UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SISTEMA DE MODELAGEM PARAMÉTRICA DE SÓLIDOS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DENILSON DOMINGOS

BLUMENAU, JUNHO/2002

2002/1-20

PROTÓTIPO DE UM SISTEMA DE MODELAGEM PARAMÉTRICA DE SÓLIDOS

DENILSON DOMINGOS

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo César Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Paulo César Rodacki Gomes

_____

Prof. Dalton Solano dos Reis

_____

Prof. Antonio Carlos Tavares

DEDICATÓRIA

...à memória de meu pai, pela educação e exemplos de vida que me deixou como herança.

AGRADECIMENTOS

A Deus, no qual sempre encontrei forças para alcançar meus objetivos, agradeço por me guiar em mais esta caminhada.

A minha noiva Elisangela, pelo carinho, compreensão, e principalmente amizade, que me incentivaram em todos os momentos.

A minha família, por tudo que representa em minha vida.

Ao professor e amigo Paulo César Rodacki Gomes, pela atenção, apoio, e orientação neste trabalho.

A todos professores do Departamento de Sistemas e Computação, e a meus amigos do curso de Ciências da Computação, pelo convívio e amizade em todos estes anos.

A todas as pessoas que contribuíram direta ou indiretamente na realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho de conclusão de curso visa a construção de um protótipo de um sistema de modelagem paramétrica de sólidos. Através da utilização do 3D ACIS *Modeller* como núcleo de geometria, o protótipo possibilita a construção e visualização de modelos sólidos parametrizados baseados na implementação da técnica para modelagem de sólidos denominada CSG (*Constructive Solid Geometry*), que combina primitivas sólidas e operações booleanas.

ABSTRACT

This work of course conclusion aims for the construction of an prototype of a system of parametric solid modeling. Through the use of 3D ACIS Modeler as geometry nucleus, the prototype makes possible the construction an vizualization of basead parametrics solid models in the implementation of the technique for called solid modeling CSG (Constructive Solid Geometry), that it combines solid primitive and boolean operations.

1 INTRODUÇÃO

Este trabalho visa explorar a área da Computação Gráfica denominada Modelagem Geométrica, mais especificamente a Modelagem de Sólidos, na qual são estudadas alternativas e metodologias para a criação e manipulação de estruturas de dados capazes de representar objetos sólidos utilizando o computador.

A área de engenharia mecânica talvez tenha sido a grande beneficiada pelos avanços ocorridos nesta área. Isto porque através da representação de modelos sólidos no computador, várias propriedades do objeto modelado podem ser definidas na fase de projeto.

Com o contínuo avanço tecnológico, a soma de informações e conhecimentos que devem estar sob o domínio do engenheiro cresce ininterruptamente. Os sistemas CAD (*Computer Aided Design*) se propõem a auxiliar a manipulação e criação destas informações, sistematizando os dados de projeto envolvidos, possibilitando uma rápida reutilização de informações quando necessário (Kerry, 1997).

A fase de projeto de um artefato compreende não apenas a definição da geometria e materiais com os quais ele será construído, mas também o conhecimento prévio de seu comportamento. Isto é feito através de métodos de simulação por computador. Os métodos de simulação podem calcular o comportamento físico e estrutural do artefato sob condições de funcionamento previstas no projeto ou mesmo simular problemas relativos a funcionalidade e ergonomia. No primeiro caso, o método mais usado é o Método dos Elementos Finitos (Reddy, 1993), e para o caso de simulações de ergonomia, o produto mais conhecido é o JACK, da Universidade da Pensylvania, USA (Badler, 2000).

As simulações por computador reduzem o custo de se ter que construir protótipos reais dos artefatos. Como exemplo de simulações, pode-se citar simulações de comportamento aerodinâmico de automóveis e aviões.

Os modelos sólidos são o ingrediente básico dos sistemas CAD e sistemas CAM (*Computer Aided Manufacturing*), e além da simulação, também podem ser usados em arte por computador, jogos 3D, robótica e animação. Avanços em qualquer destes campos dependem de quão bem definidos estão os modelos sólidos criados, e sua aplicação se estende

por diversas áreas tais como indústria cinematográfica, de autopeças, arquitetura, engenharia, medicina, entre outras.

A necessidade de modelar objetos como sólidos, levou ao desenvolvimento de várias técnicas para representá-los. Segundo Mortenson (1985), existem seis famílias para modelagem e representação de sólidos, as mais populares e utilizadas são conhecidas pelas siglas CSG (*Constructive Solid Geometry* ou Geometria Sólida Construtiva) e B-rep (*Boundary Representation* ou Representação por Bordas ou Faces Limitantes).

Existem no mercado vários sistemas CAD voltados a modelagem de sólidos. Estes sistemas comerciais apresentam funcionalidades direcionadas as mais diversas necessidades do usuário. Apesar desta disponibilidade de sistemas, não são comuns as pesquisas e desenvolvimento de aplicações nesta área.

Este trabalho propõe a construção de um protótipo de um sistema de modelagem paramétrica de sólidos utilizando a técnica CSG. A modelagem paramétrica aliada a modelagem de sólidos, permite a criação de modelos com dimensões ligadas através de expressões, o que proporciona a reconstrução do modelo no caso de alterações nos seus parâmetros.

O protótipo do sistema de modelagem permite a criação de modelos sólidos parametrizados previamente definidos, através da implementação da técnica CSG, que combina primitivas sólidos e operações booleanas para a construção do modelo. Com a criação de uma estrutura de dados do tipo árvore binária, será possível armazenar as informações referentes a geometria do modelo para posterior acesso e alteração.

Para a implementação do protótipo foi utilizada a linguagem de programação C++ através do ambiente Microsoft Visual C++ com suporte da biblioteca ACIS. O ACIS é uma coleção de componentes de software, orientado a objetos e escrito em C++, que fornece uma biblioteca de código para representação e manipulação de geometria sólida.

1.1 MOTIVAÇÃO

De efeitos especiais a filmes de longa metragem completos, os produtores estão utilizando computação gráfica para criar efeitos e cenas 2D e 3D. Este é somente um dos

exemplos de como a computação gráfica está presente no cotidiano, e como ela pode ser fascinante. Também pode-se observar que a computação gráfica está intimamente ligada à áreas de pesquisa e tecnologia. A modelagem de sólidos une a experiência profissional do autor na área de projetos mecânicos, e os conhecimentos obtidos durante a graduação em Ciências da Computação, proporcionando através do desenvolvimento deste trabalho, uma grande chance de aplicar conceitos e assimilar novos conhecimentos através de pesquisa e utilização de ferramentas que até então não tinha conhecimento.

1.2 OBJETIVOS

O Objetivo deste trabalho é desenvolver um protótipo de um sistema para modelagem de sólidos parametrizados utilizando a técnica de modelagem CSG.

Os objetivos específicos do trabalho são:

a) utilizar a técnica CSG para modelagem de sólidos através da implementação do protótipo;
b) criar modelos sólidos previamente definidos e parametrizados;
c) realizar alterações nos parâmetros dos sólidos gerados e regenerá-los;
d) salvar e visualizar os modelos sólidos gerados através do protótipo.

1.3 ORGANIZAÇÃO DO TEXTO

O segundo capítulo desta monografia trata da fundamentação teórica do trabalho. Abrange entre outros assuntos modelagem geométrica, sistemas CAD e modelagem de sólidos.

O terceiro capítulo apresenta uma descrição do núcleo de geometria 3D ACIS *Modeller*, suas funcionalidades e características.

O quarto capítulo apresenta a construção do protótipo, sua definição, especificação, implementação, e funcionamento.

No quinto capítulo tem-se as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Até o inicio da década de 80 a Computação Gráfica era um campo pequeno e especializado, principalmente devido ao alto custo dos equipamentos e da escassez de programas gráficos fáceis de utilizar e com preços acessíveis. A modelagem geométrica começava a se tornar conhecida do grande público através dos sistemas CAD, grande novidade tecnológica de então.

Para Mortenson (1985), o termo modelagem geométrica começou a ser utilizado na década de 70 com o rápido desenvolvimento dos computadores, programas CAD e tecnologias de manufatura, e se referia a um conjunto de métodos usados para definir a forma e características geométricas de um objeto. Segundo ele, a modelagem geométrica abrange outra área muitas vezes chamada de geometria computacional e estende-se além dessa para o campo da modelagem de sólidos, criando uma interessante união da geometria com a computação.

O modelamento geométrico permite a representação e descrição geométrica de objetos através de fórmulas matemáticas. Esse modelamento pode representar entidades geométricas (pontos, retas e superfícies) em duas ou três dimensões, dependendo da capacidade do equipamento e do programa utilizado.

A modelagem geométrica provê uma descrição ou modelo muito mais analítico, matemático, e abstrato que o real. Cria-se um modelo porque ele é mais conveniente e econômico que o objeto ou o processo real. Através desse modelo fica mais fácil e prático analisá-lo e testá-lo, uma vez que elimina-se o processo de construção de protótipos físicos (Mortenson, 1985).

Métodos de modelagem geométrica são uma síntese de técnicas de muitas áreas como cálculo vetorial, geometria analítica e descritiva, teoria de conjuntos, análise numérica e matemática matricial, entre outras. Esta combinação de ferramentas matemáticas com a complexidade dos modelos geométricos requer grande esforço computacional (Casacurta, 1999).

A modelagem geométrica é aplicada em diversas áreas onde se utiliza o computador para geração de imagens. No caso particular da modelagem de sólidos, sua principal aplicação é na indústria de manufatura. Com os avanços em hardware que permitiram maior capacidade de processamento, a geração de imagens e definição de estruturas de dados complexas, puderam ser desenvolvidas ao ponto de se tornar acessíveis a um grupo de novos usuários que, através da possibilidade da utilização de sistemas poderosos sem dispensar altos investimentos, fez com que estes se popularizassem.

2.1 SISTEMAS CAD

[SAF1] Comentário: Talvez isto seja eliminado..... ou siga como extenção

Até o inicio da década de 80 a Computação Gráfica era um campo pequeno e especializado, principalmente devido ao alto custo dos equipamentos e da escassez de programas gráficos fáceis de utilizar e com preços acessíveis. A modelagem geométrica começava a se tornar conhecida do grande público através dos sistemas CAD, grande novidade tecnológica de então.

Para Mortenson (1985), o termo modelagem geométrica começou a ser utilizado na década de 70 com o rápido desenvolvimento dos computadores, programas CAD e tecnologias de manufatura, e se referia a um conjunto de métodos usados para definir a forma e características geométricas de um objeto. Segundo ele, a modelagem geométrica abrange outra área muitas vezes chamada de geometria computacional e estende-se além dessa para o campo da modelagem de sólidos, criando uma interessante união da geometria com a computação.

O modelamento geométrico permite a representação e descrição geométrica de objetos através de fórmulas matemáticas. Esse modelamento pode representar entidades geométricas (pontos, retas e superfícies) em duas ou três dimensões, dependendo da capacidade do equipamento e do programa utilizado.

A modelagem geométrica provê uma descrição ou modelo muito mais analítico, matemático, e abstrato que o real. Cria-se um modelo porque ele é mais conveniente e econômico que o objeto ou o processo real. Através desse modelo fica mais fácil e prático analisá-lo e testá-lo, uma vez que elimina-se o processo de construção de protótipos físicos (Mortenson, 1985).

Métodos de modelagem geométrica são uma síntese de técnicas de muitas áreas como cálculo vetorial, geometria analítica e descritiva, teoria de conjuntos, análise numérica e matemática matricial, entre outras. Esta combinação de ferramentas matemáticas com a complexidade dos modelos geométricos requer grande esforço computacional (Casacurta, 1999).

A modelagem geométrica é aplicada em diversas áreas onde se utiliza o computador para geração de imagens. No caso particular da modelagem de sólidos, sua principal aplicação é na indústria de manufatura. Com os avanços em hardware que permitiram maior capacidade de processamento, a geração de imagens e definição de estruturas de dados complexas, puderam ser desenvolvidas ao ponto de se tornar acessíveis a um grupo de novos usuários que, através da possibilidade da utilização de sistemas poderosos sem dispensar altos investimentos, fez com que estes se popularizassem.

Sistemas CAD (*Computer Aided Design*, ou Projeto Assistido por Computador) são utilizados para desenhar ou modelar componentes e sistemas mecânicos, elétricos, eletromecânicos e eletrônicos, incluindo edifícios, automóveis, aviões, navios, circuitos integrados e redes telefônicas e de computadores. Sua utilização pelas indústrias vem crescendo de acordo com a evolução constante que estes sistemas apresentam, a ponto de serem encarados hoje pelas indústrias como fator estratégico nos investimentos em tecnologia e soluções de engenharia. Estes sistemas sempre foram encarados como pranchetas eletrônicas, ou seja, eram utilizados para facilitar o trabalho do projetista na sua tarefa de desenhar artefatos. A evolução destes sistemas possibilitou não apenas migrar de representações em duas dimensões para representações completas em suas três dimensões, como também propiciou à engenheiros e projetistas a elaboração de projetos completos e muito mais analíticos, onde várias propriedades podem ser testadas nesta fase, e simulações de montagem e funcionamento podem ser realizadas com total confiança.

A primeira demonstração do computador como ferramenta de desenho e projeto foi feito pelo *Massachusetts Institute of Techonology* em 1963 pelo Dr. Ivan Sutherland. Seu sistema, chamado Sketchpad, utilizava um tubo de raios catódicos e uma caneta óptica para desenhar. Antes de 1976, o projeto e o *layout* de circuitos impressos era o maior uso para

CAD. A engenharia mecânica, deste então, suplantou a eletrônica e continua a expandir o uso dos sistemas CAD e suas aplicações em projeto, análise, e comando numérico (Giesecke, 2002).

Os sistemas CAD eram utilizados originalmente somente para criar desenhos de execução 2D, mas o advento de programas CAD 3D gerou avanços na fabricação dos produtos e também no teste de projeto, utilizando programas de análise como o Método dos Elementos Finitos. Além disso, sistemas CAD podem ser ligados à equipamentos de comando numérico, ou robôs para fabricação ou controle de sistemas.

Na atividade de projeto, o emprego do computador através de sistemas CAD tem como finalidade:

a) otimização do trabalho do projetista;

b) melhoria da qualidade do projeto através da rapidez de análise;

c) desenvolvimento de um número maior de soluções possíveis em menor tempo e com menor número de erros;

d) melhoria da comunicação visual, através da padronização dos desenhos;

e) criação de um banco de dados para os processos de manufatura, pela documentação detalhada dos dados de especificação do produto (Giesecke, 2002).

Alguns sistemas CAD podem suportar qualquer atividade de projeto, na sua criação, modificação, recuperação ou documentação. Apesar da sigla CAD incluir o termo *Design*, observa-se que são poucos os casos em que o computador efetivamente projeta alguma coisa, ele serve como uma poderosa ferramenta de auxílio à confecção de modelos e desenhos de engenharia. Sua maior contribuição ocorre no modelamento dos produtos e componentes, e no detalhamento de seus desenhos, o que propicia no primeiro caso a utilização dos modelos em testes e simulações, e no segundo a atualização automática dos detalhes dos componentes após modificações realizadas nos mesmos.

Os sistemas que atuam na área de cálculos de engenharia são chamados de CAE (*Computer Aided Engineering* ou Engenharia Assistida por Computador). Nestes sistemas são realizadas análises e simulações do tipo análise estrutural por elementos finitos, análise de escoamento, simulações multi-corpos, análise de tensões, entre outras (Giesecke, 2002).

O ingrediente básico destes sistemas é o modelo geométrico sólido desenvolvido a partir de sistemas CAD, daí em diante tem início uma série de processos que são realizados a partir do modelo desenvolvido na fase de projeto do produto. Nota-se que todo o processo de fabricação de um produto está condicionado a sua definição, desta forma têm-se nos sistemas CAD a base de um processo integrado de engenharia e manufatura, justificando assim toda a atenção dispensada a tal sistema.

Outros sistemas que utilizam o CAD como ponto de partida para suas aplicações são os sistemas de manufatura assistida por computador, ou CAM, e os sistemas de manufatura integrada por computador, ou CIM (*Computer Integrated Manufacturing*). Os sistemas CAM estão relacionados aos programas para operação de máquinas de controle numérico que vão fabricar ou materializar o modelo matemático criado no sistema CAD. CAD/CAM é a integração de computadores nos processos de projeto e produção, indispensável hoje em qualquer indústria. CAD, CAE, CAM e CIM formam o processo completo com o uso de computadores, incluindo projeto, produção, publicação de material técnico, propaganda e contabilidade de custos.

2.1.1 EVOLUÇÃO DOS SISTEMAS CAD

Os sistemas CAD somente saíram dos laboratórios de pesquisas e universidades para se tornarem sistemas comercias no início da década de 80. Juntamente com explosão do IBM PC, surgiram os primeiros programas de CAD para microcomputadores. Estes primeiros programas eram bastante limitados em seus recursos em função da capacidade dos processadores, memória, interfaces gráficas e dispositivos de entrada interativa. Nesta época, era até mesmo incomun que os computadores fossem dotados de *mouse*, que hoje é um periférico indispensável. Por isso, a entrada de dados era feita basicamente através do teclado com comandos escritos e usando-se teclas de movimentação de cursor para deslocar um apontador gráfico na tela (Bogado, 1997).

Em 1982, a empresa americana Autodesk (*http://www.autodesk.com*), lançou o que viria a ser o mais popular de todos os programas de CAD: o AutoCad. Comparado com os padrões atuais pode-se dizer que o AutoCad era rudimentar. Os seus recursos limitavam-se ao desenho de linhas, círculos, elipses, textos e outros elementos geométricos simples. Entre outros recursos, era possível mover, copiar, apagar e refletir partes do desenho. Mesmo com

todas as limitações, houve um grande interesse por parte dos usuários do IBM PC e a pirataria encarregou-se de fazer com que o AutoCAD surgisse em milhares de computadores pessoais em todo o mundo.

O sucesso da Autodesk estimulou outros produtores de software a investir neste mercado. Muitas empresas que tradicionalmente operavam no mercado de *workstations* e *mainframes* resolveram portar suas aplicações para os microcomputadores. Após poucos anos já era possível escolher dentre uma vasta gama de produtos (Bogado, 1997).

A rápida evolução dos microcomputadores baseados em processadores Intel fez com que os programas ficassem cada vez mais sofisticados e fossem empregados para tarefas cada vez mais complexas. O aumento da complexidade das tarefas fez com que aumentasse a complexidade dos programas, tornando estes não mais meras pranchetas eletrônicas, mas verdadeiras ferramentas de projeto e simulação. Para que isso acontecesse foi necessário abandonar os conceitos aplicados até então para a geração de desenhos 2D, e partir para a geração de desenhos 3D, evoluindo posteriormente para a definição de modelos sólidos, utilizados atualmente em grande escala por uma vasta gama de sistemas CAD.

Apesar de toda a evolução desses sistemas, muitas vezes o usuário deseja simplesmente produzir desenhos precisos de componentes e conjuntos usando o computador como uma prancheta eletrônica, da mesma forma que o editor de textos substitui a máquina de escrever, muitos usuários encaram os programas e equipamentos de CAD como meros instrumentos de desenho. Restringindo a visão a este aspecto mais evidente da tecnologia, muitas vezes deixa-se de aproveitar todas as possibilidades que estes sistemas oferecem.

Muitas pessoas ligadas ao campo da informática desconhecem o real significado do termo CAD. Muitos associam CAD a AutoCad considerando que sejam sinônimos. A associação não é de todo errada uma vez que AutoCad é o nome de um programa comercial de CAD extremamente popular, mas este é um dentre muitos programas. Para citar alguns, têm-se MicroStation (similar ao AutoCad), e ProEngineer, SolidEdge, Unigraphics e SolidWorks entre os sistemas CAD 3D mais populares.

Encarados em uma perspectiva mais ampla, os sistemas CAD permitem interagir com o componente ou sistema que está sendo projetado. Portanto não têm-se um mero desenho, mas sim um modelo matemático que representa de maneira fiel o objeto. Este modelo apesar de não ser palpável pode ser submetido a esforços mecânicos, aquecido, percorrido por

correntes elétricas, deslocado no espaço tridimensional, isto é, submetido a diversas situações encontradas no mundo real. Esta manipulação virtual permite economizar muito tempo e dinheiro eliminando a necessidade de construção de protótipos ou maquetes.

2.1.2 SISTEMAS CAD 2D

Como visto acima, os primeiros sistemas CAD ofereciam ao usuário primitivas geométricas simples que permitiam a geração de desenhos em duas dimensões. Com o avanço destes sistemas e a possibilidade de modelar objetos em três dimensões, os sistemas que oferecem apenas a interface para desenhos em duas dimensões se tornaram menos atrativos e um pouco ultrapassados. Mesmo assim são utilizados em larga escala nas indústrias, pois os desenhos em duas dimensões são ainda quase que indispensáveis nos processos de fabricação, tanto que a grande maioria dos sistemas CAD 3D apresentam recursos para geração de desenhos 2D, com a vantagem de a partir do objeto modelado, gerar e atualizar vistas e cortes do desenho.

Uma das vantagens dos sistemas CAD 2D é o rápido treinamento de operadores, geralmente habituados ao uso das pranchetas comuns. Mas o seu uso é limitado, correndo-se o risco de transformar o sistema em uma simples prancheta eletrônica, pouco mais produtiva que as pranchetas comuns (Bogado, 1997).

Outra vantagem é que para algumas aplicações a representação 2D é suficiente, como por exemplo em projetos de esquemas elétricos, hidráulicos, circuitos e placas eletrônicas, onde não há necessidade de informações volumétricas. Nestes casos não justifica-se o investimento em sistemas mais avançados, e aí está o grande mercado dos sistemas CAD 2D. Também na criação de vários tipos de croquis, para, por exemplo, suportar a produção, o CAD 2D é mais apropriado. Pode-se concluir que os sistemas CAD 2D, continuam quase que indispensáveis, visto que são raros os casos de indústrias que os substituíram totalmente por sistemas 3D. Geralmente utiliza-se sistemas CAD 3D para algumas áreas e os sistemas 2D continuam sendo empregados para as áreas onde são mais vantajosos, isto pode ser comprovado pela grande aceitação que o AutoCad, utilizado principalmente para desenhos 2D, encontra no mercado.

2.1.3 SISTEMAS CAD 3D

Os sistemas CAD 3D introduziram novos conceitos de desenvolvimento de produtos, onde podem ser definidos na fase de projeto além dos aspectos geométricos do artefato, várias outras informações que antes apenas poderiam ser definidas com a construção de protótipos, isto porque o advento dos sistemas CAD 3D propiciou o desenvolvimento dos sistemas CAE.

Os sistemas CAD 3D estão ligados a modelagem de sólidos, visto que os principais sistemas fornecem não apenas a representação em três dimensões através de linhas, mas a representação de modelos sólidos completos, assim não se torna necessário aqui expor suas vantagens, uma vez já demonstrado anteriormente todo o universo de possibilidades de aplicação de modelos sólidos.

A utilização dos sistemas CAD 3D apresenta as dificuldades que são próprias do processo de desenho, pois o projetista é obrigado a considerar as três dimensões simultaneamente. Em alguns casos, a utilização do modelo 3D é imprescindível, por isso a implantação destes sistemas é necessária, e neste caso é exigido um nível de conhecimento maior do projetista para utilização do sistema, bem como um investimento maior em treinamento por se tratar de um sistema muito mais complexo que os sistemas 2D.

2.1.4 SISTEMAS CAM

Os sistemas CAM podem ser considerados como uma das áreas de maior aplicação dos modelos sólidos gerados por sistemas CAD. Nem sempre um modelo necessita ser submetido a análises através dos sistemas CAE, porém para sua fabricação, a utilização de máquinas de comando numérico computadorizado requer programação de usinagem, realizada através dos sistemas CAM.

Pode-se definir CAM como auxílio via computador da preparação da manufatura, representando as tecnologias utilizadas no chão de fábrica, dizendo não só a respeito da automação da manufatura, como CNC (Comando Numérico Computadorizado), CLP (Controle Lógico Programável), coletores de dados (DNC), como também a tomada de decisão, plano operacional, etc (Giesecke, 2002).

Apesar de toda esta abrangência, o termo CAM, as vezes, ainda é sinônimo da programação CN (Comando Numérico), conceito que ficou muito difundido com a sigla CAD/CAM, que representa módulos de programação CN em sistemas CAM. Isto é muito normal, visto que esta é a aplicação mais difundida dos sistemas CAM. Os sistemas CN normalmente são utilizados para o cálculo do caminho da ferramenta, a partir da representação geométrica da peça disponível na forma computacional. Outra opção é a simulação final do programa, onde pode-se visualizar a usinagem.

2.2 TÉCNICAS PARA REPRESENTAÇÃO 3D

Como visto anteriormente, a representação de objetos em três dimensões possui inúmeras vantagens sobre a representação através de duas dimensões, que representa o objeto com auxílio de projeções planares e projeções perspectivas. A evolução das técnicas para representação de objetos 3D permitiram a criação de modelos sólidos, de modo que a representação de um objeto passou a definir este como um objeto sólido, com todas as propriedades e definições que este tipo de modelo exige. Isto implica na utilização de algoritmos e estruturas muito mais complexos do que os utilizados para representação do desenho em três dimensões. Neste estágio uma técnica deixa de ser exclusivamente voltada para a representação, e passa a ser uma técnica para definição de modelos sólidos. As técnicas para representação de objetos 3D são descritas a seguir.

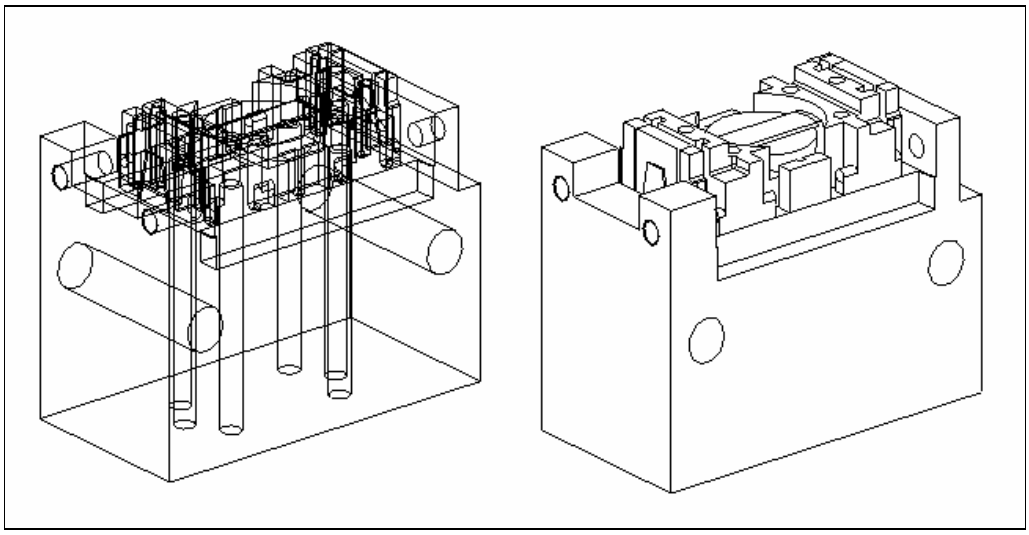
2.2.1 REPRESENTAÇÃO POR *WIREFRAME*

Nesta representação os objetos são descritos por um conjunto de arestas que definem as bordas do objeto. O método nada mais é do que uma extensão 3D do método de representação de objetos 2D por arestas. Em resumo, o objeto é representado em suas três dimensões através de linhas.

A principal vantagem desta técnica é a sua velocidade na exibição dos modelos pois é necessário apenas exibir um conjunto de linhas. A representação por *wireframe* é a mais comumente encontrada nos softwares comerciais. Nesta representação as arestas dos objetos são mostradas como linhas. Em objetos com superfícies curvas, são adicionadas linhas de contorno (Giesecke, 2002).

Existem alguns inconvenientes na representação por *wireframe*, principalmente na representação de objetos complexos, onde forma-se um emaranhado de linhas, prejudicando o entendimento do desenho. Este problema pode ser amenizado com a utilização de algoritmos para remoção de linhas ocultas. Na figura 1 têm-se um mesmo objeto representado em *wireframe*. O primeiro através de uma representação completa, e o segundo após a aplicação de remoção de linhas ocultas.

FIGURA 1 - OBJETO REPRESENTADO EM *WIREFRAME*



Outro problema, é que a representação por *wireframe* pode gerar um representação "ambígua", ou seja, quando o modelo é exibido pode dar margem a mais de uma interpretação. O problema maior não reside propriamente no fato de que a simples exibição das linhas gera ambigüidades, mas sim, na constatação de que o modelo de representação por *wireframe* não fornece informações suficientes para que seja determinada qual das possíveis interpretações é a correta. Assim é bastante difícil, e em alguns casos impossível, realizar certas operações como a remoção de linha ocultas, determinação de massa, volume, inclusão de pontos dentro do modelo, etc.

No passado, a modelagem por *wireframe* era o principal método utilizado pelos sistemas CAD, possibilitando ligar linhas entre pontos nos espaços 3D, permitindo a criação de modelos espaciais e garantindo a consistência de vistas 2D derivadas. Com o avanço tecnológico e maior capacidade de processamento dos computadores, esses sistemas

começaram a ser substituídos pelos baseados nos métodos de modelagem sólida. Isto também aconteceu, em parte, devido a dificuldade de uso dos modelos *wireframe* quando necessário incorporá-los em softwares de análise ou manufatura, já que não possuem nenhum tipo de informação relacionada a características físicas dos componentes reais (Giesecke, 2002).

2.2.2 REPRESENTAÇÃO POR SUPERFÍCIES

Através desta técnica um objeto é representado através de suas superfícies ou faces. Para entender esta técnica pode-se fazer uma analogia com a técnica *wireframe*. Naquele caso um conjunto de arestas define o objeto representado, e no caso da representação por superfícies, um conjunto de arestas define uma face, que é uma região fechada, e o conjunto de faces define o objeto a ser representado.

Segundo Giesecke (2002), as faces que compõem o objeto podem ser implementadas de duas formas:

a) através de uma lista de vértices explícitos, onde percorre-se a face através das coordenadas dos vértices no espaço:

   - FACE: (x1, y1, z1,) – (x2, y2, z2) - ... - (xn, yn, zn);

b) através de duas listas, onde na primeira é armazenada apenas a topologia da face, ou seja, o número dos vértices que a compõe:

   - FACE: v1, v2, v3, ..., vn;

   e na segunda onde é definida a geometria da face, com a explicitação das coordenadas de cada vértice do objeto:

   - 1 – (x1, y1, z1)

   - 2 – (x2, y2, z2)

   - ... ...

   - n – (xn, yn, zn).

Esta representação pode atingir um nível de complexidade ao ponto de conseguir representar modelos sólidos, a partir daí torna-se uma técnica não apenas para representação de desenhos em três dimensões, e sim para a modelagem de sólidos. Para isso, é necessário armazenar além das informações das faces, outras informações para testar e validar a consistência do modelo criado, como o relacionamento das faces com as faces vizinhas, o

lado interno ou externo da face, etc. A técnica para modelagem de sólidos que utiliza faces para definir o modelo chama-se representação por limites ou B-rep, e será detalhada no próximo capítulo.

2.2.3 REPRESENTAÇÃO SÓLIDA

A representação de um objeto sólido envolve muito mais que a simples visualização de sua geometria em três dimensões. O grande diferencial da representação sólida é que ela envolve a criação e manutenção do modelo sólido, que é o objeto em questão. Os modelos sólidos necessitam captar adequadamente a geometria do objeto, assim uma variedade de operações úteis podem ser executadas utilizando este modelo. A necessidade de modelar objetos como sólidos resultou no desenvolvimento de várias maneiras especializadas de representá-los. Cada técnica necessita validar o modelo criado, ou seja, garantir que ele represente de forma consistente a geometria criada.

2.3 MODELAGEM DE SÓLIDOS

Antes de iniciar a abordagem sobre modelagem de sólidos faz-se necessário definir o que é um modelo sólido. Um modelo sólido é uma representação digital da geometria de um objeto físico existente ou idealizado (Requicha, 1999). Quando se fala em modelo sólido, geralmente faz-se uma ligação com artefatos manufaturados, já que a palavra sólido está muito ligada a massa ou volume, e também devido a grande aplicação da modelagem de sólidos nas indústrias. Porém um modelo sólido pode ser qualquer objeto do mundo real ou um objeto imaginário que esteja representado computacionalmente através das técnicas de modelagem correspondentes.

Cada vez mais a computação gráfica tem sido utilizada para a criação de imagens que representam modelos do mundo real (Foley, 1990). Diante desta afirmação e do que foi exposto até este capítulo sobre a evolução dos sistemas CAD, pode-se observar a tendência e a preocupação com que objetos representados computacionalmente, descrevam modelos cada vez mais fiéis aos objetos do mundo real. A área da computação gráfica na qual discutem-se metodologias e técnicas para definição de modelos sólidos é denominada modelagem de sólidos.

Uma definição para a modelagem de sólidos pode ser encontrada em Requicha (1985): “o termo modelagem de sólidos engloba um conjunto de teorias, técnicas e sistemas focalizados em representações completas de sólidos. Representações que permitam, pelo menos em princípio, o cálculo de qualquer propriedade geométrica bem definida de qualquer sólido representável ”.

Objetos do mundo real podem e sempre foram representados através de desenhos. Nas áreas de engenharia os desenhos em duas dimensões representam objetos através da utilização de projeções planares. Com o advento dos sistemas CAD, ficou mais fácil representar objetos através da utilização das projeções planares e também projeções perspectivas, que auxiliam na interpretação da geometria do objeto representado. Uma nova geração destes sistemas permitiu a modelagem de planos e superfícies 3D que representam objetos em suas três dimensões.

Porém, assim como um conjunto de linhas e curvas 2D não necessariamente descrevem os limites de uma área fechada, uma coleção de planos e superfícies 3D não necessariamente formam um volume fechado. Em muitas aplicações no entanto, é importante distinguir entre o lado de dentro, o de fora, ou uma superfície de um objeto 3D, e ser capaz de computar propriedades do objeto que dependem desta distinção. A necessidade de modelar objetos como sólidos, resultou no desenvolvimento de várias técnicas especializadas em representá-los (Foley, 1990).

Um objeto simples como um cubo pode ser representado e muito bem compreendido através de um desenho de duas dimensões com o auxílio das projeções planares. Utilizando um desenho de três dimensões este cubo fica mais facilmente compreendido, porém ainda não pode ser considerado um modelo sólido. Nestes dois exemplos de representação pode-se facilmente determinar características geométricas do objeto tipo cubo representado. Porém, se for necessário analisar alguma propriedade física como volume, massa ou centro de gravidade, se faz necessária a utilização de cálculos físicos e matemáticos para obtenção dos respectivos valores. Neste caso a tarefa não parece complicada, porém na indústria são os artefatos complexos e não triviais que necessitam ser submetidos a tais análises. Desta forma, os cálculos podem ser muito extensos, além de que a representação e a interpretação de objetos complexos através de desenhos em duas dimensões pode ser considerada trabalhosa. Se o objeto do tipo cubo for definido por modelagem sólida, ele passa a adquirir propriedades

físicas como volume, e caracterizando sua densidade, consegue-se obter outras características como peso e massa, assim o computador pode calcular diversas propriedades do modelo, como centro de gravidade, momento de inércia, etc.

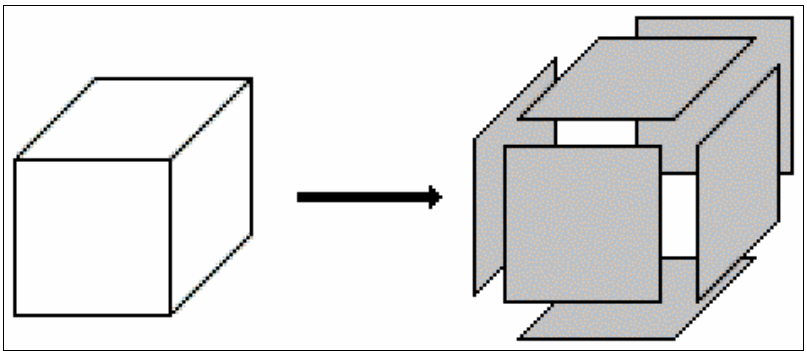
A modelagem de sólidos impulsionou outra área já citada, o CAE. As análises estruturais realizadas por sistemas CAE só foram possíveis graças a possibilidade de construir modelos sólidos que preservam as características geométricas de um objeto real.

No estado atual da modelagem de sólidos, dispõe-se de várias formas para representar computacionalmente um sólido. Cada qual possui vantagens e desvantagens provenientes do paradigma sobre o qual se encontram baseadas. As diversas técnicas para modelagem de sólidos são descritas nesta sessão.

2.3.1 REPRESENTAÇÃO POR LIMITES

A representação de um sólido por limites, do inglês: *Boundary Representation*, é também conhecida como representação B-rep. O modelo por representação de limites define um sólido indiretamente através da representação das suas superfícies limitantes (Foley, 1990). Computacionalmente as superfícies do modelo são divididas em faces. Cada face é limitada por um conjunto de vértices e arestas. Em outras palavras, elas representam um objeto sólido através da subdivisão deste objeto em faces.

FIGURA 2 - SÓLIDO REPRESENTADO POR FACES



FONTE: Casacurta (1999)

De acordo com (Silva, 1999), 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 sólido definido pela representação por limites incluem as seguintes condições:

a) o conjunto de faces que compõem o modelo se fecha, ou seja, forma um invólucro completo em torno do objeto a ser representado e sem partes pendentes;

b) as faces do modelo não interseccionam outras faces, exceto as faces que possuem vértices e arestas em comum;

c) as faces são superfícies simples que não se auto-interseccionam.

Uma face pode definir diversos tipos de superfícies, tais como planos, superfícies quadráticas, ou superfícies paramétricas. Pode-se identificar em uma face dois lados: o interno e o externo. Portanto, quando se descreve o conjunto de arestas que compõe uma face, deve-se levar em conta a orientação da superfície, ou seja, se demonstra o lado de dentro ou de fora do sólido. Para fazer esta distinção, normalmente enumera-se os vértices da face. Dependendo do sentido com que estes são percorridos (horário ou anti-horário), determina-se o lado que esta sendo definido por esta face, interno ou externo ao sólido.

As faces que delimitam o modelo são também conhecidas como polígonos. Um conjunto de polígonos que satisfaça as condições descritas acima delimita uma região fechada do espaço, e esta região define o interior do modelo. O objeto formado por esta técnica é normalmente chamado de Poliedro, ou seja, composto de muitos Diedros (Diedro = semi-espaço). Um Poliedro é um sólido limitado por um conjunto de polígonos, no qual cada aresta é membro de um número par de polígonos (Casacurta, 1999).

Esta técnica também pode ser considerada uma extensão da modelagem 2D por arestas vista no capitulo 3. A primeira geração de modeladores B-rep representava objetos sólidos apenas por tabelas de faces, arestas e vértices. Desta forma, eles somente suportavam objetos com faces planas. Superfícies curvas eram modeladas utilizando-se aproximação por polígonos.

A segunda geração de modeladores B-rep incluiu objetos primitivos com superfícies analíticas, como cilindros, esferas, cones, etc. Eles permitem a criação de modelos muito mais

complexos e com geometria exata. Para isto foi necessário o uso de algoritmos de interseção muito mais complexos.

A implementação desta técnica necessita conter uma estrutura de dados com a descrição das superfícies e das bordas que delimitam o modelo gerado, e para agilizar o processo torna-se conveniente manter informações sobre o relacionamento de cada face com suas vizinhanças.

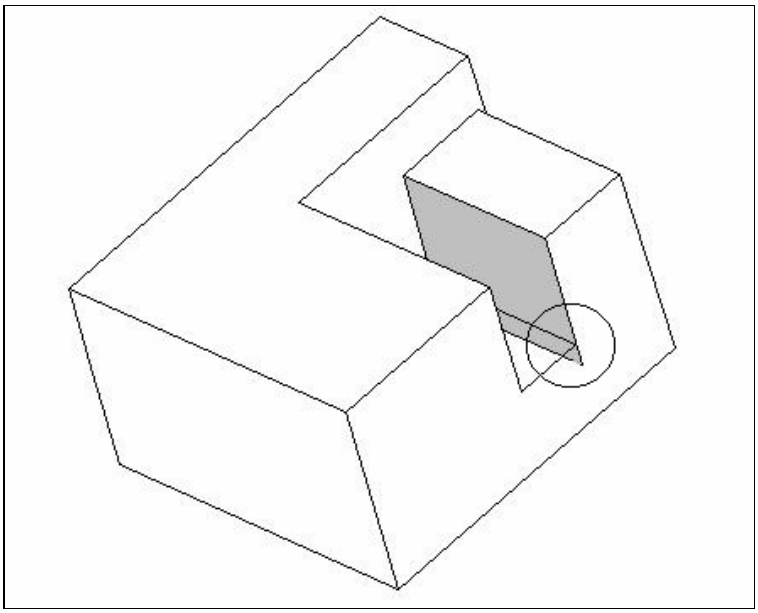
Um problema para este tipo de modelagem é a descrição de superfícies inválidas, geralmente superfícies que não possuem interseção com outras ou excedem esta interseção. Normalmente este problema é verificado em objetos complexos, onde a determinação das faces e seus relacionamentos se torna extremamente difícil para o usuário. Neste caso as faces que delimitam o modelo não se encontram totalmente fechadas, e este torna-se inválido, uma vez que não é possível determinar suas características, muito menos submetê-lo à simulações.

A figura 3 demonstra um modelo inválido, pois a face destacada excede sua interseção com a face perpendicular a ela. A parte excedente torna-se na interpretação do modelo uma nova face, porém esta não possui interseção com nenhuma outra face, assim quando o modelo for submetido a um teste de validade, não será possível determinar um volume fechado formado por estas faces.

Apesar da representação de modelos sólidos por limites não ser única, já que vários conjuntos de faces podem representar o mesmo sólido, não há ambigüidade na representação, ou seja, uma vez fixado o conjunto de faces, este define apenas um e somente um modelo sólido.

Além da criação de objetos complexos ser tarefa difícil, a sua instabilidade no que tange à garantia de validade geométrica e o custo de armazenamento e manipulação das estruturas de dados utilizadas, também dificultam a implementação desta técnica. A representação por limites é particularmente interessante em aplicações que necessitem de visualização rápida, pois sua estrutura de dados possibilita a aplicação da maioria dos algoritmos de visualização de forma simples e direta.

FIGURA 3 - FACE INVÁLIDA DE UM MODELO SÓLIDO



2.3.2 PARTICIONAMENTO ESPACIAL

Também descrita por alguns autores como Representação por Decomposição Volumétrica. Nesta técnica, um sólido é decomposto em um conjunto de sólidos adjacentes e não interseccionados que são mais primitivos, porém não necessariamente do mesmo tipo do modelo sólido original. A estes sólidos dá-se o nome de primitivas ou blocos básicos (Foley, 1990).

Primitivas podem variar no tipo, tamanho, posição, parametrização e orientação, mais ou menos como nos jogos de montar infantis, onde diversas blocos diferentes dão forma a um modelo final. O quanto um objeto é decomposto depende de quão primitivos os sólidos devem ser para que se possa executar rapidamente as operações necessárias, e a maneira como os blocos são combinados é que distingue cada variação desta técnica. Isto implica na definição de diferentes tipos de primitivas que possibilitem atender diferentes necessidades conforme a aplicação desejada. Assim cada variação desta técnica é direcionada para uma aplicação diferente, como descrito a seguir.

2.3.2.1 ENUMERAÇÃO POR OCUPAÇÃO ESPACIAL

A enumeração por ocupação espacial (*spatial occupancy enumeration*) é um caso especial da decomposição celular na qual o sólido é decomposto em células idênticas organizadas em uma grade regular e fixa. Essas células são freqüentemente chamadas de *voxels* (*volume elements*), em analogia a *pixels* (Foley, 1990).

Pode-se interpretar um modelo sólido criado por esta técnica como um conjunto contíguo de pontos tridimensionais, porém como não é possível enumerar todos os pontos de um objeto, enumera-se pequenos sólidos contidos parcialmente ou totalmente no objeto. Estes sólidos são os *voxels*. O tipo mais comum de *voxel* é o cubo, e a representação de espaço como uma matriz regular de cubos é chamada de *cuberille* (Foley, 1990). Os cubos são de tamanho uniforme e possuem a mesma orientação, ou seja, eles formam uma subdivisão regular do espaço tridimensional com o qual se está trabalhando. 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 um objeto é representado através da enumeração por ocupação espacial, controla-se somente a presença ou ausência 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 ocupação espacial pode ser composto por uma lista de todos os cubos que são considerados como constituintes do objeto. É fácil determinar se um cubo está dentro ou fora do sólido, e se dois objetos são adjacentes.

A enumeração por ocupação espacial é um esquema de modelagem que representa não somente a parte ocupada pelo material, mas a parte não ocupada também. Sua utilização pode acelerar operações em outras formas de representação, além de ser freqüentemente utilizada em aplicações biomédicas para representar dados volumétricos obtidos de fontes como tomografias axiais computadorizadas.

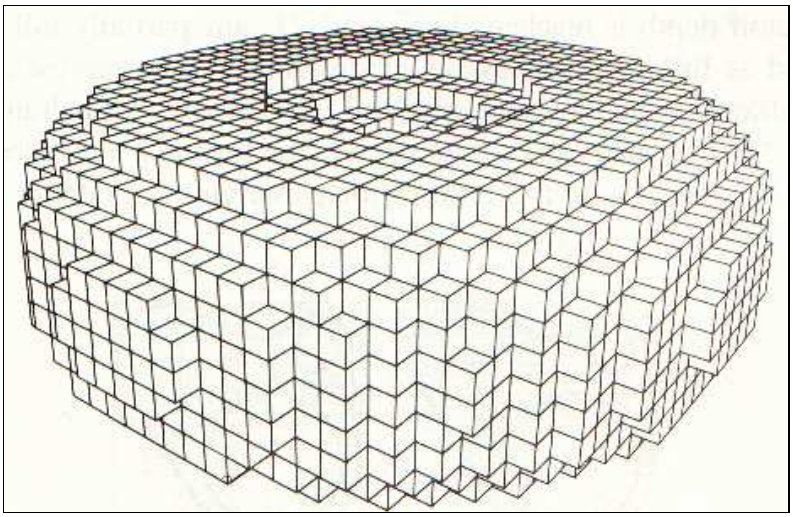
Apesar de suas vantagens e importantes aplicações, a ocupação espacial é obviamente uma técnica aproximativa. Isto significa que superfícies que não são coplanares com qualquer dos planos coordenados, não poderão ser representadas com perfeição, somente de forma aproximada. Entretanto, qualquer objeto pode ser aproximadamente representado por este

esquema. As falhas na visualização dos modelos são similares às da representação de formas 2D em *bitmaps* de 1 *bit* por *pixel*. Não há conceito de ocupação parcial. Se as células são cubos, os únicos objetos que podem ser representados com exatidão são aqueles cujas faces são paralelas aos lados dos cubos e cujos vértices caem exatamente em cima da grade.

Uma forma de diminuir o problema da representação aproximada é aumentar a resolução do modelo. As células podem ser em princípio tão pequenas quanto o necessário para aumentar a precisão da representação. O tamanho da célula determina a resolução e o espaço necessário para o armazenamento de um objeto. Portanto, quanto maior for a resolução, maior será o espaço de armazenamento exigido e o tempo de processamento para a representação do objeto.

Pode-se notar na figura 4 que a representação do objeto é bastante aproximativa, nestes casos dá-se mais importância à velocidade e ao espaço de armazenamento do que às características geométricas do objeto. Na maioria dos casos os objetos criados através desta técnica são utilizados em aplicações onde não se exige grande precisão na representação da geometria do modelo, e a maior preocupação é a rápida construção e visualização de imagens.

FIGURA 4 - OBJETO REPRESENTADO POR OCUPAÇÃO ESPACIAL



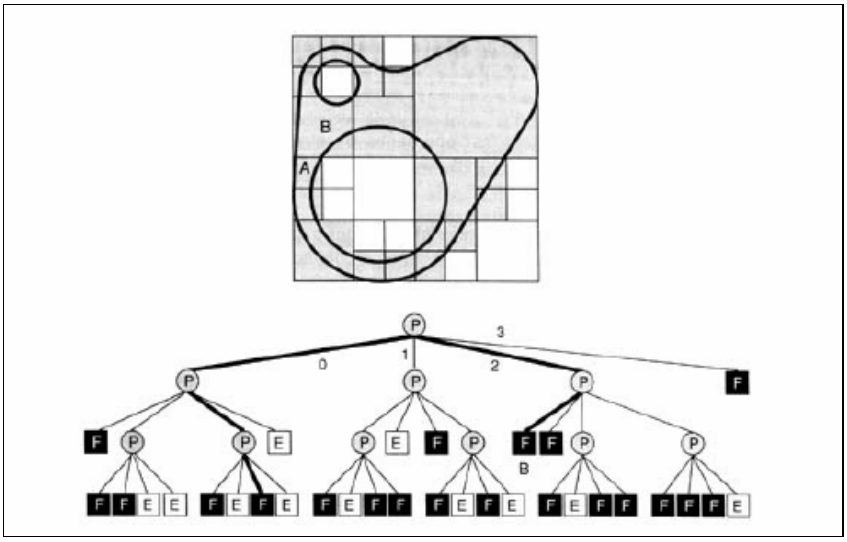
FONTE: Foley (1990)

2.3.2.2 OCTREES

Octrees são uma variação hierárquica da enumeração por ocupação espacial, concebida para combater a grande demanda de espaço de armazenamento daquela representação. Ela é derivada da *quadtree*, um formato para codificar imagens.

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 5.

FIGURA 5 - OBJETO UTILIZANDO A REPRESENTAÇÃO *QUADTREE* E SUA ESTRUTURA DE DADOS



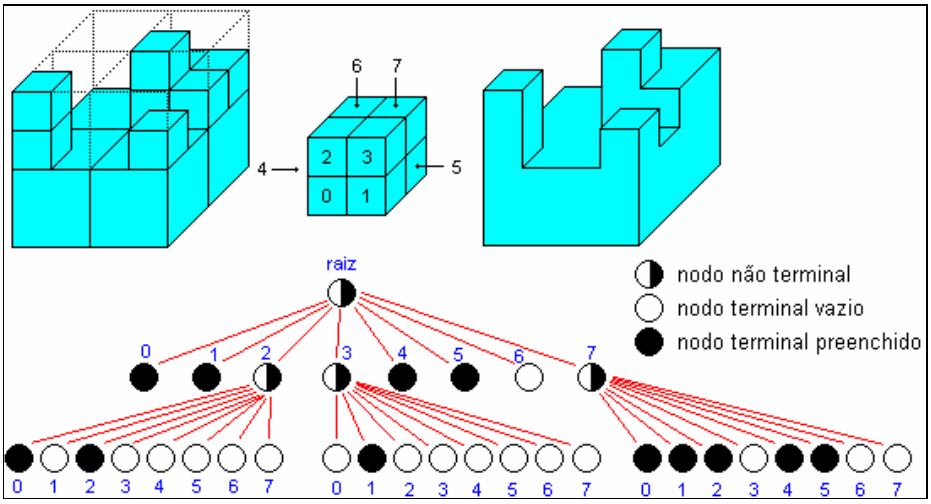
FONTE: Foley (1990)

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 (Foley, 1990).

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 6 mostra a representação por *octree* e sua estrutura de dados.

FIGURA 6: REPRESENTAÇÃO POR *OCTREE* E SUA ESTRUTURA DE DADOS



FONTE: Casacurta (1999)

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 possuir os mesmos tipos de estado da *quadtree*: cheio, parcialmente cheio ou vazio, representando, respectivamente:

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

A propriedade essencial da *octree* é que ela 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 clara e completa.

Assim como a ocupação espacial, a *octree* é uma técnica de representação aproximada, de modo que o modelo é somente semelhante ao objeto real, porém nesta técnica os *voxels* passam a ser cubos de dimensões variáveis. Também toda representação *octree* pode ser

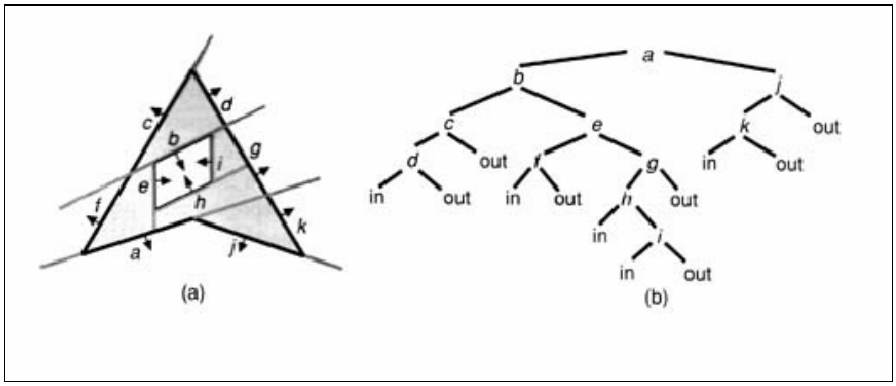
considerada válida, pois parte-se de um conjunto de primitivas sólidas que podem ou não estar representadas na estrutura de representação.

2.3.2.3 ÁRVORE BSP

A árvore BSP (*Binary Space-Partitioning Tree*), também chamada de árvore de decomposição espacial, divide o espaço recursivamente em pares de subespaços separados por um plano de orientação e posição arbitrária (Foley, 1990).

Cada nodo interno da árvore BSP é associado a um plano e tem dois ponteiros filhos, um para cada lado do plano. Assumindo que as normais apontam para fora de um objeto, o nodo esquerdo representa o filho da esquerda, atrás ou dentro do plano e o nodo direito representa o filho da direita, à frente ou fora do plano. Se o subespaço de um dos lados do plano é subdividido novamente, então o filho é a raiz de uma subárvore, caso contrário, se o subespaço é homogêneo, então este filho é uma folha da árvore representando uma região inteiramente dentro ou fora do poliedro. Estas regiões homogêneas são chamadas células *in* e *out*. A figura 7 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 7 - REPRESENTAÇÃO POR ÁRVORE BSP



FONTE: Foley (1990)

Segundo Foley (1990), a árvore BSP é umas da técnicas mais importantes para a determinação de superfícies visíveis em tempo real. Para Gomes (2002), as árvores BSP são estruturas de dados que permitem a rápida determinação de superfícies visíveis em ambientes

onde o ponto de vista é alterado e os objetos permanecem estáticos. Tais ambientes podem ser cenários de jogos ou mundos virtuais, por exemplo.

2.3.2.4 DECOMPOSIÇÃO CELULAR

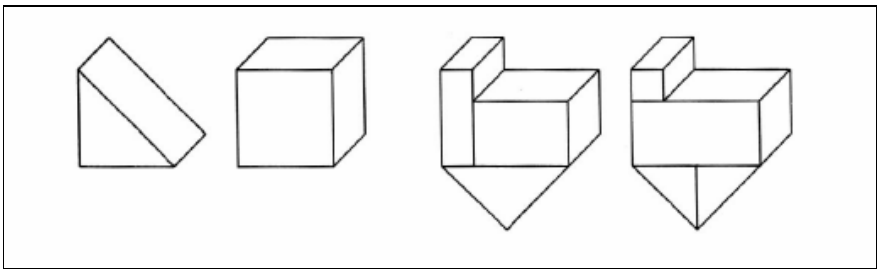
A decomposição celular (*cell decomposition*) é uma das formas mais gerais de particionamento espacial, muito utilizada para resolver os problemas da enumeração por ocupação espacial vistos anteriormente, porém preservando as suas propriedades (Souza, 2000). Esta técnica utiliza outros tipos de elementos básicos além de cubos. Cada sistema de decomposição celular define um conjunto de células primitivas que são tipicamente parametrizadas e freqüentemente curvas. A decomposição celular difere do instanciamento de primitivas que será visto posteriormente porque aqui pode-se compor objetos mais complexos a partir de outros mais simples aglutinando-os e colando-os.

A operação de colar pode ser vista como uma forma restrita de união na qual os objetos não se interseccionam. Um sólido é, portanto, modelado através de um conjunto de células onde cada célula se une à outra nas suas superfícies limitantes apesar de não possuir pontos interiores em comum. Apesar da representação por decomposição celular não ser ambígua, ela não é necessariamente única.

Decomposições celulares são difíceis de validar. Enquanto a enumeração por ocupação espacial e a representação por *octree* possuem propriedades que garantem a sua validade, a decomposição celular em geral, é somente um conjunto desordenado de células, de modo que é necessário testar todos os pares de células para verificar a sua conexão. Normalmente é muito difícil de criar um objeto através de decomposição celular, de modo que da mesma forma que os modelos por ocupação espacial e por *octree*, os modelos são criados a partir de uma conversão de um outro modelo. Apesar disso, a decomposição celular é uma representação importante para uso em Análise de Elementos Finitos.

Pode-se observar na figura 8 duas primitivas definidas por um esquema de modelagem por decomposição celular, a primeira do tipo pirâmide e a segunda do tipo cubo. Com a possibilidade de modificar as dimensões destas primitivas, pode-se obter diversas formas de combinar estas primitivas para a geração do modelo final, como representado no exemplo.

FIGURA 8 - DECOMPOSIÇÃO CELULAR



FONTE: Foley (1990)

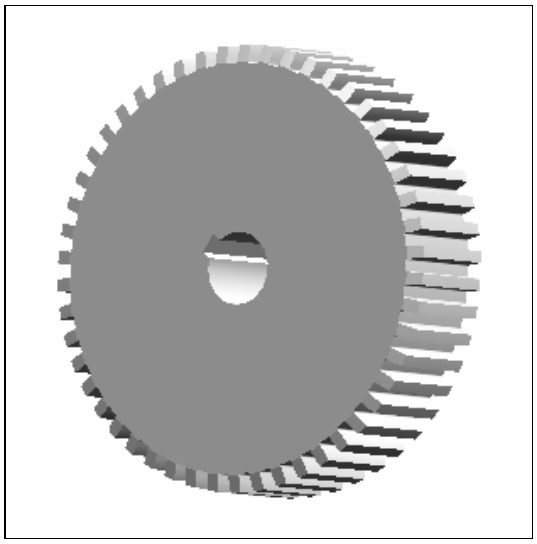
2.3.3 INSTANCIAMENTO DE PRIMITIVAS

Segundo Foley (1990), no caso da técnica de instanciamento de primitivas 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 (rotação, escala, etc) mas também em termos de outras propriedades relevantes à primitiva.

Por exemplo, uma primitiva de um objeto pode ser uma pirâmide regular com um número de faces definido pelo usuário. Instanciamento de primitivas é usado 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 primitiva parametrizada pode ser vista como a definição de uma família de peças cujos membros variam em poucos parâmetros. O exemplo mais comum de uma primitiva é a engrenagem, que pode ser parametrizada por diâmetro e número de dentes como mostra a figura 9.

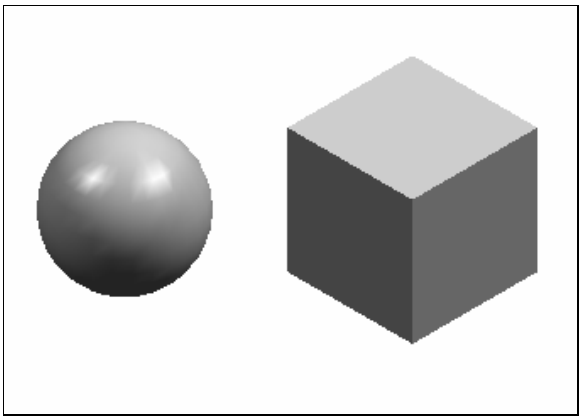
A engrenagem é um exemplo clássico de primitiva sólida utilizada em instanciamento de primitivas por ser um objeto utilizado muito especificamente para uma determinada área, neste caso a mecânica, e como existem várias padronizações e normas internacionais que definem tipos e dimensões de engrenagens, fica fácil parametrizá-las.

FIGURA 9 - PRIMITIVA DO TIPO ENGRENAGEM



Em geral, as primitivas geométricas que darão forma as primitivas parametrizadas são objetos simples de descrever e representar, constituindo os blocos básicos da construção de modelos. A figura 10 mostra um exemplo das primitivas mais simples de serem descritas: a primeira é a esfera, que pode ser descrita pelo seu centro e raio, e a segunda é o cubo, descrito pelo seu centro e dimensão de suas arestas.

FIGURA 10 - PRIMITIVAS GEOMÉTRICAS



As primitivas podem sofrer algumas transformações. Tais transformações são utilizadas tanto para posicionar as primitivas no espaço quanto modificar sua geometria. 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. O uso de transformações para modificar a geometria das primitivas permite reduzir o número de primitivas.

2.3.4 SWEEP

Segundo Foley (1990) o sweep é uma maneira natural e intuitiva de construir uma variedade de objetos. Através da técnica de modelagem por *sweep*, define-se um sólido pelo movimento de um objeto ao longo de uma trajetória no espaço.

O tipo mais comum de *sweep* é definido por uma área 2D arrastada ao longo de um caminho linear normal ao plano da área para criar um volume. Isto é conhecido como *sweep* translacional, técnica de varredura linear, ou extrusão. Na indústria, a extrusão é um processo de construção de peças onde a matéria prima (metal, plástico, etc) é estrudada através de uma matriz vazada com o perfil que se deseja dar a peça. Numa analogia a técnica de *sweep*, o perfil ou furo da matriz é a área 2D e o sólido é o volume criado pelo material que flui através desta.

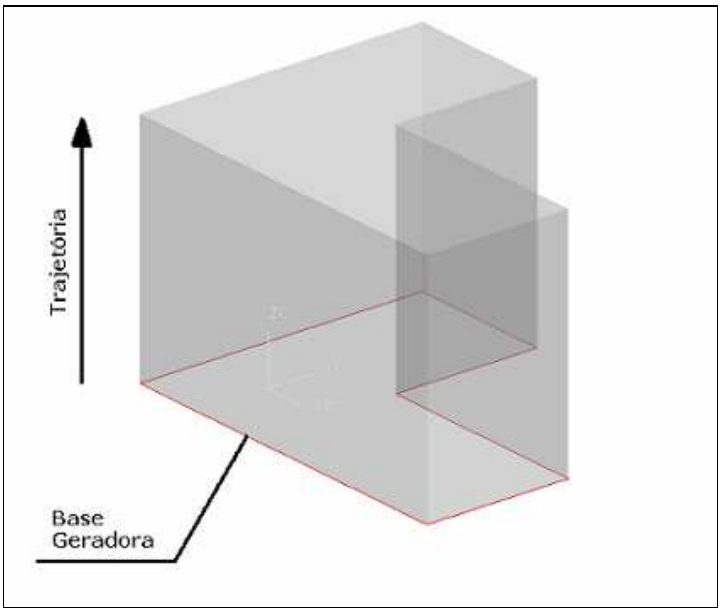
Existem outras variações da técnica de modelagem por *sweep*, a mais conhecida dentre elas é a modelagem por *sweep* rotacional. Outras técnicas incluem *sweep* cônico e com torção. Em todas elas, é necessário que se defina dois ingredientes básicos para a geração de modelos sólidos válidos: um objeto a ser movido e uma trajetória descrevendo o seu movimento. Um objeto pode ser uma face ou superfície, uma curva, ou um outro sólido. A trajetória é um caminho no espaço definido analiticamente. As diversas variações da técnica de *sweep* serão descritas com mais detalhes a seguir.

2.3.4.1 SWEEP TRANSLACIONAL

É a forma mais comum de *sweep*, representada pelo deslocamento de uma área 2D por uma trajetória normal ao plano desta área para gerar um modelo sólido (figura 11). Como característica deste modelo pode-se observar que, pelo deslocamento através de uma trajetória normal ao plano da área primitiva, as faces superior e inferior do modelo sempre serão paralelas e idênticas, e as faces laterais estarão dispostas a 90° da base. Esta é uma técnica muito implementada por modeladores de sólidos pela facilidade de se definir uma área 2D

para depois gerar um sólido, e pela facilidade por parte do usuário de assimilar este conceito de geração de modelos sólidos com os próprios processos de fabricação de artefatos nas indústrias.

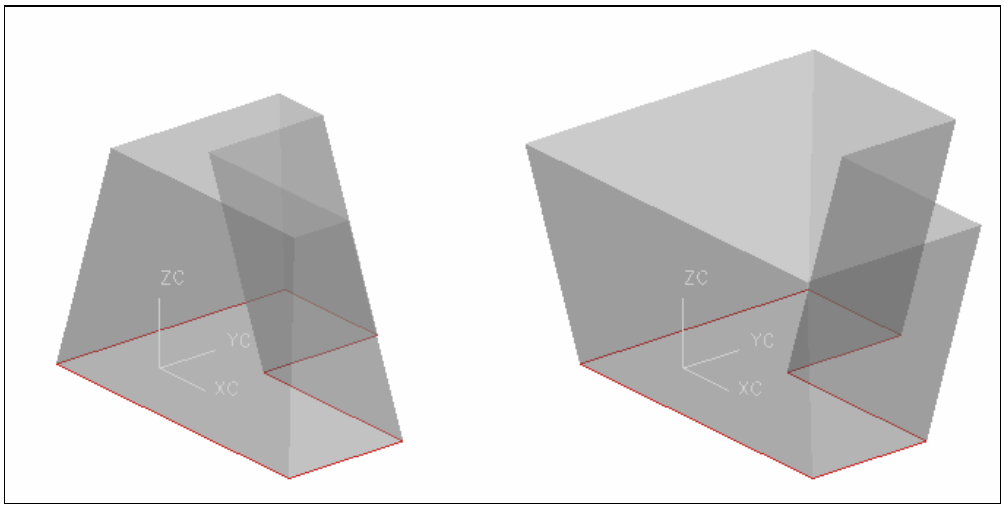
FIGURA 11 - OBJETO DEFINIDO POR *SWEEP* TRANSLACIONAL SIMPLES



Nem sempre se quer obter modelos como os da figura 11, por isso pode-se definir também sólidos através de uma trajetória e um ponto de fuga. Um ponto de fuga pode definir uma trajetória diferente de 90° pela qual pode-se deslocar uma área e formar um sólido, ou definir um ponto para onde as faces do modelo irão divergir ou convergir.

A figura 12 apresenta dois modelos sólidos definidos por uma mesma área 2D deslocada no espaço de formas diferentes. O primeiro representa um sólido gerado pelo deslocamento perpendicular ao plano normal da área, porém com um ponto de fuga que determina a convergência das faces laterais do modelo. O segundo representa também uma trajetória perpendicular, porém com um ponto de fuga negativo que determina a divergência das faces laterais em relação ao plano normal. Pode-se notar que nestes dois exemplos as faces superior e inferior são paralelas, porém não idênticas. Vale lembrar que um ponto de fuga pode determinar que a face superior seja reduzida a um único vértice, neste caso gera-se uma pirâmide.

FIGURA 12 - MODELOS SÓLIDOS GERADOS POR *SWEEP*



Os parâmetros como trajetória, ângulo de divergência e ângulo de convergência definidos para geração de modelos sólidos através destas técnicas especiais de *sweep* translacional são definidas de acordo com a interface disponibilizada pelos modeladores de sólidos.

Uma forma especial de modelagem por *sweep* translacional também encontrada é o *sweep* translacional com torção. Neste caso uma área 2D é submetida a uma trajetória e sofre uma rotação de sua base geradora. Para definir um modelo sólido por esta técnica geralmente são necessários parâmetros como altura total da translação, incremento de torção (em graus) e o incremento de translação. A cada incremento na translação é executada uma rotação (Casacurta 1999).

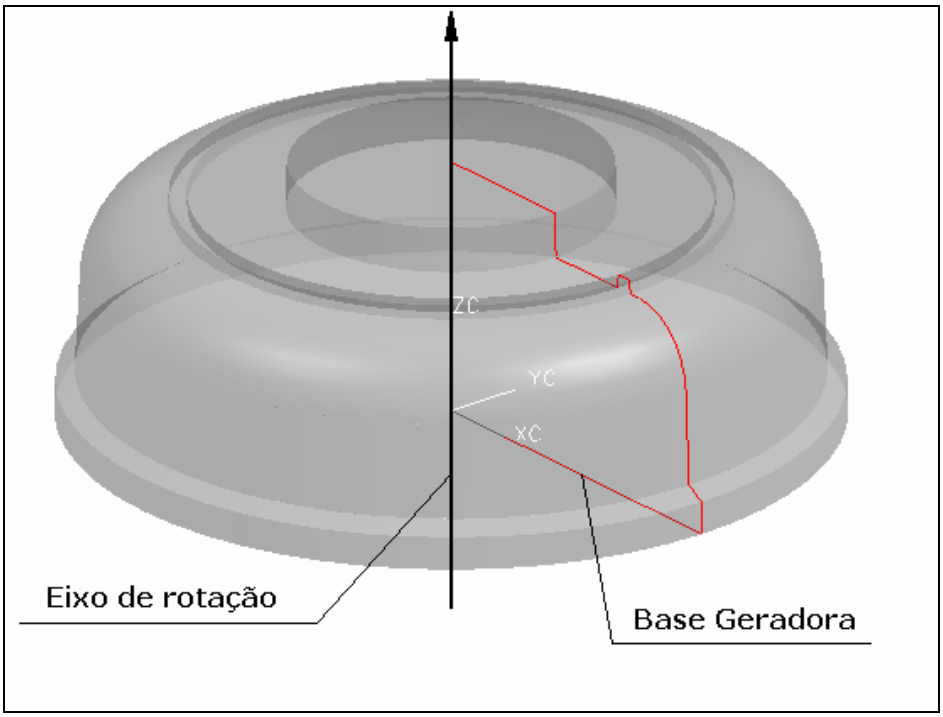
2.3.4.2 SWEEP ROTACIONAL

A técnica de *sweep* rotacional consiste em definir um sólido através da rotação de uma área sobre um determinado eixo. Existem muitas variações desta técnica, pois pode-se definir uma área 2D fechada ou aberta, e uma rotação entre 0° e 360°. A combinação destas variáveis juntamente com a definição do eixo de revolução pode gerar diversos tipos de sólidos.

Geralmente os modelos sólidos gerados por esta técnica são sólidos de revolução (figura 13), onde define-se uma área 2D fechada, um eixo de revolução e uma rotação de

360°. Outros tipos de modelos sólidos podem ser gerados através da revolução de faces ou arestas com um incremento angular, neste caso pode-se obter um objeto com um número *n* de faces. Porém a técnica mais aconselhada para gerar estes tipos de objetos é o *sweep* translacional, visto anteriormente.

FIGURA 13 - SÓLIDO DE REVOLUÇÃO GERADO POR SWEEP ROTACIONAL



2.3.5 CSG – CONSTRUCTIVE SOLID GEOMETRY

Segundo Foley (1990), na técnica CSG primitivas simples são combinadas através de operadores booleanos normalizados, que são incluídos diretamente na representação. Um objeto é armazenado como uma árvore com operadores nos nodos internos e primitivas simples nas folhas. Alguns nodos representam operadores booleanos, enquanto outros executam transformações como rotação, translação e escala.

A grande vantagem desta técnica é o poder de definição de sólidos complexos a partir de sólidos regulares simples. Sabe-se que objetos complexos são difíceis de modelar, porém quando se parte de uma primitiva simples e a partir desta são realizadas operações envolvendo

outras primitivas simples, consegue-se chegar mais facilmente ao resultado final, ou seja, o objeto modelado com todos seus detalhes e características. Além disso, através da técnica CSG pode-se obter modelos sólidos a partir de operações envolvendo outros modelos existentes, isto é muito utilizado nas indústrias quando depois de definido um produto, necessita-se a partir de sua geometria, criar ferramentas para que a fabricação do produto final se torne possível.

2.3.5.1 PRIMITIVAS GEOMÉTRICAS

As primitivas geralmente utilizadas na técnica CSG são sólidos simples, tais como cubo, esfera, cilindro e cone. A utilização de sólidos simples garante que as combinações entre estes tipos de primitivas vão gerar um modelo sólido válido.

Primitivas também podem incluir sub-espaços, que por si só não são sólidos válidos. Por exemplo, um cubo pode ser definido como a interseção de seis sub-espaços, ou um cilindro finito pode ser podado em cima e embaixo por planos e formar um cilindro infinito. Usar sub-espaços introduz o problema da validade, já que nem todas as combinações irão gerar sólidos válidos. Sub-espaços são úteis no entanto, para operações tais como cortar ou fatiar um objeto por um plano. Esta operação pode também ser realizada por primitivas sólidas utilizando somente a face de um sólido, porém a sobrecarga de processamento e armazenamento é bem maior, pois a operação é feita em todo o objeto, mesmo que apenas uma das faces seja de interesse (Foley, 1990).

Para posicionar as primitivas no espaço e alterar suas dimensões, são utilizadas transformações geométricas.

2.3.5.2 TRANSFORMAÇÕES GEOMÉTRICAS

Para posicionar primitivas no espaço, são utilizadas transformações geométricas como translação e rotação, e para modificar a sua geometria é utilizada a mudança de escala (Foley, 1990).

A translação permite posicionar uma primitiva no espaço de acordo com a utilização desta para a geração do modelo final. Geralmente as translações acontecem através do posicionamento por coordenadas aplicadas sobre os eixos x, y e z. Interfaces implementadas

por modeladores comerciais permitem ao usuário posicionar uma primitiva sólida através destas coordenadas e através de sub-funções onde se pode determinar um ponto de origem e um destino, um vetor e um deslocamento, etc.

Transformações geométricas do tipo rotação permitem que primitivas sejam posicionadas em um ângulo diferente daquele em que são criadas. Interfaces implementadas por sistemas de modelagem de sólidos podem facilitar o trabalho de posicionamento das primitivas no espaço fornecendo sub-funções como rotação em tempo real.

A mudança de escala permite que a partir de uma primitiva obtenha-se várias outras primitivas de diferentes dimensões. Mudanças de escala podem ser realizadas mantendo-se as proporções da primitiva original, ou podem ser aplicadas somente a partir de um eixo, podendo assim obter-se um paralelepípedo das mais diversas dimensões a partir da mudança de escala aplicada sobre um dos eixos de uma primitiva do tipo cubo.

As transformações geométricas foram incorporadas ao esquema CSG, pois facilitam o trabalho do projetista na construção de modelos sólidos, além de agilizar a construção de modelos e diminuir o número de primitivas implementadas e disponibilizadas pelos sistemas de modelagem de sólidos.

Após criar e posicionar devidamente as primitivas no espaço, a técnica CSG se utiliza das operações booleanas para combinar as diversas primitivas e criar o modelo sólido final.

2.3.5.3 OPERAÇÕES BOOLEANAS

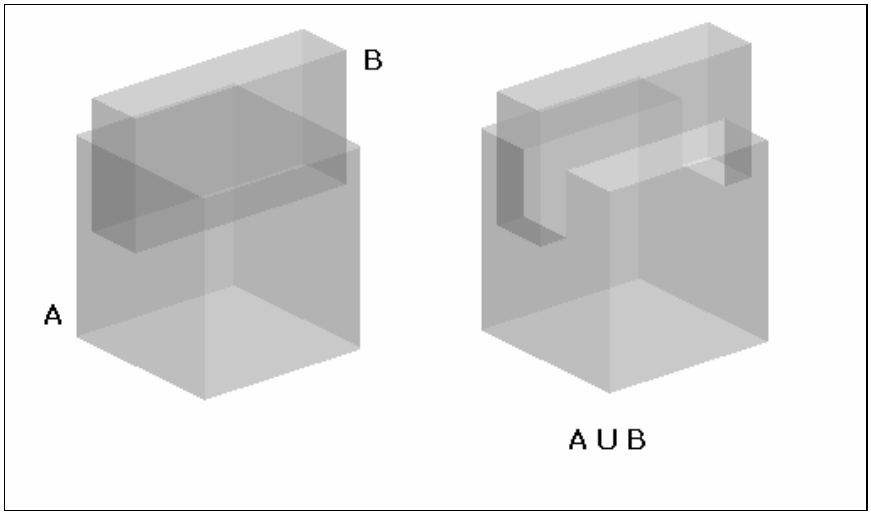
George Boole (1815-1864) inventou a álgebra utilizada para a combinação de conjuntos. Enquanto as operações algébricas são adição, subtração, e multiplicação, na Álgebra Booleana, as operações são união, interseção e diferença, e são chamadas de operações booleanas (Mortenson, 1999).

Em CSG, as operações booleanas são utilizadas para combinar primitivas sólidas ou modelos sólidos.

A operação de união, combina o volume de dois sólidos incluindo suas partes comuns ou intersecionadas, para gerar um único volume que representa um novo modelo sólido. A união pode ser descrita como um conjunto *C* cujos elementos são todos de *A* ou *B*, ou de

ambos, o que pode ser expresso por $C = A \cup B$. É importante observar que não existe repetição de elementos em *C*, correspondendo ao operador lógico "ou inclusivo" (Mortenson, 1999).

FIGURA 14 - OPERAÇÃO DE UNIÃO



A operação de interseção define um novo modelo sólido através das partes comuns ou intersecionadas de dois sólidos. O novo objeto é formado então pelo volume determinado pelas partes intersecionadas dos sólidos submetidos a esta operação. A interseção é então um conjunto *D* contendo os elementos comuns entre *A* e *B*, expressa por $D = A \cap B$, significando o conjunto sem repetição dos elementos que se acham em ambos os conjuntos *A* e *B* (Mortenson, 1999). O resultado da operação de interseção aplicada sobre dois modelos sólidos está representado na figura 15.

A operação de diferença subtrai do volume de um sólido determinado, o volume de outro sólido. A diferença pode ser descrita como o conjunto de elementos de *A* que não são elementos de *B*, ou seja $A - B$. Se o complemento (*c*) é formado em relação ao conjunto universo *E* contendo os conjuntos *A* e *B*, então $A - B = A \cap cB$ (Mortenson, 1999). A figura 16 ilustra a operação de diferença.

FIGURA 15 - OPERAÇÃO DE INTERSEÇÃO

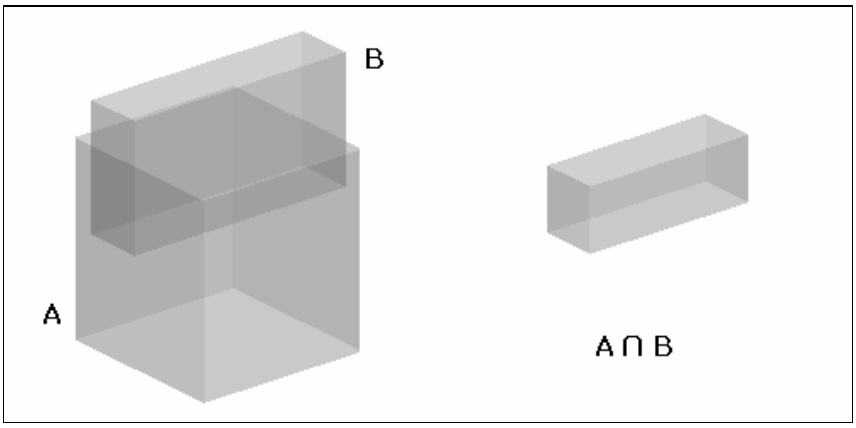
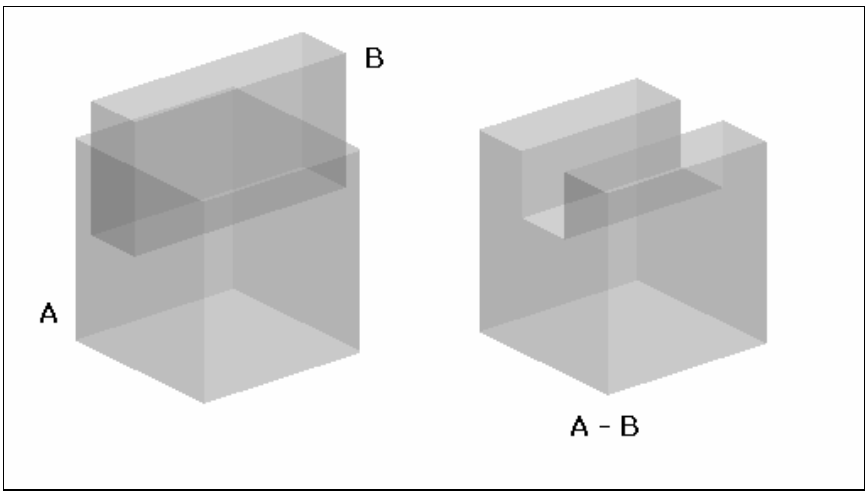


FIGURA 16 - OPERAÇÃO DE DIFERENÇA OU SUBTRAÇÃO

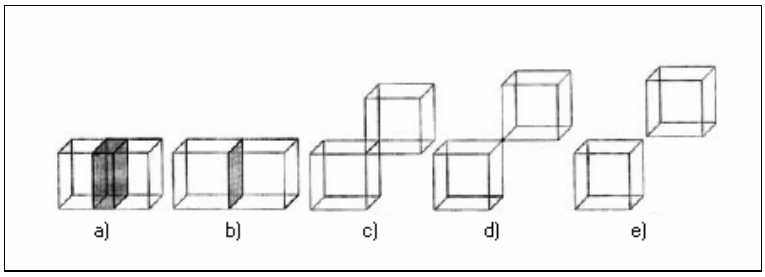


Nem sempre a aplicação das operações booleanas vistas acima resulta em modelos sólidos válidos. Algumas operações podem resultar em além de objetos sólidos, um plano, uma linha, um ponto e até mesmo um conjunto vazio. Este efeito é extremamente indesejável e existem diversas implementações capazes de tratar estas exceções. Nos sistemas de modelagem de sólidos encontrados no mercado, o tratamento destas situações está implementado pelo núcleo de geometria do sistema, que fornece todo o gerenciamento do modelo sólido. Além das diferenças entre cada núcleo no tratamento destas situações, cada sistema oferece ao usuário uma interface com as ações a serem tomadas em algum destes

eventos. Apesar de geralmente indesejável, alguns sistemas permitem que se realize operações como por exemplo a interseção de dois sólidos onde o resultado gera um plano, pois apenas as faces dos modelos coincidem no espaço. A maioria dos sistemas, porém, não permitem tais operações pelo risco de se criar modelos inválidos, que muitas vezes somente são percebidos ao final do trabalho de construção do modelo.

A figura 17 demonstra diversos resultados que podem ser obtidos da interseção de dois modelos sólidos posicionados de formas diferentes no espaço. Estes resultados podem ser um sólido (a), um plano (b), uma linha (c), um ponto (d) ou um conjunto vazio (e).

FIGURA 17 - INTERSEÇÃO ENTRE CUBOS



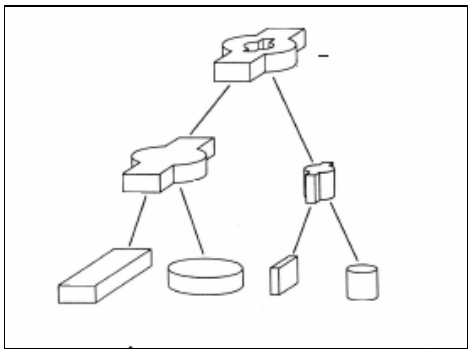
FONTE: Silva (1999)

2.3.5.4 ÁRVORE CSG

A estrutura utilizada para representar a técnica CSG é dita árvore CSG. A árvore CSG é uma árvore binária ordenada, onde a raiz representa o modelo sólido final, e cada nodo possui no máximo dois filhos que representam as primitivas sólidas (Silva, 1999).

A figura 18 mostra a estrutura de dados chamada árvore CSG. Como pode-se observar, as folhas (na base da árvore) representam as primitivas sólidas e os nodos que ligam as folhas representam as operações booleanas, e podem também representar transformações geométricas. Percorrendo-se a árvore a partir da base em direção ao topo, tem-se uma sub-árvore, onde os nodos que representam a operação sofrida pelas primitivas passam a ser folhas ligadas a um novo nodo que representa uma operação entre os novos modelos criados, e assim sucessivamente até chegar ao topo da árvore que representa o modelo sólido final.

FIGURA 18 - A ÁRVORE CSG



Adaptado de Silva (1999)

De acordo com Foley (1990) a técnica CSG é uma das formas dominantes de modelagem de sólidos. Isto pode ser comprovado ao observar que os principais modeladores de sólidos apresentam esta técnica como uma vantagem oferecida pelo sistema para a simplificação e agilidade na criação de modelos sólidos. A técnica de modelagem CSG combinada a conceitos modernos de modelagem, como parametrização e modelagem baseada em *features*, que serão abordadas a seguir, permite a construção de poderosos sistemas de modelagem de sólidos utilizados hoje nas mais diversas e importantes áreas de tecnologia.

2.3.6 MODELAGEM SÓLIDA PARAMÉTRICA

A modelagem paramétrica foi incorporada à modelagem sólida para facilitar o trabalho de projetistas e engenheiros. A modelagem sólida paramétrica permite que se crie modelos de produtos com dimensões variacionais (Kerry, 1997). Isto significa que na criação de um modelo sólido, o projetista não necessita definir e informar todos os parâmetros do modelo, uma vez que depois de parametrizado, este apresenta uma série de informações definidas em função dos parâmetros que realmente necessitam ser definidos pelo projetista.

As dimensões do modelo podem ser ligadas através de expressões. Ligações bidirecionais entre o modelo e o esquema de dimensionamento permite a regeneração automática de modelos depois de mudanças nas dimensões e atualização automática das dimensões relacionadas. As expressões utilizadas para criar estas ligações são equações, e podem ou não ser bidirecionais.

Como exemplo de parametrização unidirecional e bidirecional pode-se imaginar um eixo escalonado, onde deseja-se parametrizar o diâmetro menor do eixo. Considera-se que por motivos de padronização, seja interessante ao projetista que o diâmetro menor do eixo seja a metade do diâmetro maior. No caso de uma parametrização unidirecional, pode-se definir que os parâmetros para criação do eixo sejam o comprimento e o diâmetro maior, assim o diâmetro menor passa a ser parametrizado em função do valor do diâmetro maior. No caso de uma alteração nos parâmetro do modelo, qualquer alteração no valor do diâmetro maior será automaticamente refletida no valor do diâmetro menor. Nota-se que neste caso, o valor do diâmetro menor não está explícito para o usuário, se for interessante a definição de qualquer um dos dois diâmetros para a criação do modelo, necessita-se implementar a bidirecionalidade.

Neste caso, define-se diâmetro menor (de) como: de = da/2, e diâmetro maior (da) como sendo a inversa da função de ligação: da = de * 2. Na criação do modelo qualquer dos dois valores informados pode definir as dimensões do eixo. Na alteração de algum destes valores, faz-se necessário a atualização das dimensões relacionadas para a regeneração do modelo.

Neste exemplo, a implementação da bidirecionalidade é bastante simples, porém nem todos os sistemas CAD paramétricos provêem esta bidirecionalidade, devido a complexidade que isto envolve. No caso de modelos complexos, a ligação bidirecional entre parâmetros requer a utilização de algoritmos complexos para garantir a consistência do modelo.

Uma outra forma de implementar a parametrização é a utilização de restrições. As restrições ajudam a controlar a geometria do modelo, elas são utilizadas principalmente para delimitar os valores máximos ou mínimos dos parâmetros. Assim as dimensões e restrições paramétricas controlam a geometria do modelo, e a medida em que são feitas alterações no projeto, assim como nas restrições e dimensões, o modelo e os desenhos são atualizados automaticamente.

Os sistemas CAD que permitem a modelagem sólida paramétrica, contribuem significamente para a diminuição do tempo de ciclo do projeto. Por exemplo, permitem projetos concorrentes nos quais membros da equipe de projeto, membros das divisões de

manufatura e *marketing* da empresa trabalhem juntos e ao mesmo tempo, para encontrar uma solução global para o projeto (Kerry, 1997).

A grande vantagem dos modelos parametrizados é que, a partir do momento de sua definição, a alteração de qualquer parâmetro pode ser rapidamente efetuada e refletida na produção do novo produto com as alterações necessárias.

2.3.7 MODELAGEM BASEADA EM *FEATURES*

Uma *feature* pode se definida como um elemento físico de um objeto que tem algum significado para a aplicação na qual está inserida (Kerry, 1997). As *features* são utilizadas principalmente em sistemas CAD/CAM. De forma independente da área de aplicação, as *features* são também denominadas *form features*. Nos sistemas CAD, pode-se encontrar a denominação *design features*, e nos sistemas CAM, *manufacturing features*.

A modelagem por *features* encontra grande espaço principalmente dentro da engenharia mecânica. O método permite criar furos, chanfros, rasgos, etc, para serem associados com outras entidades ou faces. A modelagem por *features* é baseada na idéia de se desenhar utilizando *building blocks* - blocos de construção. Ao invés de se usar formas analíticas como paralelepípedos, cilindros, esferas e cones como primitivas, o usuário cria o modelo usando primitivas de maior nível que são mais relevantes para sua aplicação específica (Kerry, 1997).

Neste caso, a interface do sistema baseado em *features* fica muito mais amigável ao usuário, uma vez que ele utiliza elementos de construção de mais alto nível com nomenclaturas que podem ser mais facilmente associadas aos modelos sólidos criados.

Esta abordagem traz o problema de o usuário dispor de um conjunto fixo de *features* implementadas pelo sistema, o que diminui o poder de definição dos modelos. Por isso as *features* são utilizadas pelos sistemas juntamente com outras técnicas de modelagem de sólidos.

Uma *feature* pode representar não apenas uma informação geométrica do objeto, como um furo ou um chanfro, mas também uma informação relativa a produção que necessita ser especificada no projeto. Este tipo de *feature* é utilizada principalmente por sistemas CAM.

Nelas estão contidas informações como tolerâncias, acabamento de superfícies, etc. A grande vantagem na utilização das *manufacturing features* é que suas informações não necessitam ser especificadas em folhetos de produção ou desenhos 2D, e este é o elo de ligação para a integração totalmente informatizada entre CAD/CAM, ou projeto e produção.

2.3.8 OUTRAS CONSIDERAÇÕES SOBRE MODELAGEM DE SÓLIDOS

Não se pode confundir a técnica utilizada por um sistema de modelagem para definir os modelos sólidos, com a forma com que os modelos são criados na interface do sistema. Um sistema de modelagem de sólidos pode permitir ao usuário criar objetos em termos de várias representações diferentes, enquanto armazena-os em outra. A maneira utilizada para criar um modelo pode ter alguma vantagem expressiva que faz dela uma escolha natural na criação do objeto. Por exemplo, um sistema de modelagem B-Rep pode permitir que um objeto seja definido através de um *sweep*. A interface do sistema também pode fornecer diversas maneiras de o mesmo objeto ser criado dentro da mesma representação.

O próximo capítulo apresenta o 3D ACIS *Modeller* e suas características principais.

3 ACIS

O ACIS é um núcleo de geometria para modelagem de sólidos desenvolvido pela Spatial Corporation (*http://www.spatial.com*), sobre o qual pode-se construir aplicações 3D. Escrito em C++, o ACIS fornece uma coleção de classes (incluindo dados, funções membro e métodos) e funções. Os desenvolvedores utilizam estas classes e funções para criar aplicações 3D ao usuário final.

O ACIS oferece um ambiente unificado, orientado a objetos para a modelagem de curvas, superfícies, e sólidos, e pode ser adaptado e estendido a diversas exigências particulares da aplicação devido a flexibilidade de suas funções para modelagem (Spatial Corporation, 2001).

O ACIS integra repreesentação por *wireframe*, representação por superfícies e modelagem sólida, permitindo que estas representações alternativas coexistam naturalmente através da implementação de uma hierarquia de classes C++. Assim os modelos criados podem ser representados por uma destas formas, ou uma combinação delas. Além da geometria múltipla o ACIS pode representar *nonmanifold geometry* (Corney, 2000).

Uma geometria do tipo *nonmanifold* refere-se a um modelo não sólido. Este tipo de geometria é encontrada, por exemplo, quando aplica-se operações utilizando sub-espaços e sólidos. Maiores detalhes podem ser vistos na seção 5.5.1. Este tipo de representação é possível devido a estrutura de dados utilizada pelo ACIS, que permite a representação de geometria sólida e *nonmanifold* simultaneamente pela implementação de uma hierarquia de classes. Desta forma a geometria do modelo pode ser limitada ou fechada, não fechada, e semi fechada, permitindo representar sólidos completos ou incompletos (Spatial Corporation, 2001).

Utilizando ACIS, fica-se livre da necessidade de criar núcleos de geometria próprios, acelerando o processo de desenvolvimento. Além disso, a troca de informações entre aplicações torna-se transparente, tornando desnecessária a utilização de padrões como o IGES (*Initial Data Exchange Specification*).

A estrutura de dados implementada pelo ACIS para definição da geometria dos modelos está baseada na técnica B-rep para modelagem de sólidos. Desta forma, o ACIS define modelos sólidos através das informações referentes à geometria e topologia dos modelos. Os conceitos que envolvem estas duas estruturas são abordados a seguir.

3.1 GEOMETRIA DO MODELO

A geometria refere-se aos elementos físicos representados pelo modelo, como pontos, curvas e superfícies, independente de seus relacionamentos espaciais ou topológicos. O ACIS representa a geometria dos modelos de duas formas distintas: geometria construtiva e geometria do modelo.

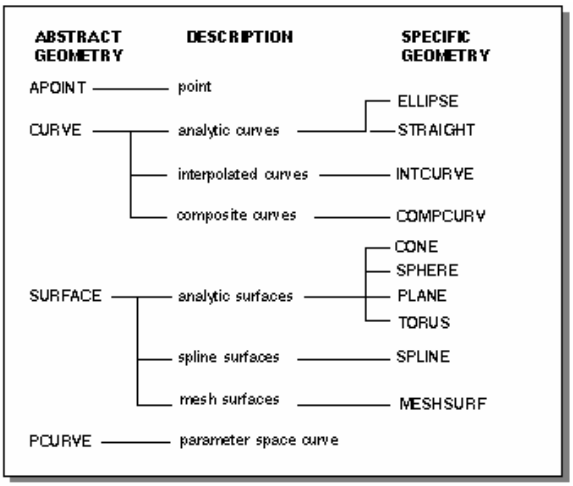
Segundo a Spatial Corporation (2001), a geometria construtiva refere-se as classes C++ que possuem as definições matemáticas dos objetos geométricos. As classes de geometria construtiva possuem nomes com letras minúsculas. A geometria do modelo refere-se as classes C++ que adicionam funcionalidades de gerenciamento do modelo para sua construção geométrica (incluem ponteiros às classes de geometria construtiva como parte de sua estrutura de dados). As classes de geometria do modelo possuem nomes com letras maiúsculas.

As classes de geometria são geralmente divididas em *abstract geometry* e *specific geometry*, ou classes de geometria abstratas e classes de geometria especificas. As classes abstratas se referem a elementos mais genéricos e implementam características comuns a todas as classes especificas correspondentes (Spatial Corporation, 2001).

Como exemplo de classe de geometria abstrata pode-se citar uma classe que represente uma curva. Neste caso as classes específicas podem definir curvas simples, representadas por fórmulas algébricas simples (elipse, círculo, etc), e curvas interpoladas, que são outras entidades geométricas geralmente representadas por equações (interseção entre duas superfícies, curvas não parametrizadas, etc).

A figura 19 demonstra as classes de geometria abstrata, sua descrição, e as classes de geometria específicas implementadas pelo ACIS.

FIGURA 19 - CLASSES DE GEOMETRIA



FONTE: Spatial Corporation (2001)

3.2 TOPOLOGIA DO MODELO

A topologia refere-se aos relacionamentos espaciais entre os vários elementos geométricos de um modelo, ou seja, ela descreve como esses elementos são conectados (Spatial Corporation, 2001).

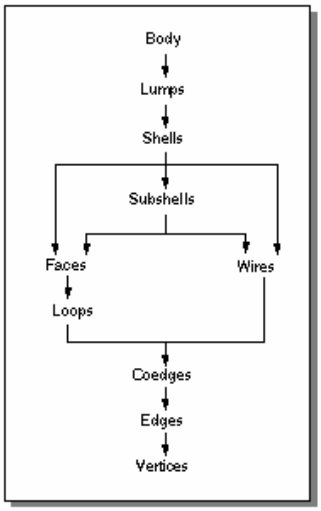
Para se diferenciar topologia e geometria do modelo, pode-se imaginar uma esfera de borracha. Deformando esta esfera de modo que ela fique de forma elíptica, não altera-se sua topologia, mas sua geometria sim.

O ACIS representa a topologia dos modelos decompondo-a em uma hierarquia. Esta hierarquia de elementos topológicos é representada na figura 20.

Body é o mais alto nível de representação do modelo, e é formado por l*umps*.

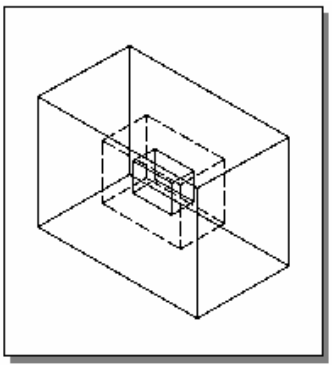
Uma *lump* representa uma região fechada do espaço, conectada por pontos. Ela é delimitada pelas s*hells*. A figura 21 representa regiões do espaço que podem definir um sólido.

FIGURA 20 – TOPOLOGIA DOS MODELOS EM ACIS



FONTE: Spatial Corporation (2001)

FIGURA 21 – *LUMPS*

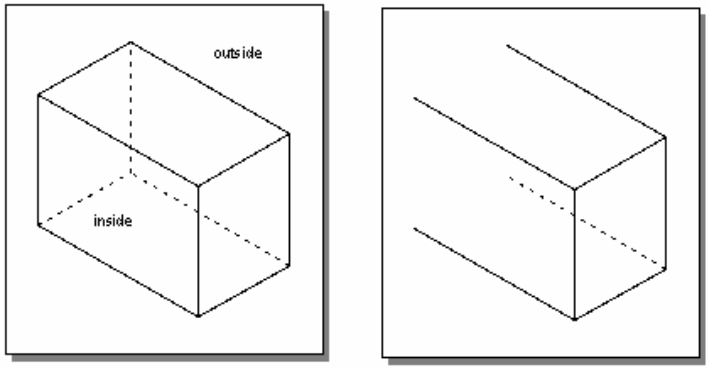


FONTE: Spatial Corporation (2001)

Shell é um conjunto de *faces* e *wires* conectados, que representam o contorno externo de um sólido. Furos ou detalhes internos dos sólidos são representados por *subshells*. As shells podem representar também *nonmanifold geometry*, como representado na figura 22.

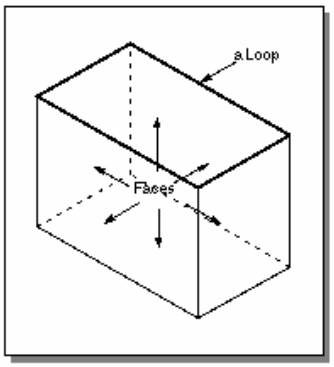
Face é uma superfície ou um uma parte dela, fechada por um conjunto de *loops*. A figura 23 demonstra uma *face* de um sólido e as *loops*.

FIGURA 22 – *SHELL* COMPLETA E INCOMPLETA



FONTE: Spatial Corporation (2001)

FIGURA 23 – *FACE*



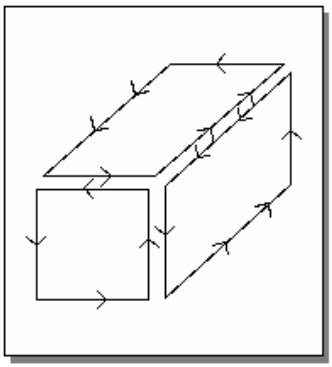
FONTE: Spatial Corporation (2001)

Uma *loop* é formada por uma série de *coedges*. Geralmente as *loops* são fechadas, não existindo ponto inicial ou final. A figura 24 ilustra este conceito.

Wire é um conjunto de *coedges* que não fazem parte de uma face, ou não formam um volume, como representado na figura 25.

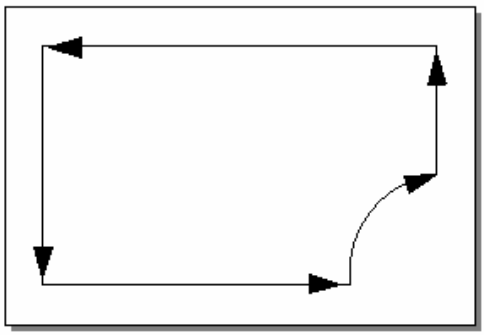
Coedge representa o uso de *edge* por uma *face*, ou seja, define qual a orientação da *edge* na face, como representado na figura 26. Pode também representar o uso de uma *edge* por um *wire*.

FIGURA 24 – *LOOPS*



FONTE: Spatial Corporation (2001)

FIGURA 25 – *WIRE*

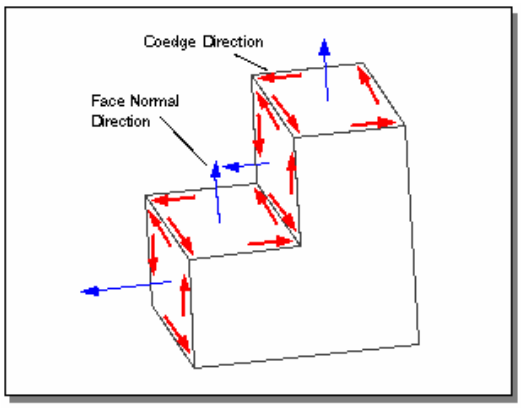


FONTE: Spatial Corporation (2001)

Edge é uma curva formada por *vertices*.

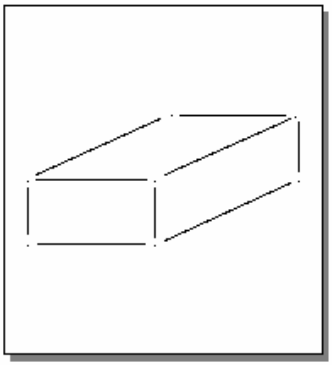
Um *vertex*, ou um vértice, pode ser definido como um canto de uma *face* ou de um *wire* (ver figura 27). Com um único vértice definindo um ponto de uma *face*, os demais podem ser encontrados pela consulta às *coedges*.

FIGURA 26 – *COEDGES*



FONTE: Spatial Corporation (2001)

FIGURA 27 – *VERTEX*



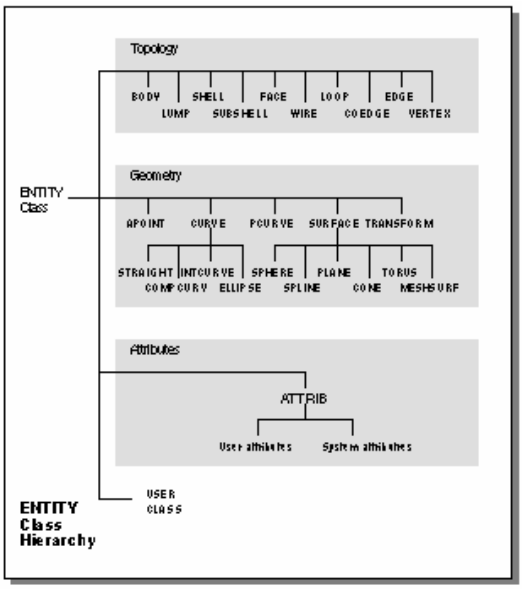
FONTE: Spatial Corporation (2001)

3.3 ENTIDADE E MODELOS DE OBJETOS

Uma entidade é o objeto mais básico do ACIS. A entidade é uma classe C++, implementada no ACIS com o nome *ENTITY*. Todas as entidades possuem um conjunto comum de funcionalidades como a habilidade em clonar, restaurar para, e de um arquivo, copiar e depurar. Todos os outros modelos de objetos de mais alto nível do ACIS são derivados da classe *ENTITY*. Um modelo de objetos é todo objeto que pode ser salvo ou recuperado de um arquivo do tipo SAT (*Standard ACIS Text*). Modelos de objetos ACIS são implementados em C++ utilizando hierarquia de classes derivada da entidade (Spatial Corporation, 2001).

A figura 28 apresenta as classes C++ derivadas da classe *ENTITY* do ACIS. Elas podem ser classes de geometria, topologia, atributos e classes definidas pelo usuário.

FIGURA 28 – HIERARQUIA DE CLASSES *ENTITY*

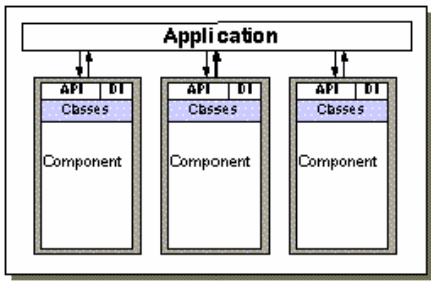


FONTE: Spatial Corporation (2001)

3.4 INTERFACE C++ / ACIS

Aplicações C++ podem utilizar interface com o ACIS através das funções API (*Application Procedural Interface*), classes C++, ou em alguns casos, através das funções DI (*Direct Interface*), que serão abordadas nas próximas seções. Pode-se também utilizar ACIS criando funções APIs e classes próprias (Spatial Corporation, 2001). A figura 29 ilustra uma aplicação C++ que utiliza interface para integração com ACIS através de funções APIs, classes, e funções DIs.

FIGURA 29 - INTERFACE DA APLICAÇÃO C++ / ACIS

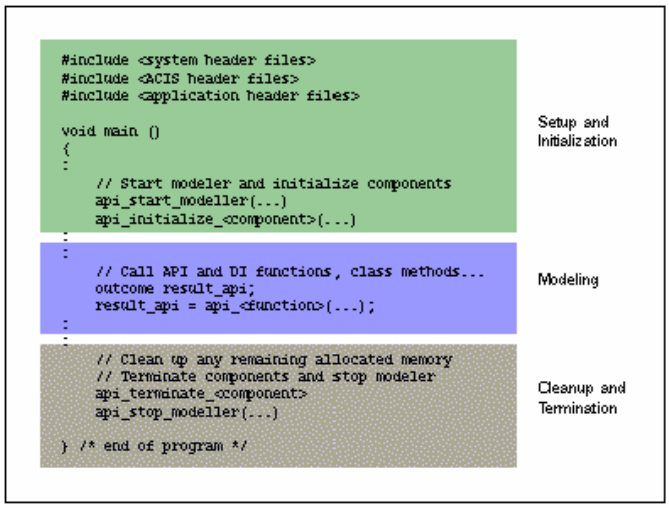


FONTE: Spatial Corporation (2001)

A mais simples aplicação C++ utilizando ACIS, pode ser dividida em três seções: *setup* e inicialização, modelagem, e *cleanup* e finalização (Spatial Corporation, 2001)

No setup e inicialização, devem estar contidos os arquivos apropriados através do *include* do C++, a inicialização do modelador, e a inicialização dos componentes de modelagem. Na área de modelagem são implementadas todas as chamadas a funções de modelagem e outras funcionalidades da aplicação. A área de finalização e *cleanup* deve fechar os componentes chamados na inicialização, parar o modelador, liberar memória alocada pela aplicação, e terminar o programa.

QUADRO 1 - LAYOUT DE UMA APLICAÇÃO C++ UTILIZANDO ACIS



FONTE: Spatial Corporation (2001)

3.4.1 FUNÇÕES API

As funções API são um conjunto de funções implementadas no ACIS para acesso as funcionalidades que ele oferece, principalmente para modelagem. As funções API combinam funcionalidades de modelagem com suporte à aplicação, como checagem de erros na execução de sua rotina (Spatial Corporation, 2001).

Quando ocorre um erro durante a execução da rotina de uma API, o ACIS automaticamente retorna o estado inicial do modelo antes da função ser chamada, isto assegura que o modelo mantenha sua consistência. As funções API possuem sempre o prefixo *api_* no nome, elas representam a principal forma de interface entre a aplicação e o ACIS.

3.4.2 FUNÇÕES DI

As funções DI fornecem acesso as funcionalidades do ACIS sem as características adicionais de suporte a aplicação oferecidas pelas APIs. Ao contrário das APIs, as funções DI não garantem a consistência do modelo após sua chamada para execução, e não oferecem

acesso a todas as funcionalidades do ACIS. São geralmente utilizadas para operações que não provocam alterações no modelo, tais como consultas diversas a informações do modelo sólido gerado (Spatial Corporation, 2001).

3.4.3 CLASSES

As classes implementadas pelo ACIS definem a geometria, topologia e outras características do modelo. As classes podem ser utilizadas pelas aplicações para interagir diretamente com o ACIS, acessando seus dados membro públicos e protegidos, e seus métodos ou funções membro. Pode-se também definir novas classes derivadas das classes do ACIS implementando-as em C++ (Spatial Corporation, 2001).

3.5 ARQUIVOS SAT

ACIS pode salvar ou recuperar informações dos modelos através de arquivos chamados ACIS *save files*, ou *part files.* Estes arquivos possuem um formato aberto, de modo que as aplicações não baseadas em ACIS possam ter o acesso ao modelo gerado através do ACIS. Vários sistemas comerciais de CAD lêem arquivos do tipo ACIS.

Existem dois tipos de arquivos ACIS: arquivos do tipo SAT (*Standard ACIS Text*), com extensão .sat, e arquivos SAB (*Standard ACIS Binary*), com extensão .sab. A única diferença entre estes arquivos, é que os dados estão armazenados como texto ASCII em arquivos SAT, e no formato binário em arquivos SAB. A organização de um arquivo SAT, e um arquivo SAB é idêntica. O termo SAT *file*, ou arquivo SAT, é usado geralmente para se referir a qualquer um dos tipos de arquivos.

No próximo capítulo a utilização do ACIS será abordada através da construção do protótipo.

4 CONSTRUÇÃO DO PROTÓTIPO

A construção do protótipo do sistema para modelagem de sólidos parametrizados foi desenvolvida em paralelo com a pesquisa bibliográfica dos assuntos envolvidos no seu desenvolvimento. Primeiramente, identificou-se as ferramentas e os métodos utilizados para sua construção. A escolha pelo 3D ACIS *Modeller* se deu ao fato de ele ser um núcleo de geometria aberto, ou seja, disponível para que desenvolvedores construam aplicações baseadas nele. Outros núcleos de geometria como o Parasolid, são proprietários, e as empresas detentoras de seus direitos, utiliza-os para desenvolvimento exclusivo de seus sistemas de modelagem de sólidos comerciais. Além disso a empresa Spatial Corporation disponibiliza uma licença educacional de seu produto à FURB. Como o ACIS é escrito em linguagem C++ utilizando Programação Orientada a Objetos, também utilizou-se estas ferramentas para o desenvolvimento do protótipo.

Os temas abordados na pesquisa bibliográfica referente a construção do protótipo são o 3D ACIS *Modeller*, o qual se destinou um capítulo deste trabalho à sua apresentação, Programação Orientada a Objetos, UML, linguagem de programação C++, e ambiente Microsoft visual C++.

4.1 PROGRAMAÇÃO ORIENTADA A OBJETOS

Orientação a Objetos é, com certeza, o maior avanço em desenvolvimento de softwares nos últimos anos. É uma forma mais natural de se analisar o mundo. Ela permite construir sistemas melhores e, além disso, de maneira mais fácil. A programação orientada a objetos foi a tecnologia emergente mais importante na área de software nos anos 90 (Bernardi, 1999).

As técnicas estruturadas obtiveram grande aceitação desde que foram lançadas no final dos anos 70. Contudo a medida que foram sendo utilizadas, a decomposição funcional mostrou-se inadequada em situações de sistemas complexos e principalmente para profissionais iniciantes. Além disso, os sistemas criados através de técnicas estruturadas ainda são difíceis de serem incrementados com novas funções e as alterações em funções existentes, muitas vezes, provocam sérios problemas em outras partes do software.

A orientação a Objetos traz vários benefícios no desenvolvimento e manutenção de software. Como vantagens diretas, pode-se citar a maior facilidade para reutilização de código e por conseqüência do projeto, a possibilidade do desenvolvedor trabalhar em um nível mais elevado de abstração, a utilização de um único padrão conceitual durante todo o processo de criação de software, maior adequação à arquitetura cliente/servidor e maior facilidade de comunicação com os usuários e com outros profissionais de informática. As vantagens reais, oferecidas pela programação orientada a objetos e observadas na prática são o ciclo de vida mais longo para os sistemas, o desenvolvimento acelerado de sistemas, possibilidade de se construir sistemas muito mais complexos, pela incorporação de funções prontas, e o menor custo para desenvolvimento e manutenção de sistemas (Bernardi, 1999).

4.2 LINGUAGEM C++

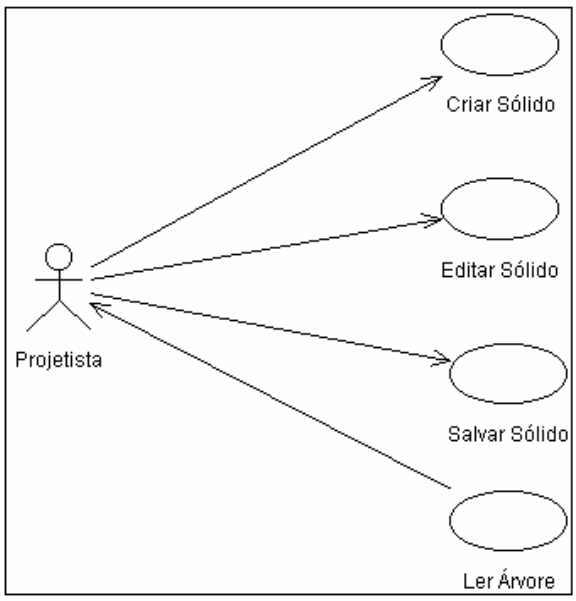
C++ é uma linguagem de programação voltada à programação orientada a objetos das mais amplamente utilizadas. Em virtude de a maioria de seus aperfeiçoamentos da linguagem C suportar a criação de classes com a mesma riqueza encontrada nos tipos de dados incorporados, a linguagem C++ suporta vários conceitos e mecanismos utilizados na programação clássica. Todos os recursos oferecidos pela linguagem C++, fizeram dela a linguagem ideal para desenvolvimento orientado a objeto de sistemas extensos. Tanto que os principais sistemas CAD, inclusive os mais avançados, encontrados no mercado, são escritos nesta linguagem.

4.3 ESPECIFICAÇÃO DO PROTÓTIPO

A especificação do protótipo foi realizada utilizando a linguagem para modelagem de sistemas UML (*Unified Modeling Language*). Como ferramenta para modelagem e especificação do protótipo, utilizou-se o Rational Rose versão 4.0 Demo. Seguindo a linguagem UML, são apresentados os principais diagramas que compõem a especificação do sistema.

A figura 30 apresenta o diagrama de casos de uso.

FIGURA 30 - DIAGRAMA DE CASOS DE USO

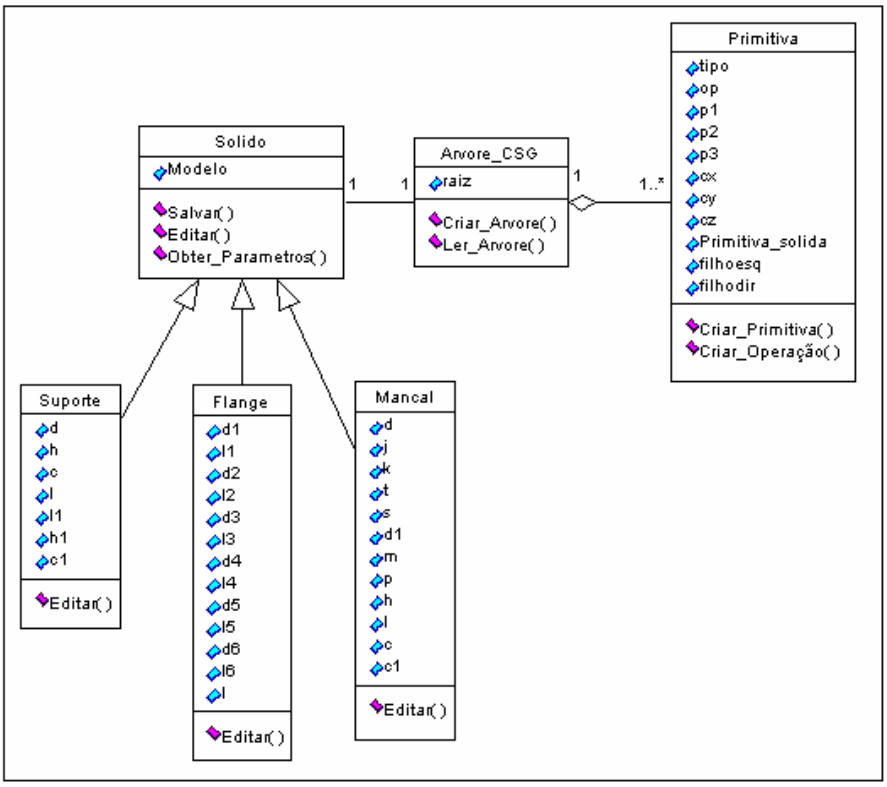


Descrição dos casos:

- criar Sólido: O usuário escolhe o tipo do sólido que deseja criar e informa os parâmetros necessários para sua criação;
- editar Sólido: O usuário informa quais o parâmetros que devem ser alterados;
- salvar Sólido: O usuário informa o nome do arquivo para salvar o modelo sólido;
- ler Árvore: O usuário visualiza a árvore CSG criada para a definição do modelo correspondente.

A figura 31 apresenta o diagrama de classes relativo a especificação do protótipo.

FIGURA 31 - DIAGRAMA DE CLASSES



A classe *Solido* representa o modelo sólido criado pelo protótipo, e é classe base para as classes *Suporte*, *Flange* e *Mancal*. Estas classes herdam os atributos e métodos da classe *Solido* e possuem atributos que são os parâmetros correspondentes à criação do modelo.

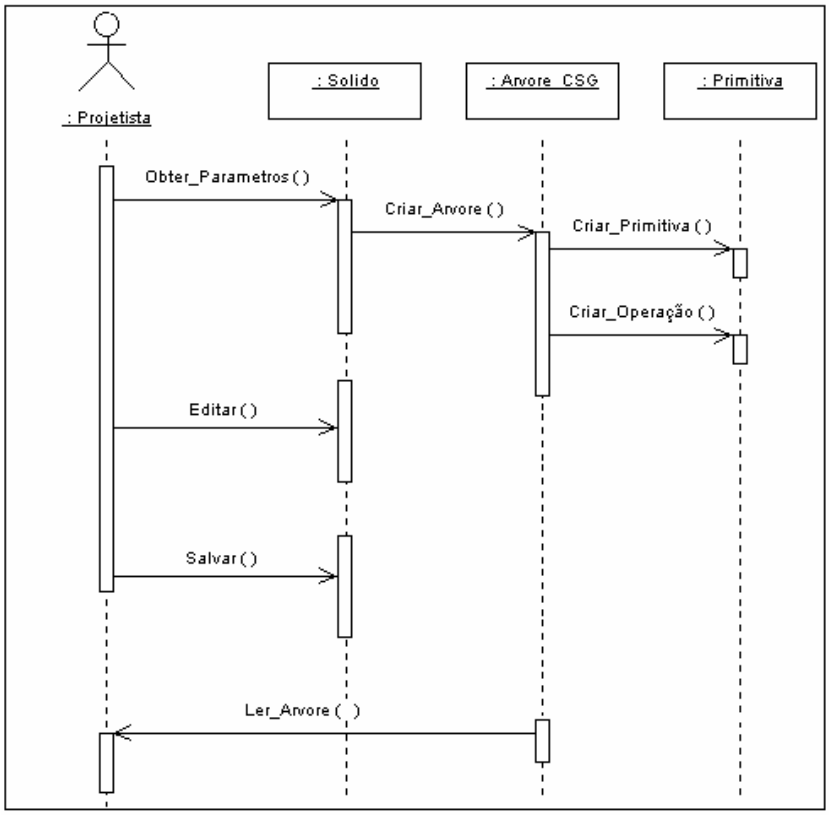
A classe *Arvore_CSG*, representa a árvore CSG definida para a criação de cada modelo. O atributo *raiz* armazena o endereço de memória da primitiva que é a raiz da árvore. Os métodos *Criar_Arvore* e *Ler_Arvore* realizam a construção e leitura da árvore, respectivamente.

A classe *Primitiva* representa as primitivas sólidas utilizadas para construção do modelo. Os atributos desta classe são genéricos, assim ela pode representar qualquer tipo de primitiva implementada no protótipo. O método *Criar_Primitiva* define uma nova primitiva

básica, e o método Criar_Operacao define uma nova primitiva através da aplicação de operações boolenas.

A figura 32 apresenta o diagrama de seqüência do protótipo do sistema de modelagem paramétrica de sólidos.

FIGURA 32 - DIAGRAMA DE SEQUÊNCIA

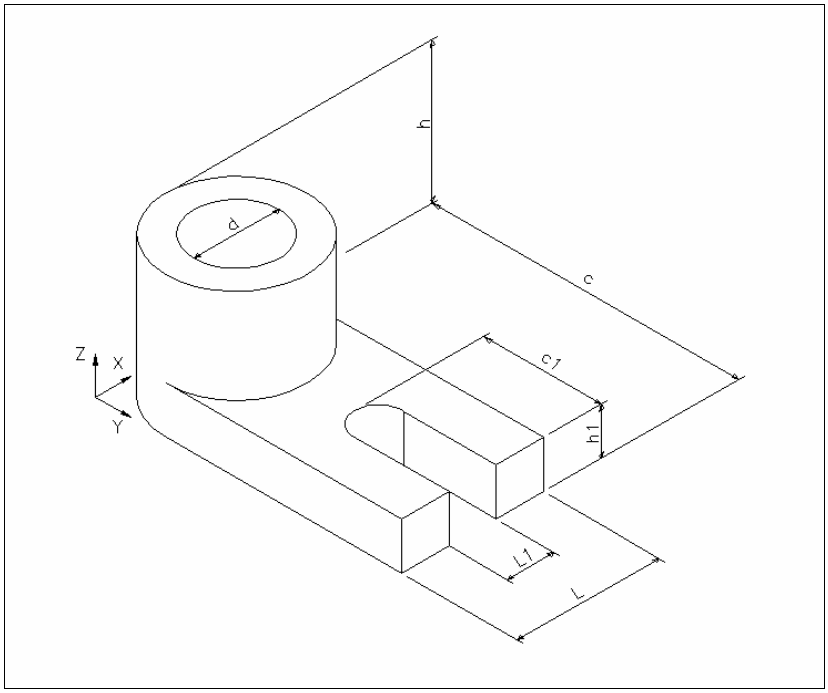


Na criação dos modelos sólidos, são gerados os eventos *Obter_Parametros*, *Criar_Arvore*, *Criar_Primitiva* e *Criar_Operacao*, nesta seqüência. O projetista executa ainda as operações de editar e salvar os modelos, e vizualiza o nome das primitivas e operações realizadas na árvore CSG.

Na escolha dos tipos de modelos sólidos procurou-se definir objetos funcionais da indústria mecânica que pudessem ser representados pela técnica CSG, e onde sua

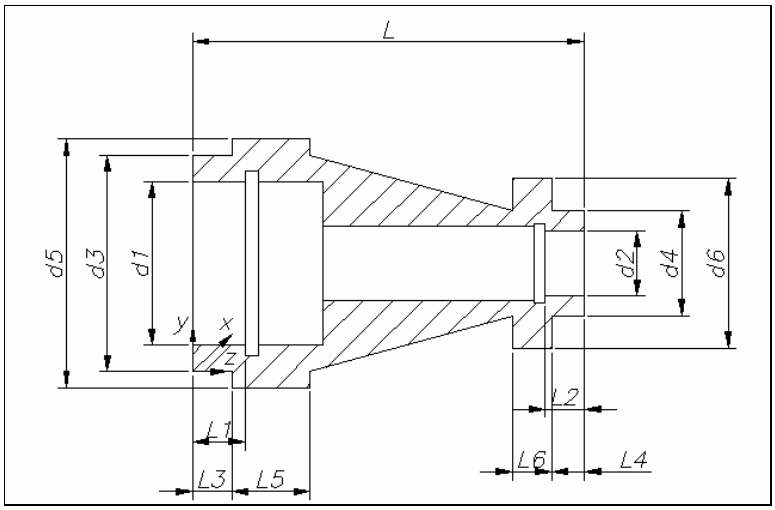
parametrização fosse interessante para a criação dos modelos. A figura 33 demonstra o modelo do tipo suporte, e os parâmetros para sua criação.

FIGURA 33 – PARÂMETROS DO MODELO TIPO SUPORTE



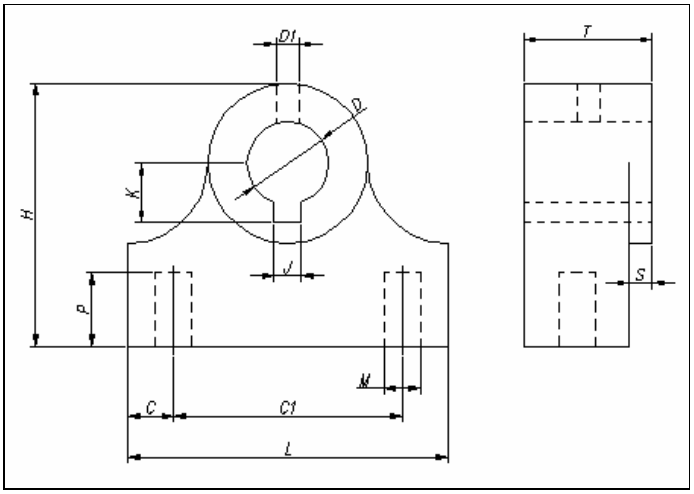
A figura 34 demonstra o modelo do tipo Flange com a definição de seus parâmetros.

FIGURA 34 – PARÂMETROS DO MODELO TIPO FLANGE



A figura 35 demonstra os parâmetros para criação do modelo do tipo Mancal.

FIGURA 35 – PARÂMETROS DO MODELO TIPO MANCAL



4.4 IMPLEMENTAÇÃO DO PROTÓTIPO

O protótipo foi totalmente implementado em linguagem C++ no ambiente Microsoft Visual C++ versão 6.0. Para definição e construção dos modelos, foi utilizado o modelador

3D ACIS *Modeller* 7.0. Conforme especificação vista anteriormente, o protótipo permite a construção de três modelos sólidos diferentes.

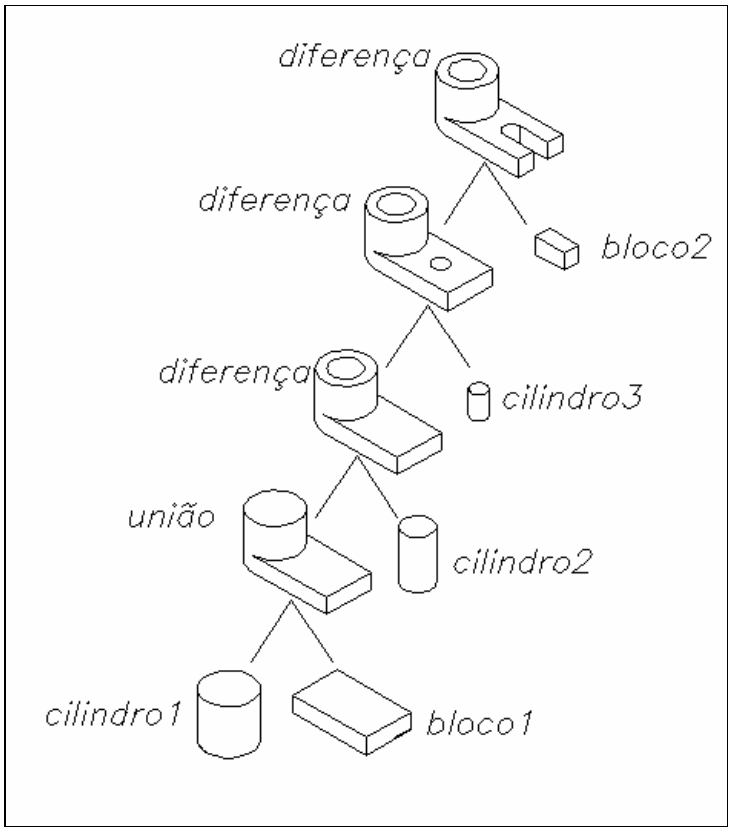
Com a identificação dos parâmetros dos modelos sólidos, definiu-se as primitivas sólidas necessárias para criação dos mesmos, bem como as operações booleanas necessárias para combinar as primitivas. Com estas definições, pode-se representar a árvore CSG do modelo, onde ficam armazenadas todas as informações das primitivas e das operações booleanas utilizadas para construção do modelo final. Em suma, a árvore CSG guarda a história da construção do modelo. Para parametrizar os modelos, utilizou-se definições que padronizassem algumas de suas dimensões em relação aos parâmetros definidos pelo usuário, bem como parametrizações que assegurassem a consistência da geometria do modelo. Além disso, o protótipo possibilita a edição dos modelos sólidos criados, isto ocorre através de alterações nas informações armazenadas na árvore CSG, e reconstrução do sólido com os novos parâmetros. A figura 36 demostra a árvore CSG definida para a criação do modelo sólido do tipo Suporte.

A árvore CSG armazena também ponteiros para o endereço de memória das primitivas sólidas criadas em cada nível da árvore, e resultados das operações booleanas. O quadro 2 mostra a implementação da classe Primitiva, com a qual é construída a estrutura da árvore CSG.

QUADRO 2 - CLASSE PRIMITIVA

```
class Primitiva
{
  public:
    char tipo, op;
    char nome[30];
    double p1, p2, p3, cx, cy, cz;
    BODY *primitiva_solida;
    Primitiva *filhoesq;
    Primitiva *filhodir;
    criar_primitiva (char tipo, double p1, double p2,
      double p3, double cx, double cy, double cz, BODY *&primitva_solida);
    criar_operacao (char op, BODY *primitiva_solida1,
      BODY *primitiva_solida2, BODY *&primitiva_solida);
};
```

FIGURA 36 – ÁRVORE CSG DO MODELO TIPO SUPORTE



A classe Primitiva pode definir tanto as folhas da árvore, que representam apenas primitivas básicas, como os nodos, que representam o resultado das operações boolenas realizadas. A variável *tipo*, define o tipo da primitiva. Definiu-se quatro tipo de primitivas para este protótipo: cilindro, cone, bloco e sólido. As três primeiras são primitivas sólidas básicas, enquanto o sólido é o resultado da aplicação de alguma operação booleana sobre as primitivas.

A variável *op* (de operação), define qual operação booleana será realizada entre as primitivas indicadas, e a variável *nome* define um nome para a primitiva, interessante para a visualização da árvore CSG. As variáveis *p1*, *p2* e *p3* recebem os parâmetros para a criação das primitivas, como raio e altura ou largura, altura e comprimento. As variáveis *cx*, *cy*, e *cz* recebem os valores das coordenadas absolutas correspondentes a cada primitiva.

A variável *primitiva_solida*, do tipo *BODY** , armazena um ponteiro para um objeto ACIS, no caso um sólido, que pode ser uma primitiva básica, um sólido intermediário, ou o modelo sólido final. Para fazer o encadeamento das folhas e nodos da árvore, são utilizadas as variáveis *filhoesq* e *filhodir*, que apontam respectivamente para o filho esquerdo e para o filho direito do nodo.

O método *criar_primitiva* recebe os parâmetros correspondentes, inclusive o endereço da variável que aponta para o sólido, e cria um objeto ACIS de acordo com parâmetros informados. O quadro 3 apresenta a implementação do método *criar_primitiva*.

QUADRO 3 - MÉTODO CRIAR PRIMITIVA

```
Primitiva::criar_primitiva (char tipo, double p1, double p2,
      double p3, double cx, double cy, double cz, BODY *&primitiva_solida)
{ api_initialize_constructors(); //inicializa funções de construção
  position ponto1, ponto2;
  if (tipo == 'C') // construir cilindro
    {//setagem das coordenadas
     ponto1.set_x(cx);
     ponto1.set_y(cy);
     ponto1.set_z(cz);
     ponto2.set_x(cx);
     ponto2.set_y(cy);
     ponto2.set_z(cz+p2); //constroi cilindro
     api_solid_cylinder_cone (ponto1, ponto2, p1, p1, p1, NULL, primitiva_solida);
    }
  else
    if (tipo == 'N') // construir cone
    {//setagem das coordenadas
     ponto1.set_x(cx);
     ponto1.set_y(cy);
     ponto1.set_z(cz);
     ponto2.set_x(cx);
     ponto2.set_y(cy);
     ponto2.set_z(cz+p2);//constroi cone
     api_solid_cylinder_cone (ponto1, ponto2, p1, p1, p3, NULL, primitiva_solida);
    }
    else  //construir bloco
    {//setagem das coordenadas
     ponto1.set_x(cx);
     ponto1.set_y(cy);
     ponto1.set_z(cz);
     ponto2.set_x(cx+p1);
     ponto2.set_y(cy+p2);
     ponto2.set_z(cz+p3);//constroi bloco
     api_solid_block (ponto1, ponto2, primitiva_solida);
    }
  api_terminate_constructors();
};
```

Antes da chamada de qualquer função de modelagem, é necessário fazer a inicialização destes componentes. Neste caso, a inicialização e encerramento são feitos diretamente neste método, através das funções APIs correspondentes. Para cada tipo de primitiva criada, é feita

a determinação das coordenadas, e chamada da função de modelagem com os respectivos parâmetros.

Para combinar as primitivas sólidas, é utilizado o método *criar_operacao* (quadro 4), que submete dois sólidos à operação booleana correspondente, e retorna o sólido resultante desta operação.

Este método recebe como parâmetros primeiramente a operação correspondente: “U” para união, “S” para diferença ou subtração, e caso nenhuma destas, a operação é a interseção. As outras três variáveis definem as primitivas envolvidas na operação. A primeira recebe o sólido que será descartado após a realização da operação, a segunda o sólido base para operação, e a terceira recebe o endereço deste sólido, retornando assim o sólido resultante da operação.

QUADRO 4 - MÉTODO CRIAR OPERAÇÃO

```
Primitiva::criar_operacao (char op, BODY *primitiva_solida1,
      BODY *primitiva_solida2, BODY *&primitiva_solida)
{
  api_initialize_booleans (); //inicializa operadores booleanos
  if (op == 'U')
    {//criar união solido1 com solido2;
     api_unite (primitiva_solida1, primitiva_solida2);
     primitiva_solida = primitiva_solida2;
    }
  else
    {if (op =='S')
      {//subtrair do solido1 o solido2;
       api_subtract (primitiva_solida1, primitiva_solida2);
       primitiva_solida = primitiva_solida2;
      }
    else
      {//criar intersecao do solido1 com solido2
       api_intersect (primitiva_solida1, primitiva_solida2);
       primitiva_solida = primitiva_solida2;
      }
    }
  api_terminate_booleans (); //encerra operadores booleanos
};
```

Estes dois métodos (*criar_primitiva* e *criar_operacao*) são acessados pelo método *criar_arvore*, que faz toda a construção da árvore CSG. O quadro 5 apresenta uma parte do código deste método que define uma folha da árvore.

QUADRO 5 - DEFINIÇÃO DE UMA FOLHA DA ÁRVORE CSG

```
if (tipo == 1)
{//criar primitiva 1
  Primitiva *aux1;
  aux1= new Primitiva;
  aux1->tipo = 'C'; //primitiva do tipo cilindro
  aux1->p1 = p4/2; //raio de c1
  aux1->p2 = p2;  //altura de c1
  aux1->p3 = 0; //inexistente
  aux1->cx = aux1->p1;  //coordenada x
  aux1->cy = aux1->p1; //coordenada y
  aux1->cz = 0; //coordenada z
  aux1->op = 'N'; //não existe operação
  aux1->filhoesq = NULL; //não tem filhos
  aux1->filhodir = NULL; //não tem filhos
  strcpy (aux1->nome, "cilindro1");//descricao do solido
  aux1->criar_primitiva (aux1->tipo, aux1->p1,aux1->p2, aux1->p3,
    aux1->cx, aux1->cy, aux1->cz, aux1->primitiva_solida);
```

No caso de uma folha da árvore, o sólido é uma primitiva simples, identificada pelas particularidades de não possuir filhos e não realizar nenhuma operação booleana. Os demais dados da classe são definidos de acordo com a geometria do modelo e parâmetros informados.

A definição de um nodo da árvore está demonstrada no quadro 6. Neste caso uma nova primitiva é definida como o sólido resultante de uma operação booleana.

QUADRO 6 - DEFINIÇÃO DE UM NODO DA ÁRVORE CSG

```
//criar uniao c1/b1
  Primitiva *auxraiz;
  auxraiz= new Primitiva;
  auxraiz->tipo = 'S'; // primitiva do tipo solido
  auxraiz->op = 'U'; //operacao de uniao
  auxraiz->filhoesq = aux1; //filho esquerdo é c1
  auxraiz->filhodir = aux2; //filho direito é b1
  strcpy (auxraiz->nome , "solido_result1");//descricao do solido
  auxraiz->criar_operacao (auxraiz->op, aux2->primitiva_solida,
                  aux1->primitiva_solida, auxraiz->primitiva_solida);
```

Quando cria-se um nodo da árvore, cria-se também mais um nível da sua estrutura. Por isso é necessário fazer o encadeamento do nodo com seus filhos. Para isto utiliza-se de variáveis auxiliares para fazer esta amarração, conforme quadro 6. Ao definir os filhos esquerdo e direito, têm-se também acesso as primitivas sólidas que serão submetidas a operação booleana. Isto porque para o protótipo definiu-se como padrão que o sólido

apontado pelo filho esquerdo é o sólido base da operação, ou seja, o sólido resultante, e o sólido apontado pelo filho direito é o sólido a ser eliminado.

Os nodos da árvore são definidos a partir das folhas em direção à raiz da árvore. A raiz representa o modelo sólido final. A maneira como as primitivas são combinadas e a árvore é estruturada depende de cada modelo sólido definido.

O protótipo permite também alterações nos modelos sólidos gerados. Uma vez que têm-se os parâmetros de criação do sólido e a árvore CSG implementada, pode-se em função das alterações realizadas nos parâmetros, alterar os dados das primitivas e dos sólidos gerados para a construção do modelo final, acessando a árvore CSG. O quadro 7 mostra como é feito o acesso e atualização dos dados da árvore CSG.

QUADRO 7 – ALTERAÇÃO DOS DADOS DA ÁRVORE CSG

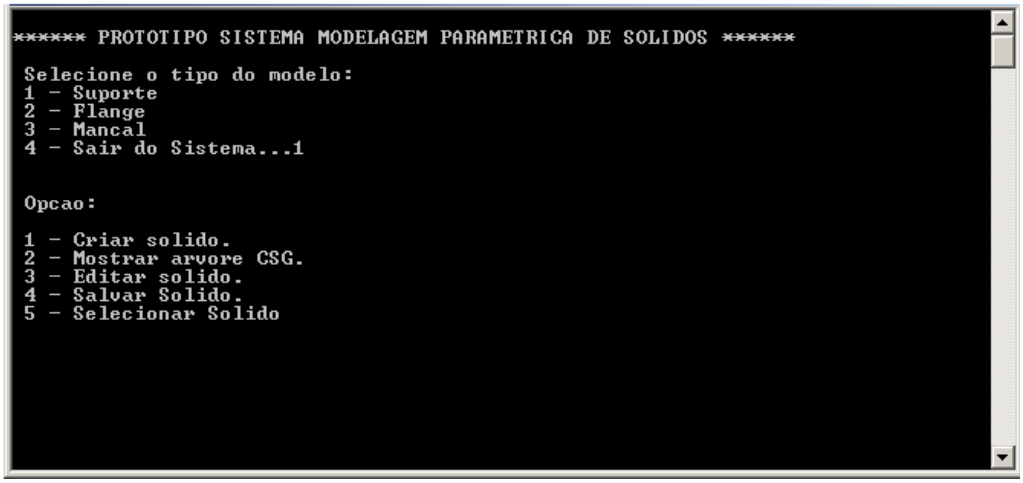
```
{
  if (raiz)
   {Primitiva* aux1;
    aux1= raiz->filhoesq->filhoesq->filhoesq->filhoesq; //acessa primitiva
    aux1->p1 = l/2; //atualiza parametros
    aux1->p2 = h;
    aux1->cx = l/2;
    aux1->cy = l/2;
    aux1->criar_primitiva (aux1->tipo, aux1->p1, aux1->p2, aux1->p3,
      aux1->cx, aux1->cy, aux1->cz, aux1->primitiva_solida); //atualiza sólido

//editar primitiva bloco1
    Primitiva* aux2;
    aux2= raiz->filhoesq->filhoesq->filhoesq->filhodir; //acessa primitiva
    aux2->p1 = l; //atualiza parametros
    aux2->p2 = c-(l/2);
    aux2->p3 = h1;
    aux2->cy = l/2;
    aux2->criar_primitiva (aux2->tipo, aux2->p1, aux2->p2, aux2->p3,
      aux2->cx, aux2->cy, aux2->cz, aux2->primitiva_solida); // atualiza sólido

//editar solido resultante uniao c1 / b1
    Primitiva* auxraiz;
    auxraiz= raiz->filhoesq->filhoesq->filhoesq; //acessa primitiva
    auxraiz->criar_operacao (auxraiz->op, aux2->primitiva_solida,
      aux1->primitiva_solida, auxraiz->primitiva_solida);//atualiza sólido resultante
```

A atualização dos dados ocorre das folhas em direção à raiz da árvore. O quadro 7 apresenta as alterações realizadas em dois níveis da árvore. Para acessar as primitivas, cria-se variáveis auxiliares que apontam para o endereço de memória das primitivas. Desta forma, os dados que realmente necessitam ser alterados podem ser atualizados, e a variável *primitiva_solida* passa a armazenar um novo modelo sólido onde são refletidas as alterações dos parâmetros. Ao se percorrer toda a árvore CSG, têm-se o modelo sólido final alterado.

4.5 FUNCIONAMENTO DO PROTÓTIPO

A interface do protótipo é extremamente simples, e toda a interação com o usuário é feita em linha de comando. O protótipo possibilita ao usuário criar os modelos sólidos descritos anteriormente, com dimensões variadas. Primeiramente faz-se a opção por um dos três modelos: Suporte, Mancal ou Flange. Selecionado um dos tipos de modelo, têm-se as opções criar, visualizar árvore CSG, editar, salvar e selecionar novo tipo, conforme demonstrado na figura 37.

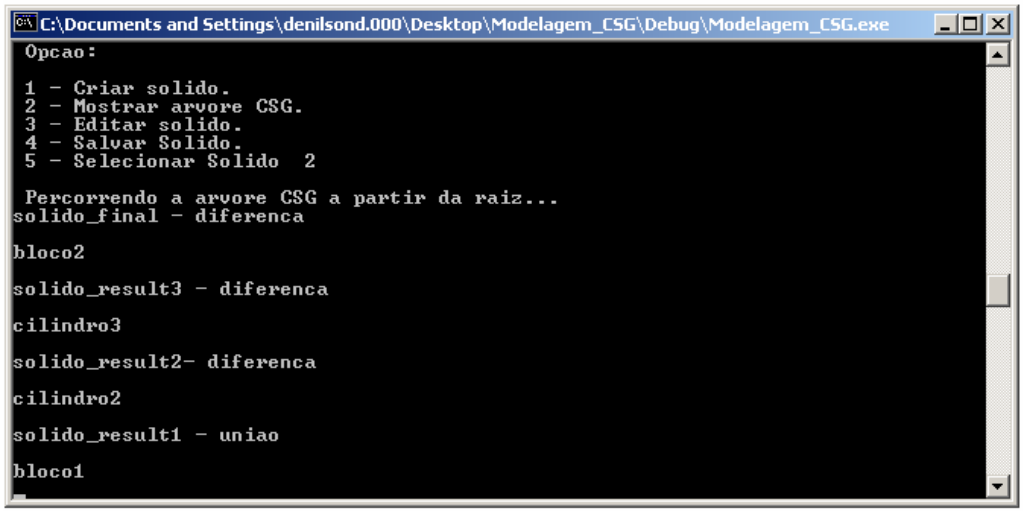
FIGURA 37 – TELA PRINCIPAL DO PROTÓTIPO



A opção criar, requisita que o usuário informe os valores dos parâmetros necessários a criação do modelo selecionado, e com todos os parâmetros informados cria-se o modelo sólido. Neste instante pode-se optar por visualizar a árvore CSG. Esta opção mostra ao usuário uma descrição das primitivas e das operações criadas para a construção do modelo sólido final, percorrendo a árvore da raiz em direção as folhas, conforme demonstrado na figura 38.

A opção editar permite que os modelos sólidos já criados sejam alterados. Para isso informa-se os valores dos parâmetros a serem alterados, e estas alterações são refletidas nas dimensões do modelo.

FIGURA 38 – VISUALIZAÇÃO DA ÁRVORE CSG



Para a visualização do modelo sólido criado, é necessário salvá-lo em arquivo. Através da opção Salvar Sólido, informa-se o nome do arquivo, e tem-se o sólido armazenado em um arquivo tipo SAT.

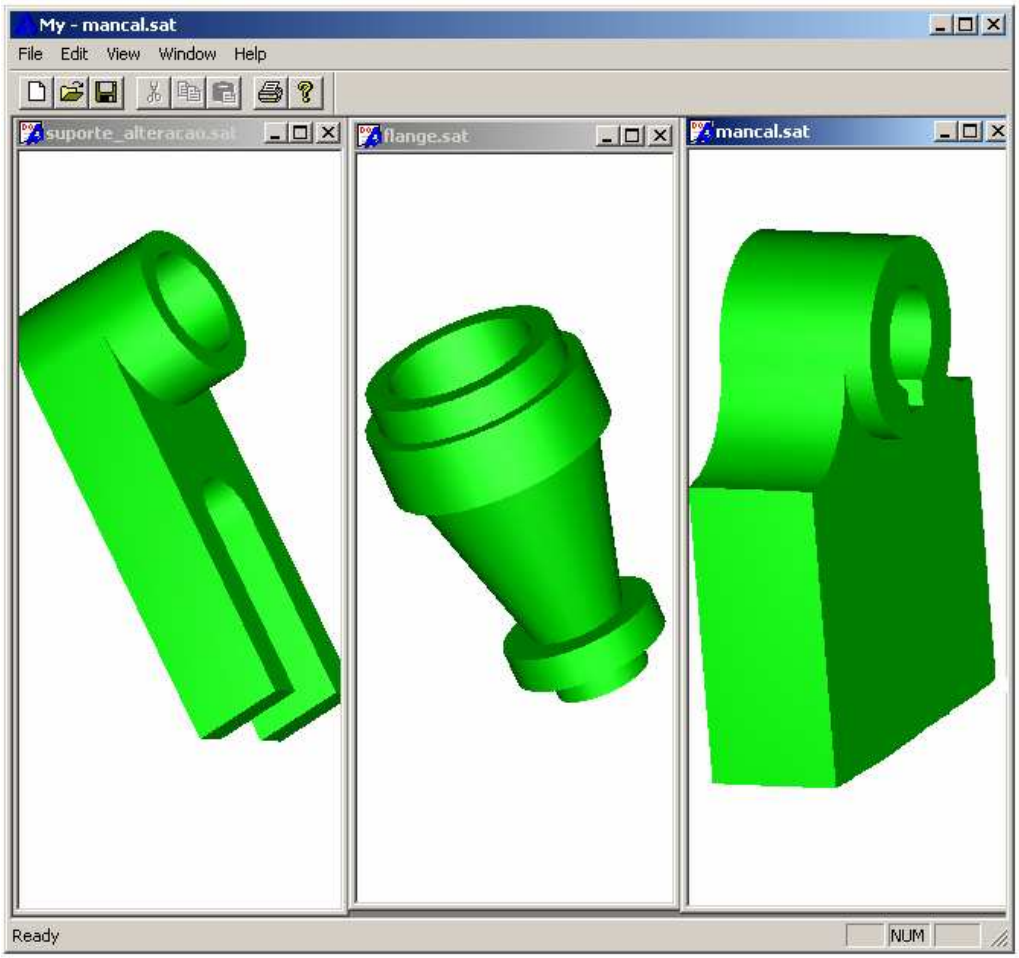
Uma parte da formatação de um arquivo SAT pode ser observada no quadro 8. Na primeira linha está definida a versão do ACIS utilizada para a criação do arquivo. A segunda e terceira linhas definem informações de criação do arquivo e precisão decimal. As demais linhas representam os elementos geométricos e topológicos do modelo e seus respectivos dados.

QUADRO 8 – ARQUIVO SAT

```
700 0 1 0
21 ACIS/3D Toolkit - 7.0 11 ACIS 7.0 NT 24 Sun Jul 07 14:40:27 2002
1 9.9999999999999995e-007 1e-010
body $-1 -1 $-1 $1 $-1 $2 #
lump $-1 -1 $-1 $-1 $3 $0 #
transform $-1 -1 1 0 0 0 1 0 0 0 1 5 5 10 1 no_rotate no_reflect no_shear #
shell $-1 -1 $-1 $-1 $-1 $4 $-1 $1 #
face $-1 -1 $-1 $5 $6 $3 $-1 $7 forward single #
face $-1 -1 $-1 $8 $9 $3 $-1 $10 forward single #
loop $-1 -1 $-1 $-1 $11 $4 #
plane-surface $-1 -1 $-1 -2.5 18.75 -7 1 0 0 0 0 -1 forward_v I I I I #
face $-1 -1 $-1 $12 $13 $3 $-1 $14 forward single #
loop $-1 -1 $-1 $-1 $15 $5 #
```

A visualização dos modelos sólidos pode ser feita pelo visualizador fornecido pela Spatial Corporation. Este visualizador possui o código fonte aberto, pois trata-se do esqueleto de uma aplicação ACIS que fornece toda interface de visualização de arquivos SAT. Assim pode-se também acrescentar novas funcionalidades a este visualizador. A figura 39 demonstra a interface do visualizador e os modelos tipo Suporte, Flange e Mancal, criados pelo protótipo nesta ordem.

FIGURA 39 – MODELOS CRIADOS PELO PROTÓTIPO



O visualizador de arquivos SAT, permite múltiplas janelas, assim pode-se visualizar o modelo original, e as possíveis alterações a serem realizadas. No próximo capítulo são apresentadas as conclusões e sujestões para trabalhos futuros.

5 CONCLUSÕES

A utilização do 3D ACIS *Modeller* foi o ponto chave no desenvolvimento deste trabalho. Através do *Online Help Guide for ACIS*, fornecido pela empresa Spatial Corporation, pode-se obter o conhecimento de sua estrutura e funções que este disponibiliza. Na implementação do protótipo ocorreram vários problemas em relação à configuração de um projeto de compilação do código C++. Após inúmeros testes e várias consultas à lista de discussões do ACIS, chegou-se finalmente as configurações de projeto necessárias para compilar o código fonte e gerar o código executável.

Como pré-requisito para a implementação do protótipo, houve a necessidade de se definir um modelo sólido capaz de representar de forma satisfatória o conjunto de técnicas exploradas neste trabalho. Através de sua definição, tornou-se necessário desenvolver e validar uma estrutura de dados que representasse a árvore CSG. Foram então elaboradas as definições paramétricas de três modelos sólidos.

A opção pela construção de modelos sólidos pré-definidos está de acordo com os objetivos do trabalho, que compreendem o estudo e a implementação de modelagem paramétrica de sólidos com construção de árvores CSG. Para criação de um modelador paramétrico genérico, é necessária a implementação de um sistema para resolução de restrições geométricas de forma a garantir a validade dos modelos gerados. A resolução de tais tipos de problemas está fora do escopo deste trabalho, sendo sugerida, como tema para trabalhos futuros. Entretanto, novos modelos paramétricos podem ser incluídos no protótipo existente através da implementação de novos métodos para criação de modelos na classe *Primitiva*, descrita nas seções 4.3 e 4.4.

A implementação da interface de comunicação entre a aplicação C++ e as funções de geometria do ACIS exigiu conhecimentos em desenho técnico para a definição da geometria dos modelos. Os testes do protótipo demonstraram que sua funcionalidade estava de acordo com os objetivos propostos para este trabalho de pesquisa. Como a construção de um visualizador de modelos sólidos através do Microsoft Visual C++ é bastante complexa, fez-se necessária a decisão de optar por uma interface DOS através de uma aplicação *console* para a construção do protótipo. Assim, visto que o protótipo implementado tratou da construção

paramétrica da geometria dos modelos, a sua visualização passou a ser realizada através de uma aplicação fornecida pelo fabricante do ACIS.

Pode-se afirmar que os objetivos deste trabalho foram alcançados, uma vez que se tem implementado o protótipo do sistema de modelagem paramétrica de sólidos, que permite a criação de modelos sólidos baseados na técnica de modelagem CSG, e também a edição destes modelos e opção de salvá-los em arquivos do tipo SAT, que pode ser interpretados por diversos sistemas CAD, CAM e CAE.

5.1 LIMITAÇÕES

Naturalmente por se tratar de um protótipo, notou-se que o mesmo possui algumas limitações, como por exemplo:

- Falta de uma interface gráfica mais amigável para a geração dos modelos sólidos;
- Falta de restrições geométricas que permitam manter a consistência do modelo no caso de falha na definição dos dados por parte do usuário.

5.2 EXTENSÕES

Como extensão deste trabalho pode-se adicionar mais objetos a serem modelados pelo protótipo, ou então implementar outras técnicas para modelagem de sólidos utilizando o ACIS como núcleo de geometria do sistema.

Pode-se também implementar a técnica CSG disponibilizando um conjunto de primitivas sólidas, que combinadas através das operações booleanas e transformações geométricas permitam ao usuário criar modelos sólidos de formas variadas.

Outra extensão sugerida é o aprofundamento das questões mais específicas à modelagem paramétrica de sólidos. Em especial, sugere-se o estudo dos problemas relacionados a consistência geométrica do modelo. Este tipo de questão pode ser tratada através da resolução de sistemas de restrições. Vários métodos e *frameworks* para sistemas deste tipo são sugeridas nas referências bibliográficas, em especial, destaca-se a proposta de

Gomes (1999) e Feijó (2001), que constitui um projeto de pesquisa em andamento no Departamento de Sistemas e Computação da FURB.

Por fim, sugere-se a implementação de um protótipo com interface gráfica para a modelagem paramétrica de sólidos.

REFERÊNCIAS BIBLIOGRÁFICAS

BADLER, Norman I. **Simulating humans:** computer graphics, animation and control, Philadelphia, [2000?]. Disponível em: <http://www.cis.upenn.edu/~hms/jack.html>. Acesso em: 12 fev. 2002.

BERNARDI, André. **Curso de extensão – Microsoft Visual C++**, Itajubá. [1999]. Disponível em: <http://www.projesom.com.br/simone/Download/ApostilaVisuaC.pdf>. Acesso em: 22 maio 2002.

BOGADO, Wilson Horstmeyer. **Customização de sistemas CAD comerciais**. 1997. 82 f. Dissertação de Mestrado (Métodos Numéricos para Engenharia) – Setor de Tecnologia/Setor de Ciências Exatas. Universidade Federal do Paraná, Curitiba.

CASACURTA, Alexandre. **Modelagem geométrica,** São Leopoldo, [1999]. Disponível em: <http://www.inf.unisinos.br/~marcelow/ensino/grad/cg/modelagem/modelagem.html>. Acesso em: 28 fev. 2002.

CORNEY, Jonathan. **3D modeling with the ACIS kernel and toolkit**. 2. ed. Chichester: Wiley & Sons, 2000.

FEIJÓ, Bruno et al. Online algorithms supporting emergence in distributed CAD systems. **Advances in Engineering Software**, Amsterdam, v. 32, n. 10-11, p. 779-787, Oct-Nov 2001.

FOLEY, James D et al. **Computer graphics :** principles and practice. 2. ed. Washington: Addison-Wesley, 1990.

GIESECKE, Frederick E. et al. **Comunicação gráfica moderna**. Tradução Alexandre Kawano. Porto Alegre: Bookman, 2002.

GOMES, Paulo César Rodacki. **Prototipação Virtual em Modelagem de Sólidos Distribuída**. 1999. 94f. Tese de Doutorado - Departamento de Informática, PUC-Rio Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

GOMES, Paulo César Rodacki; ODEBRECHT, Marcelo; WISINTAINER, Miguel Alexandre. **Implementação de árvores BSP na plataforma Playstation**, Blumenau, 2002.

KERRY, H. T. **Planejamento de processo automático para peças paramétricas**. 1997. Dissertação de Mestrado – Escola de engenharia de São Carlos. Universidade de São Paulo.

MORTENSON, Michael E. **Geometric modeling**. New Yrok: J. Wiley, 1985.

MORTENSON, Michael E. **Mathematics for computer graphics applications**. 2. ed. New York: Industrial Press, 1999.

REDDY, Junuthula Narasinha. **An introduction to the finite element method**. 2. ed. New York: McGraw-Hill, 1993.

REQUICHA, A. A. G.; VOELCKER, H. B. Boolean operations in solid modeling: boundary evaluation and merging algorithms. **Proceedings of IEEE**, v. 73, n. 1, p. 30-44, 1985.

REQUICHA, A. A. G.; ROSSIGNAC, J. Solid modeling. **Encyclopedia of Electrical an Electronics Engineering**, J. Webster , 1999.

SILVA, André Tavares da. **Integração de técnicas de modelagem com VRML**. 1999. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Informática – Habilitação em Software Básico) – Centro de Ciências Exatas e Tecnológicas. Universidade do Vale do Rio dos Sinos, São Leopoldo. Disponível em: <http://www.inf.unisinos.br/~osorio/vr/ATSWorlds/ ATSWorlds.pdf>. Acesso em: 28 fev. 2002.

SOUZA, Rogério Pinheiro de. **Avaliação de bordo exata**. 2000. 112 f. Tese de Mestrado em Ciências de Engenharia de Sistemas e Computação. Universidade Federal do Rio de Janeiro, Rio de Janeiro. Disponível em: <http://orion.lcg.ufrj.br/~rogerio/Tese/Tese-AvaliacaoDeBordoExata.pdf>. Acesso em: 13 fev. 2002.

SPATIAL CORPORATION. On line help for ACIS version 7.0. Westminster, 2001.