Proposta para descrição de malhas de polígonos (MESH)

Sílvia Delgado Olabarriaga

INESC - Projecto CAD/CAM
Av. Alves Redol, 9 - 2º
1000 - Lisboa

Janeiro 1989

INDICE

1. INTRODUÇÃO

Neste relatório apresentamos uma proposta para descrição de uma malha de polígonos designada "mesh". Esta estrutura foi definida para atender aos seguintes requisitos básicos:

a) descrever malhas de polígonos quaisquer, a serem usadas na modelagem de actores e cenário de animação por computador;

b) ser suficientemente genérica para permitir a modelagem de diferentes tipos de objectos, no que se refere à cor e à forma;

c) conter dados suficientes para permitir a utilização de diferentes algoritmos de rendering, nomeadamente shading constante, Gouraud, Phong e ray-tracing.

Para atender a estes requisitos, era necessário que a estrutura projetada permitesse:

a) definição da cor e demais propriedades (como brilho) individualmente às facetas;

b) representação de polígonos quaisquer, planares ou não, com qualquer número de arestas;

c) representação do vector normal à faceta (para shading constante);

d) representação dos vectores normais aos vértices (para Gouraud e Phong shading);

e) cálculo da intersecção de uma faceta com uma linha (para ray-tracing).

Nas seguintes secções, descreve-se esta estrutura, as funções para acesso aos seus elementos básicos, o formato do ficheiro para armazenamento. Por fim, comenta-se a implementação.

2. ESTRUTURA DA MALHA

A malha de polígonos representa a envoltória de objectos tri-dimensionais através de facetas com qualquer número de arestas.

É descrita através de três listas:

- lista de facetas;
- lista de vértices (coordenadas 3D);
- lista de propriedades visuais.

A partir destas três, pode-se opcionalmente criar outra lista, que contém as arestas incidentes a cada vértice da malha.

As facetas (polígonos) são descritas através de uma lista de número variável de vértices associada a um conjunto de características visuais (propriedades como cor e brilho). Os vértices e propriedades visuais podem ser compartilhados por diversas facetas para redução do espaço usado no armazenamento da malha.

A estrutura descrita neste capítulo tem as seguintes características:

a) compacta, pois há compartilhamento de vértices e de propriedades visuais das facetas;

b) flexível, pois não impõe restrições quanto à forma dos polígonos (número variável de arestas) ou do próprio objecto 3D (não precisa ser um sólido);

c) contém pouca informação topológica, o que é compacto, mas resulta na necessidade de percorrer toda a estrutura de dados a cada vez em que se queira obter informações adicionais. Uma solução para este problema é a criação opcional de listas auxiliares, contendo a informação topológica necessária (ver item 2.4);

d) as facetas não precisam ser planares, o que é conveniente durante a modelagem, mas deixa de sê-lo na fase de rendering;

e) facetas com número variável de arestas permitem flexibilidade na definição de formas, mas exigem espaço extra para sua gestão (ver comentários no capítulo 7).

Nas seguintes secções, descreve-se o formato e organização das listas que compõem uma malha de polígonos do tipo "mesh".

2.1 Lista de facetas

Cada nó descreve um polígono fechado da malha através de suas arestas e propriedades visuais. São do tipo "FACENODE", especificado como uma estrutura com os seguintes campos:

a) pro: posição do nó que descreve as características visuais da faceta na lista de propriedades visuais;

b) nedges: número de arestas da faceta;

c) edges: array de vértices que compõem as arestas da faceta, descritos em sentido anti-horário. O último vértice é sempre implicitamente ligado ao primeiro da lista. Os vértices são referenciados através de sua posição na lista de vértices (ver item 2.2);

d) vx, vy, vz: vector normal à faceta (calculado apenas durante a fase de rendering).

Esta lista é manipulada através das seguintes funções: make_face, free_face, get_face, pos_face e length_face, descritas no capítulo 3.

2.2 Lista de vértices

Cada nó contém as coordenadas de um vértice da malha, relativas ao sistema de referência do próprio objecto. O formato é do tipo "VERTEXNODE", que tem os campos descritos a seguir:

a) x, y, z: coordenadas 3D do vértice (números reais).

b) vx, vy, vz: vector normal ao vértice (calculado apenas durante rendering).

Os nós desta lista são referenciados na descrição de facetas através de sua posição (campo "edges" de "FACENODE"). O primeiro nó da lista está na posição "1".

Esta lista é manipulada através das funções: make_vertex, free_vertex, get_vertex, pos_vertex e length_vertex, descritas no capítulo 3.

2.3 Lista de propriedades visuais

Cada nó contém propriedades visuais de uma ou mais facetas da malha. Seu formato corresponde à estrutura "PROPNODE", que tem os seguintes campos:

a) r, g, b: componentes da cor da faceta no sistema RGB, entre 0 e 1;

b) ref: coeficiente de reflectância da faceta, entre 0 e 1;

c) shine: coeficiente de brilho, entre 0 e 1.

Os nós da lista de propriedades visuais são referenciados na descrição de facetas através de sua posição (campo "pro" de "FACENODE"). O primeiro nó está na posição "1".

Esta lista é manipulada pelas seguintes funções: make_prop, free_prop, get_prop, pos_prop e length_prop, descritas no capítulo 3.

2.4 Lista de arestas incidentes

Os nós desta lista contêm as arestas incidentes a cada vértice da malha e têm o formato da estrutura "INCIDENTNODE", com os seguintes campos:

a) nincid: número de arestas incidentes ao vértice;

b) incid: array contendo os vértices que são conectados a este através de arestas. Estes vértices são referenciados através de sua posição na lista descrita em 2.2.

A posição de um nó que descreve determinado vértice é a mesma nesta ou na lista de vértices (ver item 2.2).

Esta lista é construída opcionalmente pela função "make_incident_edges" e manipulada pelas funções get_incid e pos_incid, descritas no capítulo 3.

3. FUNÇÕES DA BIBLIOTECA

A biblioteca de funções que acessam a malha de polígonos têm por objectivo fornecer uma interface entre a aplicação e a estrutura de dados do "mesh". Para

usar estas rotinas, a aplicação deve incluir o ficheiro "mesh.h", que contém definições de tipos e constantes usadas pelas funções de manipulação da malha, explicadas a seguir.

```
int for_all_vertex ( m, p, func )
MESH m;
VERTEX p;
(* int) () func;

int for_all_face ( m, p, func )
MESH m;
FACE p;
(* int) () func;

int for_all_prop ( m, p, func )
MESH m;
PROP p;
(* int) () func;
```

DESCRIÇÃO: Executa a função "func" uma vez para cada nó da respectiva lista, passando como parâmetro o ponteiro "p" e um ponteiro para o nó que deve ser processado. A função deve ter o seguinte cabeçalho:

```
int func ( p, n )
NODE *p;          /* ponteiro qualquer */
NODE *n;          /* ponteiro p/ faceta, vértice, etc */
```

PARAMETROS:

m = identificador da malha;

p = ponteiro para outro parâmetro que se deva passar a função. Usando-se "cast", não é necessário que o parâmetro seja do mesmo tipo especificado acima (NODE), que corresponde a um ponteiro para os nós da lista.;

func = ponteiro para a função que deve ser executada.

RETORNO:

somatório de todos os valores retornados pela função "func" em suas repetidas execuções.

```
free_mesh ( m )
MESH m;
```

DESCRIÇÃO: Remove da memória todas as estruturas de dados pertencentes à malha m (lista de facetas, vértices, propriedades e arestas incidentes, se houver). O identificador da malha não pode ser usado depois disto.

PARAMETRO:

m = identificador da malha a remover.

```
free_vertex ( m, i )
MESH m;
int i;

free_face ( m, i )
MESH m;
int i;

free_prop ( m, i )
MESH m;
int i;
```

DESCRIÇÃO: Remove o i-ésimo nó da respectiva lista. Atenção: a partir de então, as referências às posições dos demais nós da lista ficam inconsistentes!

PARAMETROS:

m = identificador da malha de polígonos;
i = posição do nó na respectiva lista.
Se i < 1, remove primeiro nó
Se i > tamanho da lista, remove último nó

```
free_incidents ( m )
MESH m;
```

DESCRIÇÃO: Remove toda a lista de arestas incidentes aos vértices da malha m.

PARAMETROS:

m = identificador da malha de polígonos.

```
VERTEX get_vertex ( m, i )
MESH m;
int i;

FACE get_face ( m, i )
MESH m;
int i;

PROP get_prop ( m, i )
MESH m;
int i;

INCIDENT get_incid ( m, i )
MESH m;
int i;
```

DESCRIÇÃO: Retorna o ponteiro para o i-ésimo nó da lista de vértices, facetas, propriedades visuais ou arestas incidentes, respectivamente.

PARAMETROS:

m = identificador da malha;
i = posição do nó na lista.

RETORNO:

ponteiro para o nó.
Se for NULL, o i-ésimo nó não existe na lista.

```
int length_vertex ( m )
MESH m;

int length_face ( m )
MESH m;

int length_prop ( m )
MESH m;
```

DESCRIÇÃO: Retorna o número de nós da respectiva lista. Pode ser usada também para obtenção da posição do último nó inserido na lista.

PARAMETRO:

m = identificador da malha.

RETORNO:

tamanho da respectiva lista.

```
int load_mesh ( file, m )
char  *file;
MESH m;
```

DESCRIÇÃO: Carrega uma malha de polígonos na memória a partir do conteúdo de um meta-ficheiro (ver capítulo 6). Os nós das listas de facetas, vértices e propriedades são alocados durante a leitura do meta-ficheiro e anexadas às listas já existentes. A lista de arestas incidentes NAO é criada por esta função.

PARAMETROS:

file = ponteiro para nome do ficheiro a carregar;
m = identificador da malha que deve ser carregada.

RETORNO:

código que indica o tipo de erro ocorrido:

0	= carga OK;
E_OPEN	= não pode abrir ficheiro;
E_FMT	= ficheiro com formato inválido;
E_FACE	= erro durante leitura de uma faceta;
E_VERT	= erro durante leitura de um vértice;
E_PROP	= erro durante leitura de propriedades.

```
MESH make_mesh ()
```

DESCRIÇÃO: Cria estrutura de dados em memória para uma malha de polígonos, simplesmente alocando área para as estruturas de controle das listas que serão posteriormente preenchidas pela aplicação.

RETORNO:

ponteiro para o identificador da malha criada.
Se for NULL, houve erro na criação da estrutura de dados da lista.

```
VERTEX make_vertex ( m )
MESH m;

FACE make_face ( m, nedges )
MESH m;
int  nedges;

PROP make_prop ( m )
MESH m;
```

DESCRIÇÃO: Aloca um nó na respectiva lista de vértices, facetas ou propriedades visuais da malha m. O nó é encadeado ao final da lista e seu endereço retornado à aplicação. No caso de make_face, é necessário indicar, também, o número de arestas da faceta, a fim de que o array de vértices possa ser alocado com

tamanho correto. Os campos "nedges" e "edges", neste caso, são preenchidos pela própria rotina make_face.

PARAMETROS:

m	= identificador da malha;
nedges	= número de arestas da faceta.

RETORNO:

ponteiro para o nó alocado.
Se for NULL, houve erro na alocação.

```
int make_incident_edges ( m )
MESH m;
```

DESCRIÇÃO: Cria uma lista contendo as arestas incidentes a cada um dos vértices da malha m.

PARAMETROS:

m = identificador da malha de polígonos.

RETORNO:

0 = OK;
1 = erro de alocação de memória.

```
int pos_vertex ( m, v )
MESH m;
VERTEX v;

int pos_face ( m, f )
MESH m;
FACE f;

int pos_prop ( m, p )
MESH m;
PROP p;

int pos_incid ( m, i )
MESH m;
INCIDENT i;
```

DESCRIÇÃO: Retornam a posição de um nó na respectiva lista (vértices, facetas, propriedades visuais ou arestas incidentes), dado o seu endereço.

PARAMETROS:

m	= identificador da malha;
v, f, p, i	= ponteiro para nó.

RETORNO:

posição do nó na lista.
Se for 0, o nó não pertence à lista.

```
int save_mesh ( file, m )
char  *file;
MESH m;
```

DESCRIÇÃO: Copia para um meta-ficheiro (ver capítulo 6) o conteúdo de uma malha de polígonos armazenada na memória. O ficheiro, se já existir, é destruído (fopen, opção "w"). A estrutura de dados em memória não é alterada.

PARAMETROS:

file = ponteiro para nome do ficheiro a gravar;
m = identificador da malha que deve ser salva.

RETORNO:

código que indica o tipo de erro ocorrido:

0	= gravação OK;
E_OPEN	= não pode abrir o ficheiro;
E_FACE	= erro durante gravação de uma faceta;
E_VERT	= erro durante gravação de um vértice;
E_PROP	= erro na gravação de propriedades.

4. COMPILAÇÃO E LIGAÇÃO

A aplicação deve incluir o ficheiro "mesh.h" para ter a definição dos tipos VERTEX, FACE, etc., como também para ter acesso a algumas funções implementadas como macros. Para ligar ao objecto, usar o módulo "mesh.o" da seguinte forma:

```
cc aplic.c mesh.o -llp
```

ou

```
cc aplic.c mesh.o -lflp
```

O parâmetro "-llp" ou "-lflp" indica a ligação com a biblioteca de funções para manipulação de listas "listpack" [Zimmermann 85]. A diferença entre as duas opções diz respeito à geração de código adicional para verificação de acesso às listas (lp = com verificação, flp (fast listpack) = sem verificação).

Estes ficheiros estão na diretoria "~silvia/mesh".

5. EXEMPLO DE USO NUMA APLICAÇÃO

É importante salientar alguns pontos importantes que devem ser observados por uma aplicação que utiliza as funções da biblioteca "mesh":

- antes de carregar uma malha com um ficheiro, deve-se criá-la através da função "make_mesh";

- depois de salvar a malha em um ficheiro, a aplicação deve liberar sua área, se esta não for mais usada, pois a função "save_mesh" não faz isto automaticamente;

- a criação da malha através de "make_mesh" não aloca área para os vértices, facetas, etc. Se estes não forem carregados de um ficheiro em disco, é necessário criá-los individualmente através de "make_vertex", "make_face", etc. O preenchimento dos nós alocados (coordenadas, componentes RGB, etc.) é de reponsabilidade da aplicação;

- os vectores normais às facetas e vértices da malha não são calculados pelas rotinas do mesh, mas ocupam área na memória;

- a lista de arestas incidentes aos vértices só é gerada sob comando da aplicação, não ocupando memória antes disto.

O programa-exemplo contido no ficheiro "meshexemplo.c" realiza os seguintes procedimentos, ilustrando o uso de algumas funções de acesso à malha de polígonos:

- cria uma malha na memória;

- preenche esta malha com a descrição de um cubo, criando todos os vértices, facetas e propriedades. O cubo tem tamanho 20 X 20 X 20 e está localizado na origem do sistema de coordenadas (entre 10 e -10). Todas as facetas têm a mesma cor (branco) e características luminosas;

- exibe textualmente o conteúdo das listas montadas;

- grava o meta-ficheiro correspondente em disco;

- cria outra malha;

- carrega nova malha com o meta-ficheiro criado anteriormente;

- exibe a malha em wire-frame. A função "wire_inic" estabelece o ambiente adequado à visualização do cubo (ver [IRIS] para mais pormenores);

- libera a estrutura de dados de ambas as malhas.

Outro programa que pode ser usado como referência de uso do mesh é o "selips", que gera uma malha de polígonos através das equações das super-elipsóides.

6. META-FICHEIRO

Uma malha de polígonos é armazenada num ficheiro do tipo "texto", com caracteres ASCII organizados em linhas separadas por "New-Line". Símbolos especiais são usados para introduzir a descrição de uma faceta, vértice ou propriedade visual. O símbolo especial fica numa linha e a descrição propriamente dita na linha seguinte. Os campos desta descrição devem ser separados por branco e rigorosamente obedecer à ordem especificada a seguir:

- '#': indica que a próxima linha contém a descrição de uma faceta no seguinte formato:

posição do descritor de propriedades visuais na respectiva lista
número de arestas
posição dos vértices da faceta na lista de vértices

- '@': indica que a próxima linha contém a descrição de um vértice da malha, no seguinte formato:

coordenada no eixo x (real)
coordenada no eixo y (real)
coordenada no eixo z (real)

- '!': indica que a próxima linha contém a descrição de propriedades visuais de uma ou mais facetas da malha, no seguinte formato:

componente "RED" da cor
componente "GREEN" da cor
componente "BLUE" da cor
coeficiente de reflectência do material
coeficiente de brilho

- '*': indica que a linha corrente deve ser considerada comentário.

A ordem em que estas linhas aparecem no ficheiro é irrelevante, mas convém que as facetas, os vértices e as propriedades visuais sejam agrupados para facilitar o processamento manual da malha de polígonos.

Quando gerados manualmente, estes ficheiros podem conter ainda comentários após os símbolos "!", "#" e "@", indicando a posição que a propriedade, faceta ou vértice ocupa na respectiva lista.

EXEMPLO

```
* definição de um cubo de 20X20X20, cor branca
* lista de facetas
# faceta 1
1 4 1 3 4 2
# faceta 2
1 4 2 4 5 6
# faceta 3
1 4 6 5 8 7
# faceta 4
1 4 7 8 3 1
# faceta 5
1 4 7 1 2 6
# faceta 6
1 4 3 8 5 4
* lista de vértices
@ vértice 1
-10 10 -10
@ vértice 2
10 10 -10
@ vértice 3
-10 -10 -10
@ vértice 4
10 -10 -10
@ vértice 5
10 -10 10
@ vértice 6
10 10 10
@ vértice 7
-10 10 10
@ vértice 8
-10 -10 10
* lista de propriedades visuais
! propriedade 1
1 1 1 0.5 0.5
```

7. IMPLEMENTAÇÃO

As funções do mesh foram implementadas utilizando o "listpack" [Zimmermann 85], que é um conjunto de procedimentos para gestão de listas na linguagem C. As funções que manipulam a malha de polígonos são descritas individualmente no programa fonte (ficheiro "mesh.c"), em cabeçalhos padronizados segundo as normas do projecto CAD/CAM.

A implementação é bastante simples: cada lista da malha de polígonos (facetas, vértices, propriedades visuais e arestas incidentes) é armazenada num estrutura do tipo LIST, gerida pelo listpack. Deve-se, entretanto, fazer alguns comentários sobre os formato dos nós destas listas, cujos tipos são definidos no ficheiro "mesh.h":

a) vértices (VERTEXNODE): além das coordenadas do vértice, há espaço também para o vector normal à superfície naquele ponto, que é calculado apenas durante a fase de rendering da imagem. Estes campos foram incluídos na estrutura do nó de

forma fixa para evitar que nova lista tivesse que ser criada apenas para contê-los durante a síntese de imagens. Desta maneira, gasta-se mais espaço quando estes vectores não são usados, mas ganha-se tempo e memória na maioria das situações, que envolve exibição com realismo;

b) facetas(FACENODE): o tamanho dos nós que descrevem facetas não é fixo, pois o número de vértices pode variar. Para facilitar a gestão desta lista, utilizou-se a estrutura apresentada na figura 1. O campo "edges" contém o endereço do array com os vértices que formam a faceta. A área para este array é alocada contiguamente aos demais campos do nó, em uma única chamada à função "make_node". Para o utilizador (programador) isto é transparente, pois ele pode fazer o acesso a cada um dos elementos deste array da seguinte forma: **...edges[ind]**

Exemplo:

```
FACE f;
     f = make_face(malha, 5);
     f->edges[0] = vertice1;
     f->edges[4] = vertice5;
```

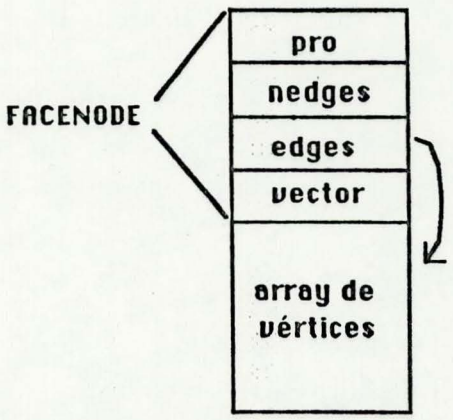

Figura 1 - Nó descritor de uma faceta

c) arestas incidentes (INCIDENTNODE): a estrutura usada para alocação dos campos de tamanho variável dos nós da lista de arestas incidentes (representadas pelos vértices conectados) é apresentada na figura 2 e tem as mesmas características descritas no item b.

Exemplo :

```
INCIDENT i;
     i  =  get_incid(malha,1);
     v  =  i->incid[0];
```

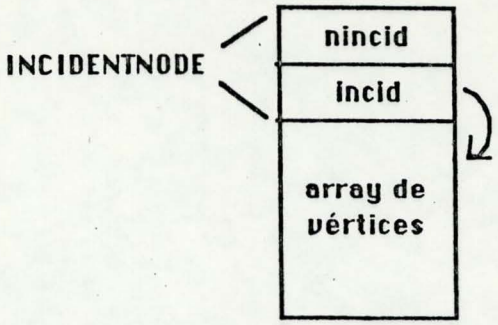

Figura 2 - Nó descritor das arestas incidentes a um vértice da malha

REFERENCIAS BIBLIOGRAFICAS

[Zimmermann 85] Robert Zimmermann. "Listpack - A Generic List Package for C programs". Carnegie-Mellon University, 1985.

[IRIS] "IRIS User's Guide - Volume I - Programming Guide".