MOD - Um Pequeno Modelador de Solidos B-rep[1]

André Luiz Pires Guedes
Milton Ramos Ramirez
Laboratório de Matemática Aplicada, IM, UFRJ
Caixa Postal 68530 - CEP 21945 Rio de Janeiro, RJ

Ronaldo Cesar Marinho Persiano[2]
Laboratório de Computação Gráfica, COPPE, UFRJ
Caixa Postal 68511 - CEP 21945 Rio de Janeiro, RJ

RESUMO - Estudo sobre modelagem de sólidos com base na representação de "Boundary Representation" (B-rep). Geramos um modelador B-rep, bem simplificado, que serve como exemplo de utilização das técnicas abordadas no estudo.

1. INTRODUÇÃO

O objetivo deste trabalho é realizar uma experiência inicial na área de modelagem de objetos geométricos, que em Computação Gráfica é mais conhecida como modelagem de sólidos. Esta área engloba um corpo de teorias, técnicas e sistemas computacionais voltados para a representação das relações geométricas que permitem uma definição precisa, e *sem ambiguidade*, de um sólido fisicamente.

Um modelador baseado em B-rep (*Boundary representation*) representa o sólido por sua fronteira, ou seja, a sua superfície. Os sólidos poliédricos por exemplo, são representados pelo conjunto de suas faces, arestas e vértices. Para se manipular estes conjuntos de modo eficiente, usamos como ferramenta matemática a **Topologia Algébrica**, e vemos um sólido como uma Álgebra de Arestas, que representa as relações de adjacência entre as entidades geométricas do sólido. A Álgebra de Arestas é representada por uma estrutura de dados que é

---

[1] Projeto Final do Curso de Graduação em Informática da UFRJ

[2] Orientador.

criada e modificada por operadores topológicos básicos.

O modelador implementado pode representar sólidos poliédricos como prismas, pirâmides, sólidos de revolução e toros. Este trabalho está em uma fase inicial, e foi gerado apenas como um exemplo das possíveis aplicações da teoria estudada.

2. A MATEMÁTICA DO B-REP

A representação B-rep de um sólido utiliza os conjuntos de faces, arestas e vértices do mesmo e as informações de adjacência entre os elementos destes conjuntos. A teoria matemática que opera com estes elementos é Topologia, que é mais natural que Geometria neste tipo de problema ([ZEEMAN]).

As superfícies com as quais o B-rep manipula possuem três propriedades de fundamental importância. Informalmente falando, são superfícies bi-dimensionais compactas, que são superfícies conexas, fechadas e trianguláveis.

Para a manipulação dos elementos de uma subdivisão, utilizamos uma Álgebra de Arestas, que trata de todo o sólido através das arestas deste. Esta álgebra de arestas é uma álgebra abstrata (A,A*,ProximaOrg,Rot,Flip), onde A e A* são conjuntos arbitrários finitos, e ProximaOrg, Rot e Flip são funções de arestas que se referem a estes conjuntos. A relação de algumas das funções de arestas está abaixo:

Origem (e) → Anel de arestas em torno da origem.

Destino (e) → Anel de arestas em torno do destino.

Esq (e) → Anel de arestas da face esquerda.

Dir (e) → Anel de arestas da face direita.

Flip (e) → Versão da aresta com orientação contrária, do outro lado da superfície.

Simetrica (e) → Versão da aresta com sentido contrário, do destino para a origem.

ProximaOrg (e) → Próxima aresta com mesma origem que e, percorrendo-se no sentido trigonométrico em torno do vértice origem de e segundo a sua orientação.

ProximaDes (e) → Idem ProximaOrg, em volta do destino de e.

ProximaEsq (e) → Próxima aresta com mesma face esquerda.

ProximaDir (e) → Idem ProximaEsq, em volta da face direita.

Dual (e) → Leva uma aresta na sua Dual.

Rot (e) → Usada no lugar de Dual em alguns casos.

Estas funções de arestas possuem algumas propriedades que são usadas para testar se uma dada estrutura de dados é consistente ou não ([GUIBAS]).

É importante observar que qualquer função definida, com exceção do Flip, pode ser expressa como uma composição de um número finito de operações Rots e ProximaOrgs, independente do tamanho ou complexidade da subdivisão. A Figura 2.1 resume todas as funções de arestas apresentadas até agora,com exceção do Flip(e).

Podemos verificar que a topologia de uma subdivisão é completamente determinada pela sua álgebra de arestas, e vice-versa, e consequentemente o sólido passa a ser a álgebra e vice-versa.

3. ESTRUTURA DE DADOS

Uma estrutura de natural implementação computacional da álgebra de aresta é a chamada estrutura de dados 4-Aresta. Esta estrutura representa uma subdivisão P simultaneamente com a sua subdivisão dual P^*, existindo assim oito versões de cada aresta, consistindo das quatro versões orientadas e direcionadas de uma aresta não direcionada de P, mais as quatro versões do seu dual.

A estrutura coloca todas as oito versões em um unico registro juntando cada aresta e com a sua versão Flip(e) num mesmo campo do registro da estrutura, obtendo 4 campos-aresta por registro. Os quatro campos do registro de arestas E podem ser referenciados por E[r], com r variando de 0 até 3, e correspondem as arestas $Rot^r(e)$, como mostra a Figura 3.1. A referencia a versão Flip da aresta é feita com um bit de informação, que representa o f.

Cada campo de um registro aresta contém dois sub-campos, **Dados** e **Proxima**. O campo de Dados é usado para guardar possíveis informações não topológicas, que pode ser, ou não, afetado pelas operações topológicas, e o seu conteúdo e formato é completamente dependente da aplicação. O campo Proxima contém

uma referência para a aresta:

ProximaOrg (e).

A Figura 3.2 ilustra uma pedaço de uma subdivisão e sua estrutura 4-Arestas correspondente. Nós podemos imaginar cada registro como sendo pertencente a quatro listas circulares que correspondem às duas faces e aos dois vértices incidentes a aresta.

A estrutura 4-arestas não contém registros separados para as faces e os vértices, sendo estes implicitamente definidos pelos seus anéis de arestas correspondentes.

Em muitas aplicações práticas todas as superfícies manipuladas são orientáveis, isto significa que nós podemos atribuir uma orientação para cada aresta, vértice e face da subdivisão de tal maneira que qualquer dois elementos incidentes possuam orientações compatíveis, acabando com a necessidade do bit de Flip f das arestas.

Nós podemos também representar uma subdivisão sem o seu dual, simplificando ainda mais a nossa álgebra de arestas, que fica somente com as funções ProximaOrg(e) e Simetrica(e) como primitivas - (**A,ProximaOrg,Simetrica**) - e, consequentemente, nossa estrutura de dados passa de uma 4-aresta para uma 2-aresta, que só possui os campos referentes a subdivisão primal. Estas simplificações são utilizadas no MOD.

4. OPERADORES TOPOLÓGICOS

Para finalizar a teoria básica do B-rep vamos apresentar os operadores topológicos necessários para se criar e alterar a estrutura de dados 4-Aresta, atentando para fato de que estes operadores devem levar de uma álgebra de aresta para outra de modo consistente com as propriedades das funções de aresta.

Uma das vantagens de se usar a estrutura 4-Aresta é que necessitamos de um pequeno número de operadores topológicos básicos para criar e alterar sólidos, em relação a outras propostas, tais como os operadores topológicos de Euler ([MANTYLA]).

Usaremos um conjunto de 3 operadores, o **CriaAresta**, o **MataAresta** e o **MudaTudo**. O primeiro cria um registro aresta e retorna uma de suas aresta orientada e direcionada. Seu efeito é

descrito por:

e <-- CriaAresta

(P.1) Esq(e)' = Dir(e)

(P.2) Origem(e) ≠ Destino(e)

(P.3) ProximaEsq(e) = ProximaDir(e) = Simetrica(e)

(P.4) ProximaOrg(e) = AnteriorOrg(e) = e

O operador inverso, o MataAresta(e), retira da estrutura a aresta e, não orientada nem direcionada.

O terceiro operador, MudaTudo(a,b), é o de alteração da estrutura, que recebe como parâmetros duas arestas, e não retorna nada. Este operador afeta os dois anéis de arestas, Origem(a) e Origem(b), e independentemente, os dois anéis de arestas Esq(a) e Esq(b). Ele é seu própio inverso e atua de maneira diferente, dependendo de como as arestas a e b estejam uma em relação a outra na estrutura topológica, isto é:

(M.1) Se os dois anéis são diferentes, o MudaTudo irá juntá-los em um único anel.

(M.2) Se os anéis são o mesmo, o operador irá separá-los em dois pedaços distintos.

(M.3) Se os dois são o mesmo anel com orientações diferentes, o operador irá trocar a orientação e inverter a ordem de um segmento do anel.

Resumindo, existem 4 casos distintos onde o operador pode atuar, e o MudaTudo leva do caso 1 para o 4, do 2 para o 3, e vice-versa:

Origem \ Esq	≠	=
≠	caso 1	caso 3
=	caso 2	caso 4

O efeito do MudaTudo pode ser descrito por modificações nas ProximaOrgs referentes as arestas a e b. Sejam:

a' = Rot(ProximaOrg(a)) e b' = Rot(ProximaOrg(b)),

então as modificações são:

(1) trocar ProximaOrg(a) com ProximaOrg(b)
(2) trocar ProximaOrg(a') com ProximaOrg(b')
(3) trocar ProximaOrg(Flip(ProximaOrg(a))) com ProximaOrg(Flip(ProximaOrg(b))), e
(4) trocar ProximaOrg(Flip(ProximaOrg(a'))) com ProximaOrg(Flip(ProximaOrg(b'))).

Note que todas estas alterações devem ser feitas ao mesmo tempo.

5. EXEMPLO DE MODELADOR

Apresentaremos aqui a implementação do MOD, que tem como objetivo estudar a ação dos operadores topológicos sobre uma estrutura de dados. Portanto o MOD não é um sistema robusto, nem tão pouco completo, mas que atende com certa folga os nossos objetivos. O MOD pode representar sólidos simples, quais sejam, prismas e pirâmides de base poligonal, sólidos de revolução com geratrizes lineares por parte, e toros, cuja seção reta é também um polígono.

Direcionamos a nossa implementação à topologia, deixando a geometria um pouco de lado, já que nossas arestas seriam retas e nossas faces planos. Com estas simplificações na geometria, a nossa estrutura de dados não contém outros dados geométricos além das coordenadas dos vértices.

A parte central, e mais importante, do MOD está dividida em 3 módulos. O dois primeiros são referentes à topologia do sólido, e são: o **Núcleo**, onde estão a estrutura de dados e os operadores topológicos básicos, e a **Base**, com os percursos e os operadores topológicos gerais. Esta separação em dois módulos foi feita para conseguirmos uma independência entre os operadores gerais e a estrutura de dados.

O terceiro é um módulo de transição, que liga a topologia com a geometria, contendo operadores mistos, que se utilizam de operadores topológicos e de transformações geométricas, com a finalidade de criar os sólidos já citados, estes são os **Aplicativos**. Neste módulo está a base geométrica do modelador,

que se restringe às coordenadas dos vértices.

A parte relativa a computação gráfica interativa está dividida em dois outros módulos, o de **Visualização**, que contém basicamente, procedimentos de visualização do sólido tridimensional, e que se utiliza dos módulos anteriores, e o **Localizador**, que é independente dos outros módulos. É o **Localizador** que torna mais simples a interação com o usuário, utilizando-se de mouse ou teclado, conforme o ambiente de trabalho (Figura 5.1).

Utilizamos no MOD as simplificações mencionadas na seção 3, mudando a estrutura de dados para uma 2-arestas. Com estas simplificações, as funções Dual, Rot e Flip não são necessárias, e as funções de arestas com as quais precisamos trabalhar são:

Simetrica
ProximaOrg **AnteriorOrg**
ProximaDir **ProximaEsq**

Em particular, podemos escrever todas as funções em termos da Simetrica e da ProximaOrg.

$$\begin{aligned}
\text{ProximaDes}(e) &= \text{Simetrica}(\text{ProximaOrg}(\text{Simetrica}(e))) \\
\text{ProximaEsq}(e) &= \text{ProximaOrg}^{-1}(\text{Simetrica}(e)) \\
\text{ProximaDir}(e) &= \text{Simetrica}(\text{ProximaOrg}^{-1}(e)) \\
\text{AnteriorOrg}(e) &= \text{ProximaOrg}^{-1}(e) \\
\text{AnteriorDes}(e) &= \text{Simetrica}(\text{ProximaOrg}^{-1}(\text{Simetrica}(e))) \\
\text{AnteriorEsq}(e) &= \text{Simetrica}(\text{ProximaOrg}(e)) \\
\text{AnteriorDir}(e) &= \text{ProximaOrg}(\text{Simetrica}(e))
\end{aligned}$$

Assim, a nossa estrutura fica representada por pares de arestas simétricas, ou seja, uma Simetrica da outra, cada uma apontando para a sua ProximaOrg (Figura 5.2).

Implementamos os três operadores básicos: CriaAresta, MataAresta e MudaTudo. A implementação destes operadores sobre a nossa estrutura ficou bastante simples. O primeiro cria um par de arestas simétricas com origens nos vertices dados, o campo Proxima de cada uma aponta para ela própria.

CriaAresta $(v_1, v_2) \rightarrow a$

Cria arestas a e b
a.Proxima ← a
a.Origem ← v_1
b.Proxima ← b
b.Origem ← v_2
Retorna a

O operador MudaTudo, de resultados complexos, é implementado de maneira trivial. Este operador se resume a uma troca dos campos Proxima das arestas, já que neste caso, basta que os valores das funções ProximaOrg sejam trocados.

```
MudaTudo (a,b)
    Troca (a.Proxima,b.Proxima)
```

Operadores gerais foram implementados fazendo uso dos operadores básicos. Desta forma, não ficam dependentes da estrutura de dados. Os operadores implementados são:

Face (n,Vertices[]) → cria uma face a partir do vetor Vertices[] com n ponteiros para vértices.

PrismaTopologico (a,b) → liga duas faces, cada uma representada por uma aresta. Liga face direita de a com a face esquerda de b.

ColaFaces (a,b) → cola duas faces, direita de a com a face esquerda de b.

No módulo Aplicativos, fazemos a construção de quatro tipos de sólidos: Prismas, Pirâmides, Sólidos de Revolução e Toros. Cada um destes sólidos é construído com os operadores topológicos gerais e com transformações geométricas.

BIBLIOGRAFIA

[**HILLYARD**], R. ; The Build Group of Solid Modelers. IEEE Computer Graphics & Aplications, Marc,o 1982.

[**MANTYLA**], M. & **SULONEN,** R. ; GWB: A Solid Modeler with Euler Operators. IEEE Computer Graphics & Aplications, Setembro 1982.

[**REQUICHA**], A. A. G. & **VOELCKER,** H. P. ; Solid Modeling: Current Status and Research Direction. IEEE Computer Graphics & Aplications, Outubro 1983.

[**GUIBAS**], L. & **STOLFI,** J. ; Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. ACM Transactions on Graphics, Vol. 4, No. 2, Abril 1985.

[**ZEEMAN**], E. C. ; Uma Introdução informal à Topologia das Superfícies. IMPA/CNPq.

[**FOLEY**], James D. & **VAN DAM,** Andries ; Fundamentals of Interactive Computer Graphics. Addison-Wesley, 1984.

[**BERGER**], Marc ; Computer Graphics with Pascal. Benjamin/Cummings, California, 1986.

[**LIMA**], Elon Lages ; Curso de Análise. Projeto Euclides, IMPA/CNPq, 1982.

Turbo Pascal 5.0 ; Manuais de Referencia. Borland 1988.

Mouse4 - Text Mouse Unit version .9, 11/20/87 by Richard Sadowsky released to the public domain. (Fonte de rotinas para uso de mouse).

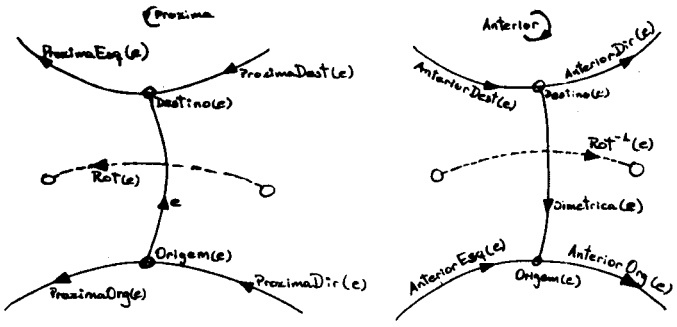

Figura 2.1

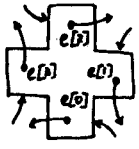

Figura 3.1

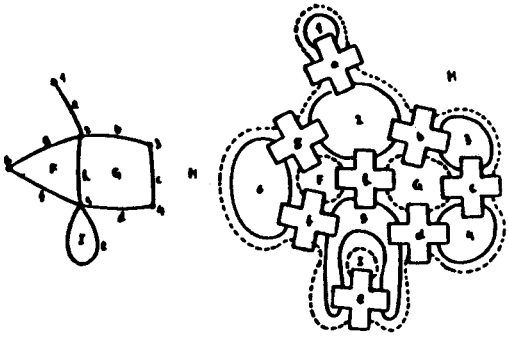

Figura 3.2

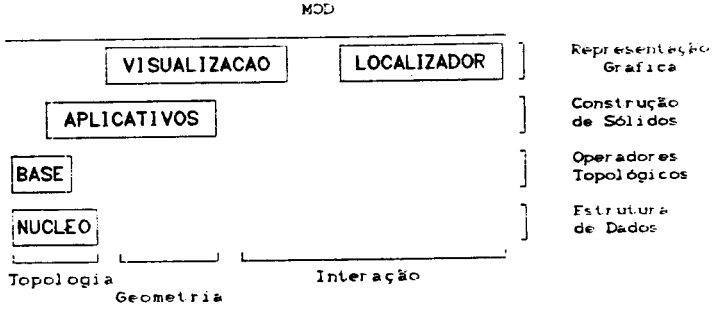

Figura 5.1