APROXIMAÇÃO LINEAR POR PARTES DE SÓLIDOS CSG ATRAVÉS DE SUBDIVISÃO SIMPLICIAL ADAPTATIVA

Marcelo Salim da Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHAFUA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁFUOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Ronaldo Cesar Marinho Persiano, D. Sc.
(Presidente)

Prof. Geovan Tavares dos Santos, D. Sc.

Prof. Luiz Carlos Guimarães, Ph. D.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1992

SILVA, MARCELO SALIM DA
Aproximação Linear por Partes de Sólidos CSG Através de Subdivisão Simplicial Adaptativa [Rio de Janeiro] 1992 VI, xiii + 122 p. 29,7 cm
(COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 1992)
Tese – Universidade Federal do Rio de Janeiro, COPPE
1. Modelagem de Sólidos I. COPPE/UFRJ
II. Título (série).

Resumo da Tese apresentada a COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M. Sc.).

APROXIMAÇÃO LINEAR POR PARTES DE SÓLIDOS CSG ATRAVÉS DE SUBDIVISÃO SIMPLICIAL ADAPTATIVA

Marcelo Salim da Silva

Junho de 1992

Orientador: Prof. Ronaldo Cesar Marinho Persiano

Programa: Engenharia de Sistemas e Computação

Apresentamos, neste trabalho, uma nova proposta de subdivisão espacial para fins de aproximação linear por partes da fronteira de sólidos CSG: a Subdivisão Simplicial Adaptativa. Nela, a região onde o sólido está inserido vai sendo decomposta hierarquicamente, em simplexos de tamanhos cada vez menores. O refinamento final da malha é mais denso naquelas regiões que contêm partes da fronteira do sólido.

Para que a convergência do processo de subdivisão para a fronteira do sólido seja eficiente, existe a necessidade de critérios seguros de classificação de cada simplexo da decomposição com respeito ao sólido. Somente naqueles simplexos disjuntos da fmnteira é que o processo de subdivisão pode ser interrompido precocemente.

Discutiremos alguns destes critérios, especialmente dois que se mostraram bastante eficientes e que utilizam a representação das funções definidoras das primitivas CSG na Base de Bernstein. Os coeficientes de Bézier das primitivas são usados, direta ou indiretamente, para se obter a classificação dos simplexos.

A representação das funções na Base de Bernstein propicia a utilização de primitivas CSG definidas por funções polinomiais de, em princípio, qualquer grau, aumentando consideravelmente o poder de expressão dos modeladores.

Além da aproximação linear por partes da fronteria do sólido, a árvore de subdivisão simplicial, gerada durante a busca pela fronteira, pode ser usada para otimizar algoritmos de RAY-TRACING e/ou assemelhados. Pode, também, fornecer informações topológicas sobre as faces lineares computadas, viabilizando um conversor CSG→B-Rep.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.).

LINEAR APPROXIMATION OF CSG SOLIDS BY ADAPTIVE SIMPLICIAL SUBDMSION

Marcelo Salim da Silva

June, 1992

Thesis Supervisor: Prof. Ronaldo Cesar Marinho Persiano

Department: Engenharia de Sistemas e Computação

This work presents a new proposal for computing linear approximations of CSG solid boundaries through spatial subdivision: The Adaptive Simplicial Subdivision. In this approach, the region in which the solid is supposed to be is decomposed hierarchically into simplices, each one smaller than the other. The final mesh refinement has greater density at those regions that contain parts of the solid boundary.

In order to be successful, the subdivision process needs some careful and safe criteria to classify the simplices against the solid. Only in those simplices disjoint to the solid boundary the subdivison process might stop precociously.

We will discuss some of those criteria, mainly two that have shown to be very efficient. They work with the representation of the functions that define the CSG primitives in the Bernstein Basis, using directly or indirectly their Bézier coefficients, with respect to each simplex, to obtain the classifications.

The function representation in the Bernstein Basis allows CSG primitives defined by trivariate polynomials of, theoretically, arbitrary degrees. This strongly increases solid modelers expression capabilities.

Besides the linear approximation of the solid boundary, the spatiaí subdivision tree, generated during the boundary search, may be used to optimize RAY-TRACING-like algorithms as well as supplying topological information about the linear faces computed by a CSG→B-Rep converter.

"O homem apenas persegue fantasmas"
Pierre Simon De Laplace

Para Juju, Mauro, Dani e, especialmente, Zeca Limão e Monica.

AGRADECIMENTOS

Como em todos os projetos de pesquisa, muitas pessoas contribuiram, direta ou indiretamente, para a realização deste trabalho e, a todas, expresso meus sinceros agradecimentos. Hora trabalhando, hora atrapalhando o trabalho alheio, me diverti um bocado e acho que algumas pessoas merecem especial atenção.

Sua Divina Graça Ronaldo Cesar Marinho Persiano fez muito mais do que simplesmente orientar a tese. Com seu estilo *Zen-Budista-Pós-Bhagavad-gita*, ensinou-me muito sobre inúmeros assuntos que incluem Geometria Pmjetiva, Estruturas de Dados e Astrologia. A extensão dos seus conhecimentos, sua didática e sua paciência são realmente impressionantes. Igualmente notável é a dominação passiva que exerce sobre as pessoas que o cercam. O que mais me aproximou dele, contudo, foi sua peculiar maneira de sempre discordar de mim (sem querer ser presunçoso, acho que nós os polêmicos precisamos uns dos outros para nos mantermos equilibrados). Invejava-me a ingenuidade que aparentava quando dizia que o dinheiro não é muito importante. Fiz o que pude para livrá-lo dessas perniciosas idéias *Zen-Franciscanas* e espero tê-lo conseguido com o koan do iate repleto de alunas de Modelagem Geométrica. O espaço é findo e fica a enorme admiração e a esperança de algum dia poder voltar a trabalhar em sua companhia.

O Físico de *Plantão* Luis Paulo Bueno muito me ajudou desde o início do trabalho, com sugestões, discussões, dicas e explicações sobre os mais diversos temas. Seguindo sua filosofia de falar difícil de temas simples para simplificar temas difíceis, diria que ele é uma das pessoas mais bacanas do R^3 e que me foi muito importante estar na sua vizinhança.

O colega de graduação e mestrado Vitor Pêgo Hottum se revezou comigo na sempre árdua tarefa de fazer perguntas burras ao orientador. Juntos desenvolvemos vários trabalhos, discutimos pontos-chaves de nossas teses, modelamos objetos CSG, perturbamos a Helena e, fundamentalmente, zombamos de toda e qualquer pessoa que tenha passado próximo ao LCG. Agradeço, especialmente, sua inconveniente mania de apontar todos os meus erros e sua irritante teimosia em fazer-me reconhecê-los e consertá-los.

Ricardo Dias Campos, Abel Tadeu Amorim dos Reis e Alberto de Freitas Escarlate forneceram todo suporte computacional necessário a implementação das idéias apresentadas no trabalho, além de consertarem os compiladores e depuradores sempre que eu os quebrava. Ricardo (A Esfinge) respondia a qualquer coisa em forma de enigmas, Abel mandava procurar no manual (independentemente do quê) e o Alberto na de tudo (não sei se *pra* mim ou de mim). Foram, também, grandes companheiros nas jornadas diurnas da COPPE, discutindo futebol, política, música, cidadania, cinema, sexo, drogas e *Rock'nRoll*. Não há como esquecer, tampouco, as aventuras empreendidas em Santa Teresa, a noite, nas tradicionais Festas do *Ricardo*®, tão badaladas e politicamente corretas que Tim Maia, João Gilberto, além de Rogério Magri e Edir Macedo (e alguns outros ladrões com pernas quebradas), tinham sempre presença garantida.

Cláudio Esperança e João Luiz Dihl Comba eram os veteranos quando ingressei no mestrado e muito me aturaram desde então. Explicando seus programas e dando orientações diversas, colaboraram bastante comigo. Foi extremamente proveitoso trabalhar ao lado de pessoas tão preparadas e competentes. Ao Cláudio, especialmente, obrigado pela simplicidade e boa vontade com que sempre me atendeu. Quanto ao João, espero tê–lo convertido a Seita Botafoguense e poder vê–lo mais vezes torcendo conosco, na suficientemente imensa torcida alvi–negra (pelo menos, sempre que o Grêmio não estiver envolvido).

André Luiz Pires Guedes e Leonardo Cardoso Monteiro me ajudaram bastante com dicas, palpites, sugestões, chutes e, também, fornecendo programas e/ou trechos de suas bibliotecas C particulares.

José Henrique Gandra e Silva foi excelente companheiro de mestrado e muito me ajudou na estressante (e gratificante) tarefa de modelagem do carrinho de fórmula 1, além de sempre aparecer com uma piada diferente para cada situação. Pena o nosso convívio ter sido relativamente curto.

Os colegas do LCG André, Margareth, Giselli, Paulo, Célia, Luiz e, também, a Astrid, sempre me apoiaram e ofereceram um ótimo ambiente de trabalho, além de fornecerem muitos subsídios para minhas brincadeiras. Neste último sentido, agradeço especialmente a Helena (*/dev/null* ou *vibrião helênico*), fornecedora por atacado de motivos para achincalhes.

Agradeço muito ao pessoal do Laboratório de Hidrologia da COPPE (Canedo, Otto, Luciene, Ana Paula, Mário, Daísy ...), cuja paciência e compreensão, em vários momentos, foram imprescindíveis a realização deste trabalho.

A todos os demais colegas do Programa de Engenharia de Sistemas e Computação da COPPE, corpo técnico e docentes, pela colaboração, ambiente de trabalho, equipamentos etc, sou muito grato!

Meu padrinho, Fioravanti Alonso Di Piem, mostrou vontade incomum para ler atenciosamente cada capítulo, sugerindo várias modificações para melhorar a clareza do texto. Imagino o quão tediosa deve ter sido esta análise *léxica* pois para ele, um Médico, o texto deve ser pior do que duas caixas de sonífero. (Talvez o seja para qualquer um!)

Não poderia esquecer de agradecer, também, a todas aquelas pessoas que me incentivaram durante o andamento da tese e, aproveitando a oportunidade, gostaria de dizer–lhes que perguntas do tipo "...e a tese, como é que vai ?", são extremamente inconvenientes para os mestrandos que ainda não a terminaram.

Agradeço, também, ao Emerson, Senna, Piquet e ao automobilismo em geral; a Mazolinha, Maurício e ao Botafogo F. R.; a Eric Clapton, Vivaldi, Jeff Beck, Glenn Miller, Iron Maiden, Villa Lobos, AC/DC, Rick Wakeman, Jeff Healey, Led Zeppelin, Strauss, Spyro Gyra, Louis Armstrong, Steve Ray Vaughan, Mozart, Whitesnake, Chuck Berry, Bachman–Turner–Overdrive, Frank Sinatra, Deep Purple, Buddy Guy, Mendelsohn, Talking Heads, Gilberto Gil, Jimi Hendrix, Bach, The Cult, Djavan, Automat, John Lee Hooker, Altamiro Carrilho, Bateria da Mocidade Independente, Pink Floyd, Jamelão, U2, Duke Ellington, Caetano Veloso, Paganini, Tom Jobim, Robert Johnson, Ozzy Osbourne, Raul Seixas, Rachmaninoff, Yellow Jackets, Sivuca, Rush, Tchaikovsky, Black Sabbath, Albert Collins, Orff, Emerson Lake & Palmer, Ravel, Robert Cray, Beethoven, Magic Slim, Gary Glitter, Wagner, Dire Straits, Chopin, Scorpions, Hermeto Paschoal, Steppenwolf... e a tudo relacionado a música; a Regina Casé, Luis Femando Guimarães, Jô Soares e a Turma da Casseta & Planeta; ao Cinema, ao Teatro e, fundamentalmente, as mulheres, por tornarem o mundo tão imprevisível.

ÍNDICE

FIGURAS E TABELAS

INTRODUÇÃO

CAPÍTULO 1 – Modelagem de Sólidos

CAPÍTULO 2 – Técnicas Simpliciais

CAPÍTULO 3 – Sólidos CSG e Técnicas Simpliciais

CAPÍTULO 4 – Subdivisão Simplicial Adaptativa

CAPÍTULO 5 _ Aspectos Gerais da Implementação

CAPÍTULO 6 –Avaliações e Considerações Finais

REFERÊNCIAS BIBLIOGRÁFICAS

APÊNDICE A – Manual dos Programas de Subdivisão Simplicial Adaptativa

LISTA DE FIGURAS

LISTA DE TABELAS

"Tired of lying in the sunshine staying home to watch the rain..."

Introdução

O QUE VEREMOS NESTE TRABALHO

Dentre as diversas áreas da Computação Gráfica, uma das que mais se destaca é a Modelagem de Sólidos. A criação e manutenção de estruturas de dados que representem adequadamente no computador os objetos do mundo real, com suas diversas propriedades e peculiaridades, são tarefas das mais importantes e interessantes, haja visto o enorme escopo de aplicações.

Existem várias maneiras de se representar sólidos no computador, cada qual com grande bagagem de aspectos positivos para determinadas finalidades e negativos para outras. Entre estas formas de representação, três se sobressaem: CSG, B-Rep e Decomposição Espacial.

No presente trabalho, abordaremos a construção de aproximações lineares por partes para a fronteira de sólidos CSG, por intermédio de decomposição hierárquica do espaço em simplexos. Para esta finalidade, adotaremos alguns conceitos e técnicas derivados dos frequentemente utilizados Métodos Simpliciais.

Nossa proposta é a Subdivisão Simplicial Adaptativa, que considera o sólido inserido em determinada região do espaço e vai subdividindo-a adaptativamente em simplexos cada vez menores. O nível de refinamento da subdivisão em cada simplexo é tão maior quanto mais próximo ele estiver da fronteira do sólido. Daí o caráter adaptativo.

Para que a adaptação da subdivisão seja eficiente, são necessários critérios precisos de classificação dos simplexos com relação ao sólido. Apresentamos dois critérios de classificação bastante eficientes. Ambos utilizam a representação das funções definidoras das primitivas CSG na Base de Bemstein e, através dos seus coeficientes de Bézier em relação a cada simplexo da subdivisão, conseguem categorizar estes últimos em relação ao sólido como interiores, exteriores ou interceptantes a sua fronteira.

A representação das primitivas na Base de Bemstein possibilita a utilização, por parte dos modeladores CSG, de primitivas definidas por semiespaços polinomiais de, em

princípio, qualquer grau. Isto constitui um avanço na medida em que a maioria dos modeladores disponíveis atualmente utiliza primitivas definidas por pequenos subconjuntos das funções quadráticas.

Em nosso primeiro capítulo abordaremos a área de Modelagem de Sólidos. Descreveremos suas principais características, falaremos sobre os sistemas modeladores de sólidos e sobre os principais paradigmas sobre os quais se baseiam.

No segundo capítulo faremos breve exposição sobre os Métodos Simpliciais. Discutiremos para que servem, como funcionam e em que se baseiam, apresentando suas vantagens e questionando algumas de suas características. Apresentaremos, também, as idéias sobre decomposições hierárquicas do espaço, coordenadas baricêntricas e funções afins interpoladoras.

O capítulo três é reservado para explicar como as técnicas simpliciais podem ser utilizadas para computar aproximações para sólidos CSG. Introduziremos as funções características associadas as árvores CSG e veremos que os sólidos podem ter uma representação implícita através de funções do tipo MIN–MAX. Falaremos, também, dos aspectos relacionados a regularização dos sólidos quando da utilização destas funções, assim como de alguns pormenores técnicos que devem ser devidamente tratados.

O quarto capítulo expõe a maioria das idéias da nova proposta. Nele apresentaremos um processo de subdivisão do espaço baseado em *bintrees* de simplexos. Falaremos sobre aspectos relacionados com a localização do sólido no espaço e de como isto pode simplificar o processo de cálculo de uma aproximação para sua fronteira. Discutiremos, também, alguns critérios de classificação dos simplexos da decomposição contra o sólido CSG, principalmente aqueles que utilizam a representação das funções definidoras das primitivas na Base de Bemstein. Finalmente, falaremos sobre as técnicas, por nós utilizadas, de obtenção dos coeficientes de Bézier necessários as classificações dos simplexos.

Os aspectos principais da implementação serão apresentados no capítulo cinco. Lá abordaremos como as novas idéias introduzidas pelo trabalho foram implementadas no computador, em uma série de módulos que se interrelacionam. Cabe salientar que durante a elaboração dos algoritmos, dedicamo–nos apenas aqueles exequíveis em máquinas sequenciais. Esta escolha deveu–se ao fato de ainda serem estas as máquinas mais frequentemente utilizadas. Reconhecemos, no entanto, que arquiteturas especializadas e de processamento paralelo vem sendo objeto de intensos estudos e podem vir a fornecer, a médio prazo, soluções bem melhores para problemas tradicionais de computação, como, por exemplo, os de Modelgem.

No sexto e último capítulo, procuramos fazer uma análise crítica dos resultados obtidos, assim como apresentar sugestões para algumas otimizações do sistema *e* para futuros projetos de pesquisa baseados na Subdivisão Simplicial Adaptativa.

"...you are young and life is long and there is time to kill today..."

Capítulo I

1. MODELAGEM DE SÓLIDOS

1.1. INTRODUÇÃO

A Modelagem de Sólidos e um dos ramos da Computação Gráfica que mais tem crescido nos últimos anos. Com aplicações em várias áreas da ciência, ela vem se destacando pelo auxílio prestado ao Homem na elaboração de projetos de Engenharia e Arquitetura, na construção de peças mecânicas e fabricação de circuitos eletrônicos, na modernização da Medicina, na simulação de fenômenos físicos como, por exemplo, na análise do fluxo aerodinâmico em protótipos de carros de competição ([BECK90]), em robótica e em outras inumeráveis aplicações.

Inicialmente descreveremos algumas de suas principais características, citando algumas de suas propriedades e, em seguida, faremos um apanhado geral das linhas de pesquisa que têm sido exploradas durante o seu desenvolvimento. Finalmente, falaremos brevemente sobre seus principais paradigmas, em especial sobre B–Rep, Decomposição Espacial e CSG. Neste trabalho, contudo, estaremos basicamente interessados apenas nos dois últimos.

1.2. A ÁREA DE MODELAGEM DE SÓLIDOS

Entende–se por Modelagem de Sólidos ([REQU83]), um corpo de teorias, técnicas e sistemas focalizados em representações completas de sólidos que permitam, ao menos em princípio, o cálculo de qualquer pmpriedade geométrica bem definida de qualquer sólido representável. A noção de sólido deve capturar matematicamente algumas propriedades ([REQU80]) como, por exemplo: a invariância da forma independentemente da localização e orientação, a existência de interior, a ausência de porções isoladas ou penduradas na

fronteira e a ocupação finita do espaço. Além disto, movimentos rígidos (translações, rotações e escalas) e operações que adicionam ou removem material através da união, interseção e diferença entre sólidos (operações booleanas regularizadas) quando aplicadas a um sólido devem produzir novos sólidos.

É desejável, embora nem sempre possível, que os esquemas de representação de sólidos possuam algumas propriedades formais como:

(a) Domínio ou Poder de Expressão:
conjunto das entidades representáveis no esquema.

(b) Validade:
cada entidade deve estar relacionada com objetos que fazem sentido no mundo real.

(c) Completude:
cada entidade deve possuir informações suficientes para poder-se computar qualquer propriedade geométrica bem definida.

(d) Unicidade:
cada entidade representável deve possuir uma única representação.

Os modelos devem possuir, também, algumas outras propriedades informais que têm grande importância prática como:

(e) Concisão:
as representações devem requerer um conjunto pequeno de informações.

(f) Facilidade de criação:
a tarefa de criação de nova entidade para representar determinado objeto deve ser simples.

(g) Eficácia no contexto das aplicações:
as entidades devem ser suficientemente robustas para serem utilizadas por ampla gama de algoritmos para as mais diversas finalidades.

Como exposto em [REQU82], a modelagem de sólidos começou a tomar forma nos anos 60 com muito pouco suporte teórico. No início dos anos 70, a medida que pmblemas patológicos iam sendo identificados durante o desenvolvimento de programas, grupos de mini-teorias iam sendo construídos para obtenção de soluções imediatistas. Este padrão de comportamento dificultou a organização de um corpo de fundamentos teóricos adequado a maioria das aplicações. O interesse pela modelagem de sólidos crescia rapidamente, mas a comunidade científica se via envolvida em uma espécie de "Torre de Babel", que causava diferentes problemas e desentendimentos que afugentavam usuários potenciais. No final dos anos 70, esse corpo teórico começou a ser moldado com rigor matemático e relevância a aspectos computacionais outrora menosprezados. Atualmente já existe vocabulário comum, onde termos definidos matematicamente como "operadores regularizados" e "componentes conexas", além de pmpriedades das representações como "completude" e "validade" são igualmente compreendidos em quase todos os nichos de computação gráfica que lidam com modelagem.

1.3. SISTEMAS MODELADORES DE SÓLIDOS

O modelador de sólidos pode ser visto ([REQU83]) como a junção de um conjunto de entidades representativas de sólidos com uma coleção de algoritmos necessários a construção e manutenção destas representações. As pesquisas realizadas para o aprimoramento dos modeladores podem ser dispostas nas categorias descritas abaixo.

1.3.1. EXTENSÃO DO PODER DE EXPRESSÃO

A maioria dos modeladores trabalha apenas com sólidos passíveis de serem definidos por superfícies planas, quádricas ou, no máximo, toroidais. Embora estas superfícies sejam capazes de representar vasta gama de objetos, satisfazendo, pois, a maioria das aplicações, algumas vezes tomam-se necessárias deformações ou sinuosidades na casca do sólido que só podem ser alcançadas mediante o emprego de outros tipos de superfícies (Bézier, B-Splines, algébricas de grau mais elevado etc).

1.3.2. TÉCNICAS DE AQUISIÇÃO DE INFORMAÇÕES

Técnicas para construção de sólidos a partir de projeções (ou plantas) vêm sendo objeto de intensos estudos, visto serem estas as formas de descrição de sólidos mais comuns entre projetistas, engenheiros e arquitetos. A automação por completo desta tarefa, entretanto, constitui-se em problema de difícil solução uma vez que projeções são, em geral, ambíguas e a presença humana se faz necessária para a seleção da interpretação correta. Outra abordagem que vem sendo igualmente pesquisada parte de restrições geométricas (especificação de distâncias e ângulos) para construção e modificação dos sólidos. Os problemas aqui também são vários e bastante delicados, relacionados com a validade e a ambiguidade das representações. Alguns casos já se mostraram, inclusive, computacionalmente intratáveis. Maiores informações em [HARA82], [LIGH82] e [ALDE83].

1.3.3. REPRESENTAÇÃO DE TOLERÂNCIA

Em geral, não são oferecidas muitas facilidades para o tratamento de tolerâncias pelos modeladores. A consideração de incertezas numéricas são essenciais nas atividades de produção como, por exemplo, na utilização de máquinas de controle numérico (NC *Machines*) acopladas a sistemas de modelagem.

1.3.4. AUMENTO DA VELOCIDADE COMPUTACIONAL

A melhoria da qualidade dos programas associada ao avanço tecnológico formam, obviamente, a principal plataforma de pesquisa para o ganho de velocidade computacional. Os algoritmos têm procurado explorar aspectos de similaridade, isto é, regiões próximas na superfície do objeto tendem a possuir as mesmas características. No que se refere ao *hardware*, as arquiteturas especializadas e as máquinas paralelas (com memória distribuída ou compartilhada] despontam como as principais esperanças de futuros ambientes de modelagem fáceis de se utilizar, eficientes, robustos e, principalmente, rápidos o suficiente para permitirem verdadeira interatividade. Ver [CLAR82], [TORB87] e [AKEL88].

1.3.5. INTERSEÇÕES DE CURVAS E SUPERFÍCIES

Muitas das aplicações de modelagem exigem mecanismos rápidos e robustos para cálculos de interseções entre curvas e/ou superfícies. Quando os modeladores trabalham essencialmente com quádricas, existe a possibilidade destes cálculos serem feitos analiticamente. Para domínios mais gerais, no entanto, a degradação da velocidade e a instabilidade numérica podem ser muito grandes já que as soluções destas interseções consistem, geralmente, da solução de enormes sistemas não–lineares. Maiores informações podem ser obtidas em [KAJI82] e [HANR89].

1.3.6. A TAREFA DE MODELAGEM

Apesar da crescente evolução da tecnologia com o desenvolvimento de novos e melhores equipamentos e algoritmos, a construção do modelo representativo do objeto real, isto é, a modelagem propriamente dita, ainda se constitui em grave problema. Fornecidos operadores geométricos e topológicos, primitivas, possibilidades de transfomações destas etc, permanecem as questões: como facilitar, para o usuário, a construção de um modelo para o objeto que ele tem em mente ? Como evitar que as vantagens oferecidas por operações booleanas, técnicas de mapeamento de textura etc caiam por terra quando utilizadas através de rudimentares interfaces com o usuário ?

Estas dificuldades podem ser atribuídas, em grande parte, a ausência de interatividade satisfatória no contexto dos sistemas e equipamentos disponíveis. Como exemplo, analisemos a construção da figura 1.1. que contém uma cena com a união regularizada de duas esferas e um bloco. Os sólidos foram posteriormente processados, de modo a se obter imagem realista.

Durante a construção, seria bom que o usuário pudesse interativamente designar como objetos de trabalho esferas com centros e raios inicialmente indeterminados. Com o

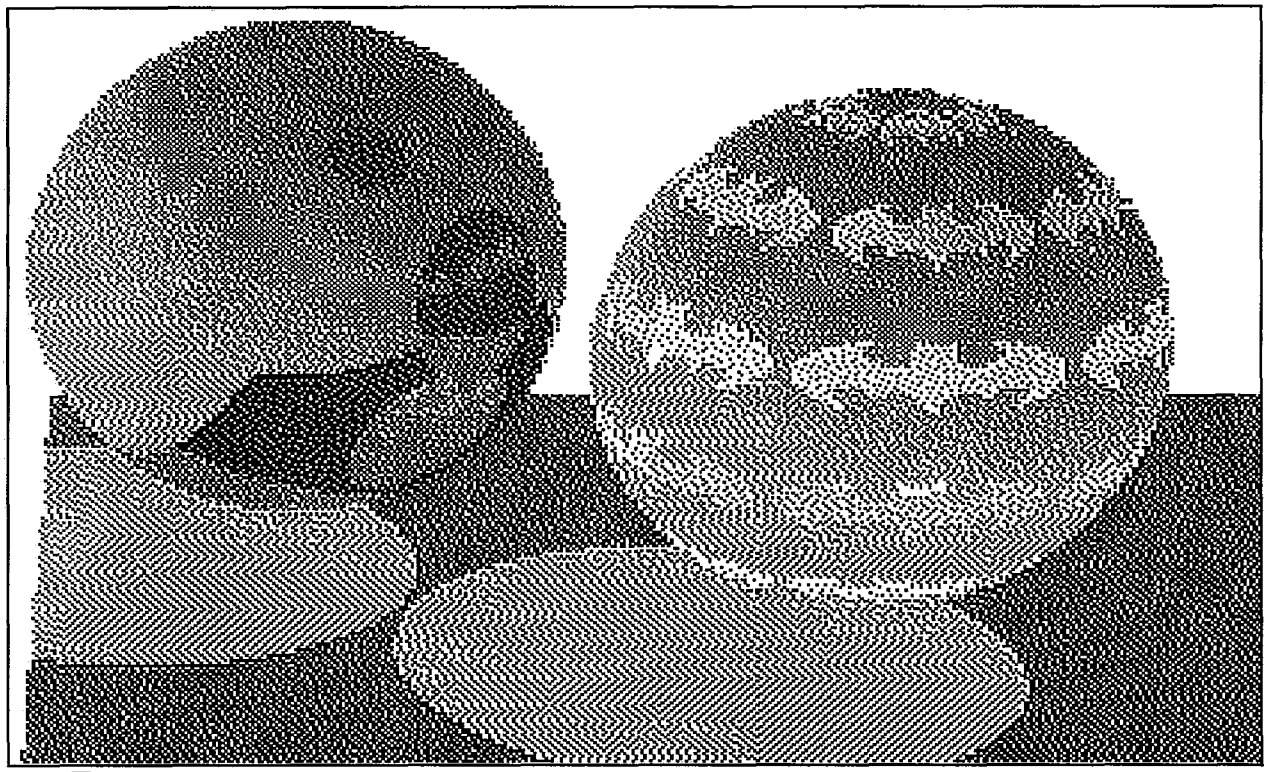


Fig. 1.1: cena trabalhada contendo a união regularizada de três sólidos.

auxílio do dispositivo localizador ele movimentaria cada uma das esferas, definindo posições e tamanhos e, finalmente, escolheria a operação a ser executada sobre o conjunto, no caso, operação booleana regularizada de união. Seria igualmente desejável que em cada passo da construção ele tivesse razoável resposta do sistema, isto é, sempre soubesse o que o passo que ele acabou de dar fez com o modelo e, obviamente, pudesse desfazê–lo.

Sem o auxílio desta tecnologia, utilizamos a linguagem LDS ([ESPE90]) de especificação de sólidos CSG e, em seguida, com a ajuda de subdivisão espacial e de técnicas de acompanhamento de raios (*Ray-Tracing*), geramos a imagem de uma aproximação do modelo. A modelagem, desta forma, ficou muito mais complicada. Inicialmente foi necessário calcular o tamanho e a posição no espaço de cada uma das esferas e, posteriormente, codificar em LDS a descrição da cena, fornecendo as informações imaginadas. Nessa fase, não houve resposta suficiente do sistema para se saber o que estava acontecendo, muito menos a possibilidade de se desfazer algum passo errado. O usuário só teve contato com o produto de seu trabalho através de uma imagem quando esse já estava pronto, inteiramente acabado. Todas as alterações tiveram que ser feitas desde o início: imaginando, programando etc.

Convém notar que as questões relativas ao projeto de interfaces com o usuário não são privilégios dos sistemas de modelagem geométrica, já que sistemas de interesses diversificados como Banco de Dados, Computação Científica e Hipermídia compartilham do mesmo problema. Fornecer ferramentas suficientes para o estabelecimento de razoável comunicação entre o Homem e a máquina vem sendo, há bastante tempo, objeto de estudo de várias áreas da computação.

Tem-se procurado estabelecer padrões nos quais a simplicidade, consistência e eficiência são os principais objetivos. Recentemente surgiram os chamados Sistemas Gerenciadores de Interface com o Usuário (User *Interface Management Systems* ou UIMS) que apresentam como idéia central o encapsulamento da comunicação entre o usuário e o computador. Tipicamente, o UIMS e o sistema aplicativo estão ligados formando um único programa e se comunicando através de chamadas de procedimentos. Em implementação mais flexível, considera-se funcionamento análogo ao de um sistema operacional onde o UIMS é um processo a parte, separado dos aplicativos, oferecendo quando requisitado, os recursos disponíveis. A comunicação, feita neste caso entre processos concorrentes (ou paralelos), é realizada por troca de mensagens que podem ser efetuadas através de mecanismos de sinalização ou por intermédio de arquivos compartilhados. Esta implementação permite ao usuário comunicação simultânea com vários aplicativos, cada um com sua própria janela na tela do computador. Maiores informações sobre o tópico "Interface com o Usuário" podem ser obtidas em [TAKA85], [SHNE87] e [SING91].

Nem sempre a tecnologia ajuda! Em alguns casos o usuário despreparado utiliza modeladores com grandes sofisticações para a realização de tarefas simples, aumentando desnecessariamente sua carga de trabalho e o da máquina. Em outros casos, os construtores do modelador não detêm experiência (ou conhecimento) suficiente sobre as potenciais aplicações e fornecem uma série de recursos supérfluos deixando de oferecer algumas ferramentas realmente úteis, entregando ao usuário verdadeiros "elefantes brancos".

Introdução consistente a Modelagem de Sólidos pode ser encontrada em [REQU80], [REQU82], [REQU83], [MANT90] e [MILL90]. Passamos, em seguida, a descrição sumária de seus principais paradigmas.

1.4. REPRESENTAÇÃO POR FRONTEIRAS (B-Rep)

A origem dos sistemas de modelagem de sólidos baseados em representação por fronteiras remonta aos primórdios da Computação Gráfica, quando eram utilizados aramados (conjuntos de pontos e linhas localizados convenientemente no espaço) para representarem os objetos do mundo real. Era a época dos chamados Modelos *Wireframes*.

Embora estes modelos, em certo senso, facilitassem a construção de determinadas figuras, eles não eram adequados para operações mais gerais como, por exemplo, o cálculo de volume ou da área de superfície do objeto. Até mesmo a visualização de determinados objetos, tarefa teoricamente simples para estes modelos, ficava complicada quando se fazia necessária alguma sofisticação como a eliminação das linhas ocultas ao observador Além disto, esse tipo de representação é ambígua na medida em que cada modelo não designa unicamente um objeto do mundo real.

Na tentativa de solucionar os problemas mais sérios dos modelos *Wireframes*, surgiram os modelos B-Rep (*Boundary Representation*), nos quais além das informações geométricas, são também armazenadas informações topológicas a respeito da fronteira do

objeto. Matematicamente falando ([MANT90]), os modelos B–Rep descrevem a geometria da fronteira topológica do sólido, subdividindo–a em uma coleção de superfícies (faces). Cada superfície é descrita através de um conjunto de curvas (arestas das faces) que a limitam. As curvas, por sua vez, são limitadas por pontos (vértices das arestas). As relações de adjacência são explícitas entre faces, arestas e vértices. As representações B–Rep podem ser entendidas, portanto, como uma hierarquia de modelos. A figura 1.2 ilustra o esquema de representação B–Rep para um cubo.

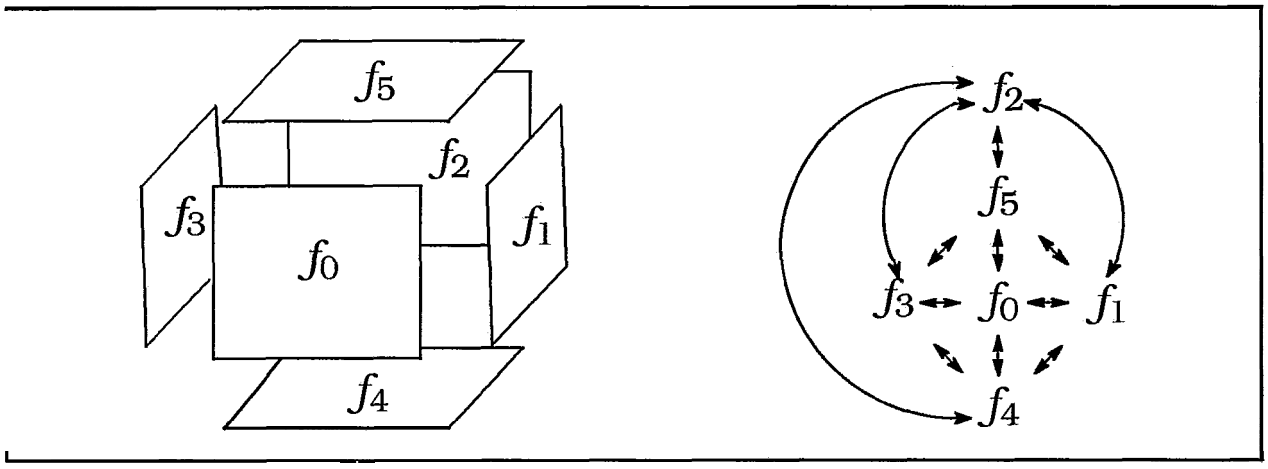


Fig. 1.2: representação B–Rep de um cubo.

Normalmente as faces são escolhidas de maneira que sua geometria possa ser representada por uma única forma paramétrica ou expressão algébrica. É justamente a diversificação das geometrias de faces passíveis de serem representadas que define o poder de expressão dos modeladores B–Rep. Não são todas as coleções de faces, no entanto, que definem um sólido fisicamente válido. A integridade topológica de um modelo B–Rep impõe restrições as faces tratáveis.

Para acelerar determinadas operações, os modeladores B–Rep armazenam explicitamente (com maior ou menor grau de redundância) diversas informações na estrutura de dados responsável pelo arranjo combinatório relativo das faces, arestas e vértices da fronteira do objeto. Estas informações são comumente chamadas de "topologia" enquanto que informações como equações da face ou coordenadas dos vértices são chamadas de "geometria". Com respeito as informações topológicas armazenadas existe grande variedade de formas de representação. A escolha entre estas informações e o grau de redundância com que elas são armazenadas depende das combinações entre operações desejadas para o modelo.

A representação por fronteiras exige estruturas de dados sofisticadas e, em geral, muito complexas, fazendo com que o gerenciamento seja trabalhoso e propenso a vários tipos de erros. Para facilitar esta tarefa, as operações complexas sobre estes modelos podem ser partidas em uma série de instruções primitivas chamadas de "Operadores de Euler", que trabalham sobre a topologia do modelo. Estes operadores não são intuitivos e, portanto, não devem ser utilizados diretamente pelo usuário. Eles são de grande auxílio, entretanto, na manipulação do modelo pois permitem sua construção passo a passo,

escondem a estrutura de dados efetivamente usada, favorecendo sua portabilidade, permitem armazenamento mais eficiente e, principalmente, garantem a consistência topológica ([MANT82] e [MANT84]).

Como vantagens dos esquemas de representação baseados em B–Rep podemos apontar a simplicidade dos algoritmos de exibição, armazenamento e recuperação de cenas e a possibilidade dos modeladores trabalharem diretamente com máquinas de controle numérico, sugerindo muitas aplicações em automação. Como desvantagens podemos citar o elevado consumo de memória para manipulação dos modelos e a alta complexidade de determinadas operações entre eles, como, por exemplo, as operações booleanas.

1.5. MODELOS DE DECOMPOSIÇÃO

Os modelos de decomposição descrevem o sólido como um arranjo de células do espaço cuja união o aproxima. A forma das células juntamente com os tipos de arranjos utilizados caracterizam as várias versões deste esquema de representação. A "Decomposição Regular (DR)" (também chamada de "Enumeração Exaustiva" ou "Decomposição Cúbica") é um exemplo, onde o espaço é totalmente subdividido em partes de mesmo tamanho. O sólido é visto como a união de voxels (Volume *Elements*) cúbicos, arbitrariamente pequenos, que estão no seu interior (total ou parcialmente).

A despeito de sua natureza aproximativa, o poder de expressão dos modeladores baseados em decomposição regular do espaço é bem amplo. Quase todos os tipos de objetos podem ser representados e a validade dos modelos depende exclusivamente dos tipos de interseção permitidos entre os cubos da decomposição. Se dois cubos quaisquer de uma determinada decomposição só se interceptam em faces, arestas ou vértices então os modelos são todos válidos (não necessariamente simplesmente conexos), únicos e não–ambíguos, embora, pouco concisos. Mais informações sobre esta forma de decomposição podem ser obtidas em [SHIR81].

Uma maneira independente de descrição de sólidos, para modeladores baseados em DR, deve consistir em uma lista de todos os cubos considerados partes integrantes do sólido. Isto pode ser facilmente obtido com a utilização de varredores digitais (*Scanners*) e alguns outros equipamentos utilizados no processamento de sinais tais como os tomógrafos digitais ([UDUP83]). Em Computação Gráfica, porém, raramente tem–se pontos de partida tão adequados e a abordagem mais prática é criar o modelo através da conversão de outro meio de representação e, a partir daí, usar técnicas apmpnadas para manipulá–lo. A DR, por conseguinte, constitui–se em excelente ferramenta para acelerar determinadas operações em outros esquemas de representação.

A principal razão para se utilizar os modelos de decomposição é que, geralmente, representar o objeto como um todo é mais difícil do que representar apenas as partes que o constituem. Apesar de herdarem algumas vantagens do clássico paradigma da "Divisão e Conquista" (Divide *and Conquer*), estes modelos apresentam algumas desvantagens como

a ausência de relação explícita entre as partes do objeto e a grande demanda de memória necessária ao armazenarnento das informações.

O alto consumo de memória se deve, em grande parte, a não exploração conveniente de similaridade do objeto, isto é, determinadas porções do espaço tendem a ser constituídas de um mesmo material, não havendo a necessidade de serem armazenadas individualmente. Esta total independência entre as regiões faz com que a quantidade de voxels necessários a representação seja proporcional ao volume do objeto.

Visando diminuir esse gasto excessivo de memória, foi proposta uma subdivisão adaptativa do espaço e, neste contexto, destacam–se as árvores octais (Octrees). O espaço passa a não ser mais subdividido regularmente em um conjunto de voxels. Apenas as regiões que contém parte da fronteira do objeto é que são esquadrinhadas até um nível mínimo ([JACK80] e [MEAG82]).

Os modelos Octrees consideram o sólido inserido em um cubo de dimensão 2^n x 2^n x 2^n chamado de cubo universo ou espaço Octree. A representação se dá pela subdivisão recursiva do cubo universo em voxels cúbicos cada vez menores. Cada um com dimensões 2^d x 2^d x 2^d ($1<d\leq n$) é subdividido em oito filhos com dimensões 2^{d-1} x 2^{d-1} x 2^{d-1}. Um critério é estabelecido para que o processo de subdivisão continue ou seja interrompido. Cada voxel, recém subdividido, é testado de forma a se saber em qual dos três seguintes casos ele se encaixa:

- totalmente contido no sólido
- completamente fora do sólido
- parcialmente contido no sólido

Nas duas primeiras situações o voxel e rotulado como CHEIO ou VAZIO, respectivamente, e o processo de subdivisão é interrompido. Na terceira situação, o voxel é rotulado como MISTO e a subdivisão recursiva continua, gerando seus oito filhos. Um nível mínimo de subdivisão é estipulado, onde estão os chamados voxels mínimos. Neste nível, o processo de subdivisão para incondicionalmente e os voxels mínimos que forem MISTOS são rotulados como CHEIOS ou VAZIOS arbitrariamente. A figura **1.3** ilustra o modelo Octree.

Assim como na DR, nos modelos Octrees a discretização favorece o cálculo de propriedades do sólido como massa, volume e área de superfície; assim como, ajuda na avaliação de operações booleanas. A ordenação espacial intrínseca, simplifica o problema da eliminação dos voxels ocultos ao observador e, consequentemente, simplifica a exibição. A quantidade de memória necessária é consideravelmente menor do que na DR, pois o número de voxels passa a ser proporcional a área da superfície do objeto, porém, continua sendo demasiadamente grande. Existem, também, dificuldades nos cálculos de transformações afins relativamente simples como translações e rotações e a qualidade da imagem do objeto discretizado, em geral, deixa muito a desejar.

A despeito de todos os problemas, com a queda do preço das pastilhas de memória e dos dispositivos de armazenamento secundário, as Octrees têm se tomado bastante

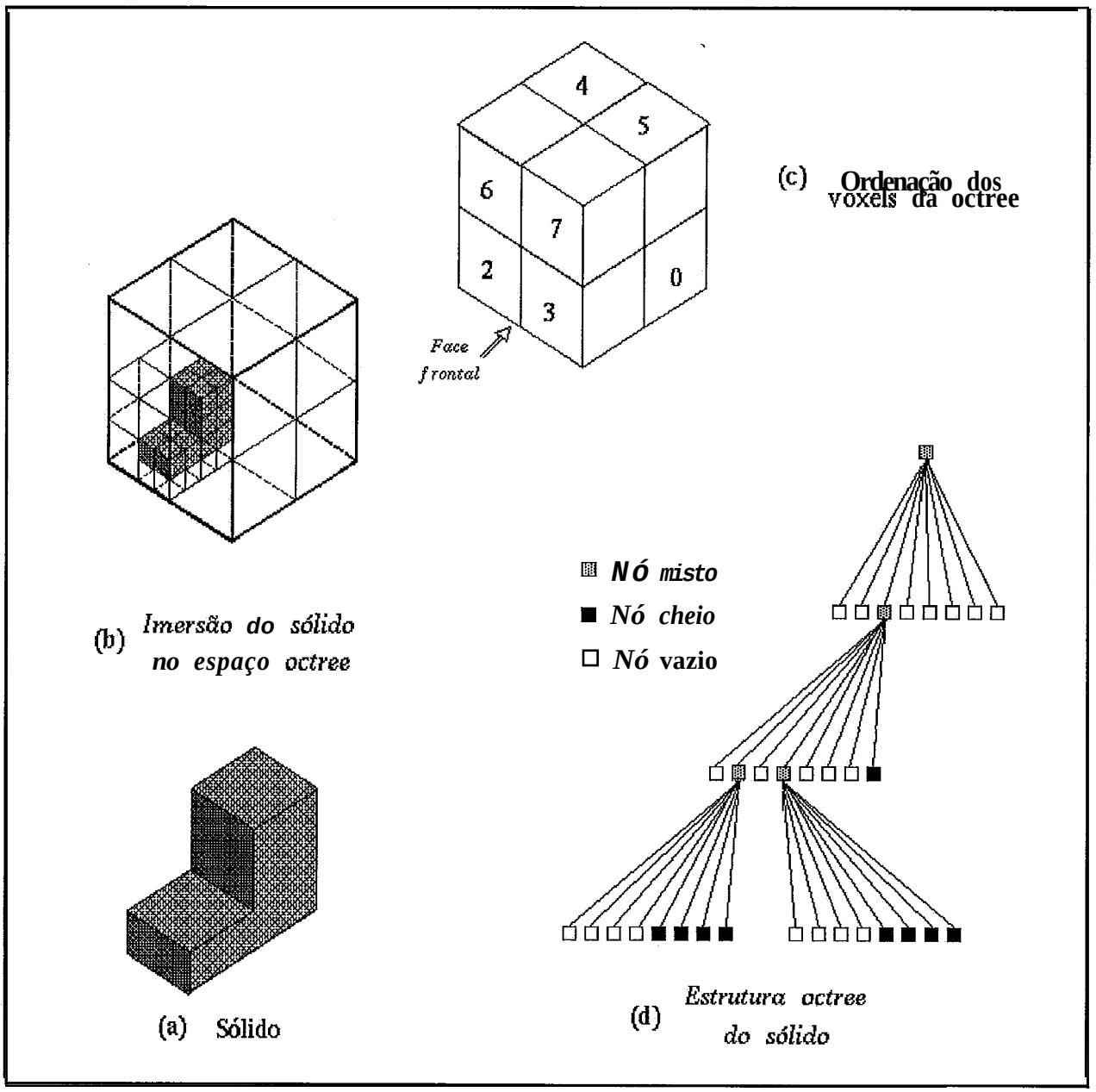


Fig. **1.3:** modelo octree.

populares nos últimos anos e, alem disto, sua estrutura em árvore favorece a implementação dos algoritmos em arquiteturas paralelas.

Ainda no escopo dos modelos de decomposição, existem os modelos de decomposição celular ([ELLI78]), no qual o objeto é subdividido em células cujas faces podem ser curvas. Na verdade, as células só precisam ser topologicamente equivalentes a uma esfera, ou seja, sem buracos em sua superfície. Estes modelos podem ser vistos como generalizações das triangulações (que veremos mais adiante) onde os tetraedros (3-simplexos) são substituídos por 3-células, que podem ter número arbitrário de faces. As decomposições celulares formam excelente base para a utilização de métodos de elementos finitos ([HINT77]), bastante requisitados nos dias de hoje.

1.6. GEOMETRIA SÓLIDA CONSTRUTIVA

A maioria dos conceitos sobre *Constructive Solid Geometry* ou CSG, foram introduzidos pelo grupo ligado ao projeto de automação da produção da Universidade de Rochester. O ferramental teórico e computacional desenvolvido por este e por outros importantes grupos de pesquisa desde os anos 70 vem mantendo CSG entre os paradigmas mais utilizados na modelagem de sólidos.

A idéia básica dos modelos CSG ([REQU82]) é a de que sólidos complexos podem ser representados por adições e subtrações ordenadas de sólidos mais simples através de versões modificadas dos operadores booleanos. Estes esquemas de representação podem ser vistos ([MORT85]) como árvores binárias (raramente balanceadas) cujos nós terminais (folhas) são sólidos que podem ser primitivas (sólidos básicos previamente definidos) ou transformações de algum sólido primitivo (instanciações). Os nós não-terminais são, geralmente, operações booleanas a serem aplicadas em seus dois subnós (subsólidos). Podem ser, também, transformações geométricas (translações e rotações) a serem aplicadas a um dos subnós. Cada nó não-terminal representa o sólido resultante da transformação ou composição dos subsólidos imediatamente abaixo dele. A raiz da árvore representa o sólido final.

Modelos CSG mais gerais estendem o conceito de primitiva e consideram semiespaços como possíveis primitivas. A noção de semiespaço é muito importante, portanto, para melhor compreensão desses modelos.

Alguns conjuntos de pontos de E^3 (espaço euclidiano de dimensão 3) podem ser representados por uma função analítica f, definida em todo o espaço, que associa a cada tripla (x,y,z) um número real $f(x,y,z)$. Esta função define uma superfície de nível do espaço, dividindo-o em três regiões distintas: interior, exterior e fronteira.

$$\forall p \in E^3 \Rightarrow \begin{cases} f(p) < 0 \ ; \ p \in \textit{interior} \\ f(p) = 0 \ ; \ p \in \textit{fronteira} \\ f(p) > 0 \ ; \ p \in \textit{exterior} \end{cases} \qquad \textit{onde}: \ \begin{array}{l} p = (x, y, z) \\ f: E^3 \rightarrow R \end{array}$$

Os conjuntos definidos pelas desigualdades $f(p) \leq 0$ e $f(p) \geq 0$ são chamados de semiespaços. Os semiespaços, por se tratarem de conjuntos de pontos, são bastante adequados a utilização de operações booleanas e de transfomações rígidas. Alguns semiespaços são especialmente úteis pois auxiliam a construção de primitivas CSG de vasta utilização e de fácil manipulação, é o caso dos semiespaços planos ($ax+by+cz+d \leq 0$), cilíndricos ($ax^2+by^2-r^2 \leq 0$) e esféricos ($ax^2+by^2+cz^2-r^2 \leq 0$). A figura 1.4 ilustra a representação CSG de um cilindro limitado através da interseção de dois semiespaços planos e um cilíndrico.

Os conjuntos de pontos adequados as representações CSG são, de acordo com [REQU80], os subconjuntos compactos de E^3 que são regulares e semi-analíticos, denominados *r-sets*. Os r-sets ocupam uma porção limitada do espaço (limitados), contém suas fronteiras (fechados) e estas são bem comportadas (semi-analíticos). Eles são ditos

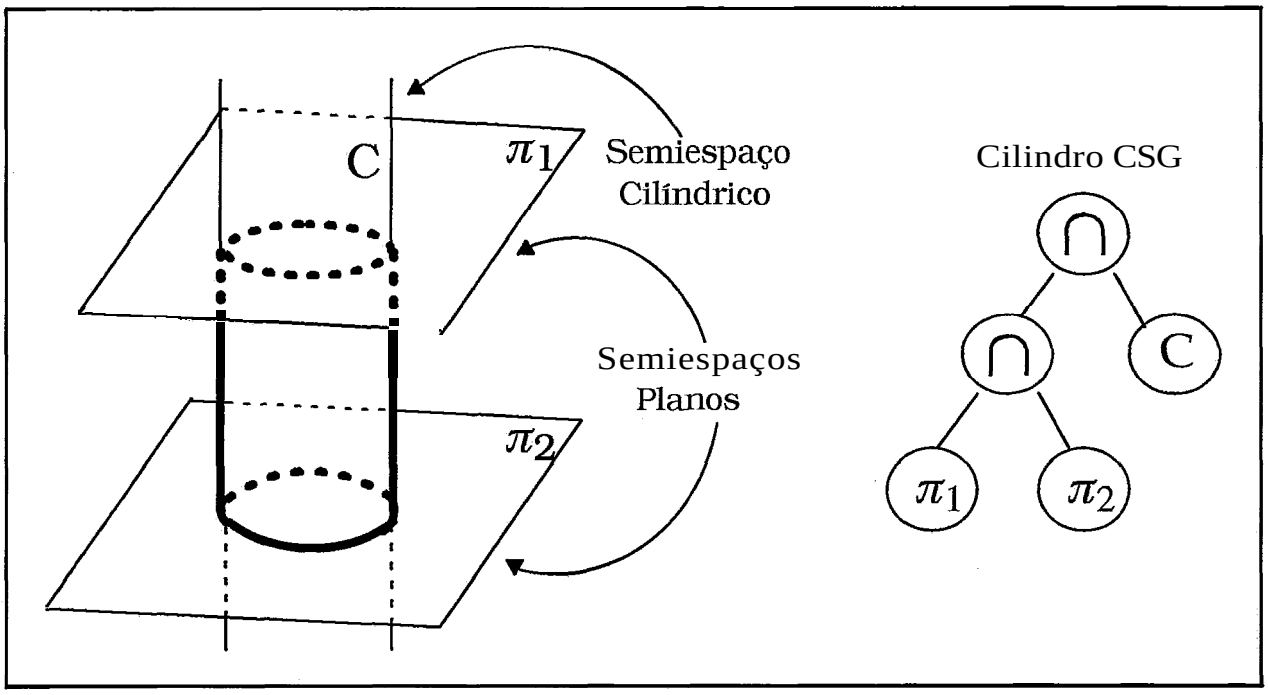


Fig. 1.4: representação CSG de um cilindro limitado.

regulares por serem homogeneamente tridimensionais, isto é, por não possuirem linhas ou superfícies penduradas (*dangling edges* ou *dangling* faces) em suas fronteiras. Estes conjuntos podem ser vistos como poliedms curvos com fronteiras bem comportadas.

Sob as operações booleanas convencionais, os r-sets não são algebricamente fechados, ou seja, a composição booleana convencional entre quaisquer dois r-sets não necessariamente gera outro r-set. Algumas composições nem mesmo satisfazem a noção usual de objeto sólido, apresentando porções não tridimensionais penduradas na fronteira. A figura 1.5(a) mostra uma composição problemática.

Para se evitar estes tipos de problemas introduziu-se o conceito de operação booleana regularizada, que consiste no seguinte: se A e B são dois subconjuntos de E^3 e Φ qualquer uma das operações booleanas convencionais, define-se:

$$A\ \Phi^*\ B = \text{Regularização}\ (A\ \Phi\ B) = \text{Fecho}\ (\text{Interior}\ (A\ \Phi\ B))$$

A regularização é uma espécie de filtro que retira do objeto as porções que não são homogeneamente tridimensionais, dando consistência aos resultados. A figura 1.5(b) mostra a correção da composição problemática do caso anterior através do uso de operações regularizadas.

Se os sólidos primitivos são limitados e válidos e os operadores booleanos são regularizados então os sólidos resultantes de qualquer combinação serão igualmente válidos e limitados ([MORT85]). Pode-se pensar em uma analogia na qual os operadores booleanos regularizados estão para CSG assim como os Operadores de Euler estão para B-Rep.

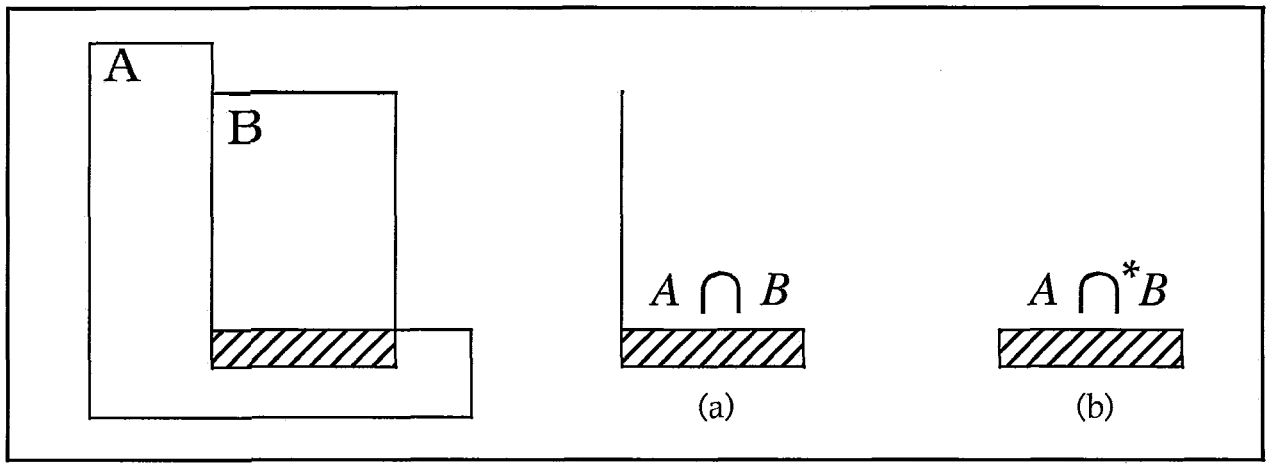


Fig. 1.5: interseções booleanas simples e regularizada.

No que diz respeito as propriedades dos modelos CSG, eles não são ambíguos porém não possuem unicidade. O seu poder de expressão depende da classe de semiespaços utilizada por suas primitivas, da variedade de operações booleanas disponíveis e das transformações geométricas permitidas. Aqueles modelos cujas primitivas estão bem de acordo com os objetos que vão ser modelados são bastante concisos e os modeladores baseados em primitivas limitadas são, em geral, mais concisos do que aqueles baseados em semiespaços genéricos, uma vez que requerem menos primitivas na fase de elaboração do modelo. Estes últimos modeladores, contudo, possuem flexibilidade bem maior.

Aparentemente, a tarefa de modelagem propriamente dita, fica mais simples e concisa quando se utiliza CSG, existindo a possibilidade, inclusive, da definição de linguagens formais de especificação e construção de sólidos como, por exemplo, a PADL desenvolvida pelos pesquisadores de Rochester e a LDS, já mencionada. Os sistemas baseados em CSG não são muito eficientes, no entanto, na realização de tarefas razoavelmente simples como produção de imagens diretas de suas representações assim como não apresentam muitas facilidades para a interatividade. Esta particular combinação de virtudes e defeitos faz de CSG um esquema de representação ideal para modeladores híbridos seja em parceria com B–Rep ou com Decomposição Espacial.

Há de se levar em conta, também, que os principais defeitos de CSG são devidos, em grande parte, aos algoritmos existentes para realização das tarefas (*Ray-Casting* por exemplo), pouco eficientes no contexto das máquinas sequenciais, todavia, bastante auspiciosos no que se refere a utilização de arquiteturas paralelas.

1.7. OUTROS ESQUEMAS DE REPRESENTAÇÃO

Existem, ainda, diversos outros paradigmas de modelagem de sólidos, todos com prós e contras suficientes para não serem rotulados de "melhor" ou "pior" em termos absolutos. Não nos estenderemos muito mais sobre eles para não fugirmos do escopo deste trabalho. Maiores informações podem ser obtidas nas referências já citadas, nas

quais se encontra material suficiente a respeito. Não obstante, não podemos deixar de mencionar os modelos baseados em varredura (*sweeping*) e em CNRG.

Os modelos baseados em varredura caracterizam o sólido como a soma vetorial de um conjunto que representa uma área por outro que representa uma trajetória, isto é, determinada área percorre uma dada trajetória esculpindo o sólido. Este esquema de representação, apesar da relativa facilidade de construção, apresenta reduzido poder de expressão. Quase que somente objetos que possuem simetria (translacional ou rotacional) podem ser representados. Algumas formas generalizadas de varredura possuem poder de expressão mais amplo mas, apresentam em contra-partida, inúmeras complicações matemáticas e computacionais. Estes modelos são bastante úteis como formas opcionais de entrada de dados pelo usuário, sendo internamente convertidos para outras formas de representação.

Algumas aplicações de CAD/CAM requerem modelos geométricos que não podem ser obtidos através da utilização de r-sets. Algumas vezes as operações regularizadas alteram características do sólido, deixando de satisfazer determinadas expectativas do usuário.

Na tentativa de minimizar estes problemas, surgiu recentemente na literatura o trabalho de [ROSS91] propondo a Geometria Construtiva Não-Regularizada (*Constructive Non-Regularized Geometry*). Os modelos CNRG representam o objeto como agregados de regiões disjuntas através de árvores (semelhantes as árvores CSG) que permitem formas primitivas não-regularizadas e que possuem, em seus nós internos, operadores-filtro que constroem objetos CNRG a partir de agregados de regiões selecionadas em outros objetos CNRG. Os esquemas baseados em CNRG são adequados a construção de objetos constituídos de materiais diferentes cuja fronteira varia de acordo com as características de cada material. Um exemplo e a construção de placas de circuitos integrados onde a representação da geometria de cada pastilha depende diretamente dos tipos de materiais ali presentes.

1.8. CONCLUSÃO

Apresentamos, neste capítulo, um sumário da área de Modelagem de Sólidos. Daqui em diante utilizaremos alguns dos conceitos expostos, principalmente o que se refere aos esquemas de modelagem baseados em CSG e, também, alguns dos aspectos relacionados com decomposição espacial, especialmente aqueles envolvendo octrees.

"...and ttíen one day you find..."

Capítulo 2

2. TÉCNICAS SIMPLICIAIS

2.1. INTRODUÇÃO

Os métodos simpliciais apoiam-se em decomposições regulares do espaço, mais especificamente em malhas regulares, e surgiram com o objetivo de aproximar o conjunto de soluções de sistemas de equações não-lineares. Um dos primeiros trabalhos neste sentido foi o de [SCAR67] que visava aproximar pontos fixos de mapeamentos contínuos. A partir de então, uma série de trabalhos correlacionados apareceram, estendendo os objetivos para aproximação de curvas na forma implícita, localização de pontos críticos, cálculo de interseções entre variedades implicitamente definidas etc. Entre estes trabalhos podemos citar [SAIG79], [ALLG80], [ALLG85], [ALLG87] e [TAVA90].

Neste capítulo apresentaremos as ferramentas necessárias para a utilização destes métodos, além de breve discussão sobre seu funcionamento. Apresentaremos, também, os esquemas de decomposição do espaço que lhes servem de base (decomposições celulares e triangulações), assim como um velho novo esquema de subdivisão espacial que achamos ser mais adequado em algumas situações: as decomposições hierárquicas.

Discutiremos, na sequência, alguns aspectos ligados a aproximação linear de funções definidas implicitamente que, a grosso modo, é o objetivo principal deste trabalho. Falaremos dos sistemas de coordenadas baricêntricas e de como podemos utilizá-los para definição de funções afins capazes de aproximarem essas funções definidas implicitamente.

Finalmente, apresentaremos o Método da Continuação, uma das mais utilizadas estratégias de processamento das triangulações para cálculo de aproximações afins de funções definidas implicitamente, onde faremos breve análise de suas qualidades e de suas deficiências.

2.2. CÉLULAS, SIMPLEXOS E TRIANGULAÇÕES

Para que se possa entender melhor como funcionam os métodos simpliciais, precisa-se compreender alguns conceitos básicos. É necessário que apresentemos uma notação mais conveniente, as definições e os resultados pertinentes, tal como em [MIRA89] e [CAST91].

Def. 2.1: Célula

A célula gerada por um conjunto de pontos $\{v_0, v_1, ..., v_k; v_i \in E^n\}$ é a combinação convexa destes pontos, ou seja, é o conjunto:

$$< v_0, v_1, \ldots, v_k > \; = \; \{ \, u \in E^n \; ; \; u = \sum_{i=0}^{k} \lambda_i \; v_i \; , \; \lambda_i \geq 0 \; , \; \sum_{i=0}^{k} \lambda_i \; v_i \; = \; 1 \, \}$$

O termo "simplicial" é derivado de uma classe especial de células que são vastamente utilizadas na solução de problemas de programação linear, não-linear e análise por elementos finitos: a classe dos simplexos.

Def. 2.2: Simplexo

Uma célula $\sigma = \langle v_0, v_1, \ldots, v_k \rangle$ é dita um simplexo k-dimensional (ou um k-simplexo) em R^n se os k vetores $u_i = v_i - v_0$ $(i=1,\ldots,k)$ forem linearmente independentes. Os pontos $v_i,\ldots,v_k$ são chamados de vértices do simplexo σ.

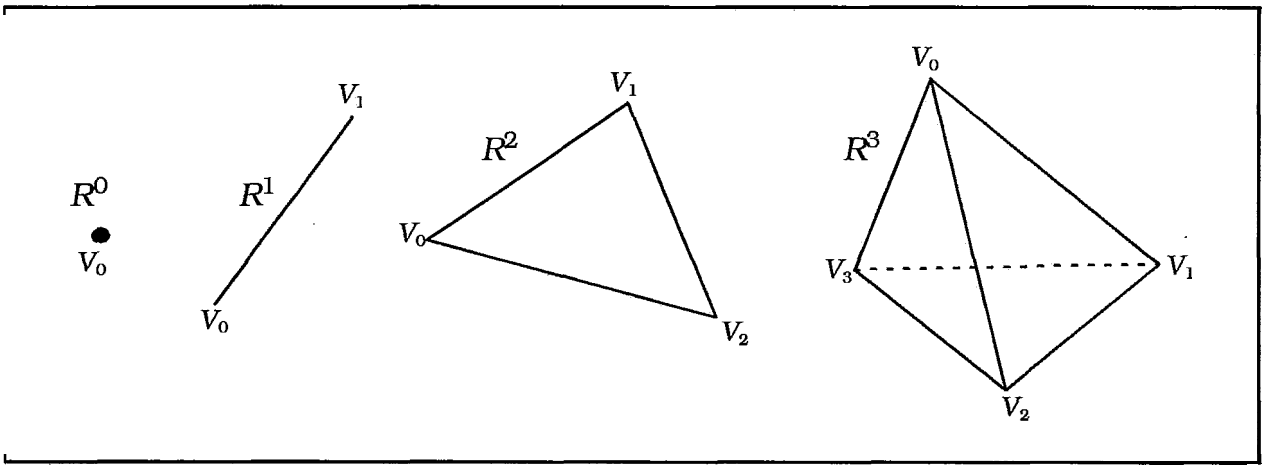


Fig. 2.1: simplexos de várias dimensões.

Def. 2.3: Subsimplexos

Dado um k-simplexo em R^n $\sigma = \langle v_0, \ldots, v_k \rangle$, os l-simplexos $(l \leq k)$ $\sigma_l = \langle v_0, \ldots, v_l \rangle$, formados por reagrupamentos de vértices de σ, são chamados de subsimplexos de σ de dimensão l. Particularmente no R^3, os subsimplexos $\langle v_i, v_j \rangle$ são chamados de arestas e os subsimplexos $\langle v_i, v_j, v_m \rangle$ de faces de σ. De maneira geral, os subsimplexos de dimensão l são chamados de 1-faces, subfaces ou, simplesmente, faces de σ. As (n-1)-facessão chamadas de facetas.

Os simplexos podem ser compreendidos como as formas poliédricas mais simples. A figura 2.1 fornece exemplos: os simplexos de dimensão 1 são os segmentos de reta, os de dimensão 2 são os triângulos, os 3–simplexos são os tetraedros e assim por diante.

Podemos passar agora para uma definição mais formal das decomposições celulares e das triangulações, citadas no primeiro capítulo, que se constituem na plataforma básica para o emprego das técnicas simpliciais.

Def. 2.4: Decomposição Celular

Uma decomposição celular de um subconjunto $D \subset E^n$ é um conjunto de células $C = \{c_i; i \in N\}$ com as seguintes pmpriedades:

i) $D = \cup c_i$

ii) Se $c_i, c_j \in C$ então $c_i \cap c_j = \emptyset$ ou é uma face comum a c_i e a c_j.

iii) Todo subconjunto compacto $K \subseteq D$ intercepta um número finito de células de C.

Def. 2.5: Triangulação

Uma triangulação de um subconjunto $D \subset E^n$ é uma decomposição celular de D onde todas as células são simplexos. Neste caso, D é chamado de um complexo simplicial.

As triangulações de E^n são coleções de n-simplexos que cobrem todo o espaço e cujos interiores são disjuntos. Elas estão entre as formas de decomposição do espaço mais utilizadas pelo fato dos simplexos serem as estruturas lineares mais simples, o que facilita a tarefa de exploração do espaço. Os cubos, por exemplo, tidos como formas lineares bastante simples e amplamente utilizados em decomposições do espaço tridimensional, podem ser subdivididos em famílias de poucos tetraedros enquanto que a recíproca não é verdadeira. Em verdade, na grande maioria dos casos, os simplexos são os *ladrilhos* ideais para os ladrilhamentos do espaço.

Dependendo do contexto das aplicações, as triangulações podem ser regulares, como as triangulações CFK, da qual falaremos mais adiante, ou não–regulares como as triangulações de Delaunay.

Em várias aplicações, que vão desde robótica ([BOIS89]) até os Sistemas de Informações Geográficas ([BURR86]), utilizam–se triangulações nas quais o espaçamento entre os vértices dos simplexos é variável, isto é, a decomposição é não–regular. O exemplo mais conhecido é o das Triangulações de Delaunay, que passaram a ser mais frequentemente utilizadas quando [BOOT74], aparentemente pela primeira vez, notou que elas trabalham no espaço dual ao dos diagramas de Voronoi (estruturas de dados fundamentais para a área de Geometria Computacional), facilitando sobremaneira o cálculo destes.

De acordo com [GUIB85], os diagramas de Voronoi/Delaunay podem ser utilizados para solução de uma série de problemas, por exemplo, na localização eficiente dos vizinhos mais próximos de determinado ponto em coleção de pontos aleatoriamente espalhados pelo espaço ([BOIS89]).

As triangulações de Delaunay também podem ser aproveitadas nos Modelos Digitais de Terreno. Neste caso, dispõe-se de amostras da topografia, mais especificamente, de amostras da elevação do terreno para determinado conjunto de pontos de uma região e deseja-se saber o valor da elevação (cota) de um ponto que não está no universo amostral. Eventualmente, deseja-se esboçar uma aproximação para o relevo em toda a região. Algoritmos para estas finalidades, interpoladores ou aproximadores, já existem em grande quantidade, a maioria utilizando Delaunay ([NEVE88], [SIMO89] e [GUTI91]).

As triangulações também podem ser regulares. Existe uma classe de triangulações, chamada de "classe das triangulações congruentes geradas por reflexões", onde todos os simplexos são congruentes e podem ser obtidos a partir de um único simplexo fixo, através de sucessivas reflexões pelas faces deste. A identificação das regras de reflexão é essencial e nem sempre é uma tarefa fácil. Esta classe de triangulações, de natureza regular, está bem caracterizada pelo trabalho de [COXE34], que fornece meios, inclusive, para se identificar a triangulação a partir do formato dos simplexos.

Uma das principais representantes desta classe é a Triangulação CFK que tem seu nome em homenagem aos três pesquisadores que, independentemente, a descobriram: Coxeter, Freundenthal e Kuhn. De acordo com [MIRA89], inicialmente Coxeter a descobriu durante estudos sobre grupos finitos de simetria do espaço e, posteriormente, Freundenthal redescobriu-a estudando a robustez de triangulações, finalmente, Kuhn utilizou esta mesma triangulação em nova proposta para localização de pontos fixos de funções contínuas baseada em [SCAR67].

A CFK pode ser obtida através de decomposição regular do espaço de interesse, seguida de triangulação consistente de cada parte da decomposição. Analogamente, poder-se-ia obtê-la tomando-se uma única parte da decomposição, triangulando-a e replicando-a (através de reflexões por suas faces) por todo o espaço. Esta triangulação é muito empregada nas já mencionadas tarefas de localização de pontos fixos de funções continuas ([KUHN68]), aproximação de soluções de equações diferenciais ([CAST90] e [FREI91]) e na aproximação de domínios tridimensionais implicitamente definidos ([WIDM90]). A figura 2.2 mostra dois exemplos de triangulações congruentes geradas por reflexões no plano, em particular, a triangulação da esquerda é CFK.

A discussão sobre a utilização de triangulações regulares ou não-regulares para solução de determinado problema é muito ampla mas deve levar em consideração dois importantes aspectos: conhecimento prévio de características do objeto de estudo e a manutenção de estrutura de dados eficiente associada a triangulação.

O conhecimento prévio de características do objeto sugere a construção de triangulações não-regulares, visto que estas permitem que determinadas regiões sejam examinadas, com maior ou menor intensidade, de acordo com as informações obtidas de antemão. Este caráter não-regular, em geral, obriga a manutenção de estruturas de dados complicadas para representação eficiente da triangulação. Em algumas situações, as informações impingem urna lei de formação não-regular, como no caso dos diagramas de Voronoi, no qual o conjunto de vértices dispostos no espaço é arbitrário.

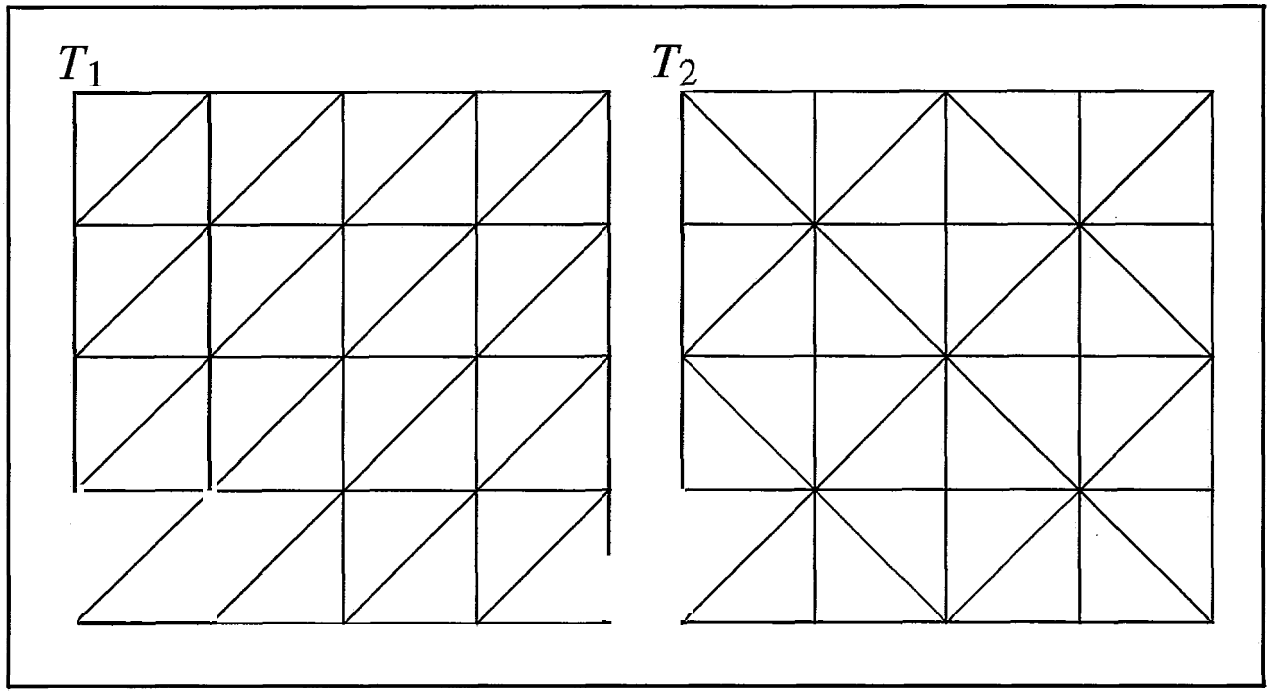


Fig. 2.2: triangulações geradas por reflexões no plano.

Os esquemas regulares, por sua vez, parecem mais adequados as situações em que não se têm muitas informações *a priori*. A subdivisão do espaço tem que ser feita de qualquer forma para que este seja completamente examinado e, assim sendo, melhor fazê–la de maneira regular.

Apesar das triangulações regulares serem, em princípio, menos robustas do que as não–regulares (até porque são casos particulares destas últimas), sua própria disposição no espaço possui uma representação implícita de simples manipulação, dispensando a utilização de estruturas de dados complicadas. A regularidade favorece, também, o cálculo de propriedades intrínsecas da triangulação ([ALLG80] e [DOBK90]) como a relação de adjacências entre os simplexos. Nas triangulações CFK, por exemplo, estas relações são obtidas através de um conjunto de regras, chamadas de "regras de pivoteamento", que trabalham somente com aritmética inteira ([MIRA89]), evitando consequentemente erros numéricos na localização dos vizinhos de determinado simplexo. Maiores informações podem ser obtidas em [CAST90] e [CAST91].

O trabalho de [TODD76] introduz uma série de medições de eficiência teórica de triangulações e afirma que se os simplexos forem descritos através de suas faces e estas por seus vértices, a validade da triangulação pode ser demonstrada sem muitas dificuldades assim como as regras de pivoteamento podem ser mais facilmente obtidas.

2.3. DECOMPOSIÇÕES HIERÁRQUICAS

Até agora vimos formas de decomposição do espaço, regulares e não–regulares, onde de um modo ou de outro o espaço de interesse é inteiramente subdividido. Esta é, em geral,

uma maneira razoavelmente segura de se encontrar todo o conjunto solução do problema que se está procurando resolver.

O exame exaustivo de todo o espaço é inevitável na maioria dos casos, mas se pode pensar em maneiras de tomá–lo mais eficiente. Uma delas é através das decomposições hierárquicas que, a semelhança das octrees, subdividem o espaço de maneira adaptativa, eliminando passo a passo regiões relativamente grandes que seguramente não contém elementos do conjunto solução desejado. Neste tipo de decomposição, procura–se aproveitar todas as vantagens do paradigma de "dividir para conquistar", trabalhando–se melhor com as partes ao invés de se trabalhar com o todo.

Def. 2.6: Decomposição Hierárquica
Uma decomposição hierárquica (DH) de um domínio $K \subset E^n$ pode ser uma célula ou uma decomposição celular de K onde, a cada célula é associada uma DH que tem esta célula por domínio.

Convém notar que as decomposições hierárquicas não são casos particulares das decomposições celulares, visto que podem haver pares de células de uma DH cuja interseçao não é vazia mas, tampouco, é uma subface comum. Estas decomposições não satisfazem, portanto, ao item (ii) da definição 2.4. A figura 2.3 nos dá exemplo de uma DH e ilustra interseções problemáticas.

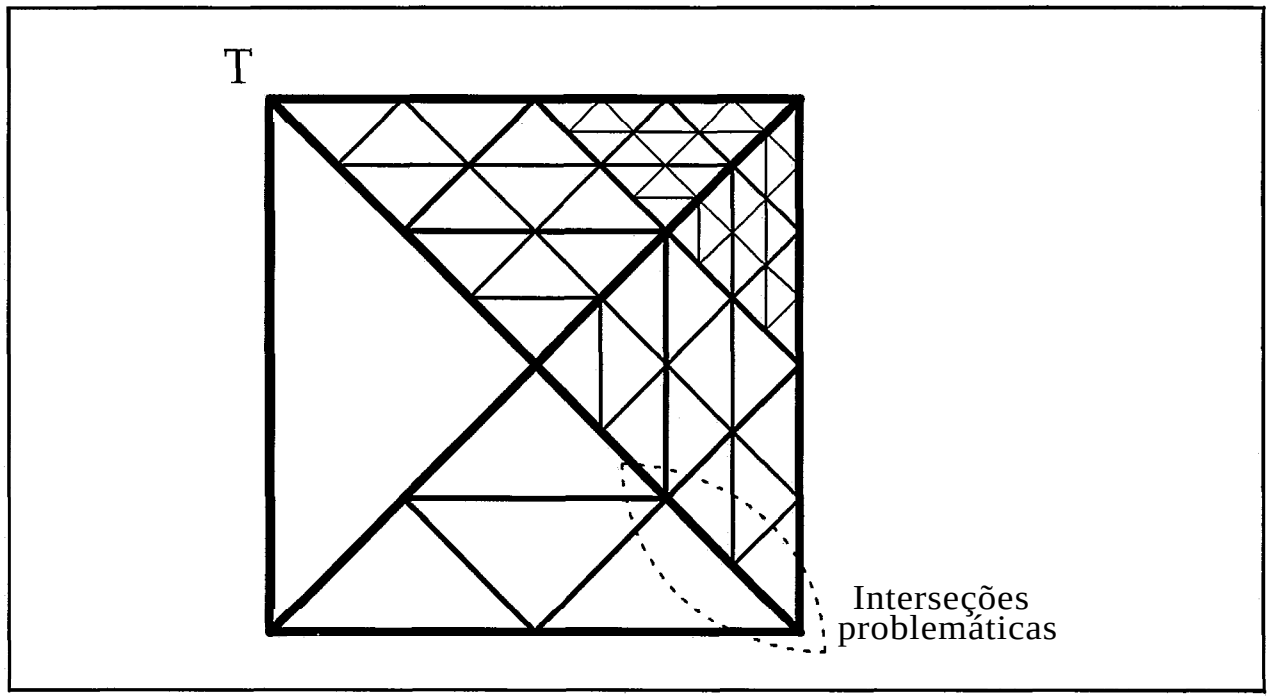


Fig. 2.3: decomposição hierárquica de uma região do plano.

Deve–se notar, também, que as decomposições hierárquicas, assim como as decomposições regulares, esquadrinham todo o espaço. A maneira como é feita a subdivisão é que pode torná–las mais "inteligentes" do que estas ultimas, na medida em que não subdividem o espaço todo de antemão, mas gradativamente, sempre que

estritamente necessário. Uma prova informal de que as duas subdivisões são equivalentes pode ser obtida, por exemplo, ao tentarmos resolver o problema da localização de uma das curvas de Peano, em uma região do plano, através de uma DH. Como estas curvas preenchem todo o plano, em nenhuma etapa da DH conseguir-se-á interromper o processo de subdivisão de qualquer célula antes que ele chegue até um nível mínimo, previamente arbitrado. Neste nível, a DH coincide exatamente com uma DR da região.

Em casos mais específicos, no entanto, a utilização das DH's é plenamente justificável, pois grande parte da região de interesse garantidamente não contem elementos do conjunto solução, podendo ser seguramente eliminada, otimizando consideravelmente o processo de localização de soluções do problema que se está estudando. Em [SALI91] tem-se um exemplo disto, onde procura-se aproximar variedades definidas implicitamente utilizando-se DH's.

2.4. COORDENADAS BARICÊNTRICAS

Os simplexos, em qualquer dimensão, podem ser vistos como bases geradoras do espaço. Qualquer ponto do R^n pode ser representado pelas suas "coordenadas baricêntricas", isto é, pelos coeficientes da única combinação afim de vértices de um simplexo de referência. Dado um simplexo $\sigma = \langle v_0,\ldots,v_n\rangle$:

$$\forall p \in R^n \Rightarrow p = \sum_{i=0}^{n} \lambda_i \, v_i \quad , \quad \sum_{i=0}^{n} \lambda_i = 1$$

A $(n+1)$-upla $(\lambda_0, \lambda_1,\ldots,\lambda_n)$ define as coordenadas baricêntricas de p em relação a σ e estas são, obviamente, invariantes por transformações afins.

Sem perda de generalidade, vejamos um exemplo desta representação no plano. Seja $\sigma = \langle v_i, v_j, v_k\rangle$ um 2-simplexo arbitrário e $\Lambda(p) = (\lambda_i(p), \lambda_j(p), \lambda_k(p))$ a representação baricêntrica (em relação a σ) de um ponto p qualquer do plano. Cada coordenada $\lambda_r(p)$ pode ser vista como uma função afim $\lambda_r: R^2 \rightarrow R$ definida de tal modo que $\lambda_r(v_r) = 1$ e $\lambda_r(v_s) = 0$ $(r \neq s)$. A figura 2.4 esclarece melhor.

Não é difícil verificar que, no interior do simplexo todas as coordenadas baricêntricas são estritamente positivas, que nos vértices somente uma coordenada é não-nula (igual a 1) e que fora do simplexo algumas coordenadas são negativas. Devemos notar, também, que embora a representação baricêntrica possua $(n+1)$ coordenadas em um espaço de dimensão n, a condição de que o seu somatório é unitário garante que somente n coordenadas são linearmente independentes. Maiores informações podem ser obtidas em [FARI88].

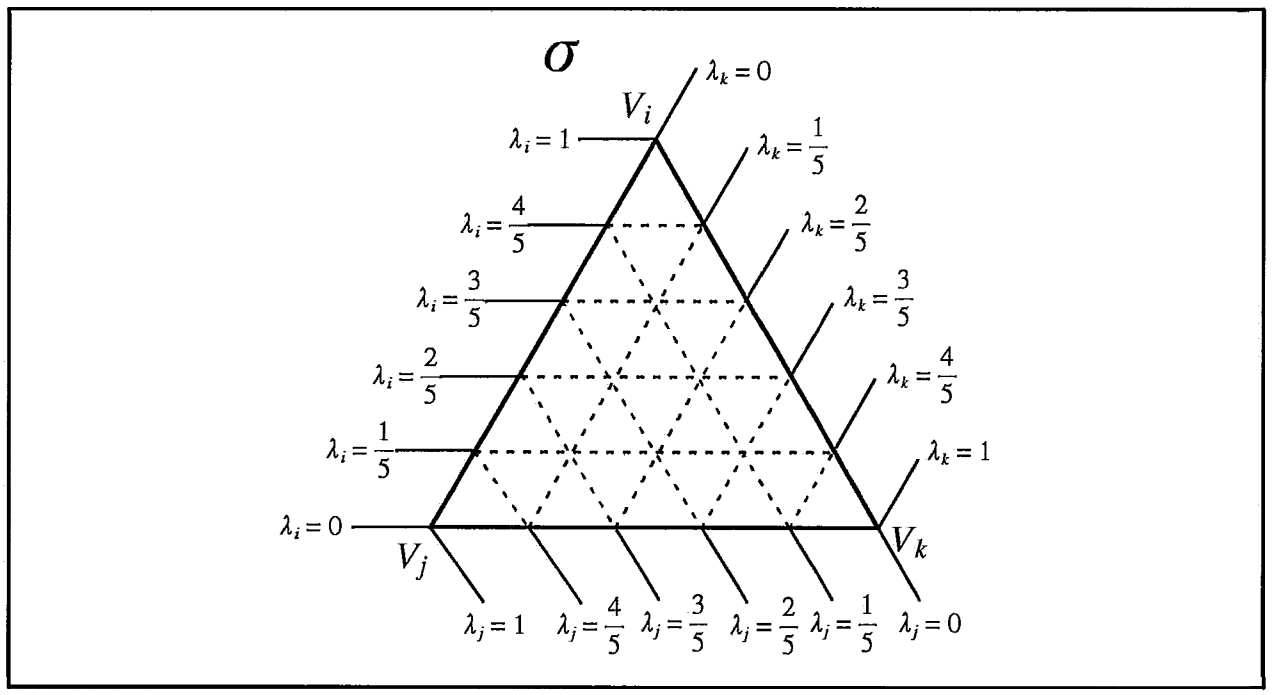


Fig. 2.4: coordenadas baricêntricas no plano.

2.5. INTERPOLADORES AFINS

Baseados nas representações baricêntricas de pontos, podemos definir uma classe de funções afins interpoladoras, que nos permitam computar uma aproximação afim de um mapeamento contínuo qualquer $f:R^n \rightarrow R$ dentro de o, bastando apenas o conhecimento dos valores deste mapeamento nos vértices de o. No escopo deste trabalho estamos particularmente interessados na solução destes problemas no R^3, isto é, buscamos aproximações afins para funções $f:R^3 \rightarrow R$ definidoras de superfícies no espaço tridimensional.

O pedaço de superfície contido em σ é o conjunto $S = \{q \in R^3; f(q)=0\} \cap \sigma$ cujo cálculo, que em geral consiste da solução de sistemas não-lineares, pode ser bastante custoso (dependendodo comportamento da função *f*), o que pode inviabilizar a tentativa de solução analítica. Ao invés de tentarmos computar o pedaço de superfície, procuramos uma aproximação afim expressa pelo conjunto $S_\sigma = \{p \in R^3; f_\sigma(p)=0\} \cap \sigma$ onde a função afim $f_\sigma:R^3 \rightarrow R$ depende apenas dos valores da função original nos vértices de o.

De maneira geral, dado um n-simplexo de referência $o = \langle v_0,\ldots,v_n \rangle$, definimos $f_\sigma:R^n \rightarrow R$ tal que:

$$f_\sigma\left(\sum_{i=0}^{n} \lambda_i \; v_i\right) = \sum_{i=0}^{n} \lambda_i \; f(v_i)$$

e para obtermos a aproximação afim da função f aplicada a um determinado ponto $p \in R^n$, calculamos as coordenadas baricêntricas de *p* com respeito a o, avaliamos f nos vértices de

o e computamos a combinação afim destes, ponderados pelas coordenadas baricêntricas obtidas. Estas funções são ditas interpoladoras por coincidirem com o valor da função original nos vértices do simplexo e isto é justificado pelo fato de, nos vértices, apenas uma coordenada baricêntrica ser não–nula.

Não existe nenhuma restrição, em princípio, sobre qual deve ser o simplexo o escolhido como sistema de referência para que a qualidade da apmximação seja boa. Esta análise de qualidade é, em geral, tarefa bem complexa e necessita de algumas restrições sobre a classe de funções tratáveis. Estudo sobre o comportamento da aproximação poderá ser razoavelmente feito, por exemplo, se as funções definidoras das superfícies forem lipschitzianas dentro de o. Em casos mais gerais, no entanto, não se pode afirmar muita coisa.

É razoável, no entanto, que sempre se tome o simplexo de referência de tal modo que ele contenha a região onde há interesse em computar as aproximações. As avaliações, desta forma, são feitas por interpolações ao invés de por extrapolações estando, portanto, menos sujeitas a erros numéricos. É o caso, por exemplo, dos Modelos Digitais de Terreno, onde a cada vértice da triangulação associa–se a cota do terreno e se deseja saber a cota de um outro ponto qualquer dentro da triangulação. Basta localizar o simplexo o que contém o ponto e utilizar a função interpoladora f_σ, especificamente definida para ele. A cota associada ao ponto será urna apmximação afim ponderada pelas cotas dos seus pontos vizinhos na triangulação. Deve–se tomar cuidado, nestes casos, com problemas ligados a topologia da aproximação.

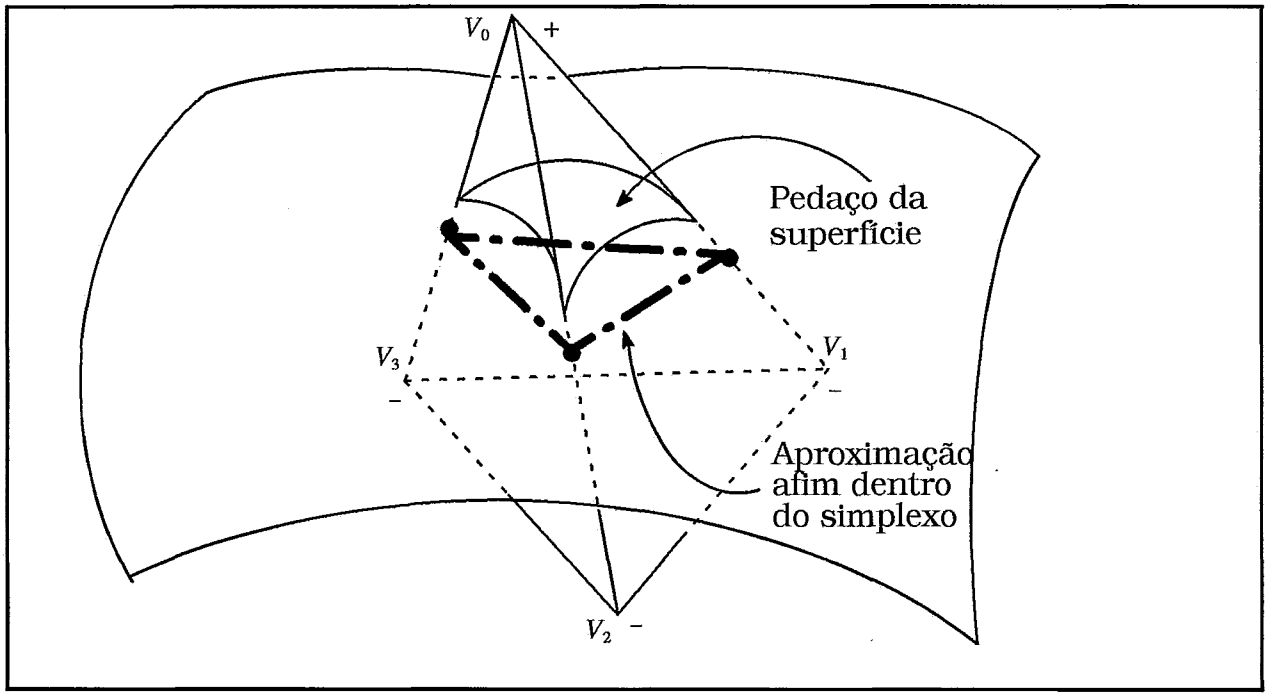


Fig. 2.5: aproximação afim de uma superfície dentro de um simplexo.

Voltando ao problema de aproximação afim de superfícies, o cálculo do pedaço de superfície que está no interior de o consiste na localização dos zeros da função f interiores

a o enquanto que o cálculo da aproximação linear deste pedaço se dá pela avaliação dos zeros, também contidos em o, da função afim f,. Como os simplexos são estruturas lineares, o conjunto S, = {p E R?; $f_\sigma(p)$=0} n o pode ser vértice, aresta ou poliedro cujos lados estão apoiados nas faces de o. Nos dois primeiros casos, o cálculo da apmximação linear é trivial pois ela consiste dos próprios vértices de o. No terceiro caso, a construção da aproximação também é facilitada pois estando os lados do poliedro sobre as faces de o, eles são segmentos de reta (em cada face) definidos pelas aproximações das interseções da superfíce com as arestas de o.

Simplificando, para computarmos a aproximação linear do pedaço de superfície contido em o é suficiente calcular apenas as aproximações das interseções da superfície com suas arestas. A lista de interseções, convenientemente ordenada, forma o poliedro aproximador. A figura 2.5 ilustra a situação.

Tal como mencionado no seção 1.3, a curva $f(p)$ = 0 divide o plano em três regiões disjuntas, onde ela assume valores positivos, negativos e nulos. Estamos interessados em aproximar esta última região através da utilização de f_σ. A figura 2.5 nos dá uma pista de como conseguir isto. Lá podemos observar que todos os simplexos que são interceptados pela superfície possuem vértices nos dois semiespaços definidos por f, ou seja, existem sempre vértices com valores de função positivos e outros com valores negativos.

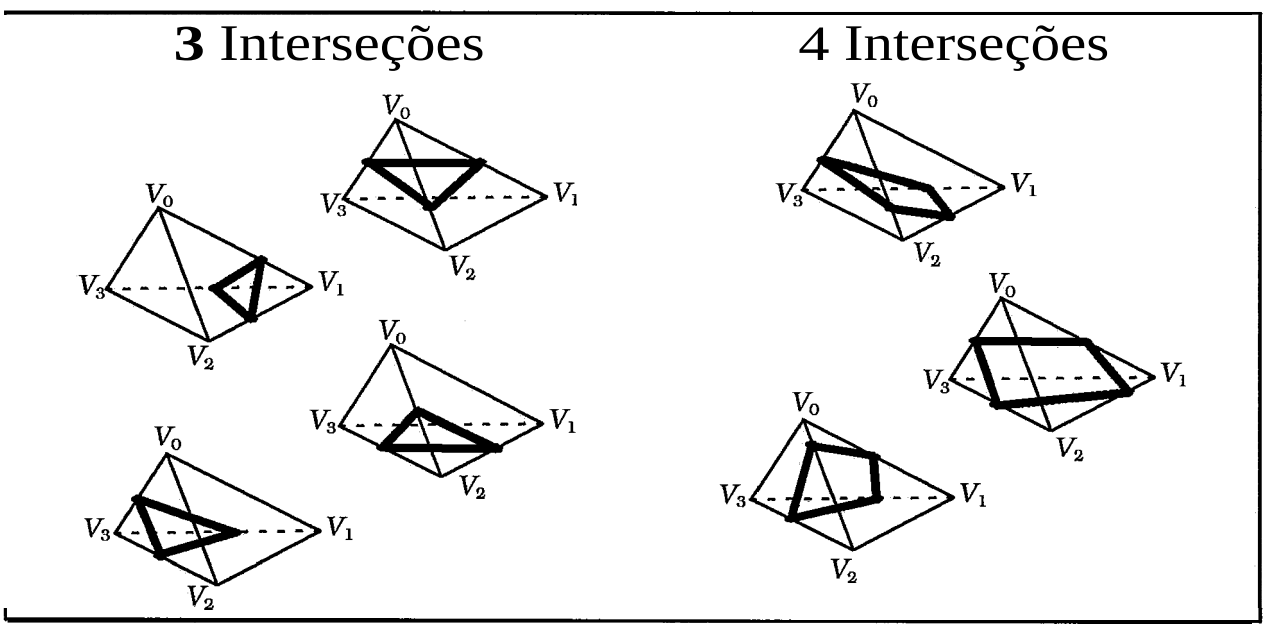


Fig. 2.6: aproximações lineares nos casos de 3 e de 4 interseções.

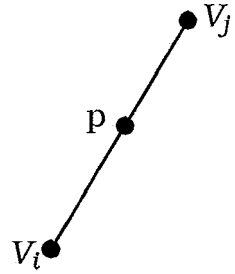
Esta variação de sinal nos fornece um primeiro critério de classificação dos simplexos. Podemos classificá–los em relação a função f em três tipos básicos, aos quais chamaremos de vazios, interceptados (interceptantes) e F–interceptados (F–interceptantes). Os simplexos vazios são aqueles que não fazem interseção com a superfície, os interceptantes, por outro lado, são todos aqueles que o fazem. Dentre os simplexos interceptantes estão os F–interceptantes que apresentam sinais de função distintos entre os vértices, ou seja, são aqueles interceptados por f_σ=0. A caracterização de

um simplexo como F–interceptante é muito importante, pois somente nestes pode–se computar uma aproximação linear satisfatória para as raízes de f. As subfaces dos simplexos também podem ser classificadas como F–interceptantes ou não, dependendo se elas apresentam sinais de função distintos em seus vértices.

As configurações possíveis de sinais de função nos vértices de um simplexo o F–interceptante garantem que, descartados os casos degenerados onde a aproximação linear coincide com um vértice ou com uma aresta, a superfície $f_\sigma(p)=0$ intercepta no mínimo três e no máximo quatro das arestas de o. A figura 2.6 mostra as apmximações lineares nestes dois casos.

A construção da aproximação linear dentro do simplexo, portanto, pode ser feita gradativamente, aresta por aresta. Naquelas que forem F–interceptantes, é calculada uma aproximação para a interseção com a superfície. Quando todas elas tiverem sido processadas, os pontos calculados definirão um poliedro plano que é aproximador afim da superfície no interior do simplexo.

O cálculo do ponto de interseção da aresta F–interceptante com a superfície é feito através de uma interpolação linear entre os vértices da aresta, tal como mostrado na figura 2.7. A interpolação, sugerida pela primeira vez em [MIRA89], utiliza como ponderadores os valores da função nos vértices, fazendo com que a aproximação da interseção fique mais próxima do vértice que possuir valor de função mais próximo de zero.



$$\lambda = \frac{F(V_i)}{F(V_i) - F(V_j)}$$

$$p = (1-\lambda)\ V_i + \lambda\ V_j$$

Fig. 2.7: cálculo da aproximação afim da interseção objeto–aresta.

Um exemplo prático de utilização desta técnica é dado a seguir. Novamente, por comodidade e na tentativa de facilitar a compreensão do texto, nos restringiremos ao plano. Afirmamos, entretanto, que as extensões para o caso do R^3 são imediatas.

Dado um 2–simplexo $o = \langle v_0, v_1, v_2 \rangle$ e uma curva definida pela equação $f(p) = 0$, onde $f: R^2 \rightarrow R$ é contínua, desejamos computar uma aproximação linear f_σ para f no domínio de o. Abaixo apresentamos o algoritmo 2.1 com esta finalidade.

Algoritmo 2.1: AproxLinSimp

```
AproxLinSimp(<v0, v1, v2>, f, fσ) {
  S0 ← Sinal(f(v0))
  S1 ← Sinal(f(v1))
```

```
S2 ← Sinal(f(v2))
SE (S0 = S1) {
  SE (S0 ≠ S2) {  /* v2 tem sinal distinto de v0, v1 */
    fσ0 = CalcIntersAresta(v0, v2 )
    fσ1 = CalcIntersAresta(v1, v2 )
  }
SENÃO SE (S0 = S2) { /* v1 tem sinal distinto de v0, v2 */
  fσ0 = CalcIntersAresta(v0, v1 )
  fσ1 = CalcIntersAresta(v2, v1)
}
SENÃO { /* v0 tem sinal distinto de v1, v2 */
  fσ0 = CalcIntersAresta(v1, v0 )
  fσ1 = CalcIntersAresta(v2, v0)
}
} /* AproxLinSimp */
```

O algoritmo recebe um simplexo e uma função f como entrada e devolve o vetor f_σ, de duas posições, contendo aproximações das interseções das arestas com a curva. Este vetor pode ser visto como um segmento de reta que aproxima linearmente a curva no interior do simplexo.

O procedimento *Sinal* retorna o sinal de um número real através do código: positivo (1), nulo (0) e negativo (–1). Pelas configurações possíveis de sinais, somente duas arestas podem ser interceptadas pela curva. O procedimento *CalcIntersAresta* calcula o ponto que é uma aproximação da interseção da curva com uma dada aresta.

Pode–se observar que quando os três sinais nos vértices são iguais, nenhuma aproximação é computada. Esta configuração de sinais, em geral, equivale a dizer que o simplexo não é interceptado pela curva, isto é, ou ele está totalmente contido no semiespaço negativo ou está no semiespaço positivo e, neste caso, f_σ não possui raízes no seu interior. Veremos, pouco mais adiante, que existem casos de dúvida.

Deve–se notar que até o momento, não nos preocupamos em tomar nenhuma atitude no caso da função se anular em algum dos vértices. Este caso constitui–se em um dos grandes problemas das técnicas simpliciais e será visto, com mais calma, no próximo capítulo. Por enquanto podemos supor que isto jamais acontece.

De posse do algoritmo *AproxLinSimp* podemos construir uma aproximação para a curva em todo o espaço triangulado. O algoritmo 2.2 sugere uma implementação para isto.

Algoritmo 2.2: AproxLinTriang

```
AproxLinTriang(f, T) {
  PARA CADA σ ∈ T {
```

```
        SE (ClassSimp(σ, f) = F–INTERCEPTANTE) {
            AproxLinSimp(σ, f, f_σ);
            ProcessaAprox(f_σ);
        }
    }
} /* AproxLinTriang */
```

O algoritmo recebe a função f, definidora da curva e a triangulação T na qual a região de interesse está decomposta. Cada simplexo da triangulação é classificado perante a função através do procedimento *ClassSimp* e, caso seja F–interceptante é computada uma aproximação linear para a curva dentro dele através do procedimento *AproxLinSimp*, visto anteriormente. O procedimento *ProcessaAprox* trata a aproximação convenientemente, por exemplo, exibindo–a ou armazenando–a em estrutura de dados responsável pela aproximação da curva em toda a triangulação.

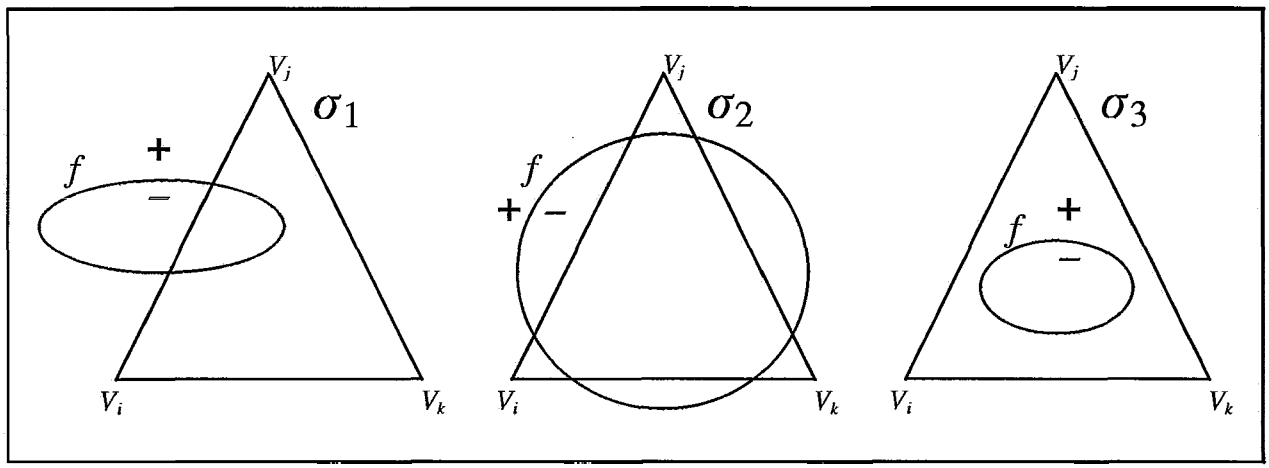


Fig. 2.8: simplexos interceptantes que não são F–interceptantes.

Não são todos os simplexos que possibilitam o cálculo de uma aproximação linear. Como mencionamos, podem existir simplexos que nem mesmo são interceptados pela curva. Existem, ainda, algumas outras situações mais sutis e bem mais problemáticas do que estas. Nestas novas situações, os simplexos contêm partes da curva mas o simples exame do valor da função nos seus vértices é incapaz de caracterizá–los como interceptados. A figura 2.8 dá alguns exemplos.

Uma questão interessante que se coloca é sobre a continuidade da aproximação linear dentro de toda a triangulação. Estariam os vários pedaços lineares, computados independentemente dentro de cada simplexo F–interceptante, aleatoriamente espalhados pela triangulação ou convenientemente encaixados, formando uma poligonal (ou poliedro no R^3)? Evidentemente, a aproximação resulta em uma poligonal (poliedm)e a figura 2.9 mostra o porquê.

Na figura 2.9(a) vemos a curva definida pela função f sendo aproximada linearmente por partes no interior da triangulação T por f_T. A garantia da continuidade de f_T vem do fato de, independentemente da ordem como os simplexos são pmcessados (classificação e

eventual cálculo da aproximação), as interseções são sempre determinadas pelas arestas F–interceptantes. Estas arestas são comuns aos dois simplexos atravessados pela curva e como no cálculo da aproximação linear somente elas são relevantes, a aproximação da interseção resulta sempre no mesmo ponto.

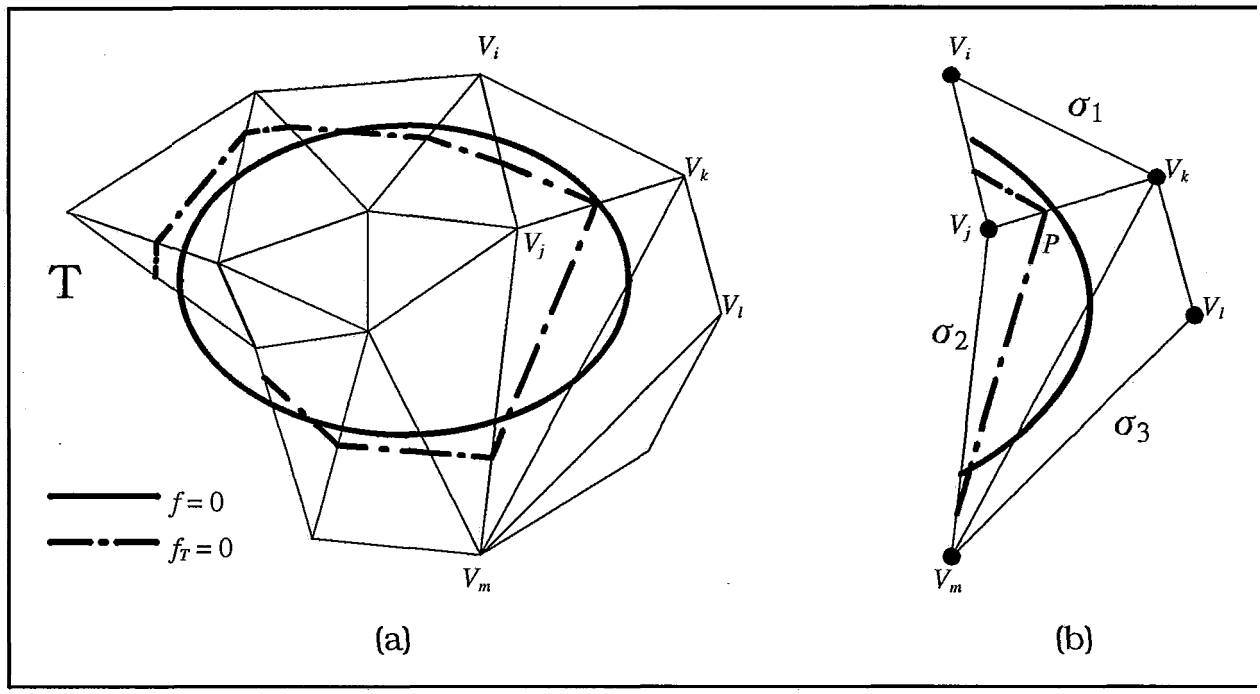


Fig. **2.9:** a função afim interpoladora é contínua.

Tomando a figura 2.9(b), vemos que a aresta $e = \langle v_j, v_k \rangle$ é F–interceptante e está presente nos dois simplexos σ_1 e σ_2. O cálculo do ponto p, aproximador da interseção de e com a curva, e feito da mesma forma tanto para σ_1 como para σ_2, o que garante a continuidade da aproximação naquela junta.

Analogamente, no $\mathbf{R}^3$ os simplexos compartilham facetas formadas pelas arestas F–interceptantes e, consequentemente, os vários pedaços de plano computados independentemente encaixam–se perfeitamente nestas facetas, formando um poliedro que se espalha pela triangulação, acompanhando as variações da superfície definida por f.

Outro aspecto interessante na figura 2.9(b) é que o simplexo σ_3 é interceptante mas não é F–interceptante e, por isto, nenhum segmento de reta é computado em seu interior. Isto pode ser visto como uma espécie de filtro ou regularização da aproximação, que elimina sinuosidades da curva menores que as dimensões do simplexo. Falaremos melhor disto no próximo capítulo.

Até o momento trabalhamos com a aproximação de variedades definidas por funções implícitas, exigindo destas funções apenas a continuidade. Na verdade, a condição suficiente para que a aproximação seja adequada é pouco mais forte: as funções devem ter zero como valor regular.

Def. 2.7: Valor Regular
Um ponto $p \in R^n$ é dito valor regular de uma função C^1 $F:R^m \rightarrow R^n$ se e somente se para todo $x \in F^{-1}(p)$ a matriz jacobiana de F tem posto máximo. No caso de $F:R^m \rightarrow R$, isto é equivalente a dizer que $\nabla F(x) \neq 0$ sempre.

Uma função $f:R^m \rightarrow R^n$ que tem zero por valor regular, define uma $(m-n)$-variedade diferenciável ([HIRS76]). Esta função nos casos de $n=2,3$ define, respectivamente, curva e superfície diferenciáveis.

2.6. MÉTODO DE CONTINUAÇÃO

A aproximação linear de funções implicitamente definidas, mas especificamente a aproximação de curvas no plano, tal como foi descrita até agora, evidencia uma característica interessante dos simplexos F-interceptantes: eles sempre têm uma região de entrada e outra de saída para a aproximação.

Como a triangulação cobre toda a região de interesse e a função definidora do objeto que está sendo rastreado é contínua, dado um simplexo o F-interceptante, sempre existirão simplexos vizinhos de o igualmente F-interceptantes. Em última análise, o trajeto da aproximação linear por entre os simplexos contém apenas os F-interceptantes, sendo que alguns destes podem pertencer a fronteira da triangulação.

As subfaces F-interceptantes são sempre comuns a dois simplexos distintos e servem como pontes para a aproximação linear. Na figura 2.10, por exemplo, vemos que a subface (aresta)$<v_j, v_k>$ é comum a σ_1 e σ_2, servindo de ponte entre eles. Da mesma forma, a subface $<v_k, v_l>$ serve de ponte entre σ_2 e σ_3. Como já mencionado, o cálculo da aproximação da interseção do objeto com a aresta é feito utilizando-se somente estas subfaces e, por conseguinte, a "colagem" dos pedaços lineares é feita consistentemente, garantindo a continuidade da função afim aproximadora.

Explorando melhor este conceito de ponte, atribuído as subfaces F-interceptantes, [ALLG80] propõe uma maneira mais eficiente para o cálculo de f_F, o Método de Continuação. Inicialmente, o método considera conhecido um simplexo F-interceptante (ou pelo menos um ponto do objeto que possibilite a localização deste primeiro simplexo F-interceptante) e, a partir daí, vai saltando para os vizinhos também F-interceptantes utilizando-se das subfaces pontes, construindo trechos da aproximação linear em cada simplexo por onde passa. Se o simplexo inicial é novamente alcançado a partir de algum outro, o procedimento de aproximação é interrompido. Se algum simplexo de fronteira for alcançado, o método recomeça do simplexo inicial só que fazendo o percurso em sentido contrário ao do inicial.

Os métodos de continuação têm sua eficiência calcada principalmente em dois pontos:

(i)somente são processados os simplexos F-interceptantes, ou seja, aqueles que realmente contêm partes da aproximação linear (raizes de f_σ=0), evitando-se que simplexos vazios ou simplesmente interceptantes sejam, sequer, analisados.

(ii) as aproximações das interseções nas subfaces F-interceptantes (efetivamenteos únicos cálculos necessários para a obtenção da aproximação linear) passam a ser feitas uma única vez para cada subface.

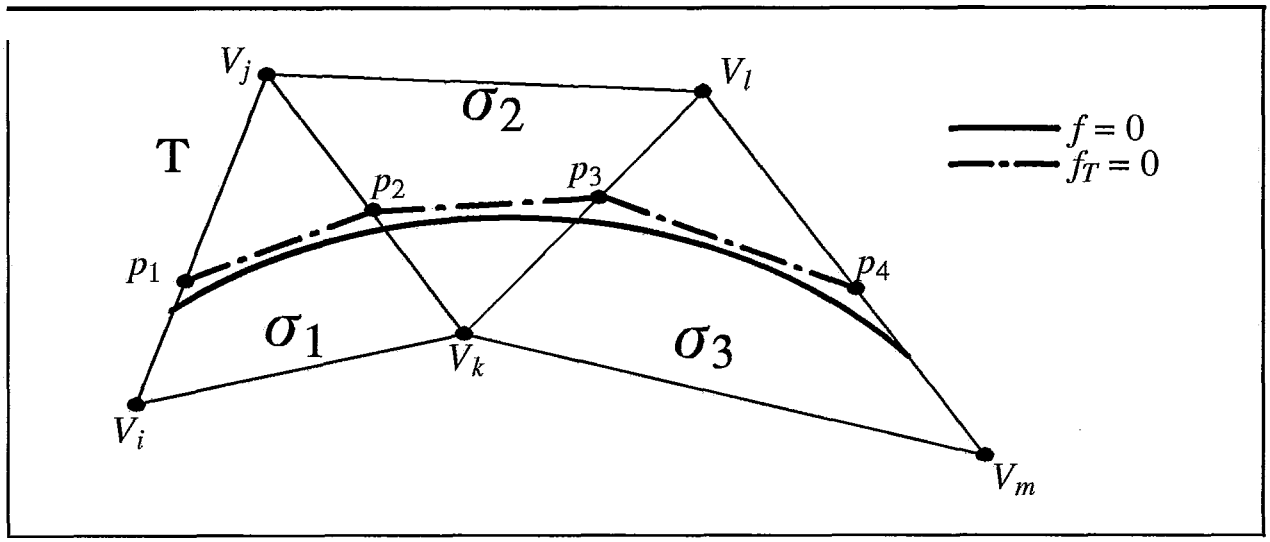


Fig. 2.10: aproximação afim usando o Método de Continuação.

Na figura 2.10, vemos uma pequena região do plano decomposta na triangulação T= $\{\sigma_1, \sigma_2, \sigma_3\}$ e uma curva, definida pela função f, para qual queremos computar uma aproximação linear f_T. Inicialmente, o algoritmo calcula p_1, a aproximação da interseção da curva com a aresta $e_1 = \langle v_i, v_j \rangle$ e, em seguida, prossegue para a aresta $e_2 = \langle v_j, v_k \rangle$, calculando p_2, extremo do segmento de f_T contido em σ_1. A aresta e_2 é usada como ponte e σ_2 é acessado. Neste ponto, só é necessário o cálculo de p_3 para obtenção do pedaço de f_T contido em σ_2. Da mesma forma, através da aresta $e_3 = \langle v_k, v_l \rangle$, chega-se em σ_3, calcula-se p_4 e obtem-se, finalmente, a poligonal aproximadora $f_T = \{p_1, p_2, p_3, p_4\}$.

Além da eficiência em termos da diminuição dos cálculos e consequente aumento da velocidade de execução, os métodos de continuação apresentam outra grande vantagem: a facilidade de obtenção da topologia da aproximação linear. Quando os simplexos F-interceptantes são visitados aleatoriamente, tal como no algoritmo 2.2, apesar da garantia da continuidade da aproximação, é complicado compreender o arranjo relativo dos vários pedaços lineares que foram computados independentemente, ao passo que através do processo de continuação, o estabelecimento das relações topológicas é bastante simplificado pela utilização das subfaces-ponte.

No caso das triangulações regulares, como a CFK, a transição entre os simplexos pode ser feita através das regras de pivoteamento, e não há a necessidade de estruturas de dados muito complexas para o armazenamento da triangulação. Isto toma o procedimento mais rápido, econômico e menos propenso a erros.

A seguir apresentamos o algoritmo 2.3, bastante simplificado, que visa a aproximação linear de parte de uma superfície em triangulação de uma região do espaço através do Método de Continuação.

Algoritmo 2.3: Continuação

```
Continuação(f, T) {
  σ0 ← LocalSimpInic(f, T)
  ProcessaSimpInic(f, σ0)
  σ ← ProxSimp(σ)
  ENQUANTO (σ ≠ σ0) {
    ProcessaSimp(f, σ)
    σ ← ProxSimp(σ)
    SE (Fronteira(σ)) {
      ProcessaSimp(f, σ)
      σ ← σ0
    }
  }
} /* Continuação */
```

O algoritmo recebe a função f definidora da supefície e a triangulação T da região de interesse. Primeiramente é localizado um simplexo de partida através do pmcedimento *LocalSimpInic* que, em seguida, é processado pelo procedimento *ProcessaSimpInic*. Este processamento é diferente do caso geral, tratado pelo pmcedimento *ProcessaSimp*, pois é preciso calcular as interseções da supefície com todas as arestas F–interceptantes, enquanto que para os outros simplexos é necessário apenas computar as interseções com as arestas que ainda não foram tratadas. O procedimento *ProxSimp* fornece o sucessor de um simplexo e Fronteira verifica se o simplexo pertence a fronteira da triangulação.

De posse do simplexo inicial, é executado um laço que percorre todos os simplexos F–interceptantes vizinhos. Se, por acaso, o simplexo inicial for alcançado através de algum outro, significa que toda a superfície já foi aproximada (supondo–a simplesmente conexa) e o processamento deve ser interrompido. Da mesma forma, o procedimento é interrompidos e determinado simplexo pertencer a fronteira da triangulação. Neste caso, é necessário computar o ramo da aproximação que começa no simplexo inicial e se estende no sentido contrário ao do inicial.

As triagulações, de acordo com [DOBK90], devem possuir algumas propriedades para que demonstrem utilidade prática nas aplicações destes métodos de continuação. Entre estas propriedades encontram–se:

(a) A obtenção de todos os simplexos que compartilham da mesma subface deve ser facilitada;

(b) Deve–se poder rotular os vértices de toda a triangulação simultaneamente com índices 0,...,n (onde n é a dimensão do espaço triangulado), de tal modo que

cada um dos n+1 vértices de cada simplexo possua rótulo diferente, isto é, a rotulação dos vértices da triangulação deve ter caráter global e não deve gerar confusão;

(c)Todos os simplexos da triangulação devem ter, aproximadamente, o mesmo tamanho e suas dimensões devem ser parecidas em todas as direções.

A segunda propriedade visa facilitar a construção das regras de pivoteamento e, juntamente com a primeira, ajudar na tarefa de transição entre os simplexos. A terceira é relacionada com uma característica dos simplexos chamada de gordura ([FREI91]) e procura garantir que, durante a fase de aproximação, não serão visitados simplexos muito "achatados" ou "magros" e, consequentemente, que o algoritmo progredirá igualmente para uma mesma quantidade de cálculos.

A eficiência do Método de Continuação é posta em dúvida quando imaginamos situações mais gerais. A primeira questão que se coloca é a respeito da localização do primeiro simplexo F-interceptante. Em alguns casos particulares, como por exemplo, na aproximação de soluções de algumas equações diferenciais, isto não é muito difícil pois as condições iniciais (e/ou de contorno) fornecem pistas suficientes para que um primeiro simplexo seja encontrado. Na maioria dos casos, todavia, deve ser feita algum tipo de busca (possivelmente exaustiva) para localizá-lo e, a partir daí, dar continuidade ao método, transitando pelos simplexos.

Outra questão relevante é que, mesmo de posse do ponto de partida para o método, não se pode garantir que todo conjunto solução do problema, presente na triangulação, será aproximado. No caso, por exemplo, de aproximação de superfícies desconexas, isto é, com mais de uma componente conexa, não basta localizarmos apenas um simplexo inicial, pois na maioria dos casos não se conseguirá alcançar a família de simplexos F-interceptantes de uma das componentes conexas a partir de um simplexo F-interceptado por outra. Para estes casos, em geral, é necessário um simplexo inicial para cada componente conexa.

A necessidade de um ponto de partida para cada componente conexa é o grande contra-senso da técnica de continuação. De posse deste ponto de partida o método é autosuficiente para obter toda uma componente conexa, de maneira eficiente e rápida, somente processando simplexos que são realmente importantes. Para se garantir, contudo, que todas as componentes conexas presentes na triangulação serão encontradas, geralmente é necessária busca exaustiva para localização de todos os pontos de partida. Esta busca, nos piores casos, pode tornar o método tão oneroso quanto aqueles que processam aleatoriamente os simplexos.

2.7. CONCLUSÃO

As técnicas simpliciais, como pudemos observar, formam excelente plataforma para a construção de funções afins interpoladoras. A utilização destas funções simplifica muito

o cálculo de aproximacões de funções definidas implicitamente. O processo de aproximação, além de simples é bastante geral, servindo para grande variedade de classes de funções.

O Método de Continuação é bastante eficiente no que se refere aos problemas em que temos idéia, *a priori*, de como é o conjunto solução que procuramos, no caso em que sabemos, de antemão, quantas componentes conexas existem e temos maneiras de iniciar a avaliação, em sequência, dos simplexos para cada uma delas. Em situações mais gerais, no entanto, ele deixa a desejar na medida em que a localização dos pontos de partida necessários pode ser bastante custosa, uma vez que a busca pelos simplexos F–interceptantes pode ser exaustiva.

O principal problema nesta avaliação exaustiva de simplexos é que a grande maioria deles não possui raízes da função que se está aproximando sendo, portanto, inúteis para efeitos do cálculo de aproximação linear. Consequentemente, quase todo processamento gasto nesta fase é desperdiçado.

Apesar destes aspectos negativos, o método apresenta uma séria vantagem que é a possibilidade de se estabelecer as relações topológicas entre os vários pedaços lineares da aproximação.

"...ten years have got behind you..."

Capítulo 3

3. SÓLIDOS CSG E TÉCNICAS SIMPLICIAIS

3.1. INTRODUÇÃO

Os métodos simpliciais, como acabamos de ver, são bastante úteis para solução de vários problemas, principalmente o da aproximação linear por partes de funções implícitas. Neste capítulo procuraremos conjugar a utilização de técnicas simpliciais com os esquemas de representação de sólidos baseados em CSG. Veremos como é possível definir funções que caracterizam os sólidos globalmente e como podemos utilizá–las no cálculo de aproximações lineares por partes para as fronteiras destes.

Abordaremos, também, a questão da regularização da superfície do sólido. Veremos como os métodos simpliciais lidam com as questões relativas à consistência da geometria da aproximação linear.

Finalmente, estudaremos melhor a questão que foi deliberadamente negligenciada nas seções anteriores: o fato de um vértice da triangulação anular a função para qual buscamos uma aproximação. Falaremos, brevemente, sobre o que isto significa e sobre uma técnica, bastante simples e amplamente utilizada, de se evitar que estes casos ocorram sempre que eles forem indesejados.

3.2. FUNÇÕES CARACTERÍSTICAS

Inicialmente pode parecer estranho que exista relação entre os esquemas de representação de sólidos baseados em CSG e os métodos simpliciais. Como vimos, a base destes métodos consiste na avaliação, da função definidora da superfície a ser aproximada, em determinados vértices de uma triangulação do espaço e, em geral, não

nos apercebemos da existência de funções que caracterizam um sólido representado por uma árvore CSG.

Todo sólido S representado por uma árvore CSG A, cujas folhas são primitivas formadas por semiespaços, tal como vistos na seção **1.3,** possui uma função $f_S: R^3 \rightarrow R$ associada a A, capaz de representá-lo globalmente. Na verdade, existem várias funções que podem representá-lo, todas recebendo o nome de funções características de S.

Def. 2.1: Função Característica

Um sólido CSG S pode ser visto como um conjunto de pontos $\Gamma(f_S) = \{p \in R^3 ; f_S(p) \leq 0\}$ caracterizado por uma desigualdade, onde f_S é dita *função característica* de S.

As funções características são muito úteis para se expressar operações booleanas entre sólidos. A função característica da interseção entre dois sólidos A e B, por exemplo, é o máximo entre as funções características f_A e f_B. De maneira geral, tem-se:

Proposição **3.1:**

O sólido $S = A \cap B$ formado pela interseção de dois sólidos A e B, com respectivas funções características f_A e f_B, é o conjunto $\Gamma(\text{Max}(f_A, f_B)) = \{p \in E^3 ; \text{Máximo}(f_A, f_B) \leq 0\}$ enquanto que $S = A \cup B$ é o conjunto $\Gamma(\text{Min}(f_A, f_B)) = \{p \in E^3 ; \text{Mínimo}(f_A, f_B) \leq 0\}$.

Demonstração:

$\forall\ p \in A \cap B,\ f_A(p) \leq 0$ e $f_B(p) \leq 0 \Leftrightarrow \text{Máximo}(f_A, f_B) \leq 0 \Leftrightarrow p \in \Gamma(\text{Max}(f_A, f_B))$ ♦

$\forall\ p \in A \cup B,\ f_A(p) \leq 0$ ou $f_B(p) \leq 0 \Leftrightarrow \text{Mínimo}(f_A, f_B) \leq 0 \Leftrightarrow p \in \Gamma(\text{Min}(f_A, f_B))$ ♦

Por sua simplicidade na manipulação de operações booleanas, as funções características são ideais para representação dos sólidos CSG. Através do uso recursivo das identidades acima, pode-se representar qualquer sólido CSG substituindo-se, convenientemente, os operadores booleanos pelas funções MIN-MAX e as primitivas por suas funções características, tal como no exemplo abaixo:

$$S = ((A \cap B) \cup C) \cap D = \Gamma(f_S) \quad \text{onde} \quad f_S = \text{Max}(\text{Min}(\text{Max}(f_A, f_B), f_C), f_D)$$

As modificações são tão simples que a expressão f_S pode ser avaliada a partir da própria árvore CSG do sólido, por intermédio de reinterpretação conveniente dos seus nós.

Utilizando as funções características, podemos computar uma aproximação linear por partes para a casca de determinado sólido CSG. A função característica, como representaçáo implícita, é capaz de responder se um ponto qualquer do espaço está no interior, exterior ou na fronteira do sólido, o que é, em última análise, tudo o que os métodos simpliciais precisam para o cálculo da aproximação. Note-se que se as primitivas do sólido possuirem funções características contínuas então a sua função característica MIN-MAX será igualmente contínua. Perde-se com o MIN-MAX, entretanto, a diferenciabilidade.

Tal como nos exemplos anteriores, nos quais se buscava aproximações para curvas no plano, é feita uma triangulação da região na qual se deseja computar a aproximação da casca do sólido e, em seguida, cada simplexo da triangulação é tratado convenientemente. Naqueles F–interceptantes, em relação a função característica do sólido, é computado um "pedaço de plano" aproximador da fronteira.

3.3. REGULARIZAÇÃO

A manipulação algébrica que envolve as funções características nos permite representar facilmente um sólido CSG por uma desigualdade característica. A obtenção da fronteira deste sólido, contudo, é bem mais complicada.

Na fase de aproximação linear, estamos interessados na fronteira da regularização do sólido e, portanto, nos subconjuntos de zeros da função característica que possuam, em sua vizinhança, valores tanto negativos quanto positivos. A figura 3.1 exemplifica.

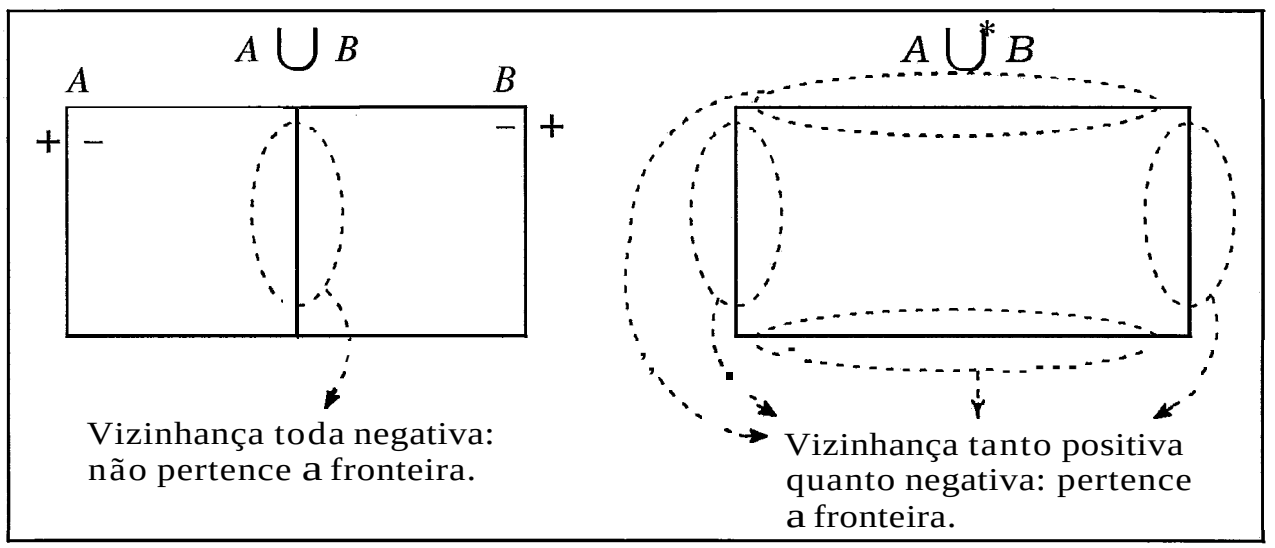


Fig. 3.1: a fronteira da regularização está entre valores positivos e negativos.

Podemos melhor compreender a fronteira do sólido regularizado imaginando a seguinte situação: retiramos da triangulação todos os simplexos interiores ao sólido, ou seja, todos aqueles cujos vértices têm valores negativos de função característica. Isto constitui uma primeira aproximação da fronteira a partir do interior. Em seguida descartamos todos os simplexos exteriores, em aproximação da fronteira por fora. Após isto, a fronteira da regularização estará contida na união dos simplexos que sobraram.

As expressões MIN–MAX são bastante úteis para aproximar o interior e a regularização do sólido, como assegurado pelo seguinte teorema:

Teorema 3.1:

Seja $f_s: E^3 \rightarrow R$ contínua tal que Interior($\{p \in E^3 ; f_s(p) = 0\}$) = $\emptyset$. Então valem:

(a) $\{p \in E^3 ; f_s(p) < 0\} \subset$ Interior($\Gamma(f_s)$)

(b) Regularização($\{p \in E^3 ; f_s(p) < 0\}$) = Regularização($\Gamma(f_s)$)

Este teorema evidencia o fato de que embora as funções características definam objetos com fronteiras que podem ser complicadas, a regularização destes objetos faz com que essas fronteiras fiquem mais simples. Como a aproximação linear cobre, não a fronteira do sólido, mas a fmnteira da sua regularização, não há problema em se utilizar as funções características.

Para a demostração do teorema **3.1** adotaremos a notação abaixo e precisaremos do auxílio do lema **3.1.**

Notação:

$\forall$ X conjunto: $\overline{X}$ = Fecho(X);

$\Gamma = \{p \in E^3 ; f_s(p) \geq 0\}$, $\Gamma^- = \{p \in E^3 ; f_s(p) < 0\}$, e $\Gamma^0 = \{p \in E^3 ; f_s(p) = 0\}$;

Γ^* = Regularização(Γ) = $\overline{\text{Interior}(\Gamma)}$.

Lema **3.1:**

Sejam A aberto e B tal que Interior(B) = $\emptyset$ então Interior(A $\cup$ B) $\subset \overline{A}$.

Demonstração: Lema **3.1**

mostraremos que Complementar($\overline{A}$) $\subset$ Complementar(Interior(A $\cup$ B)).

Se $b \notin \overline{A}$, $\exists$ V vizinhança de b tal que V $\cap$ A = $\emptyset$.

Então V $\cap$ (A $\cup$ B) = V $\cap$ B $\subset$ B

e, por hipótese Interior(V $\cap$ (A $\cup$ B)) = $\emptyset$

mas daí (Interior(V) $\cap$ Interior(A $\cup$ B)) = $\emptyset$

e como Interior(V) é vizinhança de b decorre que b $\geq$ Interior(A $\cup$ B)+

Demonstração: Teorema **3.1**

(a) Pela continuidade de f_s, Γ^- é aberto e $\Gamma^- \subset \Gamma$ e, então, $\Gamma^- \subseteq$ Interior(Γ) ♦

(b) Provemos, primeiramente, que $\overline{\Gamma^-} \subset \Gamma^*$:

do item (a) temos que $\Gamma^- \subset$ Interior(Γ)

e daí $\overline{\Gamma^-} \subset \Gamma^*$ ♦

Provemos, agora, que $\Gamma^* \subset \overline{\Gamma^-}$:

Por hipótese, Interior(Γ^0) = $\emptyset$

então, pelo lema, Interior(Γ) = Interior($\Gamma^- \cup \Gamma^0$) $\subset \overline{\Gamma^-}$.

Logo: $\Gamma^* \subset \overline{\Gamma^-}$ ♦

A condição de que Interior(Γ^0) = $\emptyset$ é geralmente atendida no nosso caso pois, como veremos, as funções com que trabalhamos são polinomiais que definem, em geral, superfícies no espaço. A única polinomial que não satisfaz esta condição e a polinomial constante igual a zero, que pode ser facilmente eliminada em preprocessamento. A continuidade de f_s, por sua vez, vem do fato das funções MIN-MAX serem contínuas.

Uma primeira utilização deste teorema é a possibilidade de se representar o operador diferença nas funções características. Em princípio poderíamos pensar que:

$$A \setminus B = A \cap \text{Complementar}(B) = \Gamma(\text{Max}(f_A, -f_B))$$

mas isto nem sempre é verdade pois, em geral:

$$\text{Complementar}(B) = \{p \in E^3 ; f_B(p) > 0\} \neq \{p \in E^3 ; f_B(p) \geq 0\} = \Gamma(-f_B)$$

entretanto:

$$A \cap^* \text{Complementar}(B) = \text{Regularização}(A \setminus B) = \text{Regularização}(\Gamma(\text{Max}(f_A, -f_B))).$$

Na aproximação da casca do sólido, através de funções características, aparecem alguns outros aspectos relacionados a regularização. Como já visto anteriormente, podem existir simplexos interceptantes, isto é, contendo partes da superfície do sólido, que não fornecem aproximações lineares satisfatórias. Nos vértices destes simplexos, a função característica possui sempre o mesmo sinal, sugerindo a inexistência de raízes da função afim aproximadora. Isto se deve ao fato dos vértices das triangulações serem amostragens finitas do espaço fazendo com que estas sofram dos pmblemas inerentes as discretizações.

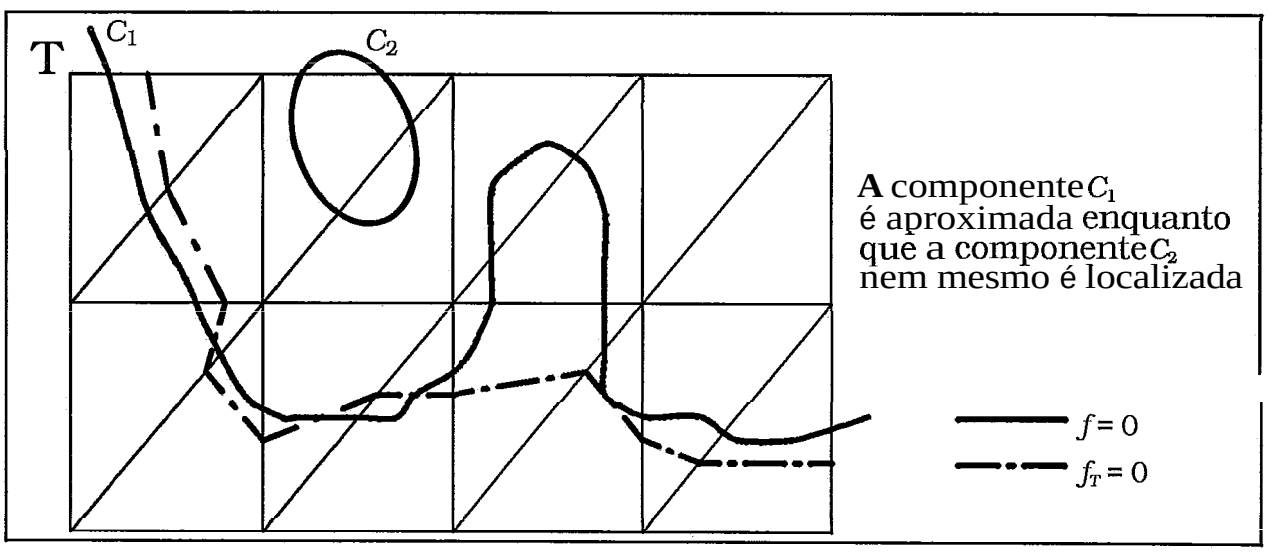


Fig. 3.2: exemplo de filtragem feita pela triangulação.

As partes da superfície do sólido que podem estar contidas nesses simplexos são relativamente pequenas para serem percebidas pela amostragem representada pela triangulação. Quando computada a aproximação linear por partes, desaparecem "pontas" da fronteira assim como todas as componentes conexas inteiramente contidas em apenas um simplexo. Este processo pode ser entendido como uma espécie de filtragem morfológica do sólido ([SERR82]). A figura 3.2 mostra um exemplo no plano.

Procurando estabelecer um limiar para as aproximações, o trabalho de [SUFF90] define dois níveis de refinamento. O espaço é inicialmente decomposto até um nível de aceitação, em seguida, vasculhado exaustivamente. As componentes conexas que eventualmente não forem localizadas neste nível, são sumariamente ignoradas. Passa-se,

então, para o segundo nível de refinamento, com simplexos bem menores e procede-se ao cálculo da aproximação.

Dependendo da resolução da triangulação e do seu posicionamento no espaço, alguns grupos de componentes conexas da casca do sólido podem ser perdidos enquanto que alguns outros podem ser fundidos em uma única componente, fazendo com que a topologia da aproximação afim seja diferente da topologia do sólido original. Este problema é bastante sério e evidencia a questão da qualidade da aproximação. As técnicas simpliciais fornecem, na maioria dos casos, resultados bastante bons mas trabalham levando em consideração apenas a geometria dos sólidos sem se preocuparem com questões relativas a sua topologia.

A qualidade da aproximação, assim como em todos os problemas ligados a discretização, é diretamente dependente da qualidade da amostragem, isto é, quanto mais densa a triangulação e, portanto, quanto menores os simplexos, melhor a aproximação. A definição do nível de refinamento ideal para não se perder nenhuma componente conexa e para que a aproximação seja aceitável, é tarefa complicada, que depende de estudo cuidadoso do comportamento da função característica na região de interesse. Uma análise qualitativa das aproximações é, atualmente, objeto de intensos estudos e pesquisas feitas com o intuito de se obter aproximações que preservem a topologia original estão em andamento.

Convém notar que as operações booleanas regularizadas adotadas por [REQU80] atuam como filtros infinitesimais, na medida em que somente detalhes da superfície do sólido não-tridimensionais são eliminados. As filtragens que ocorrem com a utilização das técnicas simpliciais, por outro lado, são capazes de remover detalhes tão grandes quanto a resolução da triangulação, ou seja, na mesma ordem de grandeza dos simplexos. Isto, do ponto de vista prático, é bem melhor pois possibilita o tratamento de incertezas numéricas, sugerindo aplicações eminentemente práticas como os sistemas de modelagem que trabalham ligados a máquinas de controle numérico ou a geradores de malhas representativas de sólidos para análise através de métodos de elementos finitos.

3.4. VÉRTICES QUE ANULAM A FUNÇÃO CARACTERÍSTICA

Na descrição das aproximações lineares por partes feitas nas seções anteriores, escolhemos cuidadosamente as triangulações de forma a evitar que os vértices dos simplexos anulassem as funções definidoras dos objetos. Nesta seção estudaremos melhor estes casos, procurando dar as triangulações um caráter mais geral.

Quando nenhum dos vértices de uma triangulação anula a função característica do objeto, a aproximação linear por partes é garantidamente uma variedade ([ALLG80]). Isto evidencia uma propriedade bastante expressiva dos métodos simpliciais: através deles conseguimos aproximar não-variedades por variedades, o que pode ser extremamente útil em muitas situações nas quais a diferença de topologia entre o objeto e a aproximação não se constitui em um problema grave no contexto da aplicação.

O fato de um vértice anular a função característica de um sólido significa que este vértice está na casca do sólido e que alguns pedaços da aproximação linear passarão por ele. Isto pode acarretar alguns problemas bastante sérios. Na figura 3.3, ao se considerar F-interceptante a aresta $e = \langle v_i, v_j \rangle$ cujo vértice v_i anula a função característica, está-se abrindo perigoso precedente. No cálculo da aproximação da interseção desta aresta com o objeto, privilegia-se aquele vértice cujo valor de função é mais próximo de zero e, portanto, v_i será sempre o escolhido como ponto de interseção. No caso de tanto v_i como v_j anularem a função, a própria aresta será considerada um pedaço da aproximação linear por partes. Nos simplexos vizinhos que também possuirem v_i como vértice, o cálculo será feito da mesma forma e ter-se-á um vértice pertencendo a mais de duas arestas, caracterizando a aproximação linear por partes como não-variedade.

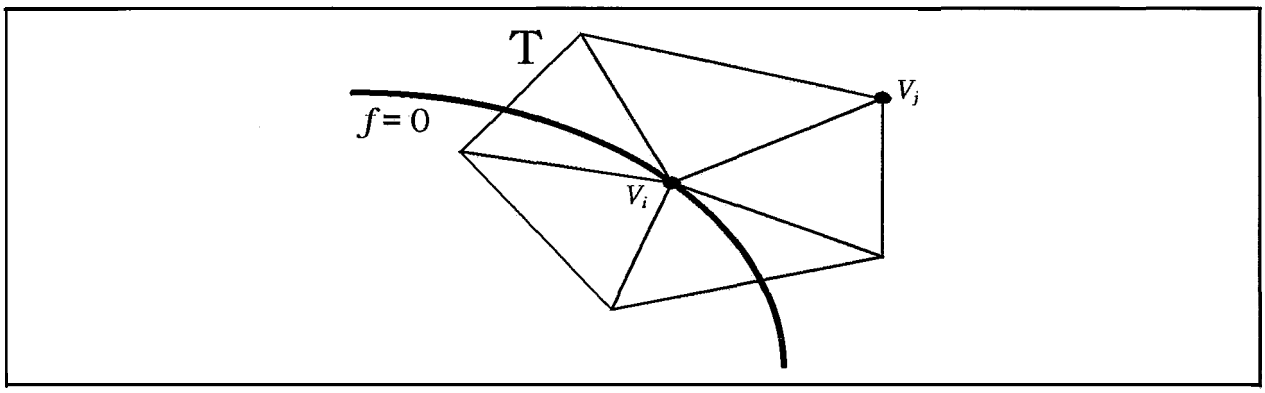


Fig. **3.3:** vértice que anula a função característica.

Além do fato das aproximações lineares deixarem de ser variedades, existe um outro ponto crucial que nos força a evitar que qualquer vértice da triangulação anule a função característica do sólido. Como vimos, nem todos os pontos que anulam a função característica estão na fronteira do sólido e os subconjuntos de raízes que não estão em uma vizinhança com altemância de sinais devem ser eliminados pela regularização. Ao considerarmos os pontos que anulam a função característica como partes da aproximação linear, estaremos correndo o risco de aproximar pedaços que não fazem parte da fronteira do sólido e, consequentemente, poderemos estar arruinando a regularização. Para tal, basta que algum vértice da triangulação pertença a um subconjunto de zeros da função característica que não faça parte da fronteira da regularização do sólido. A figura **3.4** ilustra uma situação onde a regularização estaria comprometida.

Alguns trabalhos ([CAST90] e [CAST91]) sugerem maneira interessante de evitar estes casos. A idéia é alterar localmente a geometria da triangulação toda vez que um vértice possuir valor nulo de função, em técnica comumente chamada de *Perturbação de* Vértices.

Esta técnica consiste, basicamente, em aplicar ligeira alteração nas coordenadas do vértice problemático, fazendo com que ele não mais anule a função característica. Estas alterações devem ser feitas cuidadosamente para que a geometria da triangulação não seja demasiadamente distorcida e que, por conseguinte, o valor da função avaliada no novo vértice seja próximo de zero, fazendo com que a aproximação linear passe por perto dele.

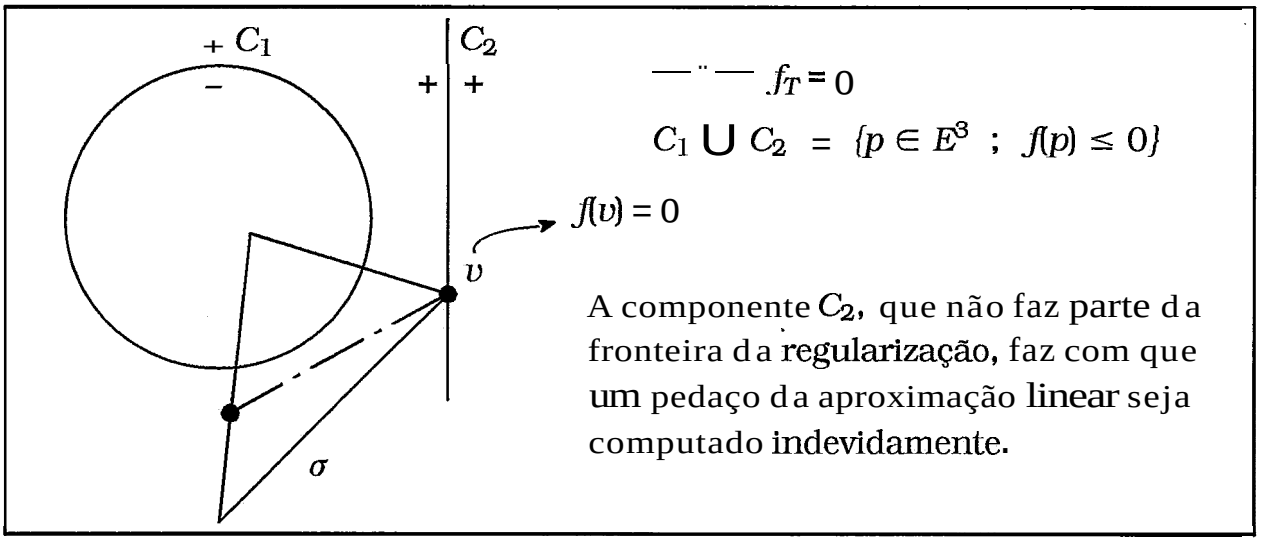


Fig. 3.4: regularização comprometida por um vértice que anula a função.

Para se garantir a consistência das aproximações é necessário que estas perturbações sejam feitas de maneira coerente. Exceto alguns vértices da fronteira da triangulação, todos os outros pertencem a mais de um simplexo e a alteração local da geometria deve levar isto em consideração. Quando determinado vértice é perturbado, todos os simplexos que o possuem têm sua geometria alterada. Isto pode ser perfeitamente controlado mediante o emprego de estruturas de dados adequadas mas, maneira ainda mais simples, consiste em controlar apenas o valor da função aplicada aos vértices.

Podemos pensar em um procedimento de avaliação da função característica que jamais retorna valores nulos. Quando um ponto anula a função, sua geometria é aleatoriamente modificada e novo valor de função é calculado. Enquanto não se obtem valor não–nulo, o ponto continua a ser movido no espaço. O pseudo–código abaixo descreve um procedimento com este fim.

Algoritmo 3.1 : AvaliaFuncCarac

```
AvaliaFuncCarac(f, p) {
  Valor ← f(p)
  Tentativa ← 0
  ENQUANTO (Valor = 0) {
    Valor ← f(PerturbaGeometria(p, Tentativa))
    Tentativa ← Tentativa + 1
  }
  RETORNE(Valor)
} /* AvaliaFuncCarac */
```

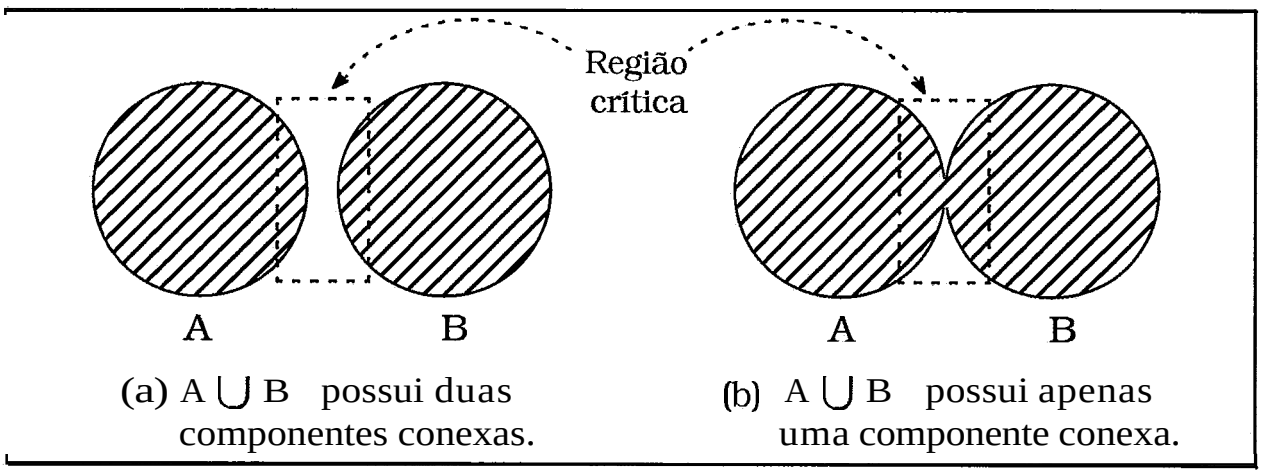
O algoritmo recebe a função característica f do sólido e um ponto p do espaço onde a função deve ser avaliada. O valor da função no ponto é calculado e, caso ele seja nulo, a geometria do ponto é ligeiramente alterada pelo procedimento *PerturbaGeometria* e novo

valor é calculado. As perturbações em sequência são feitas sempre de maneira diferente, isto é. uma alteração diferente para cada tentativa.

Com perturbações aleatórias, com distribuições normais no espaço, é nula a probabilidade de se encontrar um novo valor nulo de função, pelo fato do conjunto de zeros da função característica ter medida nula no espaço.

A escolha de um ponto para substituir um vértice problemático é feita, na verdade, de maneira pseudo-aleatória. A sequência de perturbações na geometria de um mesmo ponto deve ser feita sempre da mesma maneira. Desta forma, forçamos a avaliação de determinado vértice a retomar sempre o mesmo valor não-nulo, independentemente do simplexo que está sendo analisado, garantindo-se a continuidade nas emendas dos pedaços lineares entre os simplexos F-interceptantes.

Como as perturbações movem pelo espaço os vértices da triangulação, obviamente podem ocorrer modificações nas classificações dos simplexos. Alguns, que antes da perturbação eram considerados interceptantes, pelo fato de possuirem vértice na casca do sólido, passam a ser considerados vazios enquanto outros que, pelo mesmo motivo, eram classificados como interceptantes somente, passam a ser F-interceptantes. Aparece aqui, mais uma vez, a questão da diferença de topologia entre o sólido e a aproximação linear.



(a) $A \cup B$ possui duas componentes conexas.

(b) $A \cup B$ possui apenas uma componente conexa.

Fig. 3.5: a topologia da aproximação depende da geometria da triangulação.

Na figura 3.5 vemos a união de duas primitivas A e B. Existe uma região do espaço que é problemática pelo fato de ali se encontrarem muitos vértices da triangulação que anulam a função característica da composição final das primitivas. Nesta região, dependendo do modo como a triangulação está localizada e, também, de como forem alteradas as geometrias desses vértices pmblemáticos, podemos ter resultado final com duas componentes conexas (Fig. 3.5(a)) ou com apenas uma (Fig. 3.5(b)).

Este fato, como mencionado na seção anterior, pode ser visto como outro aspecto de regularização no qual, dependendo das posições dos simplexos na triangulação e da disposição desta no espaço, alguns detalhes da casca do objeto menores que a resolução da triangulação podem ser tratados de maneira diferente em situações diferentes. Pode-se

pensar, por exemplo, na especificação deste objeto para uma máquina de controle numérico responsável pela sua manufatura. Se a região de contato entre as duas primitivas for muito pequena, comparada com a resolução da máquina que vai esculpir a peça, a máquina acabará por separar o objeto em duas componentes conexas. Como mencionado, as triangulações podem ser utilizadas como mecanismos de representação de tolerância nos sistemas de representação de sólidos.

3.5. NOVA PROPOSTA

Fazendo uso das funções características dos sólidos CSG, vimos que podemos computar uma aproximação linear por partes para suas fronteiras utilizando técnicas simpliciais. Podemos triangular uma região de interesse com resolução previamente definida e, em seguida, utilizar o Método de Continuação para computar os pedaços da aproximação dentro de cada simplexo F–interceptante.

Como visto na seção 2.5, o Método de Continuação apresenta, em certas situações, a inconveniente necessidade de um simplexo inicial para cada componente conexa do objeto. A busca por estes simplexos iniciais pode tomar o processo tão oneroso quanto o processamento independente de todos os simplexos da triangulação. Esta sobrecarga muitas vezes não é tão relevante, principalmente quando não se deseja uma aproximação muito fiel mas apenas um esboço da superfície do sólido. Quando, todavia, existe a necessidade de aproximação mais realista e, consequentemente, de maior refinamento da triangulação, a enorme quantidade de simplexos a ser pesquisada pode inviabilizar a utilização da técnica de continuação.

O problema do processamento indiscriminado dos simplexos na tentativa de localização de todas as componentes conexas do objeto é que a medida que a triangulação vai sendo refinada, o número de simplexos F–interceptantes aumenta com taxa bem inferior a do surgimento dos simplexos não–interceptantes. Isto é intuitivamente fácil de se perceber pois os simplexos F–interceptantes cortam a casca do sólido, sendo sua quantidade, por conseguinte, proporcional a área de superfície deste. Os simplexos não–interceptantes, por sua vez, ocupam todo o restante da triangulação, sendo sua quantidade proporcional ao volume representado por esta.

Nas triangulações CFK obtidas pela subdivisão do espaço em cubos e posterior subdivisão destes em 6 tetraedros, por exemplo, o número N_s de simplexos para determinado nível n de refinamento é dado pela relação: $N_s = 6 \times 2^{3n}$.

No nível zero temos apenas um cubo e, portanto, 6 tetraedros. Quando n = 1 temos 8 cubos formando 48 tetraedros e assim por diante. Deste modo, a cada nível que a triangulação é refinada, o total de simplexos aumenta oito vezes. Na maioria dos casos, boa parte deste total é não–interceptante e o ideal seria, portanto, evitar que eles fossem sequer visitados.

Neste trabalho apresentamos a decomposição simplicial hierárquica do espaço, que procura eliminar regiões da triangulação, tão grandes quanto possível, seguramente

disjuntas da casca do objeto. Esta decomposição hierárquica, derivada das triangulações CFK, consiste na subdivisão recursiva dos tetraedros em pares de novos tetraedros, a semelhança das Bintrees ([SAME90]).

Nesta decomposição, não existe a preocupação com a localização de cada componente conexa do sólido. A região de interesse é inteiramente subdividida adaptativamente, ou seja, regiões mais próximas da casca do sólido são mais refinadas do que aquelas que estão no seu interior ou exterior. A casca é totalmente aproximada – dentro dos limites de tolerância definidos pelo nível de refinamento da triangulação – independentemente do número de componentes conexas e de como elas estão localizadas.

Para que o processo de aproximação linear dentro da decomposição simplicial hierárquica seja viável, existe a necessidade de critérios seguros de classificação dos simplexos, em cada nível de subdivisão, que garantam a localização de toda a casca do sólido perceptível para este nível. No próximo capítulo estudaremos melhor esta decomposição e esses critérios de classificação.

"...no one told you when to run..."

Capítulo 4

4, SUBDIVISÃO SIMPLICIAL ADAPTATNA

4.1. INTRODUÇÃO

As decomposições regulares são formas de subdivisão do espaço extremamente perdulárias, quando utilizadas para representação de sólidos, na medida em que representam em detalhe todo o espaço onde o sólido está imerso e não apenas as legiões que são realmente relevantes.

As decomposições hierárquicas surgiram com o objetivo de otimizar a subdivisão do espaço. Nelas, o refinamento da malha de subdivisão é feito aos poucos e adaptativamente, ou seja, somente as regiões mais próximas da fronteira do sólido são subdivididas.

O salto entre as duas formas de decomposição do espaço só é possível se existirem critérios seguros de classificação de cada célula da decomposição em relação ao sólido. Somente com a utilização destes critérios é que grandes porções do espaço podem ser eliminadas, uma vez que através deles conseguimos garantias de que nenhuma parte relevante do sólido estará sendo também perdida.

Neste capítulo apresentaremos nossa proposta para otimizar o cálculo de aproximações lineares por partes para fronteiras de sólidos CSG, descrevendo o que chamamos de Subdivisão Simplicial Adaptativa. Apresentaremos um processo de decomposição do espaço em simplexos de tamanhos variáveis a partir de esquema de subdivisão de cada simplexo em dois filhos, assunto da próxima seção.

Na seção posterior falaremos brevemente sobre o conceito de localização de sólidos CSG, que ajuda a evitar desperdícios de memória e processamento e que, portanto, nos é muito importante, Discutiremos maneira de representar nas árvores CSG somente as partes do sólido realmente relevantes.

Em seguida apresentaremos um dos pontos principais deste trabalho, a seção relativa aos critérios de classificação dos simplexos, na qual abordaremos formas de avaliá-los contra as funções definidoras das primitivase, a partir daí, eliminar do processo de subdivisão aqueles que não contêm parte da fronteira do sólido. Nesta seção aparecerá o porquê da restrição de primitivas definidas apenas por funções polinomiais, uma consequência direta do critério de classificação por nós adotado, além de algumas outras questões como, por exemplo, a conversão entre representações das polinomiais da Base Algébrica para a Base de Bemstein.

4.2. BINTREES SIMPLICIAIS

Tradicionalmente, as células mais utilizadas nas decomposições espaciais, sejam elas regulares ou adaptativas, são cúbicas. O principal motivo desta escolha é o tradicional uso de sistemas de coordenadas cartesianas ortogonais, nos quais o cubo é a célula que apresenta a mesma simetria. Nestes sistemas, as representaçõesdas células cúbicas são bastante simples assim como os algoritmos para transformá-las, subdividí-las etc.

Para algumas finalidades, contudo, os cubos deixam um pouco a desejar, dificultando e as vezes até inviabilizando a realização de algumas tarefas. É o caso, por exemplo, da aproximação linear por partes de sólidos CSG. A apmximação afim para um sólido restrito a determinado cubo resulta em problema de interpolação cujo sistema é super-restrito e, em geral, não admite solução. Isto dificulta sobremaneira a tarefa de aproximar a casca do sólido eficientemente no interior de uma decomposição regular. Para todo cubo da decomposição deve ser feito estudo com os seus vizinhos de modo a não se comprometer a consistência global da aproximação. E interessante, portanto, a utilização de células mais simples na subdivisão do espaço. O trabalho de [MOOR90], por exemplo, apresenta uma técnica para construção de malhas uniformes de tetraedros.

As triangulações, por outro lado, oferecem excelentes condições para o cálculo das aproximações lineares, uma vez que a interpolação afim de determinada função no interior de um simplexo é única. Pode-se computar aproximação para a casca de um sólido, como vimos, triangulando-se a região de interesse e, em seguida, utilizando-se o Método de Continuação a partir de algum simplexo sabidamente interceptado pela fronteira deste sólido. Fixada a triangulação, esta apmximação é única.

A incoveniente necessidade de um ponto de partida para cada componente conexa do sólido, no entanto, prejudica substancialmente a eficiência do Método de Continuação. Convém notar que não são raras as situações nas quais o sólido é formado por mais de uma componente. A simples união de duas esferas disjuntas, por exemplo, forma um sólido constituído de duas componentes e a localização de uma delas em nada facilita a localização da outra. Em sólidos CSG mais complicados o pmblema é ainda mais sério, já que muitas vezes nem mesmo o número de componentes conexas consegue-se descobrir a *priori*. Este é um problema de difícil solução mesmo para funções $f:\mathbb{R}^2 \rightarrow \mathbb{R}$.

Um método experimental, onde percorre-se toda uma triangulação regular na tentativa de se localizar as componentes conexas também não parece ser solução muito razoável pois, dependendo do nível de refinamento da triangulação e do tamanho da região triangulada, a busca pode mostrar-se completamente inviável.

Parece-nos que solução razoável consiste em aliar as vantagens oferecidas pelas decomposições hierárquicas, na fase de localização do sólido no espaço, as propriedades dos simplexos, que facilitam muito a fase de cálculo da aproximação linear.

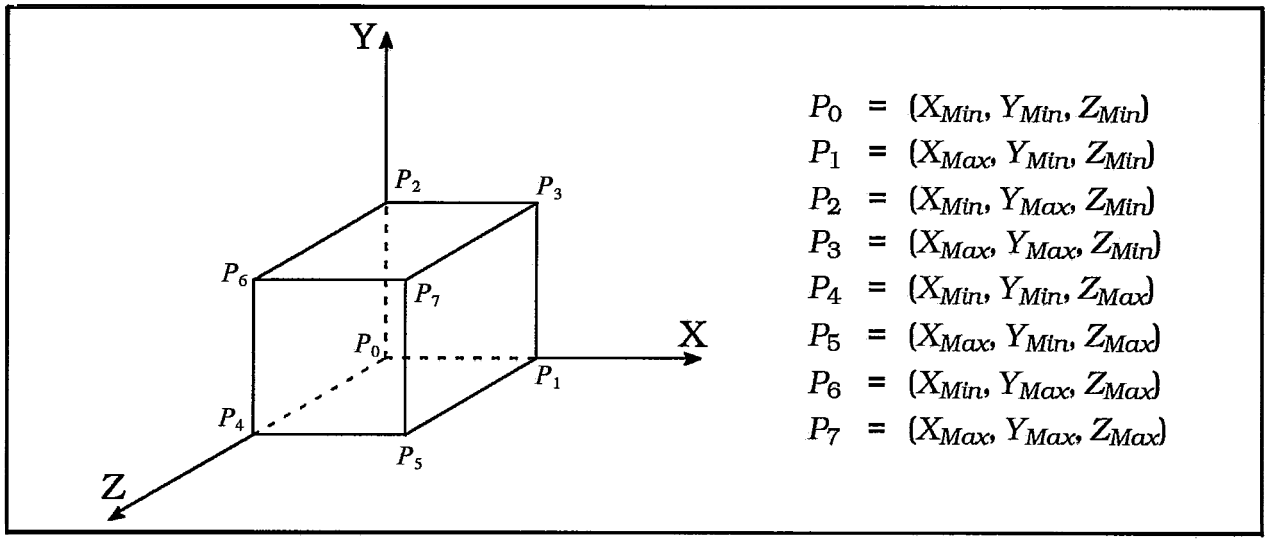


Fig. 4.1: sistema de referência para a caixa envolvente ao sólido.

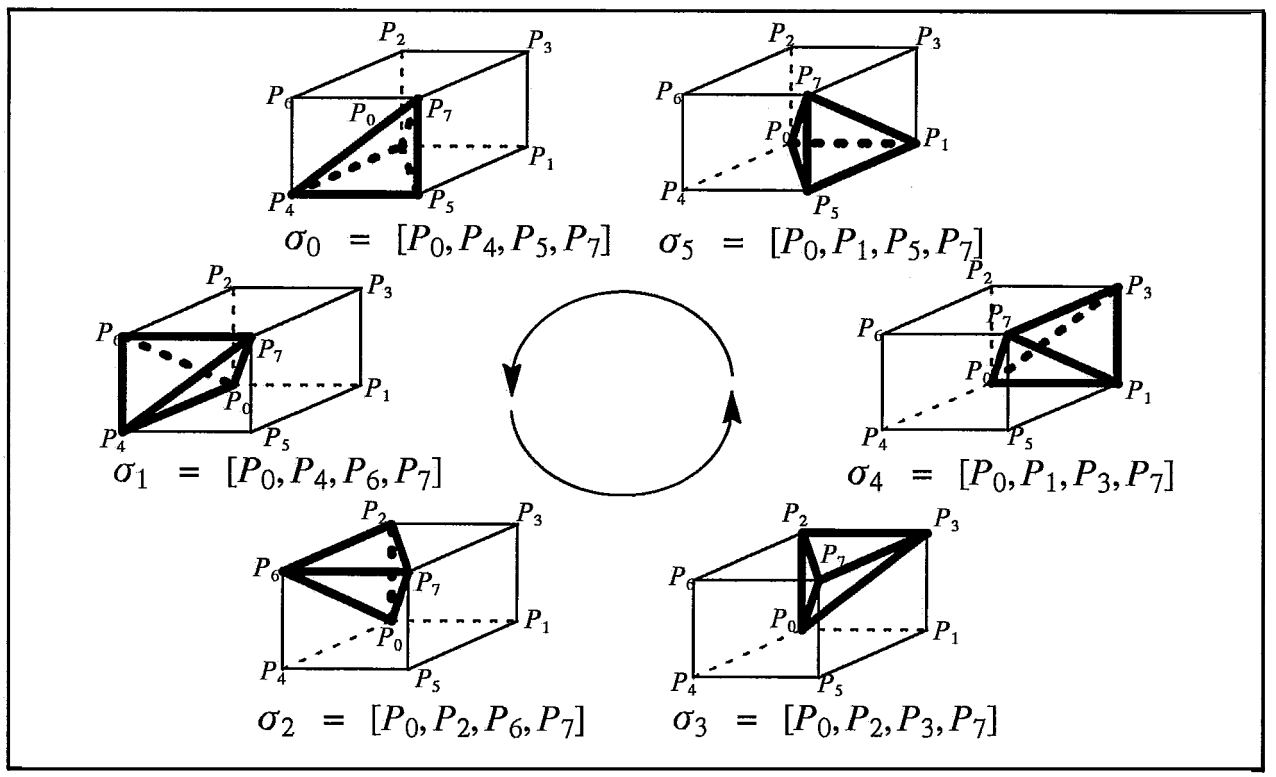


Fig. 4.2: subdivisão da caixa envolvente ao sólido nos 6 simplexos iniciais.

Apresentaremos, agora, uma decomposição hierárquica onde as células são simplexos que, obviamente, possuem tamanhos diferentes dependendo do nível de refinamento exigido por cada porção da região que se está avaliando. Esta decomposição é

derivada das triangulações CFK e é obtida através de subdivisões binárias dos simplexos que contêm parte da fronteira do sólido.

O sólido é inicialmente suposto dentro de uma "caixa envolvente" (*Bounding Box*). A noção de caixa envolvente varia muito conforme o contexto, mas para nós ela é a de um paralelepípedo que envolve completamente o sólido. Esta caixa é a região do espaço que deve ser decomposta e é tal como mostrada na figura 4.1.

A caixa envolvente é, então, subdividida em seis simplexos iniciais, formando o primeiro nível da decomposição hierárquica. Esta primeira subdivisão, mostrada na figura 4.2, é equivalente a que é feita com os cubos por uma triangulação CFK.

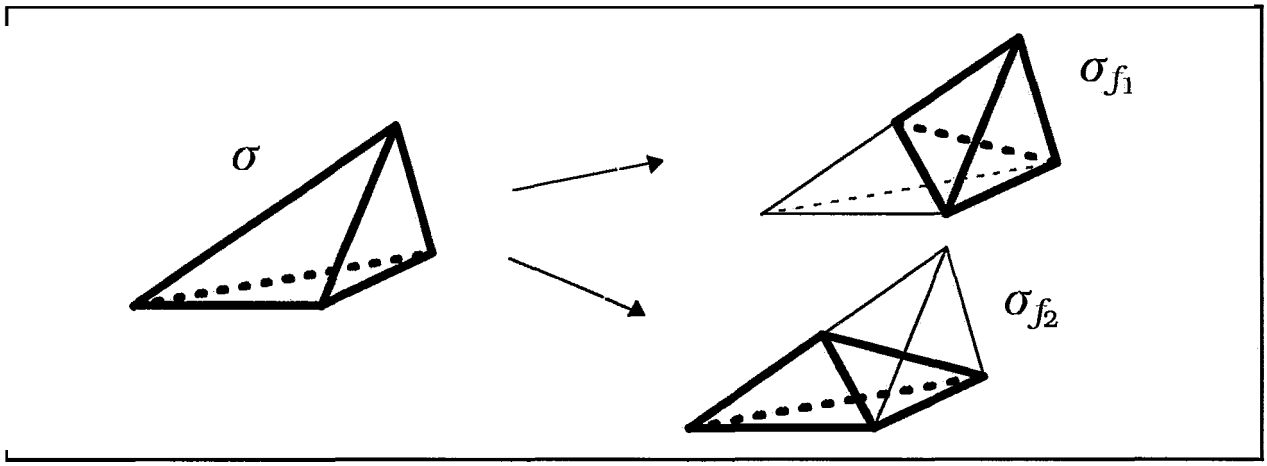


Fig. **4.3:** subdivisão binária da geometria de um simplexo.

A partir deste primeiro nível de decomposição, cada um dos seis simplexos iniciais é tratado independentemente dos demais. Cada simplexo é avaliado e, caso contenha parte da fronteira do sólido, é subdividido em dois simplexos filhos, em processo semelhante ao das Bintrees ([SAME90]). O processo de avaliação e eventual subdivisão continua recursivamente para cada simplexo filho. Um nível mínimo de subdivisão é arbitrado e, neste nível, é calculada a aproximação linear para a fronteira.

A subdivisão de cada simplexo em seus dois filhos é feita de maneira especial, que procura preservar o formato dos simplexos, evitando degenerações a medida que o processo de subdivisão progride. A figura 4.3 mostra a subdivisão. Admitindo que a caixa envolvente é um cubo, a maior aresta do simplexo é cortada ao meio por um plano que passa pela aresta oposta a ela. Este processo faz com que o menor ângulo sólido interno dos simplexos seja preservado.

A subdivisão sendo feita desta maneira, o formato dos simplexos varia muito pouco de um nível de subdivisão para outro e, consequentemente, não ocorrem degenerações onde os simplexos acabam ficando excessivamente achatados. Uma característica interessante desta subdivisão é que a cada d níveis (d= dimensão dos simplexos) o formato dos simplexos volta a ficar igual ao original. A figura 4.4 dá um exemplo no plano.

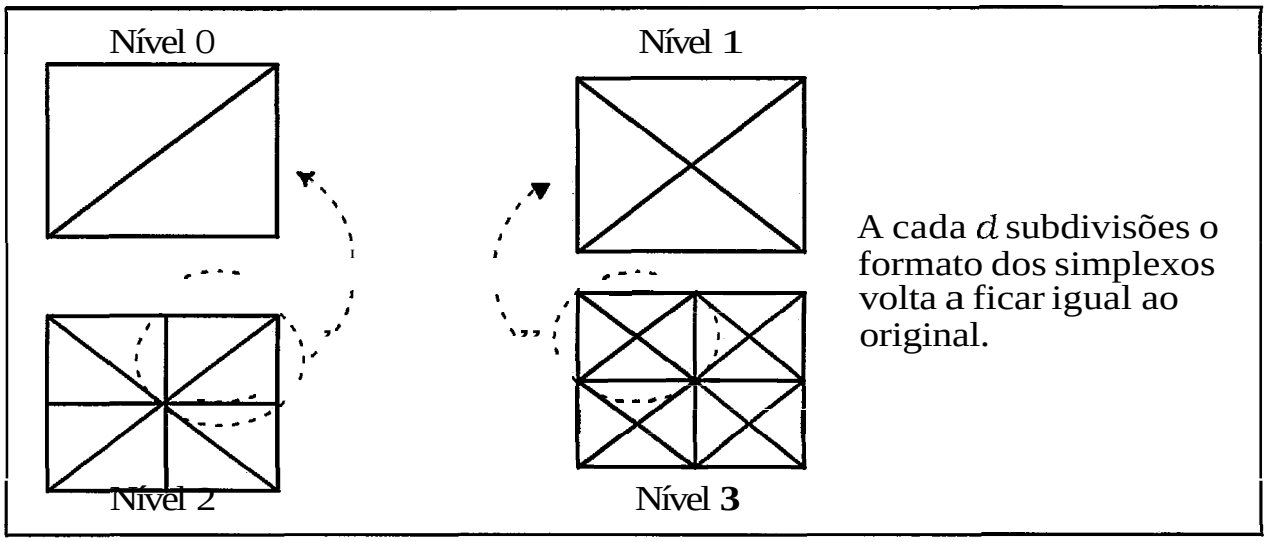


Fig. 4.4: a subdivisão faz a variação no formato dos simplexos ser cíclica.

No espaço, o processo de subdivisão gera três diferentes formatos dos simplexos, aos quais chamamos de tipo 0, 1 e 2. Os simplexos do tipo 0 geram, quando subdivididos, os de tipo 1 e estes, por sua vez, geram os de tipo 2 que, novamente, geram os de tipo 0. Assumindo que o simplexo pai é $o = \langle v_0, v_1, v_2, v_3 \rangle$, o esquema da subdivisão é dado na tabela 4.1.

índices dos vértices filhos	Pai de tipo 0 Filho de tipo 1		Pai de tipo 1 Filho de tipo 2		Pai de tipo 2 Filho de tipo 0	
	Filho 1	Filho2	Filho 1	Filho2	Filho 1	Filho2
0	v_0	v3	v_0	v_3	v_0	v_3
1	$(v_0 + v_3)/2$	$(v_0 + v_3)/2$	$(v_0 + v_3)/2$	$(v_0 + v_3)/2$	$(v_0 + v_3)/2$	$(v_0 + v_3)/2$
2	v_1	v_2	v_1	v_1	v_1	v_1
3	v_2	v_1	v_2	v_2	v_2	v_2

Tabela 4.1: esquema de subdivisão para cada tipo de simplexo.

Especificamente no nosso caso, como partimos de triangulação CFK da caixa envolvente, a cada três níveis de subdivisão temos nova triangulação CFK da caixa, com grau de refinamento maior (supondo que a decomposição hierárquica não interrompeu o processo de subdivisão para nenhum simplexo).

Na verdade, o que se tem estabelecido é um critério fixo de subdivisão para cada um dos três tipos de simplexo. Ao mencionarmos as propriedades da subdivisão, admitimos que a caixa envolvente ao sólido é um cubo, fazendo com que as arestas de cada simplexo da decomposição tenham tamanhos proporcionais aos mostrados na tabela 4.2.

Não existe nenhuma forte restrição a utilização de paralelepípedo ao invés de cubo como caixa envolvente. A subdivisão será feita da mesma forma, bastando-se imaginar o

Tipo(σ) \ Aresta	0	1	2	3	4	5
0	$\sqrt{3}$	$\sqrt{2}$	$\sqrt{2}$	1	1	1
1	$\sqrt{2}$	1	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{2}$
2	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$

Tabela 4.2: comprimentos relativos das arestas dos simplexos.

processo como sendo constituído de três fases: transformação do paralelepípedo em um cubo, cálculo dos simplexos iniciais para o cubo e transformação dos simplexos iniciais de volta para o espaço do paralelepípedo. Podemos pensar na subdivisão de cada simplexo desta forma também, ou seja, eles são levados em simplexos homeomorfos, pela transformação do paralelepípedo no cubo, estes simplexos são subdivididos pela metade de sua maior aresta e, em seguida, os simplexos filhos são mapeados de volta.

A deformação muito grande dos simplexos iniciais, causada pela utilização de um paralelepípedo muito achatado como caixa envolvente, deve ser, contudo, evitada. O processo de subdivisão binária tende a preservar o formato dos simplexos iniciais e, assim sendo, a decomposição será constituída apenas por simplexos achatados, o que pode levar a um comprometimento da qualidade da aproximação linear.

De posse deste esquema de subdivisão dos simplexos, podemos pensar em um primeiro algoritmo para o cálculo de aproximações lineares para a casca de sólidos CSG.

Algoritmo 4.1: AproxLinBintree

```
AproxLinBintree(C, S, NívelMax)
{
  {σ0, σ1, σ2, 03, 04, σ5} ← TriangCaixa(C)
  PARA i ⊂ 0 ATÉ 5 {
    GeraAproxLin(σi, S, NívelMax)
  I
} /* AproxLinBintree */

GeraAproxLin(σ, S, Nível)
{
  Classe ← ClassificaSimp(σ, S)
  SE ((Classe ≠ VAZIO) E (Classe ≠ CHEIO)) {
    SE (Nível > 0) {
      {σF1, σF2} ← SubdivSimp(σ)
      GeraAproxLin(σF1, S, Nível–1)
```

```
        GeraAproxLin(σF2, S, Nível–1)
      }
      SENÃO SE (Classe = F_INTERCEPTANTE) {
        ApLin ← CalcAproxLin(σ, S)
        TrataAproxLin(ApLin)
      }
    }
  } /* GeraAproxLin *l
```

O algoritmo recebe a caixa C, envolvente ao sólido S e o nível máximo de subdivisão de decomposição *NívelMax*. Inicialmente é feita a triangulação da **caixa** envolvente, através do procedimento *TriangCaixa* e os seis simplexos iniciais são gerados. Após isto, é feito o laço que trata individualmente cada um dos seis simplexos através de GeraAproxLin.

O procedimento GeraAproxLin é responsável pela geração da aproximação linear para a porção da casca do sólido S que está no interior do simplexo o. Inicialmente o simplexo é classificado em relação a *S* para se saber se ele não faz interseção com a casca do sólido e, por conseguinte, se ele não precisa ser subdividido e pode ser eliminado. No caso de não se poder garantir que ele não faz interseção, o procedimento verifica se ele ainda é relativamente grande, isto é, se o seu *Nível* ainda é maior que o nível mínimo de subdivisão arbitrado e, neste caso, subdivide* em seus dois filhos e chama–se recursivamente para cada um deles. Se, por outro lado, o simplexo já está no nível mínimo, o procedimento verifica se ocorre alternância nos sinais da função característica do sólido quando esta é aplicada aos vértices de *o* e, em caso afiativo, um pedaço da aproximação da casca é calculado através do pmcedimento *CalcAproxLin* e tratado por *TrataAproxLin.* Se o simplexo está no último nível mas não é F-interceptante, ele é descartado pois não é capaz de fornecer uma aproximação linear satisfatória.

As subdivisões recursivas dos simplexos fazem com que os vários pedaços da aproximação linear sejam computados em uma ordem a que estamos pouco acostumados mas isto não acarreta grandes problemas por três razões:

i) a aproximação linear dentro de cada simplexo é única
ii) a aproximação linear por partes é uma função contínua
iii) a decomposição cobre toda a região onde está a fronteira

4.3. LOCALIZAÇÃO DO SÓLIDO

As funções características, como vimos anteriormente, são expressões MIN–MAX, diretamente obtidas das árvores CSG, que descrevem o sólido globalmente. Elas ajudam a simplificar bastante algumas tarefas mas comprometem sensivelmente a realização de outras. Nem sempre é melhor trabalhar com o sólido completo. Às vezes é mais interessante ter um controle local, trabalhando somente com as partes relevantes para um determinado contexto.

Tanto para a classificação dos simplexos como para o cálculo de apmximações lineares dentro deles, é necessária a avaliação da função característica nos seus vértices. Para que a qualidade da aproximação seja boa, o refinamento da decomposição hierárquica nas regiões próximas da fronteira do sólido deve ser razoavelmente alto o que implica em um considerável número de simplexos gerados e, consequentemente, em um grande número de vértices onde a função característica deve ser avaliada.

Cada avaliação consiste de uma série de avaliações de expressões MIN–MAX, uma para cada operação booleana necessária a construção do sólido. Cada uma destas operações booleanas e responsável por parte do sólido que ocupa determinada região do espaço e que, portanto, não influencia em absolutamente nada as demais legiões cobertas pela decomposição. Em virtude disto, na maioria das vezes, a utilização da função característica completa para avaliar os vértices dos simplexos da decomposição constitui grave desperdício de memória e de tempo de processamento. A figura 4.5 mostra um exemplo onde haveria desperdício.

Convém notar que a função característica de um sólido constituído de apenas uma primitiva é a própria função que define esta primitiva.

Seria bastante interessante que durante a avaliação da função característica para determinado simplexo, somente estivessem representadas na função as primitivas que efetivamente influenciam a região onde o simplexo está localizado. Seria, portanto, bastante razoável que, para cada simplexo a se avaliar, fosse feita simplificação na árvore CSG do sólido e, consequentemente, na sua função característica, de modo a se localizar melhor a porção dele presente no simplexo.

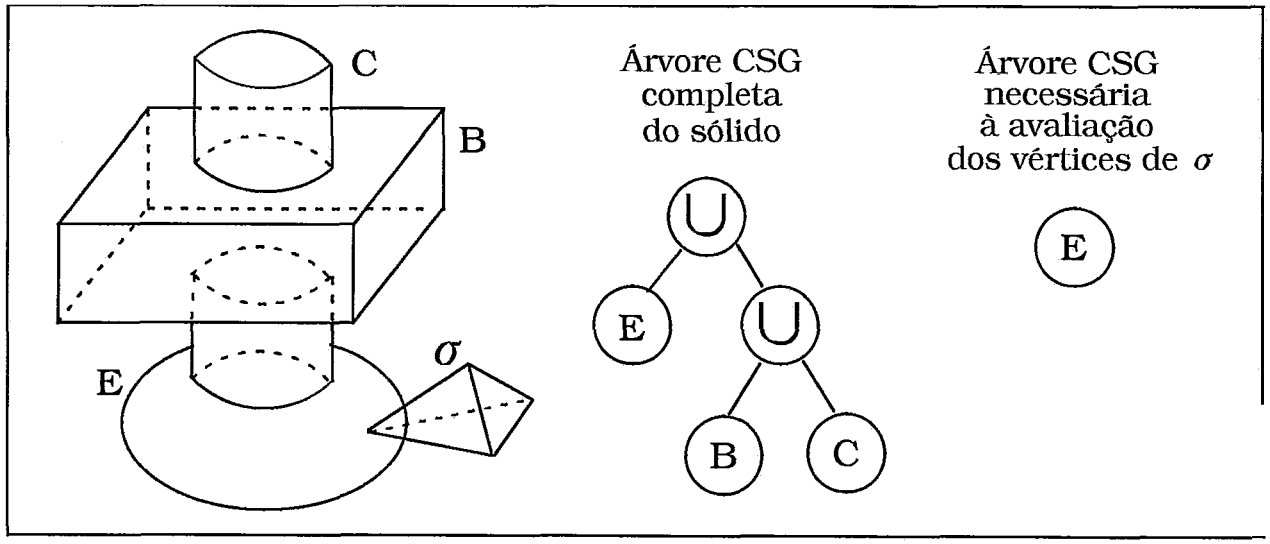


Fig. 4.5: exemplo de desperdício ao se utilizar a árvore CSG completa do sólido.

As idéias de localização para sólidos CSG já estão, nos dias de hoje, bastante desenvolvidas e, em especial, o trabalho de [TILO84] fornece subsídios suficientes para que as simplificações das árvores CSG, pertinentes em cada situação específica, possam ser feitas sem grandes dificuldades. Para que possamos adaptar estas idéias ao nosso

trabalho, no entanto, é necessário que adotemos uma classificação para os simplexos ligeiramente diferente da anterior. Os simplexos passam, agora, a ser dispostos nas quatro seguintes categorias: vazios, cheios, interceptantes e F-interceptantes.

Os simplexos vazios são aqueles inteiramente fora do sólido e os cheios, por outro lado, são aqueles inteiramente no seu interior. As classificações de interceptantes e F-interceptantes são exatamente iguais as da definição anterior, com a primeira sendo dada para aqueles simplexos que contêm parte da fronteira do sólido e a segunda para aqueles que, além de interceptantes, apresentam alternância nos sinais dos valores da função característica aplicada aos seus vértices.

As idéias de simplificação das árvores CSG são bastante simples e baseiam-se em algumas regras da teoria dos conjuntos que envolvem os conjuntos vazio ($) e universo (W):

$$A \cap \emptyset = \emptyset$$
$$A \cap W = A$$
$$A \cup \emptyset = A$$
$$A \cup W = W$$

onde A é qualquer conjunto. A tabela 4.3 mostra como são feitas as classificações.

∪	V	C	I
V	V	C	I
C	C	C	C
I	I	C	I

V – VAZIO
C – CHEIO
I – INTERCEPTANTE

∩	V	C	I
V	V	V	V
C	V	C	I
I	V	I	I

Tabela 4.3: classificação dos simplexos perante as operações booleanas

Basicamente, procura-se localizar as subárvores da árvore CSG que representam porções do sólido que podem perfeitamente ser substituídas pelo conjunto vazio ou pelo conjunto universo. Segundo [TILO84], estas porções são $-redundantes ou W-redundantes, respectivamente.

As primitivas da árvore CSG do sólido são avaliadas em relação ao simplexo e classificadas em uma das quatro categorias mencionadas. As operações booleanas entre elas são então avaliadas e, caso algumas sejam ϕ-redundantes ou W-redundantes, simplificações podem ser feitas. No nosso caso, como estamos inteiessados somente na fronteira do sólido, uma primitiva é ϕ-redundante sempre que o simplexo for considerado vazio em relação a ela e é W-redundante sempre que ele for classificado como cheio. A figura 4.6 mostra um exemplo de localização.

Utilizando as idéias de localização podemos otimizar sensivelmente o algoritmo de decomposição hierárquica pois, a medida que os simplexos vão sendo subdivididos, as

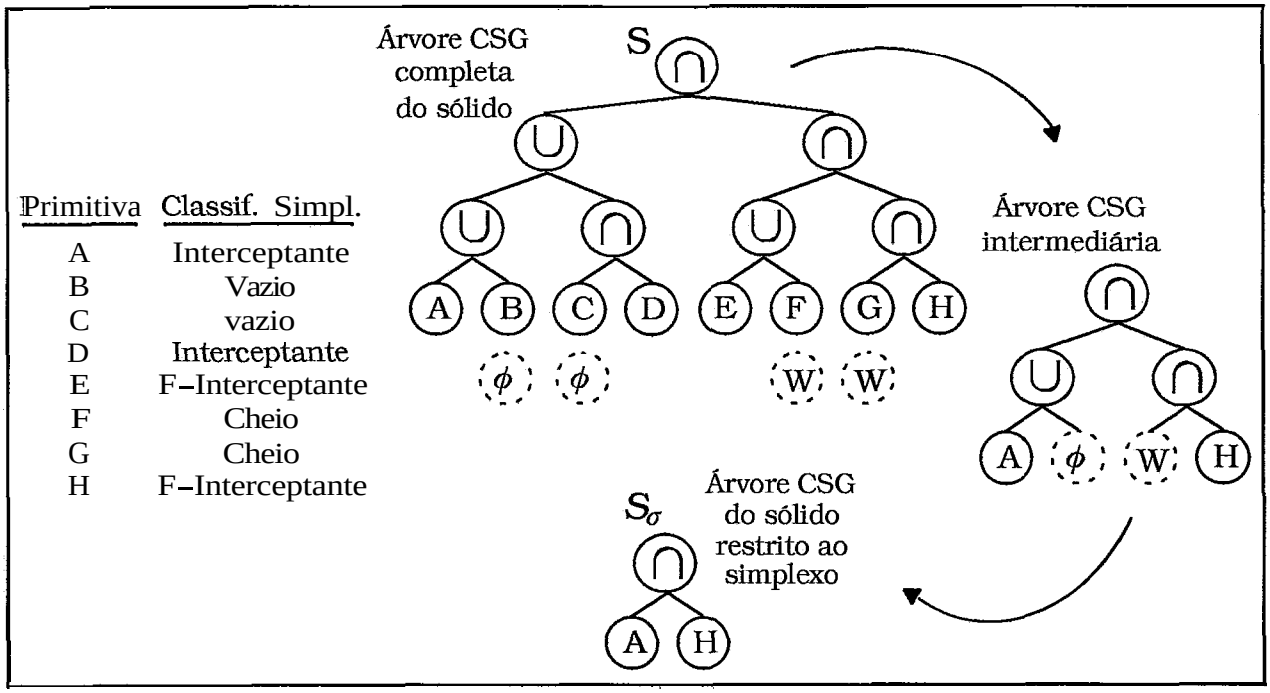


Fig. 4.6: exemplo de simplificação de uma árvore CSG.

árvores CSG representativas do sólido em cada um deles vão sendo bastante simplificadas, o mesmo acontecendo com as respectivas funções características. A utilização das expressões MIN–MAX, por conseguinte, fica reduzida a algumas poucas regiões do espaço onde existem mais de urna primitiva na árvore CSG do sólido restrito ao simplexo, já que nas demais regiões apenas uma primitiva é responsável pela fronteira do sólido. Segue–se abaixo um algoritmo de classificação de simplexos perante árvores CSG que, utilizando os conceitos de localização, simultaneamente simplifica a árvore, restringindo o sólido ao simplexo.

Algoritmo 4.2: ClassSimplifCSG

```
ClassSimplifCSG(S, o)
{
  SE (Simples(S)) {
    RETORNE(ClassPrim(Primitiva(S), o))
  }

  ClasseEsq ← ClassSimplifCSG(FilhoEsquerdo(S), o)
  SE (((ClasseEsq = VAZIO) E (Operador(S) = INTERSEÇÃO)) OU
      ((ClasseEsq = CHEIO) E (Operador(S) = UNIÃO)))

    EliminaCSG(S)
    RETORNE(ClasseEsq)
  }
```

```
SENÃO SE ((ClasseEsq = VAZIO) OU (ClasseEsq = CHEIO)) {
  EliminaFilhoCSG(S, ESQUERDO)
  S ← FilhoDireito(S)
  RETORNE(ClassSimplifCSG(S, o))
}

ClasseDir ← ClassSimplifCSG(FilhoDireito(S), σ)
SE (((ClasseDir = VAZIO) E (Operador(S) = INTERSEÇÃO)) OU
    ((ClasseDir = CHEIO) E (Operador(S) = UNIÃO)))
{
  EliminaCSG(S)
  RETORNE(ClasseDir)
}

SENÃO SE ((ClasseDir = VAZIO) OU (ClasseDir = CHEIO)) {
  EliminaFilhoCSG(S, DIREITO)
  S ← FilhoEsquerdo(S)
  RETORNE(ClasseEsq)
}

RETORNE(INTERCEPTANTE ou F-INTERCEPTANTE)
} /* ClassSimplifCSG */
```

O algoritmo recebe a árvore CSG S do sólido e o simplexo o a ser classificado. Ele é obviamente recursivo e, por isto, dividido em duas partes. A primeira parte cuida da condição de parada da recursividade que, no caso, classifica as folhas da árvore, ou seja, as primitivas CSG.

A segunda parte trata dos nós internos da árvore, isto é, das operações booleanas responsáveis pela composição do sólido. Para cada nó interno o procedimento chama-se recursivamente para classificar o simplexo perante as duas subárvores filhas. De posse das classificações, ele verifica as possibilidades de simplificação.

Inicialmente o algoritmo verifica se o sólido é composto de apenas urna primitiva, através do procedimento *Simples*. Em caso afirmativo, o simplexo é classificado perante esta primitiva e esta classificação é retornada. Se, no entanto, o sólido não for simples, o simplexo é classificado recursivamenteperante a subárvore esquerda. Dependendo desta classificação e do operador booleano plesente no nó CSG que está sendo avaliado, a árvore completa a partir deste nó pode ser eliminada (classificação VAZIO com operador $\cap$, por exemplo) através do procedimento *EliminaCSG* ou a subárvore esquerda do nó pode ser eliminada (classificação CHEIO com operador $\cup$, por exemplo) através do procedimento *EliminaFilhoCSG*. Se não houver simplificações, o simplexo é considerado interceptante em relação a subárvore esquerda. É necessário, portanto, que uma análise semelhante seja feita com a subárvore direita. Caso também não ocorram simplificações no ramo

direito, o simplexo é interceptante em relação a árvore completa, restando apenas a avaliação final para se saber se ele é F-interceptante.

Além das simplificações nas árvores CSG de cada simplexo a medida que o processo de subdivisão progride, existe ainda outra otimização que pode ser feita em determinados casos. Sempre que a árvore do sólido dentro do simplexo for simples, isto é, composta de apenas uma primitiva e, além disto, esta primitiva for linear, não existe a necessidade de se continuar o processo de subdivisão. A decomposição hierárquica pode ser interrompida porque a primitiva linear só pode ser aproximada de uma única maneira independentemente do tamanho do simplexo.

Pormenor interessante, a cerca da implementação da simplificação das árvores CSG, é a necessidade de uma simplificação adicional nos simplexos mínimos. Os simplexos vizinhos não necessariamente contêm a mesma árvore simplificada e, portanto, na fase de cálculo de aproximação linear, precisamos simplificar novamente a árvore contra as arestas dos simplexos mínimos para garantir que somente as primitivas interceptantes estarão representadas na função característica do sólido.

Utilizando estas duas formas de otimização - a simplificação da árvore CSG e o critério de parada da subdivisão para simplexos que contêm somente primitivas lineares - podemos repensar o algoritmo de cálculo da aproximação linear para a casca do sólido CSG baseado em decomposição hierárquica.

Algoritmo 4.3: AproxLinBintree

```
AproxLinBintree(C, S, NívelMax)
{
  {σ0, σ1, σ2, σ3, σ4, σ5} ← TriangCaixa(C)
  PARA i ← 0 ATÉ 5 {
    S̄ ← CopiaCSG(S)
    Classe ← ClassSimplifCSG(S̄, σi)
    SE ((Classe ≠ VAZIO) E (Classe ≠ CHEIO)) {
      GeraAproxLin(σi, S̄, NívelMax)
    }
  }
} /* AproxLinBintree */

GeraAproxLin(σ, S, Nível)
{

  {σF1, σF2} ← SubdivSimp(σ)
  S̄ ← CopiaCSG(S)
  Classe ← ClassSimplifCSG(S̄, σF1)
```

```
SE ((Classe ≠ VAZIO) E (Classe ≠ CHEIO)) {
  SE ((Nível > 1) E (NÃO PrimitivaLinear(S̄))) {
    GeraAproxLin(σF1, S̄, Nível–1)
  }
  SENÃO SE (Classe = F–INTERCEPTANTE) {
    ApLin ← CalcAproxLin(σF1, S̄)
    TrataAproxLin(ApLin)
  }
}

S̄ ← CopiaCSG(S)
Classe ← ClassSimplifCSG(S̄, σF2)
SE ((Classe ≠ VAZIO) E (Classe ≠ CHEIO)) {
  SE ((Nível > 1) E (NÃO PrimitivaLinear(S̄))) {
    GeraAproxLin(σF2, S̄, Nível–1)
  }
  SENÃO SE (Classe = F–INTERCEPTANTE) {
    ApLin ← CalcAproxLin(σF2, S̄)
    TrataAproxLin(ApLin)
  }
}
} /* GeraAproxLin */
```

O algoritmo é basicamente igual ao 4.1 com as modificações que otimizam o processo de subdivisão. Deve–se ressaltar, todavia, a necessidade de se fazer cópias da árvore CSG antes de se classificar um simplexo, uma vez que o procedimento de classificação (4.2) pode fazer simplificações, eliminando diversos nós da árvore. Outro detalhe relevante é que o procedimento *PrimitivaLinear* verifica duas coisas: se a árvore CSG é simples e, neste caso, se a primitiva presente no seu único nó é linear. É importante notar também que, assim como no algoritmo 4.1, o nível da decomposição onde são calculadas as aproximações lineares dentro dos simplexos é sempre o nível 0, a menos que o processo de subdivisão tenha sido interrompido precocemente pelo fato de alguma primitiva linear ter sido encontrada.

4.4. CRITÉRIOS DE CLASSIFICAÇÃO DOS SIMPLEXOS

O algoritmo de aproximação linear por partes para a casca de sólidos CSG através de subdivisão simplicial adaptativa tem sua eficiência (ou até mesmo sua existência) diretamente vinculada a existência de critérios seguros de classificação dos simplexos com respeito as interseções destes com a casca do sólido. A segurança da classificação, por sua vez, é diretamente relacionada com o nível de tolerância requerido pela aplicação e, por conseguinte, com o tamanho dos simplexos da subdivisão.

O critério de classificação dos simplexos é a chave do algoritmo pois, somente com ele conseguimos categorizar os simplexos e, a partir daí, reduzir drasticamente a carga de trabalho, ou seja, o processamento exclusivo daqueles simplexos realmente relevantes para fins da aproximação linear.

Classificar árvores CSG, como vimos no algoritmo 4.2, consiste, em última análise, da classificação de suas primitivas. A avaliação é feita por partes, das folhas para a raiz da árvore, obtendo-se a classificação de cada nó a partir das classificações de suas duas subárvores filhas.

A avaliação do comportamento de uma primitiva no interior de determinado simplexo é, geralmente, tarefa bem complicada que exige conhecimento razoável do comportamento da função que a define na região delimitada pelo simplexo. O conhecimento do gradiente da função e as vezes até mesmo o conhecimento de derivadas de ordens mais elevadas se fazem necessários. De posse do gradiente da função, por exemplo, podemos estipular limites para a variação da função no interior do simplexo e, por conseguinte, estabelecer um primeiro critério de classificação como descrevemos a seguir.

4.4.1. CRITÉRIO DE LIPSCHITZ

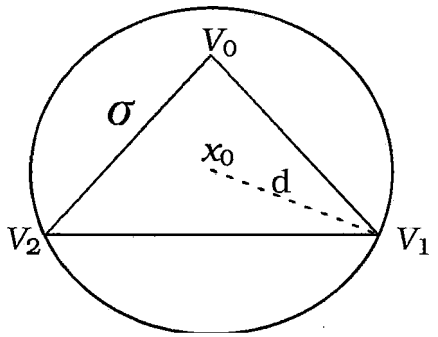
Em tentativa de melhorar a técnica de lançamento de raios para superfícies definidas implicitamente, [KALR89] propõe a subdivisão adaptativa do espaço cujo critério de classificação dos voxels utiliza basicamente a constante de Lipschitz da função definidora da superfície.

Uma constante de Lipschitz L de urna função f numa região X é qualquer real tal que, para quaisquer dois pontos x_1, x_2 em X, observa-se $\| f(x_1)-f(x_2) \| \leq L \| x_1-x_2 \|$. Para as funções diferenciáveis, portanto, seu menor valor possível é o supremo do gradiente de f, em norma, na região X.

$$L \geq Max\{\nabla f(x); \quad x \in X\}$$

No trabalho de [KALR89] utilizam-se células retangulares (caixas) mas podemos adaptar a técnica para avaliação de simplexos. A classificação de um simplexo o, baseada na constante de Lipschitz L de f em o, pode ser feita da seguinte forma: seja x_0 o baricentro de o e d a maior das distâncias entre x_0 e os vértices de o, o par (x_0, d) define uma bola envolvente a o. Como a maior taxa de variação de f em o é dada por L, a maior variação possível no valor de f a partir de $f(x_0)$ é dada pelo produto Ld. Se verificarmos, portanto, que $\| f(x_0) \| > Ld$ então saberemos que f, garantidamente, não se anula em nenhum ponto de o. A figura 4.7 ilustra o caso no R^2.

Este critério é bastante seguro na medida em que nenhum simplexo que contenha parte da supefície definida pela função f será classificado como não-inteiceptante. Ele é, também, bem geral, uma vez que pode ser aplicado para vasta gama de funções. Essas



x_0 = Baricentro de a

L = Constante de Lipschitz p/f em a

$\|f(x_0)\| > Ld \Rightarrow$ NÃO INTERCEPTA

Fig. 4.7: utilização do critério de Lipschitz no plano.

aparentes vantagens, no entanto, ficam comprometidas na prática pelo fato do cálculo da constante de Lipschitz ser, em geral, bastante oneroso e, além disto, o critério não é muito estrito, deixando de eliminar grande quantidade de simplexos que não interceptam a superfície. Isto se deve ao fato da constante fornecer limites muito relaxados para a taxa de variação das funções.

Para algumas classes de funções, a tarefa de maximizar o gradiente é simplificada: são aquelas funções cujo módulo do gradiente é uma função convexa. De acordo com [ROCK72], toda função convexa em um domínio convexo assume o máximo na fronteira deste domínio. Em particular, se o domínio for um poliedro convexo, o máximo ocorrerá em um vértice. Maximizar $\|\nabla f(x)\|$ é equivalente a maximizar $\|\nabla f(x)\|^2$ e como os simplexos são domínios poliédricos convexos, se $\|\nabla f(x)\|^2$ for convexa, a menor constante de Lipschitz para f resulta do maior valor desse quadrado nos vértices do simplexo. A classe para as quais o critério pode ser utilizado sem muitas dificuldades na avaliação da constante, portanto, é a das quadráticas. Classes mais amplas são objeto de investigação.

Para ampliar a classe das funções tratáveis, necessitamos de critérios mais gerais e que sejam, também, mais fáceis de serem verificados. A classificação dos simplexos é frequentemente utilizada em cada nível da decomposição e um critério oneroso poderia comprometer todas as vantagens oferecidas pelo caráter adaptativo da subdivisão. É muito importante que as classificações sejam feitas rapidamente para que haja ganho efetivo de velocidade computacional em relação ao método tradicional, que avalia todos os vértices da triangulação.

Parece-nos, entretanto, que a combinação desses dois aspectos – generalidade do critério e facilidade de verificação – é muito difícil de ser alcançada posto que eles são quase que mutuamente exclusivos. Devemos, pois, abdicar de alguns pontos nos dois aspectos e nos contentar com o critério que não seja tão geral mas que seja razoavelmente fácil de se verificar.

Partindo deste princípio, se aceitarmos que o nosso modelador de sólidos trabalhe somente com primitivas definidas por funções polinomiais, isto é, primitivas que são semiespaços polinomiais, podemos estabelecer um critério bastante eficiente de classifxação dos simplexos. A restrição as primitivas definidas por polinomiais não é tão

drástica na prática se levarmos em conta que a grande maioria dos modeladores utiliza apenas um subconjunto das quadráticas. Pelo contrário, a possibilidade de utilização de polinomiais quaisquer constitui, isto sim, avanço na medida em que aumenta o poder de expressão da maioria dos modeladores.

4.4.2. CRITÉRIO DE BÉZIER

Este critério de classificação é fundado na representação da polinomial na Base de Bernstein generalizada. Esta base tem se mostrado, por suas propriedades, muito adequada a representação de curvas e superfícies paramétricas polinomiais ([FARI86] e [FARI88]). Ela apresenta, contudo, propriedades também adequadas a aferição da existência de raízes de equações polinomiais.

A Base de Bernstein para polinômios trivariados de grau m é definida por:

$$B^m_{ijkl}(t,u,v,w) = \frac{m!}{i!j!k!l!} t^i u^j v^k w^l \; ; \qquad t+u+v+w = 1 \;\; e \;\; i+j+k+l = m$$

Por ser base do espaço de polinômios de grau m, todo polinômio P deste espaço pode ser expresso na forma:

$$P(t,u,v,w) = \sum_{i+j+k+l=n} C_{ijkl} B^m_{ijkl}(t,u,v,w)$$

onde a quádrupla (t,u,v,w) representa as coordenadas baricêntricas dos pontos de E^3 em relação a um simplexo de referência. Uma propriedade muito interessante desta base é o fato que os seus elementos, isto é, os polinômios B^m_{ijkl}, são positivos no interior desse simplexo de referência, além de possuirem soma unitária. Deste modo, se o polinômio se anula no interior do simplexo, ele necessariamente possui alguns coeficientes de Bézier negativos e alguns outros positivos. Na verdade, para toda polinomial P em qualquer simplexo de referência o, verifica-se:

$$\forall x \in \sigma \quad Min\{C_{ijkl}\} \le Min\{P(x)\} \le Max\{P(x)\} \le Max\{C_{ijkl}\}$$

onde C_{ijkl} são os coeficientes de Bézier de P com respeito a σ.

Isto significa que os valores extremos dos coeficientes de Bézier da polinomial são limitantes para a própria polinomial no interior do simplexo de referência e, a partir daí, podemos estabelecer o seguinte critério de classificação de simplexos contra primitivas polinomiais: se os valores extremos dos coeficientes de Bézier forem positivos ou negativos então a fronteira da primitiva não intercepta o simplexo. Mais ainda, se os extremos forem positivos então a primitiva exclui o simplexo e este pode ser classificado como VAZIO. Analogamente, se os extremos são negativos então a primitiva inclui o simplexo e este é CHEIO.

O fato dos coeficientes extremos apresentarem sinais distintos não garante, em geral, que exista interseção do simplexo com a fronteira da primitiva. Não é raro acontecer de alguns dos coeficientes de Bézier terem sinais opostos e, no entanto, não haver interseção. Na figura 4.8 temos exemplificados os casos possíveis de classificação pelo critério de Bézier. Neste exemplo consideramos, sem perda de generalidade, os simplexos da reta e uma polinomial cúbica.

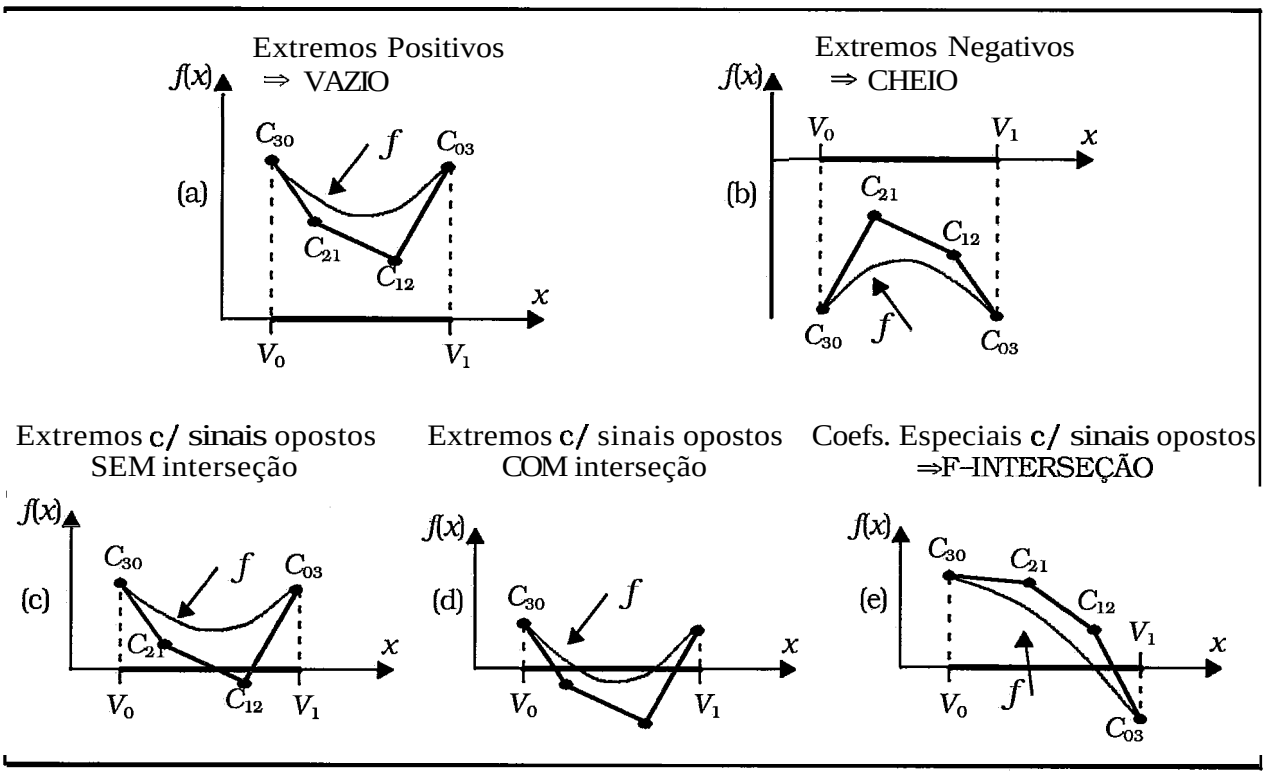


Fig. 4.8: classificações para os simplexos da reta pelo critério de Bézier.

Os casos mostrados na figura 4.8(a) e 4.8(b) são típicos de quando o simplexo é garantidamente não-interceptante. As figuras 4.8(c) e 4.8(d) mostram situações nas quais pairam dúvidas. Nestes casos, nenhuma afiiativa pode ser feita com segurança visto que a interseção pode acontecer ou não.

Existe um caso, entretanto, onde a diferença de sinal entre os extremos dos coeficientes de Bézier assegura que o simplexo é interceptante. Mais do que isto, existe a garantia de que ocorre uma F-interseção e o simplexo pode ser utilizado para fins de aproximação linear. Este caso, mostrado na figura 4.8(e), se deve a uma propriedade bastante interessante dos coeficientes de Bézier. Alguns deles são especialmente importantes por coincidirem com o valor da polinomial nos vértices do simplexo.

Como vimos na seção 2.4, nos vértices dos simplexos somente uma das coordenadas baricêntricas é não-nula e igual a 1. As funções da Base de Bemstein neste vértices, por conseguinte, são todas nulas com exceção de uma que assume o valor 1. O coeficiente de Bézier associado a esta única função de base não-nula coincide, portanto, com o valor da polinomial no vértice. No R^n para uma polinomial de grau m, estes coeficientes especiais

são: $\{C_{m0\ldots0}, C_{0m\ldots0}, \ldots, C_{00\ldots m}\}$. No R^3, onde os simplexos são tetraedros da forma $\sigma = \langle v_0,v_1,v_2,v_3\rangle$, os valores de uma polinomial P de grau m nos vértices de σ são dados por: $P(v_0)=C_{m000}$, $P(v_1)=C_{0m00}$, $P(v_2)=C_{00m0}$ e $P(v_2)=C_{000m}$.

Esse critério de classificação dos simplexos, é muito mais eficiente do que o critério de Lipschitz para polinomiais, pois fornece limites bem melhores para a variação das primitivas no interior dos simplexos. A grosso modo, isto pode ser explicado pelo fato de, em geral, o poliedro de controle ser uma estimativa razoável para a superfície Bézier por ele gerada.

4.4.3. CRITÉRIO DE RIVLIN

Apesar da qualidade das classificações apresentarem considerável melhora com a utilização do critério de Bézier, ele ainda deixa a desejar em algumas situações nas quais o comportamento da primitiva varia muito bruscamente dentro do simplexo. Mais especificamente, quando a derivada segunda da polinomial que define a primitiva é muito grande no interior do simplexo, os limites fornecidos pelos extremos dos coeficientes de Bézier são muito dilatados, dando margem a algumas classificações duvidosas. É o caso, por exemplo, da figura 4.9 onde vemos um elipsóide excessivamente achatado para qual o critério de Bézier forneceria limites muito relaxados, quando da classificação dos simplexos próximos aos "bicos".

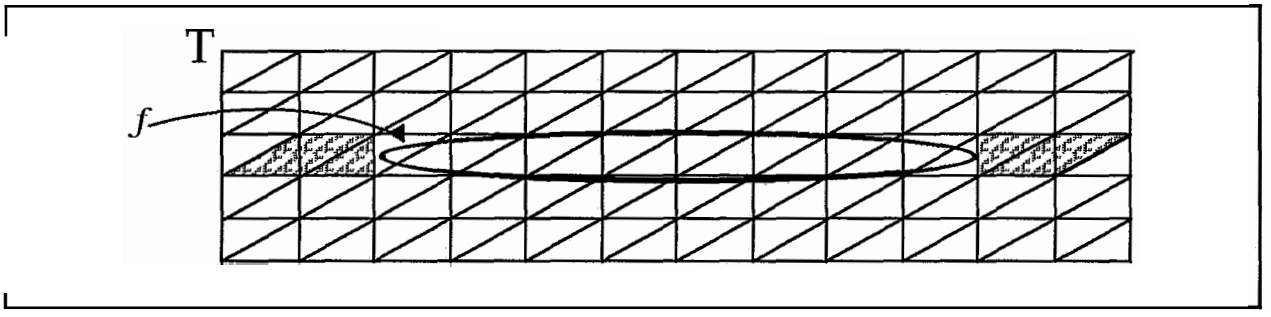


Fig. 4.9: o critério de Bézier não é muito eficiente em alguns casos.

O trabalho de [RIVL70] propõe um critério mais "apertado" que o de Bézier para polinômios univariados mas pode, perfeitamente, ser estendido para o caso n-dimensional. A idéia é não considerar somente os dois coeficientes extremos mas também o segundo maior (ou o segundo menor), obtendo avaliação mais precisa do comportamento da derivada segunda da polinomial no interior do simplexo.

O critério de Bézier estabelece cotas para a variação da polinomial em uma região:

$$\alpha = Min\{C_{ijkl}\} \leq \mu \leq M \leq Max\{C_{ijkl}\} = A$$

onde μ e M são os valores mínimo e máximo assumidos pelo polinômio na região. Estas cotas, entretanto, podem ser ainda melhores, bastando-se encontrar novas cotas β e B tais que:

$$\alpha < \beta \leq \mu \leq M \leq B < A$$

Seja a polinomial $P:R^d \rightarrow R$ de grau n, expressa na Base de Bernstein em relação ao simplexo o na forma:

$$P(\mathbf{u}) = \sum_{\|\mathbf{i}\|=n} C_{\mathbf{i}} \, B_{\mathbf{i}}^{n}(\mathbf{u}) \qquad \begin{cases} \mathbf{u} = (u_0, \; u_1, \; \ldots, \; u_d) \\ \mathbf{i} = (i_0, \; i_1, \; \ldots, \; i_d) \end{cases}$$

$$B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{n!}{i_0! \; i_1! \; \ldots \; i_d!} \; u_0^{i_0} \; u_1^{i_1} \; \ldots \; u_d^{i_d} \; ; \; i_0 + i_1 + \ldots + i_d = n$$

onde $\{C_{\mathbf{i}}\}$ são os coeficientes de Bézier de P e $B_{\mathbf{i}}^{n}(\mathbf{u})$ é uma função da Base de Bemstein d–dimensional. Se M é o máximo de P em o e $C_{\mathbf{m}}$ e $C_{\mathbf{l}}$, respectivamente, o maior e o segundo maior coeficientes de Bézier, então:

$$M \leq (C_{\mathbf{m}} - C_{\mathbf{l}}) \; B_{\mathbf{m}}^{n}\left(\frac{\mathbf{m}}{n}\right) + C_{\mathbf{l}} \leq C_{\mathbf{m}}$$

Demonstração:

Se m é índice de algum vértice então a relação é imediata pois $C_{\mathbf{m}} = M$. Senão, pelas propriedades da Base de Bemstein, sabemos que $\forall \, x \in o \; 0 \leq B_{\mathbf{i}}^{n}(\mathbf{x}) \leq 1$ (quando i não é índice de vértice) e, além disto, $B_{\mathbf{m}}^{n}(\mathbf{x}) < 1$. Retirando do somatório o termo relativo ao maior coeficiente e dividindo o restante por $(1 - B_{\mathbf{m}}^{n}(\mathbf{u}))$, temos:

$$P(\mathbf{u}) = C_{\mathbf{m}} \; B_{\mathbf{m}}^{n}(\mathbf{u}) + (1 - B_{\mathbf{m}}^{n}(\mathbf{u})) \sum_{\substack{\|\mathbf{i}\|=n \\ \mathbf{i} \neq \mathbf{m}}} \frac{C_{\mathbf{i}} \; B_{\mathbf{i}}^{n}(\mathbf{u})}{(1 - B_{\mathbf{m}}^{n}(\mathbf{u}))}$$

Como:

$$\frac{B_{\mathbf{i}}^{n}(\mathbf{u})}{(1 - B_{\mathbf{m}}^{n}(\mathbf{u}))} \geq 0 \quad \text{e} \quad \sum_{\substack{\|\mathbf{i}\|=n \\ \mathbf{i} \neq \mathbf{m}}} \frac{B_{\mathbf{i}}^{n}(\mathbf{u})}{(1 - B_{\mathbf{m}}^{n}(\mathbf{u}))} = 1$$

então podemos concluir que:

$$\sum_{\substack{\|\mathbf{i}\|=n \\ \mathbf{i} \neq \mathbf{m}}} \frac{C_{\mathbf{i}} \; B_{\mathbf{i}}^{n}(\mathbf{u})}{(1 - B_{\mathbf{m}}^{n}(\mathbf{u}))} \leq C_{\mathbf{l}}$$

e daí:

$$P(\mathbf{u}) \leq B_{\mathbf{m}}^{n}(\mathbf{u}) \; C_{\mathbf{m}} + (1 - B_{\mathbf{m}}^{n}(\mathbf{u})) \; C_{\mathbf{l}} = (C_{\mathbf{m}} - C_{\mathbf{l}}) \; B_{\mathbf{m}}^{n}(\mathbf{u}) + C_{\mathbf{l}}$$

Como $B_{\mathbf{i}}^{n}(\mathbf{u})$ assume máximo em $\mathbf{u} = \frac{\mathbf{m}}{n}$, obtemos:

$$M \leq (C_{\mathbf{m}} - C_{\mathbf{l}}) \; B_{\mathbf{m}}^{n}\left(\frac{\mathbf{m}}{n}\right) + C_{\mathbf{l}} \leq C_{\mathbf{m}} \quad \blacklozenge$$

Analogamente, se considerarmos agora C_p e C_q como o menor e o segundo menor coeficientes de Bézier, obteremos:

$$C_p \geq (C_p - C_q)\, B_p^n\left(\frac{p}{n}\right) + C_q \geq \mu$$

O critério de Rivlin nos fornece, portanto, uma cota inferior $\beta = (C_p - C_q)\, B_p^n\left(\frac{p}{n}\right) + C_q$ e uma outra superior $B = (C_m - C_l)\, B_m^n\left(\frac{m}{n}\right) + C_l$ melhores que os valores extremos dos coeficientes.

Baseados, então, neste novo critério, podemos definir um algoritmo seguro de classificação de simplexos contra primitivas.

Algoritmo 4.4: ClassPrim

```
ClassPrim(n, Cijkl, Nível)
{
  PerturbaVérticesNulos(n, Cijkl)
  Fv0 ← Cn000
  Fv1 ← C0n00
  Fv2 ← C00n0
  Fv3 ← C000n

  SE ((Fv0>0) E (Fv1>0) E (Fv2>0) E (Fv3>0)) {
    Classe ← VAZIO
  }

  SENÃO SE ((Fv0<0) E (Fv1<0) E (Fv2<0) E (Fv3<0)) {
    Classe ← CHEIO
  }
  SENÃO {
    Classe ← F–INTERCEPTANTE
  }

  SE ((Nível>0) E ((Classe = VAZIO) E (CotaInferiorNeg(n,Cijkl)) OU
                  (Classe = CHEIO) E (CotaSuperiorPos(n,Cijkl)))
  {
    Classe ← INTERCEPTANTE
  }

  RETORNE (Classe)
} /* ClassPrim */
```

O algoritmo recebe a primitiva através dos parâmetros n e C_{ijkl} que são, respectivamente, o grau e os coeficientes de Bézier da polinomial que a define em relação ao simplexo que está sendo classificado e o nível *Nível* deste simplexo na decomposição. É importante notar que o simplexo esta implícito nos coeficientes. A mesma polinomial tem diferentes representações, uma única para cada simplexo. O grau da polinomial serve para estabelecer, inequívocamente, o número de coeficientes através da relação:

$$N_{Coef} = \frac{(n+d)!}{n!\ d!}$$

onde d é a dimensão no espaço.

Inicialmente é feita a perturbação nos vértices que eventualmente anulam a função caractenstica do sólido e, em seguida, é feita a classificação do simplexo através da avaliação da polinomial nos seus vértices, Esta classificação é a definitiva para fins de aproximação linear, isto é, se o simplexo está no último nível de subdivisão. Ela pode ser alterada, porém, para aqueles simplexos relativamente grandes. Usando-se somente o valor da polinomial nos vértices, existem somente três classificações possíveis: VAZIO, CHEIO ou F-INTERCEPTANTE. Para um simplexo grande, as classificações CHEIO e VAZIO podem ser alteradas para INTERCEPTANTE caso as cotas inferior ou superior da polinomial no seu interior não assegurem a inexistência de interseções.

A avaliação da polinomial nos vértices, como mencionado, consiste unicamente do exame de alguns coeficientes especiais enquanto que a avaliação de não-interseções para os simplexos inicialmente classificados como CHEIOs ou VAZIOs é feita com o auxílio dos procedimentos *CotaInferiorNeg* e *CotaSuperiorPos*. Estes dois procedimentos são usados separadamente com *CotaInferiorNeg* analisando os coeficientes da polinomial e verificando se a cota inferior (definida pelo critério de Rivlin) é negativa, o que sugere a existência de interseção dentro de um simplexo classificado como VAZIO e que a classificação deve ser, consequentemente, alterada para INTERCEPTANTE. Analogamente, *CotaSuperiorPos* é utilizada para verificar possíveis alterações de CHEIO para INTERCEPTANTE.

4.5. OBTENÇÃO DOS COEFICIENTES DE BÉZIER

Tanto o critério de Bézier como o de Rivlin para classificação dos simplexos perante primitivas polinomiais precisam das representações destas na Base de Bemstein. Tradicionalmente, no entanto, as representações utilizadas estão na Base algébrica, o que levanta uma questão importante que é a conversão entre estas bases.

Arepresentação na Base de Bernstein demanda o conhecimento dos coeficientes de Bézier da polinomial com respeito a algum simplexo de referência e precisamos, portanto, saber obtê-los a partir dos coeficientes algébricos. No caso unidimensional esta conversão

é direta, com os coeficientes sendo dados pela relação abaixo:

$$P(x) = \sum_{i=0}^{n} a_i\, x^i = \sum_{j=0}^{n} b_j \binom{n}{j} x^j (1-x)^{n-j} \qquad b_j = \sum_{i=0}^{j} a_i \frac{\binom{j}{i}}{\binom{n}{i}}$$

Para dimensões maiores, todavia, a conversão é, em geral, não trivial e exige grande quantidade de cálculos.

A conversão Base Algébrica→Base de Bemstein é, na verdade, o grande "gargalo" da nossa idéia de classificação dos simplexos. Esta mudança de representação pode tomar o processo de classificação demasiadamente lento e numericamente instável e, em virtude disto, devemos atacar este problema de maneira eficaz.

A primeira idéia é evitar por completo a conversão, isto é, trabalhar desde o inicio com a representação da polinomial na Base de Bemstein. Agindo–se desta maneira, não somente a classificação dos simplexos seria facilitada como, também, a tarefa de modelagem, uma vez que a representação Bézier oferece melhores condições para se alterar características locais da superfície do objeto, quando comparada com a representação algébrica. Em outras palavras, os coeficientes de Bézier dão ao usuário um "sentimento" sobre a forma que ele está modelando enquanto que os coeficientes algébricos raramente o fazem. A utilização desta idéia, contudo, é muito complicada pois exige dos modeladores a capacidade de manipular as primitivas na forma Bézier, o que muito poucos dispõem, por enquanto.

Outra idéia, aparentemente mais factível, é a de trabalhar com a representação algébrica somente na fase de modelagem, onde ela é mais conveniente pois tanto os modeladores como o usuário podem trabalhar com as primitivas da maneira tradicional. A partir desta fase, todavia, é feita a conversão e, doravante, somente se utiliza a representação Bézier.

Esta última idéia é especialmente interessante no contexto das decomposições hierárquicas, já que nestes modelos de subdivisão adaptativa do espaço, a quantidade de simplexos inicialmente é bem pequena e cresce conforme o processo de subdivisão progride. Desta forma, podemos trabalhar com a representação algébrica das primitivas durante a modelagem e no momento de computar uma aproximação linear por partes para a casca do sólido, convertemos cada uma delas para a representação Bézier em relação a cada um dos simplexos iniciais da decomposição.

Estas conversões só precisam ser feitas um única vez por primitiva para cada simplexo inicial pois, de posse dos coeficientes de Bézier de uma polinomial em relação a determinado simplexo e conhecido o processo de subdvisão da geometria deste simplexo em seus filhos, podemos construir um algoritmo de subdivisão adequado para os coeficientes de modo a se saber a representação desta mesma polinomial em relação a cada um dos simplexos filhos.

Abordaremos brevemente, agora, a técnica de conversão entre as bases adotada neste trabalho e, em seguida, discutiremos o processo de subdivisão dos coeficientes de Bézier.

4.5.1. FUNÇÃO DE *BLOSSOM* DE POLINOMIAIS

A conversão Base Algébrica→Base de Bernstein é bastante complexa no caso n-dimensional (n>1) e exige demasiado esforço computacional. Neste trabalho utilizamos a alternativa proposta por [SEID89] que é bastante elegante, embora igualmente custosa.

Def. 4.1: Blossom

Uma função *n-afim* simétrica $f:(R^3)^n \rightarrow R$ é dita a Blossom de uma polinomial $P:R^3 \rightarrow R$ de grau n se e para todo $x \in R^3 \Rightarrow P(x) = f(x,...,x)$, onde f é n-afim por ser afim em cada uma das coordenadas e simétrica por retornar sempre o mesmo valor para qualquer permutação de seus argumentos.

Pode-se demonstrar que a função de Blossom para uma polinomial sempre existe e é única ([SEID89]), além de possuir algumas propriedades interessantes. Para nós, a propriedade mais importante é que, para uma polinomial $P(x,y,z)$ de grau n que tem $f:(R^3)^n \rightarrow R$ como Blossom, seus coeficientes de Bézier com respeito ao simplexo $o = \langle v_0, v_1, v_2, v_3 \rangle$ são dados pela relação:

$$C_{ijkl} = f(\underbrace{v_0 \ldots v_0}_{i \text{ vezes}}, \underbrace{v_1 \ldots v_1}_{j \text{ vezes}}, \underbrace{v_2 \ldots v_2}_{k \text{ uezes}}, \underbrace{v_3 \ldots v_3}_{1 \text{ vezes}}) \; ; \qquad i+j+k+l = n$$

Outra propriedade igualmente importante é que a Blossom de uma polinomial é, por construção, invariante sob transformações afins. Isto nos é muito útil para tratar instanciações de primitivas, já que ao invés de aplicarmos a transformação de instanciação aos semiespaços polinomiais, o que poderia ser bem complicado em algumas situações, aplicamos a transformação inversa aos vértices do simplexo de referência e calculamos os coeficientes de Bézier em relação a este simplexo deformado. Em outras palavras, ao invés de levarmos a primitiva para o espaço da decomposição, levamos os simplexos para o espaço da primitiva e lá calculamos sua representação Bézier.

A construção da função f por nós utilizada é feita algoritmicamente e será omitida para não fugirmos ao escopo deste trabalho. Maiores informações, contudo, podem ser obtidas em [BUEN92]. Além disto o método de construção especificamente não é muito relevante pelo fato de f ser única.

4.5.2. SUBDIVISÃO DOS COEFICIENTES DE BÉZIER

A conversão entre as bases, além de numericamente instável, é extremamente custosa do ponto de vista computacional e, portanto, só é feita seis vezes para cada

primitiva: uma vez para cada um dos simplexos da triangulação inicial da caixa envolvente, ou seja, para os simplexos iniciais da decomposição hierárquica.

A partir dos coeficientes de Bézier da polinomial que define a primitiva em um simplexo, felizmente podemos dispor de uma técnica mais eficiente para calcular os coeficientes da mesma polinomial em relação a cada um dos simplexos filhos do processo de subdivisão. Esta técnica consiste de uma extensão do método de Casteljau para avaliação de um ponto em uma curva de Bézier.

No caso unidimensional o algoritmo de Casteljau pode ser descrito da seguinte forma: dados os coeficientes de Bézier C_i ($i=0,\ldots,n$) de uma polinomial P de grau n, o valor desta polinomial no ponto t_p do espaço de parâmetros é dado por $B_0^n(t_p)$ onde:

$$\begin{aligned} B_i^r(t) &= (1-t)\, B_i^{r-1}(t) + t\, B_{i+1}^{r-1} \\ B_i^0(t) &= C_i \end{aligned} \qquad \begin{cases} r = 1,\ldots,n \\ i = 0,\ldots,n-r \end{cases}$$

Uma propriedade interessante do método de Casteljau é que a curva fica dividida em dois ramos, um relativo a variação do parâmetro no intervalo $[0,1-t_p]$ e outro associado ao intervalo $[1-t_p,1]$, com alguns dos termos calculados em cada etapa do algoritmo sendo os coeficientes de Bézier de cada um dos ramos. Isto significa que, ao avaliamos a polinomial em determinado ponto com o método de Casteljau, automaticamente calculamos os coeficientes de Bézier para os dois ramos da subdivisão da curva determinada por este ponto.

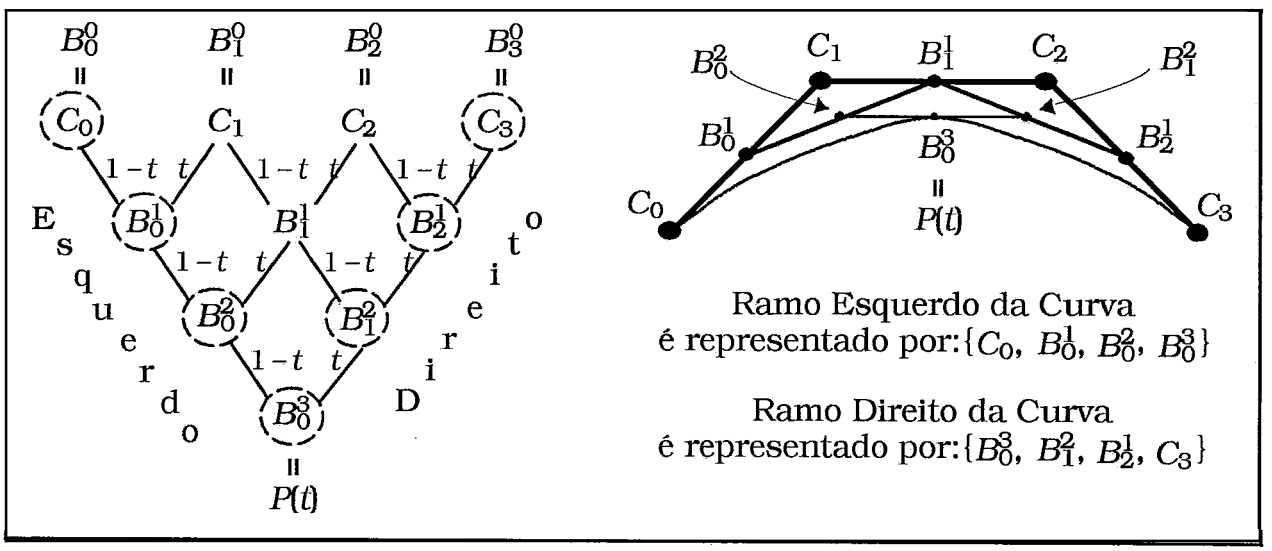


Fig. 4.10: avaliação de uma cúbica pelo método de Casteljau.

Na figura 4.10 temos um exemplo de uma cúbica, definida pelos coeficientes C_0, C_1, C_2 e C_3, sendo avaliada em $t = 0,5$ o que proporciona uma subdivisão da curva em dois ramos equivalentes.

É importante ressaltar que o parâmetro t define como a subdivisão da curva será feita e, consequentemente, os coeficientes de Bézier para cada um dos ramos filhos da

subdivisão. Isto significa que para se saber os coeficientes dos filhos a partir dos coeficientes do pai é imprescindível conhecer o processo de subdivisão: óbvio mas importante!

O método de Casteljau, do ponto de vista computacional, é tão custoso quanto avaliar a polinomial do modo clássico (fazendo todos os produtos e exponenciações), mas possui enorme vantagem em relação a este último: o processo é numericamente estável.

Estendendo o método para o nosso problema em que o processo de subdivisão é binário onde os simplexos podem ser de até três tipos diferentes, dependendo do nível da decomposição onde estejam (ver tabela 4.1), verificamos que o processo de subdivisão para os coeficientes de Bézier é igualmente simples. A idéia é avaliar a polinomial no meio da maior aresta do simplexo através do algoritmo de Casteljau, convenientemente adaptado para o espaço tridimensional. É exatamente neste ponto, lembramos, que o simplexo será cortado por um plano que passa pela aresta contraposta.

Dada uma polinomial de grau n expressa pelos coeficientes de Bézier P_{ijkl} em relação a um simplexo pai σ_p, a relação recursiva abaixo juntamente com a tabela 4.4 fornece os coeficientes de Bézier $F1_{ijkl}$ e $F2_{ijkl}$ que expressam esta mesma polinomial em relação a σ_{F1} e σ_{F2}, filhos de σ_p pelo processo de subdivisão por nós utilizado.

$$\begin{cases} b^0_{ijkl} = P_{ijkl} \\ b^r_{ijkl} = \frac{1}{2}\, b^{r-1}_{i+1jkl} + \frac{1}{2}\, b^{r-1}_{ijkl+1} \quad ; \quad i+j+k+l+r = n \end{cases}$$

$Tipo(\sigma_p) = 0$		$Tipo(\sigma_p) = 1$		$Tipo(\sigma_p) = 2$	
$F1_{ijkl}$	$F2_{ijkl}$	$F1_{ijkl}$	$F2_{ijkl}$	$F1_{ijkl}$	$F2_{ijkl}$
$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$
b^j_{ikl0}	b^j_{0ikl}	b^j_{ikl0}	b^j_{0ikl}	b^j_{ikl0}	b^j_{0ikl}

Tabela 4.4: subdivisão dos coeficientes de Bézier.

Nas expressões acima, a indexação dos vértices dos filhos é escolhida de forma que a aresta $\langle v_i, v_l \rangle$ deles é sempre a que deve ser subdividida no próximo nível. Naturalmente, os simplexos iniciais são indexados com o mesmo critério.

4.6. CONCLUSÃO

Apresentamos um esquema de decomposição hierárquica do espaço em simplexos, baseado em subdivisões binárias. Com o auxílio deste esquema podemos computar aproximações lineares por partes para a fronteira de sólidos CSG. A decomposição, além de simples, possui algumas propriedades interessantes como, por exemplo, a capacidade de não deformar muito os simplexos a medida que as subdivisões vão sendo feitas.

Para que a decomposição fosse hierárquica, isto é, para que regiões tão grandes quanto possível não fossem subdivididas desnecessariamente, estudamos critérios de classificação dos simplexos contra a fronteira do sólido. Alguns destes critérios se mostraram bem gerais mas pouco eficazes enquanto que outros, com potenciais de aplicação mais restritos, se mostraram bastante eficientes.

O critério por nós utilizado restringe o poder de expressão dos modeladores para o espaço das polinomiais, ou seja, somente podem ser usadas pelo modelador primitivas definidas por semiespaços polinomiais. Apesar disto, o critério se mostra bem eficiente tanto em termos de segurança das classificações quanto de tempo de processamento.

A idéia por trás deste critério é representar cada polinomial definidora de uma primitiva na Base de Bemstein e utilizar os coeficientes de Bézier, direta ou indiretamente, para estipular cotas para a variação da polinomial dentro de cada simplexo da decomposição. Estes coeficientes oferecem vantagem adicional que é o fato de alguns deles coincidirem com o valor da polinomial nos vértices do simplexo, evitando que estes valores tenham que ser calculados quando do cálculo de um pedaço da aproximação linear.

A utilização da Base de Bemstein exigiu a aplicação de técnicas para conversão dos coeficientes algébricos para os de Bézier, visto ser a base algébrica mais tradicionalmente usada para representação de polinomiais. Empregamos para esta finalidade a teoria de *Blossom* de polinomiais que se mostrou bastante geral propiciando, inclusive, uma maneira muito elegante de se tratar as instanciações de primitivas. As funções de Blossom, todavia, são extremamente dispendiosas em termos de processamento.

Também com o objetivo de otimizar o processo de classificação, utilizamos uma técnica de simplificação de árvores CSG que elimina as primitivas que não delimitam a fronteira do sólido em determinada região do espaço. Esta técnica, na maioria dos casos, reduz consideravelmente o número de primitivas da árvore durante a evolução do processo de subdivisão e, por conseguinte, simplifica substancialmente a função característica do sólido.

No próximo capítulo discutiremos algumas questões ligadas a implementação das idéias apresentadas.

"...you missed the starting gun..."

Capítulo 5

5. ASPECTOS GERAIS DA IMPLEMENTAÇÃO

5.1. INTRODUÇÃO

As idéias introduzidas neste trabalho foram implementadas numa série de programas de computador. O que foi expresso no texto certamente não reflete tudo aquilo que foi implementado por causa de limitações das mais variadas, que incluem o tamanho do texto e a capacidade de síntese do autor. Apresentaremos, neste capítulo, breve discussão sobre esta irnplementação e procuraremos mostrar, na medida do possível, onde e como foram adaptadas estas idéias.

Toda a implementação foi desenvolvida na linguagem C (padrão K&R) sob o sistema operacional SunOS 4.1 (UNIX/BSD) sendo executável em estações gráficas Sun 260,360, SPARC 1+,SPARC 2+. Ela consiste de vários módulos, alguns de apoio e outros que produzem resultados. Os módulos de apoio são utilizados em tempo de compilação, fornecendo recursos para os módulos principais através de várias funções. Estes últimos, por sua vez, comunicam-se por intermédio de arquivos e foram construídos de forma a poderem aproveitar um dos grandes recursos do sistema UNIX: "pipelines". Cada módulo principal pode ser visto como um filtro que tem determinada função e, conforme o resultado desejado pelo usuário, são feitas combinações diversas entre estes filtros. A figura 5.1 ilustra a relação entre todos os módulos.

Como podemos observar na figura 5.1, existem oito módulos sendo quatro principais e quatro de apoio. Os módulos principais são os pmgramas efetivamente executáveis CSG2CSPG, SUBSIMPL, VISUAL, PROPSIMPL enquanto que os de apoio são BEZIER, CSG, APROX e BLOSSOM. Faremos agora sucinta descrição destes módulos.

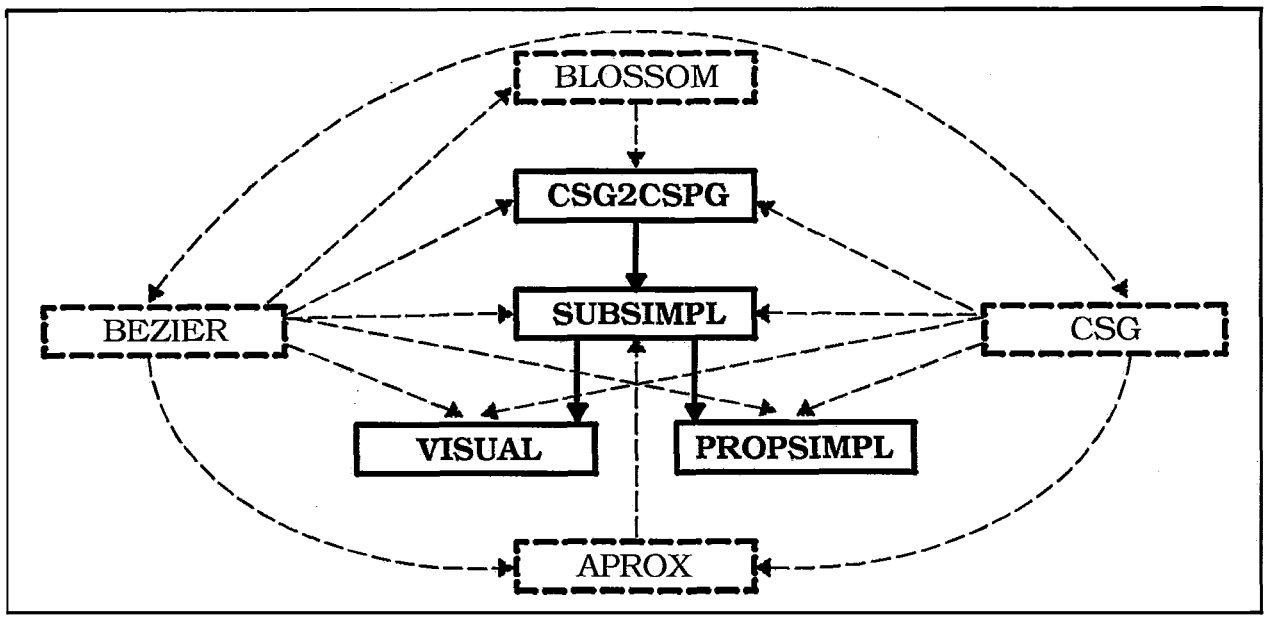


Fig. 5.1: relacionamento entre os módulos da irnplementação.

5.2. MÓDULOS DE APOIO

Os quatro módulos de apoio são, na verdade, bibliotecas C com diversas funções que visam fornecer subsídios aos módulos principais para a realização das tarefas pertinentes a cada um. Um módulo de apoio contém tipos, variáveis e funções específicas para alguma finalidade e os módulos principais utilizam-nas sempre que necessário.

A disposição das bibliotecas em módulos é muito útil pois separa convenientemente os tópicos, facilitando a construção, compreensão e depuração dos programas. É interessante, também, se considerarmos que outros módulos podem vir a ser construídos a partir destes (ou não), eventualmente por outras pessoas e com outras finalidades, integrando-se muito mais facilmente a filosofia do sistema. Falaremos um pouco, a seguir, sobre cada um destes módulos de apoio.

5.2.1. BEZIER

Este módulo é resposável pela manipulação das primitivas na forma Bézier e, também, pelo tratamento da geometria dos simplexos. No seu interior estão procedimentos para criação, classificação e subdivisão de primitivas Bézier além de um algoritmo de avaliação das primitivas pelo método de Casteljau. Estão igualmente presentes procedimentos para criação e subdivisão dos simplexos além de outros procedimentos que fornecem informações genéricas como, por exemplo, tipos dos simplexos, grau da polinomial que define determinada primitiva, vértices de um simplexo etc.

No que se refere a classificação das primitivas, foram implementados dois procedimentos, um utilizando o critério de Bézier e outro que usa o critério de Rivlin. A escolha entre os dois é feita durante a fase de compilação dos programas.

Os procedimentos de subdivisão, tanto de primitivas quanto de simplexos, foram implementados tal como discutido no capítulo anterior, com o método de Casteljau sendo o pivô das subdivisões binárias.

Foram construídos, também, procedimentos de eliminação das primitivas e simplexos criados, que servem para liberar a memória ocupada por aqueles que não mais são importantes para o processo de decomposição hierárquica. Maiores informações sobre este módulo podem ser obtidas no próprio programa fonte "bezier.c" (1608 linhas) que está devidamente comentado.

5.2.2. CSG

O módulo CSG cuida da criação, subdivisão e classificação das árvores CSG representativas dos sólidos. Ele tem estreitas ligações com o módulo BEZIER pelo fato das folhas das árvores serem primitivas Bézier.

O procedimento de classificação utiliza os conceitos de simplificação discutidos anteriormente, de modo que as árvores CSG em cada nível da decomposição hierárquica são as mais simples possíveis.

Foram desenvolvidos procedimentos para leitura e gravação das árvores em disco para facilitar a interação com os outros módulos. Algoritmos com finalidades mais genéricas como, por exemplo, a avaliação do número de primitivas de uma árvore também foram implementados assim como para eliminação de trechos ou de árvores completas. Maiores informações podem ser encontradas no arquivo "csg.c" (758 linhas).

5.2.3. APROX

O cálculo da aproximação linear, propriamente dito, é feito inteiramente por este módulo. Nele estão procedimentos capazes de calcular o pedaço de plano que melhor aproxima a casca do sólido presente em determinado simplexo.

Aqui são feitas as interpolações lineares nas arestas dos simplexos F–interceptantes, as avaliações das funções características (com auxílio do módulo CSG) e, eventualmente, perturbações pseudo–aleatórias na geometria de vértices que anulam estas funções.

Este módulo tem muitas ligações com os módulos BEZIER e CSG. Informações adicionais no arquivo "aprox.c" (838 linhas).

5.2.4. BLOSSOM

O único objetivo deste módulo é cuidar da mudança de representação das primitivas polinomiais da Base Algébrica para a Base de Bernstein. Nele estão os procedimentos necessários a construção da função de Blossom para polinomiais de, em princípio, qualquer grau no espaço. Arquivo: "blossom.c" (540 linhas).

5.3. MÓDULOS PRINCIPAIS

Diferentemente dos módulos de apoio, que são bibliotecas C, os quatro módulos principais são programas passíveis de serem executados. Cada um tem particular função no contexto da subdivisão simplicial adaptativa e pode ser usado individualmente ou em grupos, formando *pipelines* com funções específicas. Isto faz com que exista uma sequência em que os programas devem ser executados visto que a entrada de uns pode ser, diretamente, a saída de outros.

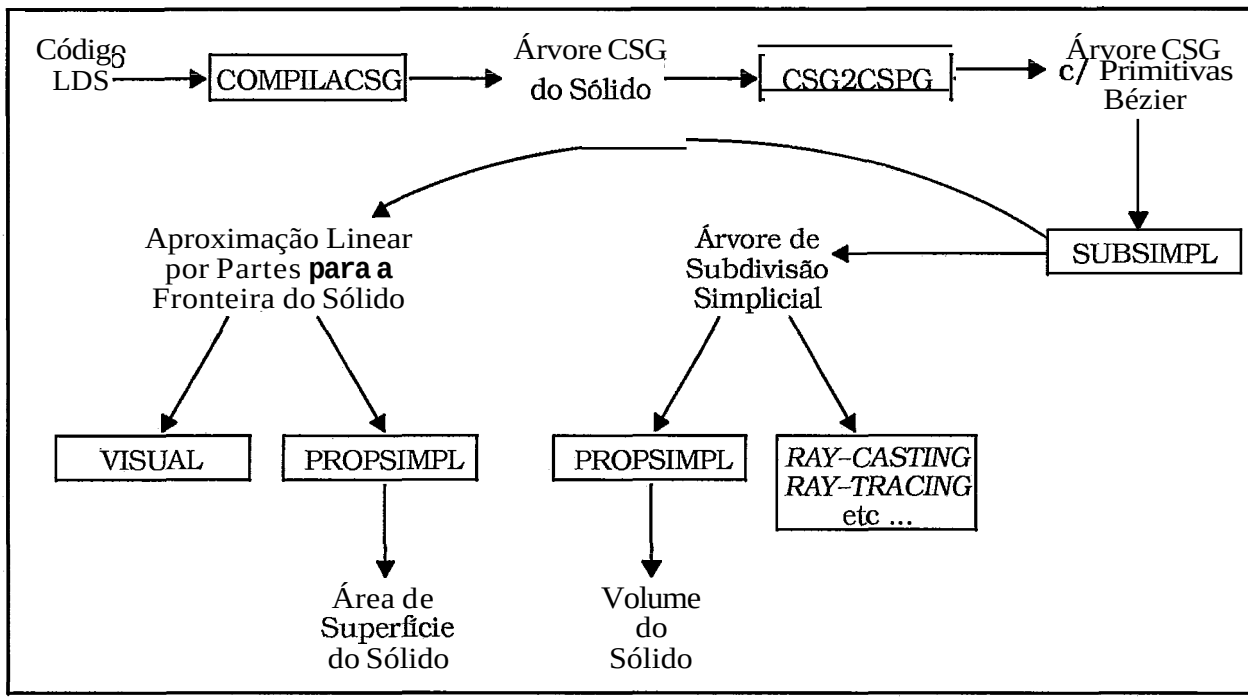


Fig. 5.2: organização funcional do sistema.

Podemos imaginar a integração entre os módulos principais tal como mostrado na figura 5.2. Inicialmente é necessária a construção do programa LDS, ou seja, a descrição do(s) sólido(s) através da linguagem especificada em [ESPE90]. Através do programa "CompilaCsg" ([ESPE90]), o código LDS é transformado em uma árvore CSG do sólido. Esta árvore é então processada pelo módulo CSG2CSPG que converte as primitivas para composições de semiespaços polinomiais, gerando nova árvore, na qual todas as primitivas estão na forma Bézier.

A partir desta versão modificada da árvore CSG, o módulo SUBSIMPL procede a Subdivisão Simplicial Adaptativa, computando uma aproximação linear por partes para a fronteira do sólido, que pode ser visualizada pelo módulo VISUAL ou utilizada para uma estimativa da área de superfície do sólido através do módulo PROPSIMPL. Outra alternativa possível é a geração da árvore de subdivisão simplicial por SUBSIMPL, que combinada com PROPSIMPL, pode fornecer estimativa para o volume do sólido. A árvore de subdivisão também pode ser usada para otimização de algoritmos diversos como *RAY-TRACING* etc.

Passamos, em seguida, a descrever brevemente cada um dos módulos principais. Maiores informações podem ser encontradas no Apêndice A.

5.3.1. CSG2CSPG

Este módulo é o primeiro da sequência e serve para converter a árvore CSG construída em LDS para o formato adequado a subdivisão simplicial. Ele tem três funções principais: substituir o operador diferença, decompor as primitivas LDS em semiespaços polinomiais e mudar a representação de cada primitiva para a Base de Bemstein.

A substituição do operador diferença se dá tal como mencionado no capítulo 3:

$$A \setminus^* B = A \cap^* \overline{B}$$

$\overline{B}$ = Complementar(B)

Φ^* = Operação Booleana Regularizada

e serve para otimizar a avaliação das funções características dos sólidos, já que ficam restando somente operações de união e interseção que podem ser substituídas diretamente por funções MIN-MAX.

A linguagem LDS oferece a possibilidade de se trabalhar com algumas primitivas simples (ESFERA, PLANO, TORO e POLINOMIAL) e outras compostas (BLOCO, CILINDRO, CONE e HELICÓIDE), o que simplifica bastante a tarefa de modelagem para o usuário. Para que se possa efetuar a subdivisão simplicial, no entanto, somente podem ser consideradas primitivas simples e, consequentemente, se faz necessária a decomposição de cada primitiva composta no conjunto de primitivas simples que a define.

A necessidade de decomposição de algumas primitivas LDS está diretamente relacionada com o critério de classificação de simplexos contra primitivas adotado. Os simplexos são testados contra cada primitiva na forma Bézier e, assim sendo, elas devem estar individualmente representadas.

A função mais importante deste módulo é, sem sombra de dúvidas, a conversão entre as representações das primitivas. É neste programa (e somente nele) que o módulo de apoio BLOSSOM é utilizado. Para que esta conversão possa ser feita, é preciso que se especifique os simplexos iniciais da decomposição hierárquica. O usuário os define na

chamada do programa, ao especificar a caixa envolvente ao sólido. Maiores informações no arquivo "csg2cspg.c" (1958 linhas).

5.3.2. SUBSIMPL

O módulo SUBSIMPL é o mais importante de todos. É ele quem realiza efetivamente a subdivisão simplicial adaptativa, relacionando-se com todos os outros módulos de apoio e principais. A partir da árvore CSG tratada por CSG2CSPG ele executa uma entre três funções passíveis de serem requisitadas pelo usuário. A primeira delas diz respeito ao cálculo puro e simples de estatísticas, onde a decomposição é feita somente para se avaliar características de cada execução. Neste modo não são produzidos resultados, apenas avaliações sobre o processo de subdivisão.

Em outro modo de execução, o programa faz a decomposição hierárquica exportando, em cada nível, os simplexos e suas respectivas classificações. Este modo gera, portanto, a árvore de subdivisão simplicial e pode ser explorado para otimizar algoritmos de *RAY-CASTING*, *RAY-TRACING*, conversores CSG $\rightarrow$ B-Rep ou mesmo diretamente, por exemplo, para computar uma estimativa para o volume do sólido através do módulo PROPSIMPL.

O terceiro modo de execução é o mais importante para a visualização e calcula a aproximação linear por partes para a fronteira do sólido, utilizando as idéias expostas neste trabalho.

O programa permite controle razoável da execução pelo usuário, que pode especificar a tolerância para a decomposição de duas maneiras diferentes: nível máximo de subdivisão ou tamanho do simplexo mínimo. São oferecidas, também, opções que visam acelerar o processo de decomposição em alguns casos particulares. O usuário pode, por exemplo, definir o número máximo de primitivas da árvore CSG permitido para um simplexo mínimo e especificar um subnível da decomposição onde deve ser computada a aproximação linear, o que permite um discernimento maior sobre quais primitivas influenciam determinadas regiões. Maiores informações no arquivo "subsimpl.c" (1008 linhas).

5.3.3. VISUAL

O objetivo deste módulo é única e exclusivamente a exibição da aproximação linear por partes computada por SUBSIMPL no modo apropriado. O usuário pode interativamente alterar os parâmetros do sistema projetivo utilizado para ver a aproximação de várias formas diferentes.

São oferecidos dois modos de exibição: WIREFRAME e SÓLIDO. O primeiro exibe apenas os pedaços da aproximação computados, aresta por aresta de cada polígono. O

segundo faz o preenchimento dos polígonos e utiliza a técnica de *z-buffer* para eliminar as partes ocultas ao observador. Em ambos os casos são utilizadas cores para diferenciar as primitivas do sólido, cada uma identificada por um índice de material definido em LDS. Maiores informações no arquivo "visual.c" (690 linhas).

5.3.4. PROPSIMPL

O módulo PROPSIMPL é utilizado para calcular estimativas para propriedades integrais do sólido. Somente foram implementadas funções para o cálculo de volume e área de superfície.

Este módulo trabalha, independentemente, com os dois formatos de saída do módulo SUBSIMPL. De acordo com a propriedade desejada, um dos modos de execução deve ser requisitado a SUBSIMPL.

Volume do Sólido ⇒ modo de geração da árvore de subdivisão
Área de superfície ⇒ modo de geração da aproximação linear por partes.

Maiores informações no arquivo "propsimpl.c" (426 linhas).

5.4. CONCLUSÕES

A implementação foi, sem dúvida alguma, muito facilitada pelo fato de se trabalhar em um ambiente UNIX. As ferramentas, os filtros e as bibliotecas disponíveis, além de outras facilidades inerentes a própria estrutura do sistema operacional, simplificaram bastante o trabalho, fazendo do desenvolvimento dos programas tarefa bastante agradável.

A utilização da linguagem C pmporcionou soluções fabulosas para diversas questões assim como algumas "dores de cabeça" na fase de depuração. Nem sempre a relação ideal entre eficiência de código e legibilidade do programa é fácil de ser alcançada e neste trabalho, por tratar-se de objeto de pesquisa, demos prioridade a segunda. Isto não significa, contudo, que o código final executável seja pouco eficiente. A tecnologia dos compiladores modernos faz com que as diferenças entre os programas legíveis e otimizados, embora existam, sejam pequenas. A opção pelo C deveu-se, também, a outras razões como a portabilidade dos programas e, principalmente, ao fato de compreendermos que sob o UNIX ela ainda é a melhor linguagem.

Os algoritmos implementados são típicos de estações gráficas, ou equipamentos semelhantes, por sua natureza fortemente recursiva, da necessidade de constantes alocações e desalocações de memória, além da intensa utilização de operações de entrada/saída para geração de uma invariavelmente grande quantidade de dados. A implementação em máquinas menos poderosas, embora perfeitamente viável, deve

ocasionar grande degradação de performance. Para máquinas paralelas, no entanto, as perspectivas são bastante promissoras já que a subdivisão espacial de regiões que não guardam muitas dependências entre si, parece se encaixar perfeitamente no grupo dos algoritmos mais facilmente paralelizáveis.

"...the time is gone the song is over, thought I'd something more to say."

Time - *Pink Floyd*

Capítulo 6

6. AVALIAÇÕES E CONSIDERAÇÕES FINAIS

6.1. DESCRIÇÃO DO TRABALHO

Os objetivos centrais deste trabalho são o de expandir o poder de expressão dos modeladores, em especial o daquele desenvolvido por [ESPE90] e construir aproximações para as fronteiras destes sólidos de maneira rápida e eficiente. Isto é pmporcionado através de uma adaptação da técnica de subdivisão espacial.

A maioria dos modeladores utiliza, como primitivas, superfícies definidas por um pequeno subconjunto das funções quadráticas e o enriquecimento do corpo das primitivas pode aumentar substancialmente a aplicabilidade do modelador. Caminhando neste sentido, deparamo-nos com as técnicas simpliciais que vem, já há bastante tempo, oferecendo soluções adequadas para uma vasta classe de problemas.

Os Métodos Simpliciais se encaixam entre as técnicas de subdivisão espacial e possibilitam soluções simples e de boa qualidade para muitos problemas, entre os quais, o da aproximação linear por partes de funções contínuas. A utilização destas técnicas na modelagem de sólidos é, portanto, imediata visto que os sólidos CSG podem ser representados globalmente por expressões do tipo MIN-MAX. A subdivisão regular do espaço, contudo, pode vir a inviabilizar esta tentativa em alguns casos.

Nossa proposta é a Subdivisão Simplicial Adaptativa, onde o espaço é totalmente decomposto em simplexos de maneira que as regiões mais próximas das soluções que buscamos são mais subdivididas do que aquelas mais afastadas.

Para que o processo de subdivisão seja adaptativo, existe a necessidade de se discernir entre os simplexos que estão próximos da fronteira do sólido e aqueles que dela estão afastados, ou seja, precisamos de critérios de classificação dos simplexos em cada nível da decomposição do espaço.

Os critérios de classificação devem ser suficientemente seguros para garantir que, em determinado nível da decomposição, nenhuma parte da fronteira, perceptível para este nível, seja perdida (ou deixe de ser aproximada). Além disso, a classificação não deve permitir muitos casos de dúvida, fazendo com que os simplexos que efetivamente não contêm partes da fronteira não sejam subdivididos. A classificação deve, também, ser feita da maneira mais rápida possível, pois, senão, o custo de classificar pode ser comparável ao da subdivisão indiscriminada dos simplexos.

Apresentamos três critérios de classificação: Lipschitz, Bézier e Rivlin. Todos eles procuram estabelecer limitantes para a variação das funções que definem as primitivas no interior dos simplexos. O Critério de Lipschitz analisa o gradiente da função e é bastante geral, podendo ser aplicado a uma larga classe de funções. O cálculo da constante de Lipschitz, no entanto, é muito oneroso e compromete substancialmente a eficiência do critério. Além disto, ele fornece limites muito relaxados, gerando muitos casos de dúvida na classificação.

Restringindo a classe das funções tratáveis para polinomiais, estudamos os critérios de Bézier e de Rivlin. Ambos utilizam os coeficientes de Bézier das polinomiais definidoras das primitivas para estabelecer limitantes para o seu comportamento no interior dos simplexos. Cada polinomial tem uma única representação Bézier em relação a cada simplexo da decomposição. O Critério de Bézier considera os valores extremos dos coeficientes de Bézier como cotas para a variação da polinomial, enquanto que o Critério de Rivlin, um pouco mais minucioso, utiliza alguns outros coeficientes para estabelecer cotas ainda melhores.

Apresentamos, a seguir, uma avaliação experimental da implementação das idéias introduzidas pelo trabalho. Na seção seguinte divagaremos sobre algumas otimizações que podem ser feitas no sistema. Na penúltima seção teremos algumas conclusões assim como apontaremos alguns projetos de pesquisa que podem ser encaminhados futuramente a partir da Subdivisão Simplicial Adaptativa. Finalmente, na última seção, daremos algumas impressões sobre todo o processo de desenvolvimento do trabalho.

6.2. AVALIAÇÕES TEÓRICA E EMPÍRICA DO MÉTODO PROPOSTO

Nesta seção discutiremos o desempenho do algoritmo de subdivisão simplicial adaptativa, baseados em informações obtidas de sua execução para alguns casos adequadamente escolhidos.

Primeiramente estudamos a conversão dos coeficientes algébricos para os de Bézier. Em seguida, procuramos analisar alguns aspectos teóricos do método proposto e estabelecer sua complexidade. Analisamos seu comportamento conforme a variação do nível de refinamento da subdivisão e, também, o acompanhamos enquanto variávamos o volume da caixa envolvente e, consequentemente, dos simplexos iniciais. Finalmente, fizemos breve estudo sobre a qualidade dos critérios de classificação dos simplexos.

A maioria dos testes utilizou os três sólidos mostrados na figura **6.1:** MANCHETE, ÁTOMO e CHUPETA. O sólido MANCHETE é constituído de **17** primitivas sendo **8** planos e 9 quádricas. O sólido ÁTOMO, por sua vez, é constituído de 4 primitivas, **1** quádrica e **3** *quárticas*. Finalmente, o sólido CHUPETA, com **13** primitivas, sendo **9** planos, **1** quádrica, **1** cúbica e **2** *quárticas*.

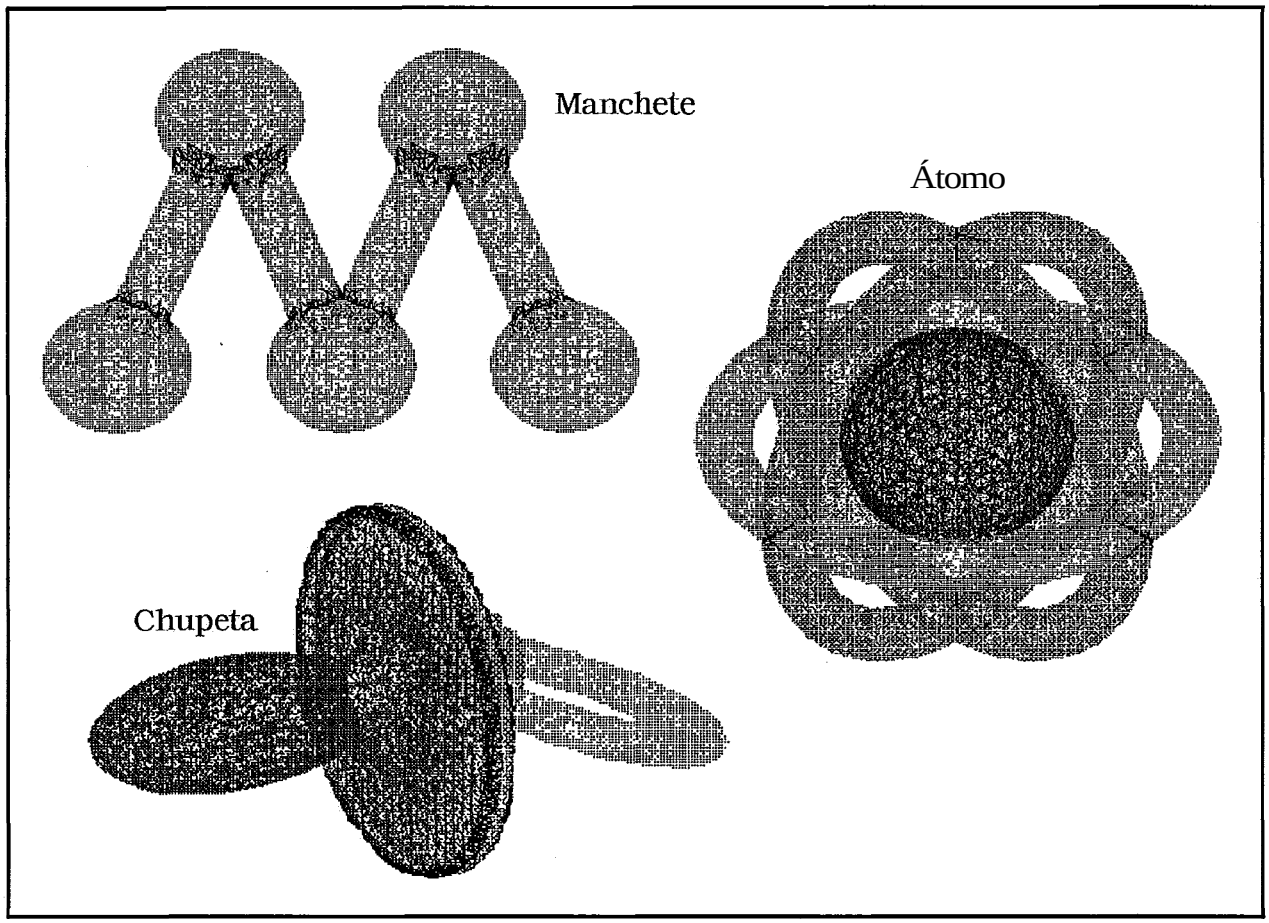


Figura 6.1: sólidos utilizados nos testes.

Antes de começarmos, devemos fazer uma ressalva no sentido de esclarecer um abuso de notação. Existem dois tipos de objetos que recebem a mesma denominação "primitiva": as primitivas da árvore CSG e as representações destas primitivas em relação a cada simplexo da decomposição. Como já vimos, uma mesma primitiva CSG têm diferentes representações na decomposição, cada qual referente a um particular simplexo. Por comodidade, durante o texto, nos referimos a cada uma destas representações distintas como sendo uma primitiva isolada quando, na verdade, isto não é formalmente correto. Isto explica o fato de, por exemplo na tabela 6.5, encontrarmos **19872** "primitivas" interceptantes para o sólido ÁTOMO, quando, na realidade, ele é constituído por apenas 4 primitivas CSG. Para que não surjam dúvidas, sempre que adotarmos o termo "primitiva" simplesmente, estaremos nos referindo a uma particular representação de qualquer primitiva da árvore CSG em relação a determinado simplexo. Por outro lado, quando quisermos nos referir aos elementos componentes do sólido CSG, empregaremos explicitamente o termo "primitiva CSG".

6.2.1. CONVERSÃO DE COEFICIENTES

Verificamos que o processo de conversão dos coeficientes algébricos para os de Bézier, por intermédio das funções de *Blossom*, é extremamente oneroso para polinomiais com graus superiores a quatro. A partir de grau cinco, inclusive, o tempo gasto com a conversão supera a ordem de grandeza do tempo gasto por todo o processo de subdivisão e cálculo de aproximação linear. A tabela 6.1 mostra os tempos gastos com a conversão dos coeficientes para polinomiais de vários graus.

Grau	NCoefs	Tempo (min)
1	4	00:00:016
2	10	00:00:033
3	20	00:00:266
4	35	00:03:433
5	56	00:52:423
6	84	14:18:265

Tabela 6.1: tempos de conversão Base Algébrica→Base de Bemstein.

Devemos ressaltar que o processo de subdivisão não guarda relação com a conversão dos coeficientes, podendo-se utilizar qualquer outra técnica para se obter os coeficientes de Bézier. Em alguns testes feitos obviamente observamos certa degradação da Subdivisão Simplicial, quando lidamos com primitivas definidas por polinomiais de graus elevados, mas esta degradação não é, em absoluto, comprometedora.

6.2.2. VARIAÇÃO DO NÍVEL DE REFINAMENTO DA SUBDIVISÃO

Procuramos analisar o comportamento do processo de subdivisão, a medida que o nível de refinamento da decomposição aumenta, no que se refere a quantidade de avaliações de primitivas necessárias a obtenção da aproximação linear. Estabelecemos comparações entre o algoritmo de Subdivisão Simplicial Adaptativa (SSA) e três outros métodos de cálculo de aproximação linear para fronteira de sólidos CSG, aos quais chamaremos de Método de Avaliação Completo (MAC), Método de Avaliação Ideal (MAI) e Método de Avaliação Ótimo (MAO).

No Método de Avaliação Completo, assumimos que o algoritmo não possui informação alguma a cerca do posicionamento do sólido no espaço, nem tampouco sobre a natureza de sua topologia (número de componentes conexas etc). Neste caso, todo o espaço deve ser examinado e o número de simplexos a se avaliar é dado por 6x21, supondo-se a decomposição da caixa envolvente feita por uma SSA quando a árvore de subdivisão gerada com profundidade n é completa.

Como o algoritmo não dispõe de informações sobre a natureza do sólido, não podem ser feitas simplificações na árvore CSG e, consequentemente, a função característica

representa sempre o sólido completo. Desta forma, o total de avaliações feito pelo algoritmo MAC é:

$$Aval_{MAC} = 6 \times 2^n \times N^o \text{ Primitivas CSG}$$

No caso do Método de Avaliação Ideal, supomos que o algoritmo conhece o número de componentes conexas do sólido, assim como um simplexo inicial para cada uma destas componentes. Também neste caso, supomos que simplificações na árvore CSG não são feitas, uma vez que os simplexos não chegam a ser classificados. Eles são percorridos em sequência, utilizando-se o conceito de subfaces-ponte (mencionado na seção 2.6), sendo processados diretamente para cálculo de pedaços da aproximação linear. O total de avaliações é, por conseguinte, dado pela relação:

$$Aval_{--} = N^o \text{ Simplexos Mínimos Interceptantes} \times N^o \text{ Primitivas CSG}$$

O Método de Avaliação Ótimo é aquele que não somente conhece os simplexos iniciais para cada componente conexa, como também sabe exatamente quais primitivas avaliar em cada simplexo mínimo que contêm parte da fronteira do sólido. Neste caso, o total de avaliações é dado diretamente pelo número de primitivas interceptantes.

$$Aval_{MAO} = N^o \text{ Primitivas Interceptantes}$$

O número de avaliações do MAO fornece uma primeira análise da complexidade da aproximação linear do modelo. Este método fornece o número mínimo de avaliações de primitivas para se obter uma aproximação do sólido e este número é inerente a natureza da aproximação, completamente independente do método utilizado para se chegar até ela.

A complexidade do MAO pode ser estabelecida se considerarmos cada primitiva P_i de grau N_i da árvore CSG S, responsável por uma área A_i da fronteira do sólido. O custo de avaliação de cada P_i é da ordem de $(N_i)^4$, visto ser este o custo de avaliação de polinomiais pelo método de Casteljau. Considerando l a maior aresta dos simplexos mínimos, a área do pedaço de aproximação linear da casca do sólido no interior de cada simplexo mínimo interceptante é da ordem de l^2. Por conseguinte, o número de simplexos mínimos necessários para cobrir A_i é da ordem de A_i/l^2.

Podemos, então, estimar a ordem do custo total de avaliação de cada primitiva:

$$O(\text{Avaliar } P_i) = (N_i)^4 \cdot A_i / l^2$$

O custo de avaliação da árvore completa do sólido é estimado por:

$$O(\text{Avaliar S}) = \frac{1}{l^2} \sum_{i=1}^{NPrim} N_i^4 \, A_i \qquad \text{(Equação 6.1)}$$

O MAO fornece N_s, o menor número de simplexos que devem ser avaliados para que se possa computar a aproximação linear. Não por acaso, N_s é igual ao número de simplexos mínimos obtidos na SSA. Como a árvore de subdivisão simplicial é binária, a SSA gera no máximo $2\,N_s$ simplexos durante todo o processo de subdivisão e possui, portanto, complexidade ótima.

Devemos salientar, no entanto, que a complexidade da SSA e do MAO são assintóticas, ou seja, somente para níveis altos de refinamento da subdivisão as ordens destes algoritmos podem ser obtidas pela equação **6.1.**

Nível	Max. Simp.	Tot. Simp.	Simp. Inter.	Max. Av. Pr.	Tot. Av. Pr.	Prim. Inter.
21	12.582.912	573.226	**173.884**	163.577.856	**1.023.544**	**279.222**
20	6.291.456	356.042	108.974	81.788.928	656.504	183.902
19	3.145.728	222.026	67.318	40.894.464	423.868	116.628
18	1.572.864	141.134	40.658	20.447.232	280.928	71.682
17	786.432	88.594	26.438	10.223.616	185.440	47.912
16	393.216	55.022	16.910	5.111.808	123.456	31.116
15	196.608	34.298	11.220	2.555.904	84.352	20.410
14	98.304	21.326	6.876	1.277.952	56.832	14.150
13	49.152	13.378	4.140	638.976	38.604	9.280
12	24.576	8.598	2.390	319.488	26.916	5.844
11	12.288	5.458	1.570	159.744	18.624	4.146
10	6.144	3.430	1.014	79.872	12.952	2.836
9	3.072	2.186	622	39.936	9.156	1.898
8	1.536	1.410	388	19.968	6.488	1.334
7	768	878	266	9.984	4.536	976
6	384	542	168	4.992	3.132	702
5	192	326	108	2.496	2.088	522
4	96	186	70	1.248	1.344	372
3	48	90	48	624	804	270
2	24	42	24	312	450	177
1	12	18	12	156	222	114
0	6	6	6	78	78	72

Tabela **6.2:** comportamento da subdivisão para diferentes níveis de refinamento.

O MAI, por sua vez, apresenta dependência linear com a natureza do modelo. Conforme o número de primitivas da árvore CSG cresce, maior o número de avaliações de primitivas que devem ser feitas. Para poucas primitivas e/ou níveis baixos de refinamento

da subdivisão, seu comportamento é semelhante ao do MAO. Em casos mais realistas, contudo, ele apresenta degradação muito elevada.

Utilizando o sólido CHUPETA, uma caixa envolvente adequada e o critério de classificação de simplexos de Rivlin, fizemos uma série de subdivisões simpliciais adaptativas com níveis de refinamento diferentes. Os resultados estão na tabela 6.2.

Na tabela 6.2, a primeira coluna indica os níveis de refinamento da subdivisão. A segunda coluna contêm o número máximo de simplexos passíveis de serem gerados pelo processo de subdivisão em determinado nível, isto é, o número de simplexos de último nível de refinamento em uma árvore de subdivisão completa. A terceira coluna informa o número de simplexos efetivamente gerados pela subdivisão e a quarta coluna fornece o número de simplexos mínimos interceptantes. A quinta coluna contêm o número de avaliações de primitivas que seriam feitas no pior caso: árvore de subdivisão completa e árvore CSG não simplificada por simplexo mínimo. Na sexta coluna encontramos o número de primitivas efetivamenta avaliadas e, finalmente, na sétima e última coluna temos o número de primitivas interceptantes, ou seja, as primitivas da árvore CSG, com representação relativa aos simplexos mínimos, que são responsáveis diretas pela formação da fronteira do sólido.

Podemos verificar que o algoritmo de subdivisão simplicial, neste exemplo, comportou-se de acordo com a complexidade estimada. A cada três níveis de subdivisão, a maior aresta de simplexo mínimo cai a metade em comprimento. Como o modelo CSG foi mantido inalterado, assim como a caixa envolvente utilizada, o número de avaliações só poderia aumentar, no máximo, de quatro vezes. Observando os dados da tabela 6.2 notamos que isto não chega a acontecer nenhuma vez, exceto abaixo do nível cinco de subdivisão, quando o processo aproximativo ainda é muito impreciso.

A tabela 6.3 fornece informações sobre o total de avaliações feitas para cada algoritmo, para alguns níveis de subdivisão selecionados.

Nível	MAC	MAI	MAO	SSA	$\frac{SSA}{MAI}$	$\frac{SSA}{MAO}$	$\frac{MAI}{MAO}$
21	163.577.856	2.260.492	279.222	1.023.544	0,45	3,67	8,09
18	20.447.232	528.554	71.682	280.928	0,53	3,92	7,37
15	2.555.904	145.860	20.410	84.352	0,58	4,13	7,15
12	319.488	31.070	5.844	26.916	0,87	4,61	5,32
9	39.936	8.086	1.898	9.156	1,13	4,82	4,26
6	4.992	2.184	702	3.132	1,43	4,46	3,11
3	624	624	270	804	1,29	2,98	2,31

Tabela **6.3:** comparação entre os métodos de avaliação para o sólido CHUPETA.

Os Métodos de Avaliação Completo e Ideal fornecem, respectivamente, limitantes superiores e inferiores para o Método de Continuação, descrito na seção 2.6. No seu pior

caso, o Método de Continuação necessita verificar todos os simplexos em que o espaço foi decomposto, avaliando a função caractenstica completa do sólido. Nos melhores casos, ele possui informações suficientes sobre o posicionamento do sólido que lhe permitem avaliar a função característica somente nos simplexos importantes para fins de aproximação linear.

Observando os dados da tabela 6.3, constatamos que o MAC é extremamente perdulário em relação aos demais e, consequentemente, que é imprescindível otimizá-lo, já que ele pode se tomar inviável até mesmo para níveis de decomposição do espaço relativamente pequenos.

Verificamos, também, que o algoritmo proposto apresenta um desempenho bastante satisfatório em relação ao MAI. Conforme o nível de refinamento da decomposicão aumenta, menor fica a relação entre o número de avaliações feitas pela SSA e aquelas feitas no MAI.

Com relação ao MAO, a SSA também se mostra eficiente. O número de avaliações foi, na média, quatro vezes superior ao número ótimo, mas esta relação tende a diminuir a medida que a profundidade da subdivisão aumenta. Acreditamos que a relação deva tender assintoticamente para dois.

Para que pudéssemos avaliar melhor o comportamento dos quatro métodos, fixamos um nível de decomposição de uma também fixada região do espaço. Construímos, então, aproximações lineares para as fronteiras de vários sólidos distintos, cada qual com mais primitivas que o outro.

# Prim.	MAC	MAI	MAO	SSA	SSA / MAI	SSA / MAO	MAI / MAO
1	1.572.864	528	528	2.484	4,70	4,70	1,00
2	3.145.728	2.240	1.144	5.080	2,27	4,44	1,96
4	6.291.456	9.008	2.388	10.144	1,13	4,25	3,77
8	12.582.912	34.992	4.812	19.920	0,57	4,14	7,27
16	25.165.824	139.200	9.644	39.252	0,28	4,07	14,43
32	50.331.648	553.728	19.328	77.660	0,14	4,02	28,65
64	100.663.296	2.202.624	38.736	154.260	0,07	3,98	56,86
128*	201.326.592	5.449.728	226090	1.120.336	0,21	4,96	24,10

Tabela 6.4: total de avaliações de cada método para diferentes modelos CSG.

Todos os sólidos eram constituídos unicamente de esferas espalhadas pelo espaço, de forma tal, que não se interceptavam. Exceção é o caso de 128 primitivas, no qual elas ficam amontoadas, umas por dentro das outras, formando um grande anel toroidal. O

nível de subdivisão foi fixado em 18. A tabela 6.4 mostra o número de avaliações feitas em cada caso.

Como qualquer avaliação feita acima do número estipulado pelo MAO constitui desperdício, o que a tabela 6.4 nos mostra é que o número de avaliações perdulárias do MAI cresce linearmente com a complexidade do modelo, enquanto que na SSA este número se mantém razoavelmente constante, tendendo a decrescer a medida que a complexidade do modelo aumenta.

No último caso (128 primitivas), a alta frequência de interseções entre as primitivas, alterou consideravelmente o comportamento dos métodos de avaliação. O MAO, por exemplo, sofreu um brusco aumento do número de avaliações, o que pode ser explicado pelo elevado número de simplexos mínimos híbridos, isto é, contendo mais de uma primitiva na árvore CSG a eles restrita. Neste caso, especificamente, a média de primitivas por simplexo mínimo ficou em tomo de cinco.

A aparente melhora do MAI neste caso, também pode ser explicada pelo mesmo motivo. Como o número de primitivas por simplexo mínimo cresceu, o custo de avaliação da árvore CSG completa, para cada simplexo mínimo, não foi muito maior do que aquele relativo à avaliação da árvore simplificada, posto que o algoritmo de simplificação (localização do sólido) não teve condições de eliminar muitas primitivas. O MAI tende para o MAO à medida que o número de primitivas da árvore CSG simplificada por simplexo mínimo se aproxima do número de primitivas da árvore original.

Por outro lado, a SSA sofre uma queda de eficiência pois, da mesma forma que no MAO, o menor número de primitivas eliminadas em simplificações acarreta um maior número de avaliações. Esta queda de eficiência, no entanto, não foi muito grande, com o número de avaliações perdulárias permanecendo nos mesmos patamares anteriores, ou seja, entre três a quatro vezes superior ao número ótimo.

Ao que nos foi dado perceber, o MAI é, quando comparado a SSA, mais eficiente para níveis de subdivisão baixos, ou seja, em rehamentos pobres. Nestes refinamentos as árvores CSG nos simplexos mínimos, mesmo simplificadas, ainda são relativamente grandes. O MAI é melhor quando o número de primitivas da árvore CSG restrita aos simplexos mínimos é, em média, próximo do total de primitivas da árvore CSG original. Em contrapartida, ele apresenta um crescimento do número de avaliações perdulárias que acompanha linearmente a natureza do modelo CSG, ao contrário da SSA que mantém uma proporção razoavelmente constante entre as avaliações inúteis e importantes para fins de aproximação linear.

Nos casos mais frequentes, a SSA se mostra mais eficiente do que o MAI e, consequentemente, do que o Método de Continuação. O número de simplexos visitados é maior, em geral, mas a quantidade de avaliações das polinomiais definidoras das primitivas é bem menor. O gerenciamento da geometria dos simplexos da subdivisão e dos coeficientes de Bézier é, contudo, uma característica que depõe contra o método proposto, uma vez que o MAI não necessita armazenar coeficientes e utiliza a estrutura implícita a decomposição regular do espaço, com regras combinatórias simples, para transitar por entre os simplexos vizinhos.

Este gasto adicional com o gerenciamento de memória na SSA se mostrou, na prática, significativo. Cerca de 20% do tempo é gasto com alocação e desalocação de memória. No entanto, os tempos totais gastos para computar aproximações para as cascas de diversos sólidos (por exemplo os da tabela 6.7 e 6.8) se mostraram bastante satisfatórios.

6.2.3. VARIAÇÃO DA CAIXA ENVOLVENTE

Fizemos testes, independentemente, com os três sólidos mostrados na figura 6.1 e utilizamos, para classificação dos simplexos, o critério de Bézier. Cada teste consistia do cálculo de uma aproximação poliédrica para a casca do sólido a partir de duas caixas envolventes diferentes. Ambas as caixas eram suficientemente grandes para envolverem completamente o sólido, com a diferença, no entanto, que uma possuia volume oito vezes maior que o da outra, isto é, suas arestas tinham o dobro do tamanho em cada dimensão.

O nível de refinamento da subdivisão foi controlado de modo que, nos dois casos, o tamanho dos simplexos mínimos da subdivisão permaneceu o mesmo. As tabelas 6.5 e 6.6 mostram os resultados obtidos.

Sólido	Class. Prim.	Prim. Inter.	Tot. Simp.	Simp. Inter.	CPU (s)
Manchete	16320	1424	4104	724	04:199
Átomo	67728	19872	61620	18864	31:032
Chupeta	82200	20256	34612	11062	29:182

Tabela 6.5: subdivisão simplicial para a caixa envolvente de volume R

Sólido	Class. Prim.	Prim. Inter.	Tot. Simp.	Simp. Inter.	CPU (s)
Manchete	19302	1424	4392	724	04:933
Átomo	68880	19872	61908	18864	31:282
Chupeta	84170	20452	36232	11636	27:198

Tabela 6.6: subdivisão simplicial para a caixa envolvente de volume 8V.

Na primeira coluna das duas tabelas encontramos o nome do sólido. Na segunda coluna, o total de classificações de primitivas, ou seja, quantas vezes a função de classificação de primitivas contra simplexos foi acionada. A terceira coluna fornece o número de primitivas interceptantes no último nível de subdivisão, que informa quantas das primitivas classificadas efetivamente influenciavam a formação da casca do sólido. A quarta e quinta colunas fornecem informações sobre a quantidade de simplexos

avaliados. A quarta coluna dá o total dos simplexos da subdivisão simplicial enquanto que a quinta informa o número de simplexos mínimos que contém parte da fronteira do sólido.

A sexta e última coluna informa o tempo gasto por todo o processo de subdivisão simplicial, incluindo o cálculo de aproximações lineares nos simplexos mínimos capazes de produzí–las. Utilizamos o tempo de CPU para evitar que flutuações, causadas pela rede ou pelo gerenciamento de processos concorrentes, pudessem influenciar os resultados. Como na maioria dos casos, no entanto, os testes foram feitos quase que em monoprocessamento, os valores presentes nas tabelas refletem, com boa fidelidade, os tempos decorridos nos processamentos.

Avaliações baseadas na comparação entre sólidos distintos são inviáveis, uma vez que diversos fatores que influenciam diretamente o processo de subdivisão (como número de primitivas, grau das polinomiais definidoras das primitivas, caixas envolventes etc) são completamente diferentes em cada caso. Podemos, entretanto, obter algumas conclusões ao compararmos o comportamento da **SSA** para um mesmo sólido nas duas caixas envolventes diferentes, isto é, ao contrastamos as tabelas 6.5 e 6.6.

Como esperado, o tamanho da árvore de subdivisão varia muito pouco conforme o volume da caixa envolvente cresce. A quantidade de simplexos mínimos se mantém aproximadamente constante, independentemente do tamanho da caixa envolvente utilizada. Isto mostra que o algoritmo proposto tem sua complexidade proporcional ao modelo que se está examinando, mais especificamente a área de superfície do sólido, sendo quase que independente da natureza da caixa envolvente utilizada (desde que ela contenha todo o sólido).

As discrepâncias por ventura encontradas na quantidade de simplexos mínimos obtida, podem ser explicadas pelo fato da alteração do volume da caixa envolvente ter ocasionado razoável modificação no processo de subdivisão, alterando a localização de alguns simplexos interceptantes no espaço. Cabe ressaltar que o processo de subdivisão é sempre alterado quando a caixa envolvente é modificada, já que os simplexos iniciais originam–se de sua decomposição, porém, o tamanho da árvore de subdivisão tende a permanecer constante. Em alguns casos, contudo, até mesmo o tamanho da árvore sofre uma pequena modificação em razão da movimentação dos simplexos iniciais no espaço.

Como a árvore de subdivisão tem o seu tamanho pouco influenciado pelo volume da caixa envolvente, os tempos do algoritmo de subdivisão também ficam próximos nos dois casos. A quantidade de primitivas classificadas, assim como o número de simplexos gerados, tendem, no entanto, a crescer conforme o volume da caixa envolvente aumenta. Isto é evidentemente explicado pelo fato de existirem mais regiões do espaço que devem ser subdivididas, classificadas etc. Podemos observar nas tabelas 6.5 e 6.6, contudo, que a sobrecarga de trabalho não é muito grande, o que indica que o processo de subdivisão elimina eficientemente as regiões disjuntas da fronteira do sólido.

6.2.4. QUALIDADE DOS CRITÉRIOS DE CLASSIFICAÇÃO

Outra questão relevante que procuramos estudar foi a qualidade dos critérios implementados de classificação de simplexos. Fizemos a subdivisão simplicial com cada um dos três sólidos da figura 6.1 usando alternadamente os critérios de Bézier e de Rivlin. Os resultados estão nas tabelas 6.7 e 6.8.

Sólido	Class. Prim.	Tot. Simp.	Sub. Indev.	CPU (s)
Manchete	19302	4392	514	04:933
Átomo	68880	61908	576	31:282
Chupeta	84170	36232	4572	27:198

Tabela 6.7: subdivisão simplicial com classificações pelo Critério de Bézier

Sólido	Class. Prim.	Tot. Simp.	Sub. Indev.	CPU (s)
Manchete	18064	4016	326	04:716
Átomo	68304	61620	432	29:715
Chupeta	83814	36036	4474	29:032

Tabela 6.8: subdivisão simplicial com classificações pelo Critério de Rivlin.

A primeira coluna informa o nome do sólido, a segunda coluna contém o número de classificações de primitivas feitas, a terceira fornece o total de simplexos gerados pelo processo de subdivisão e a quinta e última coluna contém os tempos de CPU gastos com cada sólido. A quarta coluna apresenta o número de simplexos subdivididos indevidamente, ou seja, aqueles que foram subdivididos mas que nenhum de seus descendentes continha parte da fronteira do sólido. Estes simplexos indicam os casos que o critério de classificação não conseguiu aferir devidamente a não existência de interseção e que, em última análise, falhou.

Podemos observar, contrastando as tabelas 6.7 e 6.8, que o Critério de Rivlin é realmente mais preciso que o Critério de Bézier. O número menor de simplexos gerados e de classificações de primitivas indica que, através dele, o processo de subdivisão elimina mais primitivas e, consequentemente, mais regiões do espaço, durante a decomposição hierárquica.

A pequena margem de diferença entre os números dos dois casos, todavia, evidencia que o ganho obtido com a utilização do Critério de Rivlin não é muito expressivo. Mesmo nos casos onde seria pior (ao menos em princípio), o Critério de Bézier se mostrou bastante satisfatório.

A eficiência do Critério de Rivlin, pmporcionada por sua maior precisão nas classificações, não é muito superior aquela obtida quando do uso do Critério de Bézier, haja visto que o número de simplexos subdivididos indevidamente não é muito diferente nos dois casos. Isto pode ser explicado pela rápida convergência dos poliedros de Bézier para as supefícies por eles definidas. O simples exame dos coeficientes de Bézier, na maioria dos casos, já é suficiente para se ter um esboço do comportamento da supefície em determinada região do espaço.

Além da segurança nas classificaçõesque possibilita, existe uma virtude adicional do Critério de Bézier: velocidade. O teste de sinais dos coeficientes é extremamente rápido, chegando a ofuscar a eficiência do Critério de Rivlin. Nas tabelas 6.7 e 6.8, por exemplo, podemos perceber que o cálculo de uma aproximação linear por partes para a casca do sólido CHUPETA, foi mais rápido quando a subdivisão simplicial usou o Critério de Bézier, a despeito do fato de menos simplexos terem sido gerados e menos primitivas classificadas quando da utilização do Critério de Rivlin. A explicação para isto é que o Critério de Rivlin é bem mais oneroso em cada classificação individual.

A utilização do Critério de Bézier, portanto, se mostrou bastante adequada a maioria das situações. Para níveis de refinamento razoável da subdivisão, os coeficientes de Bézier das polinomiais definidoras das primitivas CSG fornecem subsídios suficientes para classificações seguras dos simplexos, o que não ocorre em baixos níveis de subdivisão. Isto nos dá uma sugestão interessante, que é a de começar as classificações com o Critério de Rivlin e, a medida que os simplexos vão sendo subdivididos, trocar o critério de classificação pelo de Bézier. Estudo mais pormenorizado deve ser feito, contudo, com o intuito de se determinar o momento ideal de proceder a troca de critérios.

6.3. OTIMIZAÇÕES POSSÍVEIS

No decorrer da implementação, uma série de novas idéias tomou corpo e nem todas puderam ser devidamente exploradas. Nesta seção, apresentaremos algumas sugestões de otimização do sistema em várias fases e, também, apontaremos alguns tópicos que devem ter um tratamento mais adequado do que aquele que nos foi possível dar

6.3.1. FUNÇÕES DE *BLOSSOM*

Um dos tópicos que merece especial atenção é a conversão dos coeficientes algébricos para os de Bézier. O cálculo da Blossompara polinomiais com graus inferiores a cinco é satisfatório, mas para graus superiores ele se mostrou extremamente oneroso, quando comparado com o processo de subdivisão propriamente dito, além de numericamente instável. Cabe lembrar que o processo de subdivisão é numericamente estável, pelo fato de usar o método de Casteljau, mas o pmcesso de conversão é, em geral, instável. A forma de conversão deve, pois, ser muito bem estudada.

Uma maneira de contornar o problema e, como já dito, evitar a representação na Base Algébrica. O tratamento das primitivas diretamente na forma Bézier não somente elimina o problema de conversão como, também, pode propiciar um controle localizado de aspectos da superfície do sólido durante a modelagem.

6.3.2. SIMPLIFICAÇÕES DE PRIMITIVAS

Desde o início do trabalho nos apercebemos de uma idéia bastante interessante que, pela elevada complexidade de sua implementação, preferimos deixá–la de lado até que suas diversas peculiaridades tivessem sido melhor estudadas. Falamos da simplificação das primitivas a medida que o processo de subdivisão progride.

Cada primitiva é representada por uma polinomial de determinado grau que a descreve em todo o espaço. Como a Subdivisão Simplicial lida com porções do espaço individualmente, podemos imaginar as primitivas como sendo representadas não por uma única polinomial, mas sim por um conjunto delas. Para o processo de subdivisão, a primitiva é inicialmente representada por uma polinomial, mas a medida que os simplexos vão sendo subdivididos, esta polinomial é substituída por outra de grau menor que a representa adequadamente no interior de cada simplexo filho. Este processo de substituição vai, obviamente, sendo repetido para os simplexos filhos dos filhos etc. No final teremos um grupo de polinomiais definindo a primitiva, uma polinomial específica para cada simplexo mínimo.

A diminuição do grau das polinomiais representaria não somente uma enorme economia de memória, visto serem menos coeficientes de Bézier a se armazenar, como também resultaria em um tremendo aumento de velocidade computacional. Além disto, nas regiões onde as primitivas puderem ser substituídas por semiespaços planos, o processo de subdivisão poderia ser precocemente interrompido.

A simplificação de primitivas, apesar de ter enorme potencial a ser explorado, possui algumas dificuldades intrínsecas que devem ser contornadas para que sua implementação seja viável. Inicialmente existe a necessidade de estudos criteriosos para se avaliar em que circunstâncias uma polinornial pode ser substituída por outra de grau menor em determinada região do espaço.

Outra questão igualmente importante é que a substituição não deve pmporcionar descontinuidades, ou seja, quando um simplexo é subdividido, a polinomial que define a primitiva no seu interior talvez possa ser substituída por duas outras polinomiais de grau menor (uma para cada simplexo filho), mas deve–se garantir que estas novas polinomiais tenham um comportamento adequado na junta entre os dois simplexos.

6.3.3. APROXIMAÇÃO LINEAR BASEADA EM TODOS OS COEFICIENTES DE BÉZIER

Outra otimização que pode vir a ser feita diz respeito a fase de cálculo de aproximação linear no interior dos simplexos mínimos. Durante esta fase, considera-se apenas o valor da função característica do sólido nos vértices dos simplexos, isto é, utiliza-se apenas alguns dos coeficientes de Bézier das primitivas eventualmente presentes na árvore CSG localizada do sólido. Pelas propriedades da Base de Bernstein, os coeficientes de Bézier fornecem uma aproximação para a função característica do sólido e podemos pensar em melhorar a qualidade da aproximação linear utilizando todos os demais coeficientes.

Os coeficientes de Bézier subdividem a geometria do simplexo de referência em uma série de regiões e podemos aproveitar esta subdivisão adicional, que não onera de nenhuma forma o algoritmo (visto que os coeficientes já foram calculados), para computar os pedaços da aproximação linear. A qualidade da aproximação sofre, portanto, uma melhora, pelo fato dos coeficientes definirem um nível de refinamento maior do que aquele definido pela simples utilização dos simplexos mínimos. A figura 6.2 fornece um exemplo no plano, onde a primitiva é definida por uma polinomial cúbica.

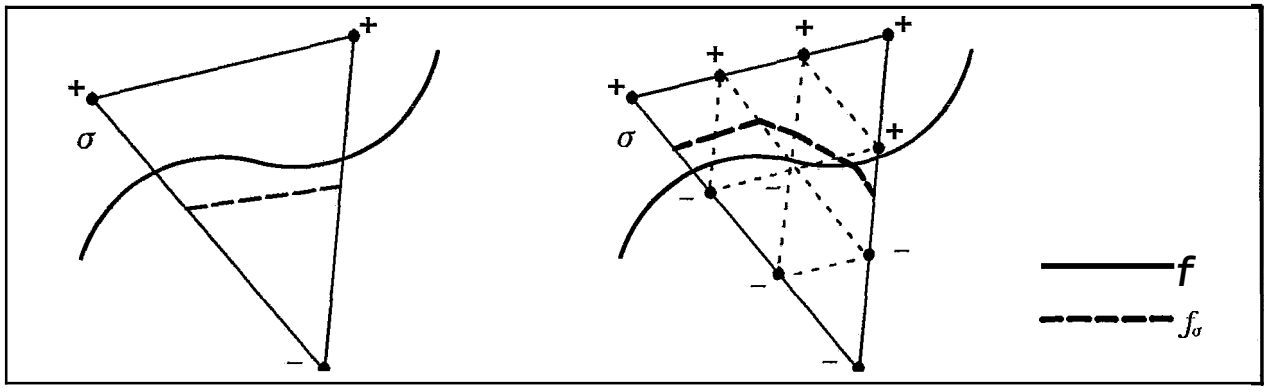


Figura 6.2: aproximações lineares com alguns e com todos os coefs. de Bezier.

A garantia de continuidade da aproximação nas juntas dos simplexos vem do fato de que os coeficientes são obtidos pelo método de Casteljau, que utiliza o mesmo critério de avaliação nos dois simplexos vizinhos. Nem mesmo erros numéricos atrapalham pois, para efeito do cálculo dos coeficientes da junta, o Casteljau é feito sempre com os mesmos coeficientes independentemente de qual simplexo está sendo avaliado.

A contrapartida vem do fato de alguns destes coeficientes poderem ser nulos. Estabelecendo-se uma analogia com as subdivisões comuns, eles deveriam ser perturbados. Esta perturbação, contudo, deve ser melhor estudada, visto ser ela bem diferente da perturbação da geometria dos simplexos, anteriormente mencionada. Um investimento neste sentido, porém, deve ser compensador pois a idéia é bastante promissora.

6.3.4. ARMAZENAMENTO DOS CAMINHOS ATE OS SIMPLEXOS MÍNIMOS

Pensamos, também, na possibilidade de não se computar a subdivisão da geometria dos simplexos a cada passo, mas apenas guardar os caminhos até os simplexos mínimos. Na fase de cálculo da aproximação linear, cada caminho é, então, interpretado convenientemente. Em termos de cálculos, o ganho com esta tática não deve ser muito grande já que, de uma maneira ou de outra, a geometria dos simplexos mínimos é computada. A vantagem aparece, no entanto, quando consideramos que não será mais necessário alocar e gerenciar memória para armazenar a geometria dos simplexos intermediários que, em última análise, são inúteis.

6.3.5. CRITÉRIOS DE CLASSIFICAÇÃO MAIS GERAIS

Obviamente, também pode-se pensar na extensão dos critérios de classificação para que se permitam primitivas definidas por funções mais gerais do que polinomiais, fornecendo ainda mais opções ao usuário na complexa fase de modelagem.

6.4. CONCLUSÕES E PERSPECTIVAS DE PROJETOS FUTUROS

Avaliações conclusivas sobre um trabalho deste tipo são sempre muito subjetivas. Em cada pequena parte foi gasta grande quantidade de energia distribuída nas fases de criação, adaptação e implementação daquilo que foi imaginado, além da depuração do sistema computacional desenvolvido. Este fato passará despercebido para a maioria das pessoas.

Em todos os momentos nos vimos envolvidos com o compromisso entre qualidade e tempo e as decisões que tomamos, em cada situação específica, sempre poderão ser questionadas. Podemos, tão somente, passar a nossa impressão sobre o todo e oferecer, apenas, mais uma opinião.

A primeira impressão que nos fica é a de que, em cada parte do trabalho, ficou faltando muita coisa, não importando o esforço empreendido e a grande quantidade de objetivos alcançados. Talvez isto seja próprio da pesquisa científica que para cada problema solucionado formula muitos outros a partir da própria solução. Talvez não...

É interessante observar, de qualquer modo, que o objetivo principal visado foi atingido, isto é, os modeladores podem dispor de primitivas mais gerais e consegue-se calcular aproximações para as fronteiras dos sólidos de modo bem rápido. Isto, contudo, ainda é insuficiente para facilitar a tarefa de modelagem. A simples possibilidade de se utilizar primitivas definidas por polinomiais de grau qualquer, não muito facilita a vida do usuário, posto que questões cruciais permanecem: como obter a equação da polinomial associada a determinada primitiva idealizada? Como localizar convenientemente no espaço determinada primitiva?

Faz-se necessário, portanto, um investimento maior na interface com o usuário. É preciso que os sistemas sejam mais inteligentes e, preferencialmente, interativos. Sob este aspecto, a representação das primitivas através da Base de Bemstein pode vir a ser característica bastante interessante.

No que se refere as perspectivas de projetos futuros, muita coisa pode ser feita. O que de imediato se apresenta é a possibilidade de se utilizar a árvore de subdivisão simplicial para otimizar algoritmos de *RAY-TRACING* e assemelhados. Na verdade, a árvore de subdivisão simplicial pode ser usada para otimizar qualquer algoritmo que explore subdivisão espacial, com a vantagem de que se utiliza de critérios de classificação mais abrangentes e precisos do que aqueles usados por decomposições espaciais mais comuns. Os testes de interseção entre os cubos de uma octree e uma superfície polinomial, por exemplo, nem sempre são resolvidos de forma simples enquanto que entre simplexos e polinomiais são.

Pode-se otimizar algoritmos de *RAY-TRACING* tal como é feito com o auxílio de octrees ([COMB91]): o espaço é decomposto em regiões que contêm material e outras que não e, em cada região, a árvore CSG do sólido é constituída apenas das primitivas que são efetivamente influentes, ou seja, a árvore CSG é simplificada.

Proposta ainda mais interessante é utilizar a árvore de subdivisão simplicial para construir um B-Rep do sólido, isto é, a estrutura de dados responsável pelo arranjo relativo dos vários pedaços de aproximação linear computados independentemente. Teríamos um conversor CSG->B-Rep por intermédio de subdivisão simplicial. A estratégia sugerida em [HOTT92] pode ser perfeitamente adaptada para este fim.

O processo de subdivisão binária entre os simplexos, por ser ordenado, pode fornecer facilmente a relação de adjacência entre os vários simplexos mínimos capazes de produzir aproximação linear, isto é, os simplexos F-interceptantes. Como a função afim aproximadora é continua, os simplexos F-interceptantes vizinhos no espaço guardam, cada um, pedaços da aproximação linear que, na face de contato, são simétricos. Nesta face eles possuem imagens especulares um do outro. Esta característica em muito facilita a tarefa de colagem das faces B-Rep.

Para que a geração do B-Rep seja viável, no entanto, é necessário que algumas otimizações do processo de subdivisão sejam melhor estudadas. O critério de parada da subdivisão para os simplexos que contêm somente uma primitiva linear; por exemplo, pode gerar simplexos vizinhos de tamanhos diferentes. Isto para efeito de simples cálculo de aproximação linear para visualização não traz nenhum problema mas pode comprometer bastante a colagem das faces B-Rep.

6.5. IMPRESSÕES

Durante o desenvolvimento deste trabalho, foi possível observar fatores que concorrem para a elaboração e realização de um projeto de pesquisa e, paralelamente, para a formação das pessoas envolvidas que, normalmente, não recebem a merecida relevância. Falaremos brevemente sobre alguns deles.

Primeiramente foi importante constatar que por mais que saibamos determinado assunto temos ainda muito a aprender sobre ele e, neste sentido, a melhor maneira é trabalharmos com ele. O conhecimento de C, Unix, Estruturas de Dados etc, obtido durante os vários meses de implementação, suplanta, em muito, aquele obtido durante toda a graduação.

Um aspecto pouco mencionado, mas que também é de suma importância, é o ambiente de trabalho e o bom relacionamento com pessoas envolvidas em pesquisas afins. O ambiente de trabalho deve dispor dos equipamentos necessários ao desenvolvimento da pesquisa assim como deve oferecer harmonia entre as pessoas que o frequentam.

A troca de informações, opiniões e sugestões com outros pesquisadores é igualmente fundamental para o enriquecimento da pesquisa. O intercâmbio de preprints, artigos e dicas com pessoas de outras instituições, além da participação em seminários, congressos e encontros do gênero, facilitam em muito a realização do trabalho e promovem, em mais larga escala, a difusão do conhecimento.

Igualmente importante é adquirir o saudável vício de vasculhar bibliotecas a procura de livros, revistas etc, que possam, de alguma forma, vir a ser úteis. Às vezes, onde menos se espera, encontram-se artigos capazes de ajudar na pesquisa ou, até mesmo, redirecioná-la. É impressionante como sempre existem pessoas em lugares diferentes pensando sobre o mesmo tema. Talvez seja o inconsciente coletivo de Jung...

Compreender, o mais rapidamente possível, que a pesquisa acadêmica no Brasil, por enquanto, é uma espécie de sacerdócio, é essencial para eliminar de antemão uma série de frustrações. O gosto pela atividade acadêmica deve ser suficientemente grande para evitar que obstáculos de ordem financeira, burocrática, política e religiosa, que certamente aparecerão, tornem-se intransponíveis.

Durante o transcurso da pesquisa, não podemos nos deixar abater pelas invariavelmente baixas remunerações, nem permitir que as constantes greves (de professores, funcionários, alunos, motoristas de ônibus, trocadores, camelôs etc), assim como *upgrades* do sistema operacional, constantes falta de luz, consertos de ar condicionado, trocas de placas gráficas e, principalmente, a absurda quantidade de feriados do calendário nacional, prejudiquem demasiadamente o andamento dos trabalhos. Definitivamente os nossos anos não têm 365 dias e a duração de cada dia está longe de 24 horas.

Devemos evitar, ao máximo, reflexões do tipo "será que vale a pena ?" pois, na maioria das vezes, a resposta é não.

Referências Bibliográficas

1. [ALDE83] B. Aldefeld. "On Automatic Recognition of 3D structures from 2D Representations". Computer Aided Design – Vol. 15 – NO 2 – 1983.

2. [ALLG80] E. Allgower e K. Georg. "Simplicial and Continuation Methods for Approximating Fixed Points and Solutions to Systems of Equations". SIAM REVIEW – Vol. 22 – Nº 1 – 1980.

3. [ALLG85] E. Allgower e P. Schmidt. "An Algorithm for Piecewise–Linear Approximation of an Implicitly Befined Manifold. SIAM J. Numer. Anal. – Vol. 22 – Nº 2 – 1985.

4. [ALLG87] E. Allgower e S. Gnutzmann. "An Algorithm for Piecewise–Linear Approximation of Implicitly Defined Two–Dimensional Surfaces". SIAM J. Numer. Anal. – Vol. 24 – Nº 2 – 1987.

5. [AKEL88] K. Akeley e T. Jermoluk. "High–Performance Polygon Rendering". ACM Comp. Graph. – Vol. 22 – NO 4 – 1988.

6. [BECK90] B. Beckert. "Computers Bring Indy Teams up to Speed". Computer Aided Engineering – Vol. 10 – Nº 5 – 1990.

7. [BOIS89] J. Boissonnat e M. Teilland. "On the Randomized Construction of the Delaunay Tree". Relatório de pesquisa NO 1140 – Robotique, Image et Vision – INRIA – 1989.

8. [BOOT74] B. Boots. "Delaunay Triangles: An Alternative Approach to Point Pattern Analysis". Proc. Assoc. Am. Geogr. – Vol. 6 – 1974. (Como citado em [WATS81])

9. [BUEN92] L. Bueno. "An Explicit Form for the Blossom of a Polynomial Function". Preprint – 1992.

10. [BURR86] P. Burrough. "Principles of Geographical Information Systems for Land Resources Assessment". Clarendon Press Oxford – 1986.

11. [CAST90] A. Castelo, S. Freitas e G. Tavares. "PL Approximation to Manifolds and Its Application to Implicit ODEs". Lect. Appl. Math. (Amer. Math. Soc.) – Vol. 26 – 1990.

12. [CAST91] A. Castelo, S. Freitas e G, Tavares. "Simplicial Approximation of Implicitly-Defined Manifolds". Preprint - 1991.

13. [CLAR82] J. Clark. "The Geometry Engine: a VLSI Geometry System for Graphics". ACM Comp. Graph. - Vol. 16 - Nº 3 - 1982.

14. [COMB91] J. Comba. "Acompanhamento de Raios Otimizado para Sólidos CSG. Dissertação de Mestrado (COPPE/UFRJ) - 1991.

15. [COXE34] H. Coxeter. "Discrete Groups Generated by Reflections". Ann. Math. - Vol. 35 - 1934.

16. [DOBK90] D. Dobkin, S. Levy, W. Thurston e A. Wilks. "Contour Tracing by Picewise Linear Approximations". ACM Trans. Graph. - Vol, 9 - NO 4 - 1990.

17. [ELLI78] W. Elliot. "Interactive Graphical CAD in Mechanical Engineering Design". Computer Aided Design - Vol. 10 - NO 2 - 1978.

18. [ESPE90] C. Esperança. "Técnicas para Visualização Direta de Sólidos CSG. Dissertação de Mestrado (COPPE/UFRJ) - 1990.

19. [FARI86] G. Farin. "Triangular Bernstein-Bézier Patches". CAGD - Vol. 3 - NO 2 - 1986.

20. [FARI88] G. Farin. "Curves and Surfaces for Computer Aided Geometric Design". Academic Press - 1988.

21. [FONS92] A. Fonseca. "LCGRENDERER: Um Visualizador de Modelos Baseado em PHIGS". Projeto Final do Curso de Informática - UFRJ - 1992.

22. [FREI91] S. Freitas. "Aproximações Simpliciais de Variedades Implícitas e Equações Algébrico-Diferenciais". Dissertação de Doutorado - PUC/RJ - 1991.

23. [GUIB85] L. Guibas e J. Stolfi. "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams". ACM Trans. Graph. - Vol. 4 - Nº 2 - 1985.

24. [GUTI91] E. Gutiérrez. "Análise de Modelos Tridimensionais de Terreno". Preprint - 1991.

25. [HANR89] P. Hanrahan. "A Survey of Ray-Surface Intersection Algorithms" In: "An Introduction to Ray-Tracing", pp 79-119 - Academic Press - 1989.

26. [HARA82] R. Haralick e D. Queeney. "Understanding Engineering Drawings". Computer Graphics and Image Processing - Vol. 20 - Nº 3 - 1982.

27. [HINT77] E. Hinton e D. Owen. "Finite Element Programming". Academic Press - 1977.

28. [HIRS76] M. Hirsch. "Differential Topology". Springer-Verlag - 1976.

29. [HOTT92] V. Hottum. "Uma Estratégia para Construção de Representações BRep de Sólidos CSG". Dissertação de Mestrado (COPPE/UFRJ) - 1992.

30. [JACK80] C. Jackins e S. Tanimoto. "Octtrees and their Use in Representing Three-dimensional Objects". Computer Graphics and Image Processing – Vol. 14 – NO 3 – 1980.

31. [KAJI82] J. Kajiya. "Ray Tracing Parametric Patches". ACM Comp. Graph. – Vol. 16 – Nº 3 – 1982.

32. [KALR89] D. Kalra e A. Barr. "Guaranteed Ray Intersections with Implicit Surfaces". ACM Comp. Graph. – Vol. 23 – NO 3 – 1989.

33. [KUHN68] H. Kuhn. "Simplicial Approximations of Fixed Points". Proc. Nat. Acad. Sci. USA – Vol. 61 – 1968.

34. [LIGH82] R. Light e D. Gossard. "Modifications of Geometric Models through Variational Geometry". Computer Aided Design – Vol. 14 – Nº 4 – 1982.

35. [MANT82] M. Mantyla e R. Sulonen. "GWB: A Solid Modeler with Euler Operators". IEEE CG&A – Vol. 2 – Nº 7 – 1982.

36. [MANT84] M. Mantyla. "A Note on the Modeling Space of Euler Operators". Computer Vision, Graphics and Image Processing – NO 26 – 1984.

37. [MANT90] M. Mantyla. "An Introduction to Solid Modeling". Computer Science Press – 1990.

38. [MEAG82] D. Meagher. "Geometric Modeling Using Octree Encoding". Computer Graphics and Image Processing – Vol. 19 – NO 2 – 1982.

39. [MILL90] R. Mills. "Solid Modeling Software". Computer Aided Engineering – Vol. 10 – Nº 5 – 1990.

40. [MIRA89] J. Miranda e G. Tavares. "Métodos Simpliciais em Computação Gráfica". 17º Colóquio Brasileiro de Matemática – 1989.

41. [MOOR90] D. Moore e J. Warren. "Multidimensional Adaptive Mesh Generation". Computer Science Technical Report – Rice University – 1990.

42. [MORT85] M. Mortenson. "Geometric Modeling". John Wiley & Sons – 1985.

43. [NEVE88] J. Neves. "Sistema Interativo para Mapeamento Baseado na Triangulação dos Pontos do Plano". Dissertação de Mestrado (COPPE/UFRJ) – 1988.

44. [PERS91] R. Persiano, M. Salim e L. Bueno. "Boundary Evaluation of CSG Solids by Simplicial Subdivision". Proceedings of COMPUGRAPHICS'91 – First International Conference on Computational Graphics and Visualization Techniques – 1991.

45. [REQU80] A. Requicha. "Representation for Rigid Solids: Theory, Methods and Systems". ACM Computing Surveys – Vol. 12 – NO 4 – 1980.

46. [REQU82] A. Requicha e H. Voelcker. "Solid Modeling: A Historical Summary and Conteporary Assessment". IEEE CG&A – Vol. 2 – NO 2 – 1982.

47. [REQU83] A. Requicha e H. Voelcker. "Solid Modeling: Current Status and Research Directions". IEEE CG&A – Vol. 3 – NO 7 – 1983.

48. [RIVL70] T. Rivlin. "Bounds on Polynomial". Journal of Research of the National Bureau of Standards / B. Math. Sc. – Vol. 74B – N01 – 1970.

49. [ROCK72] R. Rockafellar. "Convex Analysis". Princeton University Press. 1972.

50. [ROSS91] J. Rossignac e A. Requicha. "Cons-tnictive Non-Regularized Geometry". Computer Aided Design – Vol. 23 – N0 1 – 1991.

51. [SAIG79] R. Saigal. "On Piecewise Linear Approximations to Smooth Mappings". Mathematics of Operations Research – Vol. 4 – Nº 2 – 1979.

52. [SALI91] M. Salim, L. Bueno e R. Persiano. "Aproximação Linear por Partes de Variedades Implicitamente Definidas usando Poda do Espaço". Anais do SIBGRAPI'91 – IV Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens – 1991.

53. [SAME90] H. Samet. "The Design and Analysis of Spatial Data Structures". Addison–Wesley – 1990.

54. [SCAR67] H. Scarf. "The Approximation of Fixed Points of a Continuous Mapping". SIAM J. Appl. Math. – Vol. 15 – Nº 5 – 1967.

55. [SEID89] H. Seidel. "A General Subdivision Theorem for Bézier Triangles" In: "Mathematical Methods in Computer Aided Geometric Design", pp 573–581 – Academic Press – 1989.

56. [SERR82] J. Serra. "Image Analysis and Mathematical Morphology". Academic Press – 1982.

57. [SHIR81] S. Shirari. "Representation of Three–dimensional Digital Images". ACM Computing Surveys – Vol. 13 – Nº 4 – 1981.

58. [SHNE87] B. Shneiderman. "Designing the User Interface: Strategies for Effective Human–Computer Interaction". Addison–Wesley – 1987.

59. [SING91] G. Singh e M. Green. "Automating the Lexical and Syntatic Design of Graphical User Interfaces: The UofA* UIMS'. ACM Trans. Graph. – Vol. 10 – Nº 3 – 1991.

60. [SIMO89] M. Simões e H. Moura. "Modelo Digital de Terreno como Base Cartográfica e suas Aplicações em Projetos de Engenharia". Anais do XIV Congresso Nacional de Cartografia – 1989.

61. [SUFF90] K. Suffern. "Quadtree Algorithms for Contouring Functions of Two Variables". The Computer Journal – Vol. 33 – Nº 5 – 1990.

62. [TAKA85] T. Takala. "User Interface Management System with Geometric Modeling Capability: A CAD System's Framework". IEEE CG&A – Vol. 5 – Nº 4 – 1985.

63. [TAVA90] G. Tavares e J. Miranda. "Concordance Operations for Implicitly–Defined Manifolds". Preprint – 1990,

64. [TILO84] R. Tilove. "A Null–Object Detection Algorithm for Constructive Solid Geometry". Communications of the ACM – Vol. 27 – N0 7 – 1984.

65. [TODD76] M. Todd. "On Triangulations for Computing Fixed Points". Mathematical Programming – Nº 10 – North Holland Publishing Company – 1976.

66. [TORB87] J. Torborg. "A Parallel Processor Architecture for Graphics Arithmetic Operations". ACM Comp. Graph. – Vol. 21 – Nº 4 – 1987.

67. [UDUP83] J. Udupa. "Display of 3D Information in Discrete 3D Scenes Produced by Computerized Tomography". Proceedings of IEEE – Vol. 71 – NO 3 – 1983,

68. [WATS81] D. Watson. "Computing the n–Dimensional Delaunay Tesselation with Application to Voronoi Polytopes". Computer Journal – Vol. 24 – Nº 2 – 1981.

69. [WIDM90] R. Widmann. "Efficient Triangulation of 3–Dimensional Domains". Preprint – 1990.

Apêndice A

MANUAL DOS PROGRAMAS DE SUBDIVISÃO SIMPLICIAL ADAPTATIVA

A.1. INTRODUÇÃO

Para manipulação do objeto CSG através de subdivisão simplicial adaptativa do espaço, foi desenvolvido um conjunto de programas em linguagem C (padrão K&R) executáveis em estações gráficas SUN 260, 360, Sparc 1+ e Sparc 2+ sob o sistema operacional SunOS 4.1 (UNIX/BSD). Estes programas são:

- CSG2CSPG

conversão da árvore CSG gerada pelo compilador da linguagem LDS para árvore CSG composta de semiespaços polinomiais, ideal para a subdivisão simplicial.

- SUBSIMPL

subdivisão simplicial adaptativa do espaço e eventual cálculo da aproximação linear por partes para a fronteira do objeto CSG.

- VISUAL

visualização rudimentar da aproximação linear da fronteira do objeto.

- VIS2LCG

conversão da entrada do programa VISUAL para o programa LCGRENDERER.

- PROPSIMPL

cálculo de propriedades integrais do objeto.

Os programas podem ser usados individualmente ou em conjunto, formando *pipelines* específicas para alguma função. Existe, portanto, uma sequência na qual eles

devem ser executados para que os filtros possam funcionar devidamente. A figura A.1 mostra a organização funcional do sistema desenvolvido.

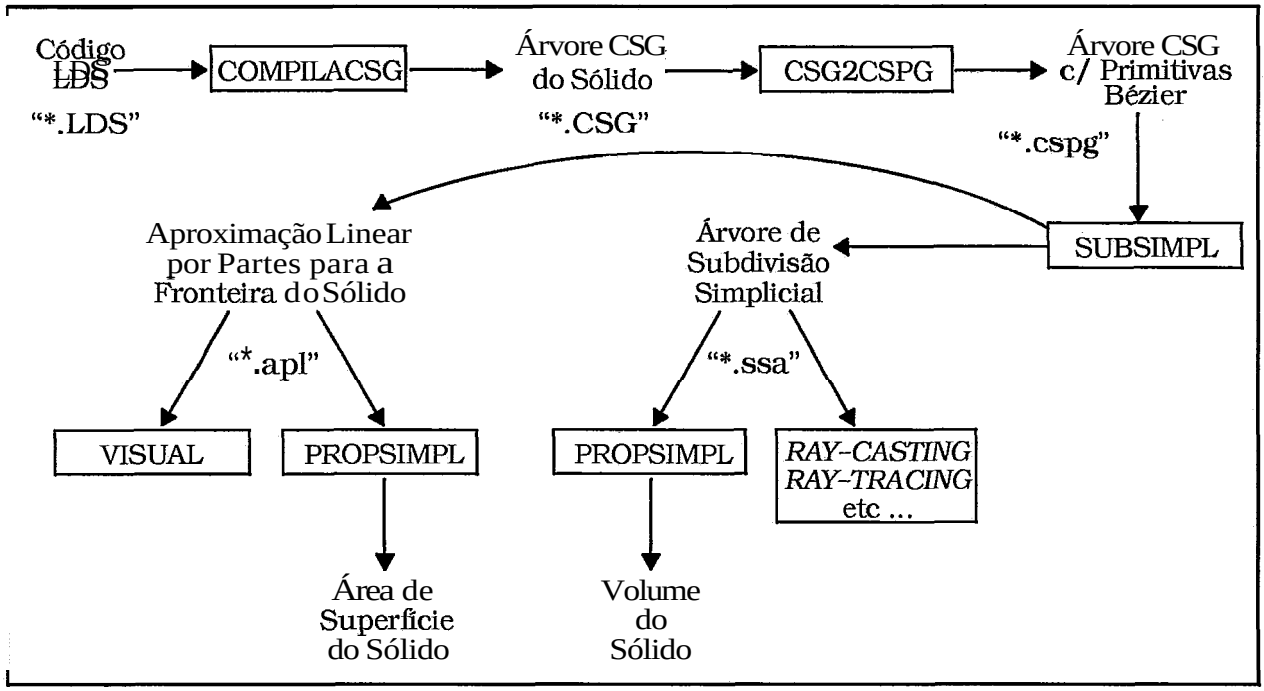


Fig. A.1: organização funcional do sistema.

O usuário inicialmente constrói o programa LDS, ou seja, descreve o sólido na linguagem especificada por [ESPE90]. Em seguida, por intermédio do programa "CompilaCsg", o código LDS é transformado em uma árvore CSG do sólido. Esta árvore é então processada pelo programa CSG2CSPG que converte as primitivas para composições de semiespaços polinomiais e gera nova árvore na qual todas as primitivas estão na forma Bézier.

A partir desta versão modificada da árvore CSG, o programa-SUBSIMPL procede a Subdivisão Simplicial Adaptativa, computando uma aproximação linear por partes para a fronteira do sólido, que pode ser visualizada pelo módulo VISUAL, ou utilizada para uma estimativa da área de superfície do sólido através do módulo PROPSIMPL. Outra alternativa possível é a geração da árvore de subdivisão simplicial por SUBSIMPL, que combinada com PROPSIMPL, pode fornecer uma estimativa para o volume do sólido. A árvore de subdivisão também pode ser usada para otimização de algoritmos diversos como *RAY-TRACING* etc.

Passaremos, agora, a breve explicação sobre como os programas funcionam e como devem ser executados, além dos respectivos arquivos de entrada/saída.

A.2. CSG2CSPG

O objeto CSG é construído pelo usuário através da Linguagem de Definição de Sólidos (LDS) especificada por [ESPE90]. Esta linguagem considera operações booleanas

entre instanciações de algumas primitivas: BLOCO, ESFERA, CILINDRO, PLANO, CONE, TORO, POLINOMIAL e HELICÓIDE.

Para que o programa de subdivisão simplicial possa tratar a árvore CSG gerada pelo compilador LDS (*CompilaCsg*) é necessário que se façam, basicamente, três modificações. Primeiramente deve-se eliminar o operador "diferença" da árvore CSG, substituindo-o convenientemente por interseções e complementos. Em segundo lugar, é necessário transformar as primitivas LDS limitadas em subárvores CSG compostas apenas por semiespaços polinomiais. Finalmente, é necessário especificar os simplexos iniciais da decomposição hierárquica e obter a expressão de todos os semiespaços polinomiais em termos de seus coeficientes de Bézier relativos a cada um destes simplexos iniciais.

A.2.1. FORMATO DO ARQUIVO DE ENTRADA

Os arquivos de entrada de dados para o programa CSG2CSPG são os arquivos de saída do compilador LDS, ou seja, são aqueles que contém a árvore CSG tal como construída pelo usuário. Normalmente estes arquivos têm extensão ".CSG". O seu formato e o seguinte:

```
- Simples (Boolean ou unsigned char) ⇒ A árvore CSG é simples (primitiva)?
  SE (Simples) ENTÃO {
      - Material (int) ⇒ Material do qual é feita a primitiva
      - TipoPrim (int) ⇒ Tipo de primitiva
       SE (TipoPrim = TORO) {
            - RaioMenor (float) ⇒ Raio do anel menor do toro
       }
       SENÃO SE (TipoPrim = POLINOMIAL) {
            - Grau (int) ⇒ Grau da polinomial
            - Coef[TOTALCOEF(Grau)] (float) ⇒ coeficientes na base algébrica
       }
       - Matriz[4][4] (float) ⇒ Matriz de instanciação: Primitiva ⇒ Cena
  }
  SENÃO {
       - Operador (Byte ou unsigned char) ⇒ Operador booleano
       - <Árvore Esquerda> ⇒ estrutura idêntica para a subárvore esquerda
       - <Árvore Direita>  ⇒ estrutura idêntica para a subárvore direita
  }
```

O tipo da primitiva é um inteiro associado as primitivas originais da seguinte forma: 0-BLOCO, 1-ESFERA, 2-CILINDRO, 3-PLANO, 4-CONE, 5-TORO, 6-POLINOMIAL e 7-HELICÓIDE.

A função TOTALCOEF(n) retoma o total de coeficientes de uma polinomial para determinado grau *n*. Este total é obtido da seguinte forma:

$$\text{TOTALCOEF}(n) = \frac{(n+d)!}{n!\ d!} \quad ; \text{onde d é a dimensão do espaço}$$

A.2.2. ELIMINAÇÃO DO OPERADOR DIFERENÇA

A remoção do operador diferença das árvores CSG facilita a tarefa de localização do pedaço do objeto presente em determinada região do espaço, mais precisamente, dentro de um simplexo. Algoritmos de simplificação de árvores CSG já são bem conhecidos ([TILO84]) e no caso da ausência do operador diferença eles são bastante simples.

A eliminação da diferença se dá pela sua substituição por uma interseção com o complemento do objeto a ser subtraído. Simbolicamente, dados dois sólidos CSG A e B:

$$A \setminus B = A \cap \overline{B}$$

No caso de semiespaços, a complementação se dá pela negação de todos os seus coeficientes, isto é, pela multiplicação de cada um deles por –1.

A.2.3. EXPANSÃO DAS PRIMITIVAS LIMITADAS

Cada primitiva composta da árvore CSG original deve ser expandida em subárvore CSG cujas folhas representam semiespaços polinomiais. Como as primitivas LDS são bastante simples (uma vez que as transformações que poderiam tomá–las complicadas ficam restritas à matriz de instanciação), as subárvores CSG, decorrentes da expansão de cada uma delas, são consequentemente fáceis de serem construídas.

Primeiramente, definimos uma série de novas primitivas, cada qual associada a um semiespaço básico:

SEPXO $\Rightarrow x \geq 0$
SEPX1 $\Rightarrow x \leq 1$
SEPYO $\Rightarrow y \geq 0$
SEPY1 $\Rightarrow y \leq 1$
SEPZO $\Rightarrow z \geq 0$
SEPZ1 $\Rightarrow z \leq 1$
SEESF $\Rightarrow x^2 + y^2 + z^2 \leq 1$
SECIL $\Rightarrow x^2 + y^2 \leq 1$
SECON $\Rightarrow x^2 + y^2 \leq z^2$
SETRD $\Rightarrow (x^2 + y^2 + z^2 - 1 - r^2)^2 - 4\,(x^2+y^2) \leq 0$
SEPOL $\Rightarrow$ Grau e coeficientes a cargo do usuário
SEHEL $\Rightarrow x\,(4\,(z-0.5)^2 - 1) + y\,(z-0.5)\,(2.2654816\,(z-0.5)^2 - \pi)$

O número real rpresente na definição de "SETRD" é o raio do anelmenor do toro que é especificado pelo usuário na primitiva original.

As primitivas ESFERA, PLANO, TORO e POLINOMIAL já são constituídas de um único semiespaço, não necessitando, portanto, de expansão em subárvores. Para as demais primitivas, no entanto, a expansão se faz necessária e é feita tal como mostrada na figura A.2.

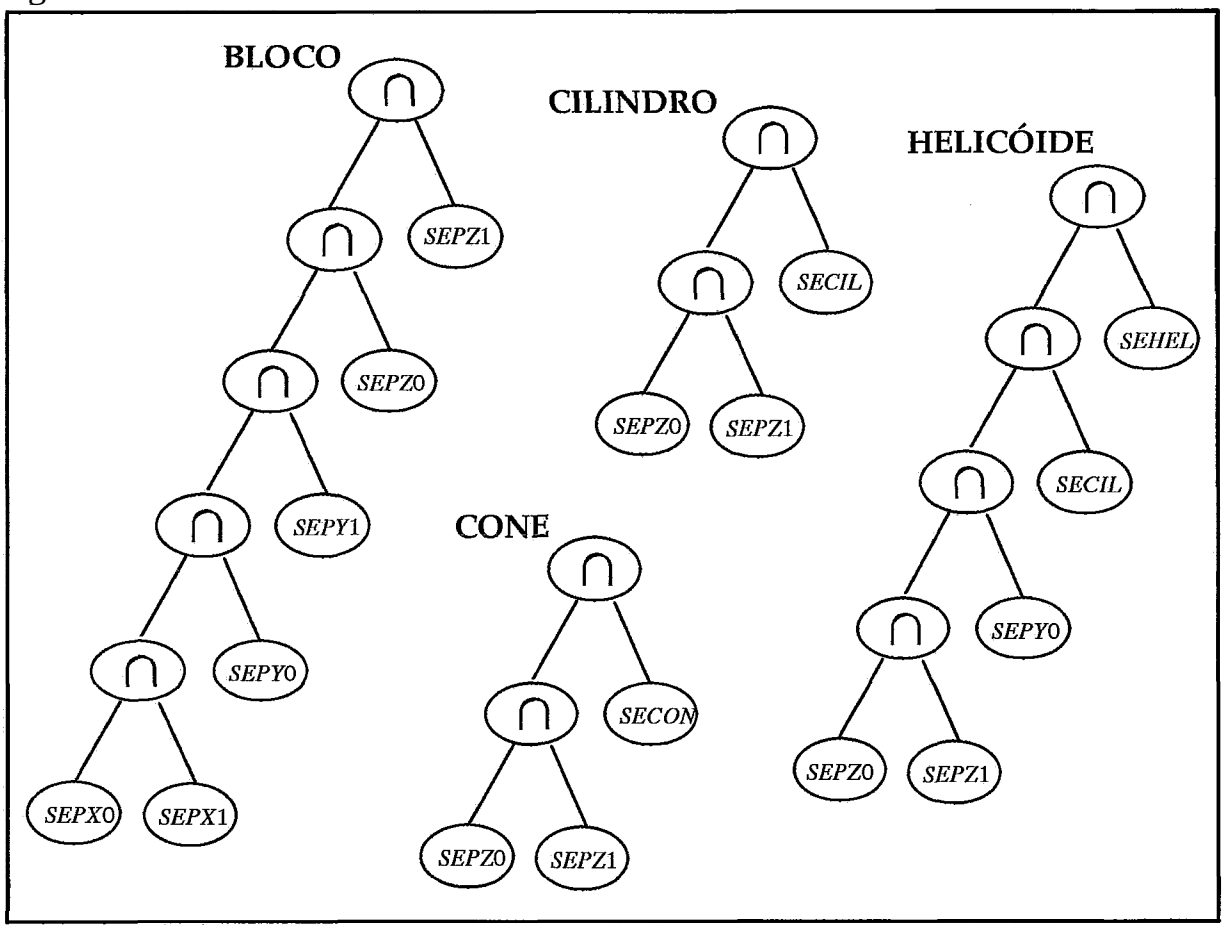


Fig. A.2: representação de primitivas limitadas através de serniespaços.

Nas primitivas polinomiais, os coeficientes devem estar em ordem lexicográfica decrescente de expoentes. Isto significa, por exemplo, que $P(x,y,z) = x^2 - 4xy + 3z^2 - 5 = 1\,x^2y^0z^0 1^0 - 4\,x^1y^1z^0 1^0 + 0\,x^1y^0z^1 1^0 + 0\,x^1y^0z^0 1^1 + 0\,x^0y^2z^0 1^0 + 0\,x^0y^1z^1 1^0 + 0\,x^0y^1z^0 1^1 + 3\,x^0y^0z^2 1^0 + 0\,x^0y^0z^1 1^1 - 5\,x^0y^0z^0 1^2$ deve ser definida pelos coeficientes {1, –4, 0, 0, 0, 0, 0, 3, 0, –5} nesta ordem.

A.2.4. SIMPLEXOS INICIAIS DA DECOMPOSIÇÃO HIERÁRQUICA

Para que o processo de subdivisão simplicial possa ser feito, é obviamente necessário que se defina a região do espaço que deve ser decomposta em simplexos, mais especificamente, os simplexos iniciais da decomposição hierárquica. Com o objetivo de facilitar esta tarefa para o usuário, o programa trabalha com a tradicional noção de caixa

envolvente (*Bounding Box*), ou seja, o usuário especifica um paralelepípedo que contem a região de interesse e o próprio programa se encarrega de subdividí-lo nos simplexos iniciais. No caso, em 6 simplexos iniciais, tal como mostrados na figura 2.2.

A caixa envolvente é fornecida pelo usuário através da opção de linha de comando "–B X_{Min} Y_{Min} Z_{Min} X_{Max} Y_{Max} Z_{Max}" onde os pontos $P_{Min} = (X_{Min}, Y_{Min}, Z_{Min})$ e $P_{Max} = (X_{Max}, Y_{Max}, Z_{Max})$ são os extremos de uma maior diagonal do paralelepípedo.

Nem todas as árvores CSG determinadas pelo usuário definem um sólido fisicamente válido, uma vez que a primitiva PLANO é não–limitada e a primitiva POLINOMIAL pode ou não ser, dependendo exclusivamente do usuário. Para garantir que a nova árvore gerada, constituída apenas de primitivas definidoras de semiespaços, representa sólidos fisicamente válidos, o programa acrescenta a árvore CSG original uma interseção com a caixa envolvente, isto é, o objeto CSG gerado passa a ser o objeto definido pelo usuário limitado a região onde está a caixa envolvente.

Se o usuário desejar que a árvore original não seja alterada, basta utilizar a opção de linha de comando "–L". Esta opção retira os limites da caixa envolvente e é muito útil em determinadas situações quando se quer analisar o comportamento de algumas primitivas isoladamente.

A.2.5. EXECUÇÃO DO PROGRAMA

A chamada do programa deve ser feita da seguinte forma:

CSG2CSPG [–D] [–L] [–B XMin YMin ZMin XMax YMax ZMax] < Input > Output

Os arquivos "Input" e "Output" devem ser sempre especificados (a menos que se utilize pipelines) e representam, respectivamente, os arquivos de entrada e saída.

A opção "–D" executa o programa no modo default. Neste modo, os limites da caixa envolvente são incluídos e a caixa envolvente definida é o paralelepípedo que uma de suas diagonais tem pontos extremos $P_{Min} = (-5, -5, -5)$ e $P_{Max} = (5, 5, 5)$.

A opção "–L" retira do objeto os limites da caixa envolvente.

A opção "–B" define a caixa envolvente.

A.2.6. FORMATO DO ARQUIVO DE SAÍDA

O formato de saída do programa, próprio para entrada do programa SUBSIMPL, é tal como mostrado na seção **A.3.1.** Particularmente, neste caso, o número de setores iniciais e igual a seis, exatamente aqueles que decompõem a caixa envolvente.

A.3. SUBSIMPL

O programa SUBSIMPL é responsável pela principal parte do sistema: a geração da Subdivisão Simplicial Adaptativa. Ele recebe a árvore CSG, representativa do sólido, devidamente trabalhada e procede a realização de uma entre três tarefas passíveis de serem requisitadas.

De acordo com o desejo do usuário, o programa pode gerar a árvore de subdivisão simplicial, uma aproximação linear para a fronteira do sólido CSG ou estatísticas sobre o processo de subdivisão. A aproximação linear pode ser diretamente visualizada pelo programa VISUAL ou utilizada para uma estimativa da área de superfície do sólido, através do programa PROPSIMPL. A árvore de subdivisão simplicial pode também ser utilizada para otimizar algoritmos de RAY-TRACING e assemelhados ou, até mesmo, num algoritmo de conversão CSG→B-Rep.

Para geração da subdivisão simplicial, o programa faz uso da representação de cada primitiva CSG na Base de Bernstein. Com o auxílio de critérios de classificação de simplexos contra primitivas, o programa subdivide o espaço adaptativamente.

Os critérios de classificação utilizam os coeficientes de Bézier das polinomiais definidoras das primitivas, direta ou indiretamente, para estipular limitantes para a variação de cada uma em relação aos interiores dos simplexos.

A composição de várias primitivas para formação de uma função implícita capaz de representar o sólido globalmente é feita através de funções MIN-MAX. Estas funções, chamadas de funções características dos sólidos, são simplificadas a medida que o processo de subdivisão progride, de modo que nos simplexos mínimos da decomposição, somente as primitivas realmente relevantes para formação da fronteira do sólido em cada região são consideradas.

A.3.1. FORMATO DO ARQUIVO DE ENTRADA

O arquivo de entrada é constituído de uma sequência arbitrária de setores do espaço. Entendemos por setor do espaço um simplexo com uma árvore CSG associada, representativa da porção do sólido presente no simplexo. As primitivas desta árvore estão representadas na Base de Bernstein. O arquivo deve ter a forma:

- NPrimCSG (int) ⇒ Nº de primitivas da árvore CSG
- Simples (Boolean ou unsigned char) ⇒ A árvore CSG é simples (primitiva)?

SE (Simples) ENTÃO {
  - NomePrim (int) ⇒ Nome da primitiva presente no nó
}

```
    SENÃO {
        – OperCSG (Byte ou unsigned char) ⇒ Operador booleano para os nós filhos
        – <Árvore Esquerda> ⇒ estrutura idêntica para a subárvore esquerda
        – <Árvore Direita>  ⇒ estrutura idêntica para a subárvore direita
    }

– NSetores (int) ⇒ Nº de setores iniciais a serem subdivididos
Setor 1 {
  – VertSimp[4][3] (double)⇒ Coordenadas dos vértices do 3–simplexo
  PARA i ← 1 ATÉ NPrimCSG {
        – Nome_i (int)    ⇒ Nome da i-ésima primitiva
        – Material_i (int)⇒ Material do qual é feito a i-ésima primitiva
        – Grau_i (int)    ⇒ Grau da polinomial que define a i-ésima primitiva
        – Coef[TotalCoef(Grau_i)] (double)⇒ coeficientes de Bézier da primitiva
  }
}
  a
  •
  •
Setor NSetores {
      •
      •
      •
}
```

A.3.2. EXECUÇÃO DO PROGRAMA

O programa deve ser chamado da seguinte forma:

```
SUBSIMPL [–D] [–E] [–O <(S)ubdivisão | (A)prox. Linear>]
         [–P <Nº Máx. Prim.>] [–N <Nível>] [–M <Subnível>] [–T <Tolerância>]
         < Input > Output
```

Os arquivos "Input" e "Output" devem ser sempre especificados (a menos que se esteja utilizando *pipelines*) e representam, respectivamente, os arquivos de entrada e saída.

A opção "–O" define a saída do programa: árvore de subdivisão ou aproximação linear. O usuário deve especificar a letra (char) referente ao tipo de saída que ele deseja. É interessante notar que o programa ***só*** se interessa pela primeira letra que vem logo após a opção, não se importando se ela é maiúscula ou minúscula, de forma que as duas chamadas:

```
subsimpl –O a c objeto.cspg > objeto.apl
subsimpl –o APLIN c objeto.cspg > objeto.apl
```

fazem exatamente a mesma coisa, ou seja, calculam uma aproximação linear para a fronteira do sólido especificado pelo arquivo "objeto.cspg".

A opção "-P" estipula o número máximo de primitivas presentes na árvore CSG de um simplexo para que o processo de subdivisão possa ser interrompido precocemente. Definido p (int) o máximo de primitivas, em qualquer simplexo de qualquer nível de subdivisão cuja árvore CSG a ele restrita possuir menos de p primitivas (inclusive), o processo de subdivisão age como se tivesse chegado a um simplexo mínimo.

A opção "-T" define a tolerância da subdivisão. A unidade utilizada como referência é a maior aresta dos simplexos mínimos, ou seja, especificada a tolerância E, todos os simplexos do último nível de subdivisão têm todas as suas arestas com comprimentos menores que E.

Definida a tolerância E (double), um nível máximo de subdivisão é automaticamente calculado de modo a satisfazê-la. O cálculo é bastante simples: se l é o comprimento da maior aresta de um simplexo inicial, o nível de subdivisão que satisfaz a tolerância em uma dimensão é dado pela expressão:

$$n_1 = \left\lceil \log_2 \left(\frac{l}{\epsilon} \right) \right\rceil$$

O nível de subdivisão que satisfaz a tolerância nas três dimensões é então dado por:

$$NivelMax = 3 \times n_1$$

A opção "-N" define o nível máximo de refinamento de cada simplexo inicial (int). A definição do nível de refinamento implica na especificação de tolerância para o processo de subdivisão e, por conseguinte, as duas opções são mutuamente exclusivas. Quando existe conflito, a especificação de tolerância têm prioridade e o nível de refinamento fica definido em função dela.

A opção "-M" determina o nível da subdivisão em que devem ser calculados os pedaços de aproximação linear dentro de cada simplexo mínimo. Neste nível, as primitivas são linearizadas, isto é, substituídas pelas funções lineares que melhor lhes aproximam e a subdivisão continua até o nível zero.

Esta opção serve para refinar seletivamente as regiões. Somente os simplexos que contêm mais de uma primitiva na árvore CSG do sólido a eles restrita, continuam a ser subdivididos. Desta forma, as regiões da fronteira do sólido onde existe a colagem de primitivas distintas podem ser minimizadas, sem que com isto haja comprometimento das regiões onde somente uma primitiva é responsável pela fronteira.

A chamada do programa:

```
subsimpl -o a -n 21 -m 3 c objeto.cspg > objeto.apl
```

estabelece dois estágios de subdivisão para o cálculo de uma aproximação linear para a fronteira do sólido especificado por "objeto.cspg". No primeiro estágio, que vai do nível 21

até o nível 3, o processo de subdivisão é feito normalmente, trabalhando-se com todas as primitivas tal como foram especificadas no arquivo de entrada. No nível 3, escolhido para cálculo da aproximação linear, nos simplexos que contiverem árvores CSG compostas, as primitivas serão linearizadas e o segundo estágio de subdivisão toma lugar, indo até o nível 0. Isto representa um refinamento equivalente a nível 18, para as regiões que contêm apenas uma primitiva responsável pela fronteira do sólido, e de nível 21 para aquelas onde existe colagem de primitivas.

A opção "-E deve ser utilizada sempre que o usuário desejar apenas uma análise estatística do processo de subdivisão. Neste caso, o programa trabalha normalmente como se fosse gerar uma aproximação linear sem, no entanto, exportar as faces computadas. Todas as informações pertinentes ao processo de subdivisão vão sendo armazenadas e, no final, são exportadas tal como descrito na seção A.3.3.

É importante notar que, como o processo de subdivisão funciona normalmente, as outras opções de execução do programa também podem ser requisitadas, fornecendo mais opções ao usuário para que ele possa avaliar, por intermédio de estatísticas, o comportamento de todo o processo. A única opção que não pode ser utilizada no modo de estatística é aquela que define a tolerância da apmximação ("-T"). A razão disto é a flexibilidade do arquivo de entrada do programa.

A tolerância especificada pelo usuário define automaticamente o nível máximo de refinamento da decomposição de cada simplexo inicial. Como o arquivo de entrada permite uma quantidade arbitrária de setores iniciais do espaço e não faz nenhuma restrição a cerca do tamanho de cada um, a especificação de tolerância pode acarretar diferentes níveis de subdivisão para diferentes simplexos iniciais, o que inviabiliza as avaliações estatísticas.

A opção "-D" faz com que o programa seja executado em seu modo *default* que é equivalente a chamada:

subsimpl -o a -p 0 -n 15 -m 0 < Input > Output

ou seja, requisição de cálculo de apmximação linear, número máximo de primitivas por simplexo mínimo 0 (subdivisão exaustiva se necessário), nível máximo de subdivisão 15 e nível de cálculo de aproximação linear 0 (sem linearização).

A.3.3. FORMATOS DO ARQUIVO DE SAÍDA

São três os formatos possíveis para o arquivo de saída do programa SUBSIMPL. Dependendo da requisição do usuário, o programa pode gerar: (a) apmximação linear para a fronteira do sólido; (b) árvore de subdivisão simplicial e (c) estatísticas sobre o processo de subdivisão. Após cada execução, independentemente do formato de saída requisitado, é exibido o tempo de CPU gasto no processamento. Os formatos, em cada caso, são dados a seguir.

A.3.3.1. APROXIMAÇÃO LINEAR PARA A FRONTEIRA DO SÓLIDO

A saída, neste modo, consiste em uma lista de faces triangulares da aproximação, cada uma contendo o índice do material do qual é feita.

- ObjSaída (char) ⇒ objeto no arquivo de saída (neste caso "A")
Face 1 {
  - Material (int) ⇒ Índice do material do qual é feita a face
  - T[3][3] (double) ⇒ Coordenadas dos vértices da face
}
Face 2 {
  •
  •
  •
}
•
•
•

É importante notar que o número de faces não pode ser escrito no topo do arquivo pois, o processo de subdivisão é recursivo e pode ser utilizado em *pipelines*.

O mateiral de uma face construída com mais de uma primitiva, ou seja, das faces híbridas, recebe o índice 255.

A.3.3.2. ÁRVORE DE SUBDIVISÃO SIMPLICIAI,

No modo de geração da árvore de subdivisão, o programa vai exportando a árvore CSG do sólido, simplificando–a para cada simplexo da decomposição, a medida que os vai visitando. Inicialmente é gravado o número de simplexos iniciais e, em seguida, é disparado o processo de subdivisão para cada um deles. O formato do arquivo é:

- ObjSaída (char) ⇒ objeto no arquivo de saída (neste caso “S”)
- NSetores (int) ⇒ Nº de setores iniciais
Setor 1 {
  - VertSimp[4][3](double) ⇒ coordenadas dos vértices dos simplexos
  Árvore de Decomposição (
    - TipoNo (int) ⇒ Informa a classificação da do simplexo em relação ao sólido
    SE (TipoNo = NOFRONT){
      Árvore CSG {
        - Simples (char) ⇒ Informa se o nó da árvore é simples
        SE (Simples){
          - NomePrim (int) ⇒ Nome da primitiva presente no nó
          - MaterPrim (int) ⇒ Material do qual ela é feita

```
                - GrauPrim (int) ⇒ Grau da polinomial que a define
                - Coef[TotalCoef(GrauPrim)] (double)⇒ Coefs de Bézier no simplexo
              ]
              SENÃO {
                - OperCSG (char) ⇒ Operador CSG a ser aplicado nas subárvores
                <Subárvore CSG a esquerda>
                <Subárvore CSG a direita>
              ]
            }
          }
          SENÃO SE (TipoNo = NOMISTO) {
            <Árvore de Decomposição do Simplexo Filho 1>
            <Árvore de Decomposição do Simplexo Filho 2>
          ]
        }
      ]
      •
      •
      •

      Setor NSetores {
        •
        •
        •
      }
```

Os tipos possíveis de nó são: O–NOCHEIO, 1–NOVAZIO, 2–NOMISTO e 3–NOFRONT.

A.3.3.3. ESTATÍSTICA DO PROCESSO DE SUBDIVISÃO

Neste modo, o programa processa a subdivisão tal como em qualquer um dos modos anteriores, mas apenas armazena as informações pertinentes, sem gerar resultados. No final, ele fornece todas as informações armazenadas. A saída é da forma:

```
ESTATISTICAS DE EXECUCAO
Nivel de subdivisao:          n.
Simplexos visitados:          A (100.0%).
Simplexos CUT:                B (Inf%).
Simplexos mistos:             C (Inf%).
Simplexos subdiv. indev.:     D (NaN%).
Primitivas visitadas:         E (100.0%)).
Primitivas CUT:               F ( NaN%).

        •
        •
```

Nivel de subdivisao:	0.
Simplexos visitados:	G (100.0%).
Simplexos CUT:	H (X %).
Simplexos mistos:	I (Y %).
Simplexos subdiv. indev.:	J (Z %).
Primitivas visitadas:	K (100.0%).
Primitivas CUT:	L (W %).

TOTAIS GERAIS:

Total de Simplexos visitados:	T1 (100.0%).
Total de Simplexos CUT:	T2 (P %).
Total de Simplexos mistos:	T3 (Q %).
Total de Simplexos Subdiv. Indev.:	T4 (R %).
Total de Primitivas visitadas:	T5 (100.0%)).
Total de Primitivas CUT:	T6 (S %).

A.4. VISUAL

O programa VISUAL tem como objetivo exibir, de maneira rudimentar, a aproximação linear por partes computada pelo programa SUBSIMPL, para algum particular objeto CSG. A forma de visualização é dita rudimentar pois a interface com o usuário é bastante simples e provê muito poucos recursos.

O usuário dispõe de duas formas de exibição do objeto: WIREFRAME e SÓLIDO. No modo WIREFRAME, são exibidas apenas as arestas das faces da aproximação do objeto, sem nenhuma preocupação com a remoção das linhas ocultas ao observador, O modo SÓLIDO, por sua vez, preenche as faces da aproximação com a cor do material ali presente. É utilizada, neste modo, a técnica de *z-buffer* para eliminação das partes do objeto ocultas ao observador.

Existe a possibilidade de se alterar algumas características do sistema projetivo utilizado para a visualização, permitindo-se a observação da aproximação do objeto de vários pontos de vista diferentes. Os parâmetros do sistema projetivo são definidos em um arquivo de configuração que pode ser alterado convenientemente.

A.4.1. FORMATO DO ARQUIVO DE ENTRADA

Os arquivos de entrada devem ter sido gerados pelo programa SUBSIMPL no modo de geração de aproximação linear. O seu formato é, portanto, aquele descrito na seção A.3.3.1.

A.4.2. INTERFACE COM O USUÁRIO

A interface com o usuário foi totalmente construída em cima do ambiente SUNVIEW. Ao ser chamado, o programa abre uma janela, semelhante a mostrada na figura A.3.

A janela é constituída de uma região de desenho acoplada a um painel com sete botões que comandam a execução.

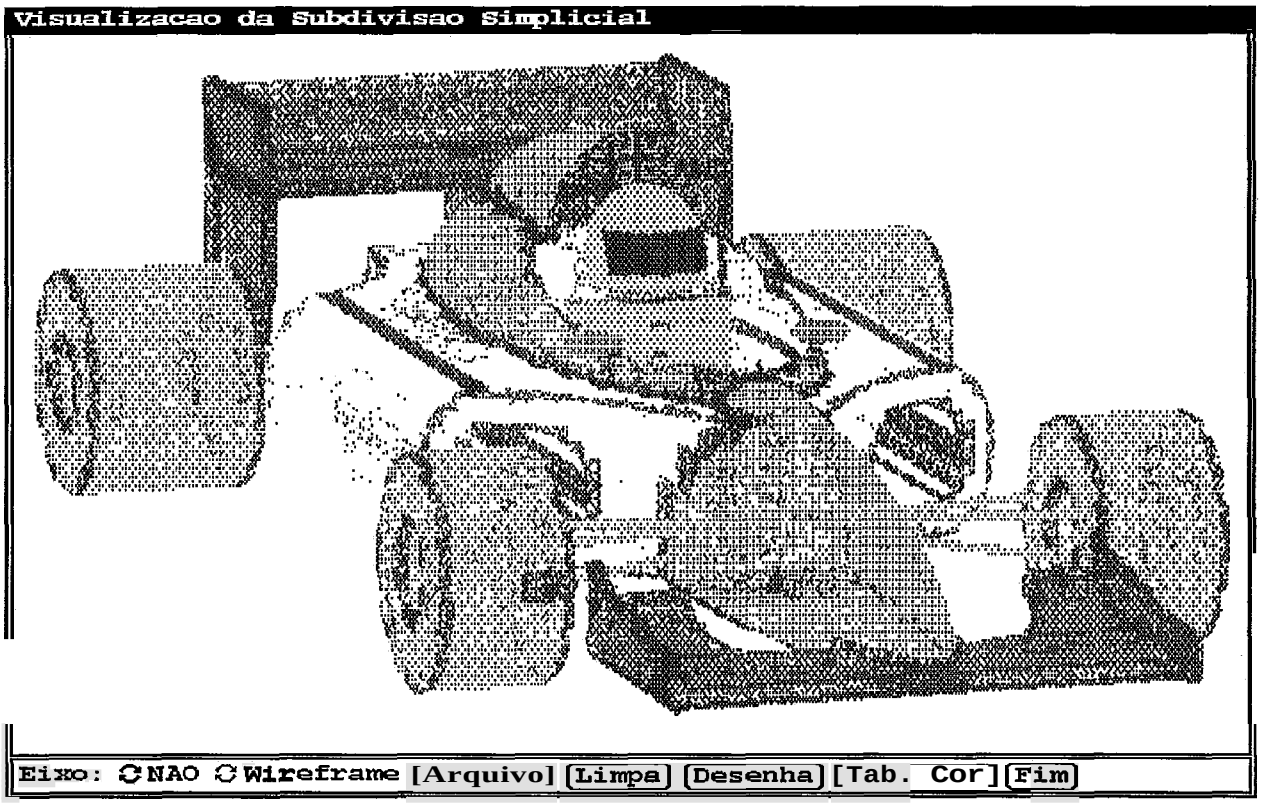


Figura **A.3:** janela de interface com o usuário do programa VISUAL.

Da esquerda para a direita, o primeiro botão permite ao usuário determinar se os eixos cartesianos devem ser plotados, para facilitar a compreensão da atual configuração do sistema projetivo.

O segundo botão especifica o formato de exibição do objeto. Como explicado, os dois formatos possíveis são WIREFRAME e SÓLIDO. Tanto num formato quanto no outro, as faces são pintadas conforme a cor do material da qual são feitas. Para o programa de subdivisão simplicial, a cor de uma face só pode ser definida se esta face for aproximadora de uma região que contém apenas uma primitiva da árvore CSG. A cor, neste caso, é a cor do material do qual é feita essa primitiva. Quando não existe apenas uma primitiva, isto é, a face aproxima uma região onde a árvore CSG é composta, o programa de subdivisão arbitra o material ali presente como sendo o de índice 255. Simplificando, todas as faces aproximadoras de regiões onde a árvore CSG é simples têm a cor da primitiva que está na árvore, as demais faces, aproximadoras de regiões onde a árvore CSG é composta, recebem a cor de índice 255. Isto ajuda a localizar as regiões onde ocorrem colagem de primitivas.

O arquivo que contém a aproximação linear por partes do objeto pode ser especificado através do terceiro botão. Quando ele é "pressionado", nova janela aparece no lugar do painel, requisitando o nome do arquivo. O nome do arquivo atual é exibido, de modo que basta teclar <*Return*> no caso de acionamento indevido do botão. Assim que o novo nome é digitado, a janela desaparece, restaurando o painel com os botões.

Podem ser plotadas várias aproximações lineares, umas por sobre as outras, todas utilizando a mesma tabela de cores. Desta forma se faz necessário um pmcedimento para limpeza da tela, sempre que o usuário achar conveniente. Esta limpeza pode ser feita por intermédio do quarto botão.

A plotagem da aproximação linear se dá, efetivamente, quando o quinto botão é pressionado. É disparado um procedimento que lê e plota cada face da aproximação presente no arquivo previamente definido. Este botão só deve ser pressionado quando todos os outros parâmetros já tiverem sido ajustados, principalmente aqueles referentes ao sistema projetivo, do qual falaremos na próxima seção. A exibição é tarefa onerosa, que demanda certo tempo e que não pode ser interrompida só devendo ser requisitada, portanto, no momento correto.

O sexto botão serve para exibir a nova configuração de cores definida pelo usuário no arquivo "tabcor.dat". Falaremos melhor disto na seção A.4.4. O sétimo e último botão, bastante intuitivo, serve para encerrar a execução do programa, retirando a janela do ambiente de trabalho.

A.4.3. SISTEMA PROJETIVO

O sistema projetivo utilizado para a visualização da aproximação do objeto é definido por quatro parâmetros que ficam a critério do usuário: ponto de referência (PR), direção do observador (DO), distância do observador ate o plano de projeção (DP) e janela no plano de projeção (JP). A figura A.4 ilustra uma interpretação geométrica para estes parâmetros.

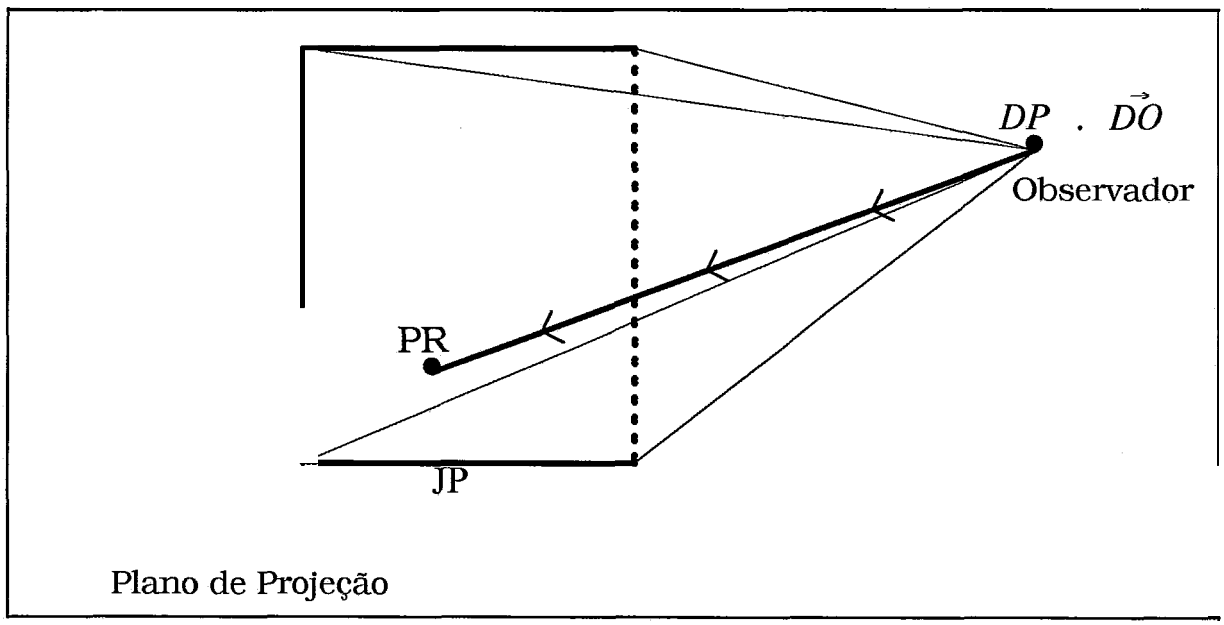


Fig. **A.4:** modelo geométrico para o sistema projetivo utilizado.

O ponto PR é o ponto do plano de projeção para aonde o observador está olhando. O vetor DO fornece a direção sobre a qual o observador está olhando. O número real DP é a distância do observador até o plano de projeção, segundo a direção por onde olha. Finalmente, o retângulo *JP* é a região do plano de projeção que será vista pelo observador.

O programa procura por estes parâmetros em um arquivo em disco antes de começar a exibição do objeto, isto é, imediatamente após o quarto botão ter sido pressionado. Deste modo, é possível observar o objeto de acordo com diversas configurações dos parâmetros dentro de uma mesma execução do programa. Basta alterar (e salvar) o arquivo de configuração antes de cada solicitação de exibição do objeto.

O arquivo em disco deve ser ASCII, ter o nome "sisproj.dat" e deve estar presente no diretório corrente (aquele de onde foi feita a chamada de VISUAL) ou no diretório *HOME* do usuário. *O* formato do arquivo é dado abaixo:

PR_x PR_y PR_z ⇒ Coordenadas do ponto de referência do sistema projetivo (double)
DO_x DO_y DO_z ⇒ vetor direção do observador (double]
DP ⇒ Distancia do observador até o ponto de projeção (double).
JP_{XMin} JP_{YMin} JP_{XMax} JP_{YMax} ⇒ Diagonal da janela no plano de projeção (double).

Uma observação importante é que a janela do plano de projeção é definida com orientação fixa, o que impede efeitos de torção da imagem do objeto. Por conseguinte, não há maneira de se colocar um objeto de cabeça para baixo alterando-se apenas os parâmetros do sistema projetivo.

A.4.4. TABELA DE CORES

Quando é executado, o programa procura pela tabela de cores que deve utilizar. Esta tabela deve estar no diretório corrente ou no *HOME* do usuário. *O* arquivo deve ser ASCII, ter o nome "tabcor.dat" e estar no formato abaixo:

ICor (int) Red_{ICor} (char) $Green_{ICor}$ (char) $Blue_{ICor}$ (char)
e
e
e
ICor (int) Red_{ICor} (char) $Green_{ICor}$ (char) $Blue_{ICor}$ (char)

Cada linha contém o índice da tabela de cores ICor e a tripla (R, G, *B*) definidora da cor. Este índice é relacionado com o material do qual é feita cada face da aproximação linear e não pode ultrapassar o tamanho máximo da tabela de cores, ou seja, deve estar entre 0 e 255. O valor de índice 0 define a cor de fundo. A atual versão do programa permite somente 128 cores mas isto pode ser modificado se o programa for recompilado.

As entradas da tabela de cores que, por ventura, não forem definidas são associadas a (255, 255, 255), ou seja, cor branca. As entradas não pertencentes a faixa permitida (atualmente 0..127), são associadas a maior entrada possível (atualmente 127). Se por acaso o arquivo "tabcor.dat" não existir, a cor de fundo fica sendo preta e todas as outras entradas da tabela passam a definir cor branca.

O usuário pode modificar a tabela de cores, depois que o programa tiver plotado uma coleção de aproximações lineares, alterando da maneira que lhe for conveniente as cores associadas a cada parte plotada. Depois de cada modificação do arquivo "tabcor.dat",

basta pressionar o sexto botão do painel e a nova configuração de cores será exibida. E interessante notar que não existe a necessidade de uma replotagem da aproximação linear para que as cores sejam modificadas.

A.4.5. EXECUÇÃO DO PROGRAMA

Para executar o programa de visualização basta digitar:

VISUAL [Input]

O parâmetro "Input" é opcional e designa o nome do arquivo a ser exibido. Após a chamada, o programa instala a janela de desenho e o painel de controle que passa a comandar a execução. Obviamente, o nome do arquivo definido na linha de comando pode ser substituído, acionando-se o botão apropriado.

Sugerimos, fortemente, a requisição de um editor de textos ao ambiente *SUNVIEW* sempre que o programa estiver sendo executado. Neste editor deve ser carregado o arquivo com os parâmetros do sistema projetivo. Isto facilita sobremaneira a tarefa de reconfiguração para novas exibições.

A.5. VIS2LCG

O programa de visualização das aproximações lineares por partes (VISUAL), como já mencionado, possui sérias deficiências no que se refere a interface com o usuário. Além disto, ele não é nada útil quando se deseja tratamento mais realístico da imagem final. Ele serve apenas para uma avaliação superficial, para fornecer uma primeira impressão da forma da aproximação linear.

De posse de arquivo com aproximação linear razoável, segundo os critérios do usuário, este pode utilizar programa melhor elaborado, desenvolvido no Laboratório de Computação Gráfica da COPPE/UFRJ, para trabalhar a imagem final (ou conjunto de imagens finais): o LCGRENDERER. Este programa possui uma interface com o usuário eficiente além de prover modelos de iluminação etc. Ver [FONS92].

O formato de entrada do LCGRENDERER, apesar de ser basicamente uma lista com vértices dos polígonos que constituem uma aproximação da casca do objeto, é bastante diferente do formato de entrada do programa VISUAL. Torna-se necessário, portanto, a presença de um conversor da saída do programa SUBSIMPL (no modo de cálculo de aproximação linear) para o formato de entrada do programa LCGRENDERER. Este conversor é o programa VIS2LCG.

A.5.1. FORMATO DO ARQUIVO DE ENTRADA

O formato do arquivo de entrada é idêntico ao formato de entrada para o programa VISUAL, descrito na seção A.3.3.1.

A.5.2. FORMATO DO ARQUIVO DE SAÍDA

O arquivo de saída, por ser próprio para o programa LCGRENDERER, recebe normalmente a extensão ".wfr" e tem o seguinte formato:

```
NV ⇒ Número de vértices (int)
Info ⇒ Informações sobre o objeto (int)
NV vezes: {
        Xi, Yi, Zi (i=0..(NV-1)) ⇒ Coordenadas do i-ésimo vértice (float)
}
NF ⇒ Número de faces (int)
NF vezes: {
        NVF ⇒ Número de vértices na face (int)
        NVF vezes: {
                IVert ⇒ Índice do vértice (IVert ∈ [0, NV-1])
        }
}
```

A.5.3. EXECUÇÃO DO PROGRAMA

Para executar a conversão entre os formatos de entrada entre os dois programas de visualização basta digitar:

VIS2LCG < Input > Output

O arquivo designado por "Input" deve conter uma aproximação linear gerada pelo programa de subdivisão simplicial (SUBSIMPL) enquanto que "Output" designa o arquivo no formato de entrada do programa LCGRENDERER.

A.6. PROPSIMPL

A partir do programa de subdivisão simplicial, tanto no modo de geração de aproximação linear como no de geração da árvore de subdivisão, é possível calcular algumas propriedades integrais de um sólido CSG.

O programa PROPSIMPL faz exatamente isto, a partir da aproximação linear calcula a área de superfície do objeto e, a partir da árvore de subdivisão, calcula o seu volume. Os valores obtidos não são muito precisos, em virtude de vários fatores, mas servem como uma primeira estimativa.

A.6.1. FORMATO DO ARQUIVO DE ENTRADA

Cada uma das propriedades é calculada a partir de determinada saída do programa de subdivisão simplicial. Sendo assim, se o usuário desejar o cálculo da área de superfície

do objeto, o arquivo de entrada deverá conter aproximação linear. No caso do cálculo de volume, o arquivo deverá conter a árvore de subdivisão simplicial. Os formatos estão, respectivamente, descritos nas seções A.3.3.1 e A.3.3.2.

A.6.2. EXECUÇÃO DO PROGRAMA

A chamada do programa deve ser feita do seguinte modo:

PROPSIMPL [-A] [-V] < Input

A opção "-A" requisita o cálculo da área de superfície do objeto enquanto que a opção "-V" requisita o cálculo do seu volume. As duas opções são mutuamente exclusivas, na medida em que os arquivos de entrada são diferentes.

O programa calcula sempre a última propriedade requisitada, ou seja, a linha de comando:

PROPSIMPL -V -V -V -A < Objeto.apl

fará com que o programa calcule a área da superfície do objeto definido pelo arquivo "Objeto.apl", que deverá conter a aproximação linear de algum objeto CSG.