CENTRO BRASILEIRO DE PESQUISAS FÍSICAS

LABORATÓRIO DE PROCESSAMENTO DIGITAL DE SINAIS E IMAGENS
(LPDSI/CAT/CBPF)

MODELAGEM DE SÓLIDOS

Aline da Rocha Gesualdi Mello, Cláudio Esperança,
Marcelo Portes de Albuquerque, Eugênio Suares Caner,
Márcio Portes de Albuquerque, Érica Marques da Silva

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2004

Sumário

Modelagem de sólidos

Aline da Rocha Gesualdi Mello, Cláudio Esperança,
Marcelo Portes de Albuquerque, Eugênio Suares Caner,
Márcio Portes de Albuquerque, Érica Marques da Silva

Resumo

A Modelagem de Sólidos é uma área da Computação Gráfica que estuda a construção e representação consistente de modelos de sólidos em computador. Podemos entendê-la como o conjunto de técnicas, teorias e sistemas que permitem calcular automaticamente qualquer informação geométrica bem definida sobre um sólido. Esta monografia se propõe apresentar três tipos de representação de sólidos e o mecanismo de conversão entre as representações. As representações abordadas são: por bordo; por geometria construtiva de sólidos, e; por enumeração do espaço em células.

1 Introdução

Na computação gráfica, o estudo sobre modelagem de sólidos permite compreender a construção e representação consistente de modelos de sólidos em computador. O termo modelagem de sólidos designa um conjunto de teorias, técnicas e sistemas que permitem criar um sólido com suas propriedades geométricas. Um sólido em seu sentido físico pode ser entendido como uma substância material , em cuja as moléculas têm situação mais ou menos estável e espontaneamente não mudam de lugar. Os corpos em estado sólido, têm forma própria e volume próprio, possuindo ainda, menor ou em maior grau, energia de forma e volume, ou seja, com resistência à deformações. De acordo com Martii Mäntyla [1] um sólido, sob a ótica da Computação Gráfica, é um subconjunto fechado e limitado do espaço Euclidiano tridimensional ($\mathbf{E}^3$).

Assumindo a afirmação acima podemos descrever sólidos por maneiras distintas como, por exemplo, através de suas fronteiras ou por campos escalares, originando desta forma três tipos de representações: por Bordo ou representação por fronteiras(B-rep - *Boundary Representation*); por geometria construtiva de sólidos ou operações de conjuntos (CSG - *Constructive Solid Geometry*), e; por enumeração do espaço em células (BSP-trees, octrees, etc).

1.1 Níveis de abstração em modelagem

Podemos distinguir quatro diferentes níveis de abstração em modelagem, que são:

1. *Modelo Físico*: descreve e caracteriza objetos reais através de suas propriedades físicas que, por sua vez, são percebidas através de nossos sentidos;
2. *Modelo Matemático*: utiliza linguagem matemática (teoria de conjuntos e topologia algébrica) para descrever de maneira rigorosa os objetos de interesse;

3. *Modelo de Representação*: associa ao modelo matemático uma representação apropriada para manuseio em computador, e;

4. *Modelo de Implementação*: adapta o modelo de representação de acordo com os recursos de implementação disponíveis, i.e., linguagem de programação escolhida, utilização de processamento paralelo, etc.

Desta forma, para efetuarmos a modelagem de objetos em computador é necessário uma idealização adequada dos mesmos. Esta idealização é feita com objetos matemáticos que podem ser definidos através da teoria de conjuntos e topologia algébrica [2].

1.2 Sólidos realizáveis e não-realizáveis

Um sólido é considerado realizável ou válido se satisfizer às seguintes condições:

- *Rigidez*: o objeto deve possuir forma invariante se for movido de um lugar para o outro, i.e., invariante às transformações de rotação, translação e a mudança de sistemas de coordenadas;

- *Finitude*: o objeto deve ter dimensões finitas e ser contido em uma posição finita do espaço;

- *Homogeneidade*: o objeto deve ter as mesmas propriedades em todos os seus pontos interiores;

- *Descritibilidade*: o objeto deve poder ser descrito através de um número finito de propriedades físicas, químicas, biológicas, etc;

- *Fechamento sobre Operações*: o resultado de operações geométricas realizadas em objetos válidos deve ainda ser um objeto válido.

Os sólidos podem ser representados por diversas formulações. As propriedades desejáveis das formas de representação são [3]:

- *Validade*: o modelo deve representar somente sólidos válidos;

- *Unicidade*: cada sólido válido deve ter apenas um modelo;

- *Não-ambigüidade*: cada modelo deve corresponder apenas a um objeto válido;

- *Completude*: o modelo deve ser completo, ou seja, conter uma variedade de informações sobre as quais as funções possam ser executadas;

- *Concisão*: o modelo deve ocupar o menor espaço de memória possível;

- *Simplicidade*: deve ser possível criar o modelo de forma simples e direta, sem que nenhuma característica especial de hardware seja exigida;

- *Eficiência*: as operações devem ser de fácil aplicação e apresentar respostas rápidas;

- *Fechamento sobre Operações*: as operações de descrição e manipulação devem preservar a validade do modelo.

Nas seções seguintes abordarei de forma mais detalhada três tipos de representação dos sólidos. Inicio pela representação por bordo ou B-rep (*Boundary Representation*), em seguida apresento o método CSG (*Constructive Solid Geometry*) e, por fim as representações por células que são: BSP-trees, octrees e quadtrees.

2 Representação de fronteira (B-rep)

A representação de fronteira, mais conhecida como B-rep, representa um sólido por uma malha de planos poligonais que define sua superfície. Tal malha é definida em função de vértices ou arestas, e é geralmente compostas por triângulos ou quadriláteros. A Figura 1 apresenta um modelo bem simples de um sólido de 6 faces. A estrutura B-rep é tida como uma representação clássica de sólidos, e é extensamente utilizada por aplicações 3D.

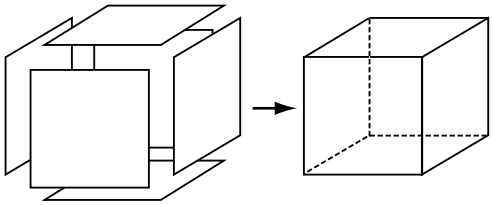


Figura 1: Modelo simples de um sólido B-rep de 6 faces.

A representação B-rep é bastante eficiente para representar sólidos cujas faces são planas. Por outro lado, as superfícies curvas podem ser aproximadas através de uma coleção de polígonos planos. Desta forma, quanto mais polígonos são utilizados, maior a aproximação em relação a superfície original da curva (Figura 2). Assim, a precisão do modelo nessas condições é diretamente proporcional ao custo computacional implicado na sua representação.

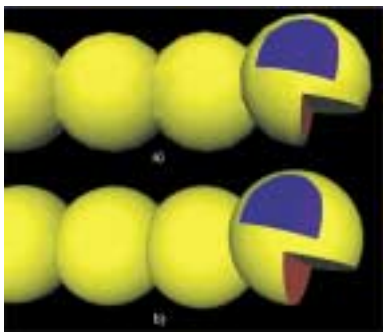


Figura 2: Modelos compostos por esferas definidas por diferentes quantidades de polígonos. (a) Modelo composto por esferas de 224 polígonos. (b) Modelo composto por esferas de 960 polígonos.

O custo computacional de se renderizar sólidos B-rep é reconhecidamente baixo. Desta forma, a B-rep é utilizada freqüentemente como forma intermediária por outras representações para fins de renderização. Portanto, outras representações são convertidas para B-rep a fim de serem renderizadas com um custo computacional menor do que seriam se o fossem no formato original.

3 Geometria Construtiva de Sólidos (CSG)

A geometria construtiva de sólidos (CSG) é um modelo de representação concebido em função das operações booleanas de sólidos. Objetos assim representados são resultantes da aplicação

das operações booleanas em sólidos elementares, chamados primitivas. Esferas, cones, cilindros e sólidos retangulares são formas comumente tidas como primitivas. A representação dos sólidos CSG se dá em função da composição de primitivas com operações booleanas segundo uma ordem lógica.

A estrutura que representa um sólido CSG é uma árvore na qual as operações booleanas são hierarquizadas. Esta estrutura é conhecida como árvore CSG (CSG tree). Cada uma das operações utilizadas é representada por um nó interno (não folha) da árvore, e cada primitiva por um nó folha. Os nós de uma árvore são organizados de forma que a operação relativa a um nó interno tenha sido aplicada aos sólidos obtidos a partir de seus nós filhos (Figura 3). Assim, uma árvore CSG estará organizada conforme a ordem em que as operações booleanas foram aplicadas, de forma que quanto mais rasos são os nós de uma árvore (mais próximos da raiz) mais recentes serão as operações correspondentes.

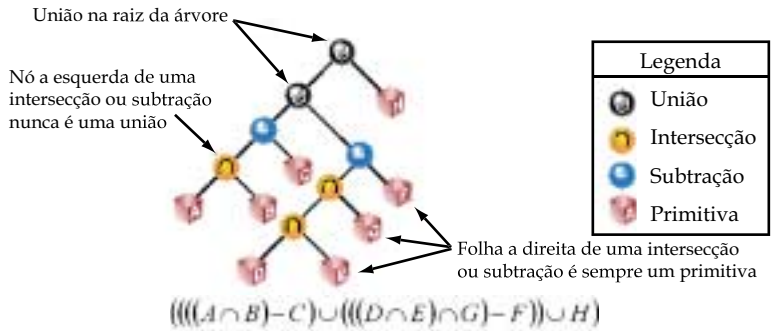
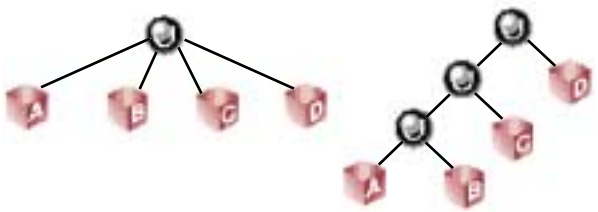


Figura 3: Exemplo de árvore CSG.

A construção de uma árvore CSG se dá em função do processo de composição do respectivo sólido: quando uma operação booleana é realizada, ela resulta em um novo sólido cuja árvore CSG terá um nó referente à operação feita como raiz e as árvores dos sólidos operandos como ramos partindo da mesma, sendo a árvore de uma primitiva composta por uma folha apenas. A ordem em que os ramos vindos de um nó (que correspondem aos operandos de uma operação booleana) estão dispostos deve ser considerada na interpretação de uma árvore CSG, pois há operações não comutativas, e portanto, a inversão de seus operandos fará com que os resultados se alterem. Árvores CSG costumam ser binárias, ou seja, suportam operações booleanas com dois operandos apenas. Existem, porém, implementações cujas árvores não o são. O que faz pouca diferença, já que árvores não-binárias podem ser facilmente convertidas em árvores binárias, quebrando-se nós com n filhos em nós de dois apenas dispondo da mesma operação e em uma seqüência hierárquica, como é feito na Figura 4.

Pode-se ter nós representando transformações lineares (rotação, translação e mudanças de escala) utilizadas para mover primitivas a partir de sua posição inicial ou sólidos relativos a subárvores a partir da posição obtida por meio das operações booleanas utilizadas para compô-lo. Entretanto, tal artifício se faz desnecessário, já que tais transformações podem ser associadas implicitamente aos nós modificados. E como tais operações são bastante freqüentes, uma árvore pode tornar-se bastante confusa pelo uso de tais nós, perdendo seu poder expressivo. Portanto, seu uso é considerado desaconselhável.

A geometria construtiva de sólidos precisa fazer uso de uma outra representação, tida como sua representação interna, para que sólidos assim representados possam ser renderizados. Dados a serem utilizados de acordo com alguma metodologia de renderização aplicável à representação interna serão mantidos na árvore CSG, sendo esta processada quando a renderização do sólido correspondente é requerida. Um nó interno deve manter uma referência para a operação booleana



(a) (b)

Figura 4: Transformação de uma árvore não binária em binária. (a) Árvore CSG não binária. (b) Árvore de (a) transformada em binária.

a qual se refere, e um nó folha deve manter dados relativos a uma primitiva. O processamento é feito pelo uso da busca em profundidade, uma vez que as operações passam a ser percorridas na ordem de aplicação das mesmas. Dessa forma, dados relativos às primitivas serão obtidos nos nós folhas, e combinações serão realizadas nos nós internos. Freqüentemente, a representação interna utilizada pela geometria construtiva de sólido é definida pelos semi-espaços (half-spaces). Semi-espaços são estruturas definidas por funções $f(x, y, z) = 0$ que dividem o espaço em duas partes. Primitivas são compostas por eles, correspondendo a espaços finitos delimitado pelos mesmos. Para criarmos um cilindro, por exemplo, bastaria a composição de um cilindro circular reto e dois planos (Figura 5).

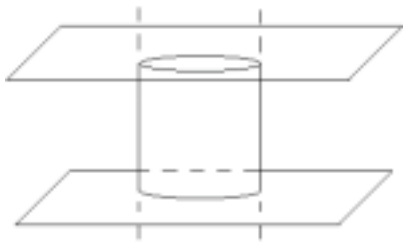


Figura 5: Cilindro composto por semi-espaços.

Sólidos representados por CSG tendo semi-espaços como representação interna têm sua visualização processada sem que suas primitivas sejam explicitamente combinadas. Tais sólidos são renderizados pelo uso de *ray tracing* de uma forma especialmente concebida para tal: a árvore CSG é processada para cada raio, obtendo-se pontos de interseção para cada primitiva nos nós folhas e selecionando-se os mesmos nos nós internos conforme a operação correspondente [4].

Ao processarmos uma árvore CSG, as primitivas em representação interna são obtidas quando nós folhas são visitados. Os nós internos podem manter a estrutura da primitiva correspondente em representação interna, ou apenas manter informações a serem utilizadas na construção da mesma, dependendo de qual seja ela. Um elipsóide, por exemplo, poderia ser definido em função de seus raios nos eixos 'x', 'y' e 'z' e de sua posição no espaço. Esses valores seriam então mantidos nos nós folhas cujas primitivas fossem do tipo "elipsóide". Tal forma de representar primitivas, caso utilizada, fará da árvore CSG uma forma bastante compacta de representar objetos. Uma árvore CSG define não somente a visualização de um sólido, mas também seu histórico de modelagem. Ela é um registro de todas as operações realizadas para a criação do sólido. Assim, pode-se retornar a uma etapa anterior da modelagem facilmente, alterando primitivas ou

operações. Basta que se faça acesso a um nó e se modifique o parâmetro desejado. Tal recurso pode ser utilizado de forma bastante produtiva por softwares de modelagem 3D, de forma que o usuário possa acessar nós de uma árvore relativa a um sólido, modificando seus parâmetros e visualizando o sólido modificado em seguida. A visualização do sólido deve ser reprocessada depois de efetuada a alteração. A fim de ilustrar a modificação de parâmetros de árvores CSG, tomemos como exemplo a esfera com um buraco ilustrada na Figura 6(a), resultante da diferença de uma esfera por um cilindro. Pode-se aumentar o diâmetro do buraco acessando o nó folha relativo ao cilindro e modificando o raio do mesmo. Pode-se modificar também a cor interna do buraco modificando a cor do cilindro. Um possível resultado para tais alterações pode ser visto na Figura 6(b). A operação aplicada no cilindro e na esfera também pode ser modificada. Basta que se acesse o nó interno relativo à diferença aplicada às primitivas e se modifique a respectiva operação. A Figura 6(c) ilustra a alteração da operação utilizada para a união.

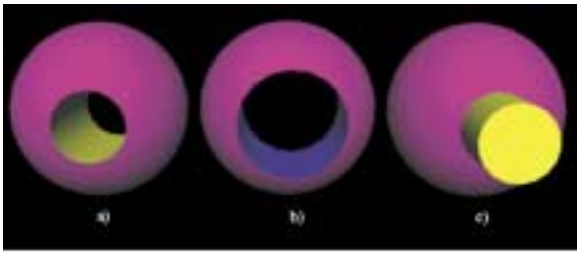


Figura 6: Modificação dos parâmetros de construção de um sólido. (a) Sólido resultante da diferença de uma esfera por um cilindro. (b) Sólido de (a) tendo raio e cor do cilindro modificados. (c) Sólido de (a) tendo a operação modificada para união.

A geometria construtiva de sólidos, devido a sua flexibilidade e à forma compacta com que representa sólidos, tornou-se um modelo de representação bastante popular para se lidar com operações booleanas. Porém, em geral não é utilizada da forma clássica, e sim adaptada para uma estrutura que suporte outros tipos de operações [5].

4 Células

4.1 Octrees

A Octree é um esquema de representação de sólidos 3D baseada no princípio da subdivisão recursiva. Ela geralmente referenciada como octree, octtree, oct-tree ou octal-tree. As octrees caracterizam-se através de uma estrutura de dados hierárquica, espacialmente endereçável e naturalmente pré-ordenada. A simplificação de operações tais como a detecção de interseção de objetos, localização de um ponto ou de um bloco no espaço, remoção de superfícies escondidas são enormemente facilitadas via manipulação das octrees.

Os modelos baseados em octrees consideram que o sólido está inserido em um cubo de dimensões $2^n \times 2^n \times 2^n$, chamado de cubo universo, universo, espaço octree ou mundo. A representação se dá pela subdivisão recursiva do espaço cúbico em octantes (ou voxels), com cada voxel de dimensão $2^d \times 2^d \times 2^d$ onde $1 <= d <= n$, sendo subdividido em oito voxels com dimensões $2^{d-1} \times 2^{d-1} \times 2^{d-1}$, tal como ilustrado na Figura 7. Um critério é definido para que o processo de subdivisão de cada voxel continue ou seja interrompido. Cada voxel recém subdividido é testado de forma a se saber em qual dos três seguintes casos ele ocorre:

- O voxel está totalmente contido no sólido.

- O voxel está completamente fora do sólido.
- O voxel está parcialmente contido no sólido

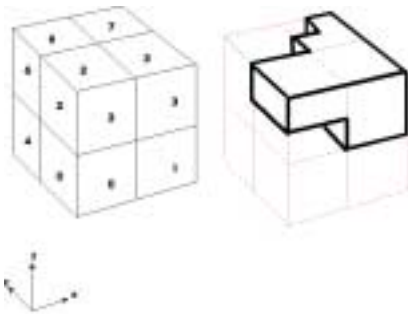


Figura 7: Exemplo de um sólido representado em uma octree.

Nas duas primeira situações, o voxel é rotulado como *full* (cheio) ou *empty* (vazio), conforme o caso, e o processo de subdivisão é interrompido. Na terceira situação o voxel é rotulado como *partial* (parcial), e o processo recursivo prossegue, gerando seus oito filhos. O processo é interrompido quando árvore atinge um nível de profundidade pré-estabelecido (*depth*), sendo esse portanto o nível máximo de subdivisões permitido. Nesse nível, os voxels são chamados de voxels mínimos, e nesse caso, são rotulados arbitrariamente.

Essa representação pode ser generalizada para modelagem de objetos n-dimensionais, através do emprego de uma árvore binária n-dimensional (ou uma árvore (2^n)-ária. Para objetos 3D a representação é realizada por uma árvore 8-ária ou "octree", conforme o exemplo ilustrado pela Figura 8.

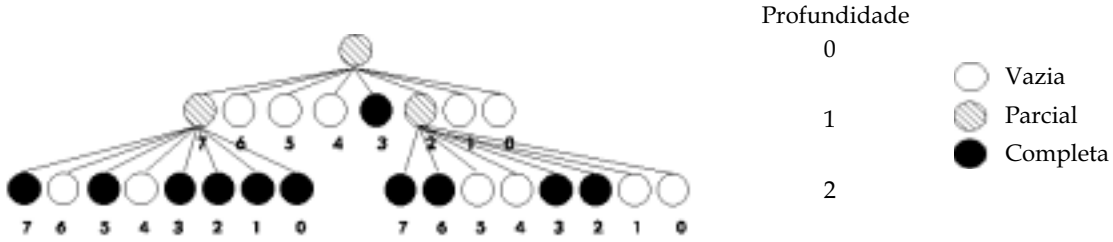


Figura 8: Representação de uma árvore 8-ária de profundidade 2.

Define-se por octree completa, aquela cuja a árvore associada possui todas as suas folhas no mesmo nível. Uma octree completa de profundidade N é dividida em 2^{3N} voxels.

As octrees apresentam diversas propriedades, entre as quais podemos citar:

- Poder de Expressão - Qualquer objeto pode ser representado, com precisão limitada.
- Validade - Pode gerar sólidos *non-manifold*, isto é, pode gerar desconectividade. Nem todo sólido é válido.
- Ambigüidade e Unicidade - Não é ambígua e é única.

- Linguagem de Descrição - Não possui uma linguagem própria. Geralmente é gerada a partir de um objeto modelado em outra representação.
- Concisão - Consome muita memória, mas normalmente menos que a enumeração exaustiva e *cell decomposition*.
- Operações Fechadas - Se a árvore original é válida, garante-se a manutenção da validade.
- Aplicabilidade e Custo Computacional - O algoritmo em si é barato, porém pode haver muitos voxels, o que torna o custo computacional sai caro.

4.2 Quadtrees

Para o armazenamento de objetos 2D ou 3D com espessura constante, usa-se as Quadtrees. Essas estruturas de armazenamento dividem o plano onde está o objeto em quatro partes iguais e classificam cada parte de forma semelhante às Octrees (Figura 9)

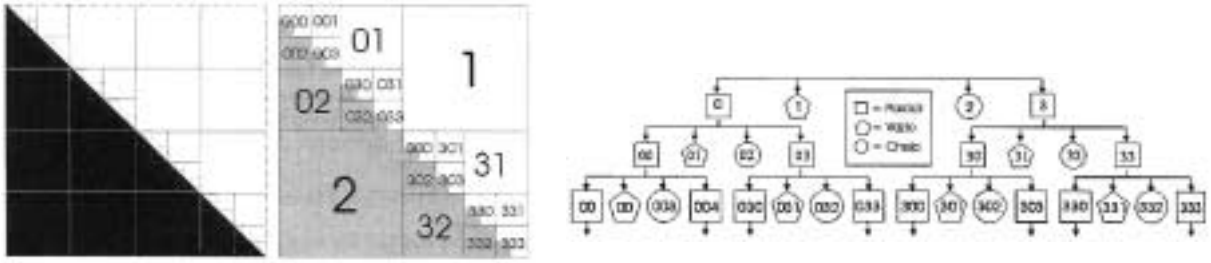


Figura 9: Representação de um sólido por Quadtrees.

Essa representação tem as mesmas vantagens e desvantagens das Octrees, além de permitirem uma representação mais detalhada que as Octrees, com um gasto menor de memória.

4.3 BSP-trees

As estruturas de árvores BSP, que representam uma partição binária do espaço, foram inicialmente propostas para solucionar problemas de superfícies ocultas em aplicações onde o observador móvel desloca-se por uma cena estática. Um exemplo clássico dessa situação ocorre nos simuladores de vôo e fogos de entretenimento [6].

As árvores BSP dividem o espaço em pares de subespaços, cada um separado do outro por planos de orientação e posição arbitrária (Figura 10). Cada nó não terminal da árvore BSP representa um único plano que divide o espaço ocupado em dois. Um nó terminal representa uma região que não será mais subdividida e contém ponteiros para representações da estrutura de dados dos objetos cruzando aquela região. Cada nó interno da árvore BSP é associado a um plano e tem dois filhos, um para cada lado do plano. Assumindo que as normais da Figura 10 apontam para fora do objeto, o filho da esquerda estará dentro do plano, enquanto o da direita estará fora do plano.

Como as octrees e as quadtrees, essa representação também pode ser uma forma linear de descrição dos objetos representados [7].

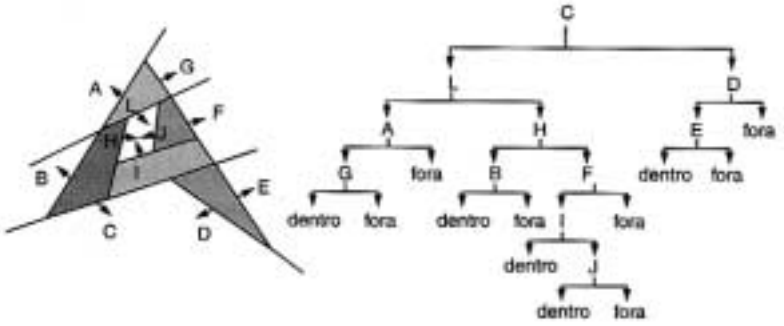


Figura 10: Uma figura 2D e sua representação em árvore binária.

5 Conversões entre representações

Nas seções anteriores foi possível observar que nenhuma das representações apresentadas têm propriedades dominantes sobre os outras. A habilidade de converter representações de um esquema para outro é de grande importância prática a fim de construir sistemas bem projetados de modelagem. A homogeneização de representações em esquemas híbridos é um caso particular de conversão de representações.

Podemos classificar as conversões como consistentes (teórica exata) ou aproximadas. Uma conversão consistente reproduz a representação do sólido original, enquanto a conversão aproximada reproduz a representação de um sólido que se aproxima ao original. Conversões consistentes entre representações não ambíguas são teoricamente irreversíveis[3].

5.1 CSG para B-rep

A conversão CSG para B-rep está restrita às varreduras de áreas bidimensionais limitadas por linhas e círculos [8]. Tal domínio é bastante limitado para aplicações práticas. Desta forma, para conseguir uma representação dual consistente, os sólidos devem ser especificados nos termos que possam ser automaticamente convertidos à representação CSG. A representação B-rep é derivada algoritmicamente. Infelizmente, a especificação torna-se tediosa quando as construções complexas, envolvendo tangências e dimensões, devem ser convertidas pelo usuário em parâmetros preliminares de CSG (posições e dimensões dos primitivos) [9].

5.2 B-rep para CSG

Pode-se utilizar a renomada representação B-rep como representação interna de sólidos CSG, o que é uma abordagem bastante robusta e flexível para a representação de sólidos compostos pelo uso das operações booleanas. Quando se faz uso dela, o sólido B-rep correspondente a uma árvore CSG é obtido quando sua renderização é requerida. Ele é construído a partir da combinação direta das primitivas conforme as operações realizadas. Para tal, a árvore será percorrida fazendo-se busca em profundidade, de forma que primitivas em B-rep sejam criadas quando nós folhas forem visitados, e os sólidos obtidos a partir dos nós descendentes combinados (conforme a operação booleana correspondente) quando nós internos forem visitados [10].

Para que se possa combinar sólidos B-rep por meio de operações booleanas, é necessário que se disponha de um algoritmo para tal fim. Existem várias abordagens para o problema, sendo uma delas descrita por Laidlaw [11], cujo algoritmo opera em dois sólidos simultaneamente. Em

um primeiro momento, ele subdivide as faces dos dois sólidos de modo que não haja interseção de faces entre eles. Em seguida, as faces de um sólido são classificadas com base na superfície do outro como estando dentro, fora ou na fronteira do mesmo. Dependendo de como as faces foram classificadas, elas serão ou não utilizadas no novo sólido de acordo com qual foi a operação booleana utilizada. Conversões B-rep para CSG podem levar a ambigüidades e, desta forma, podem se tornar não realizáveis [1, 12].

5.3 Células para B-rep

Decomposição em células são generalizações de triangulações, conforme apresentado na seção 4. Os tetraedros e os triângulos são substituídos por 3-células e 2-células na decomposição em células. Uma célula pode ter um número arbitrário de lados. A decomposição em células é inequívoca, porém ambígua e custosa computacionalmente.

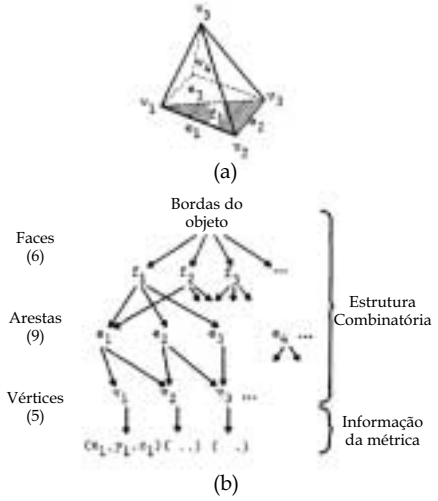


Figura 11: Representação por bordo de uma pirâmide retangular.

Como vimos na seção 2, a representação B-rep caracteriza um sólido através da segmentação de sua borda em um conjunto finito de faces. O esquema ilustrado na Figura 11 é baseado na triangulação por bordo, isto é, a segmentação das fronteiras do objeto em triângulos não sobrepostos. Desta forma, é possível notar que a decomposição em células pode ser convertidas exatamente em representações B-rep por algoritmos simples [13], i.e., que selecionam as células de face que pertencem somente uma célula na decomposição [14].

5.4 CSG para Células

A conversão CSG para células é realizada através do algoritmo de *Marching Cubes*. Este utiliza o algoritmo de dividir e conquistar (*divide-and-conquer*) para localizar a superfície dentro de um cubo lógico criado por oito pixels. Quatro destes pixels são oriundos de duas fatias adjacentes, como ilustra a Figura 12.

O algoritmo determina como a superfície intercepta o cubo e, em seguida se move para o próximo cubo. De modo a encontrar a superfície de intersecção em um cubo, é atribuído o valor 1 a um vértice do cubo se o valor do dado naquele vértice exceder (ou igualar) o valor da superfície que está sendo construída. Assim, estes vértices estão dentro ou na superfície.

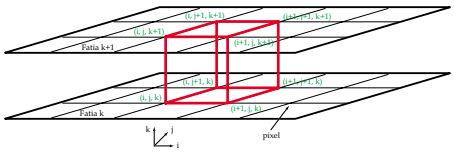


Figura 12: Cubo lógico utilizado pelo algoritmo *Marching cubes.*

A superfície intercepta aquelas arestas do cubo onde um vértice está fora da superfície (valor um) e o outro está dentro da superfície (valor zero). Utilizando esta afirmação, é determinada a topologia da superfície dentro do cubo, encontrando a localização da intersecção depois.

Como existem oito vértices em cada cubo e dois estados, dentro e fora, existem somente $2^8 = 256$ maneiras que a superfície pode interceptar o cubo. Pela enumeração destes 256 casos, criou-se uma tabela de intersecções de superfícies, dado a labelização dos vértices dos cubos. Esta tabela contém as bordas de intersecção para cada caso.

A triangulação dos 256 casos é possível mas tediosa e suscetível a erros. Entretanto, duas simetrias diferentes do cubo reduz o problema de 256 casos para 14 padrões. Primeiro, a topologia da superfície triangular não é modificada se a relação entre o valor da superfície e o cubo forem invertidos. Casos complementares, onde vértices maiores que o valor da superfície precisam ser considerados, são equivalentes. Portanto, somente casos com zero a quatro vértices maiores que o valor da superfície são considerados, reduzindo o número de casos para 128. Utilizando a segunda propriedade de simetria, simetria rotacional, o problema é reduzido para 14 padrões por inspeção.

O passo final do algoritmo *marching cubes* calcula a normal para cada vértice do triângulo utilizando diferenças centrais. Por fim, ele interpola a normal para cada vértice do triângulo. A potencialidade de modelagem de sólidos deste algoritmo está na operação booleana que permite cortar e tampar os modelos de sólidos, assim como a extração de múltiplas superfícies [15].

A utilização do algoritmo *marching cubes* possui a desvantagem de ser custoso computacionalmente, possui muitas ambigüidades e há perda na resolução devido a amostragem do espaço.

6 Conclusão

Esta monografia apresentou três tipos de modelagem de sólidos bem como as conversões entre os métodos. Inicialmente foi apresentado os níveis de abstração em modelagem; em seguida os sólidos construtíveis e não construtíveis. Foram abordados três tipos de representação de sólidos: representação por fronteiras (B-rep), por geometria construtiva de sólidos e por enumeração do espaço em células. Por fim, foram apresentadas algumas conversões entre representações.

Foi observado que as conversões de CSG para células e células para B-rep são simples, sendo a última realizada pelo algoritmo *marching cubes*. Entretanto, a conversão de B-rep para CSG alem de ser mais complexa, apresenta uma perda de informação do espaço e, portanto, não é muito utilizada.

Referências

[1] Mäntylä, M., 1988. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, 1988.

[2] Azevedo, E. e Conci, A., 2003, *Computação Gráfica - Teoria e Prática*, Rio de Janeiro, Elsevier, ISBN: 85-352-1253-3

[3] Dreux, M. e Bortolossi, H., 1995, *Apostila de Modelagem Geométrica*, Departamento de Informática, Curso de Computação Gráfica Aplicada, Pontifícia Universidade Católica do Rio de Janeiro, pp. 7-9.

[4] Foley, J. et al., 1997. *Computer graphics: principles and practice*. 2^a edição em C, Addison-Wesley, ISBN 0-201-84840-6, Inglaterra.

[5] Castanheira, D., 2003, *Geometria construtiva de sólidos combinada à representação b-rep*, Trabalho de Graduação, Universidade de Brasília, Instituto de ciências exatas.

[6] Watt A. e Policarpo, F., 2003. *3D Game: Animation and Advantages Real-time Rendering*. Volume II, Addison-Wesley, ISBN 0-201-84840-6, Inglaterra.

[7] Ferraz, I. N., 2003. *Programação com Arquivos*. Manole, Barueri.

[8] Vossler, D.L., 1985, *Sweep-to-CSG conversion using pattern recognition techniques*, IEEE Computer Graphics and Applications, vol. 5, no. 8, pp. 61-68.

[9] Rossignac, J.R., 1986, *Constraints in Constructive Solid Geometry*, Workshop on interactive 3D graphics, Chapel Hill, NC, pp. 93-110.

[10] Keyser, J., Krishnan, S. e Manocha, D., 1999, *Efficient and Accurate B-rep Generation of Low Degree Sculptured Solids Using Exact Arithmetic: I - Representations*, Preprint submitted to Elsevier preprint.

[11] Laidlaw D. H.; Trumbore W. B.; Hughes J. F., 1986, *Constructive solid geometry for polyhedral objects*. SIGGRAPH Proceedings, p.161.

[12] Schlechtendahl, E. G. (editor), 1989, *CAD Data Transfer for Solid Models*, ESPRIT Research Reports, Project 322, CAD Interfaces, Volume 3, Springer-Verlag.

[13] Agoston, M.K., 1976, *Algebraic Topology*, Marcel Dekker, New York.

[14] Requicha, A.A.G., 1980, *Representations for Rigid Solid: Theory, Methods, and Systems*, Computing Surveys, Vol. 12, No. 4.

[15] Lorensen, W.E., 1987, *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, Computer Graphics, Vol. 21, No. 4, pp. 163-169.