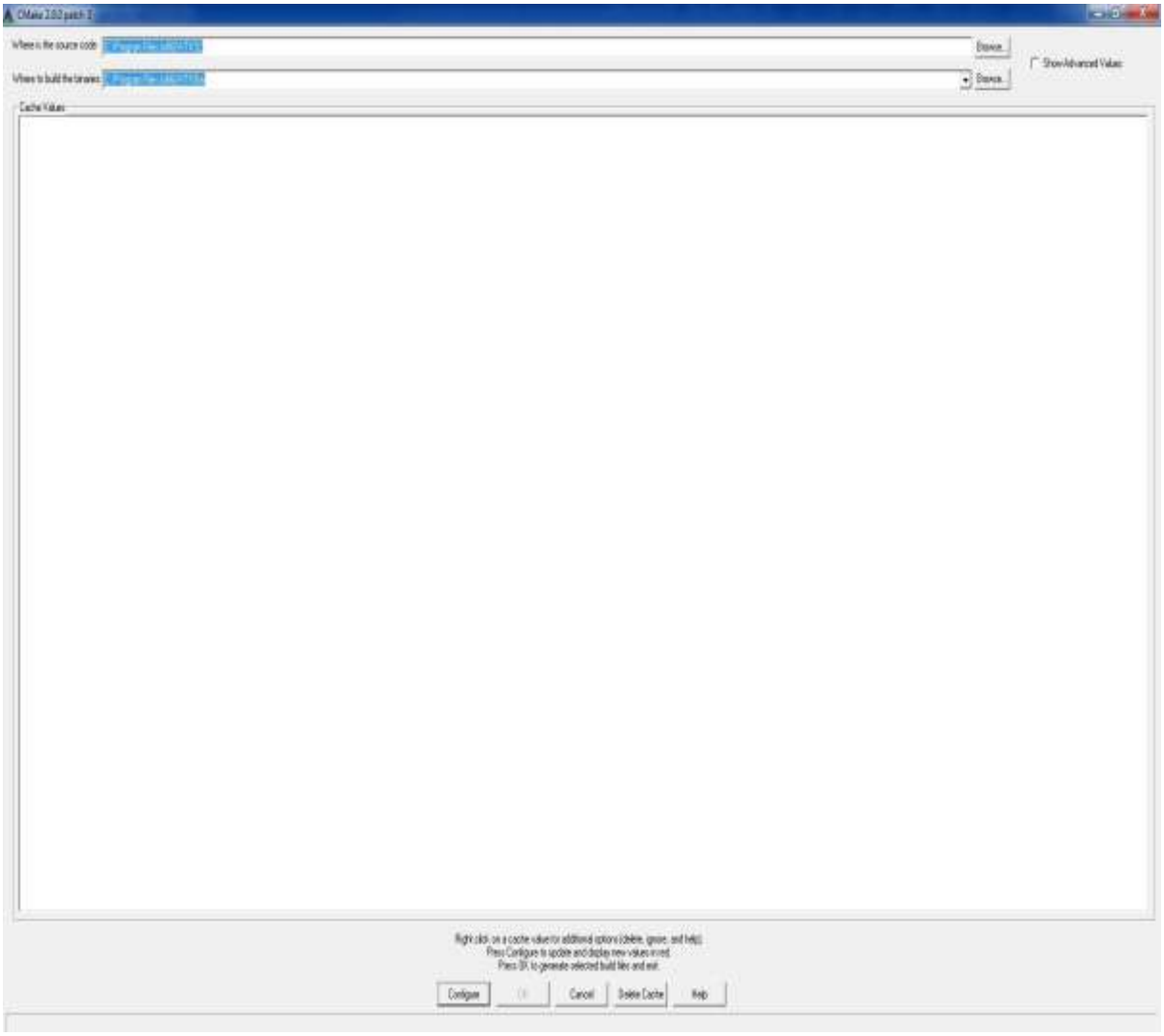

FIGURA 2: Execução do CMake.
FONTE: Autor(2012).

FIGURA 3: Localização do diretório do código fonte e o diretório de saída.
FONTE: Autor(2012).

FIGURA 4: Configuração.
FONTE: Autor(2012).

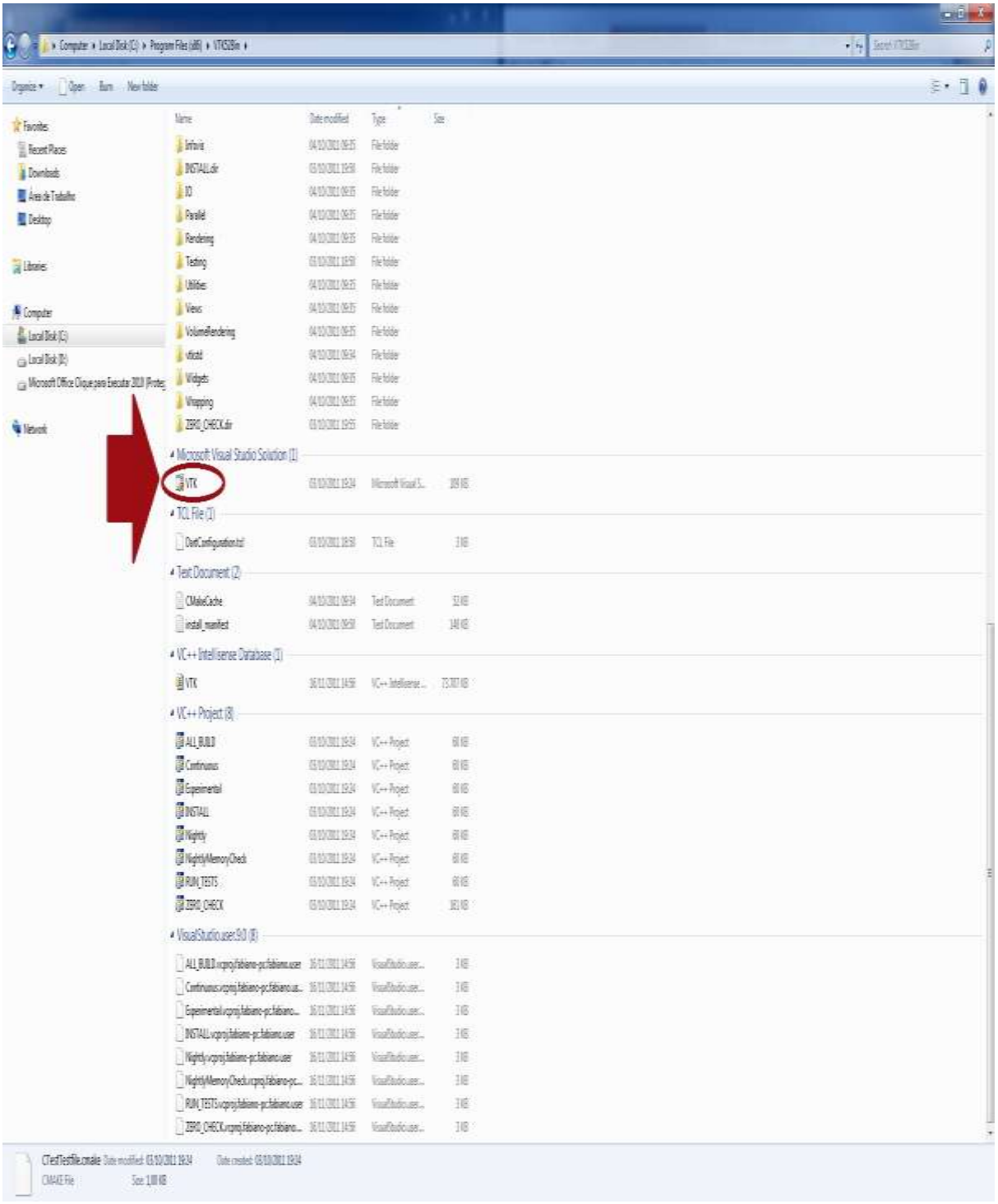

FIGURA 5: Arquivo de solução de projeto para o VTK no MS Visual Studio C++ .
FONTE: Autor(2012).

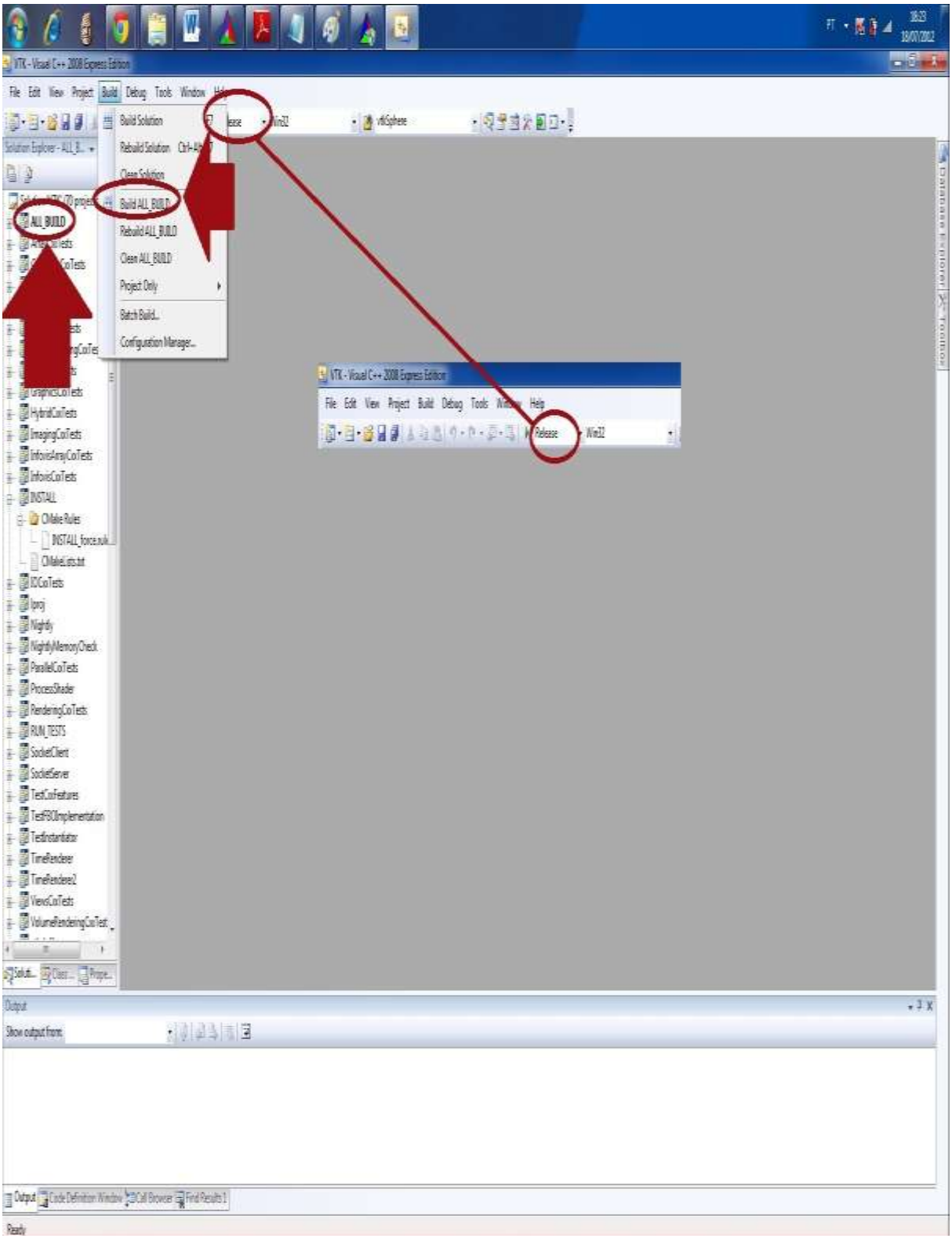

FIGURA 6: Configuração do MS Visual Studio C++ para construir o VTK.
FONTE: Autor(2012).

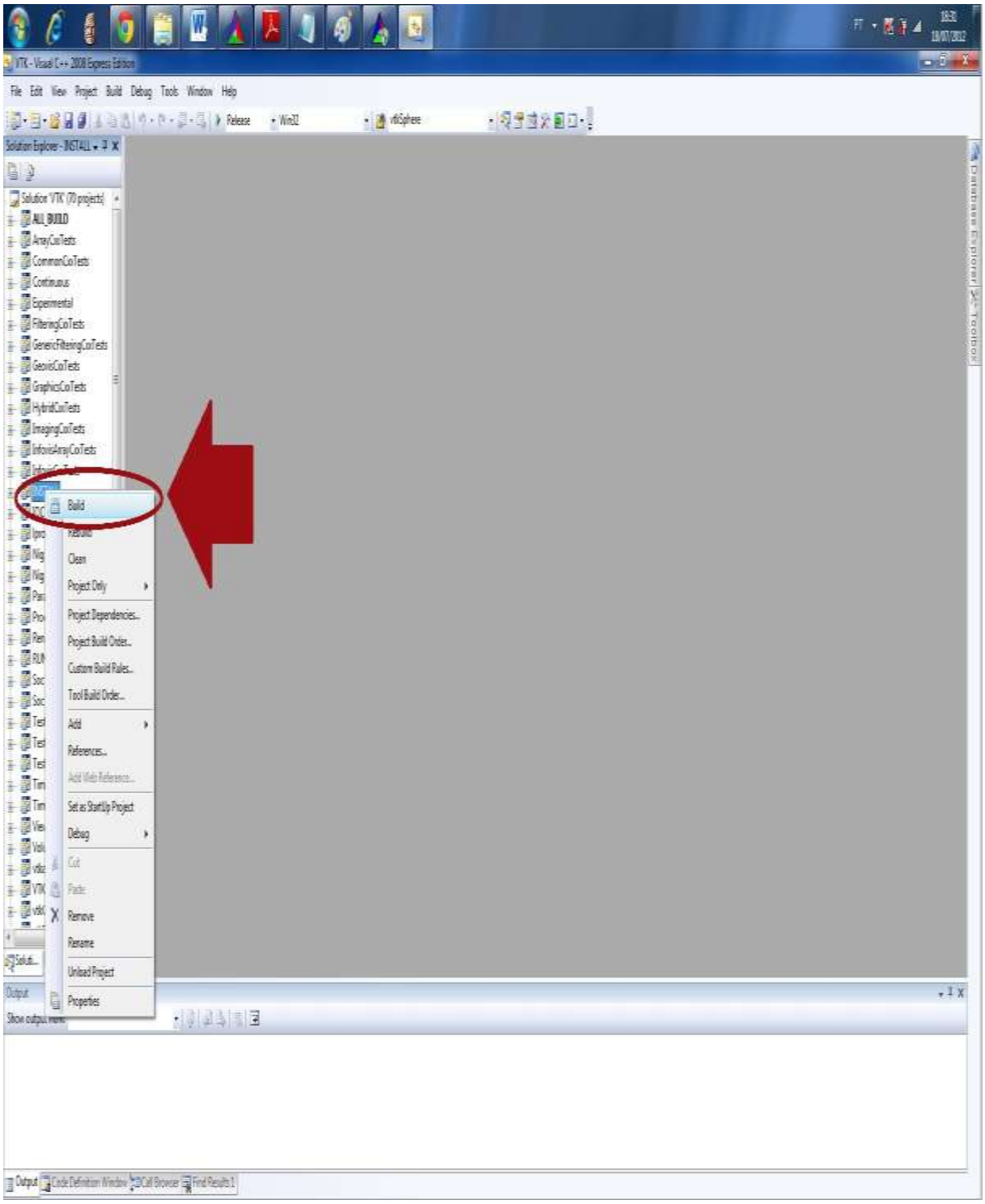

FIGURA 7: Arquivo *INSTALL* para construir o VTK.
FONTE: Autor(2012).

Após a construção do VTK, o MS Visual Studio C++ não sabe onde estão localizados os arquivos *headers* e os arquivos da biblioteca. Será necessário configurar as variáveis de ambiente dentro do MS Visual Studio C++ para mostrar como encontrar tais arquivos. O procedimento é descrito a seguir:

1. Localizar "*Tools*" na barra de *menu* e clique no botão "*Options*". (Veja a figura 8);

2. Configure o caminho no *include* dentro do diálogo (ver a figura 9);

3. Configure o caminho no *library* dentro do diálogo (ver a figura 9);

4. Utilize o comando *PRAGMA* caso ocorra algum erro quando o MS Visual Studio C++ não achar alguma biblioteca durante a compilação do aplicativo em VTK (ver a figura 10).

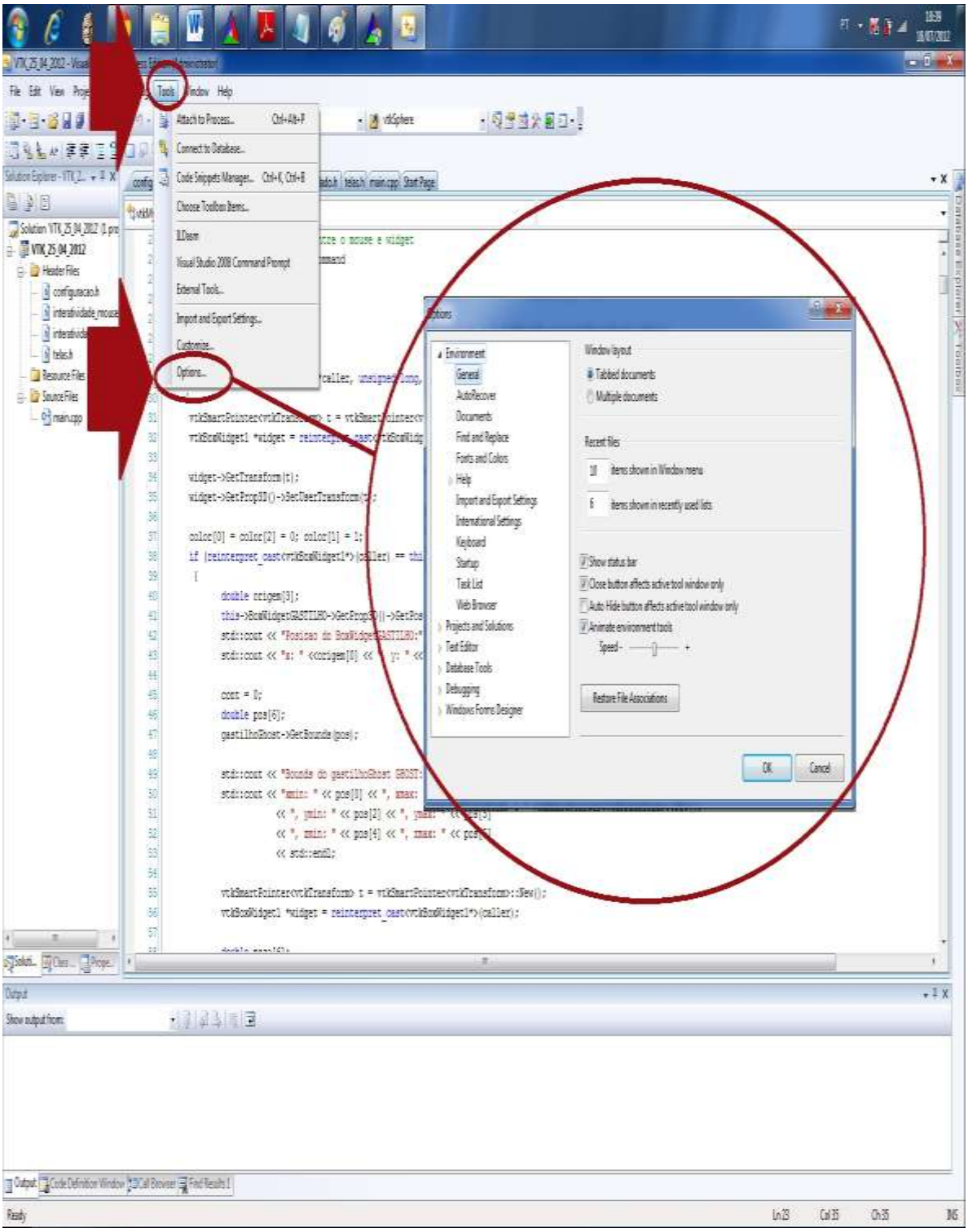

FIGURA 8: Botão tools no menu do MS Visual Studio C++.
FONTE: Autor(2012).

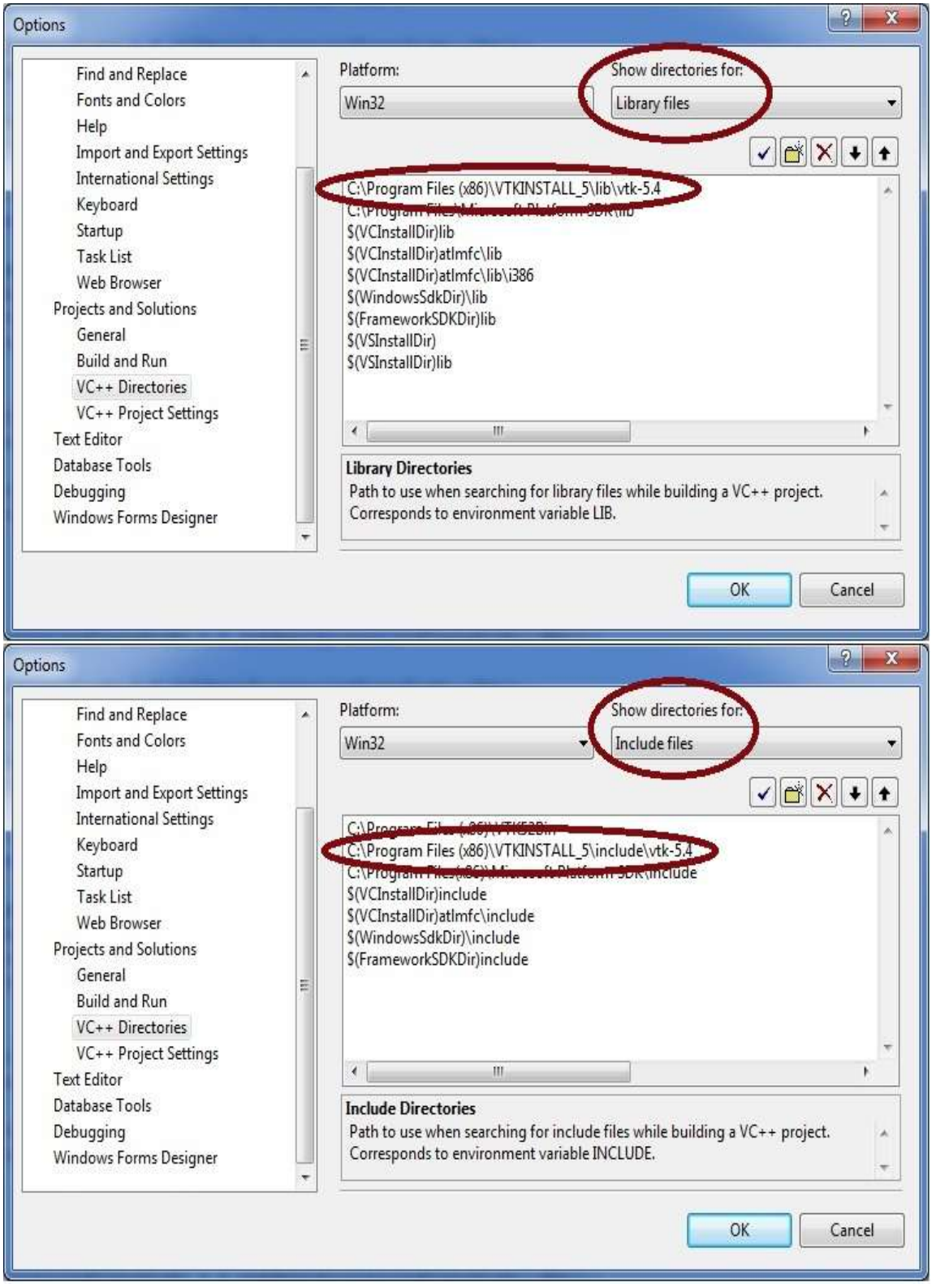

FIGURA 9: Configuração dos caminhos no MS Visual Studio C++.
FONTE: Autor(2012).

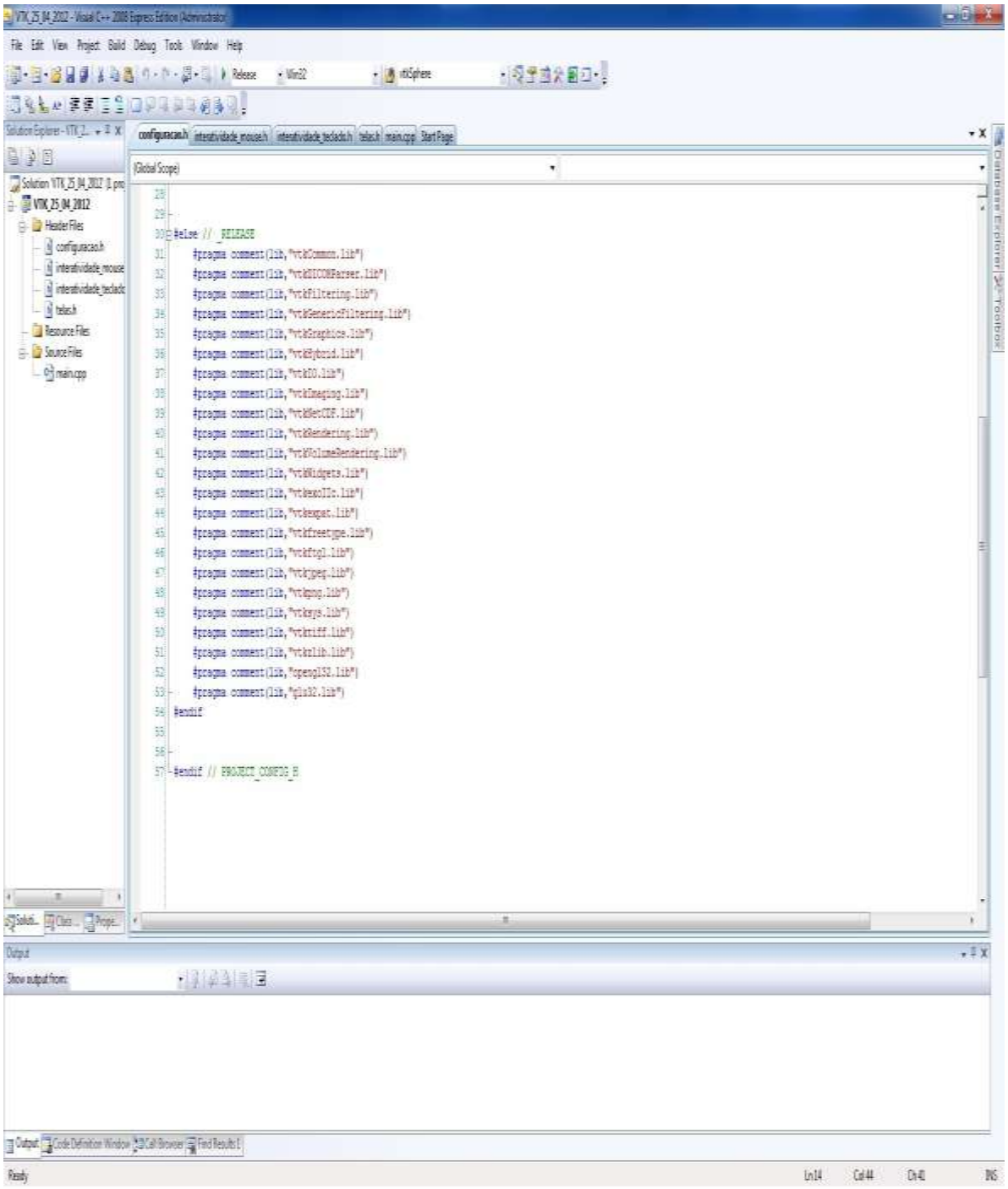

FIGURA 10: Utilização dos comando *PRAGMA*.
FONTE: Autor(2012).

APÊNDICE 3

Código da classe vtkBoxWidget1

```
class vtkBoxWidget1:public vtkBoxWidget
{
      public:
            static vtkBoxWidget1* New();
            virtual void corbox()
            {
                  this->HexActor->GetProperty()-
>SetRepresentationToSurface();

                  this->HexFace->VisibilityOff();
                  this->HexOutline->VisibilityOff();
                  this->PlaceWidget();
                  this->OutlineCursorWiresOn();
                  this->HandlesOff();
                  this->ScalingEnabledOff();
                  this->RotationEnabledOff();

            }

};
vtkStandardNewMacro(vtkBoxWidget1);
//Classe para gerenciar eventos entre o mouse e widget
```

Código da classe vtkMyCallback

```
class vtkMyCallback : public vtkCommand
{
public:
  static vtkMyCallback *New()
  { return new vtkMyCallback; }

  virtual void Execute(vtkObject *caller, unsigned long, void*)
    {
     vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
     vtkBoxWidget1 *widget = reinterpret_cast<vtkBoxWidget1*>(caller);

       widget->GetTransform(t);
       widget->GetProp3D()->SetUserTransform(t);

       color[0] = color[2] = 0; color[1] = 1;
       if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetGASTILHO)
        {
                  double origem[3];
                  this->BoxWidgetGASTILHO->GetProp3D()-
>GetPosition(origem);
                  std::cout << "Posicao do BoxWidgetGASTILHO:" <<
std::endl;
                  std::cout << "x: " <<origem[0] << ", y: " <<
origem[1] << ", z: " << origem[2] << std::endl;

                  cont = 0;
                double pos[6];
                  gastilhoGhost->GetBounds(pos);
```

```
                    std::cout << "Bounds do gastilhoGhost GHOST:" <<
std::endl;
                    std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                        << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                        << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                        << std::endl;

                    vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                    vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                    double posa[6];
                    widget->GetProp3D()->GetBounds(posa);
                          std::cout << "Bounds do gastilho OFF:" <<
std::endl;
                          std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                          << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                          << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                          << std::endl;


                    widget->GetTransform(t);
                    widget->GetProp3D()->SetUserTransform(t);
                    double position[3];
                    t->GetPosition(position);



                    std::cout << "Posicao do gastilho conforme t:" <<
std::endl;
                    std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;


                    if( (position[0] <-60) && (position[1] <-12) )
                    {
                          vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                          t1->Translate(-112.147, -42.8101, 05.2501);
                          widget->GetProp3D()->SetUserTransform(t1);
                          widget->EnabledOff();
                          gastilhoGhost->VisibilityOff();
                          pm2Ghost->VisibilityOn();
                          this->BoxWidgetPM2->On();
                          BoxWidgetPM2->corbox();
                          Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o primeiro painel             ", color);
                          Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que se encontra próximo            ", color);
                          Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "das ferramentas e posicione        ", color);
                          Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação em              ", color);
```

```
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                          ", color);
                  }


        }
      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetPM2)
        {

                double pos[6];
                  pm2Ghost->GetBounds(pos);

                  std::cout << "Bounds do pm2Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[3];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do pm2 OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[6];
                  t->GetPosition(position);

                  std::cout << "Posicao do pm2 conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -79) && (position[1] < -140) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-109.8598, -176.3434, 7.69319);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        pm2Ghost->VisibilityOff();
                        trava2Ghost->VisibilityOn();
```

```
                        this->BoxWidgetTRAVA2->On();
                        BoxWidgetTRAVA2->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a tábua que trava o           ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "painel que se encontra             ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximo da forma e                 ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "posicione conforme                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.              ", color);

                }


        }
      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTRAVA2)
        {

               double pos[6];
                 trava2Ghost->GetBounds(pos);

                 std::cout << "Bounds do trava2Ghost GHOST:" <<
std::endl;
                 std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                   << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                   << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                   << std::endl;

                 vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                 vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                 double posa[6];
                 widget->GetProp3D()->GetBounds(posa);
                       std::cout << "Bounds do trava2 OFF:" <<
std::endl;
                       std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                       << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                       << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                       << std::endl;

                 widget->GetTransform(t);
                 widget->GetProp3D()->SetUserTransform(t);

                 double position[3];
                 t->GetPosition(position);

                 std::cout << "Posicao do trava2 conforme t:" <<
std::endl;
                 std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
```

```
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] >28) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(48.68158, -11.2073, 0.75875);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        trava2Ghost->VisibilityOff();
                        trava4Ghost->VisibilityOn();
                        this->BoxWidgetTRAVA4->On();
                        BoxWidgetTRAVA4->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a tábua que trava o           ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "painel que se encontra             ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximo da forma e                 ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "posicione conforme                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.              ", color);
                  }


        }
      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTRAVA4)
        {

                double pos[6];
                  trava4Ghost->GetBounds(pos);

                  std::cout << "Bounds do trava4Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do trava4 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;
```

```
                    widget->GetTransform(t);
                    widget->GetProp3D()->SetUserTransform(t);

                    double position[3];
                    t->GetPosition(position);

                    std::cout << "Posicao do trava4 conforme t:" <<
std::endl;
                    std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                    std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                    if( (position[0] < -28))
                    {
                          vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                          t1->Translate(-52.0081, -4.7651, 0);
                          widget->GetProp3D()->SetUserTransform(t1);
                          widget->EnabledOff();
                          trava4Ghost->VisibilityOff();
                          p01Ghost->VisibilityOn();
                          this->BoxWidgetP01->On();
                          BoxWidgetP01->corbox();
                          Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o segundo  painel             ", color);
                          Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que se encontra próximo            ", color);
                          Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "das ferramentas e posicione        ", color);
                          Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação em              ", color);
                          Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                        ", color);

                    }


          }


      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetP01)
          {

                double pos[6];
                  p01Ghost->GetBounds(pos);

                    std::cout << "Bounds do p01Ghost GHOST:" <<
std::endl;
                    std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                        << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                        << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                        << std::endl;

                    vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
```

```
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do p01 OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do p01 conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -80) && (position[1] < -130) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-102.749, -176.546, 7.34072);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        p01Ghost->VisibilityOff();
                        p02Ghost->VisibilityOn();
                        this->BoxWidgetP02->On();
                        BoxWidgetP02->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o terceiro painel            ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que se encontra próximo           ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "dos tijolos e posicione       ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação em             ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                       ", color);
                  }


        }


      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetP02)
        {

                double pos[6];
                  p02Ghost->GetBounds(pos);
```

```
                  std::cout << "Bounds do p02Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do p02 OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do p02Ghost conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -85) && (position[1] < 0) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-109.836, -23.3241, 5.21935);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        p02Ghost->VisibilityOff();
                        ArmaduraGhost->VisibilityOn();
                        this->BoxWidgetARMADURA1->On();
                        BoxWidgetARMADURA1->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a armadura que se                ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "encontra no almoxarifado              ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "proximo das ferramentas               ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme                  ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.                 ", color);
```

```
                }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetARMADURA1)
        {

                double pos[6];
                  ArmaduraGhost->GetBounds(pos);

                  std::cout << "Bounds do ArmaduraGhost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do Armadura OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do Armadura conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -138)&& (position[1] < -115) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-158.7691, -142.9619, 11.3624);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        ArmaduraGhost->VisibilityOff();
                        pm1Ghost->VisibilityOn();
```

```
                        this->BoxWidgetPM1->On();
                        BoxWidgetPM1->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o terceiro painel             ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que se encontra próximo            ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "dos tijolos e posicione       ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação em              ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                        ", color);
                    }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetPM1)
        {

                double pos[6];
                  pm1Ghost->GetBounds(pos);

                  std::cout << "Bounds do pm1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do pm1 OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do pm1 conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
```

```
                    std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                    if( (position[0] < -80) && (position[1] < 0) )
                    {
                          vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                          t1->Translate(-109.9370711, -25.6599, 5.21635);
                          widget->GetProp3D()->SetUserTransform(t1);
                          widget->EnabledOff();
                          pm1Ghost->VisibilityOff();
                          trava1Ghost->VisibilityOn();
                          this->BoxWidgetTRAVA1->On();
                          BoxWidgetTRAVA1->corbox();
                          Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a tábua que trava o           ", color);
                          Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "painel que se encontra             ", color);
                          Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximo da forma e                 ", color);
                          Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "posicione conforme                 ", color);
                          Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.              ", color);
                    }


        }
            if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTRAVA1)
        {

                double pos[6];
                  trava1Ghost->GetBounds(pos);

                  std::cout << "Bounds do trava1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                          std::cout << "Bounds do trava1 OFF:" <<
std::endl;
                          std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                          << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                          << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                          << std::endl;
```

```
                widget->GetTransform(t);
                widget->GetProp3D()->SetUserTransform(t);

                double position[3];
                t->GetPosition(position);

                std::cout << "Posicao do trava1 conforme t:" <<
std::endl;
                std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                if( (position[0] < -27) )
                {
                      vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                      t1->Translate(-44.99725, 1, 0);
                      widget->GetProp3D()->SetUserTransform(t1);
                      widget->EnabledOff();
                      trava1Ghost->VisibilityOff();
                      trava3Ghost->VisibilityOn();
                      this->BoxWidgetTRAVA3->On();
                      BoxWidgetTRAVA3->corbox();
                      Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a tábua que trava o            ", color);
                      Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "painel que se encontra              ", color);
                      Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximo da forma e                  ", color);
                      Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "posicione conforme                  ", color);
                      Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.               ", color);
                }


        }
      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTRAVA3)
        {

              double pos[6];
                trava3Ghost->GetBounds(pos);

                std::cout << "Bounds do trava3Ghost GHOST:" <<
std::endl;
                std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                << std::endl;

                vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);
```

```
                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do trava3 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do trava3 conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -25) && (position[1] < 0)  )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-52.849682, -99.9118, 3.58999);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        trava3Ghost->VisibilityOff();
                        barratensor1Ghost->VisibilityOn();
                        this->BoxWidgetTENSOR1BARRA->On();
                        BoxWidgetTENSOR1BARRA->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o primeiro tirante            ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que esta  próximo da  pilha        ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "de tijolos e posicione             ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                        ", color);


                  }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR1BARRA)
        {

                double pos[6];
                  barratensor1Ghost->GetBounds(pos);
```

```
                    std::cout << "Bounds do barratensor1Ghost GHOST:" <<
std::endl;
                    std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                        << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                        << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                        << std::endl;

                    vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                    vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                    double posa[6];
                    widget->GetProp3D()->GetBounds(posa);
                          std::cout << "Bounds do BoxWidgetTENSOR1BARRA
OFF:" << std::endl;
                          std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                          << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                          << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                          << std::endl;

                    widget->GetTransform(t);
                    widget->GetProp3D()->SetUserTransform(t);

                    double position[3];
                    t->GetPosition(position);

                    std::cout << "Posicao do BoxWidgetTENSOR1BARRA
conforme t:" << std::endl;
                    std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                    std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                    if( (position[0] < -100) && (position[1] < -30)  )
                    {
                          vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                          t1->Translate(-124.959, -49.0291, 5.96115);
                          widget->GetProp3D()->SetUserTransform(t1);
                          widget->EnabledOff();
                          barratensor1Ghost->VisibilityOff();
                          tensor1Ghost->VisibilityOn();
                          this->BoxWidgetTENSOR1->On();
                          BoxWidgetTENSOR1->corbox();
                          Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava com os demais        ", color);
                          Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "tirantes que estão                ", color);
                          Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximos da pilha de tijolos      ", color);
                          Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme              ", color);
                          Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.             ", color);
```

```
                }


        }


      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR1)
        {

                double pos[6];
                  tensor1Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor1 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do tensor1G conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.956577, -50.2971848,
5.1281509);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
```

```
                        this->BoxWidgetTENSOR1FINAL->On();
                        BoxWidgetTENSOR1FINAL->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava que completa o          ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "travamento que está                  ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próxima da pilha de tijolos          ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.                ", color);

                  }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR1FINAL)
        {

                double pos[6];
                  tensor1Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor1 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do tensor1G conforme t:" <<
std::endl;
```

```
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.956577, -46.8388,
5.1281509);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        tensor1Ghost->VisibilityOff();
                        barratensor2Ghost->VisibilityOn();
                        this->BoxWidgetTENSOR2BARRA->On();
                        BoxWidgetTENSOR2BARRA->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o segundo  tirante            ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que esta  próximo da  pilha        ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "de tijolos e posicione             ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                        ", color);

                  }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR2BARRA)
        {

                double pos[6];
                  barratensor2Ghost->GetBounds(pos);

                  std::cout << "Bounds do barratensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do BoxWidgetTENSOR1BARRA
OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
```

```
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do BoxWidgetTENSOR1BARRA
conforme t:" << std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.959, -63.0321, 20.5683);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        barratensor2Ghost->VisibilityOff();
                        tensor2Ghost->VisibilityOn();
                        this->BoxWidgetTENSOR2->On();
                        BoxWidgetTENSOR2->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava com os demais        ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "tirantes que estão                ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximos da pilha de tijolos      ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme              ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.             ", color);
                  }


        }



      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR2)
        {

                double pos[6];
                  tensor2Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor2Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
```

```
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor2 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do tensor2G conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -95) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.952, -64.302, 19.7461);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        this->BoxWidgetTENSOR2FINAL->On();
                        BoxWidgetTENSOR2FINAL->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava que completa o          ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "travamento que está                  ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próxima da pilha de tijolos          ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.                ", color);
                  }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR2FINAL)
        {
```

```
                double pos[6];
                  tensor2Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor1 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do tensor1G conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-125, -60.5763, 19.7678);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        tensor2Ghost->VisibilityOff();
                        barratensor3Ghost->VisibilityOn();
                        this->BoxWidgetTENSOR3BARRA->On();
                        BoxWidgetTENSOR3BARRA->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache o terceiro tirante            ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "que esta  próximo da  pilha        ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "de tijolos e posicione             ", color);
```

```
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "conforme indicação                 ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "em magenta.                        ", color);
                    }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR3BARRA)
        {

                double pos[6];
                  barratensor3Ghost->GetBounds(pos);

                  std::cout << "Bounds do barratensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do BoxWidgetTENSOR1BARRA
OFF:" << std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do BoxWidgetTENSOR1BARRA
conforme t:" << std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.977, -77.0434, 32.2765);
```

```
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        barratensor3Ghost->VisibilityOff();
                        tensor3Ghost->VisibilityOn();
                        this->BoxWidgetTENSOR3->On();
                        BoxWidgetTENSOR3->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava com os demais        ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "tirantes que estão                ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próximos da pilha de tijolos      ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme              ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.             ", color);
                    }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR3)
        {

                double pos[6];
                  tensor3Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor3Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor3 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);
```

```
                  std::cout << "Posicao do tensor3G conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -99) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.954, -78.5173, 31.4437);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        this->BoxWidgetTENSOR3FINAL->On();
                        BoxWidgetTENSOR3FINAL->corbox();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Ache a trava que completa o         ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "travamento que está                 ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "próxima da pilha de tijolos         ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "e posicione conforme                ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "indicação em magenta.               ", color);
                  }


        }

      if (reinterpret_cast<vtkBoxWidget1*>(caller) == this-
>BoxWidgetTENSOR3FINAL)
        {

                double pos[6];
                  tensor3Ghost->GetBounds(pos);

                  std::cout << "Bounds do tensor1Ghost GHOST:" <<
std::endl;
                  std::cout << "xmin: " << pos[0] << ", xmax: " <<
pos[1]
                                    << ", ymin: " << pos[2] << ", ymax:
" << pos[3]
                                    << ", zmin: " << pos[4] << ", zmax:
" << pos[5]
                                    << std::endl;

                  vtkSmartPointer<vtkTransform> t =
vtkSmartPointer<vtkTransform>::New();
                  vtkBoxWidget1 *widget =
reinterpret_cast<vtkBoxWidget1*>(caller);

                  double posa[6];
                  widget->GetProp3D()->GetBounds(posa);
                        std::cout << "Bounds do tensor1 OFF:" <<
std::endl;
                        std::cout << "xmin: " << posa[0] << ", xmax: "
<< posa[1]
                        << ", ymin: " << posa[2] << ", ymax: " <<
posa[3]
```

```
                        << ", zmin: " << posa[4] << ", zmax: " <<
posa[5]
                        << std::endl;

                  widget->GetTransform(t);
                  widget->GetProp3D()->SetUserTransform(t);

                  double position[3];
                  t->GetPosition(position);

                  std::cout << "Posicao do tensor1G conforme t:" <<
std::endl;
                  std::cout << "x: " <<position[0] << ", y: " <<
position[1] << ", z: " << position[2] << std::endl;
                  std::cout << "dif: " << sqrt(pow(
(posa[0]/pos[0]),2)) << std::endl;

                  if( (position[0] < -100) )
                  {
                        vtkSmartPointer<vtkTransform> t1 =
vtkSmartPointer<vtkTransform>::New();
                        t1->Translate(-124.948, -74.6065, 31.4453);
                        widget->GetProp3D()->SetUserTransform(t1);
                        widget->EnabledOff();
                        tensor3Ghost->VisibilityOff();
                        Legend->SetEntry(1, static_cast<vtkPolyData *>
(NULL), "Pronto!                              ", color);
                        Legend->SetEntry(2, static_cast<vtkPolyData *>
(NULL), "Basta encaixar os tensores           ", color);
                        Legend->SetEntry(3, static_cast<vtkPolyData *>
(NULL), "nos tirantes e tensionar             ", color);
                        Legend->SetEntry(4, static_cast<vtkPolyData *>
(NULL), "para travar a fôrma.                  ", color);
                        Legend->SetEntry(5, static_cast<vtkPolyData *>
(NULL), "E verificar o prumo.                  ", color);

                  }


        }




    }

//-------------------------------------------------------------------
-------------------------------------------------------------------

  vtkMyCallback():PolyData(0),CursorActor(0) {}
  vtkPolyData  *PolyData;
  vtkActor        *CursorActor;

//-------------------------------------------------------------APONTA
PARA OS WIDGETS DA CENA-----------------------------------------------
---
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetP01;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetP02;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetPM1;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetPM2;
```

```
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetARMADURA1;

  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetGASTILHO;

  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR1;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR2;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR3;

  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR1FINAL;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR2FINAL;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR3FINAL;

  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR1BARRA;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR2BARRA;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTENSOR3BARRA;

  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTRAVA1;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTRAVA2;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTRAVA3;
  vtkSmartPointer<vtkBoxWidget1>          BoxWidgetTRAVA4;

//-------------------------------------------------------------APONTA
PARA OS GHOSTS DA CENA-------------------------------------------------
  vtkSmartPointer<vtkActor>                       ArmaduraGhost;

  vtkSmartPointer<vtkActor>                       p01Ghost;
  vtkSmartPointer<vtkActor>                       p02Ghost;
  vtkSmartPointer<vtkActor>                       pm1Ghost;
  vtkSmartPointer<vtkActor>                       pm2Ghost;

  vtkSmartPointer<vtkActor>                       gastilhoGhost;

  vtkSmartPointer<vtkActor>                       tensor1Ghost;
  vtkSmartPointer<vtkActor>                       tensor2Ghost;
  vtkSmartPointer<vtkActor>                       tensor3Ghost;


  vtkSmartPointer<vtkActor>                       barratensor1Ghost;
  vtkSmartPointer<vtkActor>                       barratensor2Ghost;
  vtkSmartPointer<vtkActor>                       barratensor3Ghost;

  vtkSmartPointer<vtkActor>                       trava1Ghost;
  vtkSmartPointer<vtkActor>                       trava2Ghost;
  vtkSmartPointer<vtkActor>                       trava3Ghost;
  vtkSmartPointer<vtkActor>                       trava4Ghost;


//-------------------------------------------------------------APONTA
PARA A LEGENDA DA CENA-------------------------------------------------
  vtkSmartPointer<vtkLegendBoxActor>     Legend;

  int cont;
  double color[3];


};
```

Includes usados no código

```
#include "configuracao.h"
#include <vtkSmartPointer.h>
#include <vtkBoxWidget.h>
#include <vtk3DWidget.h>
#include <vtkCamera.h>
#include <vtkConeSource.h>
#include <vtkSphereSource.h>
#include <vtkCubeSource.h>
#include <vtkPlaneSource.h>
#include <vtkInteractorStyleTrackballCamera.h>
#include <vtkPolyData.h>
#include <vtkPolyDataMapper.h>
#include <vtkPolyDataNormals.h>
#include <vtkRenderWindow.h>
#include <vtkRenderWindowInteractor.h>
#include <vtkRenderer.h>
#include <vtkTransform.h>
#include <vtkCellArray.h>
#include <vtkObjectFactory.h>
#include <vtkProperty.h>
#include <vtkPropPicker.h>
#include <vtkProp3D.h>
#include <vtkCommand.h>
#include <vtkCallbackCommand.h>
#include <vtkDataSet.h>
#include <vtkDataSetMapper.h>
#include <vtkAssembly.h>
#include <vtkActor.h>
#include <vtkActor2D.h>
#include <vtkImageActor.h>
#include <vtk3DSImporter.h>
#include <vtkOBJReader.h>
#include <vtkImageData.h>
#include <vtkImageMapper.h.>
#include <vtkJPEGReader.h>

#include <vtkTexture.h>
#include <vtkTextureMapToSphere.h>
#include <vtkTextureMapToCylinder.h>
#include <vtkTextureMapToPlane.h>
#include <vtkTransformTextureCoords.h>
#include <vtkTextActor.h>
#include <vtkTextProperty.h>
#include <vtkUnstructuredGrid.h>
#include <vtkCell.h>
#include <vtkCellArray.h>
#include <vtkIdList.h>
#include <vtkUnsignedCharArray.h>
#include <vtkPointData.h>
#include <vtkPoints.h>
#include <vtkLegendBoxActor.h>
#include <vtkLight.h>
#include <vtkLightActor.h>
#include <cmath>
```