

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**



**UMA APLICAÇÃO DAS TÉCNICAS DE REALIDADE VIRTUAL NA
VISUALIZAÇÃO E CORTE DE POLIEDROS NÃO-CONVEXOS**

ORIENTADOR: EDGARD LAMOUNIER JÚNIOR (PhD)

ORIENTANDA: MARIA EMÍLIA DA SILVA

**Fevereiro
2007**

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

**UMA APLICAÇÃO DAS TÉCNICAS DE REALIDADE VIRTUAL NA
VISUALIZAÇÃO E CORTE DE POLIEDROS NÃO-CONVEXOS**

Dissertação apresentada por Maria Emília da Silva à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciências.

Banca Examinadora:

- Professor Edgard Lamounier Jr., PhD. (Orientador)
- Professor Alexandre Cardoso, DSc. (Faculdade de Engenharia Elétrica)
- Professor Arquimedes Lopes, DSc. CEFET- GO - Centro Federal de Educação Tecnológica de Goiás

Uberlândia, fevereiro de 2007.

UMA APLICAÇÃO DAS TÉCNICAS DE REALIDADE VIRTUAL NA VISUALIZAÇÃO E CORTE DE POLIEDROS NÃO-CONVEXOS

MARIA EMÍLIA DA SILVA

Dissertação apresentada por Maria Emília da Silva à Universidade Federal de
Uberlândia como parte dos requisitos para obtenção do título de Mestre em
Ciências.

ProfºEdgard Afonso Lamounier Jr.
Orientador

ProfºDarizon Alves de Andrade
Coordenador do Curso de Pós-Graduação

AGRADECIMENTOS

Agradeço, primeiramente a Deus, por estar presente em todos os momentos da minha vida iluminando meus caminhos, fazendo-me perseverante na busca de meus ideais.

A professora Marilene Manara, pelas valiosas contribuições e incentivo para a realização desta pesquisa.

Ao professor e orientador Edgard Afonso Lamounier Júnior, pela orientação prestada neste trabalho.

Ao professor Kenedy Lopes Nogueira pela amizade, paciência e valiosa ajuda técnica prestada.

Aos colegas do Laboratório de Computação Gráfica: Marlene Roque, Luciano, Ezequiel, Wneiton pelo convívio, amizade e troca de experiências.

Aos meus pais pelo incentivo, carinho e compreensão.

Enfim, a todos os mestres, amigos ou conhecidos que de alguma forma me incentivaram, me apoiaram em diversas ocasiões.

RESUMO

SILVA, Maria Emília. UMA APLICAÇÃO DAS TÉCNICAS DE REALIDADE VIRTUAL NA VISUALIZAÇÃO E CORTE DE POLIEDROS NÃO-CONVEXOS, Uberlândia, Faculdade de Engenharia Elétrica- UFU, 2007.

Esta dissertação apresenta um software educacional que explora técnicas de Realidade Virtual (RV) através do desenvolvimento de ambientes virtuais educacionais. Utiliza-se a Geometria Espacial como estudo de caso. O objetivo deste trabalho é a extensão do corte para poliedros convexos e não-convexos. O ambiente virtual e as técnicas de interação propostas foram desenvolvidas utilizando-se a linguagem VRML (Virtual Reality Modeling Language) e JavaScript. Após disponibilizar o sistema para potenciais usuários, estes puderam realizar alguns experimentos e identificaram que as técnicas propostas auxiliam, de forma satisfatória, no processo de aprendizado da visualização de poliedros não-convexos, graças ao uso das técnicas de RV propostas.

Palavras-Chave: Realidade Virtual, VRML, JavaScript , Poliedro.

ABSTRACT

SILVA, Maria Emília . *APPLICATION OF VIRTUAL REALITY TECHNIQUES IN VISUALIZING AND CUTTING NON-CONVEX POLYHEDRON*, Uberlândia, Faculty of Electric Engineering - UFU, 2007.

This dissertation presents an educational software which explores Virtual Reality techniques (VR) through the development of virtual educational environments. 3D Geometry has been taken as a case study. The objective of this work is the extension of the cut for convex and non-convex polyhedron. The virtual environment and the interaction techniques have been developed using the VRML (Virtual Reality Modeling Language) and JavaScript languages. After making the system available to potential users, they were able to do some experiments and confirmed that the techniques proposed are satisfactory and helpful in the learning process and in the visualization of non-convex polyhedron.

Keywords : Virtual Reality, VRML, JavaScript, Polyhedron.

PUBLICAÇÕES

SILVA, Maria Emília; **LAMOUNIER**,Edgard Jr.; **CARDOSO**, Alexandre; **MANARA**, Marilene; **BORGES**, Eila C. Uso de Realidade Virtual na Visualização e Corte de Poliedros Não-Convexos. VII Symposium on Virtual Reality, Belém PA, maio/2006 – publicado.

SILVA, Maria Emília; **LAMOUNIER**,Edgard Jr.; **CARDOSO**, Alexandre; **MANARA**, Marilene; **BORGES**, Eila C. Realidade Virtual na Geometria. III Semana Acadêmica da Universidade Federal de Uberlândia, Uberlândia MG, agosto/2006 – publicado.

SUMÁRIO

1. INTRODUÇÃO.....	1
1.1 Motivação	1
1.2 Objetivos da Pesquisa.....	3
1.3 Realidade Virtual e Educação	3
1.4 Organização da Dissertação.....	7
2. TRABALHOS RELACIONADOS	8
2.1 Introdução	8
2.2 GEO 3D - RV no Ensino da Geometria Espacial.....	8
2.3 ATSWorlds.....	10
2.4 Protótipo LEGO	12
2.5 LVCE – Laboratório Virtual de Circuitos Elétricos	13
2.6 SEFIRV - Sistema de Experiências Físicas Instrucional em Realidade Virtual	15
2.7 LUDOS TOP.....	16
2.8 Cabri Geometry	18
2.9 Corte de Superfícies	19
2.10 Corte de Poliedros Convexos	21
2.11 Sumário e Conclusões.....	24
3. ARQUITETURA DO SISTEMA PROPOSTO.....	26
3.1 Introdução	26
3.2 VRML	26
3.3 JavaScript.....	27

3.4 Arquitetura do Sistema	28
3.4.1 Interface – GUI	29
3.4.2 Ambiente Virtual	30
3.4.3 Núcleo Gerenciador de Corte.....	31
3.5 Conclusões	31
 4. IMPLEMENTAÇÃO DO SISTEMA.....	 33
4.1 Introdução	33
4.2 Características do Ambiente Virtual	33
4.2.1 Ambiente Virtual	34
4.2.2 Painel de Controle	40
4.3 Outros Estudos de Caso:	58
4.4 Sumário e Conclusões:.....	60
 5. RESULTADOS E LIMITAÇÕES DO SISTEMA.....	 62
5.1 Introdução	62
5.2 Funcionamento do Sistema.....	62
5.3 Avaliação do sistema	65
5.4 Limitações do Sistema	72
5.5 Considerações Finais	73
 6. CONCLUSÃO E TRABALHOS FUTUROS	 74
6.1 Introdução	74
6.2 Conclusões	74
6.3 Trabalhos Futuros.....	75
 7. REFERÊNCIAS BIBLIOGRÁFICAS:	 77

LISTA DE FIGURAS

Figura 1.1: Imersão em RV [14].....	4
Figura 2. 1: GEO 3 D - Teoria de poliedros	9
Figura 2. 2: Sistema Geo 3D	10
Figura 2. 3: Geração de sólido pela técnica sweep.....	11
Figura 2. 4: Página inicial do ambiente de simulação	14
Figura 2. 5: Simulação.....	14
Figura 2. 6: Experimento Movimento Retilíneo Uniforme	15
Figura 2. 7: Ambiente do jogo [24]	17
Figura 2. 8: Ganhou pelo atributo cor [24]	18
Figura 2. 9: Construção com o uso do Cabri	19
Figura 2. 10: Plano de corte no parabolóide.....	20
Figura 2. 11: Corte no parabolóide.....	20
Figura 2. 12: Interface do cubo.....	21
Figura 2. 13: Plano de corte no cubo	22
Figura 2. 14: Corte no cubo	22
Figura 3. 1: Arquitetura do sistema	29
Figura 3. 2: GUI – Interface	30
Figura 4. 1: Bloco principal do Ambiente Virtual.....	33
Figura 4. 3: Interface do poliedro L.....	35
Figura 4. 4: Alerta - Pontos colineares	37
Figura 4. 5: Escolha de três pontos na diagonal do cubo	37
Figura 4. 6: Mensagem de pontos colineares e.....	38
Figura 4. 7 : Posição de 3 pontos A, B e C alinhados e.....	39
Figura 4. 8: Posição de 3 pontos A, B e C alinhados	39
Figura 4. 9: Pontos não-colineares	40
Figura 4. 10: Centróide.....	43
Figura 4. 11: Plano de corte.....	46
Figura 4. 12: Área externa	47
Figura 4. 13: Ponto médio interno	48
Figura 4. 14: Plano passa na região convexa.....	50
Figura 4. 15: Corte gerando dois poliedros	50
Figura 4. 16: Ponto médio externo E.....	51
Figura 4. 17: Momento do corte	56
Figura 4.18: Sólidos separados.....	57
Figura 4. 19: Traçado do plano para o sólido E.....	58
Figura 4. 20: Corte efetuado para o sólido E.....	58
Figura 4. 21: Traçado do plano para o sólido C	59
Figura 4. 22: Corte efetuado para o sólido C.....	59
Figura 4. 23: Traçado do plano para o sólido T.....	60
Figura 4. 24: Corte efetuado para o sólido T.....	60

Figura 5. 1: Página HTML inicial do sistema	63
Figura 5. 2: Poliedro não-convexo	64
Figura 5. 3: Plano de corte no sólido C.....	64
Figura 5. 4: Corte efetuado para o sólido C.....	65
Figura 5. 5: Avaliação do sistema quanto a finalidade do seu uso como ferramenta educacional.....	66
Figura 5. 6: Avaliação do sistema quanto à interface com o Usuário.	67
Figura 5. 7: Avaliação quanto a facilidade de uso do sistema.....	68
Figura 5. 8: Avaliação quanto aos recursos do programa.	69
Figura 5. 9: Avaliação quanto aos objetivos disponíveis.	70
Figura 5. 10: Avaliação das informações úteis sobre Geometria Espacial.....	71

CAPÍTULO I

1. Introdução

1.1 Motivação

A evolução tecnológica verificada em diversas áreas nos últimos anos, permitiu o aparecimento de uma nova tecnologia, a Realidade Virtual (RV), que, suportando a criação de ambientes virtuais para aplicação em diversas áreas, começa a assumir um papel de relevo cada vez maior tanto na academia quanto na indústria [1].

A Realidade Virtual permite ainda que se visite lugares em períodos diferentes de tempo e com uma rapidez tão grande que sem ela seria impossível fazê-lo em uma vida toda. Pode-se, por exemplo, ver o mundo hoje e em um instante viajar 10 mil anos no passado e ver como era naquela época [2].

A potencialidade da Realidade Virtual está exatamente no fato de permitir que se explore alguns ambientes, processos ou objetos, não através de livros, fotos, filmes ou aulas, mas através da manipulação e análise virtual do próprio alvo do estudo [2].

Hoje a informação viaja pelo ciberespaço de forma quase imperceptível chegando, em tempo real, a todo o mundo. Mais do que nunca informar, que deveria ser também sinônimo de educar, exige dos cidadãos responsabilidades acrescidas já que as suas produções passarão a fazer parte de um vasto universo informativo, navegado por múltiplos utilizadores [1].

Além disso, a Realidade Virtual possui, um grande potencial educativo e, desde que corretamente utilizada, poderá vir a tornar-se num instrumento de ensino/aprendizagem versátil e de grande eficácia que, certamente, encontrará o seu lugar, de destaque, na ainda pouco quantidade de ferramentas educativas para o ensino do futuro [1].

Atualmente, com a crescente disseminação de informação pela Internet, o desenvolvimento de ambientes virtuais que permitam uma integração entre os estudantes através da Web, pode permitir a estudantes, geograficamente dispersos, aprender de forma cooperativa, sem estarem presentes à mesma aula [3]. Nos últimos anos em relação à modelagem 3D de um sistema virtual, novas técnicas estão sendo investigadas [4] e estão sendo desenvolvidos aplicativos para a Internet [5,6].

Uma das áreas a ser beneficiada pela RV é a área educacional, não só porque apresenta um enorme potencial para o desenvolvimento de pesquisas que se utilizam da Realidade Virtual, mas também porque permite que usuários experimentem sensações únicas com manipulação e análise do objeto de estudo [7].

Observa-se que o uso dos laboratórios virtuais como suporte à interação pode ser mais eficiente, conveniente e até mais barato do que métodos tradicionais [8], já que ferramentas gratuitas podem ser utilizadas nas experiências. Estes fatores têm motivado o desenvolvimento de diversos ambientes virtuais 3D educacionais [9].

Atualmente, muitos sistemas restringem-se a serem apresentadores de simulações, obrigando o usuário a ficar apenas observando os ambientes virtuais. Ou ainda então limitados à criação e visualização de objetos, não permitindo também a interação direta do usuário com os objetos após a sua criação no mundo virtual, a exemplo dos sistemas voltados para o ensino de Geometria Espacial onde a interatividade requerida não é fortemente explorada [10].

Essa não exploração da interatividade do sistema com o usuário dificulta a fundamentação de um dos principais requisitos para as aplicações de Realidade Virtual [11] e vêm-se com a necessidade de direcionar as pesquisas no sentido de explorar técnicas de RV com ambientes mais interativos.

1.2 Objetivos da Pesquisa

O objetivo deste trabalho é investigar, através do desenvolvimento de um sistema protótipo, o uso de técnicas de Realidade Virtual que permitem um maior grau de interação do aluno com o ambiente virtual, tendo como estudo de caso a Geometria Espacial, com a extensão de corte para poliedros não-convexos. Para atingir os objetivos desejados as seguintes metas foram definidas:

- Identificar as vantagens e desvantagens de sistemas computacionais construídos através de técnicas de Realidade Virtual que são utilizadas no processo de ensino/aprendizagem.
- Implementar um sistema protótipo, com o objetivo de analisar as técnicas de interação propostas.
- Investigar ferramentas computacionais que melhor suportem a navegação e interação em ambientes virtuais para serem empregadas na visualização para o ensino de Geometria Espacial.
- Analisar o sistema junto a potenciais usuários.

1.3 Realidade Virtual e Educação

A Educação pode ser vista como um processo de descoberta, exploração e de observação, além de eterna construção do conhecimento. Diante disso, as características específicas da Realidade Virtual podem transformá-la num poderoso instrumento a serviço de todos que buscam a evolução da educação. Muitas coisas que até pouco tempo atrás eram sonhos, atualmente, com os avanços tecnológicos existentes torna-se uma realidade: uma Realidade Virtual [12].

Esta nova tecnologia se baseia em três pilares: imersão, interação e envolvimento [13]. A **imersão** é a sensação de estar inserido no contexto do ambiente virtual permitindo ao participante ver, ouvir e tocar objetos que não

existem neste ambiente. Na **interação** o usuário pode navegar livremente no ambiente alterando-o em tempo real por meio de simulações, ou por meio de algum mecanismo que modifique as imagens do ambiente. A idéia de **envolvimento** está ligada com a emoção do usuário, que pode ser a leitura de um livro, considerado um envolvimento passivo ou participar de um jogo, ou realizar simulações considerado um envolvimento ativo.

A Realidade Virtual pode ser imersiva ou não-imersiva. A RV imersiva é baseada no uso dos capacetes ou salas de projeção, enquanto que a não-imersiva consiste no uso de monitores, chamada também de interface tradicional, por ser uma forma bidimensional. A imersão física dentro da imagem, é realizada pela utilização de capacetes estereoscópicos ou de lasers retinianos, que procuram a sensação de "se passar através do espelho". Veja Figura 1.1:



Figura 1.1: Imersão em RV [14]

Com a Realidade Virtual presente na Educação pode-se descobrir, explorar e construir conhecimento (aprender) sobre lugares onde não se pensa em visitar. O grande potencial da Realidade Virtual está exatamente nessas possibilidades, não só

através de aulas ou objetos físicos, mas também através da manipulação virtual do alvo a ser explorado, analisado e estudado [12].

Na Realidade Virtual pode-se explorar os diferentes estilos de aprendizagem, uns verbais, outros visuais, outros dedutivos, etc. Assim sendo, permite que se crie ambientes personalizados onde se respeite o tempo de aprendizagem de cada um, seja pela familiarização dos conteúdos ou com a rotina dos equipamentos, na medida em que se caminha pelo mundo.

Ao abandonar-se a tradicional concepção de ensino que considera o professor como o mestre que tudo sabe e os alunos como receptáculos vazios que o professor vai enchendo, pouco a pouco, com o seu saber, por um conceito de ensino dinâmico, verdadeiramente exploratório, em que o aluno constrói o seu próprio conhecimento, o professor torna-se, mais que o sábio, um guia para os seus alunos. Com a RV este novo papel acentua-se e o professor torna-se um colega, um outro elemento dentro do grupo; ele é o facilitador da aprendizagem dos alunos, um igual que se coloca ao lado dos seus educandos, trabalhando com eles, sentindo as mesmas dificuldades e alegrias, triunfos e fracassos. Trata-se, no fundo, da defesa de uma concepção de ensino e não será difícil de verificar, um dia, que os professores mais entusiasmados com a RV serão os que, já hoje, defendem a mudança educativa [1].

A construção de ambientes de aprendizagem ricos possibilitam a criação de aplicações interdisciplinares passíveis de serem utilizadas por diferentes professores, em disciplinas curriculares específicas ou em projetos colaborativos que envolvem o tratamento de dados de diversas origens e diferentes espécies. Este gênero de experiências suscita efeitos benéficos significativos: redução dos custos, desenvolvimento de atitudes de entre-ajuda e solidariedade entre os docentes, facilitação do envolvimento dos alunos [1].

Portanto, os limites que a RV possui no meio educacional são os limites dos seus criadores. A Realidade Virtual veio para traçar um novo paradigma na área

educacional, pois ajuda delinear uma nova postura de educação, diferente da tradicional.

Neste sentido, a RV propicia as seguintes situações [12]:

- Maior motivação dos estudantes (usuários);
- O poder de ilustração da Realidade Virtual para alguns processos e objetos é muito maior do que outras mídias;
- Permite uma análise de muito perto;
- Permite uma análise de muito longe;
- Permite que as pessoas deficientes realizem tarefas que de outra forma não são possíveis;
- Dá oportunidades para experiências;
- Permite que o aprendiz desenvolva o trabalho no seu próprio ritmo;
- Não restringe o prosseguimento de experiências ao período da aula regular;
- Permite que haja interação, e desta forma estimula a participação ativa do estudante.

A potencialidade da Realidade Virtual então, está exatamente no fato de permitir a exploração de ambientes, processos ou objetos, através da manipulação e análise virtual do próprio alvo de estudo, permitindo que se faça experiências com o conhecimento de forma interativa. O seu maior diferencial diante de outras tecnologias (TV, fotos, multimídia) é permitir que um estudante aprenda sobre um assunto, inserido no contexto deste assunto, e assim, receber a cada ação que fizer, uma realimentação deste contexto [15].

Esta dissertação tem por objetivo principal mostrar a potencialidade do uso da Realidade Virtual na Educação como uma ferramenta de apoio com fortes recursos de interação e imersão.

As pesquisas que tratam do emprego da RV na educação mostram que a informação aprendida é armazenada por muito mais tempo quando o aluno é um participante ativo do processo de aprendizado e quando a apresentação do objeto em estudo envolve muitos dos sentidos dos alunos [16].

1.4 Organização da Dissertação

Este texto está dividido em seis capítulos a saber: O Capítulo 1 inclui esta Introdução.

O Capítulo 2 faz uma avaliação de vários sistemas computacionais relacionados com as técnicas de Realidade Virtual. Verifica as principais contribuições e limitações bem como as possibilidades de interação com os mesmos.

O Capítulo 3 descreve a arquitetura do sistema proposto, os quais possibilitam a visualização, manipulação e o corte dos poliedros não-convexos. Ressalta a importância das linguagens utilizadas que são: JavaScript e VRML .

A linguagem JavaScript foi criada pela Netscape com o objetivo de obter uma linguagem intermediária entre a linguagem Java e HTML. Associada a VRML permite a interação no sistema. A VRML (*Virtual Reality Modeling Language*) é uma linguagem de descrição de cenas 3D com vários participantes se comunicando através da Internet.

O Capítulo 4 relata os detalhes da implementação, bem como os algoritmos matemáticos utilizados no desenvolvimento deste software.

O Capítulo 5 mostra o funcionamento e as limitações do sistema protótipo. O Capítulo 6 apresenta as conclusões do sistema e os trabalhos futuros. Logo após apresenta-se as referências bibliográficas que serviram de fundamentação teórica para esta pesquisa.

CAPÍTULO II

2. TRABALHOS RELACIONADOS

2.1 Introdução

As técnicas de RV têm sido desenvolvidas e aplicadas nos mais diversos campos de ensino. Na educação, a tecnologia está impulsionando essa área para novos rumos, enfatizando a utilização de novas ferramentas, propiciando uma evolução no processo de ensino/aprendizagem [12].

Neste capítulo, são apresentados os vários sistemas computacionais relacionados com a área educacional e desenvolvidos a partir de técnicas de Realidade Virtual. Os sistemas apresentados aqui foram desenvolvidos a partir da linguagem VRML, amplamente utilizada em ambientes 3D para Internet.

Apresenta-se as técnicas desenvolvidas nestes trabalhos, ressaltando suas contribuições e os avanços da interação proposta pelos ambientes de aprendizagem.

2.2 GEO 3D - RV no Ensino da Geometria Espacial

Geo 3D é um software educacional implementado em um ambiente de RV para o ensino, desenvolvido pelo Grupo de Pesquisa de RV do departamento de Informática UNISC (Universidade de Santa Cruz do Sul - RS) [17].

O objetivo básico do programa é possibilitar um novo processo de aprendizagem de Geometria Espacial através da interação, descoberta, exploração e observação de sólidos geométricos, dando suporte aos métodos tradicionais de ensino.

O software foi desenvolvido para ser usado em rede através de navegadores compatíveis com JavaScript e VRML. Com a integração de JavaScript, VRML e HTML, apresenta páginas interativas, onde aprendizes podem navegar e testar seus conhecimentos da área da Geometria Espacial, denominada de Poliedros.

O sistema protótipo é composto por uma tela do ambiente, dividida em 2 *frames* ou quadros. No quadro da esquerda, está representado a sala de aula tradicional, onde se faz a apresentação de uma aula expositiva. No quadro da direita, apresenta os ambientes virtuais do projeto.

A metodologia utilizada foi baseada em sistemas [18] em que o computador é usado para controlar e acompanhar a apresentação de determinada quantidade de informações ao aluno, dispondo de interação e na qual solicitações do computador são respondidas pelo aluno. Com base nestas respostas, novas informações são fornecidas, que varia de acordo com o desempenho do aluno.

A ferramenta disponibiliza a apresentação teórica ao aluno. Por exemplo, para o conteúdo Poliedros, Figura 2.1. Uma série de exercícios para teste de conhecimento, com respostas é então apresentado (Figura 2.2).



Figura 2. 1: GEO 3 D - Teoria de poliedros

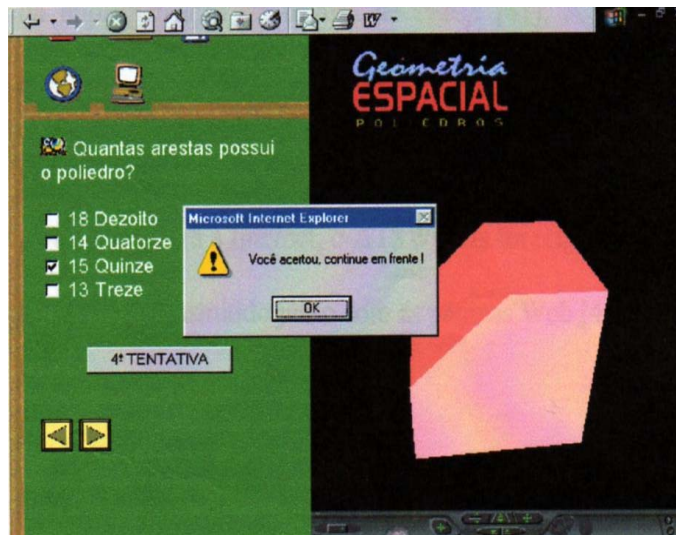


Figura 2. 2: Sistema Geo 3D

A utilização do programa se dá com o acesso através da Internet, onde se encontra a página inicial. Após selecionar “Iniciar”, o usuário encontra uma segunda tela onde pode começar a usar o programa para aprender e exercitar conhecimentos de geometria espacial. É suportado pelos navegadores tradicionais e pelos mais diversos sistemas operacionais.

O protótipo acima apresenta limitações como: Os ambientes virtuais são previamente definidos, portanto os passos do aprendizado já estão previamente determinados, desestimulando a criatividade e a falta de interação com os objetos do ambiente virtual.

2.3 ATSWorlds

ATSWorlds é um sistema desenvolvido na Unisinos (Universidade do Vale do Rio Sinos), para o ensino de Geometria (superfícies de revolução), com a finalidade de criar objetos 3D modelados através da técnica de varredura (sweep) [19].

Sweep é uma técnica de criar sólidos de revolução, onde a partir de um contorno descrito em 2D realiza-se um deslocamento e à medida que este contorno descreve uma trajetória no espaço 3D sólidos de revolução são gerados, de forma que vai compondo um objeto [19].

Este sistema foi criado usando VRML para que possa funcionar pela Internet e ser transmitido junto com páginas Web.

O sistema ATSWorlds permite que um usuário, usando um applet Java, defina a linha de modelagem e visualize o objeto final em 3D, em uma janela, dentro da mesma página. O applet permite alterar a definição da linha e coordena a visualização de novas formas na janela de apresentação do objeto gerado por sweep [19]. A área em branco é o local onde são desenhados os polígonos (2D) que gerarão os objetos pela técnica de sweep. Para desenhar um polígono, basta clicar com o mouse em pontos desta área, demarcando os pontos que definem os vértices deste.

Escolhido o tipo de sweep e ajustados os parâmetros necessários, após feito o desenho basta clicar no botão **Atualizar** para que o ATSWorlds atualize o objeto na parte superior direita da página (Figura 2.3).

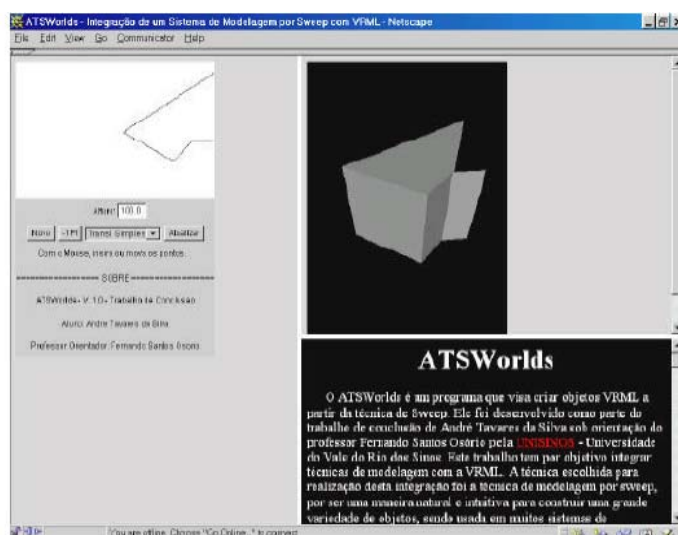


Figura 2. 3: Geração de sólido pela técnica sweep

O botão Atualizar altera o polígono gerador e deve ser usado toda vez que o usuário deseja alterar o tipo de sweep, os parâmetros ou quando quiser visualizar a partir das mudanças feitas.

A aplicação deste sistema está em praticamente qualquer ambiente, é fácil de usar e pode se criar objetos simples ou não simples em VRML. As desvantagens se referem à falta de interação com o sólido de revolução após a sua criação, pois são permitidas apenas operações de transformações.

2.4 Protótipo LEGO

Com o objetivo de facilitar e auxiliar o uso da informática na educação, buscou a elaboração de um protótipo em VRML, baseado nos brinquedos de montar LEGO [20]. Este sistema visa introduzir e auxiliar a interação aluno-máquina, buscando atingir determinados objetivos conforme o planejamento didático de cada professor. São destacadas várias características desejáveis em uma ferramenta de apoio ao ensino, como divertir aprendendo, manipulação simples, fácil compreensão das operações.

O referido protótipo consiste num portal de Internet com ambientes previamente construídos, cujos links remeteriam a variados ambientes do LEGO em VRML, sendo um para cada assunto desenvolvido, onde também é apresentado um ambiente em que constam apenas peças desmontadas, a fim de serem usadas para aplicações livres.

Este protótipo é direcionado a crianças e adolescentes, que estejam cursando o Ensino Fundamental e Pré-Escola. Para o ensino da Matemática de uma maneira geral trabalha formas geométricas, transformação de unidades, noções de geometria espacial, etc. Para a alfabetização, monta letras, palavras, etc.

As aplicações são muitas e por mais que se consiga listar um grande número, sempre existirão novos conteúdos que poderão ser trabalhados neste protótipo sugerido.

As peças foram criadas utilizando o Software 3D Studio Max: Discreet, Kinetix e atualmente Autodesk logo após exportou-se com a extensão .wrl a fim de serem visualizadas em um ambiente de VRML, como por exemplo, o Cosmo Player. Utilizou-se também o software VRML 2.0 [20]. A escolha do LEGO como assunto principal desta pesquisa é para auxiliar no aprendizado de diferentes conteúdos, por ser de fácil manipulação.

As desvantagens do protótipo é que não são permitidas interações diretas na peça LEGO. São permitidas manipulações de acordo com o conteúdo e os objetivos de cada aula.

2.5 LVCE – Laboratório Virtual de Circuitos Elétricos

O LVCE [21] é um sistema que simula um ambiente virtual para experimentos físicos, na área de eletrodinâmica, a baixos custos. Usa como tutorial uma ferramenta denominada mapas conceituais, objetivando guiar os alunos na aprendizagem. Como estudo de caso desenvolveu-se um ambiente para simular um laboratório virtual de circuitos elétricos.

O sistema permite ao aluno criar seus experimentos e, através dos mapas conceituais, conduz o aluno na construção de seu próprio conhecimento. A implementação foi feita na linguagem VRML e para a interação utilizou o JavaScript.

Veja a Figura 2.4 tem-se a página inicial do software e na Figura 2.5 um exemplo de uma simulação.

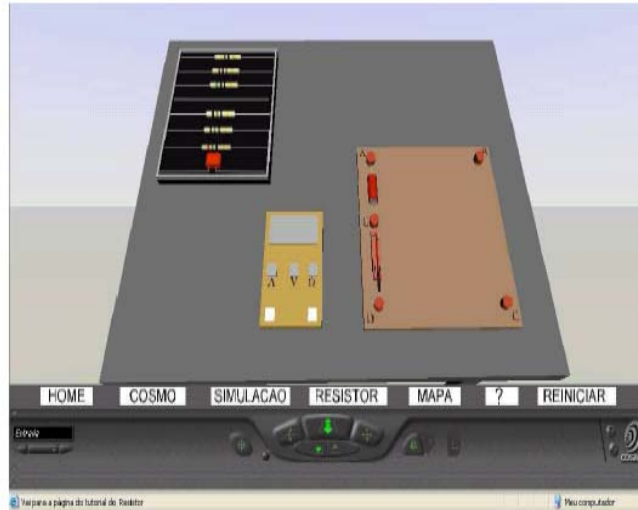


Figura 2. 4: Página inicial do ambiente de simulação



Figura 2. 5: Simulação

As principais contribuições deste sistema refere-se ao uso de desenho e processos pedagógicos integrados a técnicas de RV. Sendo bem aceito pelos avaliadores e pode se tornar uma ferramenta de apoio para as escolas e alunos que não têm acesso a laboratórios de física instalados nas dependências da escola.

As limitações do sistema são: impossibilidade de modificação dos valores dos resistores porque tais valores são fixos e limitados. Não faz referência entre os conceitos estudados e as equações matemáticas dos fenômenos físicos envolvidos no processo. Finalmente, o sistema permite apenas a montagem de circuitos elétricos e suas análises, não possibilitando uma interação que influencie na topologia dos objetos modelados.

2.6 SEFIRV - Sistema de Experiências Físicas Instrucional em Realidade Virtual

O SEFIRV [22] é um sistema que comporta 26 experimentos de Física, com o objetivo de aplicar e avaliar o software para o ensino de física que esteja direcionado ao Ensino Médio o qual deverá oferecer suporte ao Ensino presencial ou a distância. A linguagem utilizada foi o VRML e o JavaScript, garantindo assim o acesso do software via Internet. A Figura 2.6 mostra um experimento com o Movimento Retilíneo Uniforme.



Figura 2. 6: Experimento Movimento Retilíneo Uniforme

Neste cenário virtual encontra-se dois carros, onde o usuário pode alterar a velocidade dos carros, a distância entre eles e decidir o sentido do movimento. O usuário ainda pode visualizar a posição de cada carro, e o tempo gasto para o encontro entre eles.

As principais vantagens do sistema se refere ao grande potencial que a tecnologia de RV oferece à área educacional, principalmente, quando está associada a processos pedagógicos. Como desvantagens tem-se a impossibilidade de manipulação num mesmo cenário, de objetos de aprendizagens para a solução de problemas variados. Semelhante ao sistema anterior, também não possibilita que gerem mudanças topológicas nos objetos.

2.7 LUDOS TOP

O LUDOS TOP [23] é destinado a alunos do Ensino Fundamental e educadores de forma geral que deseja utilizar jogos para ensinar ou aprender conceitos matemáticos de maneira descontraída e divertida.

O objetivo deste trabalho é apresentar um protótipo de um ambiente de ensino, que envolve RV e, além disso, utiliza estratégias lúdicas que possibilita a aprendizagem pela descoberta.

As linguagens utilizadas foram VRML e JavaScript. VRML foi usada para a modelagem dos objetos virtuais e o JavaScript para o controle do jogo e animar os ambientes virtuais. A visualização ocorre por meio do Web browser e de um plugin VRML. Na Figura 2.7 abaixo vê-se o ambiente do jogo.

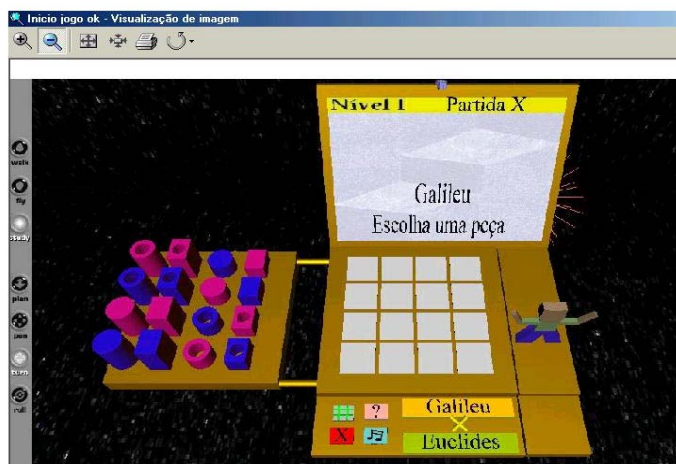


Figura 2. 7: Ambiente do jogo [24]

O jogo dá início quando os participantes escolhem ou definem um *nickname* para cada jogador. Estes *nicknames* são personagens importantes na história da matemática. Os jogadores escolhem a cor das peças e então é só clicar no botão Jogar para dar início a jogada. Dado início ao jogo na tela do Lap Top é apresentado a ordem das jogadas, ou seja, aparece na tela os *nicknames* correspondentes dos jogadores. Cada jogador escolhe a peça e arrasta para o tabuleiro. Vence o jogo quando ocupar uma linha, uma coluna ou a diagonal com quatro casas de um mesmo atributo, que pode ser: cor, furo, tamanho ou forma. Na Figura 2.8 o jogador ganhou pelo atributo cor.



Figura 2. 8: Ganhou pelo atributo cor [24]

A contribuição deste trabalho: Verificar se os alunos do Ensino Fundamental demonstrem habilidades necessárias para o entendimento de determinados conceitos matemáticos que aliados as teorias pedagógicas contribui para o desenvolvimento cognitivo do usuário. A desvantagem se refere a limitação dos objetos apenas para visualização, isto é, não permite mudar a topologia das peças.

2.8 Cabri Geometry

Cabri-Geometry é um software de construção em geometria desenvolvido pelo Institut d'Informatique et de Mathematiques Appliquees em Grenoble (IMAG) e é o resultado da colaboração constante de cientistas da informática, especialistas em educação e professores [25].

É um software de construção que nos oferece “régua e compasso eletrônicos”, sendo a interface de menus de construção em linguagem clássica da Geometria. Os desenhos de objetos geométricos são feitos a partir das propriedades que os definem [25].

É uma ferramenta auxiliar no ensino e aprendizagem da Geometria e é utilizado nos Ensinos Médio e Superior. Pode ser utilizado via Internet, via calculadora e em laboratórios da sala de aula. Veja um exemplo na Figura 2.9.

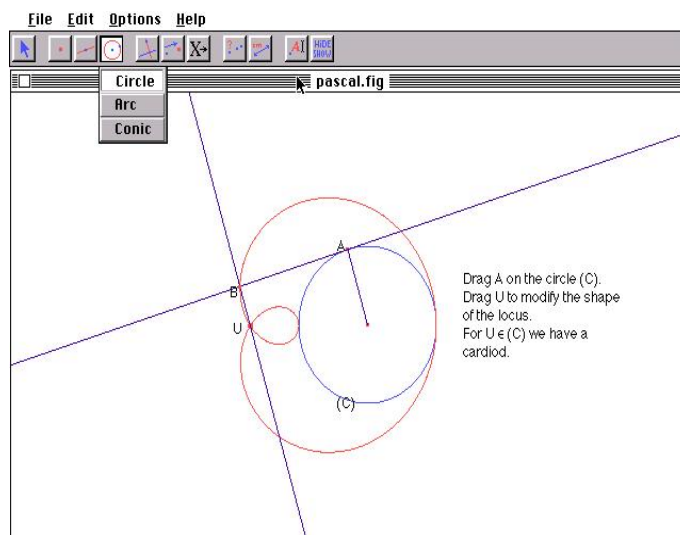


Figura 2. 9: Construção com o uso do Cabri

As vantagens de se usar este software são: os alunos constroem os desenhos dos objetos ou configurações, quando o objetivo é o domínio de determinados conceitos através da construção. Permite a simulação com o objetivo de fazer conjecturas, estabelece-se propriedades e dependendo do nível de escolarização trabalha-se as demonstrações dos resultados obtidos experimentalmente.

Uma das dificuldades em se trabalhar com este software é devido a quantidade de comandos, que torna trabalhosa para os alunos.

2.9 Corte de Superfícies

Este protótipo foi desenvolvido por [26], com o objetivo de apresentar um software educacional que explora técnicas de Realidade Virtual (RV) como uma

ferramenta alternativa no processo de ensino aprendizagem. Como estudo de caso, adotou-se o corte de Superfícies Quádricas.

As Figuras 2.10 e 2.11 abaixo ilustram o referido protótipo:

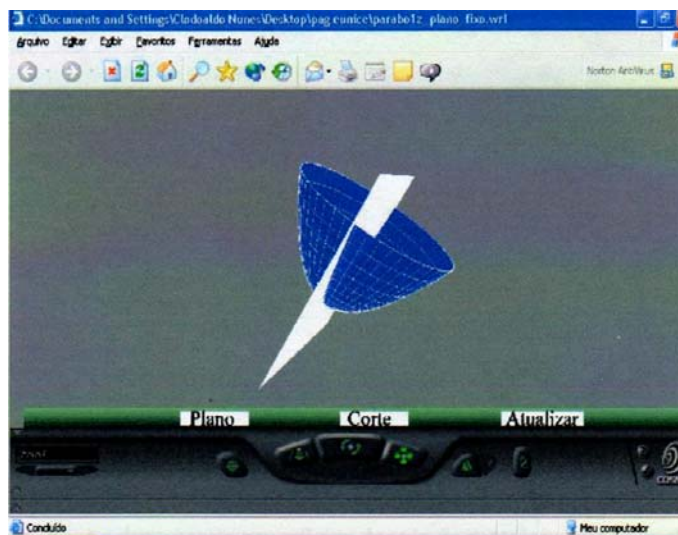


Figura 2. 10: Plano de corte no parabolóide

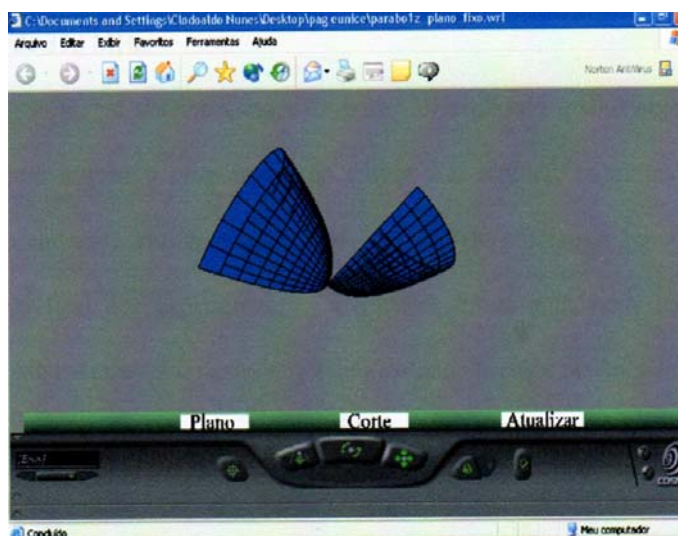


Figura 2. 11: Corte no parabolóide

As linguagens utilizadas para o desenvolvimento do trabalho foram VRML e JavaScript. Para gerar as devidas superfícies foi utilizado o software MatLab.

As principais contribuições deste trabalho: o sistema mostrou-se útil para auxiliar nas disciplinas de Geometria Analítica e Cálculo; houve uma interação maior por parte do usuário no ambiente virtual. As principais limitações se refere a não haver interação na superfície após o corte, o sistema não permite entrada de equações quádricas, para a partir da equação informada o sistema executar a visualização da superfície.

2.10 Corte de Poliedros Convexos

Este protótipo foi desenvolvido por [27], utilizando VRML e Javascript, com o objetivo de apresentar um software educacional que explora técnicas de Realidade Virtual (RV) como uma ferramenta alternativa no processo de ensino aprendizagem na Geometria Espacial. Como estudo de caso corte de poliedros convexos.

As Figuras 2.12 , 2.13 e 2.14 ilustram o referido protótipo:

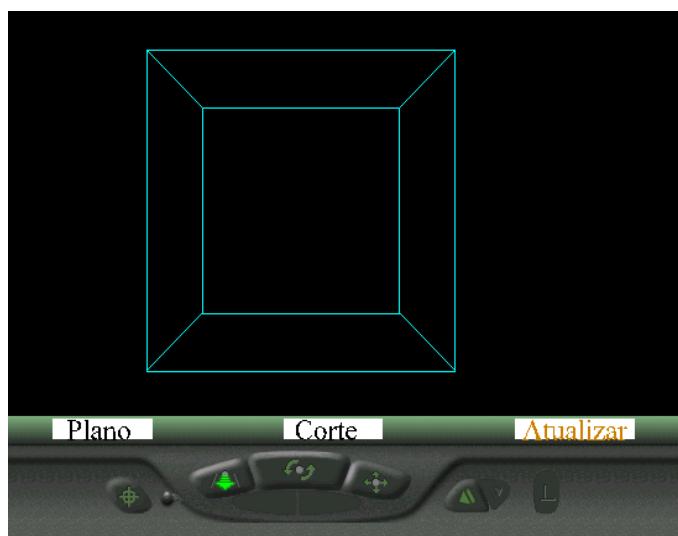


Figura 2. 12: Interface do cubo

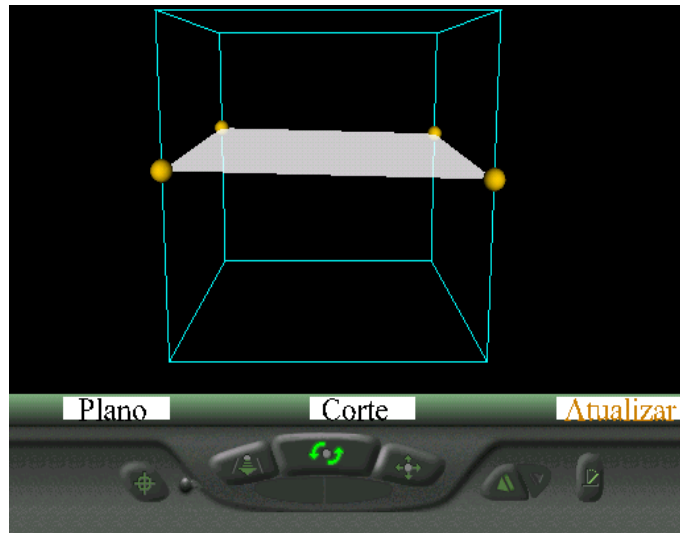


Figura 2. 13: Plano de corte no cubo

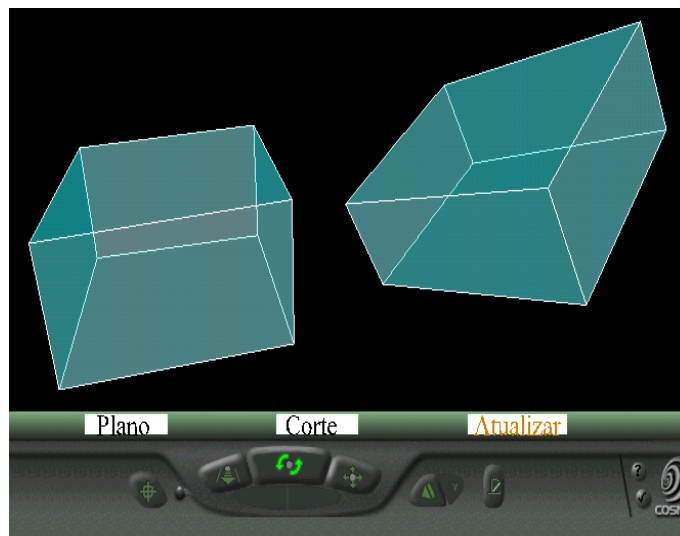


Figura 2. 14: Corte no cubo

Na Figura 2.11 tem-se a Interface do sistema, e na Figura 2.12 a escolha de três pontos não colineares pelo usuário e a visualização do plano de corte. Na Figura 2.13 o sistema gerando dois novos sólidos. Na Interface visualiza-se três botões: Plano, Corte e Atualizar onde respectivamente o primeiro botão traça o plano de corte após o usuário escolher os três pontos não colineares, o segundo botão faz a

divisão do sólido em dois novos sólidos e o terceiro botão atualiza o sistema para novos cortes.

Os objetivos deste protótipo foram alcançados no que se refere a interação direta com os objetos do ambiente virtual. A grande contribuição é permitir a visualização e a interação direta pelos usuários de figuras tridimensionais para a área de Geometria Espacial.

Dois problemas deste trabalho foram corrigidos nesta dissertação. O primeiro, um erro no algoritmo do cubo onde, ao selecionar o plano de corte nas diagonais do cubo aparecia a mensagem de pontos colineares devido a falha no algoritmo matemático. Para esta falha foi utilizado outro algoritmo para determinar colinearidade dos pontos ao invés do cálculo do determinante. Os detalhes deste algoritmo estão citados no capítulo 4 desta presente dissertação. O segundo problema foi solucionado com a generalização do corte para poliedros não-convexos, sendo portanto esta dissertação motivada por este trabalho relacionado aqui. Veja o quadro comparativo abaixo:

Propriedade Sistema	Nenhuma Interação (AV)	Pouca Interação (AV)	Significativa Interação (AV)	Domínio de Objetos Virtuais (N)	Domínio de Objetos Virtuais (P)	Domínio de Objetos Virtuais (S)
Geo3D	x				x	
ATSWorld	x					x
LVCE			x		x	
SEFIRV			x			x
LUDOSTOP		x			x	
Cabri- Geometry		x			x	
Corte de Superfícies		x			x	
Corte de Poliedros Convexos		x			x	
Corte de Poliedros Não-Convexos		x				x

Conclusão: Necessidade de desenvolver algoritmos para suportar um número maior de objetos e experimentos em Geometria Espacial (Corte).

2.11 Sumário e Conclusões

Embora os índices de informatização nas escolas tenham aumentado consideravelmente de 1999 para 2001 (último ano com dados globais levantados pelo MEC), a pesquisadora Neide de Aquino Noffs, da Faculdade de Educação da PUC-SP diz que a inclusão digital nas escolas da rede pública ainda não é uma realidade. "O laboratório de informática existe, mas não é usado com frequência. Não é uma atividade rotineira para os alunos; não é como a biblioteca, que fica aberta o tempo todo", afirma Noffs [28].

O computador chega às escolas, mas as atividades são mínimas: na rede privada, aluno faz software, na pública espera o acesso a Internet. O laboratório já existe, mas na maioria das vezes é subtilizado e somente para cursos de informática e não inserido como prática cotidiana das aulas. Muitas coisas estão surgindo rumo a educação, mas muito ainda há por fazer [29].

No Brasil, ainda há um largo fosso entre as escolas públicas e privadas que têm computadores conectados à internet. Números do Censo Educacional de 1999, do Ministério da Educação, apontam que 3,2% das escolas públicas de ensino fundamental estão conectadas versus 39,2% das particulares; e 10% das escolas públicas de ensino médio estão conectadas versus 58,9% das particulares [30].

De acordo com o MEC, apesar de o número de computadores instalados nas escolas até 2002 ter ficado abaixo das metas estabelecidas pelo governo anterior, a capacitação de professores superou as expectativas [28].

Os trabalhos relacionados acima demonstram o grande potencial de RV aplicado como tecnologia educacional.

Em Realidade Virtual a interação é uma das características mais importantes dos ambientes de aprendizagem. Outra característica é a condição da construção dos objetos; os usuários do ambiente devem ter condições de construir os seus próprios objetos, o que lhes permitirá o desenvolvimento de vários processos cognitivos [31].

Dentre os projetos expostos relata-se aqui as vantagens e as melhorias que faltam em termos de interação, como o projeto que permite visualizar e manipular objetos no ambiente virtual, faltando explorar a interação no sentido de mudar a topologia dos objetos a partir de figuras existentes [17]. E outros criando novos sólidos [19].

Além disso, existem trabalhos que usam a interação no sentido de ser possível gerar uma simulação, um experimento no ambiente virtual como se fosse real [21,22]. Também tem-se a tecnologia aliada a jogos matemáticos enriquecendo os conceitos matemáticos de forma lúdica [23].

Finalmente têm-se os trabalhos que já exploram a interação de maneira que muda a topologia dos sólidos através da criação de novos objetos a partir de figuras existentes [26,27].

No capítulo seguinte expõe-se a arquitetura do sistema proposto para esta dissertação, generalizando-se o corte para poliedros não-convexos.

Capítulo III

3. ARQUITETURA DO SISTEMA PROPOSTO

3.1 Introdução

O principal objetivo deste capítulo é apresentar uma arquitetura que permite contribuir para o ensino aprendido de visualização e corte de poliedros não-convexos, com a extensão do corte para qualquer poliedro.

Será detalhado as etapas necessárias para a construção deste protótipo e apresentado as linguagens utilizadas para a realização deste software bem como a arquitetura do sistema proposto.

3.2 VRML

VRML (*Virtual Reality Modeling Language*) é uma linguagem de descrição de simulações interativas com vários participantes se comunicando através da Internet. Espera-se que todos os aspectos referentes a apresentação, interação, *internetworking* de um mundo virtual possam ser especificados usando VRML. A intenção dos projetistas desta linguagem é torná-la padrão para a descrição de simulações interativas no WWW, assim como a linguagem HTML é o padrão para descrição de páginas Web [32].

Possibilita descrever cenas 3D de forma que construa e anime cenas ou ambientes virtuais. As cenas criadas são, em geral, disponibilizadas na web e suas áreas de aplicação são bastante diversificadas, indo desde aplicações na área científica e tecnológica até entretenimento e educação, passando por representações artísticas e em multimídia [33].

Nestes ambientes, a navegação se desenvolve de uma maneira muito mais intuitiva, dado que a forma de agir dentro do ambiente virtual é similar à da vida real. Pode-se mover em todas as direções, não só esquerda e direita, mas também para frente, para trás para cima e para baixo. Pode-se lidar com os objetos como na própria vida, tocá-los, arrastá-los, etc. e em geral, tudo o que se possa imaginar. Também os cenários são muito mais reais. Pense num exemplo de como poderia ser uma biblioteca virtual. Nela poderíamos andar por cada uma das salas, pegar determinados livros e os ler. Com o tempo, o acesso à Internet se converterá em uma experiência muito mais perto da qual realizamos na vida e as visitas aos lugares retratados na Rede serão muito mais reais [34].

Para trabalhar com VRML precisa-se de:

Um editor de textos simples, utilizamos VRML PAD [35] e um visualizador VRML, utilizamos o Cosmo Player [36].

É uma linguagem independente de plataforma e seu arquivo é lido por um browser, como por exemplo Internet Explorer, acompanha um plug-in, como o Cosmo Player.

3.3 JavaScript

A linguagem JavaScript foi criada pela Netscape com o objetivo de obter uma linguagem intermediária entre a linguagem Java e a HTML, de tal forma que fosse complementar e integrada a estas duas linguagens. Assim o JavaScript é fácil de usar, aberta, e tem capacidade de ligar objetos e recursos tanto da linguagem Java como da HTML. Enquanto as *applets* Java são desenvolvidas apenas por programadores, o JavaScript pode ser usado por autores de páginas HTML para controlar a interação e comportamento de suas páginas [32].

Um dos principais benefícios que o JavaScript oferece é a capacidade de reduzir o tráfego da rede através da execução local de tarefas simples. Assim, ao invés do servidor executar as tarefas e retornar os resultados para o *browser*, este último pode manipular algumas tarefas localmente, com um menor tempo de resposta. A linguagem JavaScript é interpretada pelos *browsers* Web que possuem este recurso (por exemplo o Netscape e Internet Explorer). O código da aplicação JavaScript é recuperado pelo *browser Web* na forma de texto, juntamente com o documento HTML [32].

Com o JavaScript pode-se facilmente implementar respostas a eventos do usuário, tais como cliques do *mouse*, movimentos do *mouse* sobre um *link*, e entrada de dados em um formulário. Dessa forma é de grande utilidade na criação de interfaces mais explicativas ao usuário. Também é possível criar-se páginas dinâmicas, cujo conteúdo muda de acordo com as requisições do usuário, ou mesmo disparar sons ou executar *applets* quando o usuário entra ou sai de uma página, permitindo assim uma maior interatividade com o usuário [32].

Para controlar a interatividade das cenas a linguagem VRML foi integrada à linguagem JavaScript, uma vez que esta pode controlar a interatividade de um ambiente VRML via um script. Os scripts fornecem uma maneira de estender as funcionalidades da linguagem VRML produzindo assim a interação com o mundo virtual [21,22,23].

3.4 Arquitetura do Sistema

O sistema possui uma interface voltada para o domínio de Geometria Espacial com o corte e visualização de poliedros não-convexos. Permite que alunos e professores tenham acesso ao software sem necessitar de conhecimentos específicos na área de informática. Na Figura 3.1 tem-se a arquitetura do sistema protótipo.

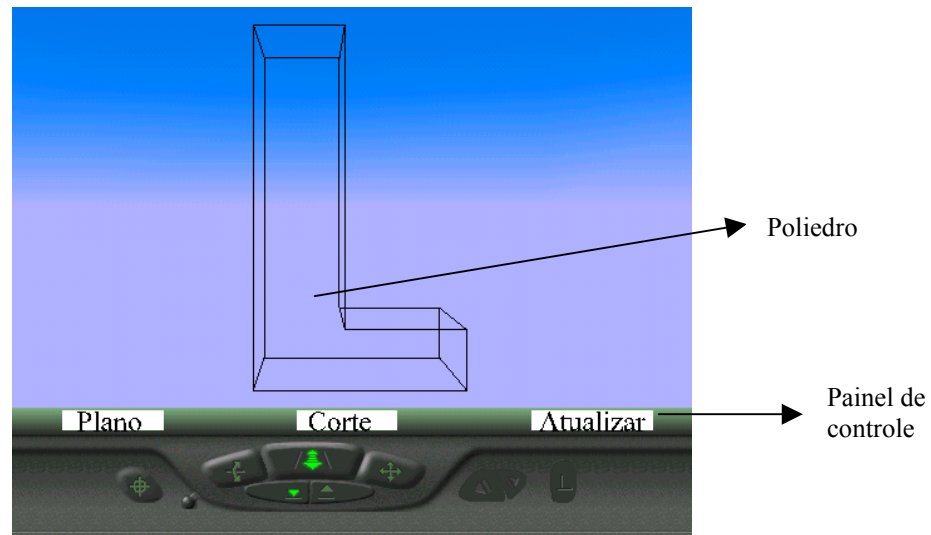


Figura 3. 2: GUI – Interface

3.4.2 Ambiente Virtual

É composto pelo módulo Elementos Virtuais e Painel de Controle. Os Elementos Virtuais e o Painel de Controle foram implementados utilizando a linguagem VRML.

O Ambiente Virtual é composto pelos poliedros não-convexos em forma de letras maiúsculas como (L, C ,T). Estes três poliedros citados foram implementados para se fazer o corte. O poliedro é exibido através de wireframe, ou seja, dando ênfase as arestas conforme visto na Figura 3.2. O motivo de se dar ênfase para a visualização através de arestas é porque em cada aresta terá sensores de toque, que será detectado quando aproximar o mouse sobre elas. O usuário irá escolher de forma aleatória três pontos não-colineares para clicar nas mesmas, gerando o plano de corte.

Todo o cenário deste ambiente virtual foi feito em VRML. O Painel de Controle contém três botões que chamam as funções scripts relacionadas o qual terá ligação direta com o Núcleo Gerenciador de Corte. Estes botões (Plano, Corte e Atualizar)

gerenciam todo o corte, é mostrado na Figura 3.2. Cada botão terá um sensor que capta o toque em cada um deles.

O botão Plano ativa uma função script que vai gerar o plano de corte após o usuário escolher os três pontos aleatórios. O botão Corte irá ativar as funções scripts que irão cortar os poliedros em dois ou mais sólidos. E por último o botão Atualizar irá ativar o sistema novamente para novas tentativas de corte.

3.4.3 Núcleo Gerenciador de Corte

No bloco Núcleo Gerenciador de Corte, estão todas as funções scripts implementadas em JavaScript que fará o controle do ambiente virtual através das interações diretas do usuário com o mundo virtual. A interação é feita através dos cliques com o mouse em cada botão do painel de controle e através da escolha dos pontos pelo usuário. O Núcleo controla todas as ações do usuário, manipula os dados e retorna ao ambiente em tempo real.

O ambiente será disponibilizado para a escolha de três pontos nas arestas do poliedro pelo usuário, gerando assim o plano de corte o qual o cortará em dois ou mais poliedros dependendo se for convexo ou não-convexo.

Este sistema foi adequado, para qualquer usuário utilizá-lo, sem necessidade de conhecimentos técnicos de informática. A utilização deste software será por meio simples como selecionar os pontos através do clique do mouse, como também clique nos botões do painel de controle.

3.5 Conclusões

O Capítulo 2 mostrou que as técnicas de RV utilizadas aproximam-se daquele observado em [27]. Considerando que este trabalho é uma extensão daquele conclui-se que está dentro deste contexto.

Esta arquitetura do sistema vai beneficiar o conteúdo de Matemática, com a produção de materiais que auxiliará o ensino de Geometria Espacial. No próximo capítulo tem-se a implementação do sistema.

CAPÍTULO IV

4. IMPLEMENTAÇÃO DO SISTEMA

4.1 Introdução

Neste capítulo, descreve-se a implementação do sistema proposto. As linguagens utilizadas foram VRML e JavaScript, tendo o sistema protótipo o intuito de oferecer recursos auxiliados pelo computador para atuar no ensino/aprendizagem da Geometria Espacial.

4.2 Características do Ambiente Virtual

O sistema está dividido em três módulos, a saber: O Painel de Controle, os objetos virtuais e o Núcleo Gerenciador do Corte, ilustrados na Figura 4.1.

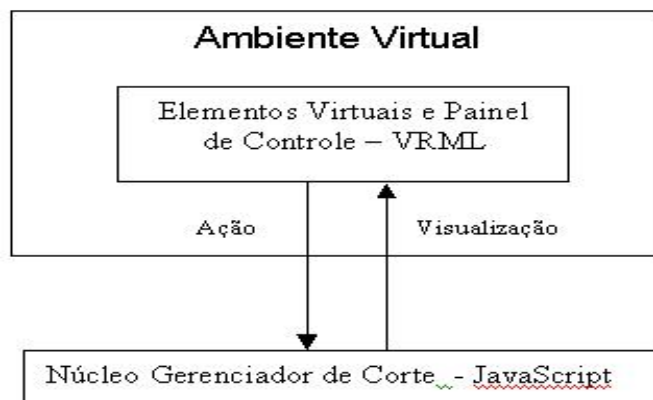


Figura 4. 1: Bloco principal do Ambiente Virtual

No Ambiente Virtual está localizado o painel de controle, juntamente com todos os elementos que pertencem ao ambiente virtual. No Núcleo Gerenciador do Corte

estão presentes todas as funções matemáticas que foram implementadas em JavaScript, as quais são responsáveis pela interação do ambiente.

4.2.1 Ambiente Virtual

O Ambiente Virtual é composto de uma interface, onde está presente o painel de controle e o poliedro não-convexo, implementados em VRML.

Os poliedros foram criados através de uma função do VRML chamada `geometryIndexedLineSet`. Nesta função, digita-se os seus vértices e logo em seguida sua posição para então efetuar as ligações entre eles formando as arestas. Assim, obtem-se seu esqueleto (wireframe). A seguir, pode ser visto o código da implementação de um poliedro não-convexo cujo formato é um L maiúsculo.

```
DEF poliedros Group      {
  children [
    DEF LetraL Transform {# Poliedro, no caso um L
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 0 0 0   specularColor .29 .3 .29
              ambientIntensity 0   shininess .08}}
          geometry IndexedLineSet {
            coord DEF vertices Coordinate {
              point [ # vertices letraL
                -1.0 3.0 1.0,  0.5 3.0 1.0,  0.5 3.0 -1.0,  -1.0 3.0 -1.0,
                -1.0 -3.0 1.0,  2.5 -3.0 1.0,  2.5 -3.0 -1.0,  -1.0 -3.0 -1.0,
                0.5 -2.0 1.0,  2.5 -2.0 1.0,  2.5 -2.0 -1.0,  0.5 -2.0 -1.0,
                -1.0 -2.0 -1.0,  -1.0 -2.0 1.0,  0.5 -3.0 1.0,  0.5 -3.0 -1.0]}
            coordIndex [0, 1, 2, 3, 0, -1, #De acordo com as posições dos vértices
              4, 5, 6, 7, 4, -1, #determinamos as ligações entre eles.
              0, 4, -1,
              3, 7, -1,
              2, 11, -1,
```

```

        1, 8,-1,
        8,9,10,11,8,-1,
        5,9,-1,
        6,10,-1,
    ]
    }#fim do geometry
    } # fim shape
    DEF SENSOR TouchSensor { }
  ] # fim children
}#fim do def letraL transform

```

Quadro 4.1: Código da implementação do poliedro não-convexo

Na Figura 4.2 tem-se a Interface do Poliedro. Abaixo os botões que controlam o ambiente virtual.

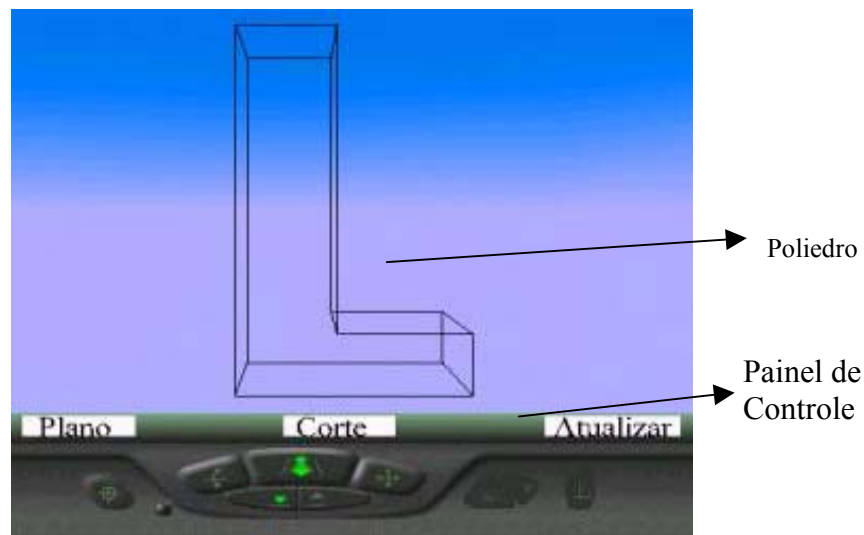


Figura 4. 2: Interface do poliedro L

A escolha dos pontos pelos usuários é feita através de um duplo clique do mouse nas arestas, daí o motivo de exibir o poliedro pelo seu esqueleto, pois nas

arestas do poliedro terá sensores – DEF SENSOR TouchSensor {} – os quais vão capturar a passagem do mouse sobre as arestas e, permitindo assim que o usuário escolha os pontos. Este sensor capta as coordenadas dos pontos, que serão armazenadas em vetores por funções scripts para serem utilizadas nos cálculos matemáticos para a geração do plano de corte.

Quando o usuário clicar nas arestas, estas serão marcadas por pequenas esferas para obter uma melhor visualização dos pontos (Figura 4.3). Essas esferas já existem no grafo de cena do ambiente virtual, e serão transladadas para a posição dos pontos onde for feito o clique com o mouse.

Na Interface existem três botões, que fará o controle do ambiente virtual, o primeiro denominado Plano, o usuário escolhe três pontos não-colineares, com um clique do mouse nesse botão será gerado o plano de corte. O segundo botão denominado Corte, com um clique gerará os novos sólidos. O terceiro botão denominado Atualizar com um clique o ambiente virtual será iniciado para novos cortes (Figura 4.3).

A escolha dos três pontos não-colineares, ou seja que não estejam em linha reta, é a condição para que exista um plano. Caso o usuário escolha pontos colineares, quando der o clique no botão plano aparecerá um alerta na tela com a seguinte mensagem: “Pontos inválidos, pois são colineares, clique no botão atualizar e escolha pontos não-colineares”, veja a ilustração na Figura 4.3.



Figura 4. 3: Alerta - Pontos colineares

Sendo esse trabalho uma extensão, alguns dos algoritmos do referido trabalho [27] foram reutilizados. Detectou-se ao longo do processo que o algoritmo, o qual detecta quando três pontos estão alinhados não funciona corretamente, quando os vetores correspondentes aos respectivos pontos estejam paralelos a um plano. Veja ilustração na Figura 4.4 e Figura 4.5.

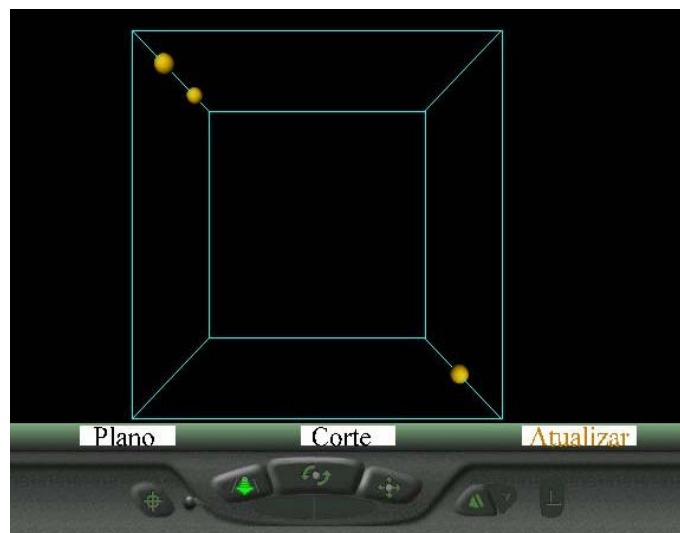


Figura 4. 4: Escolha de três pontos na diagonal do cubo

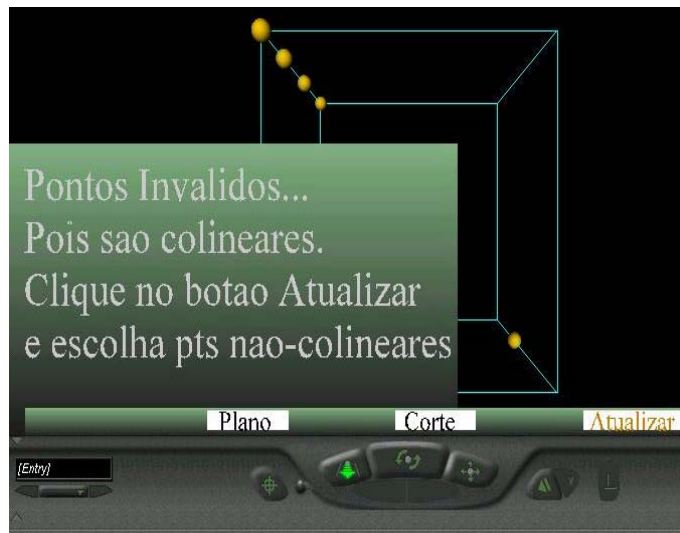


Figura 4. 5: Mensagem de pontos colineares e os pontos não são colineares

Isto ocorre porque nesse caso o determinante que nada mais é o produto misto dos referidos vetores é igual a zero sem que os pontos estejam alinhados. Pois tinha sido determinado que o determinante sendo zero os pontos estariam alinhados.

Nesse sentido, no lugar do referido algoritmo usou-se o seguinte cálculo matemático: Sejam três pontos alinhados A, B e C (Figura 4.6) e (Figura 4.7):

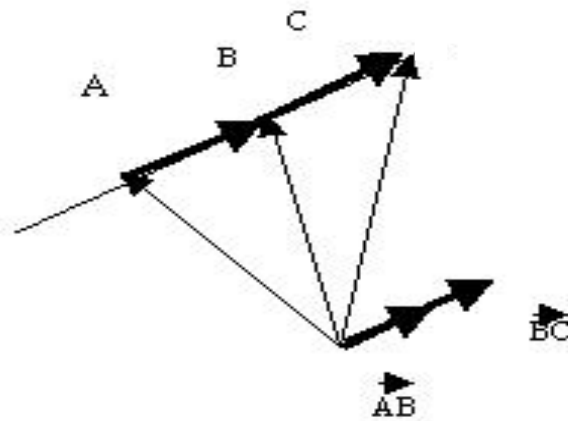


Figura 4. 6 : Posição de 3 pontos A, B e C alinhados e o ângulo entre os vetores é 0° .

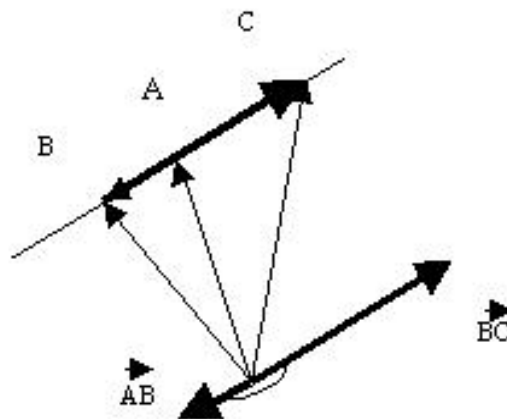


Figura 4. 7: Posição de 3 pontos A, B e C alinhados em que o ângulo entre os vetores é 180°

Da condição de colinearidade tem-se que $\vec{AB}$ e $\vec{BC}$ são dois vetores paralelos. Isso implica que o ângulo entre eles é 0° (Figura 4.6) ou 180° (Figura 4.7).

O produto escalar [37] entre eles ($\vec{AB} \cdot \vec{BC}$) é :

$$|\vec{AB}| \cdot |\vec{BC}| \text{ se o ângulo é } 0^\circ, \text{ e (Equação 4.1)}$$

$$- |\vec{AB}| \cdot |\vec{BC}| \text{ se o ângulo é } 180^\circ. \text{ (Equação 4.2)}$$

$$\text{Portanto, } (\vec{AB} \cdot \vec{BC})^2 = |\vec{AB}|^2 \cdot |\vec{BC}|^2 \text{ (Equação 4.3)}$$

Assim, se a igualdade se verifica os pontos estão alinhados, não existindo o plano de corte. (Equação 4.3)

A Figura 4.8 mostra os pontos não-colineares selecionados pelo usuário, marcados pela esfera, estes pontos fixos gerará o plano de corte, pois três pontos não-colineares determinam um plano.

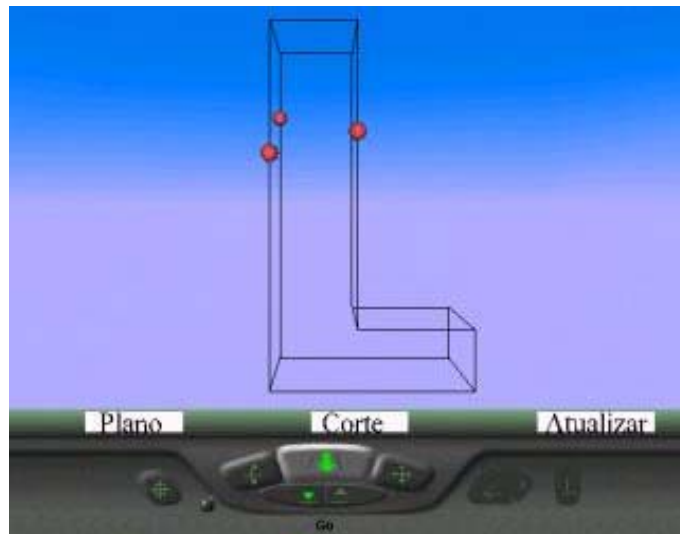


Figura 4. 8: Pontos não-colineares

4.2.2 Painel de Controle

O Painel de Controle é composto por três botões: *Plano*, *Corte* e *Atualizar*, conforme foi citado no item anterior. O usuário selecionará três pontos nas arestas

do poliedro em questão e ao clicar no botão *Plano*, este ativa uma função script que envia os pontos escolhidos pelo usuário para o Núcleo Gerenciador e, este retorna o plano de corte que será visualizado no ambiente virtual.

Ao selecionar pontos não-colineares o plano é gerado através de sua equação. Existe apenas um plano que passa por três pontos P1, P2, P3 não alinhados ou seja não-colineares [37]. Para encontrar a equação desse plano, calculamos o determinante onde x, y e z representam as coordenadas de um ponto qualquer do plano, conforme em [27]. Veja a equação 4.4.

$$\begin{vmatrix} x-x_1 & y-y_1 & z-z_1 \\ x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_3-x_1 & y_3-y_1 & z_3-z_1 \end{vmatrix} = 0 \quad (4.4)$$

Na linguagem JavaScript para se fazer os cálculos do determinante, é necessário o desmembramento dos valores de A, B, C e D as normais da equação do plano $Ax + By + Cz + D = 0$ conforme as equações 4.5:

$$\begin{aligned} A &= y_1(z_2-z_3) + y_2(z_3-z_1) + y_3(z_1-z_2) \\ B &= z_1(x_2-x_3) + z_2(x_3-x_1) + z_3(x_1-x_2) \\ C &= x_1(y_2-y_3) + x_2(y_3-y_1) + x_3(y_1-y_2) \\ D &= (x_1*z_2*y_3) - (x_1*y_2*z_3) + (x_2*y_1*z_3) + \\ &\quad -(x_2*z_1*y_3) - (x_3*z_2*y_1) + (x_3*y_2*z_1) \end{aligned} \quad (4.5)$$

Calculadas as normais do plano, o próximo passo é fazer a intersecção da equação do plano com as arestas do poliedro ou seja semi-retas, para obter todos os

pontos pertencentes ao plano que será gerado com as arestas. Encontradas todas as intersecções será possível representar o plano de corte.

O algoritmo usado para fazer a intersecção da equação do plano com as arestas, será descrito logo a seguir. As arestas estão sendo representadas por dois vértices cada e armazenadas em uma lista. O algoritmo usado foi descrito em [27]. Veja a equação geral do plano (4.6):

$$Ax + By + Cz + D = 0 \quad (4.6)$$

Os pontos x , y e z representam todos os pontos que pertencem ao plano. Substituindo a equação geral do plano (4.6) pela equação dada em 4.7:

$$P = P_i + t(P_f - P_i) \quad (4.7)$$

onde $P_i(x_i, y_i, z_i)$ e $P_f(x_f, y_f, z_f)$ são os vértices das arestas, pois conforme foi citado anteriormente, uma aresta é composta de dois vértices.

Substituindo a equação 4.7 na equação 4.6 obtemos a equação abaixo 4.8:

$$A(x_i + t(x_f - x_i)) + B(y_i + t(y_f - y_i)) + C(z_i + t(z_f - z_i)) + D = 0 \quad (4.8)$$

Isola-se o t conforme equação 4.9:

$$t = \frac{Ax_i + By_i + Cz_i + D}{A(x_i - x_f) + B(y_i - y_f) + C(z_i - z_f)} \quad (4.9)$$

Para que a intersecção pertença as arestas, é preciso limitar o valor de t , $0 \leq t \leq 1$. Os pontos de intersecção são detectados através da equação (4.10), onde I representa os pontos de intersecção e P_i e P_f os pontos pertencentes as arestas:

$$I = P_i + t(P_f - P_i) \quad (4.10)$$

Todos os pontos de intersecção calculados serão guardados em uma lista. Em VRML não tem como controlar esses pontos de intersecção e os seus respectivos

índices de maneira que determine um polígono convexo. Para ordenar esses pontos de maneira que se torne um polígono convexo para gerar o plano de corte, como também para cada face criou-se um algoritmo que ordena os pontos no sentido horário. Este algoritmo ordena pontos em 2D. Tanto para o plano de corte quanto para cada face. Separa-se cada face através dos vértices em um vetor, cada face tem uma coordenada de cada vértice igual, então para utilizar neste algoritmo ignora-se essa coordenada. Desse modo, planifica-se as faces, para utilizar neste algoritmo. O mesmo procedimento é feito para todos os pontos de intersecção, que vão gerar o plano.

Armazenados os pontos em seus respectivos vetores, encontra-se o centróide do plano de corte, ou seja de cada polígono. Para o cálculo do centróide (C) encontra-se o vértice de máximo e o vértice de mínimo, faz-se uma média entre eles. Veja a implementação: C (XC,YC)

$$XC = (x_{\max} - x_{\min})/2.0 + x_{\min};$$

$$YC = (y_{\max} - y_{\min})/2.0 + y_{\min};$$

Para cada lista de pontos armazenados, calcula-se o valor do centróide (C), que se localiza no ponto central de cada polígono, é o centro de um eixo cartesiano xy. Veja uma representação na Figura 4.9: Tem-se os vértices $P_0, P_1, P_2, \dots, P_n$ e o centróide C.

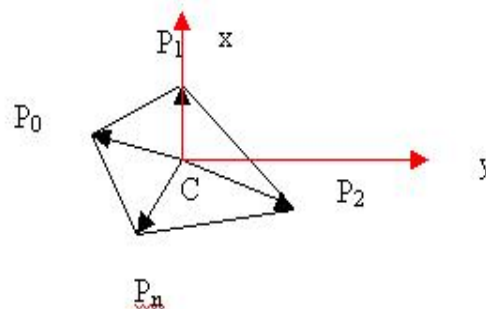


Figura 4. 9: Centróide

Localizado o centróide, calcula-se o ângulo que corresponde a cada vetor, $\overrightarrow{CP_0}, \overrightarrow{CP_1}, \dots, \overrightarrow{CP_n}$, conforme a implementação abaixo:

```
for(i=0;i<v;i++){ //v é o número de vértices
    auxx=(lista[i][vx]-XC) * (lista[i][vx]-XC);
    auxy=(lista[i][vy]-YC) * (lista[i][vy]-YC);
    d=Math.sqrt(auxx + auxy);
    coseno=(lista[i][vx]-XC)/d;
    seno=(lista[i][vy]-YC)/d;
    if(coseno>0)
    {
        if (seno>=0)

            ang[i]=Math.asin(seno);
        else
            ang[i]=Math.asin(seno)+(2*3.1415);
        }
    else
        if(seno>=0)
            ang[i]=Math.acos(coseno);
        else
            ang[i]=(2*3.1415) - Math.acos(coseno);
    }
}
```

Quadro 4.2: Implementação do cálculo do ângulo de cada vetor

Este algoritmo que também foi adotado em [27] tem o objetivo de verificar em quais quadrantes os ângulos possuem os mesmos valores, pois em determinados quadrantes os valores de senos e cossenos são iguais. Por exemplo, se a variável cosseno e seno ambas forem maior que zero temos : o cosseno pertencerá ao 1º ou 4º quadrante. O seno estará no 1º quadrante, pois o seno é positivo no 1º e 2º quadrantes. O mesmo raciocínio é usado se a variável seno for menor que zero, o ângulo estará no 4º quadrante, pois o seno é negativo no 3º e 4º quadrantes.

Para ângulos pertencentes ao 1º quadrante, basta calcular o $\arcsen(\text{seno})$. Agora, se pertencer ao 4º quadrante tem que fazer a soma $\arcsen(\text{seno}) + 2\pi$.

O mesmo cálculo é usado caso a variável cosseno seja menor que zero. Se o seno for maior que zero, o ângulo pertencerá ao 1º ou 2º quadrante. Sabemos que o cosseno pode ser negativo no 2º e 3º quadrantes, então o ângulo está no 2º quadrante. Entretanto, se o seno for menor que zero, pertencerá ao 3º e 4º quadrante, e como o cosseno é negativo, o ângulo está localizado no 3º quadrante.

O ângulo localizado no 2º quadrante calcula-se o $\arccos(\text{seno})$, pertencendo ao 3º quadrante o valor do ângulo será $2\pi - \arccos(\text{cosseno})$.

Calculados os ângulos, basta ordená-los em ordem crescente. Veja o algoritmo de ordenação utilizado na implementação abaixo:

```

for (j = 1; j < v; j++) {
    ajuda[0] = ang[j];
    ajuda1[0] = lista[j];
    for (i = j-1; i >= 0 && ang[i] > ajuda[0]; i--)
    {
        ang[i+1] = ang[i];
        lista[i+1] = lista[i];
    }
    ang[i+1] = ajuda[0];
    lista[i+1] = ajuda1[0];
} //fim do for j=1

```

Quadro 4.3: Algoritmo de ordenação de ângulos

Agora, tendo os ângulos ordenados, logo têm-se as suas respectivas coordenadas também ordenadas juntamente com os índices determinados, então é só enviá-los para o mundo. Antes o grafo de cena se encontrava assim:

```

DEF Plano2 Transform {
    children [ ]
}

```

O campo Plano2 está em aberto, ou seja, não possuem parâmetros pré-definidos, os dados serão atualizados através do *JavaScript* após serem feitos os cálculos matemáticos da equação do plano e transportados para o ambiente virtual através da função *createVrmlFromString*. Este novo children é todo criado no Núcleo Gerenciador, contendo os vértices ordenados e os seus respectivos índices, isto para cada polígono que fará parte dos novos sólidos gerados após o corte, pois irá se construir os sólidos face a face.

Logo em seguida, adiciona-se ao mundo através da função do VRML *Plano2.addChildren* e segue através do comando *ROUTE* do VRML para o campo em aberto, preenchendo-o, veja abaixo a rota como está na implementação:

```
ROUTE SCRIPT.new_f TO Plano2.children
```

Na Figura 4.10 visualiza-se o plano de corte no ambiente virtual, este criado através da ligação dos pontos de intersecção do plano com as arestas do poliedro.

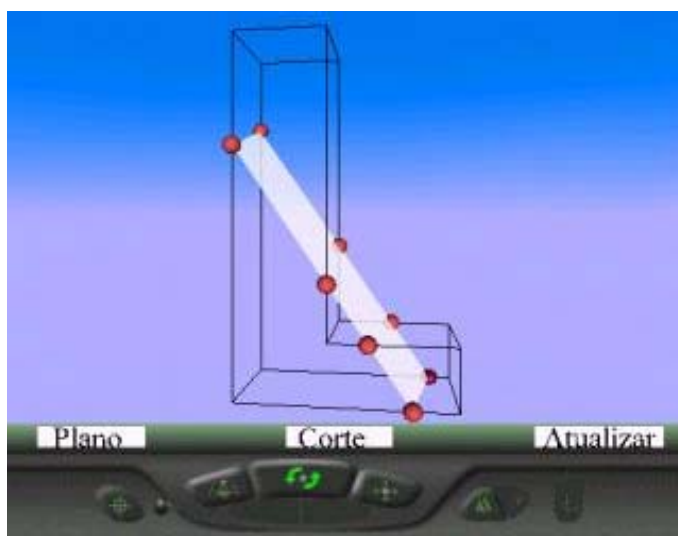


Figura 4. 10: Plano de corte

Após o traçado do plano de corte, basta separar o poliedro. Para isso o plano de corte terá que :

Passo1: Reconhecer se passa pela região convexa ou

Passo2: Região não-convexa do poliedro.

Essa região é determinada através da detecção do ponto externo a área da face não-convexa. Veja Figura 4.11 o retângulo colorido representa a área externa de cada face não-convexa que são duas de cada poliedro, nesse exemplo o poliedro cuja face não-convexa tem o formato de L.

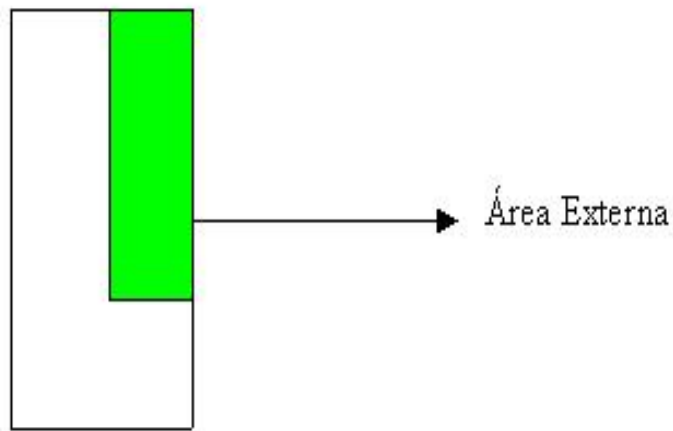


Figura 4. 11: Área externa

Primeiramente, coloca-se os pontos de intersecção que pertencem as faces não-convexas em ordem crescente ou decrescente. Logo após calcula-se os pontos médios desses pontos de intersecção: $(A+B)/2$. Se o ponto médio é interno, então o poliedro será dividido apenas em duas partes, ou seja o plano passa pela região convexa. Veja Figura 4.12.

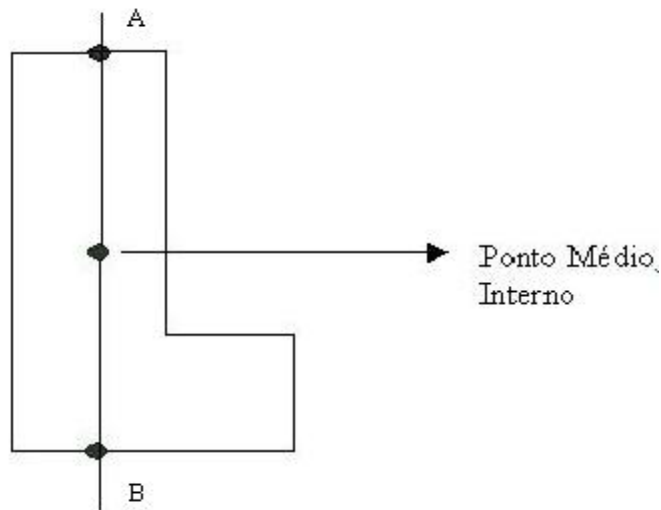


Figura 4. 12: Ponto médio interno

Um poliedro é convexo se qualquer reta não paralela a nenhuma de suas faces o corta em no máximo, dois pontos [38]. Ou, equivalentemente, um poliedro é convexo quando cada lado de um polígono é também lado de um, e apenas um, outro polígono e, além disso, o plano que contém um desses polígonos deixa todos os outros em um mesmo semi-espaco, ou seja: Se cada plano, que contém cada face de um poliedro, deixa as demais faces do mesmo lado, em relação a ele, então o poliedro é convexo, caso contrário ele é dito não-convexo [39].

Detectado a região do poliedro sendo convexo, passo1, com o clique no botão Corte são acionadas as funções scripts que determinam os dois novos sólidos. Isto é feito substituindo às coordenadas (x_i, y_i, z_i) dos pontos de intersecção e dos vértices na equação do plano de corte (4.6). Os pontos do sólido original que satisfazem a inequação $Ax_i + By_i + Cz_i + D > 0$ pertencem ao primeiro sólido gerado. Analogamente, os pontos do sólido original que satisfazem a inequação $Ax_i + By_i + Cz_i + D < 0$, pertencem ao segundo sólido gerado. Assim, são criadas duas listas de vértices referentes aos dois novos sólidos gerado. Foi feito assim em [27]. Veja implementação:

```

if(maior >0)    // maior >0 são todos os vértices que satisfazem a inequação
{              //  $Ax_i + By_i + Cz_i + D > 0$  e armazenados em solido2

    solido2[RR]=armazena[i];
    RR=RR+1;
}

if(maior<0 )    // maior<0 são todos os vértices que satisfazem a inequação
{              //  $Ax_i + By_i + Cz_i + D < 0$  e armazenados em solido4

    solido4[QQ]=armazena[i];
    QQ=QQ+1;
}

```

Quadro 4.4: Algoritmo que armazena os vértices dos dois novos sólidos gerados após o corte

Logo após esse passo insere-se os pontos de intersecção nesses dois vetores, pois o plano de corte se transforma na nova face desses dois sólidos gerados.

Veja os exemplos nas Figuras 4.13 e 4.14:

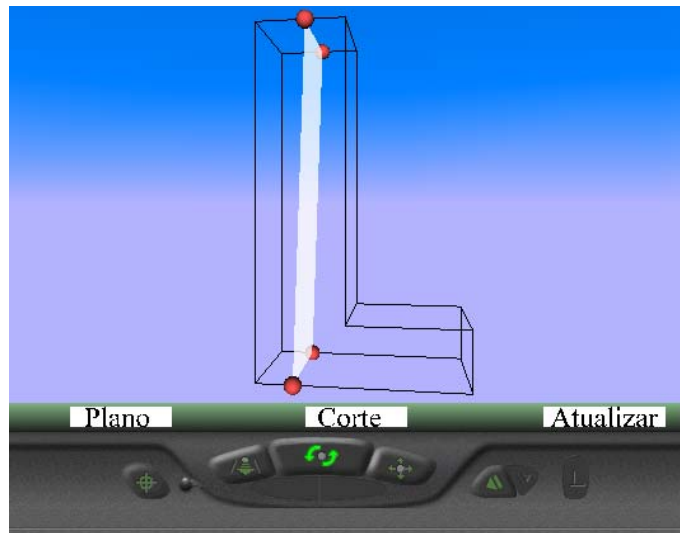


Figura 4. 13: Plano passa na região convexa



Figura 4. 14: Corte gerando dois poliedros

A região exterior do poliedro é detectada quando existir ao menos um ponto médio que pertence a área que limita-se através das coordenadas dos pontos que estão exterior ao polígono como por exemplo o ilustrado na Figura 4.15, o ponto E . Para cada poliedro restringe-se a área das coordenadas que estão na região exterior ao polígono.

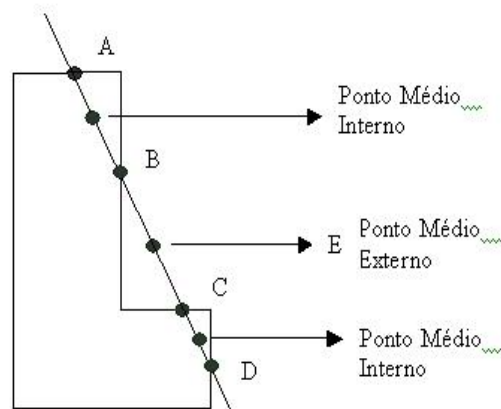


Figura 4. 15: Ponto médio externo E

Se existir algum ponto médio que encontra-se na área externa, então este plano passa pela região não-convexa. Detectado essa região, passo 2, será feita uma análise dos vértices dos pontos que satisfazem a inequação $Ax_i + By_i + Cz_i + D > 0$ e analogamente os pontos que satisfazem a inequação $Ax_i + By_i + Cz_i + D < 0$, bem como dos pontos de intersecção. Aqui inicia-se o processo de extensão do corte para poliedros não-convexos através dessa análise apresentada a seguir.

Primeiro, separa-se os pontos de intersecção que pertencem as faces cuja coordenada $z=1$ e $z=-1$ em dois vetores, logo em seguida coloca-se estes pontos em ordem decrescente, ou crescente para calcular os pontos médios dois a dois, então armazena-se estes pontos em vetores para determinar se pertencem ou não a área externa da face não-convexa, a qual restringe-se através das coordenadas.

Veja a implementação:

```

inter1=0;    inter3=0;    inter6=0;

//////////face z==1

for (i=0;i<KK;i++)
{
    if (auxiliar [i][2]==1)
    {
        pol1[inter1]=auxiliar[i]; //armazena os pontos de intersecção da face z=1
        inter1=inter1+1;
    }
}

for (i=0;i<inter1;i++)
{
    pol3[i]=pol1[i][1]; // armazena noutro vetor os pontos de
//intersecção, coloca em ordem decrescente os pontos armazenados
    for ( j=1; j<inter1;j++){
        aju1[0]=pol3[j];
        aju2[0]=pol1[j];
        for (i=j-1; i>=0 && pol3[i] > aju1[0];i--){
            pol3[i+1]=pol3[i];
            pol1[i+1]=pol1[i];}
        pol3[i+1]=aju1[0];
        pol1[i+1]=aju2[0];
    }
}

for (i=0;i<inter1; i++) //calcula os pontos médios dois a dois
{
    pol6[i][0]=(pol1[i][0] + pol1[i+1][0])/2 ;
    pol6[i][1]=(pol1[i][1] + pol1[i+1][1])/2 ;
    pol6[i][2]=(pol1[i][2] + pol1[i+1][2])/2 ;

    inter3=inter1-1;
}

for (i=0;i<inter3; i++)
{
    pol6Medio[i]=pol6[i]; //verificar se os pontos estao no interior ou exterior
    if ( pol6Medio[i][0]>0.5 && (pol6Medio[i][1]<2) &&
(pol6Medio[i][1]>-2) )
        inter6=inter6+1; } //conta os pontos médios pertencentes ao exterior
//////////fim face z==1

```

Quadro 4.5: Algoritmo que determina na face cujas coordenadas $z=1$ pontos que são exteriores a essa face.

```

inter2=0;    inter4=0;    inter5=0;
////////// inicio face z=-1 mesmos procedimentos acima
    for (i=0;i<KK;i++)
    {
        if (auxiliar [i][2]==-1){
            pol2[inter2]=auxiliar[i];
            inter2=inter2+1;
        }
    }
    for (i=0;i<inter2;i++)
        { pol4[i]=pol2[i][1]; }
        //coloca em ordem decrescente
    for ( j=1; j<inter2;j++){
        aju3[0]=pol4[j];
        aju4[0]=pol2[j];
        for (i=j-1; i>=0 && pol4[i] > aju3[0];i--){
            pol4[i+1] =pol4[i];
            pol2[i+1]= pol2[i];}
        pol4[i+1]=aju3[0];
        pol2[i+1]=aju4[0];
    }
    for (i=0;i<inter2; i++)
        {
            pol7[i][0]=(pol2[i][0] + pol2[i+1][0])/2 ;
            pol7[i][1]=(pol2[i][1] + pol2[i+1][1])/2 ;
            pol7[i][2]=(pol2[i][2] + pol2[i+1][2])/2 ;
            inter4=inter2-1; }
    for (i=0;i<inter4; i++)
        {
            pol7Medio[i]=pol7[i]; //verificar se os pontos estao no interior ou exterior
            if ( pol7Medio[i][0]>0.5 && (pol7Medio[i][1]<2) && (pol7Medio[i][1]>-
2))
                inter5=inter5+1;    } //conta os pontos médios que pertencem ao
exterior

//////////fim face z=-1

```

Quadro 4.6: Algoritmo que determina na face cujas coordenadas $z=-1$ pontos que são exteriores a essa face.

Caso seja determinado pelo sistema que a quantidade de ponto médio que pertence a região exterior ao poliedro é diferente de zero determina-se que a região é não-convexa, para isso tem-se os seguintes procedimentos :

a) Separa-se os pontos de intersecção em dois ou mais vetores. Precisa-se mandar os pontos de intersecção para o VRML ligar de forma que se torne dois ou mais semi-planos de corte, de acordo com a quantidade de poliedros que podem ser gerados.

b) Caso seja um único ponto médio externo determinado pelo sistema separa-se os pontos de intersecção em dois novos vetores, se encontrar dois pontos médios externos, separa-se os pontos de intersecção em três novos vetores e assim por diante para criar as novas faces a serem geradas. Essas novas faces são determinadas através do plano de corte.

c) Caso determine um único ponto médio externo, todos os pontos de intersecção que estão acima desse ponto médio armazenam-se em um vetor e todos os pontos de intersecção que estão abaixo desse ponto médio armazenam-se noutro vetor para criar as novas faces.

d) Caso determine mais de um ponto médio externo, separa-se os pontos do plano de corte limitando-se a esses pontos médios.

A partir dessas novas faces geradas determinam-se os pontos de máximo e de mínimo de cada uma delas. Através desses pontos de máximo e mínimo é que limita-se os pontos para a geração dos novos sólidos.

Assim, são criadas três ou mais listas de vértices referentes aos três ou mais novos sólidos gerados. Entretanto, a conectividade entre os vértices deve ser estabelecida, pois a mesma é perdida durante o processo de classificação dos vértices pertencentes aos novos sólidos. Isto porque a conectividade era conhecida pelo VRML apenas para o sólido original. Feita a ordenação de cada face através do algoritmo ilustrado na Figura 4.9, basta ligá-los.

Estes novos poliedros são definidos através de suas faces e não através de suas arestas como no trabalho de [27].

Primeiro separa-se as coordenadas dos vértices em planos fixos (faces), depois é só ordená-los, conforme o algoritmo apresentado na Figura 4.9. Concluída a ordenação das coordenadas da face, pode-se transportá-los juntamente com os índices para o ambiente virtual, através de uma função do JavaScript, pois o grafo de cena foi modificado. No início, o grafo de cena se encontrava assim:

```
DEF Solido1 Transform{
  Children[]
}
DEF Solido3 Transform{
  Children[]
}
DEF Solido5 Transform{
  Children[]
}
```

Como se pode ver, os campos filhos Sólido1, Sólido3 e Sólido5 ou mais estão em aberto, ou seja, não possuem parâmetros pré-definidos, os dados serão atualizados através do *JavaScript*, e transportados para o ambiente virtual através da função *createVrmlFromString*. Este novo children é todo criado no Núcleo Gerenciador, contendo os vértices ordenados e os seus respectivos índices, isto para cada face que fará parte dos novos sólidos, pois iremos montar os sólidos face por face.

Logo em seguida, adiciona-se ao ambiente através da função do VRML *Solido1.addChildren*, *Solido3.addChildren* e *Solido5.addChildren*, e segue através do comando *ROUTE* do VRML para o campo em aberto, preenchendo-o, veja abaixo as rotas como estão na implementação:

```
ROUTE SCRIPT.new_f1 TO Solido1.children
ROUTE SCRIPT.new_f3 TO Solido3.children
ROUTE SCRIPT.new_f11 TO Solido5.children
```

Veja os sólidos no momento do corte, Figura 4.16:



Figura 4. 16: Momento do corte

Cada sólido possuem sensores que realizam movimentos de rotação e translação para obter uma melhor visualização de todas as suas faces, conforme Figura 4.17. E as rotas que mandam transladar e rotacionar os sólidos cortados estão logo a seguir:



Figura 4.17: Sólidos separados

```
ROUTE solido1sensor.rotation_changed TO Solido1.rotation
ROUTE solido3sensor.translation_changed TO Solido1.translation
ROUTE solido5sensor.rotation_changed TO Solido3.rotation
ROUTE solido6sensor.translation_changed TO Solido3.translation
ROUTE solido51sensor.rotation_changed TO Solido5.rotation
ROUTE solido52sensor.translation_changed TO Solido5.translation
```

O terceiro botão Atualizar permite que o mundo seja carregado novamente para novos cortes. Usamos para isso um comando do VRML denominado Anchor, que cria um hyperlink com outras mídias, neste nosso caso utiliza-se o próprio ambiente virtual para ser reutilizado.

No item a seguir tem uma página com os sólidos, passando o plano de corte e outra o corte feito. Tem-se o E, um C, e um T.

4.3 Outros Estudos de Caso:

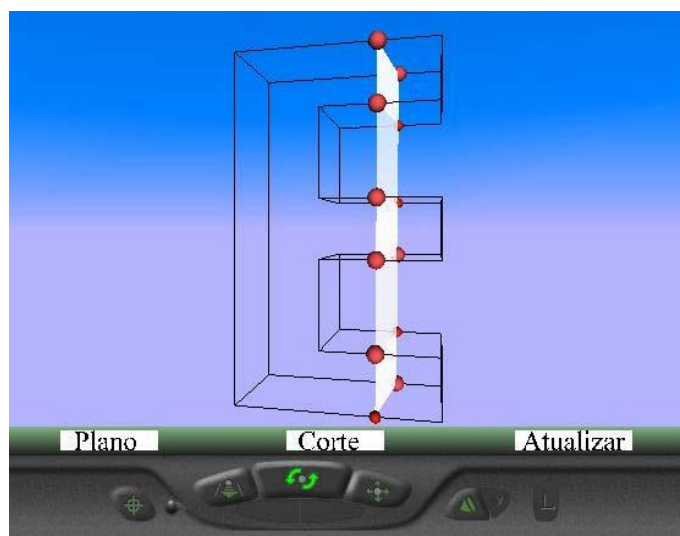


Figura 4. 18: Traçado do plano para o sólido E



Figura 4. 19: Corte efetuado para o sólido E

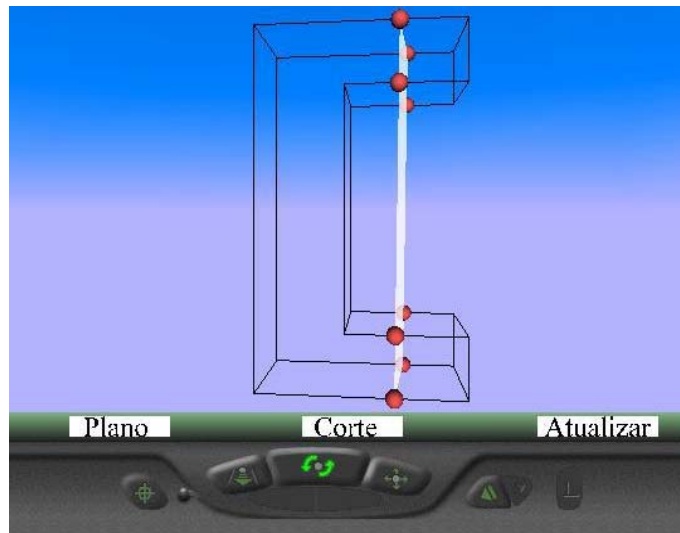


Figura 4. 20: Traçado do plano para o sólido C

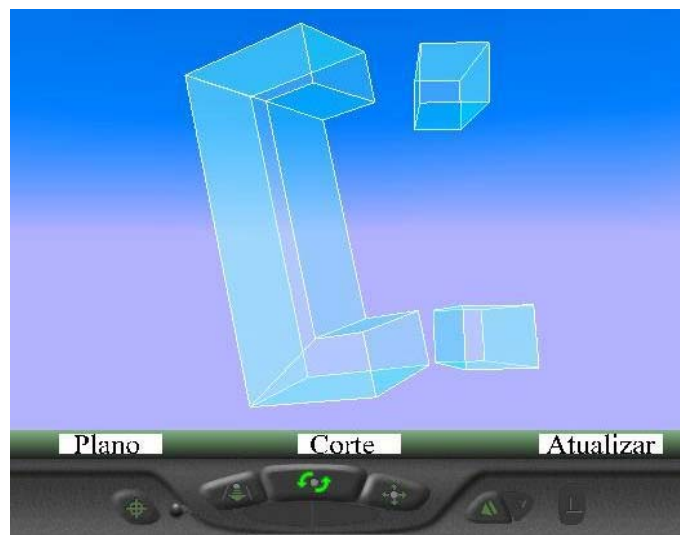


Figura 4. 21: Corte efetuado para o sólido C

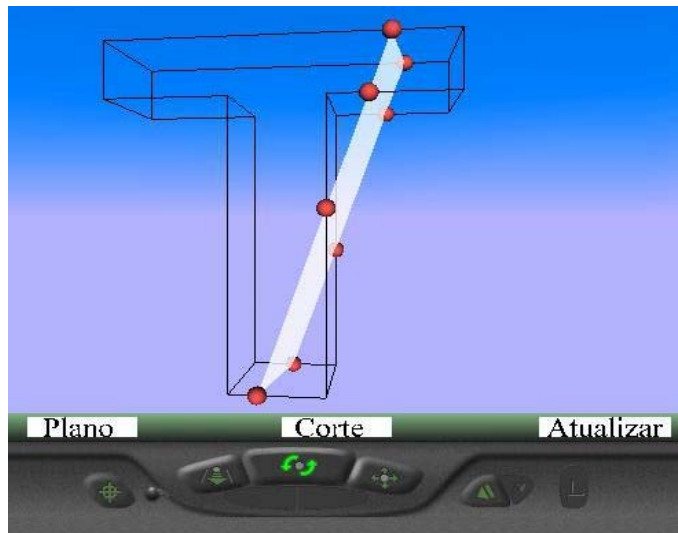


Figura 4. 22: Traçado do plano para o sólido T



Figura 4. 23: Corte efetuado para o sólido T

4.4 Sumário e Conclusões:

Neste capítulo foi apresentado a extensão do trabalho, corte de poliedros convexos para corte de poliedros não-convexos usando técnicas de Realidade

Virtual por meio da linguagem VRML usada na modelagem e JavaScript utilizada para interação do ambiente virtual.

Diversos algoritmos foram utilizados. Dentre eles: Intersecção do plano de corte com o poliedro, utilização de uma inequação do plano para separar os vértices em dois vetores para a divisão do sólido, isso se o plano passar pela região convexa [27]. O algoritmo que ordena pontos cujo contorno forma um polígono convexo [27]. Outro algoritmo utilizado foi o de classificar o plano como pertencendo à região convexa ou não-convexa através da definição de pontos externos ou internos ao poliedro. Logo após é feita uma análise dos pontos da inequação do plano para fazer uma terceira ou mais divisão do sólido no caso de ser detectado que o plano passa pela região não-convexa, divide-se o poliedro em três ou mais outros sólidos, restringindo para isso aos pontos de máximo e de mínimo das novas faces que são geradas através da equação do plano.

A interação proposta foi a mesma utilizada no trabalho [27], pois o objetivo deste trabalho foi propor a extensão do corte para poliedros não-convexos.

Na geração do plano de corte, todos os planos foram determinados de faces convexas, pois o algoritmo de ordenação, ordena os pontos apenas para polígonos convexas.

No próximo capítulo, trata-se da avaliação e descrição dos resultados, bem como as limitações do sistema.

CAPÍTULO V

5. RESULTADOS E LIMITAÇÕES DO SISTEMA

5.1 Introdução

Este capítulo tem por objetivo mostrar o funcionamento do sistema protótipo proposto neste trabalho, além disso descreve-se o processo de avaliação desse software.

A avaliação do sistema foi feita por um conjunto de professores e de alunos, através da utilização, observação da ferramenta e aplicação de um questionário preenchido pelos mesmos.

As limitações do sistema proposto e sugestões para resolver essas limitações são discutidas.

5.2 Funcionamento do Sistema

O sistema protótipo foi introduzido em uma página HTML, onde o usuário encontra explicações para auxiliá-lo a executar a ferramenta. Nesta página, apresenta-se vários botões, no qual estão disponíveis os poliedros não-convexos. Veja a Figura 5.1 que mostra a página inicial do sistema.



Figura 5. 1: Página HTML inicial do sistema

Nesta página, o usuário se depara com oito botões, no primeiro Home é exibido a página inicial conforme ilustra a (Figura 5.1). Os próximos LetraC, LetraL e LetraT são os poliedros não-convexos que o usuário poderá escolher para realizar os cortes. O botão Vídeo mostra como se realiza o corte através de um vídeo. No botão Plugging tem um link com a opção de instalar o Cosmo Player. O botão Contato localiza-se o email do orientador e da aluna responsável pelo protótipo. Finalmente, no botão Ajuda descreve-se os passos para utilizar este software.

Ao selecionar o poliedro não-convexo, clicando por exemplo no botão LetraC o ambiente virtual é aberto em um browser acompanhado do plug-in Cosmo Player, visualiza-se o poliedro e o painel de controle do sistema, veja Figura 5.2.

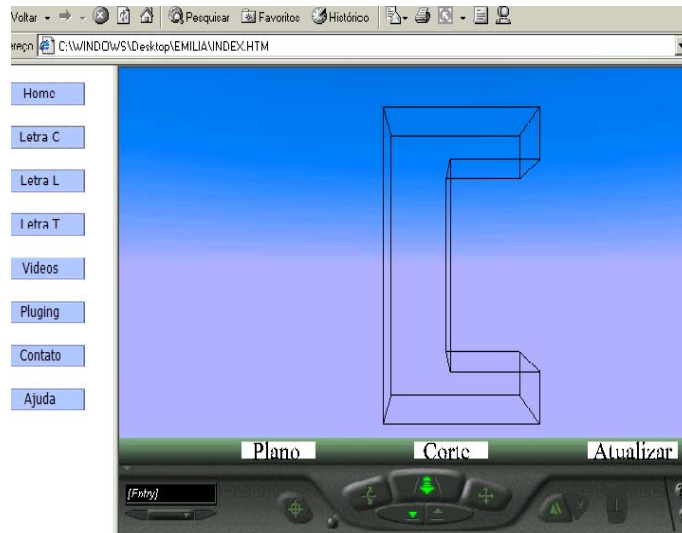


Figura 5. 2: Poliedro não-convexo

O usuário pode navegar pelo ambiente virtual e interagir com os poliedros através do plug-in. Primeiramente o usuário escolhe aleatoriamente três pontos não-colineares, logo em seguida clica no botão Plano para gerar o plano de corte. Veja a ilustração na Figura 5.3.

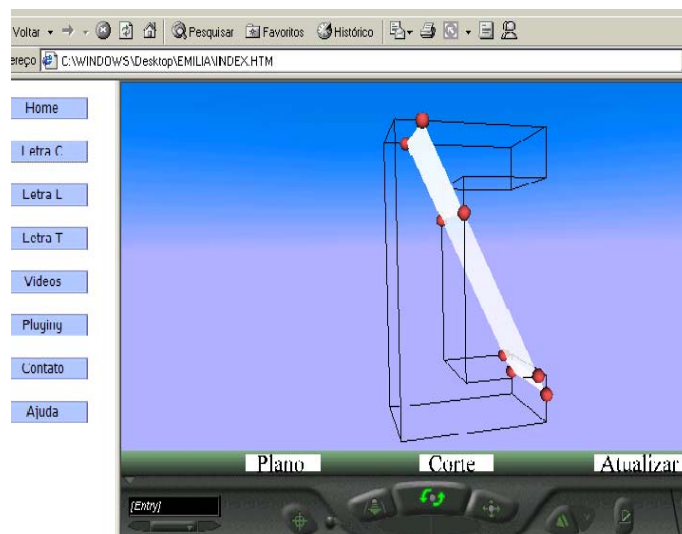


Figura 5. 3: Plano de corte no sólido C

Ao gerar o plano de corte, o poliedro será cortado. Para isso o usuário deve clicar no segundo botão Corte. Quando se aciona esse botão dois ou três novos poliedros serão construídos no ambiente virtual. Estes novos poliedros tem sensores, que permitem arrastá-los pelo ambiente virtual, através do manuseio com o mouse. A Figura 5.4 ilustra os novos poliedros gerados após o corte.

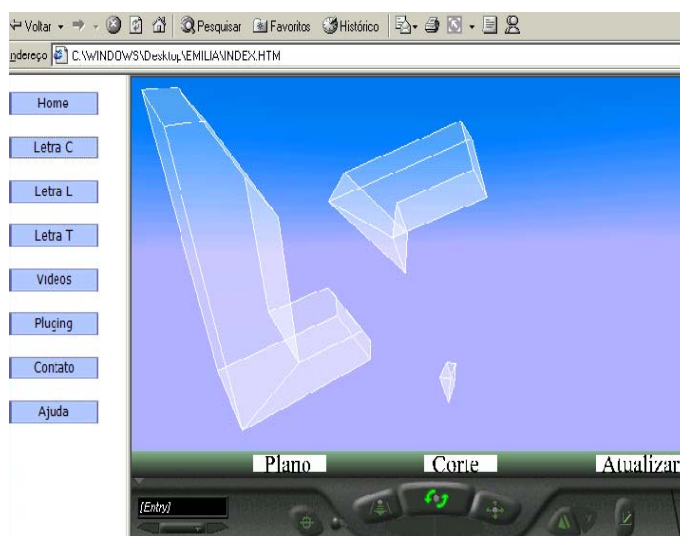


Figura 5. 4: Corte efetuado para o sólido C

O usuário tem a opção de realizar novos cortes, para isso deve clicar no terceiro botão Atualizar, inicializa-se o sistema para a escolha de novos pontos para novos cortes. Uma outra opção é o usuário escolher outro poliedro para realizar o corte, clicando no botão esquerdo da página.

5.3 Avaliação do sistema

Na avaliação do sistema protótipo, foram realizadas experimentos com potenciais usuários entre professores e alunos (da graduação, mestrado e doutorado) das ciências exatas que totalizaram 31 usuários.

Na apresentação do sistema a esses usuários, explicou-se o objetivo do sistema proposto e disponibilizou-o para a interação e navegação. Para exemplificar o sistema, na página inicial tem um vídeo, clicando no botão Vídeo que demonstra o funcionamento do mesmo, no botão Ajuda explica-se como utiliza o software.

Então, o usuário inicia-se o processo de exploração do ambiente virtual. Logo após utilizar o sistema, os usuários preenchem o questionário proposto.

Apresenta-se a seguir os itens avaliados pelos usuários, bem como um gráfico comparativo da avaliação juntamente com alguns comentários baseado nos mesmos.

- Quanto à Finalidade do seu uso como ferramenta educacional em disciplinas que tratam do assunto em questão:

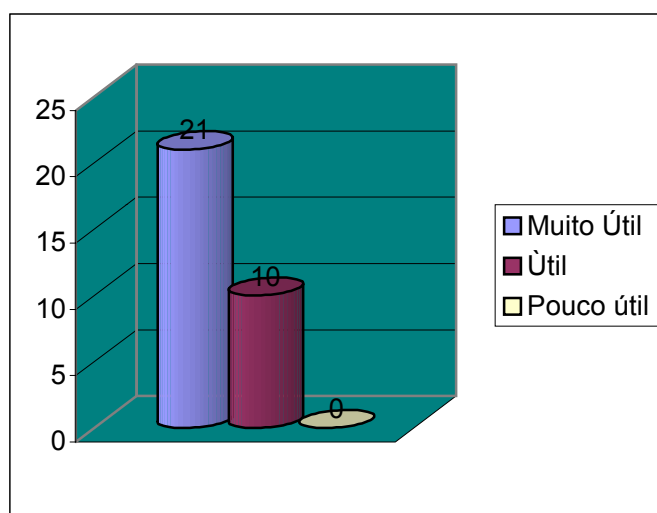


Figura 5. 5: Avaliação do sistema quanto a finalidade do seu uso como ferramenta educacional.

De acordo com o gráfico, 21 pessoas consideraram o sistema muito útil, pelo fato de facilitar a visão espacial, possibilitando um aprendizado diferente e simples através dessa interação que os métodos tradicionais não oferecem.

Dentre as 10 pessoas que consideraram o sistema útil as justificativas foram semelhantes as anteriores, apenas um aluno justificou que além de facilitar a visão espacial, os recursos oferecidos por este software são limitados.

- Quanto à interface com o Usuário:

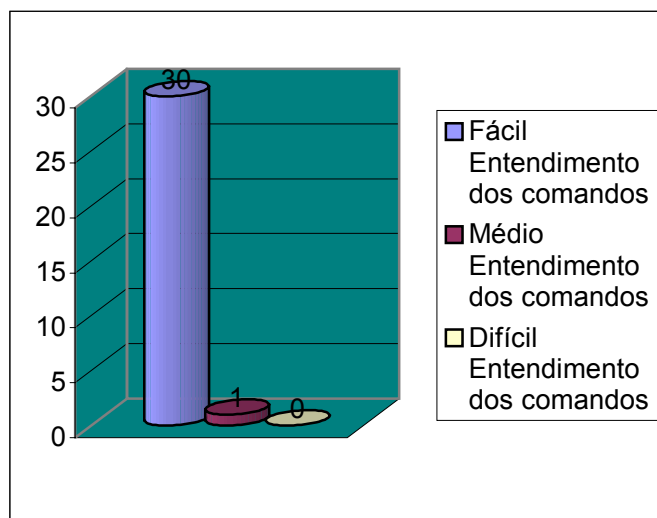


Figura 5. 6: Avaliação do sistema quanto à interface com o Usuário.

Quanto à Interface com o usuário, através do gráfico percebe-se que 30 pessoas consideraram fácil entendimento dos comandos, pois tem um botão com a ajuda explicando a utilização do software, além disso após a primeira tentativa aprende-se muito rápido a utilizá-lo. O usuário que respondeu que é médio o entendimento dos comandos não justificou.

- Quanto a facilidade de uso do sistema:

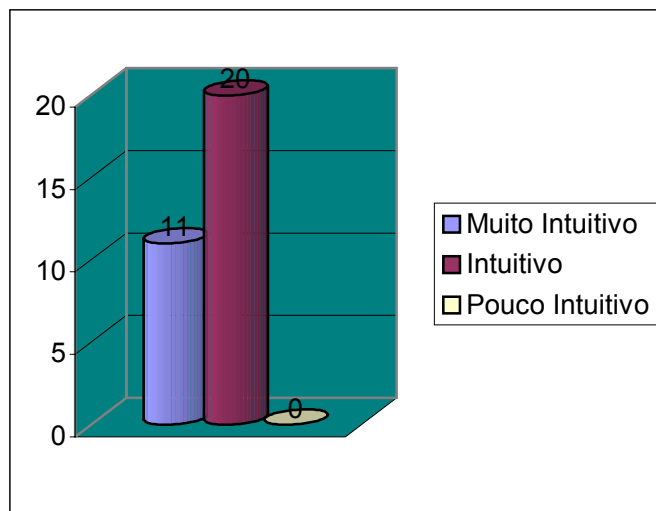


Figura 5. 7: Avaliação quanto a facilidade de uso do sistema.

Em relação a facilidade de uso do sistema 11 usuários responderam que é muito intuitivo e as justificativas foram que o usuário aprende a utilizar o sistema de uma maneira rápida, pois os comandos são muito simples e um aluno acrescentou que os comandos são semelhantes aos padrões do windows.

Os 20 usuários que responderam que é intuitivo entre as justificativas alguns responderam que são simples, poucos comandos, após a primeira experiência a adaptação é fácil. Outra justificativa é que seria interessante aparecer informações sobre a função de cada botão quando se posiciona o mouse sobre ele. Outra é que os cliques podiam estar como avisos através de uma janela em relação ao duplo clique do mouse na escolha do segundo e terceiro pontos. E uma outra justificativa é em relação a não poder remontar as figuras cortadas.

- Quanto aos recursos do programa a experiência proposta foi:

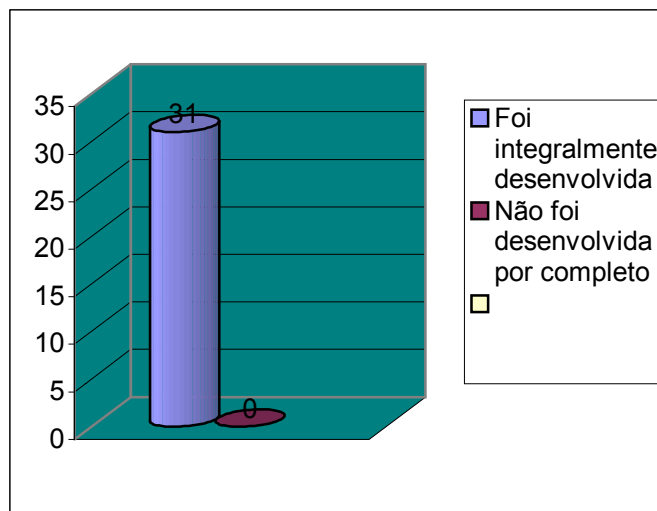


Figura 5. 8: Avaliação quanto aos recursos do programa.

Todos os usuários responderam que a experiência proposta foi integralmente desenvolvida entre as justificativas foi que permite a formação de qualquer figura geométrica a partir dos pontos selecionados pelo usuário, permite a visualização de vários ângulos diferentes e apesar de ser uma estrutura simples é de grande valia para aprendizagem de Geometria Espacial e além disso todos os recursos do programa foram utilizados.

- Quanto aos recursos do programa os objetivos disponíveis permitem:

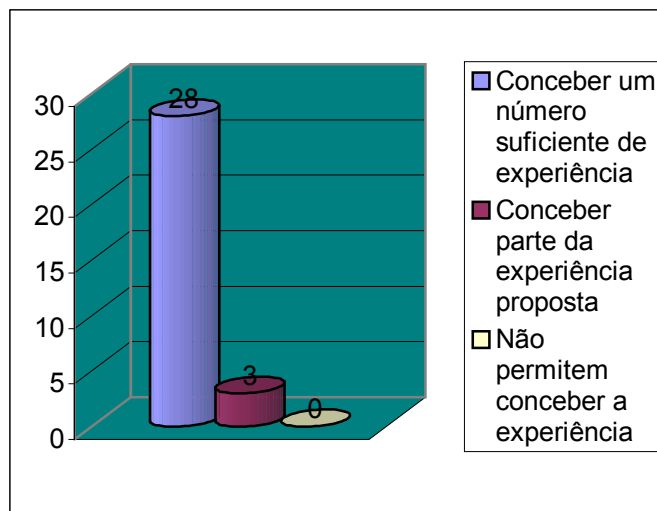


Figura 5. 9: Avaliação quanto aos objetivos disponíveis.

Os 28 usuários que responderam que concebeu um número significativo de experiência as sugestões estarão logo a seguir. Os 3 usuários que responderam que conceberam parte da experiência proposta as sugestões é que poderia cortar novamente a parte cortada e a outra sugestão é inserir figuras com furos.

- O programa permitiu a aquisição de informações úteis sobre Geometria Espacial, em especial Corte de Poliedros não-Convexos?

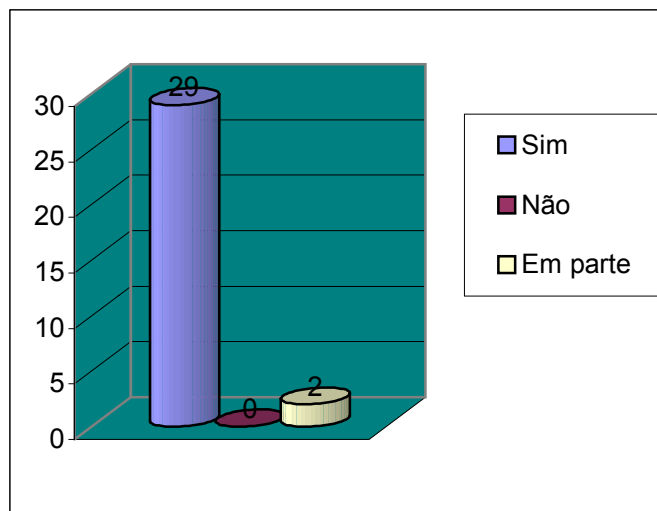


Figura 5. 10: Avaliação das informações úteis sobre Geometria Espacial.

Os 29 usuários que responderam sim, suas justificativas: através do programa pode montar novas figuras geométricas, visualiza-se os poliedros de vários pontos de vista, ressalta-se a importância da visualização espacial.

Os 2 usuários que responderam em parte não justificaram.

Dentre as várias sugestões que foram apresentadas pelos usuários destacam-se:

- Permitir que o aluno crie seu próprio objeto para cortar.
- Montar as peças após o corte.
- Mostrar as equações do plano de corte em tempo real.
- Colocar mensagens na tela comunicando o seguimento dos passos a serem utilizados.
- Aplicação de curvas.
- Mudar as cores das partes após o corte.
- Inserir figuras com furos.
- Cortar o cortado várias vezes.

- Inserir cálculo de área, comprimento.

De acordo com o exposto acima conclui-se que o software mostrou ser bastante eficiente na sua proposta inicial, de auxiliar nas aulas de Geometria Espacial. Pode também ser aprimorado para auxiliar nos cálculos de áreas, volume, como também na criação de novos sólidos pelo próprio aluno.

5.4 Limitações do Sistema

As limitações do software cita-se:

- Possibilidade de cortar apenas poliedros (não corta figuras com curvas).
- Não existe a opção de cortar os novos sólidos gerados novamente.
- Não tem a possibilidade de montar os sólidos novamente.
- Em relação ao plug-in Cosmo Player, observou que este tem compatibilidade até o Windows ME. Com relação ao plug-in Cortona precisa-se fazer algumas alterações como, ao invés de iniciar os sólidos apenas com arestas, temos que iniciá-lo com as faces, pois o sensor que reconhece as arestas com o plug-in Cosmo Player não as reconhece com o plug-in Cortona, mas com as faces se reconhece e sendo assim funciona também. O plug-in Cortona não tem problemas de compatibilidade com o Windows XP.

5.5 Considerações Finais

Este capítulo apresentou os resultados e limitações do sistema proposto, que oferece recursos para o usuário visualizar e realizar cortes em poliedros não-convexos através da interação direta com os mesmos.

Através da avaliação feita os usuários contribuíram com sugestões para aprimorar o sistema. Destaca-se que o software mostrou ser bastante eficiente no auxílio de aulas de Geometria Espacial, inclusive teve elogios no que diz respeito a ser um software voltado para Internet, a baixo custos, mostrando ser o futuro em relação a ambientes de desenvolvimento, pois aplicações voltadas a Internet tendem a dominar e crescer. Passos importantes como estes são de vital importância no desenvolvimento tecnológico de ambientes educacionais.

Embora o sistema apresente limitações, foram levantadas várias sugestões para a melhoria do software, para trabalhos futuros.

No próximo capítulo, serão apresentadas as conclusões, e as sugestões para os trabalhos futuros que poderão dar continuidade a essa pesquisa.

CAPÍTULO VI

6. CONCLUSÃO E TRABALHOS FUTUROS

6.1 Introdução

Esta dissertação teve como proposta generalizar o corte de poliedros convexos para cortes de poliedros não-convexos, usando as linguagens VRML e JavaScript.

Este capítulo tem por objetivo apresentar as principais contribuições científicas e propor sugestões para dar continuidade como trabalhos futuros a esta pesquisa.

6.2 Conclusões

Com o objetivo de generalizar o corte para poliedros não-convexos as seguintes considerações foram exploradas:

- Na construção do ambiente virtual composto por poliedros decidiu-se pela linguagem VRML integrada a linguagem JavaScript para implementar a arquitetura proposta nesta dissertação.

- Essa integração se deve ao fato da linguagem VRML não proporcionar atualizações automáticas de ambientes virtuais após a interação do usuário no sistema, assim o uso da linguagem JavaScript permite atualizar o ambiente virtual de maneira mais interativa e de acordo com a intenção do usuário.

- O desenvolvimento de algoritmos genéricos que possibilitassem o corte de poliedros em três ou mais poliedros.

- Com relação aos plug-in Cosmo Player e Cortona observou-se que existe uma pequena diferença na forma de apresentar os poliedros na sua forma inicial, com o Cosmo Player na forma de arestas e com o Cortona na forma de faces.

Das conclusões cita-se:

- Este trabalho contribui para uma interação mais direta, genérica e interativo com o usuário, pois modifica-se suas formas pela manipulação dos mesmos.
- Acredita-se que as técnicas de interação apresentadas contribuam para o aprendizado da visualização na Geometria Espacial.
- O algoritmo desenvolvido atende a uma significativa quantidade de estudo: poliedros convexos e não-convexos.
- É um sistema compatível com os protocolos da Internet, sendo fácil a distribuição e execução do sistema.

6.3 Trabalhos Futuros

Sugere-se para trabalhos futuros:

- Criação de seu próprio poliedro, para a realização de cortes.
- Desenvolver algoritmos que suportem o corte dos novos sólidos que são gerados de uma maneira iterativa.
- Estudar as principais diferenças entre os Plug-In Cortona e Cosmo Player para se ter o sistema limitado a uma determinada plataforma.
- Utilizar a Realidade Virtual Imersiva para cortar estes poliedros.

Outras sugestões levantadas pelos avaliadores para dar continuidade a esta pesquisa:

- Montar as peças após o corte, ou seja juntar os poliedros novamente.
- Mostrar as equações do plano de corte em tempo real.
- Colocar mensagens na tela comunicando o seguimento dos passos a serem utilizados.
- Aplicação do corte para figuras com curvas.

- Ter a opção de mudar as cores das partes após o corte.
- Cortar figuras com furos.
- Inserir a possibilidade de fazer cálculos de área do plano de corte e de volume nos sólidos cortados.
- Ao posicionar o mouse em cada botão na interface as informações sobre a função de cada um deles deveria aparecer.

Estas técnicas apresentadas mostraram-se baratas e eficazes na geração dos novos poliedros para o auxílio no ensino de Geometria Espacial. Sendo disponível via WEB não necessitando de computadores potentes para serem exibidos.

O software mostrou ser eficiente na sua proposta inicial, sendo assim contribuiu para o processo cognitivo do ensino/aprendizagem de Geometria Espacial.

7. REFERÊNCIAS BIBLIOGRÁFICAS:

- [1] CAMACHO A . S. M. Maria de Lourdes Realidade Virtual e Educação Disponível em <http://phoenix.sce.fct.unl.pt/simposio/30.htm> Último acesso 11/10/2006.
- [2] PINHO S. Márcio VII Simpósio Brasileiro de Informática na Educação 1996 Belo Horizonte MG - Realidade Virtual como Ferramenta de Informática para a educação- Disponível em <http://grv.inf.pucrs.br/Pagina/Educa/educa.htm> Último acesso 29/09/2006.
- [3] Escola TRI-Legal Vendruscolo Franciele 2005 - Disponível em http://www.ricesu.com.br/colabora/n9/artigos/n_9/id03c.htm Último acesso 29/03/2007.
- [4] KIRNER, C. Sistemas de Realidade Virtual. Apostila publicada pelo Departamento de Computação da Universidade Federal de São Carlos. São Carlos - SP, Brasil, 1997.
- [5] PAUSCH, R.;PROFFIT, D.;WILLIAMS, G.Quantifying Immersion in Virtual Reality In: Proceedings of the 1997 – ACM Siggraph annual conference on Computer Graphics, 1997.
- [6] SHERMAN, R. W. et al. Experiences with Virtual Reality applications. In: PROCEEDINGS OF THE 24TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES – **SIGGRAPH’ 97**. USA, 1997.
- [7] PINHO, M. Interação em Ambientes Tridimensionais. Tutorial do 3rd WORKSHOP ON VIRTUAL REALITY – **WRV’ 2000**, Gramado, RS, Outubro/ 2000.
- [8] CARDOSO, A.; LAMOUNIER E.; TORI R. Realidade Virtual no Ensino de Física a Distância e Presencial In: XI Simpósio Brasileiro de Informática na Educação – **SBIE’2000**, Maceió, Alagoas, nov.2000.

- [9] BYRNE, C. Water o Tap: The of Virtual Reality as an Educational Tool. Washington,1996. Tese(Doutorado)- University of Washington.
- [10] PASQUALOTTI, Adriano, FREITAS, Carla Maria Dal Sasso. Ambientes VRML para o Ensino-Aprendizagem de Matemática: Modelo Conceitual e Protótipo. In:3th Workshop on Virtual Reality,2000, Gramado, RS. Anais 3th SBC Workshop on Virtual Reality: Gramado, SBC, 2000,pp.65-76.
- [11] KIRNER, Claudio, SEMENTILLE, Antonio Carlos, Brega, José R. Ferreira. Um Ambiente Integrado para Reconhecimento de Gestos e Posturas Baseado na Captura de Movimentos. In 4th Symposium on Virtual Reality, 2001, Florianópolis, SC. Anais 4th SBC Symposium on Virtual Reality – Florianópolis: UFSC, SBC,2001,pp. 79-90.
- [12] BRAGA, Mariluci. Realidade Virtual e Educação. Revista de Biologia e Ciências da Terra. Volume 1 nº 1 2001 Disponível em <http://www.uepb.edu.br/eduep/rbct/sumarios/pdf/realidadevirtual.pdf> Último acesso em 08/01/2007.
- [13] DOTTO, Bruno Reis. Realidade Virtual e Suas Aplicações, outubro 2000 . Disponível <http://209.85.165.104/search?q=cache:Yg5Jt0hiVi0J:apostilas.netsaber.com.br/apostilas/825.doc+imersao+itera%C3%A7%C3%A3o+navega%C3%A7ao&hl=pt-BR&ct=clnk&cd=7&gl=br>. Último acesso 29/03/2007.
- [14] Tecnologias – Computadores Virtus Realitatis por Lu Ana. Disponível em http://www.cultkitsch.org/tecno/realidade_virtual_files/realidade_virtual.htm Último acesso 28/03/2007.
- [15] PINHO, Márcio Serolli: Realidade Virtual como Ferramenta de Informática na Educação. In: Simpósio Brasileiro de Informática na Educação, 1996, Belo Horizonte, MG. Anais SBC - Simpósio Brasileiro de Informática na Educação: Belo Horizonte, SBC, 1996.

- [16] KAESTNER, C, EBERSPÄCHER, H . Arquitetura de um Sistema de Autoria para Construção de Tutores Inteligentes Hipermídia. In : VIII Simpósio Brasileiro de Informática na Educação, São José dos Campos – SP, 1997, pp. 163-180.
- [17] KELLER, Rodrigo, SCHREIBER, Jacques. GEO-3D: A Realidade Virtual como Suporte ao Ensino da Geometria Espacial. In: 2th Workshop on Virtual Reality, 1999, Marília, SP. Anais 2th SBC Workshop on Virtual Reality: Marília, SBC, 1999, pp.182-193.
- [18] SANTAROSA, Lucila, O Computador na Avaliação Formativa, Porto Alegre/RS: UFRGS,p.165,1990.
- [19] SILVA, André Tavares, OSÓRIO, Fernando Santos. Modelagem de objetos 3D em VRML: Uma implementação multiplataforma orientada ao ensino. In: 3th Workshop on Virtual Reality,2000,Gramado, RS. Anais 3th SBC Workshop on Virtual Reality: Gramado, SBC, 2000, pp 43-53.
- [20] TAROUCO F. Fabrício - Protótipo Lego em VRML - Disponível em: http://cumincades.scix.net/data/works/att/sigradi2003_132.content.pdf#search=%22lego%20vrml%22 Último acesso 28/03/2007.
- [21] NAKAMOTO, Paula Teixeira, CARDOSO, Alexandre, LAMOUNIER Jr., Edgard. Utilização de Mapas Conceituais na Construção de Ambientes Virtuais de Aprendizagem. Dissertação (Mestrado em Ciências)- Faculdade de Engenharia Elétrica-UFU, Uberlândia, 2005.
- [22] SILVA, Luciano Ferreira, CARDOSO, Alexandre, MENDES Elise. Associando Realidade Virtual não-imersiva e Ferramentas Cognitivas para o Ensino de Física. Dissertação (Mestrado em Ciências)- Faculdade de Engenharia Elétrica-UFU, Uberlândia, 2006.
- [23] FREITAS, Marlene Roque, SILVA, Luciano Ferreira, LAMOUNIER Jr. Edgard, CARDOSO, Alexandre, Desenvolvimento de um Software para Educação

Matemática utilizando Jogos e Realidade Virtual. 1º Workshop de Aplicações em Realidade Virtual, Uberlândia, MG, 2005.

[24] FREITAS, Marlene Roque, Lamounier Edgard Jr., SILVA Luciano F.; CARDOSO, Alexandre. Proposta de integração de Jogos e Realidade Virtual no Aprendizado de Conceitos Matemáticos - Projeto LUDOS TOP – XVII Simpósio Brasileiro de Informática na Educação – poster- Brasília DF, novembro 2006.

[25] BARRETO, Marina Menna, GRAVINA Maria Alice. Como construir figuras congruentes? Disponível em : <http://www.mat.ufrgs.br/~edumatec/atividades/ativ12/cabri.htm> Último acesso 28/03/2007.

[26] NUNES, Eunice P. Santos, LAMOUNIER Jr. Edgard, CARDOSO, Alexandre. Uma Aplicação das Técnicas de Realidade Virtual Não-Imersiva na Visualização e Corte de Superfícies. Dissertação (Mestrado em Ciências)- Faculdade de Engenharia Elétrica-UFU, Uberlândia, 2002.

[27] BORGES, Eila Christiane, LAMOUNIER Jr. Edgard, CARDOSO, Alexandre, Proposta de um Ambiente Virtual Interativo como Ferramenta de Suporte no Ensino de Geometria Espacial. Dissertação (Mestrado em Ciências)- Faculdade de Engenharia Elétrica-UFU, Uberlândia, 2002.

[28] Disponível em : <http://www.comciencia.br/especial/inclusao/inc01.shtml> Último acesso 28/03/2007.

[29] SOUZA, Marcio Vieira. As redes de educação e a cidadania. Disponível em http://209.85.165.104/search?q=cache:6svBWm3WqsEJ:www.naya.org.ar/congreso2004/ponencias/marcio_vieira.rtf+informatiza%C3%A7ao+escolas+publica+dados+recentes&hl=pt-BR&gl=br&ct=clnk&cd=2 Último acesso 28/03/2007.

[30] RESENDE Juliana. Portal Educacional. Entrevista. Disponível em : <http://www.educacional.com.br/entrevistas/entrevista0007.asp> Último acesso 28/03/2007.

- [31] PASQUALOTTI, Adriano. A Realidade Virtual como Proposta de Ensino-aprendizagem de Matemática. 3º Workshop Brasileiro de realidade Virtual, Gramado, RS, pág65-67, outubro, 2000.
- [32] Recursos avançados no WWW. Disponível em <http://penta.ufrgs.br/pesquisa/joice/cap4.html>. Último acesso 28/03/2007.
- [33] CASTRO, Elisa Calhau de e MAGALHÃES Léo Pini. Computação Gráfica e Multimídia Utilização do VRML Disponível em: <http://www.dca.fee.unicamp.br/sibgrapi99/vrml/Aula1/aula1.html#introducao> Último acesso 28/03/2007.
- [34] Disponível em <http://www.criarweb.com/artigos/206.php> Último acesso 28/03/2007.
- [35] Editor de Texto. VRMLPAD. Disponível em : <http://www.parallelgraphics.com/products/vrmlpad/download/> Último acesso 28/03/2007.
- [36] Plug-in. COSMOPLAYER. Disponível em : <http://www.karmanaut.com/cosmo/player/> Último acesso 28/03/2007.
- [37] STEINBRUCH, Alfredo, WINTERLE, Paulo, Geometria Analítica -2.ed.-São Paulo: McGraw-Hill,1987
- [38] LIMA, E. L., Carvalho, P. C. P., Wagner, E., Morgado, A. C. *A Matemática do Ensino Médio 4ª* ed. Rio de Janeiro: SBM, 2002. 299p
- [39] BARCELOS, Gilmara Teixeira, BATISTA, Silvia Cristina Freitas. Poliedros. Disponível em <http://www.es.cefetcampos.br/softmat/atividades1/Poly-apostilaetatividades-2005.pdf>. Último acesso 28/03/2007.

APÊNDICE A

VRML - a Web em 3D

Alexandre Cardoso

1. Introdução e breve histórico

VRML (*Virtual Reality Modeling Language*) tem sido aplicada em diversos projetos para concepção de mundos virtuais [ELLIS 1994, CHEN 1997, MATSUBA 1999 CARDOSO et al. 1999] e é uma importante aliada no desenvolvimento de mundos tridimensionais interativos na Web. VRML tem uma história fundamentada na colaboração de diversos pesquisadores e importantes empresas relacionadas com a Computação Gráfica.

A partir de um projeto, iniciado em 1989, na *Silicon Graphics Inc.* por Rikk Carey e Paul Strass, que propunha o desenvolvimento de uma infra-estrutura para aplicações gráficas 3D interativas com duas características básicas: capacidade de criação de uma gama variada de aplicações 3D e a possibilidade de usar este ambiente para construir uma nova interface para 3D, nasce o *Scenario*.

Em 1992, fundamentado nas proposições do *Scenario*, surge o *Iris Inventor 3D*, uma ferramenta fundamentada em C++ que, posteriormente, fundamenta muito da semântica de VRML. A revisão do *Inventor* realizada em 1994 origina o *Open Inventor* caracterizada por sua portabilidade a uma grande variedade de plataformas e baseada no *Open GL* da *Silicon*. O manual de referência da mesma descrevia os objetos e formatos de arquivos que, adequados por *Gavin Bell*, originariam a especificação do VRML 1.0.

Em 1994, *Mark Pesce* e *Tony Parisi* construíram um protótipo de um navegador 3D para a *World Wide Web - WWW*, chamado *Labyrinth*. Um ano depois, *Mark* e

Brian Behlendorf criaram a lista de discussão do VRML e publicaram uma chamada para propostas de uma especificação formal para 3D na *WWW*.

De discussões nesta lista, iniciadas por *Gavin Bell*, usando como ponto de partida o manual do *Inventor* nasce a especificação do VRML 1.0. Em 1995, VRML 1.0 foi alvo de muitas correções e novas discussões, de forma que nasce a proposta de uma segunda especificação, a do VRML 1.1 (ou VRML 2.0).

Aprimorada e capaz de definir comportamentos (com mais interação e animação) para elementos 3D, a nova proposição foi submetida à lista em 1996. As propostas apresentadas pela *Silicon* em colaboração com a *Sony* e com a *Mitra* receberam a grande maioria dos votos e originaram o documento de definição de VRML 2.0. Em agosto de 1996, esta versão foi publicada no *SIGGRAPH'96* em *New Orleans*.

A versão final deste texto, com diversas correções e modificações técnicas foi publicada em Abril de 1997, ficando conhecida como VRML'97 e é a versão em uso até os dias atuais. Suas principais características estão relacionadas com o desenvolvimento de cenários mais realísticos, prototipação (capacidade de encapsular novos de forma a criar novos nós), interação direta com o usuário através de sensores, interpoladores e criação de animações usando *scripts*.

2. Características básicas de VRML

Arquivos que simulam mundos 3D em VRML, são na verdade uma descrição textual, na forma de textos ASCII. Assim, por meio de qualquer processador de textos, um desenvolvedor pode conceber tais arquivos, salvá-los e visualizar resultados no navegador de Internet associado a um *plug-in*. Estes arquivos definem como estão as formas geométricas, as cores, as associações, os movimentos, enfim, todos os aspectos relacionados com a idéia do autor [AMES 1997]. Quando um dado navegador - *browser* - lê um arquivo com a extensão *.wrl*, o mesmo constrói o mundo descrito, ativando um *plug-in* compatível.

De forma simplificada, um arquivo VRML se caracteriza por quatro elementos principais: - o cabeçalho (obrigatório); - os protótipos; - as formas, interpoladores, sensores, scripts e as rotas. Assim, um arquivo VRML pode conter:

- *Header*;
- *Prototypes*;
- *Shapes, Interpolators, Sensors, Scripts*;
- *Routes*.

Nem todos os arquivos contêm todos estes componentes. Na verdade o único item obrigatório em qualquer arquivo VRML é o *header*. Porém, sem pelo menos uma figura, o navegador não exibirá nada ao ler o arquivo. Outros componentes que um arquivo VRML também pode conter, são:

- *Comments*;
- *Nodes*;
- *Fields, field values*;
- *Defined node names*;
- *Used node names*;

O cabeçalho (*header*) é composto pela instrução "#VRML V2.0 utf8" e sua omissão impossibilita o *plug-in* do navegador de ler o arquivo em questão. Os protótipos (*proto*) contêm a definição de novos nós que podem ser usados no arquivo em definição. A seção de descrição de formas (*shapes* etc) apresenta a descrição das formas que serão exibidas no navegador e a seção de rotas (*routes*) contém a definição das trocas de mensagens entre os nós de descrição de formas, interpoladores, sensores e *scripts*.

A concepção de cenários tridimensionais, usando VRML, se baseia na elaboração de uma grafo direcionado acíclico, contendo diferentes ramos - nós - que, associados de forma correta podem ser agrupados ou estarem independentes uns dos

outros. A grande diversidade destes nós (54 pré-definidos), incluindo primitivas geométricas, propriedades de aparência, sons (e propriedades) e vários tipos de nós de agrupamentos, é uma das principais características e qualidades da linguagem.

É permitido reutilização de código através da prototipação, baseada na definição de novos nós (*protos*) que podem ser utilizados por outros arquivos e ativados dentro de um arquivo como um nó externo, sem duplicação de códigos.

A concepção de formas se dá através da associação de elementos 3D geométricos pré-definidos, tais como Cones, Cilindros, Esferas, Caixas etc de atributos variáveis e que podem estar associados a texturas.

A modificação de fundos está possibilitada pelo uso de nós específicos - *backgrounds*, - que permitem simular ambientes diferenciados que se assemelham a condições que variam de um lindo dia de sol, um dia nublado ou com muita neblina até a noites.

É possível o controle de aparência de elementos do cenário, bem como a inserção de diferentes formas de fontes de luz (pontuais, direcionais, ambiente), visando dar mais realismo ao cenário concebido. Recursos de acrescentar sons e filmes também estão disponíveis por utilização de nós específicos e são compatíveis com os principais formatos de áudio e vídeo: .mpeg, .mpg, .mid., .wav.

Podem ser elaborados *scripts* que facilitam as animações utilizando-se *Java* ou *JavaScript* de forma a complementar a troca de informações entre os elementos do mundo virtual. Esta propriedade provê possibilidade de animações e de dinamismo às formas concebidas e inseridas no cenário. O código em *JavaScript* pode fazer parte do arquivo original.

3. Gerando Mundos VRML

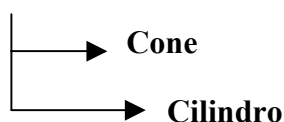
3.1 Formas primitivas e suas combinações

A produção de arquivos VRML, de forma a definir um objeto, um cenário ou um ambiente 3D com diversos objetos e links começa pela distinção do que deve ser

associado, em termos de primitivas e formas gráficas, para obter o resultado desejado. A associação de tais elementos relaciona-se com a concepção de uma árvore, onde as folhas representam os nós VRML (geralmente formas geométricas) que devem ser utilizados.

Assim, para gerar um mundo VRML inicie pelo planejamento das formas que serão necessárias e sua hierarquia. Como exemplo, suponhamos que se deseje produzir uma chaminé, onde serão associados um cone e um cilindro. A estrutura da chaminé necessitará de duas primitivas disponíveis.

Chaminé



Para gerar um cone, teríamos de definir suas características fundamentais, relativas a altura, raio da base, cor etc, como pode ser verificado na Figura 1.

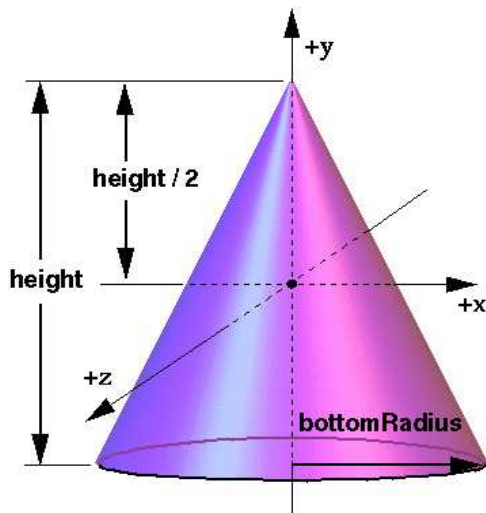


Figura 2 - Elementos Geométricos do Cone

Há dois aspectos importantes na definição: elementos da geometria (como o raio da base e a altura) e elementos da aparência da forma (cor, textura etc). Os nós VRML diferenciam tais elementos por dois campos distintos: *'geometry'* e *'appearance'*.

Assim, para definir um cone como o proposto, teríamos a descrição textual abaixo como arquivo VRML:

```
Shape {  
    appearance Appearance {}  
    geometry Cone {  
        height 2.0  
        bottomRadius 1.0 }  
}
```

Observe que não foram feitas quaisquer definições de aparência, levando a forma gerada, de acordo com o texto acima, a ter aparência branca.

Para gerar o Cilindro, os elementos de geometria seriam os apresentados na Figura 2.

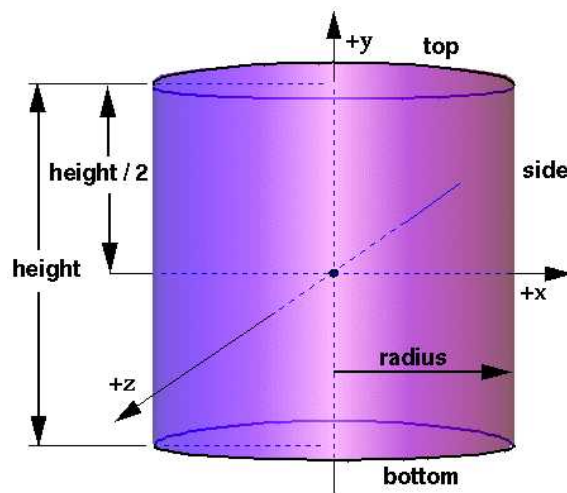


Figura 3 - Elementos Geométricos do Cilindro

A definição, em VRML, do nó *Cylinder* pode ser dada por:

```
Shape {  
    appearance Appearance {}
```

```

    geometry Cylinder {
        bottom TRUE
        top TRUE
        radius 2.0
        height 3.0
    }
}

```

Visando a construção da chaminé, temos de deslocar o cone, de forma que sua base coincida com o topo do cilindro. Tal deslocamento refere-se a uma translação da forma "*Cone*" no eixo Y, do valor da metade da altura do cilindro (*height/2*) mais a metade da altura do Cone (*height_cone/2*), uma vez que, conforme ilustra a Fig. 2 , teríamos metade do cilindro acima e metade do cilindro abaixo do eixo 'x' (o mesmo vale para o Cone).

Destacam-se as formas geométricas básicas:

- Box
- Cone
- Cylinder
- Sphere
- Text - formatar textos
- ElevationGrid
- Extrusion
- IndexedFaceSet
- IndexedLineSet
- PointSet

3.1.1 Experimentando com Textos

É possível inserir textos nos ambientes 3D concebidos através do nó *Text*, cuja sintaxe é dada por:

```
Text {  
    string []  
    length []  
    maxExtent 0.0  
    fontStyle NULL  
}
```

A apresentação do texto pode ser formatada com uso do nó *FontStyle*, que permite definir a fonte que será utilizada, seu tamanho, espaçamento de caracteres, como o texto será justificado etc.

O trecho de código abaixo, relativo à geometria, gera um texto simples como exemplo - Figura 3.

```
geometry Text {  
    string ["SVR 03 - Curso de VRML 2.0"]  
    fontStyle FontStyle {  
        size 0.9  
        justify "MIDDLE"  
        spacing      0.3  
        style "BOLD"  
        family "SERIF"  
    }  
}
```

SVR 03 - Curso de VRML 2.0

Figura 4 - Exemplo de texto em VRML

3.2 Transformações Geométricas

O nó VRML que permite que façamos transformações geométricas (translação, rotação e escala) sobre as formas definidas é o nó *'Transform'*. *'Transform'* é, de fato, um nó de agrupamento, que, para todas as formas de seus filhos (*'children'*) aplica as transformações definidas nos campos *'scale'*, *'translation'* e *'rotation'*:

Sintaxe básica do nó *'Transform'*:

```
Transform {  
    scale 1.0 1.0 1.0  
    translation 0.0 0.0 0.0  
    rotation 0.0 0.0 0.0 0.0  
    children [  
    ]  
}
```

'Scale' escala em X,Y,Z, respectivamente;

'Translation': translação nos eixos X,Y,Z respectivamente;

'rotation': rotação em X, Y, Z pelo argumento (em rad) do quarto campo;

'children': lista dos elementos a qual devem ser aplicadas as transformações.

Assim, o arquivo final para geração da chaminé deveria ter um código semelhante às linhas abaixo:

```
Group{
```

```

children [
  Transform {
    scale 1.0 1.0 1.0
    translation 0.0 3 0.0
    rotation 0.0 0.0 0.0 0.0
    children [
      Shape {
        appearance Appearance {}
        geometry Cone {
          height 3.0
          bottomRadius 2.5 }
        }
      ]
    }
  Shape {
    appearance Appearance {}
    geometry Cylinder {
      bottom TRUE
      top TRUE
      radius 1.0
      height 3.0
    }
  }
}

```

```

    ]
}

```

As linhas de código equivalem à uma árvore como a apresentada na Figura 4.

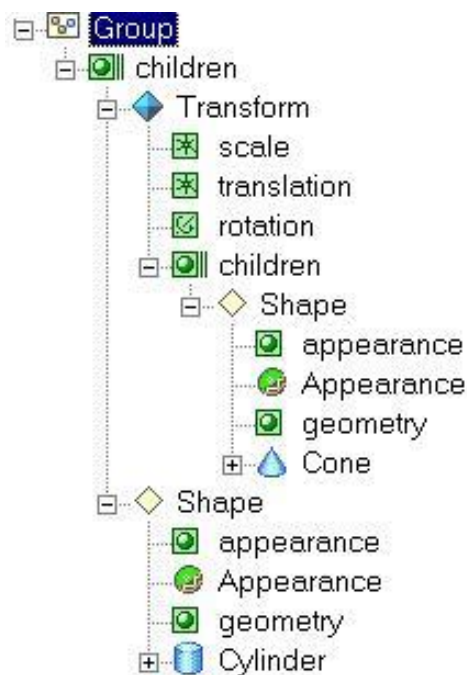


Figura 5 - Estrutura de árvore da Chaminé Exemplo

Observa-se que um código simples, iniciado por um cabeçalho padrão (#VRML V2.0 utf8) e seguido por uma árvore que contém a descrição de dois elementos geométricos, um cilindro e um cone gera o efeito da construção de uma pequena chaminé. O grafo que descreve a cena é dado por um nó principal *Group* que contém dois nós como filhos - *children*:

- nó *Transform*, capaz de promover a translação de uma dada forma, no caso, de um dado cone;
- nó *Shape* que é usado para definir uma outra forma, no caso, um cilindro.

A facilidade de implementar formas geométricas é facilmente observada através deste exemplo, que permite verificar que um arquivo que tem menos de 1kb é capaz de produzir uma forma geométrica 3D, que a partir dos recursos do navegador VRML pode ser vista de diferentes ângulos, rotacionada para verificação, aproximada ou afastada, girada, apontada etc, tornando a navegação uma agradável experiência interativa com a forma.

Até meados do ano de 2000, a edição de textos que descreviam formas em VRML não dispunha de um editor de textos exclusivo para este fim. De olho neste nicho de mercado, a empresa *Parallel Graphics* lançou o VrmIpad® [VRMLPAD 2000], que se encontra na versão 2.0 e que facilita sobremaneira a edição de tais textos.

3.3 Alterando Aparência - nó appearance

De maneira a acomodar a definição adequada da aparência de uma forma, o nó *appearance* tem a seguinte sintaxe:

```
appearance Appearance {  
    material Material {  
        ambientIntensity 1.0  
        diffuseColor 1.0 0.0 0.0  
        emissiveColor 1.0 0.0 0.0  
        specularColor 1.0 0.0 0.0  
        transparency 0.3  
        shininess 0.4  
    }  
    texture      ImageTexture {url ""}  
}
```

}

As variáveis *diffuseColor*, *emissiveColor* e *specularColor* referem-se à definição das cores do objeto relativas à iluminação difusa, especular e a cor que o mesmo emite. A transparência do mesmo é dada pelo campo '*transparency*' e caso seja necessário, podemos aplicar a textura relativa à uma imagem (JPEG, GIF etc) ao objeto dando o endereço da mesma no campo *ImageTexture*. Neste caso, as definições de cores feitas em 'Material' são perdidas.

Como exemplo, para alterar o cone já definido anteriormente, dando-lhe uma aparência acinzentada, poderíamos definir o nó *appearance* como:

```
appearance Appearance {  
    material Material {  
        diffuseColor .8 0 .13  
        specularColor .77 .77 .77  
        emissiveColor .1 0 .02  
        ambientIntensity .0767 }}}
```

O resultado da alteração da forma do cone pode ser visto na Figura 5.



Figura 6 - Cone colorido - uso de 'appearance'

Algumas vezes, no entanto, deseja-se aplicar uma textura a uma dada forma, ao invés de definir propriedades de cor para a mesma. Neste caso, imagens do tipo JPEG, GIF e PNG podem ser aplicadas à forma desejada com uso do nó '*Material*'.

São campos possíveis de serem utilizados: *ImageTexture*, *PixelTexture* e *MovieTexture*.

Como exemplo, a aplicação de uma textura feita ao cone, similar ao definido anteriormente, conforme o código abaixo pode ser visto na Figura 6 .

```
Transform {  
  
    scale 1.0 1.0 1.0  
  
    translation 0.0 3 0.0  
  
    rotation 0.0 0.0 0.0 0.0  
  
    children [  
  
        Shape {  
  
            appearance Appearance {  
  
                material Material {  
  
                    diffuseColor .8 0 .13  
  
                    specularColor .77 .77 .77  
  
                    emissiveColor .1 0 .02  
  
                    ambientIntensity .0767  
  
                }  
  
                texture ImageTexture {  
  
                    url "brick.jpg"}  
  
            }  
  
            geometry Cone {  
  
                height 3.0  
  
                bottomRadius 1.5 }  
  
        ]  
    }
```

}}



Figura 7 - Cone com aplicação de Textura

Há possibilidade de controlar o mapeamento de texturas com utilização dos nós *TextureCoordinate* e *TextureTransform*. O nó *TextureTransform* pode ser usado como um valor do campo de valor no nó *Appearance*.

O trecho de código abaixo altera a aplicação de textura no cone, por uma escala de 2.0 para as direções S e T, respectivamente. A Figura 7 apresenta o plano de aplicação de texturas S x T.

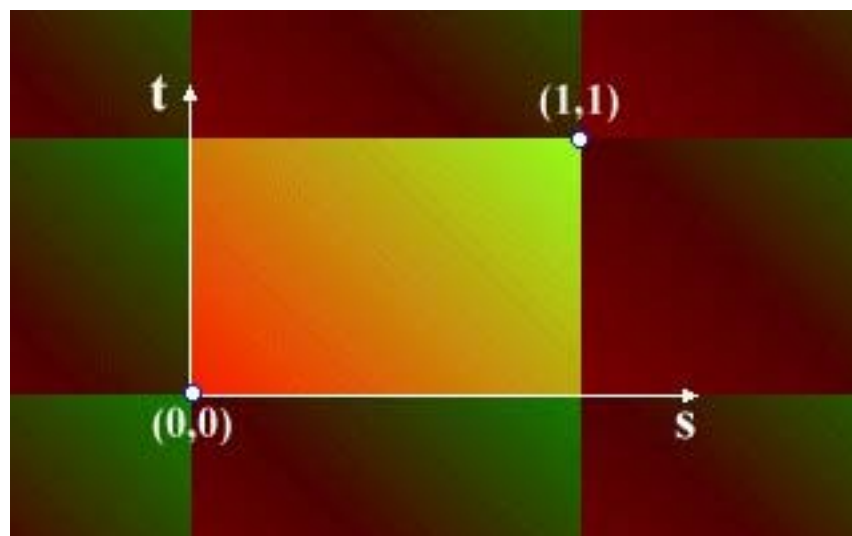


Figura 8 - Plano de aplicação de textura S x T

O resultado pode ser visto na Figura 8.

```
textureTransform TextureTransform {  
    scale 2.0 2.0  
}  
  
texture      ImageTexture {  
    url "brick.jpg"}}}
```

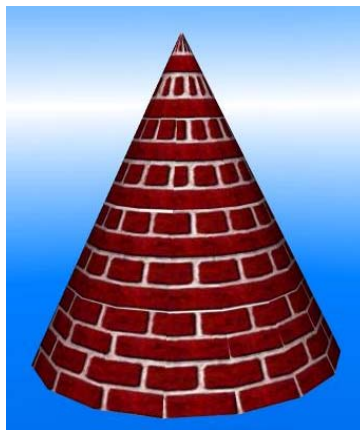


Figura 9 - Alterando Aplicação de Textura - textureTransform

3.4 Reutilizando Formas, Elementos e Definições

Quando se faz necessário reutilizar uma dada forma, são usados os termos DEF e USE. DEF define o nome de um dado elemento, forma ou definição. A reutilização deste nome a partir de sua chamada por USE dispensa a redefinição.

Como exemplo, será definido um conjunto de colunas, a partir da definição de uma única coluna. A coluna básica será definida por "Coluna" e terá aparência definida por "White". Ao longo da descrição do conjunto a reutilização se dará pela chamada destes termos com USE, como pode ser visto no código abaixo:

```
Group {  
    children [  

```

Coluna da Esquerda

```
DEF ColEsquerda Transform {  
    translation -2.0 3.0 0.0  
    children DEF Coluna Shape {  
        appearance DEF White Appearance {  
            material Material { }  
        }  
        geometry Cylinder {  
            radius 0.3  
            height 6.0  
        }  
    }  
},
```

Coluna da Direita:

```
DEF ColDireita Transform {  
    translation 2.0 3.0 0.0  
    children USE Coluna  
},
```

Coluna Superior

```
DEF Superior Transform {  
    translation 0.0 6.0 0.0  
    rotation 0.0 0.0 1.0 1.57  
    children Shape{
```

```

        appearance USE White
        geometry Cylinder {
            radius 0.3
            height 4.6
        }
    }
}

```

O resultado é mostrado na Figura 9, onde um elemento foi reutilizado com uso de DEF e USE.

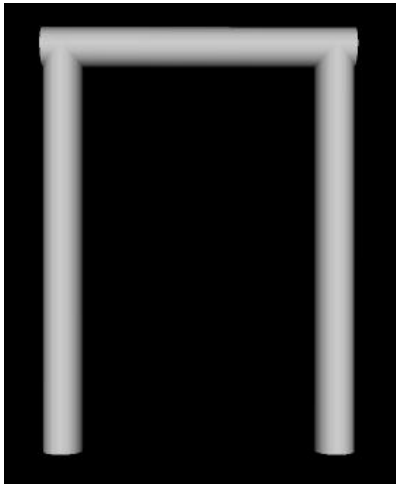


Figura 10 - Coluna com uso de DEF e USE

Podem ser inseridos elementos definidos em um dado arquivo dentro de um cenário 3D, com uso do nó *Inline*. A sintaxe de *Inline* é dada por:

```

Inline{

```

```

url []

bboxCenter -1.0 -1.0 -1.0

bboxSize 0.0 0.0 0.0

}

```

No exemplo abaixo, o grupo de colunas é combinado, formando um conjunto de grupos de colunas, através da chamada feita pelo *Inline* combinada com uso de "DEF" e "USE". O resultado pode ser visto na Figura 10.

```

Group {
  children [
    Transform {
      translation 0.0 -2.0 0.0
      children DEFcoluna Inline {url "colunas.wrl"} }
    Transform {
      translation 0.0 -2.0 -2.0
      children USE coluna
    }
    Transform {
      translation 0.0 -2.0 -4.0
      children USE coluna
    }
    Transform {
      translation 0.0 -2.0 -6.0
      children USE coluna
    }
  ]
}

```

```

    }
  }
}

```

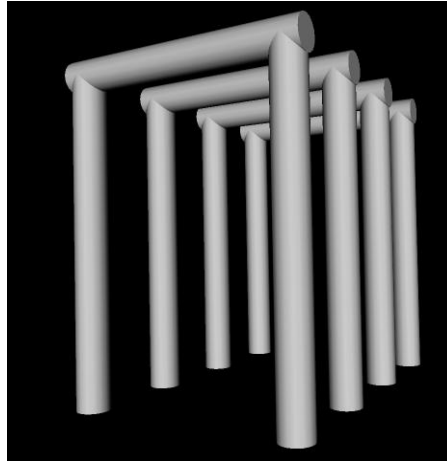


Figura 11 - Grupo de Colunas - uso de Inline

3.5 Composto formas como conjunto de Faces

A utilização do nó *IndexedFaceSet* permite construir conjuntos de faces interligadas, de forma que tal combinação gere efeitos de formas complexas. Na descrição de um conjunto de faces, uma lista de pontos coordenados deve ser explicitada, além das conexões entre estes pontos. Supondo que se deseje a construção de um cubo seriam necessárias as coordenadas da parte superior do cubo e da parte inferior, além das descrições de ligações entre estes pontos.

```

geometry IndexedFaceSet {
  coord Coordinate {
    point [
      # Coordenadas da parte superior do cubo
      -1.0 1.0 1.0,
      1.0 1.0 1.0,
      1.0 1.0 -1.0,

```

```

        -1.0 1.0 -1.0,

# Coordenadas da parte inferior do cubo

        -1.0 -1.0 1.0,

        1.0 -1.0 1.0,

        1.0 -1.0 -1.0,

        -1.0 -1.0 -1.0

    ]
}

coordIndex [
# face superior

    0, 1, 2, 3, -1,

# face de fundo

    7, 6, 5, 4, -1,

# face de frente

    0, 4, 5, 1, -1,

# face da direita

    1, 5, 6, 2, -1,

# face de costas

    2, 6, 7, 3, -1,

# face da esquerda

    3, 7, 4, 0

]
}

```

```
}
```

E o resultado pode ser visto na Figura 11, onde um Cubo foi construído a partir de suas faces.

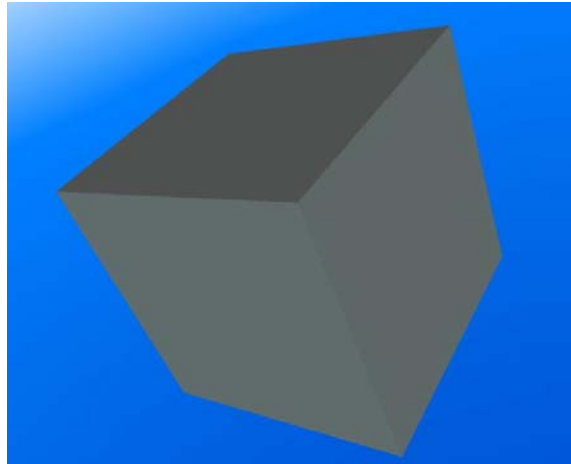


Figura 12 - Cubo - uso de IndexedFaceSet

3.6 Fundos e Cenários

O nó *Background* permite criar fundos para os mundos virtuais usando uma caixa de textura, elementos de piso e elementos do céu. A sintaxe (resumida) do nó *Background* é:

```
Background{  
    skyColor [0.0 0.0 0.0]  
    skyAngle []  
    groundColor []  
    groundAngle[]  
    backUrl []  
    bottomUrl []
```

```
    leftUrl []  
    rightUrl []  
}
```

O trecho de código a seguir, relativo à Figura 12, apresenta um cenário com um céu azulado e o chão em tom marrom:

```
Background {  
    skyColor [  
        0.0 0.2 0.7  
        0.0 0.5 1.0  
        1.0 1.0 1.0  
    ]  
    skyAngle [1.309 1.571]  
    groundColor[  
        0.1 0.1 0.1  
        0.4 0.25 0.2  
        0.6 0.6 0.6  
    ]  
    groundAngle [1.309 1.571]  
}
```

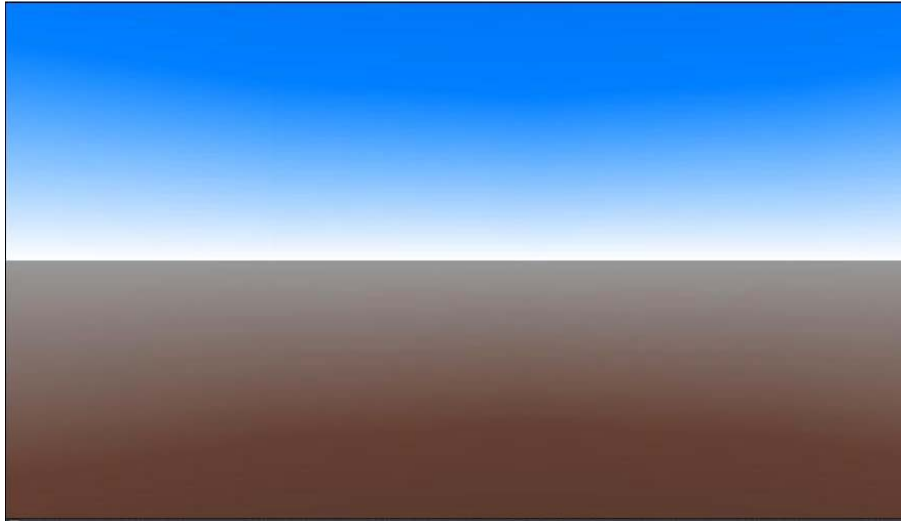


Figura 13 - Background aplicado

A definição de uma textura que é aplicada ao fundo pode ser feita como apresentado no trecho de código abaixo:

```
Background {  
    skyColor [  
        0.0 0.2 0.7,  
        0.0 0.5 1.0,  
        1.0 1.0 1.0  
    ]  
    skyAngle [ 1.309, 1.571 ]  
    groundColor [  
        0.1 0.10 0.0,  
        0.4 0.25 0.2,  
        0.6 0.60 0.6,  
    ]  
    groundAngle [ 1.309, 1.571 ]
```

```
frontUrl "montanha.gif"  
backUrl "montanha.gif"  
leftUrl "montanha.gif"  
rightUrl "montanha.gif"}
```

E o resultado pode ser visto na Figura 13, onde um elemento de imagem auxilia na definição mais realística de um fundo.

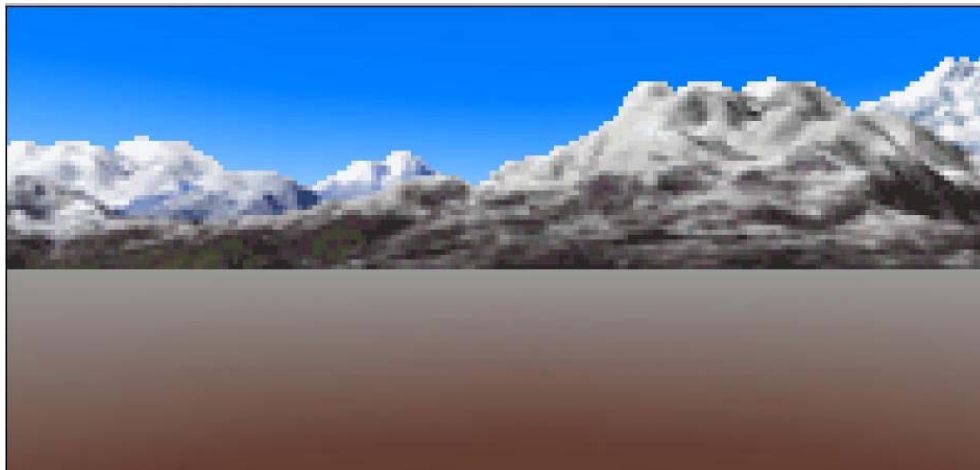


Figura 14 - Background com elementos de imagem ".gif"

3.7 Iluminação

VRML provê diversos nós para uma iluminação adequada de cenário. Todos estes nós apresentam os seguintes campos: *color*, *ambientIntensity* e *intensity*. As funções dos nós podem ser:

- iluminação do tipo direcional, com os raios emanando de forma radial em todas as direções: *PointLight*;
- iluminação do tipo direcional, com os raios pertencem a um pincel de luz paralela: *DirectionalLight*;
- iluminação do tipo 'spot', com os raios em um pincel na forma de cone, com a fonte de luz no ápice do cone: *SpotLight*;

Em todos os casos, a contribuição de uma fonte de luz é definida pelo campo *intensityColor*, enquanto que a contribuição da luz ambiente é dada em função do valor do campo *ambientIntensity*. Objetos com textura não são afetados pela fontes de luz. Pode-se desligar a luz na cabeça do usuário (navegador) através da definição a *headlight FALSE* no nó *NavigationInfo*, de acordo com a sintaxe:

```
NavigationInfo {  
    headlight FALSE  
}
```

4. Combinando VRML e JavaScript

4.1 Introdução

Os nós já apresentados para animação, tais como *TouchSensor*, *TimeSensor*, *PlaneSensor* etc são muito limitados quando se desejam animações mais complexas ou controle de elementos de cena a partir de equações (matemáticas, físicas, químicas etc). Em tais situações é imperativa a necessidade de uso dos nós *Script*, que associam VRML com JavaScript ou Java.

Um nó *Script* pode ser entendido como uma forma particular de controle ou de sensor. Como qualquer nó de controle ou de sensor, este nó requer uma lista de campos (*field*) , eventos de entrada (*eventIns*) e eventos de saída (*eventOuts*). A descrição do nó deve complementar a definição destes campos, dando a eles uma dada finalidade . Um nó *Script* deve ser entendido como um nó que recebe informações de outros nós através dos eventos de entrada, processa-as e envia informações na forma de eventos de saída.

Logicamente, não serão encontrados nós *Script* independentes da descrição de rotas, que mapeiam a troca de informações entre nós externos e o nó *Script*. Uma

exigência importante é a necessidade de equivalência de tipos de elementos entre os nós que trocam informações.

Em VRML, temos diferentes tipos de elementos, como exemplo:

- SFBool - para valores booleanos
- SFInt32 - para valores inteiros
- MFInt32 - para uma lista de valores inteiros
- SFFloat - para valores reais
- MFFloat - para uma lista de valores reais
- SFNode - para a descrição de um nó
- SFColor - para a descrição de RGB de uma dada cor
- Etc

Característica básica de um *Script*:

Script {

definição de campos, suas naturezas e valores iniciais:

field SFBool booleano1 TRUE

definição de eventos de entrada e sua natureza:

eventIn SFBool entrada

definição de eventos de saída e sua natureza:

eventOut SFBool saida

url "javascript:

function entrada(param){

// processamento e determinação da saída

saida = param;

```

"
}

```

4.2 Desenvolvendo Exemplos

Como exemplo, a função abaixo representa um Script capaz de inserir um novo elemento, no caso, uma esfera em uma área de um mundo virtual:

```

DEF Criador Script {

    eventIn SFTime sphere_touchTime

    field SFNode nopai USE TOP

    field SFFloat x 0.0

    eventOut MFNode new_child

    url "javascript:

        function sphere_touchTime(value,time) {

            newVRML = 'Group {';

            newVRML += '  children [';

            newVRML += '    DEF SENSOR PlaneSensor {';

            newVRML += '      maxPosition 0.45 0.45';

            newVRML += '      minPosition -0.45 -0.45'

            newVRML += '    }';

            newVRML += '    DEF OBJECT Transform {';

            newVRML += '      translation '

            newVRML += '        x;

```

```

newVRML += ' 0.0';
newVRML += ' 0.0';
newVRML += ' children [';
newVRML += '     Shape {';
newVRML += '         appearance Appearance {';
newVRML += '             material Material {';
newVRML += '                 diffuseColor 0 1 1';
newVRML += '             }';
newVRML += '         }';
newVRML += '     geometry Sphere {';
newVRML += '         radius 0.05';
newVRML += '     }';
newVRML += ' }';
newVRML += ' ]';
newVRML += ' }';
newVRML += ' ]';
newVRML += ' }';

newVRML += '     ROUTE     SENSOR.translation_changed     TO
OBJECT.set_translation';

    node = Browser.createVrmlFromString(newVRML);
    new_child = node;
    nopai.addChildren = new_child;
}

```

```
"
}
```

```
ROUTE SPHERESENSOR.touchTime TO Criador.sphere_touchTime
```

É possível alterar dinamicamente um texto, que funcione como um contador de esferas adicionadas (continuação do exemplo anterior), se um dado nó de texto for atualizado, a partir de um campo que conte quantas esferas foram adicionadas. O trecho de código abaixo refere-se à função `sphere_touchTime` atualizada, de tal forma que a mesma é capaz de modificar o texto, enviando, através de uma rota, o valor atual de esferas inseridas:

```
field SFFloat x2 0.0

field MFString str_sphere "0"

eventOut    MFString str_sphere_out

function sphere_touchTime(value,time) {
    x2 = x2 + 1.0;
    str_sphere[0] = String(x2);
    str_sphere_out[0] = str_sphere[0];
}
```

A rota combinada seria do tipo:

```
ROUTE Criador.str_sphere_out TO textoS.string
```

Onde `textoS` é um campo de texto pertencente a um nó `Shape`.

5. Combinando VRML e Java - dicas

A combinação de VRML, *Java* e HTML pode ser efetuada de seis diferentes maneiras, de forma a facilitar a utilização e tirar máximo proveito desta possibilidade. A lista abaixo é um sumário das hipóteses:

1. código VRML inserido em um arquivo HTML: usando caracteres de formatação do tipo <EMBED> ou <OBJECT>, códigos de VRML são inseridos em arquivos HTML;
2. código *Java* inserido em arquivo VRML: este é o padrão para VRML 2.0, um nó do tipo *Script* refere-se a um código *Java* pré-compilado;
3. *applet Java* comunicando com o navegador VRML: esta versão - não padrão para VRML 2.0 - constitui uma extensão e é conhecida como *External Authoring Interface (EAI)* e tende a se tornar o padrão;
4. classes *Java* correspondentes a nós VRML: algumas companhias estão desenvolvendo ferramentas de programação que definem representações de nós VRML que podem ser usadas por programadores quando estão escrevendo
5. código em *Java*. Esta hipótese é mais usual para implementar navegadores ou ferramentas VRML, mas, não constitui padrão para VRML ou *Java*;
6. HTML dentro do código VRML: usando um arquivo HTML como um mapa de textura para apresentar dentro de um mundo 3D, podemos ter uma interessante extensão para VRML, mas, esta proposição ainda não é suportada por qualquer navegador VRML e não é parte do VRML 2.0, sendo uma proposta dos criadores em discussão;
7. *applet Java* dentro de um arquivo VRML: usando um *applet Java* como um mapa de textura temos outra interessante possibilidade de extensão, também em discussão e ainda não implementada.

Por permitir maior flexibilidade e constituir uma forma mais usual de desenvolvimento, este trabalho opta pela forma de comunicação estabelecida no item 2 acima, na qual um *applet Java* se comunica com o navegador VRML.

6. Referências Bibliográficas

AMES, L. A.; NADEAU, R.D.; MORELAND D. **VRML Sourcebook - Second Edition**, John Wisley & Sons, Inc - USA, 1997.

CARDOSO, A.; LAMOUNIER E.; TORI R. Sistema de Criação de Experiências de Física em Realidade Virtual para Educação a Distância. In: II WORKSHOP BRASILEIRO DE REALIDADE VIRTUAL, **WRV'99**, Marília , São Paulo, Brasil, 1999, p. 174-181.

CARDOSO, A. Alexandre Cardoso - Página do Pesquisador. Contém informações sobre aplicações de Realidade Virtual, pesquisa e publicações do pesquisador, tutoriais sobre VRML e artigos indicados para leitura. Disponível em: <<http://www.compgraf.ufu.br/alexandre/>>.

CHEN, S.; MYERS, R.; PASSETO, R. The Out of Box Experience - Lessons Learned Creating Compelling VRML 2.0 Content. In: Proceedings of the Second Symposium on Virtual Reality Modeling Language, p. 83-92, 1997.

ELLIS R. S. What are Virtual Environments. **IEEE Computer Graphics and Applications**, p. 17-22, Jan. 1994.

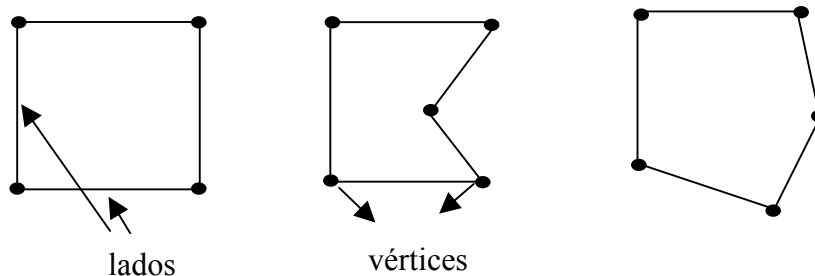
LEMAY, MURDOCK, COUCH **3D Graphics and VRML 2** Sams.Net, Indiana - USA - 1996

MATSUBA, N.; STEPHEN; ROEHL, B. Bottom, Thou Art Translated: The Making of VRML Dream. **IEEE Computer Graphics and Applications**, v. 19, n. 2, p. 45-51, Mar./Apr. 1999.

APÊNDICE B

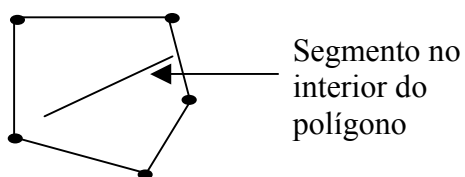
GEOMETRIA PLANA

Polígono: É uma figura plana formada por três ou mais segmentos de reta consecutivos que se intersectam dois a dois. Os segmentos de reta são denominados lados do polígono. Os pontos de intersecção são denominados vértices do polígono [1]. Exemplos:

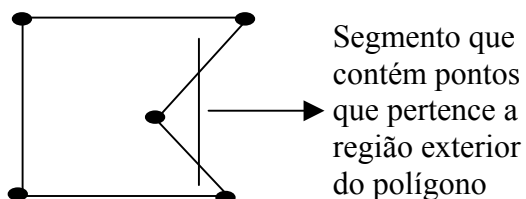


Classificação: Pode-se classificar os polígonos em:

Polígono convexo: É um polígono construído de modo que os prolongamentos dos lados nunca ficarão no interior da figura original. Se dois pontos pertencem a um polígono convexo, então todo o segmento tendo estes dois pontos como extremidades, estará inteiramente contido no polígono [1]. Exemplo:



Polígono não convexo: Um polígono é dito não convexo se dados dois pontos do polígono, o segmento que tem estes pontos como extremidades, tiver pontos que pertence a região exterior desse polígono [1]. Exemplo:

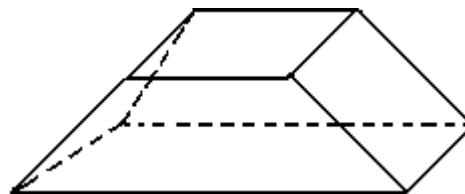
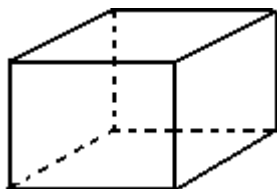


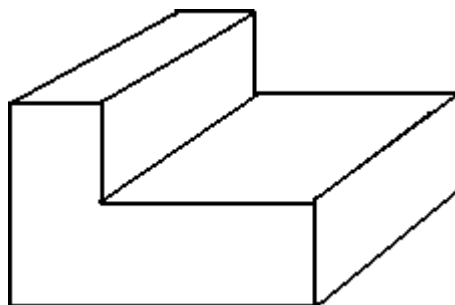
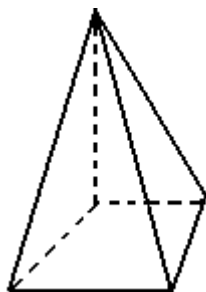
Geometria Espacial

Poliedros

Poliedro é um sólido geométrico cuja superfície é composta por um número finito de faces poligonais.

Veja alguns exemplos:[2]





Os polígonos são as faces do poliedro; os lados e os vértices dos polígonos são as arestas e os vértices do poliedro.

Poliedros convexos e côncavos [2]

Observando os poliedros acima, podemos notar que, considerando qualquer uma de suas faces, os poliedros encontram-se inteiramente no mesmo semi-espaço que essa face determina. Assim, esses poliedros são denominados *convexos*.

Isso não acontece no último poliedro, pois, em relação a duas de suas faces, ele não está contido apenas em um semi-espaço. Portanto, ele é denominado *côncavo*.

Referências Bibliográficas:

[1] Sodré Ulysses Disponível em
<http://pessoal.sercomtel.com.br/matematica/geometria/geom-elem/geometr.htm>

[2] Desenvolvido por Virtuous Disponível em:
<http://www.somatematica.com.br/emedio/espacial/espacial7.php>