UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE INFORMÁTICA – HABILITAÇÃO EM SOFWTARE BÁSICO

INTEGRAÇÃO DE TÉCNICAS DE MODELAGEM COM A VRML

ANDRÉ TAVARES DA SILVA

Monografia submetida como requisito parcial
para obtenção do grau de Bacharel em
Informática – Habilitação em Software Básico

Prof. Dr. Fernando Santos Osório
Orientador

São Leopoldo, novembro de 1999

Para meus pais, irmãos e principalmente à minha noiva Claudia, que me apoiaram e estimularam durante este período, relevando e compreendendo com carinho o meu afastamento do cotidiano do convívio familiar.

Este trabalho foi possível devido ao apoio de diversas pessoas que me incentivaram e ajudaram, especialmente os meus irmãos Paulo Rogério e João Luís. Agradeço a todos os professores que me auxiliaram, principalmente ao Professor Doutor Fernando Santos Osório pela eficiência, rapidez e precisão na orientação deste trabalho, permitindo a conclusão do mesmo em tempo hábil neste semestre.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

Cada vez mais, a computação gráfica tem sido utilizada para a criação de imagens que representam modelos do mundo real. Neste sentido, através da realidade virtual, tenta-se criar um ambiente no qual a pessoa possa sentir-se imersa total ou parcialmente em um mundo virtual, onde objetos imaginários podem ser sentidos e manipulados.

A VRML (*Virtual Reality Modeling Language*), por ser uma linguagem voltada para a Internet, se torna uma ferramenta muito poderosa para permitir a criação de novas aplicações em realidade virtual.

Este trabalho tem por objetivo a integração de técnicas de modelagem com a VRML, através da implementação prática de um software. Foi escolhida a técnica de modelagem por *sweep* para implementação deste software. O sistema de modelagem desenvolvido junto a este trabalho, escrito em Java, pode ser usado não só como aplicativo, mas também ser executado em um *browser*, juntamente com um visualizador VRML.

Este sistema de modelagem, de interface simples e de fácil uso, não requer do usuário conhecimentos avançados de técnicas de modelagem, como nos sistemas

comerciais de computação gráfica. Além disso, por ter sido implementado em Java, é um sistema independente de plataforma, podendo também ser executado através da Internet. Por isso, é também uma ferramenta com forte apelo didático, permitindo que os alunos examinem na prática a técnica de modelagem por *sweep* em um sistema simples.

Palavras-chave: Computação Gráfica, Realidade Virtual, VRML, Modelagem por *Sweep*.

ABSTRACT

Title: “Integration between Modeling Techniques and VRML”

More and more, Computer Graphic has been used to create images that represent models of the real world. This way, the virtual reality creates an environment in which one could experience a total or partial immersion in a virtual world where imaginary objects can be felt and handled.

Since the VRML (Virtual Reality Modeling Language) is a language oriented to the Internet, it is a powerful tool for the creation of new applications on virtual reality.

The objective of this work is the integration of modeling techniques and the VRML through the programming of a software. For this aim the sweep modeling technique was chosen. The modeling system developed in this work, written in the Java programming language, can be used either as an executable program or as an applet in an Internet browser together with a VRML viewer.

This modeling system, with simple user interface and straightforward operation, does not require an advanced knowledge of modeling techniques as the commercial

systems do. Also, it is a platform independent program and can be executable through the Internet, since it was written in Java. Thus, it is also a tool with a didactic appeal that allows the students to examine the sweep modeling technique in a simple system.

Keywords: Computer Graphic, Virtual Reality, VRML, Sweep Modeling

1. INTRODUÇÃO

Um dos sentidos humanos mais ricos e diversificados é a visão, cuja capacidade de percepção nos faz interagir com o mundo real gerando ações que o mudam constantemente. Atualmente, quando a proliferação da computação atinge todos os setores humanos, a tendência metafórica é justamente atingir o máximo de proximidade com a máquina humana. Temos exemplos da área de inteligência artificial, procurando imitar o cérebro, e de outra, mais atraente, a computação gráfica que procura desenvolver a habilidade visual da máquina.

O forte apelo das imagens seduz o ser humano mais do que as palavras, e quanto mais próximo da realidade for o apelo visual, mais confortável e próximo a sua própria realidade o homem se sentirá.

Cada vez mais, a computação gráfica tem sido utilizada para a criação de imagens que representam modelos do mundo real. Neste sentido, através da realidade virtual, tenta-se criar um ambiente no qual a pessoa possa sentir-se imersa total ou parcialmente em um mundo virtual, onde objetos imaginários podem ser sentidos e manipulados.

A realidade virtual é uma tecnologia que permite uma melhor interface homem-máquina, e, junto com a Internet através da VRML, se torna uma ferramenta muito poderosa para permitir a criação de novas formas de resolução de problemas. É portanto, um novo meio de comunicação, na qual pode-se manipular a informação através de uma

“*experiência em 1ª pessoa* ”.

Este trabalho tem por objetivo a integração de técnicas de modelagem com a VRML, através da implementação prática de um software. A técnica de modelagem escolhida para fazer esta integração foi a técnica de modelagem por *sweep*, por ser uma maneira prática e fácil de se construir uma grande variedade de objetos do mundo real. A interface do sistema de modelagem desenvolvido junto a este trabalho é bastante simples de usar, podendo o software ser usado não só como aplicativo para permitir a gravação dos objetos gerados e de seus polígonos, mas também ser executado através da Internet, juntamente com um visualizador VRML, já que a VRML está voltada principalmente para a Internet. Para isto ser possível, a linguagem usada para o desenvolvimento do software foi a linguagem Java.

No segundo capítulo, são apresentados alguns conceitos sobre a realidade virtual, as interfaces usadas e suas aplicações. Também neste capítulo é feita uma descrição da VRML e suas versões, e exposta algumas de suas características, importâncias e tendências futuras.

No terceiro capítulo, são estudadas algumas das principais técnicas de modelagem usadas para representar objetos do mundo real.

No quarto capítulo, é detalhado o sistema de modelagem por *sweep*, mostrando e descrevendo seus principais tipos, já que este é o tipo de sistema de modelagem utilizado para a implementação do software desenvolvido junto a este trabalho.

No quinto capítulo, é apresentada a ênfase do trabalho, a forma como foi feita a implementação do sistema de modelagem, as interfaces deste sistema, como são gerados os objetos para VRML, estudados os tipos de *sweep* implementados no sistema de modelagem bem como os algoritmos usados para gerar os objetos.

2. REALIDADE VIRTUAL

A expressão Realidade Virtual foi construída sobre a oposição do sentido usual dos seus termos. Segundo o dicionário Aurélio, a expressão *virtual* significa: “Que existe como faculdade, porém sem exercício ou efeito atual. Suscetível de se realizar; potencial”. Portanto, a rigor, real e virtual não são conceitos que se opõem, mas se complementam [CAD 97]. Para fechar com a questão das expressões, assinala-se que esta expressão não foi a única que esteve em debate. Assim, podemos encontrar outros termos como realidade artificial, mundos virtuais, ambientes multissensoriais interativos, *cyberspace*, entre outros.

Através da realidade virtual, tenta-se criar um ambiente (tridimensional), no qual o usuário tem a ilusão de estar imerso total (*Full-imersive*) ou parcialmente. Ela consiste de uma combinação de software, computadores de alto desempenho e periféricos especializados, que permitem criar um ambiente gráfico de aparência realística, no qual o usuário pode se locomover em três dimensões. Nele, objetos imaginários, criados por software, podem ser sentidos e manipulados. A realidade virtual é fundamentalmente uma tecnologia de melhoria na interface homem-máquina, e como tal, habilitou a criação de novas ferramentas de resolução de problemas. É essencialmente, um novo meio de comunicação. Ela é a simulação computacional de um ambiente no qual o participante pode penetrar e interagir [CAD 97].

2.1 Interfaces da Realidade Virtual

Para permitir que isto seja possível, foram criadas várias interfaces, como interfaces óticas e sonoras (capacete com fones e telas embutidas), luvas, bio-sensores, entre outros (figura 2.1).

Figura 2.1 A imersão em um mundo virtual.
Capacete de visão, fones de áudio e luvas de dados.
Foto: Nasa/SPL/Stock Photos.

2.1.1 Interfaces Óticas

A criação de uma interface que possibilitasse a imersão total do usuário no ambiente virtual foi conseguida com o HMD (*Head Mounted Display).* Esta interface, com aparência de um capacete, é composta basicamente de um par de *displays,* que podem ser de cristal líquido, ou tubos de raios catódicos, fornecendo ao usuário um amplo campo de visão.

Pode possuir também, um mecanismo que monitora a posição e a direção da cabeça do usuário, chamado *Tracker*. Este mecanismo transmite os movimentos da cabeça do usuário ao computador, que atualiza o modelo do mundo virtual que e o usuário está vendo de acordo com seu movimento. Portanto, se o usuário virar a cabeça para trás, ele verá o que há atrás dele no ambiente virtual. Através desse sistema, o HMD possibilita ao usuário a visão do ambiente virtual, bloqueando sua visão do mundo real.

Outra interface visual que viabiliza a utilização por um tempo mais prolongado, ao contrário dos HMDs comuns, é a BOOM *(Binocular Omni-Orientation Monitor),* que consiste de uma tela com um visor parecido com o de um periscópio presa a um braço mecânico. Essa interface propicia maior rapidez no monitoramento da posição da cabeça do usuário, permitindo que este possa tirá-lo e colocá-lo rapidamente, não imprimindo o seu peso na cabeça.

O próximo passo para as interfaces visuais é um capacete transparente que projeta a imagem diretamente na retina, através de um feixe de laser de baixa intensidade. Chamada de VRD (*Virtual Retinal Display*), espera-se que essa interface possibilite uma resolução maior de imagem que a da HDTV (*High Definition TV*).

Tal sistema deve revolucionar a concepção de interfaces ópticas, alcançando assim um nível de definição gráfica, peso e custos ideais para a sua produção em massa.

2.1.2 Luvas especiais

Para se criar a possibilidade de manipulação de objetos virtuais contidos em um ambiente virtual, o usuário imerso necessita de uma interface que foi adaptada como uma luva especial que contém sensores conectados ao computador. Estes sensores monitoram a posição da mão e grau de flexão dos dedos do usuário que a utiliza. Portanto, ao se colocar uma luva quando se está usando um HMD, é possível ver uma mão virtual flutuando a sua frente na mesma posição relativa que a real. Se o usuário mover a sua mão para a direita, movimentando seus dedos, a mão visualizada vai se mover da mesma forma.

Algumas luvas também propiciam, além da leitura dos movimentos da mão do operador, sensações táteis. Essas sensações deve-se a utilização de bolsas de ar, como o TeleTact da *Airmuscle Ltda*., que inflam e esvaziam rapidamente, pressionando diferentes partes da mão. Outras, como o *Portable Dextrous Master*, que possuem pistões que seguram as pontas dos dedos afim de propiciar um *feedback* de força.

A última geração de luvas usa um metal especial chamado *nitinol*. Na tentativa de simular sensações de textura, é colocado uma rede de cabos feitos de uma liga metálica do tipo *shape memory* sob os dedos e a palma da mão, que são eletricamente excitados.

Existe também a *DataSuit*, composta por um conjunto de sensores mais complexos, esta é uma vestimenta que pega a posição de todas as articulações do corpo.

2.1.3 Interfaces sonoras

As interfaces sonoras, como fones de ouvido, embutidos ou não nos HMDs, podem simular sons com efeito tridimensional quase perfeito, possibilitando ao usuário uma maior noção de imersão.

O sistema *Convolvotron* criado na NASA pela psicóloga Elisabeth Wenzel (especializada em percepção), por Scott Foster (presidente da *Cristal River Engineering*) e pelo Dr. Frederick Wightman (pesquisador de som 3D da *Univesity of Wisconsin*), usa um princípio chamado *Head Related Transfer Function* (função de transferência relativa à cabeça).

Esse princípio parte da idéia que cada pessoa interpreta um som e sua posição espacial de acordo com a forma que esse som chega ao seu tímpano. Portanto, quando um som chega ao ouvido de uma pessoa, ele entra e ecoa dentro deste de acordo com sua forma, que varia de pessoa para pessoa. Então, o cérebro interpreta este eco, fornecendo a posição da fonte emissora desse som.

O *Convolvotron*, aparelho que fornece um som 3D, é um grande conjunto de *chips* localizados numa placa dentro do computador, que processa sinais de som ajustados a cada tipo de ouvido e os transmite para um capacete. Este sistema, no entanto, é muito caro e complexo, sendo necessário que o usuário se acostume a este para perceber um som 3D.

O sistema de som 3D criado pelo Dr. Jack M. Loomis, muito mais barato e mais simples, consiste basicamente de um pequeno amplificador de som portátil, ao qual se conecta a dois pequenos fones de ouvido e dois pequenos microfones, além de um protetor de ouvido que quando usado isola o som ambiente. Derrubando a teoria de *Head Related Transfer Function*, esse sistema é extremamente simples e possibilita provar que não é necessário utilizar um sistema que saiba exatamente como os som vai ecoar dentro do ouvido para que se consiga um som estéreo 3D. Conseguindo, portanto, alcançar um verdadeiro som estéreo 3D.

2.1.4 Interfaces olfativas e bio-sensores

O sistema *Laser Helmet* da *Immersive Technologies Inc.*, dispõe de dispositivos, que podem estar embutidos num capacete, que propiciem sensações olfativas.

Existem outras interfaces, do tipo bio-sensor, como um sensor de stress do *Virtual-Ski* da *NEC*, que possibilita ao computador saber o estado emocional do usuário e agir de acordo com isso.

2.1.5 Desktop VR

A *Desktop VR* é considerada um tipo de realidade virtual diferente da *Full-imersive*, possuindo outros tipos de interfaces mais acessíveis ao grande público. Uma Desktop VR, é uma realidade virtual baseada em plataformas pessoais, onde a imersão se torna parcial.

Neste sistema, o HMD é substituído por um monitor de microcomputador, e eventualmente por *Shutter Glasses* que são óculos que, junto com um *software* apropriado, dão noção estéreo da imagem vista da tela. Além disso, a luva especial pode ser substituída por um *mouse*, teclado, *joystick* ou *spaceball*.

2.2 Evolução Tecnológica

Os militares foram pioneiros na utilização dos primeiros sistemas de realidade virtual, representados pelos simuladores de vôo. Os simuladores surgiram na segunda Guerra Mundial, em função do pequeno número de aviões disponíveis para treinamento e o número cada vez maior de pilotos solicitados. Com a necessidade crescente de treinamento e a complexidade para se controlar uma aeronave, tornou-se necessário a criação de simuladores de vôo.

O *LINK trainer*, foi um dos primeiros simuladores, que consistia basicamente de um *cockpit* e uma plataforma movimentada a ar comprimido que possibilitava reproduzir os movimentos básicos de uma aeronave. O único recurso visual era um simples horizonte artificial e o recurso sonoro era apenas o barulho do motor. Posteriormente, com o desenvolvimento da computação, desenvolveu-se um novo tipo de simulador, que através de um ambiente sintético, projetado numa cúpula na qual se mostravam imagens de vôo, o piloto poderia ser treinado nas mais diferentes situações e condições, sem correr o risco de perder a aeronave ou até mesmo a própria vida em caso de falhas.

Da mesma forma que os militares, a NASA também dedicou-se ao desenvolvimento de vários projetos. Porém, ao contrário dos financiamentos feitos pelos militares, estes eram menores e destinavam-se à projetos mais baratos.

Um dos projetos importantes que a NASA desenvolveu implica na utilização da noção da *Telepresença.* Para exemplificar esta utilização, imaginemos uma estação espacial que necessita de reparos externos.

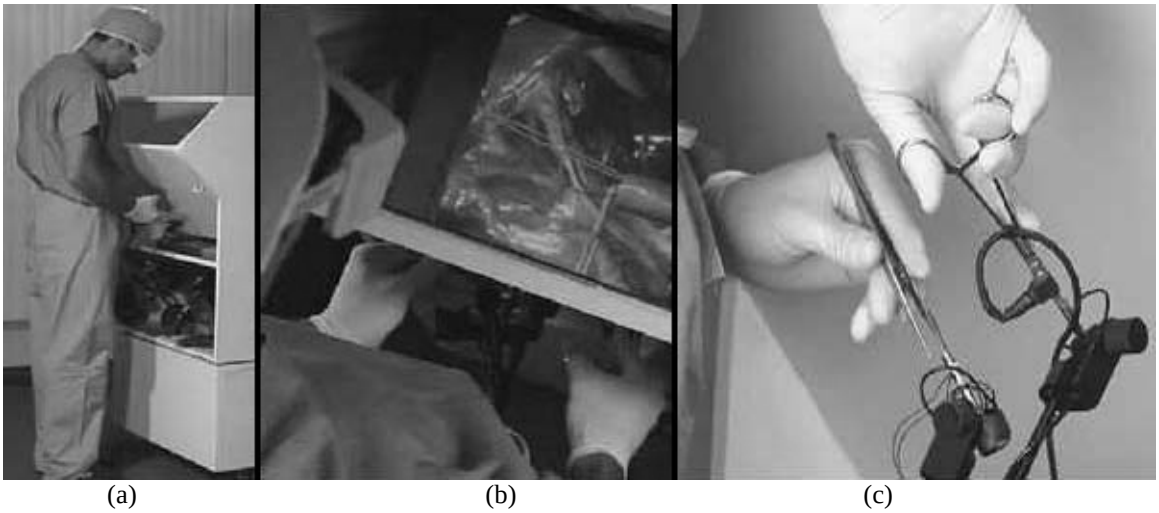
Para que os reparos fossem feitos, um astronauta teria que se deslocar para o lado de fora envolvendo um trabalho muito arriscado. Portanto, se um robô fizesse este trabalho o fator de risco de perda de vida humana estaria descartado. Esse robô possuiria então duas câmeras, uma para cada olho do operador, que estaria monitorando de dentro da estação espacial ou até mesmo da Terra. Utilizando uma luva chamada *DataGlove* que foi desenvolvida anteriormente, por Thomas Zimmermam da empresa *VPL Rearch* (em

meados da década de 80), seu braço mecânico e suas mãos articuladas seriam controlados de forma mais fácil pelo operador, atuando assim no melhor controle dos movimentos do robô.

Diferente dos simuladores de vôo, no sistema da *VPL Research* a pessoa interage diretamente no ambiente. Para isso também foi desenvolvido um HMD chamado *Eyephone* e um traje de imersão total, usado para monitorar a posição do corpo do usuário.

2.3 Aplicações da Realidade Virtual

As primeiras utilizações da realidade virtual, como foi visto, foram para simulações de situações reais como os *simuladores de vôo*, buscando proteger as pessoas e os equipamentos [CAD 97]. Esta utilização está sendo bastante utilizada pela medicina, em *simulações cirúrgicas* (ver figura 2.2). Também a telepresença (na manipulação remota de aparelhos biomédicos) é outra área que vem se desenvolvendo bastante, com sistemas de cirurgias remotas.



(a) (b) (c)

Figura 2.2 Simulação de Microcirurgias.
(a) e (b) estação com micromanipuladores, (c) sensores especializados.

Na arquitetura e na decoração, também a realidade virtual tem sido bastante difundida. Poder caminhar, tocar, interagir com a nova moradia antes de ficar pronta, ter a possibilidade de mover os móveis facilmente, interagindo com o ambiente sem muita necessidade de esforço, colocar novas mobílias e experimentar como ficará. Este, sem dúvida é um aspecto muito interessante da realidade virtual.

A realidade virtual, quando inserida em um contexto educacional pode trazer vários benefícios ao processo de ensino e aprendizagem. Principalmente pela experiência de primeira pessoa, na qual o indivíduo aprende como resultado de suas impressões e da interação. Este tipo de aprendizado é direto, natural e predomina no dia-a-dia com a interação com o mundo, diferente das experiências de 3ª pessoa na qual se conhece o mundo como é descrito por alguém.

A interação com um computador é uma experiência de 3ª pessoa. Apesar de se poder manejar o *mouse* e o teclado com um nível de habilidade tal que se torne automático, quando uma informação surge, é como ser contando por alguém. A idéia de imersão, da realidade virtual, é exatamente buscar uma forma de permitir a interação com uma informação através de uma experiência de 1ª pessoa onde o usuário não tenha que criar metáforas para relacionar o dado da tela com o real e sim possa explorar o dado como se ele de fato existisse. A realidade virtual é uma poderosa ferramenta para o ensino de química, biologia, geografia, história e física, entre outras.

Uma nova tecnologia que já se tornou realidade é a *loja virtual*, um “local” onde o usuário pode preencher seu “carrinho de compras” com os produtos visualizados nos mundos virtuais. Após a compra, basta confirmar o pedido e os produtos são entregues em casa. Também pode-se fazer encontros com outras pessoas através dos *chats* virtuais, nos quais as pessoas se expõem através de seus avatares (figuras que as representam). Estas últimas são atualmente empregadas através da Internet.

Outras aplicações da realidade virtual são as exposições, o turismo virtual (ver figura 2.3), entretenimento, modelagem, negócios, entre outros.

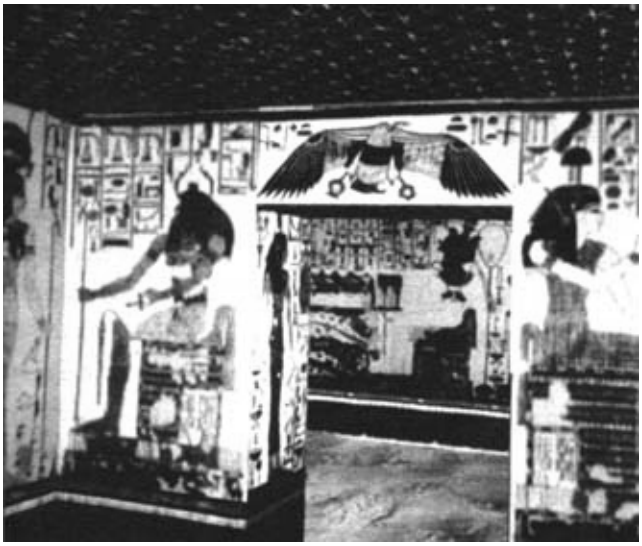


Figura 2.3 Turismo Virtual: A Tumba de Nefertari.
exposição em realidade virtual de uma vista lateral da câmara mortuária de Nefertari, mulher do faraó Ramsés II.
Foto: Ph. Plailly/Erelios/SPL/Stock Photos.

2.4 Softwares de Realidade Virtual

Diversos sistemas de realidade virtual foram desenvolvidos tais como: VREAM, Superscape, Sense8 WorldToolkit, Autodesk CDK, entre outros. Cada um deles tem seu próprio formato de arquivo para salvar mundos virtuais e usualmente não é possível o intercâmbio de dados entre eles, ou seja, não havia qualquer padronização. O único formato que todos podiam importar era o DXF, desenvolvido pela Autodesk para aplicações CAD. Esse formato não possibilitava especificar hierarquia de objetos, propriedades de materiais, mapas de textura, iluminação, entre outras coisas importantes na realidade virtual.

As diferenças entre os diversos formatos de arquivos não é apenas sintática, mas também conceitual, tornando a conversão de uma para outra mais difícil. Todos esses diferentes sistemas e formatos não possibilitavam que um trabalho feito em um sistema fosse aproveitado em outro. Havia a necessidade de se adotar um padrão.

2.5 VRML

Ao mesmo tempo que a realidade virtual amadurecia, a Internet crescia exponencialmente devido à criação de uma interface gráfica mais amigável, a WWW. E o HTML foi a linguagem que permitiu que isso ocorresse. Notava-se um grande potencial na fusão da realidade virtual com a Internet. Era necessário, então, um equivalente conceitual do HTML, uma linguagem que pudesse armazenar mundos tridimensionais assim como o HTML permite armazenar documentos.

Em 1994, Mark Pesce e Tony Parisi escreveram um programa chamado *Labyrinth*. Ele não era um verdadeiro *browser* VRML, visto que o VRML não tinha sido desenvolvido ainda. O *Labyrinth* possibilitava a recuperação de objetos 3D na Web, usando os mesmos protocolos usados por páginas HTML. Eles apresentaram o *Labyrinth* na Primeira Conferência Internacional sobre a *World Wide Web* em 1994. Então, uma *mailing list* foi criada para discussões sobre VRML, que originalmente significava *Virtual Reality Markup Language*, uma vez que HTML significava *Hipertext Markup Language*. Posteriormente o termo *Markup* foi mudado para *Modeling*, pois o armazenamento de modelos 3D era o objetivo do VRML.

2.5.1 VRML 1.0

Depois de meses de discussão, a primeira especificação do VRML foi definida. Ela herdou em grande parte o formato do OpenInventor, desenvolvida pela SGI (Silicon Graphics Inc.). Essa versão permite a criação de mundos virtuais com interação limitada. Quando um arquivo VRML (.wrl) é chamado, o *Web browser* executa o visualizador VRML, que possibilita a navegação e visualização desse mundo 3D. Estes mundos possibilitam a criação de objetos com hiperlinks associados para outros mundos, uma página HTML, ou qualquer outro tipo MIME.

Um mundo virtual VRML é um conjunto de objetos que podem conter geometrias, sons MIDI ou WAV, imagens JPEG, luzes, etc. Estes objetos são chamados nodos. Eles são

organizados em estruturas hierárquicas chamadas *scene graphs*, que definem uma ordem nos nodos. Em uma *scene graph* nodos que aparecem antes afetam os posteriores. Um mecanismo definido para delimitar os efeitos, chamados nodos separadores, permite que uma parte da *scene graph* seja isolada de outra.

2.5.2 VRML 2.0

Em 1996, o VRML Architecture Group (VAG) fez um pedido de propostas para a especificação do VRML 2.0. Seis propostas foram recebidas e debatidas por dois meses. Foi realizada a votação pela comunidade VRML, e a proposta escolhida foi a *Moving Worlds*, da SGI, sendo oficializada em março de 1996.

O VRML 1.0 fornece meios para criar e visualizar mundos 3D estáticos, apresentando uma interface "fria" com o usuário. Na versão 2.0, isso foi melhorado através de mundos estáticos mais aperfeiçoados, interação, animação, possibilidade de *scripts* e prototipação.

A melhoria dos mundos estáticos é observada através do possibilidade de maior realismo nas cenas. Isso é alcançado através da criação de cenas com fundos apresentando o chão e o céu, com a adição de paisagens com montanhas e nuvens, embaçando objetos que estão mais distantes, criação de terrenos irregulares, etc. Além disso, oferece também recursos de som 3D para aumentar o realismo do ambiente gerado.

A aparência de se mover em uma cena "morta" é desfeita nessa versão com a adição de *efeitos interativos*. Novos nodos do tipo sensor proporcionam essa interação: quando você entra em uma área do mundo ou clica em um objeto, o sensor pode setar um evento como acender uma luz. Outro tipo de sensor pode controlar a passagem do tempo, podendo ser usado em animações repetitivas. Outro recurso é a detecção de colisão, assim não será mais possível atravessar por objetos sólidos.

Houve, também, a inclusão de objetos de animação chamados interpoladores, que permitem criar animações de objetos do mundo 3D e então executá-las em outro momento. Com eles, pode-se criar objetos móveis como um pássaro voando, uma porta se abrindo, um robô se movendo. Pode-se mudar a cor ou a forma de um objeto, entre outros recursos de animação. É possível, também, a criação de um passeio guiado por um caminho determinado no mundo virtual.

Um recurso incluído é o nodo tipo *script*, que é fundamental nessa nova versão. Ele possibilita a criação de objetos animados em um mundo, e dá a eles um aspecto mais "inteligente" atribuindo ações determinadas. Esses efeitos são obtidos por meio de eventos, um *script* lê entradas de um sensor e gera eventos baseados nessa entrada que mudará outros nodos nesse mundo.

Outro recurso disponível no VRML 2.0 é a prototipação. Ela permite o encapsulamento de grupo de nodos, originando um novo tipo de nodo, um protótipo. Pode-se, então, criar instâncias desse novo tipo com diferentes valores de campos característicos. Dessa forma, um protótipo pássaro pode ser instanciado com diversas cores: um pássaro amarelo, outro vermelho; ou com outras características particulares.

2.5.3 Definições Básicas

- Nodo *(Node)*: É o componente fundamental de uma cena em VRML. Pode ser definido como abstrações dos objetos e conceitos do mundo real, por exemplo, esfera, cubos, luzes e descrição de materiais. Os nodos contém campos e eventos.

- Nodos de Geometria (G*eometry node*): é representado pelos tipos *Box, Cone, Cylinder, ElevationGrid, IndexedFaceSet, IndexedLineSet, PointSet, Sphere, Text.*. Este nodo contém uma descrição de pontos tridimensionais, linhas, superfícies, *string* de textos e objetos sólidos.

- Nodos filhos *(Children node*): São nodos nos quais vários nodos filhos são agrupados e estes são afetados pelas transformações feitas no nodo pai. Serve para criar uma hierarquia de transformações que são herdadas de pai para filho.

- Eventos *(Event*): São utilizados para a troca de mensagens de um nodo para outro, através de uma rota. Os eventos sinalizam mudanças nos valores dos campos, estímulos externos, interações entre nodos, etc.

- Campo *(Field*): Os parâmetros do campo distinguem um nodo de outro do mesmo tipo. Campos podem conter vários tipos de dados e um ou mais valores.

- Nodos de Aparência *(Appearance node*): O nodo de Aparência é representado pelos tipos *Appearance, FontStyle, ImageTexture, Material, MovieTexture, PixelTexture e TextureTransform*, e seu objetivo é controlar a aparência renderizada dos nodos geométricos (objetos) as quais estão relacionados.

- Rota (*Route)*: É a conexão entre o nodo gerador do evento e o nodo receptor do evento.

2.5.4 Características da VRML 2.0

Como os objetos VRML podem ser visualizados através de um *browser*, naturalmente estes são independentes de plataforma, podendo ser visualizado por qualquer tipo de computador, sendo disponível para diferentes ambientes e lugares através da Internet.

Um objeto VRML é um arquivo texto com extensão .wrl, .wrz ou .wrl.gz (os dois últimos compactados com gzip). Um objeto em VRML inicia sempre com a linha “#VRML V2.0 utf8”. O identificador “utf8” permite utilizar caracteres internacionais, como acentos, nos modelos VRML. Toda a especificação de objetos, *links*, atributos ou transformações, é colocada dentro de blocos que são definidos pela expressão *Transform*.

Os comentários em VRML iniciam sempre pelo símbolo # e valem na linha em que este aparece. VRML é *case-sensitive*, ou seja, diferencia letras maiúsculas e minúsculas. Quando se cria um objeto em VRML ele é colocado no centro da tela (0,0,0), isto se o usuário não aplicar uma movimentação no objeto.

O sistema de coordenadas é o cartesiano, seguindo a regra da mão direita. Inicialmente, os objetos são projetados na direção positiva do eixo Z, com a direção positiva do eixo X para a direita e a direção positiva do eixo Y para cima. Para mudar esse padrão de projeção muda-se a câmera virtual ou usam-se transformações. A unidade de comprimento é o metro e a de ângulo o radiano. O modelo de cores na VRML é o RGB, com valores normalizados entre zero e um.

As primitivas básicas da VRML são os nodos *Box*, *Cone*, *Cylinder* e *Sphere*. Já as formas avançadas são definidas por *Text*, *Evaluation Grid*, *Extrusion*, *IndexedFaceSet*, *IndexedLineSet* e *PointSet*. Os objetos em VRML podem sofrer transformações (escala, rotação e translação), terem definidos seus materiais, texturas (inclusive animadas) e transformações nas texturas. Os nodos podem ser definidos como um grupo em VRML com a capacidade de definir um conjunto em que se possa ser reutilizado em outro local.

Também são características da VRML: hyperlink para outros mundos VRML, para uma página HTML ou outro tipo que o *browser* pode ler, iluminações do tipo pontuais, direcionais ou *spot*, animações, o *browser* ser avisado quando ocorrer uma colisão, som espacial (som 3D) usando arquivos tipo MIDI ou WAV, entre outros. Maiores informações sobre as características da VRML podem ser vistas em [WEB 99].

No exemplo a seguir (figura 2.4) veremos um exemplo de um arquivo VRML e sua imagem correspondente, no qual um cone irá ser rotacionado em 90° no eixo Y, enquanto o outro cone manterá sua posição normal, transladado de 5 unidades no eixo X.

```
#VRML V2.0 utf8

Transform {
   rotation 0 1 0 1.57 #eixo 0,1,0 ângulo 1.57 radianos
   children [
      Shape {
         geometry Cone{
            bottomRadius 2
            height 5
         }
      }
   ]

}

Transform {
   translation 5 0 0
   children [
      Shape {
         geometry Cone{
            bottomRadius 2
            height 5
         }
      }
   ]
}
```

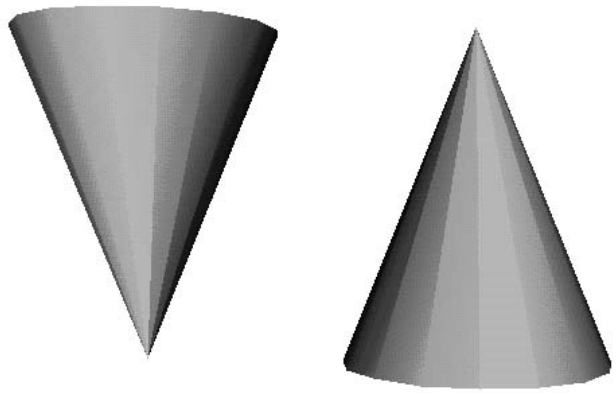


Figura 2.4 Exemplo de arquivo VRML.
código fonte VRML e imagem correspondente.

Existem atualmente várias páginas onde se pode encontrar maiores detalhes sobre a VRML (especialmente em [WEB 99]) e aprender a fazer mundos virtuais ([SIM 99] por exemplo).

2.5.6 VRML97

VRML97 é o nome informal do padrão internacional (ISO/IEC 14772-1:1997). É quase idêntico ao VRML 2.0, com muitas melhorias na documentação, mas com poucas diferenças funcionais.

O padrão internacional VRML97 foi desenvolvido pelo *Joint Technical Committee 1* (JTC 1) do ISO em parceria com o VRML Consortium no final de 1997. Para isso, teve como base o texto “VRML 2.0 Specification” (distribuído a partir de agosto de 1996).

2.5.6 Importância da VRML

A Internet, e especialmente a WWW, está mudando a maneira com que a informação é espalhada pelo mundo. A HTML é a base da WWW, fornece a interface para incorporar imagens gráficas bidimensionais (como GIF e JPEG). O paradigma é adequado para algumas formas de interação, mas têm uma série de limitações tanto na estrutura da

informação quanto na capacidade de interação. A VRML permite o próximo nível de interação, levando a Web além do paradigma orientado a documentos para um baseado em mundos virtuais 3D. Uma analogia provável das capacidades interativas da VRML em comparação com a HTML, seria a experiência de uma pessoa lendo um livro e outra jogando video game. São amplas as aplicações possíveis da VRML, como por exemplo, negócios, entretenimento, manufatura, ciência e educação.

A VRML é baseada no sucesso da HTML, que dispõe de uma interação limitada em duas dimensões. A fim de superar estas limitações, a VRML vem realizando grandes avanços. Primeiro, acrescenta a possibilidade de descrever objetos geométricos tridimensionais. Segundo, foi desenvolvida para ser um formato capaz de não só descrever estes objetos 3D, mas também descrever o procedimento da interação a ser aplicada quando um usuário encontra uma informação e aventura-se em interagir com esse objeto. Esta é a combinação da definição de uma descrição tridimensional com as novas capacidades introduzidas à VRML [WEB 99].

2.5.7 Visualizadores

A Computação Gráfica se divide, basicamente, em duas áreas: Modelagem e Visualização [ADA 94]. A primeira está relacionada com a construção dos objetos da cena a ser visualizada, utilizando bases matemáticas para isso. A segunda busca uma forma de representação visual dos objetos construídos. Esta visualização é uma seqüência de operações para transformar as informações dos objetos modelados no universo para o espaço de tela [MAN 94].

Sendo o objetivo deste trabalho fazer a integração das técnicas de modelagem com VRML, a modelagem geométrica será abordada mais detalhadamente no próximo capítulo.

Como o objetivo da VRML é a Internet, a visualização é realizada através de *browsers*. Para isto, é necessário que se instale um *plug-in* adequado. Atualmente, existem alguns visualizadores disponíveis para diferentes plataformas. Os mais usados para testar os

modelos gerados foram: Cosmo Player 2.1 da SGI, Microsoft VRML 2.0 Viewer, WorldView da Intervista e Viscape VRML da Superscape.

2.5.8 O Futuro da VRML

Depois do desenvolvimento do padrão VRML (ISSO/IEC), foi decidida que características mais ambiciosas (como interação multi-usuário ou criaturas autônomas que podem reagir com o ambiente) não fazem parte da versão original da VRML. Atualmente estão sendo estudada implementações e testes em muitas áreas, como: Um formato comprimido para aumentar a velocidade de transmissão dos arquivos e do *parsing*, um dispositivo de interface entre o mundo VRML e um ambiente externo, animação de humanóides para dar uma representação VRML padrão para humanóides, entre outros. Uma lista de grupos de trabalhos apoiados pelo *VRML Consortium* pode ser encontrado em [WEB 99].

3. MODELAGEM GEOMÉTRICA

Um sistema de realidade virtual deve ter a capacidade de determinar a forma, a posição dos objetos, quando há intervenção entre estes objetos, transparências, reflexões e texturas dos objetos, entre outros aspectos. Para isto, é necessário inicialmente modelar os objetos como sólidos geométricos, e posteriormente atribuir a estes objetos propriedades físicas como: transparência, reflexão, textura, etc.

A necessidade de modelar objetos como sólidos levou ao desenvolvimento de uma grande variedade de representações. A modelagem geométrica trata do problema da criação, manipulação e topologia dos objetos gráficos no computador.

Este capítulo dá uma breve introdução a algumas das principais técnicas usadas para representar objetos.

3.1 Instanciamento de Primitivas

Neste caso, o sistema de modelagem define um conjunto de primitivas sólidas 3D relevantes a uma área de aplicação. Estas primitivas são parametrizadas não apenas em termos de transformações, mas também em termos de outras propriedades. Por exemplo, uma primitiva de um objeto pode ser uma pirâmide regular com um número de faces definido pelo usuário. Instanciamento de primitivas são usados normalmente para objetos

complexos, como engrenagens ou parafusos, que seriam difíceis de definir por outras técnicas, que pelas quais também não seria possível definir um bom nível de parâmetros. Uma engrenagem em primitivas, por exemplo, poderia ser parametrizada por diâmetro e número de dentes como mostra a figura 3.1.

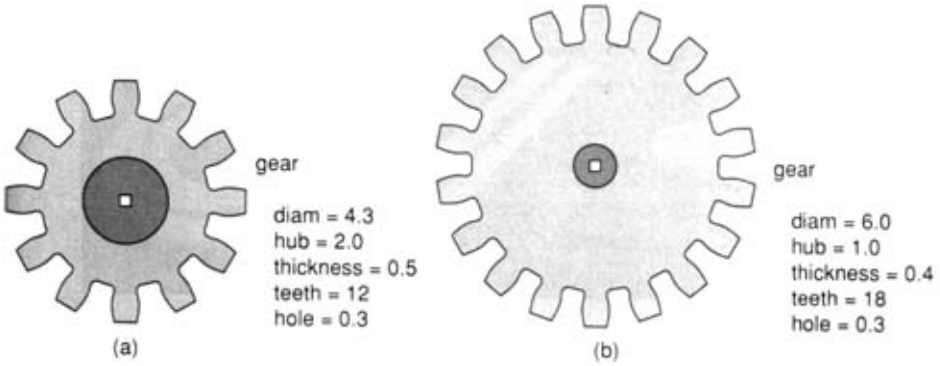


Figura 3.1 Duas engrenagens definidas por instanciamento de primitivas.

Em geral, as primitivas geométricas são objetos simples de descrever e representar, constituindo os blocos básicos da construção de modelos. A figura 3.2 mostra um exemplo das primitivas mais simples de serem descritas: O cubo (a) que é descrito pelo seu centro e tamanho das arestas e a esfera (b) é descrita também pelo centro e pelo raio.

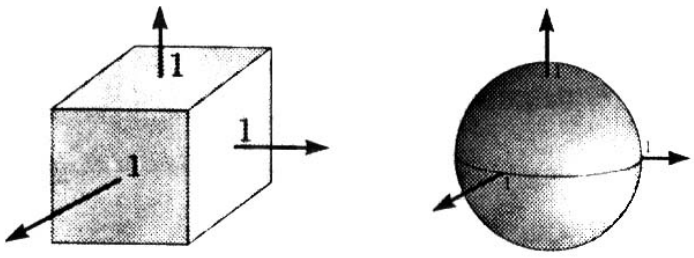


Figura 3.2 Primitivas geométricas no espaço.

Estas primitivas podem sofrer algumas transformações. Tais transformações são utilizadas tanto para posicionar as primitivas no espaço quanto modificar a geometria destas. No posicionamento das primitivas são utilizados os movimentos de rotação e translação. Para modificar a geometria, uma transformação bastante utilizada é a mudança de escala, que permite uma mudança das dimensões da primitiva. Usando uma mudança de escala linear, pode-se transformar um cubo em um paralelepípedo qualquer. Usando

transformações projetivas, pode-se obter figuras das mais variadas através de um simples cubo. O uso de transformações para modificar a geometria das primitivas permite reduzir o número de primitivas.

3.2 Sweep

Movendo um objeto ao longo de uma trajetória através do espaço pode-se definir um novo objeto através de uma técnica chamada de *sweep*. O tipo mais simples de *sweep* é definido por uma área 2D movida ao longo de um caminho normal ao plano da área para criar um volume. Este tipo é conhecido como *sweep* translacional, técnica de varredura linear ou extrusão [HER 94]. Algumas extensões envolvem escalar a outra face para fazer objetos de formato cônico. O *Sweep* rotacional, também chamado de superfície de revolução ou torneadas [HER 94], é definido pela rotação de uma área sobre um eixo. A figura 3.3 mostra dois objetos gerados usando o *sweep* translacional e o *sweep* rotacional.

O *sweep* é uma maneira natural e intuitiva de construir uma variedade de objetos. Por esta razão, muitos sistemas de modelagem permitem ao usuário construir objetos por *sweep*. A figura 3.3 mostra algumas figuras feitas pela técnica de modelagem por *sweep*.

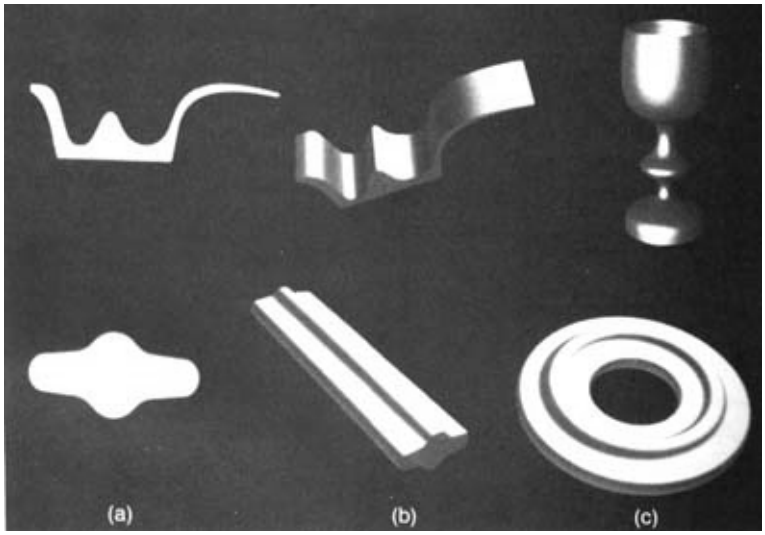


Figura 3.3 Sweep.
(a) área 2D usada para definir (b) os *sweeps* translacionais e
(c) os *sweeps* rotacionais. (criado usando o sistema Alpha_1)
foto: Universidade de Utah.

Esta foi a técnica escolhida para realizar a integração com o VRML, portanto, ela será melhor detalhada no próximo capítulo.

3.3 Representação de Limites

O modelo por representação de limites define um sólido indiretamente através da representação das suas superfícies limitantes. São baseados em uma visão orientada para superfícies, em outras palavras, eles representam um objeto sólido através da subdivisão deste objeto em faces.

Um modelo por representação de limites é considerado válido se ele consegue definir os limites de um objeto sólido. Deste modo, os critérios de validade de um modelo por representação por limites incluem as seguintes condições:

1. O conjunto de faces que compõem o modelo se fecham, ou seja, formam um invólucro completo em torno do objeto a ser representado e sem partes pendentes;
2. As faces do modelo não interseccionam outras faces, exceto as faces que possuem vértices e arestas em comum;
3. As faces são superfícies simples que não se auto-interseccionam.

A primeira e a segunda condições excluem objetos que se auto-interseccionam, e a terceira condição exclui objetos abertos como por exemplo uma caixa aberta.

Computacionalmente é conveniente dividir uma superfície do modelo em faces, sendo que cada face é limitada por um conjunto de vértices e arestas. Os limites de um objeto podem ser divididos em faces, vértices e arestas de forma ilimitada, ou seja, não existe uma forma única de representar os limites de um objeto.

Modelos por representação de limites são difíceis de se descrever diretamente, porém é possível utilizar recursos gráficos e interativos para facilitar o projeto de objetos por representação de limites.

3.4 Representação por decomposição volumétrica

Este é um modelo por decomposição (*Spatial-Partioning*), ou seja, os sólidos são descritos através da união de blocos básicos. Primitivas podem variar no tipo, tamanho, posição, parametrização e orientação, e a maneira como os blocos são combinados é que distingue cada variação deste esquema de representação.

3.4.1 Enumeração Exaustiva

Pode-se interpretar um sólido como um conjunto contíguo de pontos tridimensionais, porém, não é possível enumerar todos os pontos de um objeto. Partindo desta afirmação, pode-se facilmente representar um sólido enumerando-se pequenos sólidos básicos contidos (totalmente ou parcialmente) no sólido. Estes pequenos sólidos são muitas vezes denominados de *voxel* (volume elements – uma analogia ao *pixel*). O tipo mais comum de *voxel* é o cubo, e a representação de espaços por uma matriz de cubos é denominada *cuberille*. Assume-se que estes cubos são de tamanho uniforme e possuem a mesma orientação, ou seja, eles formam uma subdivisão regular do espaço tridimensional.

Esta forma de representação é denominada enumeração exaustiva. A figura 3.4 mostra um objeto representado por esta técnica.

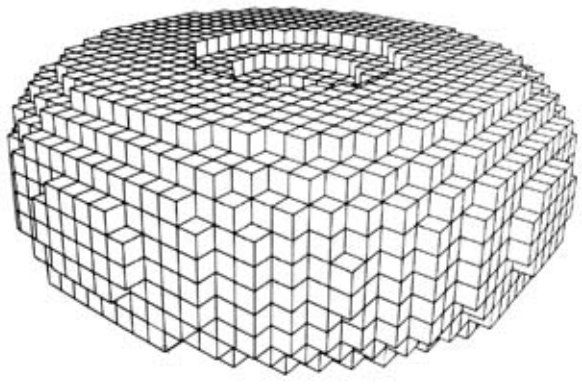


Figura 3.4 Torus representado por enumeração exaustiva.

Cada cubo pode ser descrito completamente em termos de seus vértices. Obviamente, devido à regularidade existente, pode-se simplesmente descrever um cubo através de um único vértice (ou através de seu centro). Quando representamos um objeto usando a enumeração exaustiva, controla-se somente a presença ou não de cubos em cada posição da matriz. Para representar um objeto, somente é necessário decidir quais as posições são ocupadas e quais não são. Um mecanismo de descrição de sólidos por enumeração exaustiva pode ser composto por uma lista de todos os cubos que são considerados como constituintes do objeto.

A enumeração exaustiva é um esquema de modelagem que representa não somente parte ocupada pelo material, mas a parte não ocupada também. Um uso da enumeração exaustiva é a sua utilização como forma auxiliar de acelerar operações em outras formas de representação.

A enumeração exaustiva é obviamente uma técnica “aproximativa”. Isto significa que superfícies que não são coplanares com qualquer dos planos coordenados não poderão ser representados com perfeição, somente de forma aproximada. Entretanto, qualquer objeto pode ser aproximadamente representado por este esquema.

3.4.2 Representação por “Octree”

Os pontos positivos da enumeração exaustiva contrastam com as grandes desvantagens desta técnica que são o enorme consumo de memória e a falta de exatidão. *Octree* é uma variante da enumeração exaustiva, concebida para otimizar o armazenamento. Ela é derivada da *quadtree* (ver figura 3.5), um formato para codificar imagens.

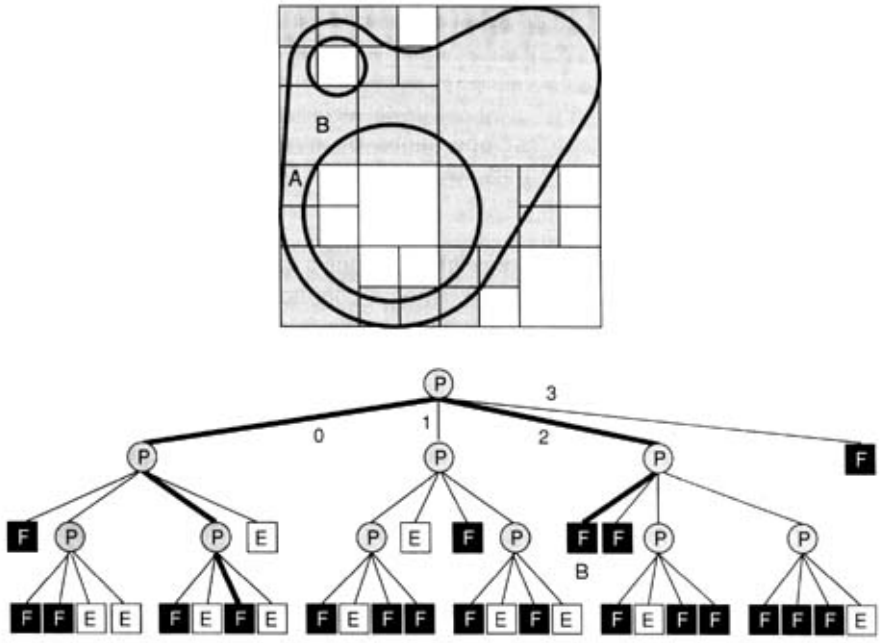


Figura 3.5 Objeto usando a representação quadtree e sua estrutura de dados.
F = cheio, P = parcialmente cheio e E = vazio

A idéia básica de ambos, é usar o poder da subdivisão binária de "*dividir e conquistar*". Uma *quadtree* é derivada de sucessivas subdivisões de um plano 2D em ambas as dimensões formando quadrantes, como é mostrado na figura 3.5. Quando um quadrante é usado para representar uma área no plano, cada quadrante pode ser cheio, parcialmente cheio ou vazio (também pode ser chamado preto, cinza e branco, respectivamente), dependendo de quanto o quadrante está ocupado na área. Um quadrante parcialmente cheio é dividido recursivamente em subquadrantes, até que todos os quadrantes estejam homogêneos (cheio ou vazio), ou até chegar a uma determinada profundidade.

As sucessivas subdivisões podem ser representadas por uma árvore cujos nodos folhas são os quadrantes cheios e vazios e os nodos internos são os parcialmente cheios.

Com base neste fato, o sistema de representação por *octree* utiliza uma subdivisão espacial recursiva de um espaço de interesse em seus oito octantes, de modo a formar uma árvore com oito nodos filhos ou ramos. A figura a seguir mostra a representação por *octree*.

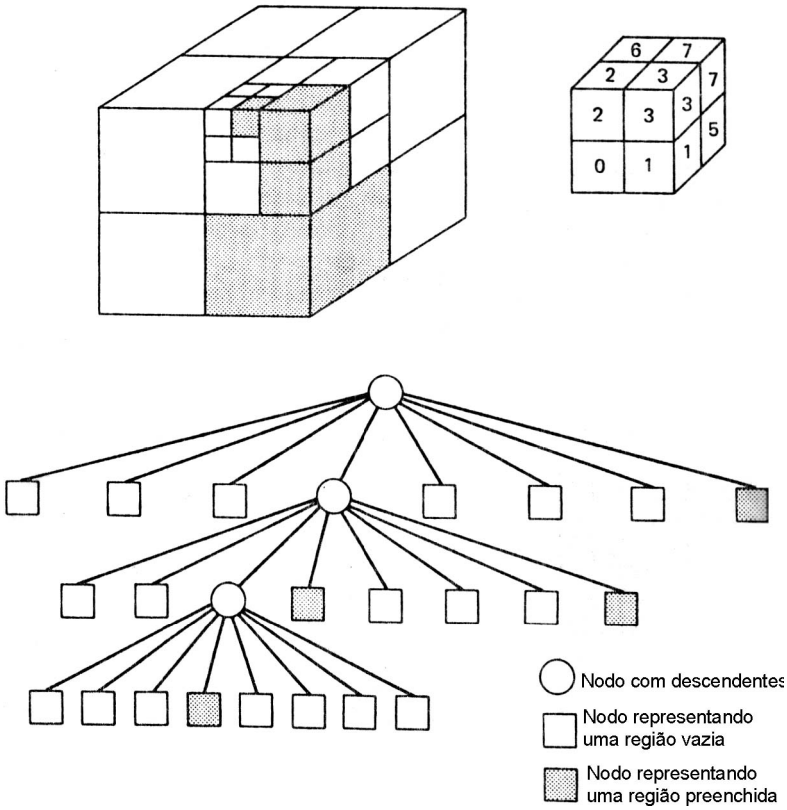


Figura 3.6 Representação por octree.

No contexto da modelagem de sólidos, os objetos *octrees* são comumente construídos a partir de primitivas sólidas. Com relação a estas primitivas, cada nodo de um octante pode possui os mesmos tipos de estado da *quadtree*: cheio, parcialmente cheio ou vazio, representando respectivamente:

- O nodo está completamente no exterior da primitiva.
- O nodo encontra-se completamente no interior da primitiva.
- O nodo encontra-se parcialmente no interior da primitiva.

A funcionalidade de um modelador *octree* deve incluir algoritmos que se enquadrem nas seguintes categorias:

- Geração de árvores que criem *octrees* a partir de primitivas parametrizadas ou outros tipos de modelos geométricos;

- Conjunto de operações que utilize dois objetos *octree* e calcule um novo objeto *octree* realizando operações como união, diferença ou interseção (vide item 3.5 – Geometria Sólida Construtiva).
- Operações geométricas que peguem um objeto *octree* e calcule um novo objeto *octree* que possua translação, rotação, escala ou outro tipo de operação geométrica.
- Procedimento de análise que calcule propriedades como volume ou área superficial de um objeto *octree*.
- Exibição que crie uma imagem gráfica de um objeto modelado por *octree*.

A propriedade essencial do *octree* é que ele armazena as informações de um objeto de uma maneira ordenada, Desta forma é possível a criação de algoritmos relativamente simples para a visualização de objetos *octree* de forma completa e clara.

Assim como a enumeração exaustiva, a *octree* é uma técnica de representação "aproximativa", de modo que o modelo é somente semelhante ao objeto real. Também, toda representação *octree* pode ser considerada válida.

3.4.3 Decomposição de Células

Outra técnica utilizada para resolver problemas da enumeração exaustiva, mas preservando as suas propriedades, utiliza outros tipos de elementos básicos além de cubos. Estes esquemas são denominados esquemas por decomposição de células. Cada sistema de decomposição de célula define um conjunto de células primitivas que são de característica paramétrica e muitas vezes curvas.

Além de um conjunto de *tipos de células*, também possui um operador de união para "aglutinar" estas células em um objeto. Um sólido é, portanto, modelado através de um conjunto de células disjuntas, ou seja, cada célula se une à outra nas suas superfícies limitantes apesar de não possuir pontos interiores em comum.

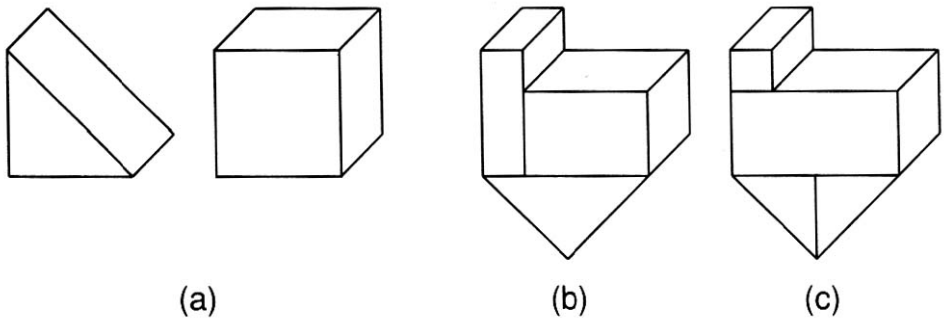


Figura 3.7 Decomposição de células.
As células mostradas em (a) podem ser transformadas para construir o mesmo objeto em (b) e em (c) de diferentes maneiras. Uma simples célula é suficiente para causar ambigüidade.

Embora um objeto por decomposição de células seja único e não ambíguo, não é necessariamente único, como mostra a figura 3.7. A validade de uma decomposição de células é difícil de estabelecer. Enquanto a enumeração exaustiva e a representação por *octree* possuíam propriedades estruturais que garantiam a sua validade, a decomposição por células, em geral, é somente um conjunto desordenado de células, de modo que para checar a validade de um objeto modelado por decomposição de células é necessário testar todos os pares de células para verificar a sua conexão. Além disto, normalmente é muito difícil de criar um objeto através de decomposição de células, de modo que, da mesma forma que os modelos por enumeração exaustiva e por *octree*, os modelos são criados a partir de uma conversão de um outro modelo.

3.4.4 Árvore Binária de Decomposição Volumétrica

A *octree* divide o espaço em planos que são sempre perpendiculares entre si e que bifurcam todas as três dimensões nos três níveis. A Árvore Binária de Decomposição Volumétrica (*Binary Space-Partitioning Trees – BSP tree*), divide o espaço recursivamente em pares de subespaços, separados por um plano. Cada nodo interno da *BSPt* é associado a um plano e tem dois pontos filhos, um para cada lado do plano. O nodo esquerdo representa o filho da esquerda ou atrás do plano, o nodo direito representa o filho da direita ou à frente do plano. Se um lado é subdividido novamente, então ele é a raiz de uma subárvore, caso contrário, é uma folha da árvore, representando uma região de dentro ou de fora do poliedro. Estas regiões homogêneas são chamadas células *in* e *out*. A figura 3.8 ilustra esta

estrutura. Para limitar a precisão com que estas operações são realizadas, cada nodo tem associado a ele uma profundidade associada ao plano.

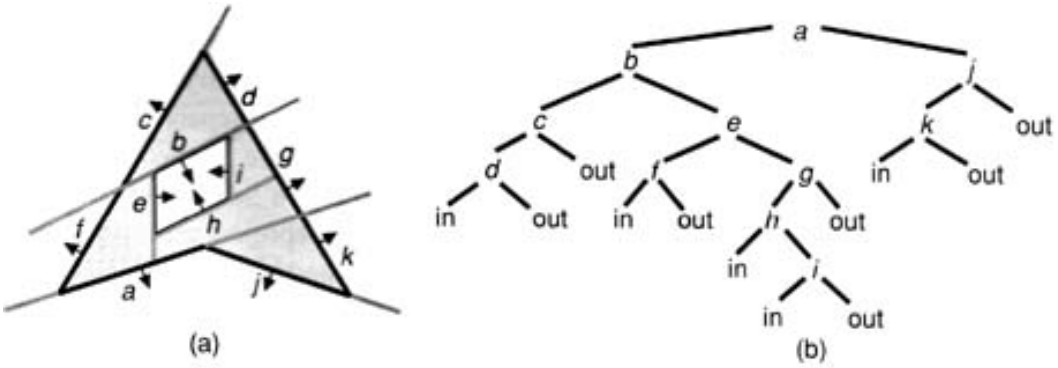


Figura 3.8 Uma representação da BSPt em 2D.

3.5 Geometria Sólida Construtiva – CSG

O método mais conhecido de representação por construção é o método CSG, que vêm do inglês "*Construtive Solid Geometric*" (Geometria Sólida Construtiva) no qual sólidos complexos são compostos de primitivas geométricas através de transformações no espaço e operações booleanas. Em geral estas primitivas geométricas são objetos simples de descrever e representar no computador, como cubos ou esferas.

As transformações são utilizadas na representação CSG com a finalidade de posicionar as primitivas no espaço ou modificar a geometria de uma primitiva. No posicionamento são utilizados movimentos de rotação e translação. Para modificar a geometria das primitivas é bastante usada a mudança de escala, permitindo uma mudança das dimensões das primitivas. Usando essas transformações a primitiva dada por um cubo pode ser transformada para obter paralelepípedos das mais variadas dimensões. Também pode-se obter um prisma qualquer usando transformações projetivas. O uso de transformações para modificar primitivas permite reduzir o número de primitivas.

Após transformar e posicionar devidamente as primitivas no espaço, o sistema CSG se utiliza das operações booleanas para combinar as diversas primitivas e criar o modelo final. As operações booleanas são a união ($\cup$), interseção ($\cap$) e diferença (–) de conjuntos.

Essas operações são ilustradas na figura 3.9.

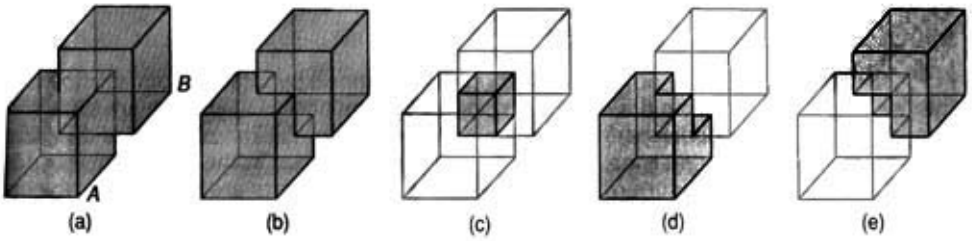


Figura 3.9 Operações booleanas.
(a) objetos A e B, (b) A ∪ B, (c) A ∩ B, (d) A – B e (e) B – A

Os objetos são armazenados como uma árvore em que os operadores são nodos internos e as primitivas são as folhas. Os nodos podem representam operadores booleanos ou transformações (translação, rotação ou escala). A figura 3.10 ilustra uma árvore deste tipo.

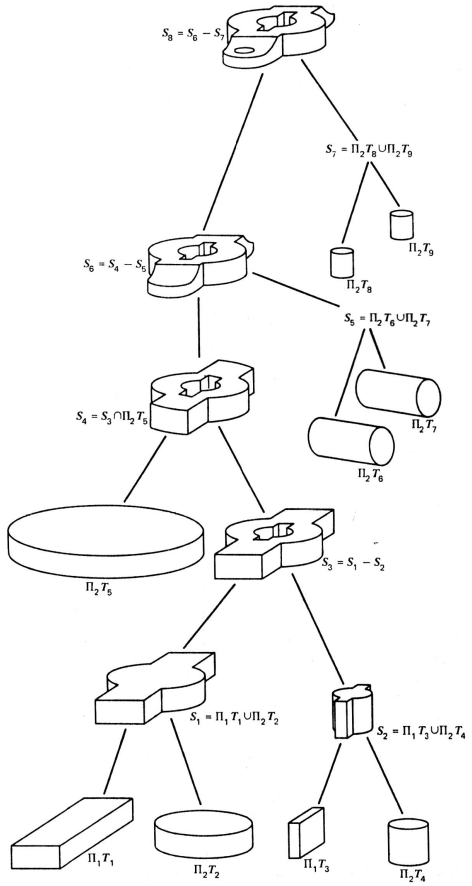


Figura 3.10 Árvore com operações booleanas gerando uma peça mecânica a partir de primitivas.

Porém, algumas combinações de primitivas CSG não satisfazem completamente as noções de solidez, uma vez que a interseção de dois objetos pode resultar além de um objeto sólido, um plano, uma linha, um ponto e até mesmo um conjunto vazio (ver figura 3.11). Este efeito é extremamente indesejável em alguns casos em que se precisa trabalhar sobre as interseções de pares de objetos para testar se eles podem ser encaixados. Nestas situações pode-se considerar que objetos que apenas se tocam não possuem uma área de interseção.

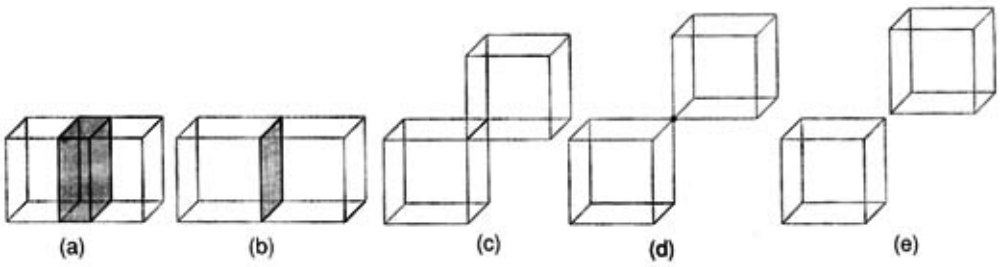


Figura 3.11 Interseção entre dois cubos.
Pode produzir (a) um sólido, (b) um plano, (c) uma linha ou (e) um conjunto vazio.

3.6 Considerações Finais Sobre Modelagem Geométrica

Neste capítulo foram estudadas algumas das principais técnicas existentes usadas para representar objetos, dando uma breve introdução das mesmas. Como o objetivo deste trabalho deste trabalho é integrar técnicas de modelagem com a VRML, foi escolhida uma técnica dentre essas estudadas para realizar essa integração.

A técnica escolhida para realização do trabalho foi a técnica de modelagem por *sweep*. Por isto, ela será o tema do próximo capítulo, onde também serão descritos seus principais tipos. Os algoritmos serão estudados junto com a descrição da construção do sistema de modelagem desenvolvido junto a este trabalho, no capítulo da implementação do sistema.

4. TÉCNICA DE MODELAGEM POR SWEEP

Como foi visto no capítulo anterior, existem muitas maneiras de modelar objetos sólidos. O objetivo deste trabalho é o de integrar técnicas de modelagem com a VRML. Sendo assim, foi escolhida a técnica de modelagem por *sweep* para fazer essa integração. Esta técnica é uma maneira natural e intuitiva para construir uma grande variedade de objetos, sendo usada em muitos sistemas de modelagem de objetos tridimensionais. Esta foi a principal razão da escolha da técnica de *sweep* para a realização deste trabalho.

Deslocar um objeto ao longo de uma trajetória no espaço, definindo um novo objeto, é chamado de *sweep*. É baseada na noção de que movendo uma curva ou uma superfície ao longo de um caminho é gerado um objeto tridimensional. O tipo mais simples, é definido por uma área bidimensional que, ao ser deslocada por um caminho linear normal ao plano dessa área, gera um objeto com volume e de base igual à área original [FOL 96]. Este tipo particular de *sweep* é conhecido como *sweep* translacional ou extrusão.

O *sweep* é uma maneira prática e eficiente para modelar partes mecânicas feitas por cortes transversais (figura 4.1) ou para modelar objetos plásticos ou metálicos feitos por extrusão. Esta técnica também é usada para detectar possíveis interferências entre peças ou mecanismos [MOR 85].

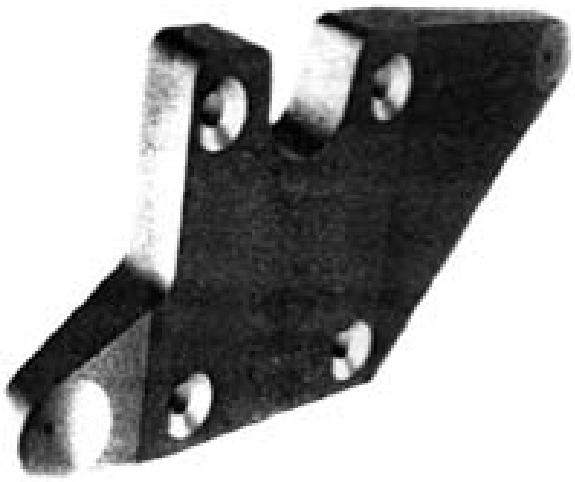


figura 4.1 – Modelo de uma peça de avião.
Gerado por um sistema usando a técnica de *sweep*.

Para modelagem de sólidos, dois ingredientes são necessários: um objeto a ser movido e uma trajetória descrevendo o seu movimento. Um objeto pode ser uma curva, uma superfície ou um outro sólido. E a trajetória é um caminho definido analiticamente. Mortenson usa o termo *generator* para definir o objeto e *director* para a trajetória [MOR 85]. Os dois principais tipos de trajetória são a **translacional** e a **rotacional**.

Os principais tipos de *sweep* serão descritos a seguir. Os algoritmos serão detalhados no próximo capítulo, no qual será abordada a implementação dos mesmos. Algumas das ilustrações de objetos tridimensionais (figuras 4.3 a 4.5 e figuras 4.8 a 4.12) que serão apresentadas a seguir foram geradas usando o sistema de modelagem desenvolvido junto a este trabalho.

4.1 Sweep Translacional

É o tipo mais simples de *sweep* como vimos acima. Neste caso, parte-se de uma área 2D que é deslocada por uma trajetória normal ao plano desta área para gerar o sólido (figura 4.2). Afonso Hermida classifica esta técnica pelo nome de *Extrusão* [HER 94], já Jonas Gomes por *Técnica de Varredura Linear* [GOM 98].

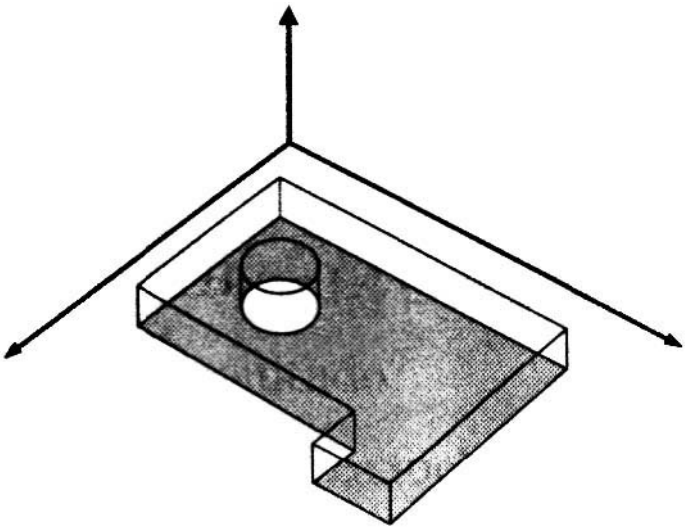


Figura 4.2 Sweep Translacional.

4.1.1 Sweep Translacional Simples

É o método de *sweep* translacional tradicional básico. Tendo-se um polígono de N vértices e uma altura de translação H, obtêm-se o polígono superior desse sólido simplesmente transladando o polígono gerador na direção perpendicular a base do polígono (figura 4.3). Isto faz com que ambas as faces (superior e inferior) sejam idênticas. As faces laterais serão sempre retangulares [CAS 91].

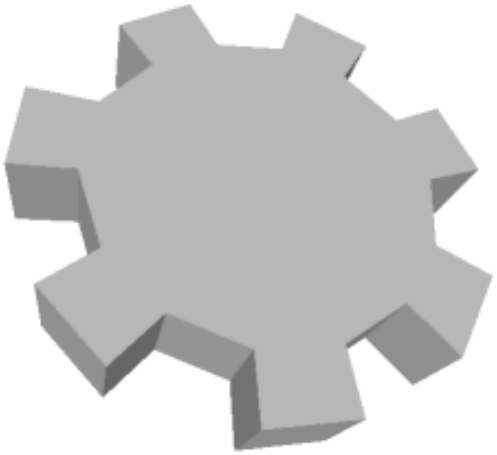


Figura 4.3 Sweep Translacional Simples.

Esta é uma maneira natural de representar objetos feitos por extrusão de materiais metálicos ou plásticos.

4.1.2 Sweep Translacional Cônico

Uma simples extensão do *sweep* translacional, consiste de se alterar a dimensão da base formadora de forma que este deslocamento forme objetos de forma cônica assim como fazer um deslocamento de maneira não perpendicular ao plano em que está a base formadora [FOL 96]. Este *sweep* é semelhante ao translacional simples, os vértices do polígono gerador além de serem transladados eles convergem para um ponto chamado de "ponto de fuga". A altura do objeto será entre a base e o ponto de fuga, gerando um tronco de prisma. As faces laterais serão em forma de trapézios. Caso a altura coincida com a distância do ponto de fuga, a face superior será reduzida a um único vértice. Nesse caso as faces laterais serão triangulares [CAS 91]. Dois exemplos de objetos gerados pela técnica de *sweep* translacional cônico são mostrados na figura 4.4. A segunda figura mostra o caso no qual a altura e o ponto de fuga coincidem.

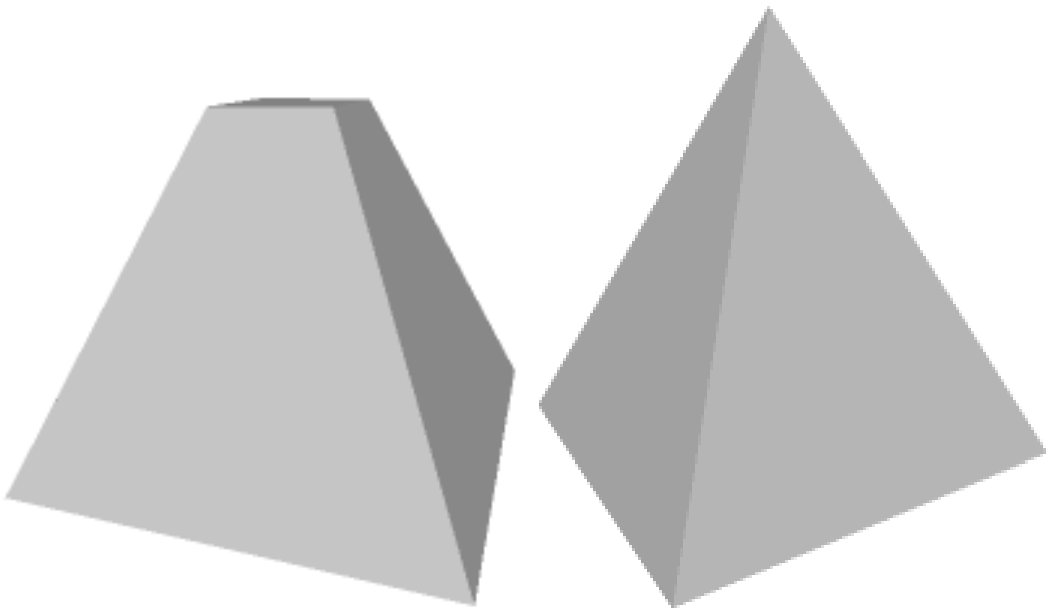


Figura 4.4 Sweep Translacional Cônico.

Se posicionarmos o ponto de fuga negativamente, será gerado um *sweep* denominado translacional cônico divergente [CAS 91]. Um exemplo de objeto gerado pela técnica de *sweep* translacional cônico divergente é mostrado na figura 4.5.

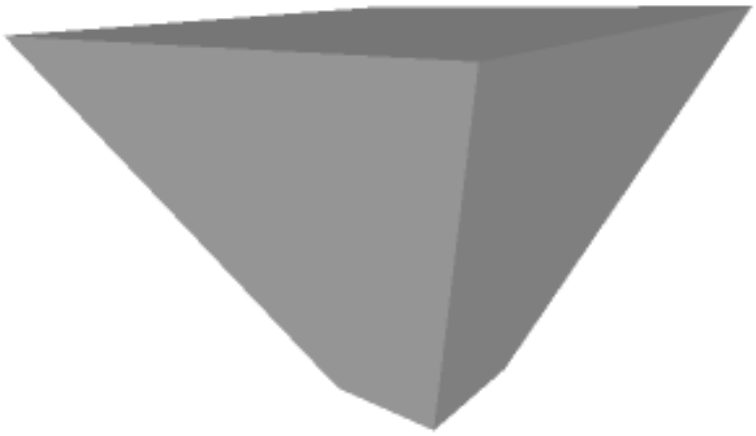


Figura 4.5 Sweep Translacional Cônico Divergente.

4.1.3 Sweep Translacional com Torção

A medida que o polígono gerador é transladado ele sofre também uma rotação de sua base geradora. Para executar este *sweep* é necessário fornecer a altura total da translação, o incremento de torção e o incremento de translação. A cada incremento na translação do polígono gerador é executado uma rotação no polígono, essa operação se repete até atingir a altura total de translação. O incremento de torção é dado em graus. Para cada incremento na translação o número total de vértices cresce em N unidades [CAS 91]. Este tipo de *sweep* pode ser visto na figura 4.6.

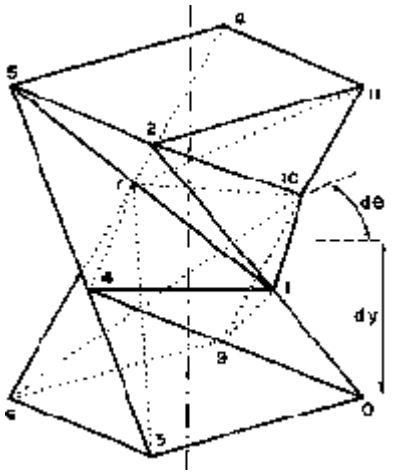


Figura 4.6 Sweep Translacional com Torção.
Incremento de torção (dθ) e incremento de translação (dy).

4.2 Sweep Rotacional

O *sweep* rotacional pode ser definido como a rotação de uma área sobre um determinado eixo. Afonso Hermida classifica esta técnica pelo nome de *Superfície Torneada* [HER 94] e Jonas Gomes por *Superfície de Revolução* [GOM 98]. Essa técnica permite construir uma superfície com simetria coaxial, a partir de uma curva ou superfície. Considerando uma reta *r* no espaço, e uma superfície *s* contida no plano que passa por *r*, obtemos pela rotação de *s* um sólido de revolução (ver figura 4.7). Diversos sólidos conhecidos são gerados com essa técnica, tais como esferas, cilindro, cone, etc.

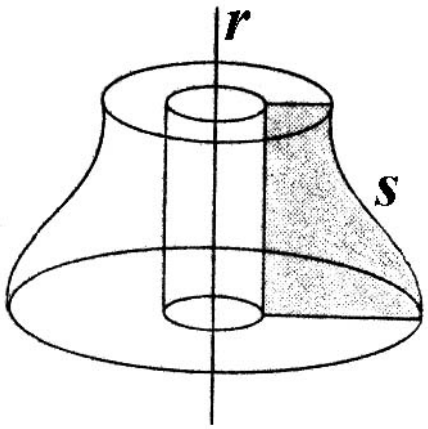


Figura 4.7 Sweep Rotacional.

Este tipo de *sweep* pode ser classificado tanto pelo polígono gerador, que pode ser fechado ou aberto, quanto pela rotação que pode ser completa ou parcial, tendo então os seguintes tipos:

4.2.1 Sweep Rotacional Completo de Polígono Fechado

Um polígono fechado nada mais é que uma superfície. Neste caso, ela é rotacionada sobre um eixo do mesmo plano desta superfície. O ângulo total de revolução é sempre de 360 graus. Nesse tipo de *sweep* pode-se parametrizar o incremento angular de revolução ou o número de fatias ou setores (a razão entre o ângulo total de revolução pelo incremento angular).

Nesse tipo de *sweep* não é recomendado definir vértices sobre o eixo de rotação pois o algoritmo causará duplicidade do vértice que fisicamente deveria ser único (no caso de se ter um vértice sobre o eixo e for feita uma rotação com 18 fatias, serão gerados 18 vértices iguais, gerando replicação de dados). Também não devem ser criados polígonos que sejam cortados pelo eixo de rotação. A figura 4.8 mostra um objeto gerado pela técnica de *sweep* rotacional completo, com seis fatias gerada pela rotação de um retângulo.

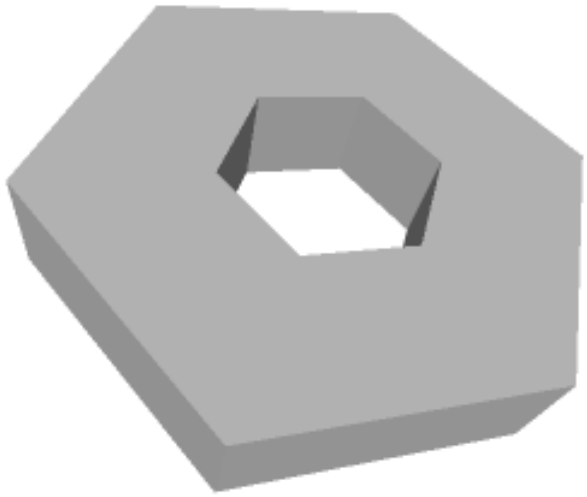


Figura 4.8 Sweep Rotacional Completo de Polígono Fechado.

O número de faces do objeto gerado é definido pelo número de arestas da superfície geradora multiplicada pelo número de fatias. No caso da figura acima, como o polígono gerador tem quatro arestas (retângulo) e rotacionado em seis fatias, o total de faces desta figura é vinte e quatro.

4.2.2 Sweep Rotacional Completo de Polígono Aberto

Neste tipo de *sweep*, ao invés do polígono gerador ser uma superfície fechada, geralmente são usadas curvas ou linhas. Os parâmetros para esse tipo de *sweep* são idênticos ao item anterior, já que ambos são rotacionais completos.

Como foi visto anteriormente, se forem definidos vértices sobre o eixo de rotação no *sweep* rotacional completo de polígono fechado, isto irá gerar duplicidade dos vértices.

Neste caso, a melhor técnica seria a rotacional completa de polígono aberto. Normalmente o primeiro e o último vértice estão sobre o eixo de rotação. Mas neste tipo de *sweep*, também não se deve criar polígonos que cortem o eixo de rotação.

A figura 4.9 mostra uma figura gerada pela técnica de *sweep* rotacional completa, com quatro fatias feitas simplesmente pela rotação de duas linhas.

Figura 4.9 Sweep Rotacional Completo de Polígono Aberto.

O número de faces do objeto gerado é definido da mesma maneira que o *sweep* rotacional completo de polígono fechado, diferindo apenas na forma do polígono gerador (aberto ou fechado). No caso da figura acima, como o polígono gerador tem duas arestas e rotacionada em quatro fatias, o total de faces desta figura é oito.

4.2.3 Sweep Rotacional Parcial de Polígono Fechado

Esse tipo de *sweep* é semelhante ao rotacional completo de polígono fechado, mas difere dele pelo fato deste não fazer uma volta completa em torno do eixo de rotação. Neste tipo de *sweep*, além dos parâmetros citados nos itens anteriores, também passa a ser definido como parâmetro o ângulo total de revolução (entre zero e 360 graus). Se for definido um ângulo igual a 360 ele passa a ser rotacional completo.

Como no tipo rotacional completo de polígono fechado, nesse tipo de *sweep* não é recomendado definir vértices sobre o eixo de rotação, tampouco ser criado polígono que seja cortado pelo eixo de rotação. A figura 4.10 mostra um objeto gerado pela técnica de *sweep* rotacional completa de polígono fechado, com seis fatias geradas pela rotação de um retângulo (mesma superfície geradora da figura 4.8) e 270° de ângulo de revolução.

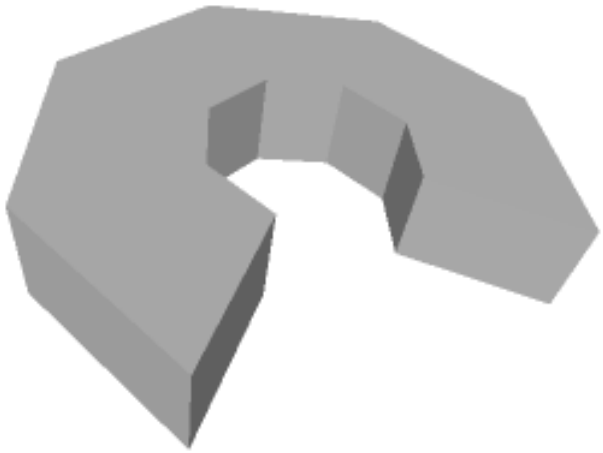


Figura 4.10 Sweep Rotacional Parcial de Polígono Fechado.

Neste caso, haverá duas novas faces no objeto, semelhantes ao polígono gerador e formadas no início e no final da revolução. Sendo assim, o número de faces do objeto gerado é definida pelo número de arestas da superfície geradora multiplicada pelo número de fatias mais dois. No caso da figura acima, como o polígono gerador tem quatro arestas (retângulo) e foi rotacionado em seis fatias, o total de faces desta figura é vinte e seis (vinte e quatro faces laterais mais face inicial e final).

4.2.4 Sweep Rotacional Parcial de Polígono Aberto

As mesmas semelhanças entre o *sweep* completo e parcial de polígonos fechados ocorre entre este tipo de *sweep* com o rotacional completo de polígono aberto. A diferença está somente nos parâmetros (passa a ser definido também como parâmetro o ângulo total de revolução) e número de faces. Se for definido um ângulo igual a 360 ele passa a ser do tipo rotacional completo. Como no tipo rotacional completo de polígono aberto, nesse tipo de *sweep* não se deve criar polígonos que sejam cortados pelo eixo de rotação.

A figura 4.11 mostra uma figura gerada pela técnica de *sweep* rotacional parcial de polígono aberto, com quatro fatias feita pela rotação de três arestas e 270° de ângulo de revolução.

Figura 4.11 Sweep Rotacional Parcial de Polígono Aberto.

Neste caso também haverá duas novas faces no objeto. Estas faces são semelhantes ao polígono gerador e formadas no início e no final da revolução. No caso da figura acima, como o polígono gerador tem três arestas e rotacionada em quatro fatias, o total de faces desta figura é quatorze (doze faces mais as faces inicial e final).

4.3 Sweep Helicoidal

O *sweep* helicoidal é uma combinação dos tipos de *sweep* rotacional parcial de polígono fechado e translacional. Além de incremento de revolução também é feito um incremento de translação no mesmo sentido do eixo de revolução. Neste caso, o ângulo de revolução não está limitado a 360 graus, podendo executar várias voltas. Quando o ângulo de revolução for maior que 360 graus, o incremento de translação deve ser suficientemente grande para que não haja interferência quando completar a volta. Neste caso, o número de fatias por volta (ou setores) multiplicado pelo incremento de translação deve ser no mínimo igual a altura máxima do polígono gerador [CAS 91]. Como no *sweep* helicoidal só é considerado o polígono fechado, devemos tomar as mesmas precauções vistas

anteriormente.

A figura 4.12 mostra uma figura gerada pela técnica de *sweep* helicoidal, com dez fatias gerada pela rotação de um retângulo (mesma superfície geradora das figura 4.8 e 4.10) e 540° de ângulo de revolução.

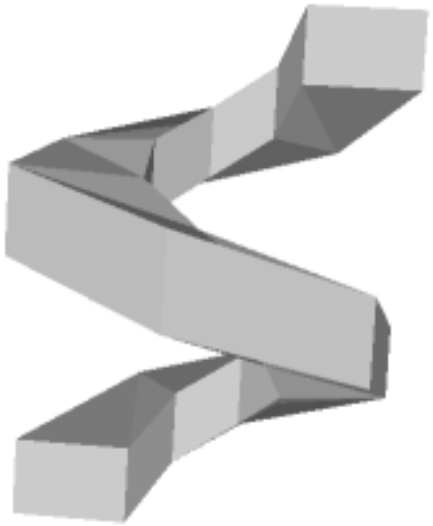


Figura 4.12 Sweep Helicoidal.

O número de faces geradas é igual à técnica de *sweep* rotacional parcial. Sendo assim, o número de faces do objeto gerado é definido pelo número de arestas da superfície geradora multiplicada pelo número de fatias mais dois. No caso da figura acima, como o polígono gerador tem quatro arestas (retângulo) e este foi rotacionado em dez fatias, o total de faces desta figura é quarenta e dois (quarenta faces laterais mais as faces inicial e final).

4.4 Sweep Geral

Esta técnica generaliza as técnicas de *sweep* vistas anteriormente. No *sweep* rotacional, uma área é girada sobre um determinado eixo. Já no *sweep* translacional, parte-se de uma área que é deslocada por uma trajetória normal ao plano desta para gerar um sólido.

No *sweep* geral, é obtido um sólido deslocando-se uma superfície (α) ao longo de

uma curva (γ). O resultado é um sólido que contém a curva γ como eixo. Jonas Gomes denomina esta técnica como *superfície tubular de γ* [GOM 98]. A curva γ é chamada de curva guia e α de seção. A figura 4.13 mostra uma representação deste tipo de *sweep*.

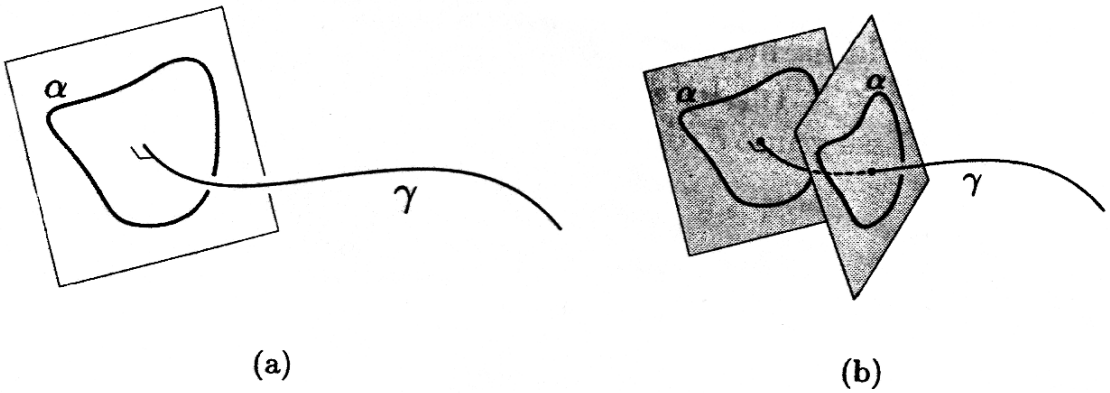


Figura 4.13 Sweep Geral.
(a) uma superfície e sua curva guia
(b) superfície original mais outra depois
de realizado um deslocamento sobre γ.

Conforme visto anteriormente, a técnica de *sweep* translacional é um caso particular onde γ é uma reta. A técnica de *sweep* rotacional é o caso particular em que γ é um círculo.

4.5 Considerações Finais Sobre a Técnica de Modelagem por Sweep

Neste capítulo foi estudada a técnica de *sweep* bem como seus tipos principais, já que esta foi a técnica escolhida para realizar a implementação deste trabalho.

Os algoritmos e alguns detalhes relativos a esta técnica serão abordados no próximo capítulo, onde será descrita a integração do sistema de modelagem por *sweep* com a VRML. Nesse capítulo será abordada a implementação de um sistema de modelagem por *sweep* para gerar objetos VRML, que foi desenvolvido juntamente a este trabalho.

5. IMPLEMENTAÇÃO

Este trabalho tem por objetivo a integração de técnicas de modelagem com a VRML, através da implementação prática de um software. Sendo assim, foi escolhida a técnica de *sweep* para fazer esta integração. Após a escolha da técnica de modelagem, o passo seguinte foi a escolha da interface. Foi decidida a criação de um sistema de modelagem com uma interface simples de usar, onde o usuário desenha seus próprios polígonos para gerar os objetos pela técnica de modelagem por *sweep*, com a possibilidade de ler e gravar arquivos contendo descrições de polígonos.

A linguagem de programação escolhida para realizar a implementação de um sistema de modelagem por *sweep* que gere objetos VRML, foi a linguagem Java da Sun Microsystems[1]. Como a VRML está voltada principalmente para a Internet, o ideal para realizar esta integração seria um sistema que funcionasse também pela Internet, podendo ser transmitido junto com as páginas Web e sendo independente de plataforma. Foi por este motivo a escolha da linguagem Java para implementar o sistema de modelagem.

Existem atualmente vários visualizadores para VRML. Grande parte destas ferramentas são de uso livre e estão disponíveis através da Internet, como os *plug-ins*

[1] http://www.sun.com

usados para testar os modelos gerados pelo sistema de modelagem desenvolvido junto a este trabalho (Cosmo Player[2] da SGI, Microsoft VRML Viewer[3], WorldView[4] da Intervista e Viscape[5] VRML da Superscape).

Resolveu-se então que o ideal para este sistema é funcionar no *browser* juntamente com um visualizador VRML. Assim, não seria necessário ao usuário utilizar dois sistemas independentes: um para gerar os modelos e outro para visualizá-los. Mas, para ser possível gravar os objetos gerados, é necessário que se execute o sistema de modelagem também como aplicação independente, já que pelo *browser* não é possível o acesso ao disco.

Assim, decidiu-se a criação de uma ferramenta que seja tanto uma aplicação independente quanto uma que seja executada a partir do *browser*. Com a linguagem Java, isto é possível de se fazer, sendo também por este motivo a melhor linguagem para implementação do sistema de modelagem. Quando o sistema é executado pelo *browser*, é denominado de *applet*. Quando é executado por um interpretador Java fora do *browser*, é chamado de aplicação.

Após ser decidida a forma de como o sistema iria gerar os modelos, o ambiente e a linguagem, o passo seguinte foi a definição da interface, que será objeto de estudo dos próximos itens.

O sistema de modelagem implementado junto a este trabalho recebeu o nome de ATSWorlds. A seguir são descritas e mostradas as duas interfaces deste sistema: executado independentemente (aplicação) e em um *browser* (*applet*). Depois, é feito uma descrição da integração do sistema de modelagem com a VRML. E, por último, são detalhados os tipos de modelagem por *sweep* implementados neste trabalho.

---

[2] http://www.sgi.com
[3] http://www.microsoft.com
[4] http://www.intervista.com
[5] http://www.superscape.com

5.1 As Interfaces

O sistema implementado junto a este trabalho gera objetos VRML a partir da técnica de modelagem por *sweep*. Como foi visto anteriormente, este sistema de modelagem é tanto uma aplicação quanto *applet*. Por isto, serão mostradas ambas as interfaces do sistema de modelagem e explicado o funcionamento do sistema.

5.1.1 Aplicação

Para que se possa salvar os objetos que são gerados pelo sistema de modelagem, é necessário utilizar o ATSWorlds como aplicação. Neste modo, também é possível importar e exportar em arquivo os polígonos geradores do *sweep* em um formato próprio. A figura a seguir mostra a interface do programa sendo utilizado como aplicação:

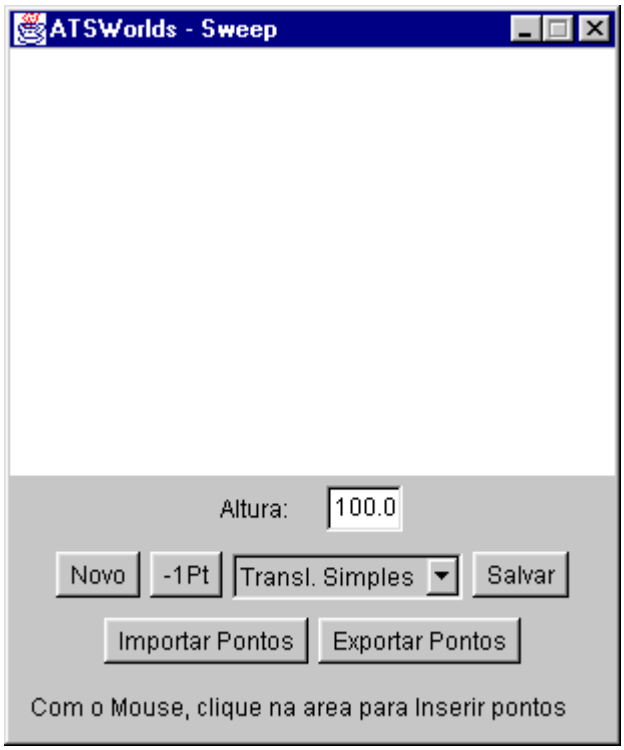


Figura 5.1 ATSWorlds como aplicação.

A área em branco é o local onde são desenhados os polígonos que gerarão os objetos pela técnica de *sweep*. Para desenhar um polígono, basta clicar com o mouse em

pontos desta área, marcando os pontos que definem os vértices deste.

Quando esta área estiver em branco, ao clicar o mouse sobre ela, a posição inicial do polígono será marcada. E, arrastando o mouse com o botão pressionado, resultará na criação de uma aresta para o polígono. Para inserir um novo ponto, gerando uma nova aresta ao polígono gerador para o *sweep*, basta clicar com o mouse e mover até a localização desejada. As coordenadas são mostradas na barra de status na medida em que o mouse é arrastado.

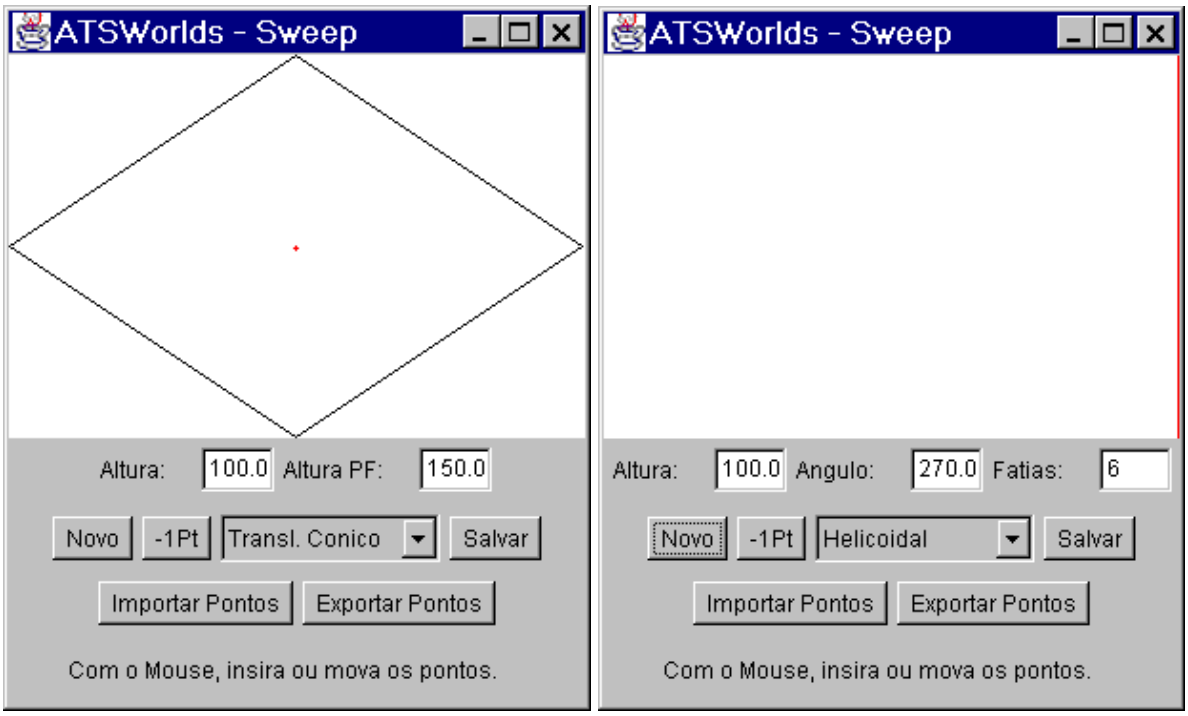
Para mover um ponto, desloca-se o ponteiro do mouse até o ponto desejado. Quando o ponteiro estiver em cima do ponto, o ATSWorlds mostrará uma marca azul neste ponto. Para movê-lo, deve-se clicar em cima deste e arrastá-lo para a localização desejada. As coordenadas são mostradas na barra de status à medida em que o mouse é arrastado. Por isso, para criar um ponto próximo a outro ou fechar o polígono, deve-se clicar com o mouse um pouco longe dos pontos e arrastar até o local desejado.

Para fechar o polígono, portanto, clica-se com o mouse não muito próximo do ponto inicial e em seguida arrasta-se o ponteiro do mouse até próximo ao início do polígono. Quando o ponteiro do mouse chegar próximo do ponto inicial, o polígono fechará.

Se for escolhido o tipo de *sweep* translacional cônico, será apresentado na área de desenho um ponto representado por uma pequena cruz vermelha, indicando as coordenadas do ponto de fuga. Este ponto pode ser movido da mesma maneira que os outros pontos, clicando nele e arrastando-o até a posição desejada.

Se forem escolhidos *sweeps* do tipo rotacional completo, rotacional parcial ou helicoidal, na parte direita da área de desenho será exibida uma linha vermelha que representa o eixo ao redor do qual será feita a rotação destes tipos de *sweep*.

A figura a seguir mostra a interface da aplicação onde estão selecionados os tipos de *sweep* translacional cônico (a), onde pode-se ver o ponto de fuga, e rotacional completo (b), onde pode-se ver o eixo de rotação.



(a) (b)

Figura 5.2 Ponto de fuga e eixo de rotação.
(a) ponto de fuga e (b) eixo de rotação.

A parte inferior da aplicação é denominada barra de status. Ela é usada para dar mensagens do tipo: “Clique com o mouse para mover o ponto” (quando o ponteiro do mouse está em cima de um ponto). Também é usada para dar mensagens de erro ou para mostrar as coordenadas do ponto que está sendo movido.

As opções logo abaixo da área de desenho, dependem do tipo de *sweep* que se quer realizar, portanto são variáveis. Estas opções serão estudadas no item em que serão definidos os algoritmos de cada tipo de modelagem por *sweep* implementados neste trabalho.

No caso de ser inserida uma aresta inconvenientemente, pressionando-se o botão **-1Pt** o ATSWorlds removerá o último ponto do polígono, removendo assim a última aresta inserida (“*undo*” da inserção do último ponto).

Para iniciar um novo polígono, descartando o polígono existente na área desenho,

deve-se pressionar o botão **Novo**. Pressionando-se o botão **Exportar Pontos** será gravado o estado desta área de desenho, ou seja, os pontos formadores do polígono e a localização do ponto de fuga. Para ler um estado gravado anteriormente, ou ler pontos gerados por outra aplicação, usa-se o botão **Importar Pontos**. Isto é útil tanto para que se possa gravar e ler um polígono feito, sendo possível reaproveitá-los, quanto para que se possa partilhar esses polígonos com outras aplicações. A figura a seguir mostra o arquivo gerado pela exportação do polígono desenhado na figura 5.2a.

```
ATSWorlds
Arquivo gerado por ATSWorlds

#PONTOS
0   100
150 0
300 100
150 200
0   100

#PONTO_FUGA
150 100

#FIM
```

Figura 5.3 Arquivo de polígonos do ATSWorlds.
Gerado pela exportação dos pontos do polígono desenhado na figura 5.2a.

A primeira linha contém o cabeçalho do arquivo que sempre inicia com uma linha contendo a palavra “ATSWorlds”. Tudo que está entre esta linha e a diretiva “#PONTOS” são comentários que podem ser colocados sobre o polígono. Após esta diretiva, são listados os pontos do polígono. Para indicar que um polígono é fechado, o primeiro e último pontos se repetem. A diretiva “#PONTO_FUGA” marca o fim da lista de pontos e o início das coordenadas do ponto de fuga. A diretiva “#FIM” indica o final do arquivo. Também pode-se colocar comentários e observações ao final do arquivo.

Depois de desenhado o polígono, escolhido o tipo de *sweep* e ajustados os parâmetros necessários, basta clicar no botão **Salvar** para que o ATSWorlds grave um objeto VRML a partir desse polígono pela técnica de modelagem por *sweep* escolhida. Para visualizar o objeto gerado, deve-se ler o arquivo gravado pelo sistema de modelagem em um *browser* com um *plug-in* VRML instalado, ou ler o arquivo em algum outro

visualizador capaz de ler arquivos VRML 2.0 ou superior.

5.1.2 Applet

O sistema quando usado como *applet*, em uma página HTML denominada ATSWorlds.htm, permite que se visualize tanto o sistema de modelagem quanto o objeto gerado na mesma tela, sem a necessidade de ficar mudando de aplicativo cada vez que um novo objeto é criado ou alguma alteração é realizada. Neste modo, os objetos não podem ser gravados, já que o *browser* não permite o acesso a disco. Este modo é útil para se realizar testes, criando objetos e visualizando-os antes de partir para o objeto final, sem precisar ficar alternando a todo o instante entre aplicação e *browser*.

O ATSWorlds neste modo é bastante útil também para alunos que queiram examinar na prática a técnica de modelagem por *sweep* em um sistema bastante simples, podendo inclusive, executá-lo através da Internet. Tornando também este sistema em uma ferramenta didática para o ensino da área de computação gráfica, geometria (sólidos geométricos feitos por superfícies de revolução), entre outros.

A interface da *applet* e da aplicação são muito semelhantes, devido ao fato de serem ambos o mesmo sistema (mesmo código). O próprio sistema se encarrega de fazer a modificação da interface dependendo do fato de ser *applet* ou aplicação.

Isto faz com que a construção dos objetos seja feita de forma muito semelhante ao item anterior, já que é o mesmo sistema. A elaboração dos polígonos com o ATSWorlds executado em uma página é realizada da mesma maneira que na aplicação, ou seja, a forma de inserir pontos, movê-los e fechar o polígono, é idêntica.

A diferença básica é que no lugar do botão “Salvar”, na *applet* existe o botão **Atualizar**. Este botão faz com que o objeto VRML seja gerado e atualizado no *browser*, em vez de salvar o objeto. Também na *applet* não há os botões para importar e exportar o polígono, pelo fato do *browser* não permitir acesso a disco.

Outra mudança que o sistema faz na interface é abaixo da barra de status, onde ele mostra o nome e versão do sistema, nome do autor e do professor orientador. São algumas informações básicas, caso alguém deseje maiores informações.

Além do sistema de modelagem e do visualizador VRML, a página também contém uma parte contendo um texto explicativo sobre o sistema de modelagem, de como utilizar o ATSWorlds e alguns requisitos necessários à execução da página.

A figura 5.4 mostra a interface do sistema de modelagem sendo executado como *applet* em uma página HTML. O ATSWorlds ocupa a parte esquerda da página, o visualizador na parte superior direita e na parte inferior direita o texto explicativo (neste exemplo foi usado o visualizador Cosmo Player da Silicon Graphics Inc. e o *browser* Netscape).

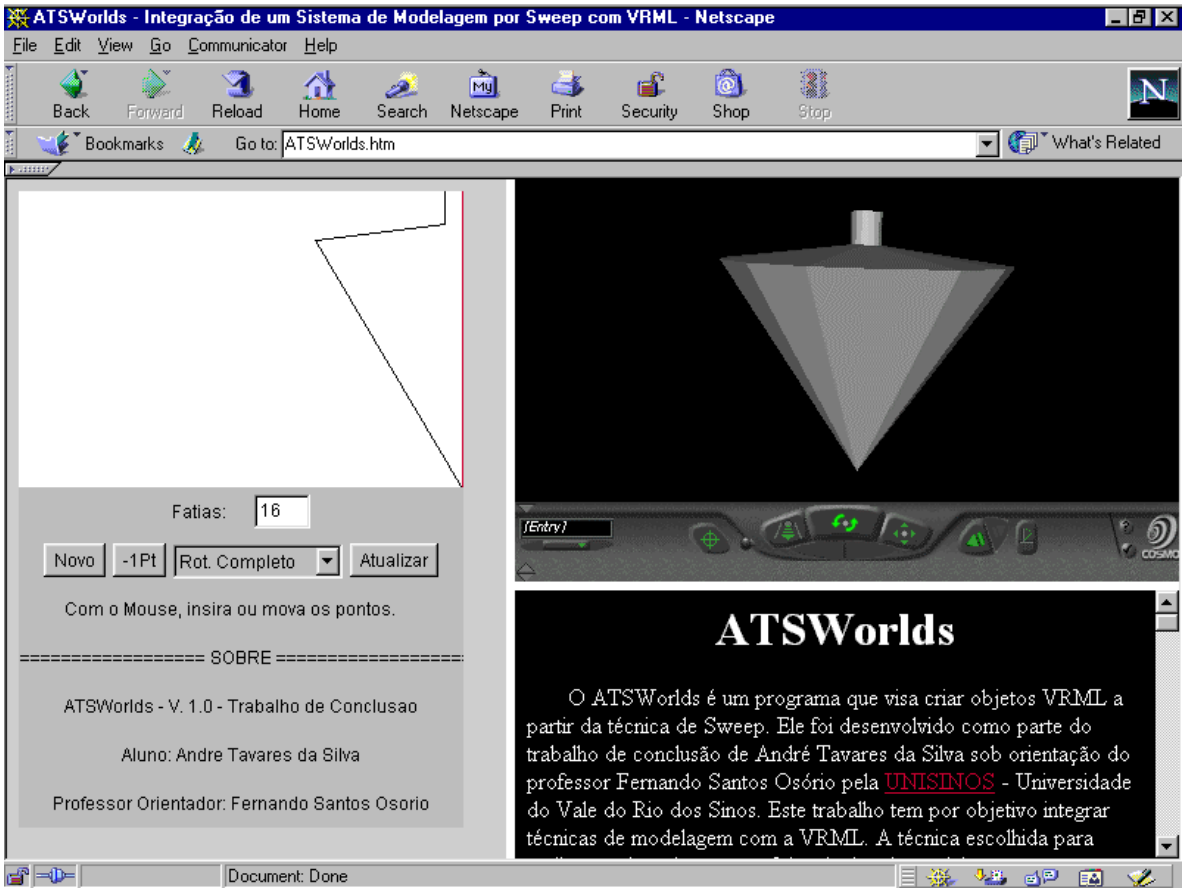


Figura 5.4 ATSWorlds em uma página HTML.

Depois de desenhado o polígono, escolhido o tipo de *sweep* e ajustados os parâmetros necessários, basta clicar no botão **Atualizar** para que o ATSWorlds atualize o objeto na parte superior direita da página. Este botão deve ser usado toda vez que for alterado o polígono gerador, o tipo de *sweep*, ou os parâmetros e se queira visualizar o objeto a partir das mudanças feitas. Neste modo não é preciso ficar alternando entre programas, a alteração dos dados e a visualização dos resultados é automática.

A seguir, será dada uma descrição de como é feita a comunicação entre o ATSWorlds e a página para a atualização do objeto no *browser*.

5.2 Comunicação entre Plug-in Java e JavaScript

A grande maioria dos *browsers* padrões tem suporte a Java, ou seja, quando uma página contém um programa Java (*applet*) eles são capazes de executar estes programas. Eles não são executados diretamente pelo *browser*, mas independente deste, em uma máquina virtual (JVM – *Java Virtual Machine*), não havendo comunicação entre ambos.

Normalmente os *browsers* também tem suporte a JavaScript. É uma linguagem de macros desenvolvida pela Netscape que permite ao *browser* uma maior versatilidade. Ao contrário de um programa Java, estas macros são executadas diretamente pelo *browser*, podendo inclusive acessar dados do mesmo e da própria página.

Como o sistema foi desenvolvido como *applet* Java, era necessário que o próprio sistema tivesse a capacidade de se comunicar com o *browser*, a fim de enviar o objeto VRML gerado pela técnica de modelagem por *sweep* para ser visualizado na mesma página.

Este recurso está disponível somente a partir da versão 1.2.2 do Java, através de uma classe denominada JSObject desenvolvida pela Netscape. Esta classe permite que *applets* possam acessar funções JavaScript de uma página HTML. Assim, esta classe foi utilizada para que o ATSWorlds possa atualizar o *frame* no qual será visualizado o objeto gerado pelo sistema.

A página “ATSWorlds.htm” é dividida em três *frames*: o sistema de modelagem, o objeto gerado e a página de texto. A página com o sistema de modelagem contém além da *applet*, uma função JavaScript que recebe como parâmetro o objeto gerado pelo sistema de modelagem e atualiza este objeto no *frame* adequado para ser visualizado. O ATSWorlds gera o objeto em memória e passa este para a função JavaScript através da classe JSObject.

Como alguns *browsers* não tem a versão mais atualizada do Java, o ATSWorlds deve ser executado por um *plug-in* Java versão 1.2.2 ou superior. Para isto, deve-se instalar um JDK (Java Development Kit) ou um JRE (Java Runtime Environment) atualizados.

No próximo item, será estudado de que forma o objeto VRML é gerado a partir do ATSWorlds, fazendo um estudo da integração entre o sistema de modelagem e a VRML. Depois, serão detalhados os tipos de modelagem por *sweep* implementados neste trabalho.

5.3 Integração entre o Sistema de Modelagem e a VRML

Foi visto no capítulo sobre realidade virtual que na VRML existem primitivas avançadas para a criação de objetos. O melhor método para definir um objeto gerado pelo sistema de modelagem é definir este objeto por suas faces limitantes. Na VRML, o nodo tipo *IndexedFaceSet* define um conjunto de faces no sistema de coordenadas. A figura a seguir mostra um exemplo de um nodo deste tipo.

```
geometry IndexedFaceSet {
  coord Coordinate {
    point [-1 -1 0, 1 -1 0, 1 1 0, -1 1 0, 0 0 1 ]
  }
  coordIndex [0 1 4 -1, 1 2 4 -1, 2 3 4 -1, 3 0 4]
  color Color { color [1 0 0, 0 1 0, 0 0 1]}
  colorIndex [1 2 0 2  ]
  colorPerVertex FALSE
  creaseAngle 0
  solid TRUE
  ccw TRUE
  convex TRUE
}
```

Figura 5.5 Exemplo de um nodo do tipo IndexedFaceSet.

O campo *coord* especifica um nodo do tipo *Coordinate* que é uma lista de coordenadas 3D separadas por espaços ou vírgulas.

O campo *coordIndex* contem uma lista de índices das coordenadas que definem as faces a serem desenhadas. A contagem dos índices inicia-se em zero, ou seja, o primeiro ponto da lista é o de índice 0 (zero). O número –1 é usado como um separador para as faces. Ele indica quando a face termina e começa a próxima.

Como as faces são sempre definidas por polígonos fechados, não é necessário definir o primeiro ponto duas vezes. No exemplo acima, a primeira face é delimitada pelos pontos 0, 1 e 4. Isto significa que o primeiro ponto se unirá com o segundo, então o segundo se juntará com o quinto, e, finalmente o quinto e o primeiro serão unidos para fechar esta região que define uma das faces do objeto.

O campo *color* define um nodo *Color*, que representa uma lista de cores a serem aplicadas às faces. O campo *color* é opcional. O valor padrão a ser aplicado neste campo não é especificado. O campo *colorIndex* serve ao mesmo propósito de *coordIndex*, mas referente a cores. Se *colorIndex* não é especificado, então *coordIndex* é usado em seu lugar. O *colorPerVertex* é um campo booleano que determina de que forma que as cores são aplicadas.

O campo *creaseAngle*, afeta a forma em que é definida a iluminação entre faces adjacentes. Se o ângulo entre os vetores normais a duas faces adjacentes é menor que *creaseAngle*, a união destas faces terá uma mudança suave de iluminação, senão as faces parecerão facetadas. O valor de *creaseAngle* deverá ser maior ou igual a zero.

O campo *solid* determina se o *browser* deve desenhar ambos os lados de uma face ou apenas a parte frontal. O valor padrão para *solid* é TRUE. Neste caso, não é necessário desenhar a parte posterior de cada face, otimizando o processamento. Se o valor de *solid* estiver FALSE então o *browser* desenhará ambos os lados de cada face.

O campo *ccw* indica a ordem em que são definidos os pontos das faces. Se o valor

de *ccw* for TRUE, o sentido é anti-horário (regra da mão direita), senão o sentido é horário. Uma face tem sempre dois lados e, em alguns casos, é importante definir qual é a face frontal. O valor padrão para *ccw* é TRUE.

O campo *convex* indica quando todas as faces do objeto definidas em *coordIndex* são convexas. Um polígono é convexo se é planar, não se intersecciona e todos os ângulos interiores são menores de 180 graus. Polígonos não planares ou que se interseccionam podem produzir resultados inesperados. O valor padrão para *convex* é TRUE (é convexo). Portanto, caso não se tenha certeza que todas as faces do objeto são convexas, deve-se colocar o valor FALSE para *convex*. Isto fará com que o *browser* divida a face em outras pequenas faces convexas. Mas, no caso de se ter certeza que o objeto é convexo, o ideal é deixar este valor como TRUE, dizendo que o *browser* não deve se preocupar fazendo esta divisão das faces, economizando tempo de processamento.

Para o sistema de modelagem ATSWorlds formar os objetos, este deve gerar os pontos e as faces deste objeto através da técnica de modelagem. O sistema constrói os objetos a partir dos pontos do polígono desenhado, gerando um conjunto mínimo de pontos e de faces do objeto, a fim de aumentar a performance na visualização dos objetos formados. Os mundos VRML atualmente ainda são bastante pesados para os sistemas existentes.

Como o sistema de modelagem não tem a capacidade de saber se todas as faces dos objetos gerados pelo usuário são convexas, o sistema monta os objetos e atribui o valor FALSE para o campo *convex*. Isto garante que todos os objetos serão visualizados corretamente. Ao se gravar os objetos e tiver a certeza de que todas as faces deste objeto forem convexas, basta remover este campo ou atribuir-lhe o valor TRUE, para melhorar a performance da visualização. Mas se o objeto tiver alguma face não convexa (uma engrenagem por exemplo), não se deve alterar este valor.

Para permitir ao usuário uma maior liberdade, foi atribuído o valor FALSE para o campo *solid*. Isto faz com que o *browser* desenhe ambos os lados de cada face do objeto,

garantindo que o objeto seja visualizado corretamente, independente da orientação que o usuário insere os pontos (sentido horário ou anti-horário). Se forem inseridos todos os pontos seguindo o sentido horário, pode-se remover este campo ou atribuir-lhe valor TRUE para remover as faces ocultas dos objetos fechados (as faces que não necessitam serem calculadas e exibidas), a fim de aumentar a performance da visualização dos objetos. Caso todos os pontos forem inseridos seguindo o sentido anti-horário, pode-se remover o campo *solid* e inserir um campo *ccw FALSE*, fazendo com que as faces sejam desenhadas seguindo a ordem inversa.

A seguir é mostrado um exemplo bastante simples de um objeto gerado pelo sistema. A figura 5.6 exibe a interface do sistema ATSWorlds com uma aresta desenhada que irá gerar um tronco de uma pirâmide e seus parâmetros. A Figura 5.7 mostra o arquivo VRML gerado pelo sistema.

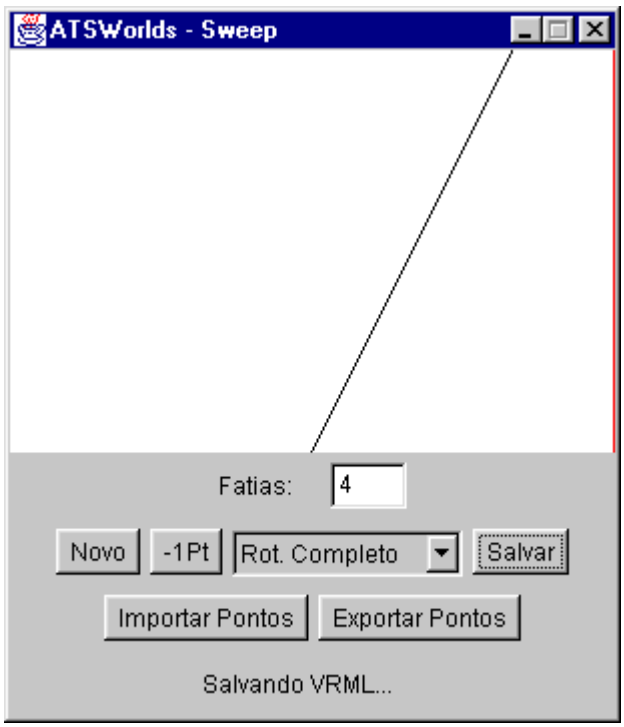


Figura 5.6 ATSWorlds gerando um objeto.

```
#VRML V2.0 utf8
WorldInfo {
  info [ "Created in ATSWorlds",
         "by Andre Tavares da Silva",
         "TC: Integracao de tecnicas de modelagem com VRML",
         "UNISINOS – Universidade do Vale do Rio dos Sinos" ]
}
Transform {
  Children [
    Shape {
        Appearance Appearance { material Material { }}
        Geometry   IndexedFaceSet {
        Coord Coordinate { point [
           1.0    0.0    0.0,
           0.0    0.0    1.0,
           -1.0   0.0    0.0,
           0.0    0.0    -1.0,
           0.33   1.33   0.0,
           0.0    1.33   0.33,
           -0.33  1.33   0.0,
           0.0    1.33   -0.33 ]
          }
          coordIndex [
           0, 4, 5, 1, -1,
           1, 5, 6, 2, -1,
           2, 6, 7, 3, -1,
           3, 7, 4, 0, -1,
           0, 1, 2, 3, -1,
           7, 6, 5, 4, -1 ]
          color NULL
          creaseAngle 0
       }
    }
  ]
}
```

Figura 5.7 Código VRML gerado pelo ATSWorlds.

As quatro primeiras faces são as faces laterais do objeto e as duas últimas, a face superior e a face inferior do objeto.

Como as faces do objeto são todas convexas e as faces internas não necessitam ser calculadas, foram removidos os campos *solid* e *convex* para melhorar a performance da visualização do objeto. A diferença só é observada quando se tem vários objetos no mundo virtual.

A Figura 5.8 apresenta este objeto gerado pelo ATSWorlds visualizado através de um *browser*.

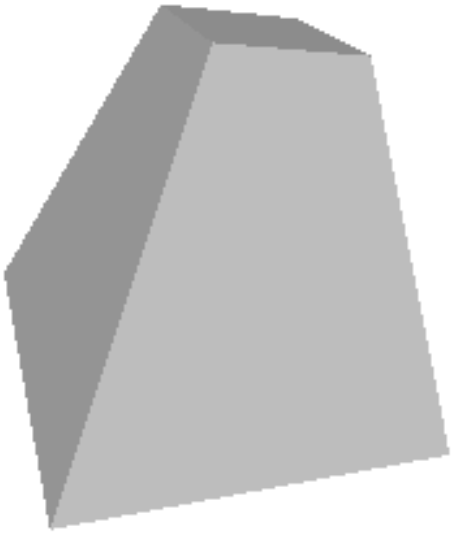


Figura 5.8 Objeto gerado pelo ATSWorlds.

Todos os objetos gerados pelo sistema de modelagem são construídos por suas faces e pontos. No próximo item serão estudados todos os tipos de modelagem por *sweep* implementados no sistema de modelagem ATSWorlds.

5.4 Técnica de Modelagem por Sweep

Os tipos de modelagem por *sweep* implementados no sistema ATSWorlds foram: Translacional Simples, Translacional Cônico, Rotacional Completo, Rotacional Parcial e Helicoidal. Com exceção do *sweep* Helicoidal, em que o polígono gerador deve ser fechado, em todos os outros tipos de *sweep* os polígonos geradores podem ser abertos ou fechados, permitindo uma maior flexibilidade.

A unidade de medida na VRML é o metro. Sendo a resolução da área de desenho do sistema de modelagem 300 pontos por 200, foi resolvido que, num primeiro momento, o sistema faria uma proporção de 150 pontos por cada metro.

A seguir serão estudados os tipos de *sweep* implementados no sistema de modelagem e os algoritmos usados para gerar os pontos e faces dos objetos.

5.4.1 Sweep Translacional Simples

Como foi visto no capítulo sobre a técnica de modelagem por *sweep*, o tipo mais simples é o *sweep* translacional, no qual cada aresta desenhada é deslocada por uma trajetória normal ao plano em que estão as arestas.

O sistema de modelagem foi elaborado de forma que, neste tipo de *sweep*, a origem (ponto (0,0,0)) do objeto gerado fique exatamente no centro da área de desenho. Esta área representa o plano XZ do sistema de coordenadas da VRML. Portanto, o deslocamento é feito no eixo Y (na vertical). Este deslocamento é feito em ambas as faces para que a origem fique no centro da vertical do objeto. Assim, se o desenho do objeto estiver centralizado na área de desenho, a origem ficará no centro geométrico do objeto formado.

Tanto a base quanto o topo do objeto formado serão o próprio polígono desenhado pelo usuário. A altura deste objeto é atribuída através do parâmetro “Altura” do ATSWorlds.

O algoritmo usado para a geração deste tipo de *sweep* é bastante simples. Para formar a lista de pontos do polígono (*Coordinate*), converte-se cada ponto do polígono desenhado pelo usuário para dois pontos no espaço da VRML. Ou seja, cada ponto formará um vértice da base e um vértice do topo do objeto.

No sistema de modelagem ATSWorlds, neste tipo de *sweep*, o eixo X do objeto VRML é representado pelo eixo horizontal da tela (Xtela), o eixo Y é dado pelo parâmetro Altura, e o eixo Z do objeto é representado pelo eixo vertical da tela (Ytela). Como foi estipulada uma proporção de 150 pontos por cada metro, cada ponto do polígono desenhado gerará um ponto (X,Y,Z) na base onde: X = (Xtela – 150) / 150, Y = – (Altura / 300) e Z = (Ytela – 100) / 150. E para o topo: X = (Xtela – 150) / 150, Y = Altura / 300 e Z = (Ytela – 100) / 150.

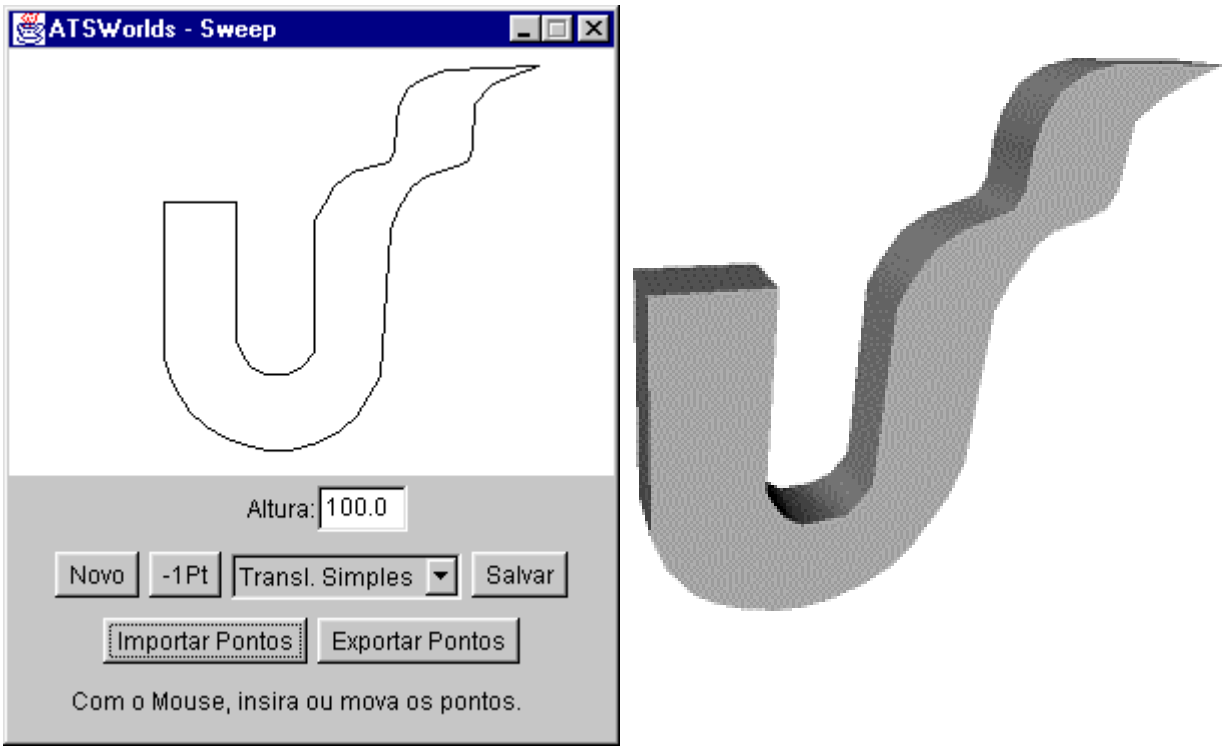
As faces são feitas pela união destes pontos (*coordIndex*). As faces laterais são formadas pela junção dos pontos que formam cada aresta do polígono gerador. Como as

arestas da base e do topo são paralelas, cada face lateral terá um formato retangular. As faces superior (topo) e inferior (base) são formadas pela união dos próprios pontos que as formam.

O ATSWorlds permite que seja feito o *sweep* translacional de um polígono aberto, para dar maior liberdade ao usuário. Neste caso, serão gerados apenas as faces laterais, não fechando o topo e a base do objeto.

O número de pontos do objeto formado será o dobro do número de pontos do polígono gerador. O número de faces será o número de arestas do polígono mais dois (topo e base).

A figura a seguir mostra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* translacional simples e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.9 Sweep Translacional Simples.
(a) polígono gerador e (b) objeto gerado.

Se a orientação da criação das arestas do polígono for no sentido horário, e o polígono gerador for fechado, o sistema formará todas as faces voltadas para seu exterior. Pode-se neste caso remover o campo *solid* para otimizar o processamento na visualização do objeto. As faces laterais são sempre convexas (retângulos). Se o polígono gerador também for convexo, o campo *convex* pode ser removido neste caso.

5.4.2 Sweep Translacional Cônico

Este tipo de *sweep* é uma extensão do translacional simples. Neste caso, somente uma das faces é igual ao polígono gerador desenhado pelo usuário. Os vértices da outra face, ao serem transladados, convergem para um ponto denominado ponto de fuga. Esta face, portanto, sofre uma alteração de escala enquanto é deslocada. O ponto de fuga não necessita ficar no centro do polígono gerador. Isto faz com que este deslocamento possa também ser não perpendicular ao plano da base do objeto gerado.

A base do objeto formado será o próprio polígono desenhado pelo usuário. A altura do objeto é atribuída através do parâmetro “Altura”. O ponto de fuga é representado por uma pequena cruz vermelha na área de desenho (figura 5.2a) e pelo parâmetro “Altura PF”. Isto facilita ao usuário visualizar a forma que terá o objeto.

Neste tipo de *sweep*, o eixo X do objeto VRML é representado pelo eixo horizontal da tela (Xtela), o eixo Y é dado pelo parâmetro Altura, e o eixo Z do objeto é representado pelo eixo vertical da tela (Ytela).

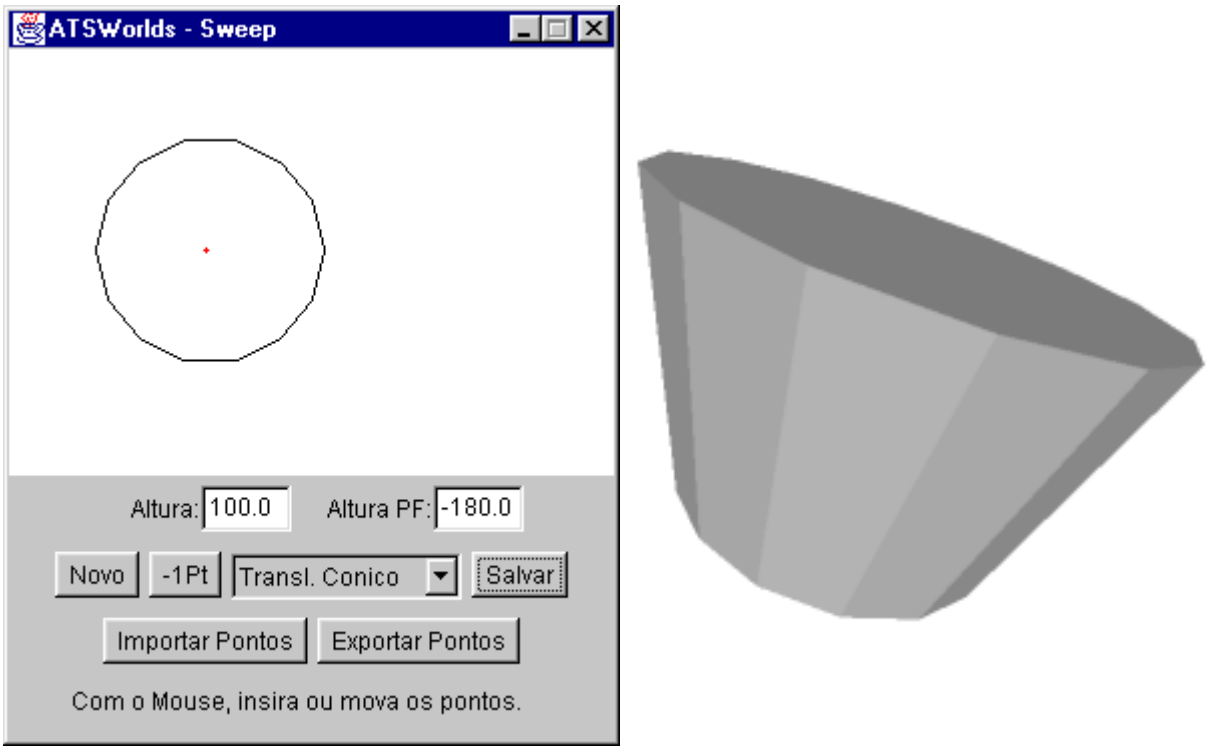
O cálculo da base neste tipo de *sweep* é o mesmo do translacional simples. O que difere ambos, é o cálculo dos pontos que formam o topo do objeto. A localização destes pontos é dada pelo cálculo de proporcionalidade entre os pontos dos vértices do polígono gerador, o ponto de fuga e a altura dada. Neste caso, os pontos do topo do objeto formado são calculados por: X = (Xtela + Altura * (Xponto_fuga – Xtela) / AlturaPF) -150) /150.0, Y = Altura/300 e Z = (Ytela + Altura * (Yponto_fuga – Ytela) / AlturaPF) -100)/150.0.

Se a altura do ponto de fuga for negativa, o *sweep* será do tipo translacional cônico divergente. E a parte menor, neste caso, será voltada para baixo.

As faces, o topo e a base são calculadas da mesma maneira que no *sweep* translacional simples. Neste caso, as faces laterais serão trapézios ou triângulos, sendo também convexas. Portanto, Se o polígono gerador também for convexo, o campo *convex* pode ser removido.

Se a orientação da criação das arestas do polígono for no sentido horário, e o polígono gerador for fechado, o ATSWorlds formará todas as faces voltadas para seu exterior. Pode-se, neste caso, remover o campo *solid* para otimizar o processamento na visualização do objeto.

A figura a seguir ilustra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* translacional cônico e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.10 Sweep Translacional Cônico.
(a) polígono gerador e (b) objeto gerado.

O sistema de modelagem também permite a criação de um objeto pela técnica de *sweep* translacional cônico de polígono aberto, gerando somente as faces laterais deste objeto.

5.4.3 Sweep Rotacional Completo de Polígono Fechado

No *sweep* translacional, a diferença entre os objetos criados por polígonos abertos de polígonos fechados está somente na geração das faces. Ou seja, se o polígono é fechado, gera-se as faces do topo e da base. No caso do *sweep* rotacional, a diferença está também na forma da geração do objeto. Por isso, os tipos de *sweep* rotacionais de polígonos fechados e de polígonos abertos, serão estudados separadamente.

O *sweep* rotacional, como foi visto no item 4.2, faz uma superfície ser rotacionada sobre um eixo do mesmo plano desta superfície. Este eixo, no sistema de modelagem ATSWorlds, é representado por uma linha vermelha localizada na parte direita da área de desenho (ver figuras 5.2b, 5.3 e 5.5). O parâmetro “Fatias” representa o número de setores que terá o objeto. Quanto maior o número de fatias, mais suave será o contorno do objeto e maior o tamanho do arquivo gerado.

Neste tipo de *sweep*, a origem do objeto gerado é representado pela extrema direita inferior da área de desenho.

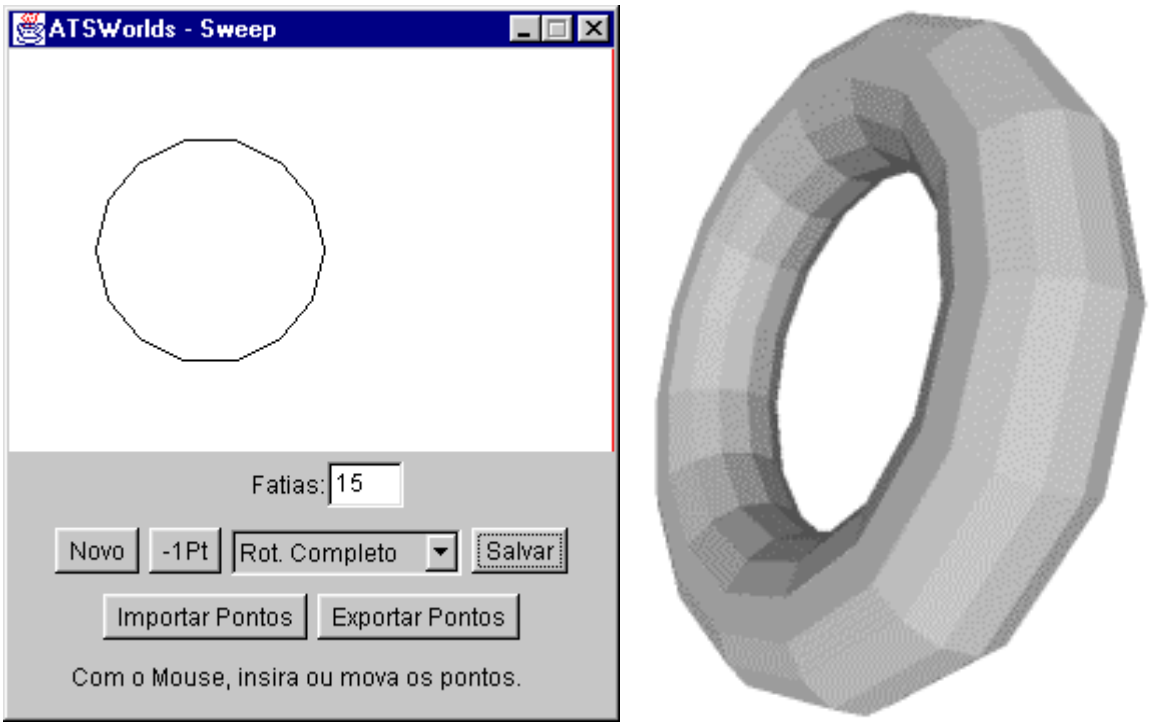
Os pontos são gerados pela rotação dos pontos das arestas desenhadas pelo usuário. Cada ponto desenhado produz tantos pontos no arquivo VRML quantos forem o número de fatias. O número total de pontos do objeto é dado pela multiplicação do número de pontos do polígono pelo número de fatias a serem feitas. Como o polígono é fechado, o primeiro e o último ponto desenhado é o mesmo. Este ponto não é repetido no sistema de modelagens para a criação dos pontos do objeto.

O eixo Y do objeto é o eixo vertical da tela. Os valores de X e Z dos pontos do objeto gerado são calculados pela distância entre o ponto e o eixo de rotação (D). Sendo

assim, o valor X de um ponto é calculado pela multiplicação entre a sua distância do eixo de rotação pelo coseno do incremento angular (IA). O incremento angular é a razão entre o ângulo total de revolução, que neste caso é de 360 graus, pelo número de fatias. O valor Y de um ponto é dado pelo limite inferior da área de desenho (LimiteTela) menos a própria altura do ponto (Ytela), já que a orientação do eixo da tela cresce para baixo e o eixo Y da VRML cresce para cima. O valor Z do ponto é calculado pela multiplicação entre a distância do ponto ao eixo de rotação e o seno do incremento angular.

Assim, para cada ponto e cada fatia é criado um ponto cujos valores são: X = D*cos(IA), Y = (LimiteTela – Ytela)/150 e Z = D*sen(IA).

A figura a seguir mostra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* rotacional completa de polígono fechado e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.11 Sweep Rotacional Completo de Polígono Fechado.
(a) polígono gerador e (b) objeto gerado.

As faces são formadas pela união de pontos vizinhos e os pontos vizinhos da próxima fatia. Por isso, as faces serão sempre formadas por quadriláteros. Como todas as faces serão sempre convexas (formadas por quatro vértices), o arquivo contendo o objeto é gravado com o valor TRUE para o campo *convex*.

Se a orientação da criação das arestas do polígono gerador for no sentido horário, e o polígono for fechado, o ATSWorlds formará todas as faces voltadas para a parte externa do objeto. Pode-se, neste caso, remover o campo *solid* para otimizar o processamento na visualização deste objeto.

5.4.4 Sweep Rotacional Completo de Polígono Aberto

Como no *sweep* rotacional completo de polígono fechado, o *sweep* é realizado pela rotação do polígono sobre um eixo que é representado pela linha vermelha da área de desenho. O parâmetro “Fatias” representa o número de setores que terá o objeto.

O sistema de modelagem foi desenvolvido de forma que o início e o fim do polígono gerador não necessite estar sobre o eixo de rotação. O próprio sistema se encarrega de fechar o objeto. O exemplo demonstrado nas figuras 5.5 a 5.7 mostram um objeto criado por este tipo de *sweep*, onde é desenhada apenas uma aresta. Vê-se que o sistema fechou a base e o topo do objeto.

A origem do objeto gerado, neste tipo de *sweep*, também é representada pela extrema direita inferior da área de desenho.

Os pontos são gerados pela rotação dos pontos das arestas desenhadas pelo usuário. Se os pontos que formam o topo e a base do objeto estiverem sobre o eixo de rotação, será gerado apenas um ponto, não havendo repetições, já que o ponto calculado é o mesmo. O número de pontos que formam o arquivo VRML é dado pela multiplicação do número de fatias pelos pontos das arestas afastadas do eixo de rotação mais os pontos sobre o eixo de rotação (inicial e/ou final). Por exemplo, um polígono formado por quatro pontos, e com

cinco fatias, onde o ponto inicial e o ponto final estão sobre o eixo de rotação, formará um objeto contendo doze pontos (2*5+2). O ponto final formará um ponto, o ponto inicial outro e, os outros dois formarão cinco pontos cada.

A rotação do polígono é realizada no mesmo eixo do *sweep* rotacional de polígono fechado. Assim, o cálculo dos pontos é o mesmo. Para cada ponto e cada fatia é criado um ponto cujos valores são: X = D*cos(IA), Y = (LimiteTela – Ytela)/150 e Z = D*sen(IA).

As faces laterais são formadas pela união de pontos vizinhos e os pontos vizinhos da próxima fatia. Por isso, as faces laterais serão sempre formadas por quadriláteros. Para formar as faces superior e inferior, se estiverem afastados do eixo de rotação, o sistema cria uma face cujos pontos formadores são os que são formados pela rotação do primeiro e do último ponto do polígono gerador. Caso algum destes pontos (inicial e/ou final) estiver sobre o eixo de rotação, serão formadas faces triangulares entre as arestas (do topo e/ou da base) e este ponto.

Então, se algum dos vértices inicial ou final não estão sobre o eixo de rotação, o sistema trata de fechar o objeto, gerando apenas uma face. Por isso, no caso em que se queira fazer objetos nos quais os vértices inicial e/ou final do polígono gerador forem perpendiculares ao o eixo de rotação, é aconselhado deixar que o sistema o faça.

As faces laterais são convexas (quadriláteros). Como os pontos das faces do topo e da base são formados por uma rotação ou por faces triangulares, elas também serão sempre convexas. Portanto, o arquivo contendo o objeto gerado pela técnica de *sweep* rotacional completa é gravado com o valor TRUE no campo *convex*.

Se o ponto inicial do polígono gerador estiver abaixo do ponto final, a orientação deste polígono será no sentido horário. Neste caso, o ATSWorlds formará todas as faces voltadas para a parte externa do objeto. Pode-se, então, remover o campo *solid* para otimizar o processamento na visualização deste objeto.

5.4.5 Sweep Rotacional Parcial de Polígono Fechado

Esse tipo de *sweep* é semelhante ao rotacional completo de polígono fechado, mas difere dele pelo fato deste não fazer uma volta completa em torno do eixo de rotação. Para realizar este tipo de *sweep*, além do eixo de rotação e do parâmetro “Fatias”, existe também o parâmetro “Angulo”. Este representa qual o ângulo total de revolução que a superfície realizará.

O valor do ângulo deverá estar entre zero e 360, excluindo estes. Um ângulo igual a 360 é realizado pelo *sweep* rotacional completo. Ângulos menores que zero não são necessários, já que todos os ângulos de valores negativos formarão objetos semelhantes aos formados por ângulos positivos.

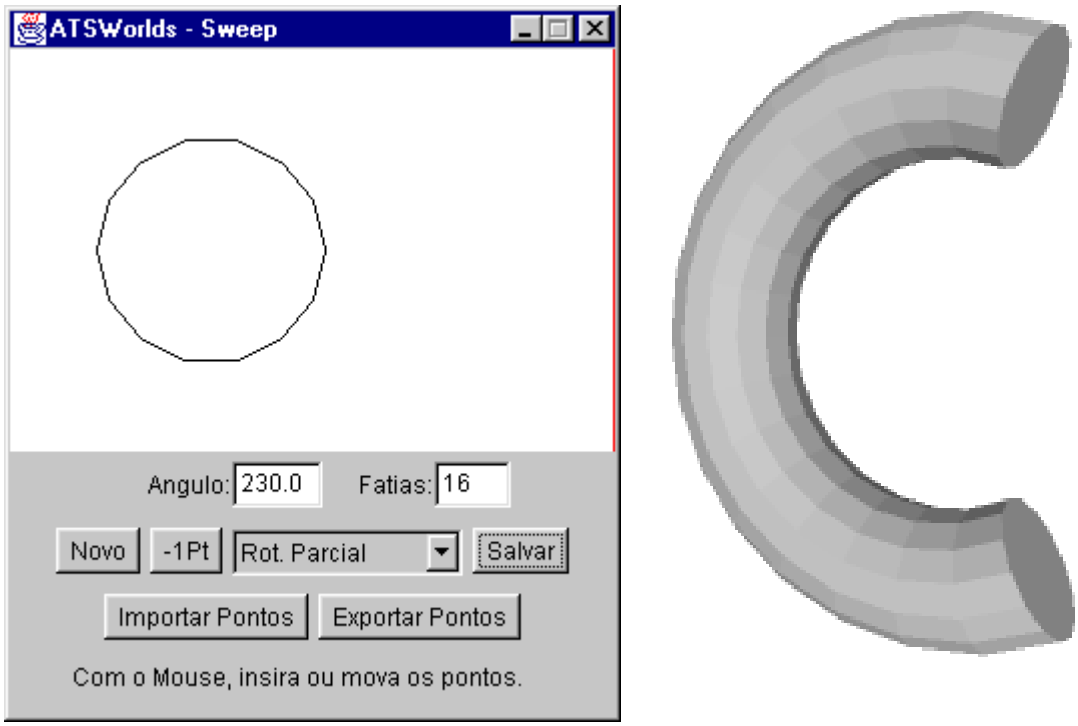
Neste tipo de *sweep*, a origem do objeto gerado é representado pela extrema direita inferior da área de desenho.

No *sweep* rotacional completo, a última fatia se une com a primeira para fechar o objeto. Como, neste caso o objeto é aberto, é necessário mais um conjunto de pontos. Sendo assim, para este tipo de *sweep*, o número total de pontos que formam o arquivo VRML é dado pelo número de pontos do polígono gerador vezes o número de fatias mais um. Por exemplo, um polígono formado por quatro pontos e com cinco fatias, formará um objeto contendo vinte e quatro pontos (4*(5+1)). A rotação do polígono é realizada no mesmo eixo do *sweep* rotacional completo. Assim, o cálculo dos valores dos pontos é o mesmo: X = D*cos(IA), Y = (LimiteTela – Ytela)/150 e Z = D*sen(IA).

As faces laterais são formadas da mesma maneira que no *sweep* do tipo rotacional completo de polígono fechado, pela união de pontos vizinhos e os pontos vizinhos da próxima fatia. Como neste tipo de *sweep* a rotação não é completa, tem-se outras duas faces para fechar o objeto: faces inicial e final do objeto. Estas outras duas faces são formadas pela união dos pontos do polígono gerador que formam o início e o final da rotação. Estas faces tem a mesma forma do polígono gerador.

Se o polígono desenhado pelo usuário for construído seguindo a orientação de sentido horário, todas as faces do objeto serão voltadas para seu exterior. Pode-se neste caso remover o campo *solid* para otimizar o processamento na visualização do objeto. As faces laterais são sempre convexas. Se o polígono gerador também for convexo, o campo *convex* pode ser removido ou atribuído valor TRUE neste caso.

A figura a seguir ilustra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* rotacional parcial de polígono fechado e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.12 Sweep Rotacional Parcial de Polígono Fechado.
(a) polígono gerador e (b) objeto gerado.

5.4.6 Sweep Rotacional Parcial de Polígono Aberto

Este tipo de *sweep* é semelhante ao *sweep* rotacional completo de polígono aberto. O início e o final do polígono gerador não necessitam estar sobre o eixo de rotação e o ângulo deve ficar entre zero e 360.

Se o ponto inicial ou final estiver sobre o eixo de rotação, será criado apenas um ponto para cada, não repetindo estes pontos. Por isso, o cálculo do número de pontos depende se o inicio ou o fim do polígono gerador estiver sobre o eixo de rotação. Por exemplo, um polígono formado por três pontos e com quatro fatias, onde o ponto final está sobre o eixo, formará um objeto contendo onze pontos (1+2*(4+1)). O ponto final irá gerar apenas um ponto, os outros dois pontos formarão outros cinco pontos cada (quatro fatias mais um). A rotação do polígono é realizada no mesmo eixo do *sweep* rotacional completo. O cálculo dos valores dos pontos que não estão sobre o eixo de rotação é o mesmo: X = D*cos(IA), Y = (LimiteTela – Ytela)/150 e Z = D*sen(IA). O ponto que está sobre este eixo, é dado por (0,Y,0), ou seja o X e o Z do ponto estão sobre o eixo de rotação.

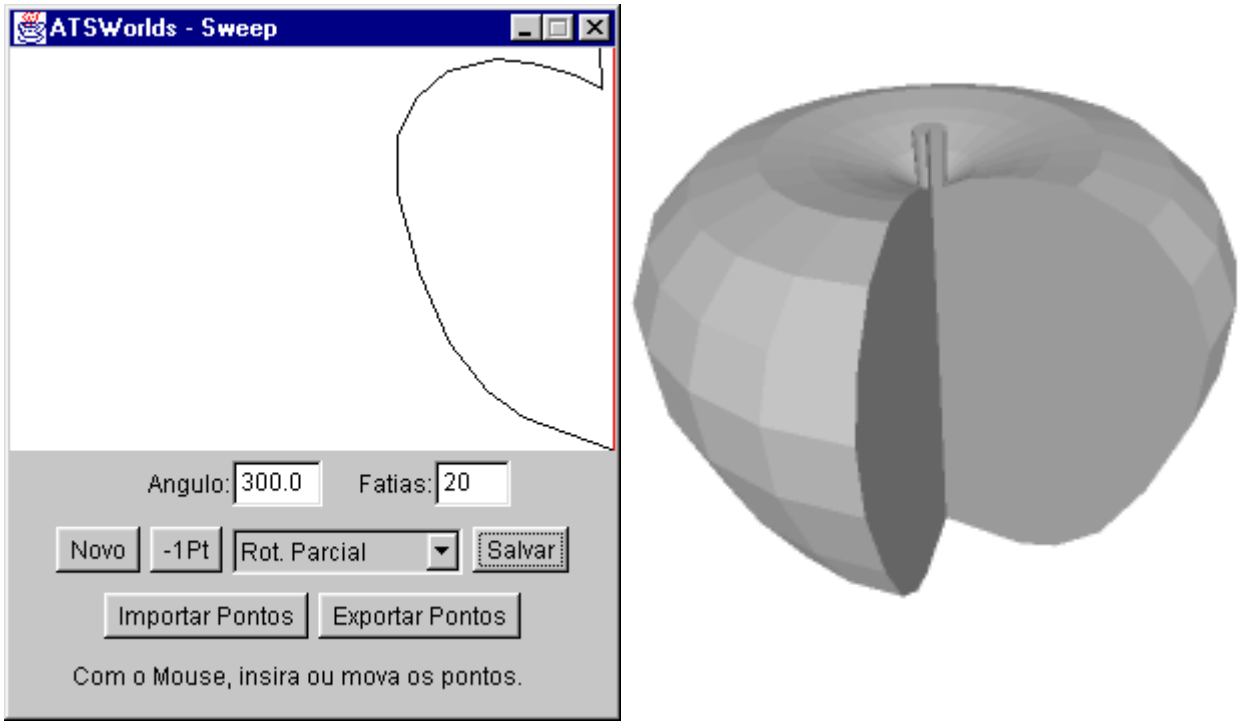
As faces laterais são formadas pela união de pontos vizinhos e os pontos vizinhos da próxima fatia. Se ou o início ou o final do polígono gerador não estiver sobre o eixo, serão criadas outras faces para fechar o polígono.

No *sweep* rotacional completo de polígono aberto, as faces do topo e da base são formadas pela união dos pontos que as formam, sendo sempre convexas as faces. No caso deste tipo de *sweep*, como a rotação não é completa, se forem apenas unidos os pontos, sempre serão formadas faces não convexas. Para se formar faces convexas, são geradas tantas faces quanto forem o número de fatias, gerando faces triangulares.

Como neste tipo de *sweep* a rotação não é completa, tem-se também outras duas faces para fechar o objeto: faces inicial e final do objeto.

As faces laterais e as faces do topo e da base são sempre convexas. Se o polígono gerador também for convexo, o campo *convex* pode ser removido ou atribuído valor TRUE neste caso. Se o ponto inicial do polígono gerador estiver abaixo do ponto final, a orientação deste polígono será no sentido horário e todas as faces do objeto serão voltadas para seu exterior. Pode-se neste caso remover o campo *solid* ou atribuindo o valor TRUE para otimizar o processamento na visualização do objeto.

A figura a seguir mostra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* rotacional parcial de polígono fechado e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.13 Sweep Rotacional Parcial de Polígono Aberto.
(a) polígono gerador e (b) objeto gerado.

5.4.7 Sweep Helicoidal

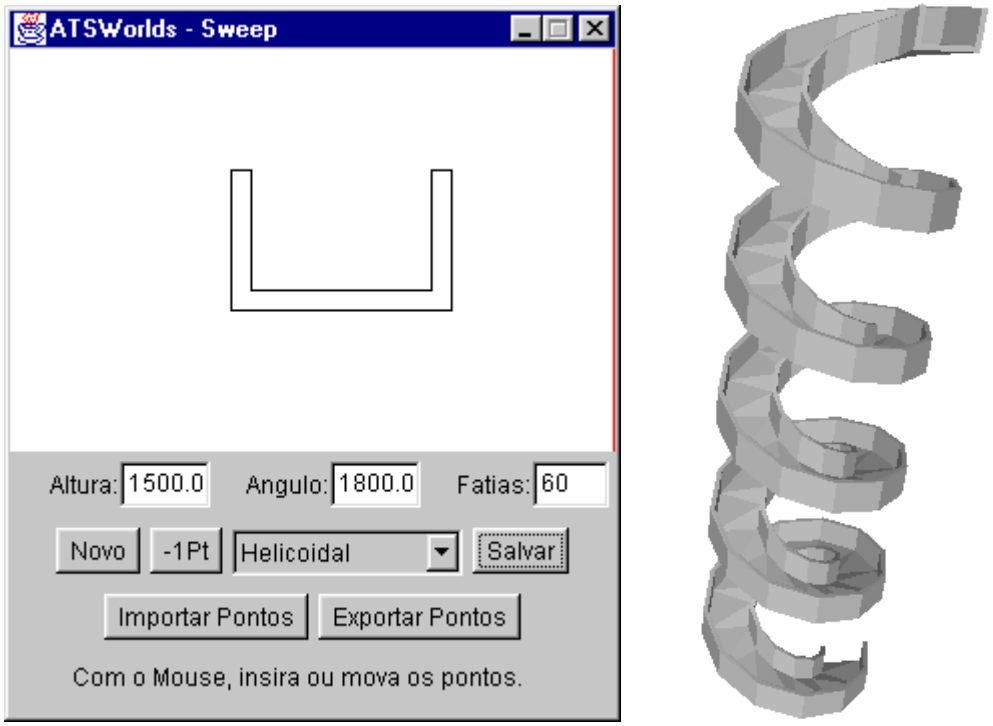
Como foi visto no capítulo sobre a técnica de modelagem por *sweep* no item 4.3, o *sweep* helicoidal é uma combinação dos tipos de *sweep* rotacional parcial de polígono fechado e translacional. Além de incremento de revolução também é feito um incremento de translação no mesmo sentido do eixo de revolução.

A figura 5.2b mostra a interface do sistema quando selecionado este tipo de *sweep*. Neste tipo de *sweep*, a origem do objeto gerado é representada pela extrema direita inferior da área de desenho. São parametrizados a altura do objeto, o ângulo total de revolução e o número de fatias.

Neste tipo de *sweep*, o ângulo de rotação pode ser superior a 360 graus, fazendo mais de uma volta em torno do eixo. O ângulo também pode ser negativo, fazendo uma rotação em sentido inverso.

O número de pontos do polígono é calculado da mesma maneira que no *sweep* rotacional parcial de polígono fechado. A diferença está no cálculo dos valores dos pontos, no qual o valor de Y sofre um incremento linear (IL) enquanto é feita a rotação, ficando assim: X = D*cos(IA), Y = (LimiteTela – Ytela)+IL e Z = D*sen(IA). Este incremento linear é dado pela razão da altura do objeto pelo número de fatias.

A figura a seguir mostra o sistema de modelagens ATSWorlds com um polígono para gerar um objeto pela técnica de *sweep* rotacional parcial de polígono fechado e o objeto gerado a partir deste polígono.



(a) (b)

Figura 5.14 Sweep Helicoidal.
(a) polígono gerador e (b) objeto gerado.

As faces para este tipo de *sweep* são calculadas da mesma forma que no *sweep* rotacional parcial de polígono fechado. As faces final e inicial também terão a mesma forma do polígono gerador.

As faces laterais são sempre convexas. Se o polígono gerador também for convexo, o campo *convex* pode ser removido ou atribuído valor TRUE neste caso.

Se o polígono desenhado pelo usuário for construído seguindo a orientação de sentido horário, todas as faces do objeto serão voltadas para seu exterior. Pode-se neste caso remover o campo *solid* para otimizar o processamento na visualização do objeto.

6 CONCLUSÃO E PERSPECTIVAS

Um sistema de realidade virtual consiste de uma combinação de software, computadores de alto desempenho e periféricos especializados, que permitem criar um ambiente gráfico de aparência realística, no qual o usuário pode se locomover em três dimensões. Um sistema de realidade virtual deve ter a capacidade de determinar a forma, a posição dos objetos, quando há intervenção entre estes objetos, transparências, reflexões e texturas dos objetos, entre outros aspectos.

Com o aumento do poder de processamento dos computadores, e baseada no sucesso da HTML, que dispõe de uma interação limitada em duas dimensões, surgiu a VRML, levando a Web além do paradigma orientado a documentos para um baseado em mundos virtuais 3D. São amplas as capacidades da VRML, como por exemplo, negócios, entretenimento, manufatura, ciência e educação

A realidade virtual é uma tecnologia que permite uma melhor interface homem-máquina, e, junto com a Internet através da VRML, ela se torna uma ferramenta muito poderosa para permitir a criação de novas formas de tratar problemas.

Este trabalho teve como objetivo a integração de técnicas de modelagem com a VRML, através da implementação prática de um software. A técnica de modelagem escolhida para fazer esta integração foi a técnica de modelagem por *sweep,* por ser uma maneira prática e fácil de se construir uma grande variedade de objetos do mundo real.

Neste trabalho foram apresentados alguns conceitos sobre realidade virtual e VRML, algumas das principais técnicas de modelagem usadas para representar objetos do mundo real, destacando-se a técnica de modelagem por *sweep* e comcluindo com a apresentação de como foi realizada a implementação do software desenvolvido junto a este trabalho.

A integração de sistemas de modelagem com a VRML é bastante natural, já que esta linguagem permite a criação de objetos complexos através da definição das faces destes objetos, como foi descrito neste trabalho.

A interface do sistema de modelagem desenvolvido junto a este trabalho é bastante simples de usar, podendo o software ser usado por qualquer pessoa que tenha uma certa noção de *sweep*, não precisando recorrer a grandes, pesados e complicados sistemas de computação gráfica.

No sistema de modelagem ATSWorlds, o usuário desenha seus próprios polígonos para gerar os objetos pela técnica de modelagem por *sweep*, gravando os objetos gerados no formato VRML (.wrl). O usuário também tem a possibilidade de ler e gravar arquivos contendo descrições de polígonos.

Este sistema pode ser usado não só como aplicativo para permitir a gravação dos objetos gerados e de seus polígonos, mas também ser executado em um *browser*, juntamente com um visualizador VRML. Isto é bastante útil para se realizar testes, criando objetos e visualizando-os antes de partir para o objeto final, sem precisar ficar alternando a todo o instante entre aplicação e *browser*.

O ATSWorlds, neste caso, também é uma ferramenta valiosa para alunos que queiram examinar na prática a técnica de modelagem por *sweep* em um sistema bastante simples, tornando este sistema uma ferramenta com forte apelo didático para o ensino da área de computação gráfica, geometria (sólidos geométricos feitos por superfícies de revolução), entre outras.

Como este sistema de modelagem pode ser executado em um *browser*, ele também pode ser distribuído e executado pela Internet. Isto é importante, já que a VRML está voltada principalmente para a Web.

Para isto ser possível, a linguagem usada para o desenvolvimento do software foi a Java. Isto torna o ATSWorlds um programa independente de plataforma, podendo ser executado em qualquer computador com um sistema operacional que tenha suporte à linguagem Java.

Apesar deste sistema ser capaz de construir uma grande variedade de objetos, existem algumas melhorias que poderão ser feitas futuramente, visando aperfeiçoar o trabalho desenvolvido até o presente. Algumas destas melhorias são:

- Construir objetos por outros tipos de *sweep*, como o *Sweep* Translacional com Torção e o *Sweep* Geral, e futuramente, também poderão ser adicionadas outras técnicas de modelagem;
- Inserção e remoção de pontos em uma parte qualquer do polígono gerador, não somente no final deste;
- Desenho de curvas e primitivas como círculo, retângulo, elipse, entre outras;
- Definir cor, transparência, reflexão e textura dos objetos gerados;
- Alterar a proporção *pixels*/m, fazendo uma transformação de escala no objeto gerado;
- Definir rotação e translação sobre o objeto final;
- Permitir a operação com mais de um objeto, unindo-os para formar um outro objeto qualquer.
- Fazer o editor gráfico com janela virtual, possibilitando *zoom* e *panning*;
- Importar e exportar os polígonos geradores para outros formatos de arquivos;

Estudos visando a implementação dos itens citados acima estão em andamento.

GLOSSÁRIO

Applet – Denominação dada a programas controladores de dispositivos e também a pequenos programas utilitários executáveis pelos browsers (como por exemplo os programas na linguagem Java ou que usam a tecnologia ActiveX) que suplementam os aplicativos *browsers* e que chegam ao computador por meio das páginas Web.

Applet Java – Códigos executados por uma aplicação Java por um browser. São muito usados para criar efeitos multimídia nas páginas Web, como animações de vídeo ou música.

BOOM (*Binocular Omni-Orientation Monitor*) – É uma interface usada em realidade virtual, que consiste de uma tela com um visor parecido com o de um periscópio presa a um braço mecânico. Essa interface propicia maior rapidez no monitoramento da posição da cabeça do usuário, permitindo que este possa tirá-lo e colocá-lo rapidamente, não imprimindo o seu peso na cabeça.

Browser – Programa que permite que se leia as informações contidas na Internet através das páginas Web. O browser, também conhecido como navegador, é a interface gráfica da Internet. Por exemplo, Netscape Navigator e Microsoft Internet Explorer.

CAD (*Computer-Aided Design*) – Refere-se ao uso do computador no desenho e projeto de peças industriais, componentes de máquinas ou projetos arquitetônicos e de engenharia.

Frame – Do inglês moldura, quadro. É uma moldura usada para dividir em janelas a tela que exibe um documento ou a página da Web em um browser. Cada janela exibe um documento separado, sendo possível manter uma delas fixas enquanto se move as outras.

GIF (*Graphics Interchange Format*) – Um formato de arquivo de imagens do tipo *bitmap*, identificado pela extensão de nome do arquivo .gif. Este formato que permite alta resolução, mas a 8 bits/pixel, o que significa o máximo de 256 cores. Um formato que se popularizou, especialmente no intercâmbio de imagens pela Internet. O formato GIF é o mais usado em gráficos e elementos hipertexto na Web.

HMD (*Head Mounted Display*) – Esta interface, com aparência de um capacete, é composta basicamente de um par de *displays,* que podem ser de cristal líquido, ou tubos de raios catódicos, fornecendo ao usuário um amplo campo de visão. Utilizado em sistemas de realidade virtual.

HTML (*HyperText Markup Language*) – É a linguagem utilizada para criar documentos em hipertexto para a Web, que serão lidos pelos *browsers*. O HTML é a linguagem mais utilizada na Internet.

Java – Linguagem aberta orientada a objeto (baseada em C++), criada pela Sun Microsystems como um aperfeiçoamento da HTML, implementada por software específico, com melhores recursos de segurança na transferência de dados via Internet e que permite acesso a programas gráficos, interação com o usuário da Web e visualização de animações em tempo real. Pequenos programas Java (denominados *applets*) podem ser transmitidos juntamente com as páginas Web, incluindo animação, planilha de cálculo e outras funções. Esses programas suportam qualquer plataforma ou sistema operacional e podem ser alterados em tempo real, permitindo o suporte a novos protocolos e padrões de dados. A **Tecnologia Java** é uma nova tecnologia, que permite a criação de programas independentes de plataformas. Ou seja, que podem rodar em qualquer computador ou sistema operacional. Essa característica torna os programas Java bastante adequados para distribuição via Internet.

JavaScript – Linguagem de macros, compacta e baseada em objeto, desenvolvida pela empresa Netscape como uma linguagem para servidores e criação de aplicativos clientes para a Internet, lançada orginalmente com o nome de LiveScript.

JDK (*Java Development Kit*) – É o kit de desenvolvimento Java, um produto da empresa Sun Microsystems que fornece o ambiente requerido para programação em Java. O JDK é disponível para uma variedade de plataformas, mais notadamente as Sun Solaris e Microsoft Windows

JPEG (*Joint Photographic Experts Group*) – É um formato gráfico que permite uma alta taxa de compressão de imagens, mantendo a boa qualidade. O JPEG se tornou muito popular com a Internet. Os arquivos JPEG normalmente possuem a extensão JPG.

MIDI – É um formato de arquivos de som instrumentais (.mid). Esses arquivos só reproduzem os instrumentos de uma partitura de música.

MIME (*Multipurpose Internet Mail Extension*) – É um sistema de identificação de dados contidos em um arquivo, baseado na sua extensão, usado na rede Internet como um método padronizado de envio e recebimento de arquivos anexos (gráficos, documentos formatados, sons, etc.) em transmissão de correio eletrônico ou páginas Internet.

Pixel – Contração das palavras "*Picture Element*". É o menor ponto cuja cor e luminosidade podem ser controladas na tela. As imagens são formadas com a combinação de um grande número de *pixels*.

Plug-In – São extensões de códigos desenvolvidos por terceiros, programas ou macros que são instalados adicionalmente, ou integrados, para acrescentar comandos ou funções a outro programa. Geralmente possibilitam carregar certos tipos de arquivos (de sons, animações, etc.). São também os programas que estendem as capacidades de um browser.

VRD (*Virtual Retinal Display*) – É uma interface utilizada em sistemas de realidade virtual. Consiste de um capacete transparente que projeta a imagem diretamente na retina, através de um feixe de laser de baixa intensidade.

VRML (*Virtual Reality Modeling Language*) – É uma linguagem que permite que se apresentem objetos e mundos tridimensionais através da Web. E mais do que simplesmente mostrar cenas estáticas, possibilita que o visitante de um mundo virtual interaja com os seus objetos.

WAV – Abreviação de *wave* (onda). Extensão de nome de arquivos com som digitalizado no Windows.

WWW (*World Wide Web*) – É uma tecnologia que permite que através de softwares de interfaces gráficas (os *browsers*) se leia arquivos do tipo HTML pela Internet.

REFERÊNCIAS BIBLIOGRÁFICAS

[ADA 94] ADAMS, Lee. **Visualização e Realidade Virtual**. São Paulo: Makron Books, 1994.

[CAD 97] CADOZ, Claude. **Realidade Virtual**. São Paulo: Ática, 1997.

[CAS 91] CASACURTA, Alexandre. **SIHMOS, sistema híbrido de modelagem de sólidos**. Porto Alegre: UFRGS, 1991. Dissertação de Mestrado, 169 p.

[DIE 89] DIEGUES, José P. P. **Modelagem Geométrica para Computação Gráfica**. Rio de Janeiro: Achiamé, 1989.

[DUA 98] DUARTE, Lucio M.; SILVA, Daniela E.; ZANONI, Cícero. **"VRML2.0"**. Porto Alegre. http://tinos.pucrs.br/~grv (30/09/1998).

[STE 93] STEIGLEDER, Mauro. **Modelagem de Sólidos Através de Superfícies Splines**. São Leopoldo: Unisinos, 1993. Trabalho de Conclusão, 147 p.

[FOL 96] FOLEY, James D.; DAM, Andries V.; FEINER, Steven K.; HUGHES, John F. **Computer Graphics: Principles and Practice**. 2a. ed. EUA: Addison-Wesley Publishing Company, 1996.

[GOM 98] GOMES, Jonas; VELHO, Luiz. **Computação Gráfica.** Rio de Janeiro: IMPA, 1998.

[HER 94] HERMIDA, Afonso. **Ray Tracing: Aventuras em Computação gráfica & Animação**. Rio de Janeiro: Berkeley, 1994.

[MAN 94] MANSSOUR, Isabel H. **Algoritmo de *Ray-Casting* para Visualização Volumétrica de Dados Obtidos por Tomografia Computadorizada**. Porto Alegre: UFRGS, 1994. Trabalho de Individual I, 63 p.

[MOR 85] MORTENSON, Michael E. **Geometric Modeling**. Jonh Wiley & Sons, New York, 1985.

[SIM 99] Universidade do Minho. Departamento de Informática. *Software, Interaction & Multimedia*. **"VRML Interactive Tutorial"**. Portugal. http://sim.di.uminho.pt/vrml/ (18/05/1999).

[WEB 99] Web 3D Consortium. **"Web 3D Consortium"**. EUA. http://www.vrml.org. (01/07/1999).