***Algoritmo e funcionamento do Half Edge***

***Por: Jonathan Iury A. dos Santos***

Half-edge é um método algorítmico que serve para fazer uma representação mesh dos vértices em um plano manifold. Manifold quer dizer que é um espaço topológico onde cada ponto possui uma vizinhança aberta equivalente a um disco, que se analisada localmente numa área pequena a suficiente no entorno de um ponto dado, uma superfície existente num espaço tridimensional pode ser considerada “chata” ou plana.

A maneira como o Half-edge funciona é similar ao funcionamento do Winged-edge (outro algoritmo), porém o nó Half-Edge descreve uma linha que forma um loop. Que consiste num ponteiro para o loop que o contém e um ponteiro para o vértice inicial da linha na direção do loop. Possui também ponteiros para as Half-Edges anterior e posterior daquele loop, formando uma lista duplamente encadeada de half-edges de um loop. Desta forma, o vértice final de uma linha é tido como vértice inicial da próxima Half-Edge.

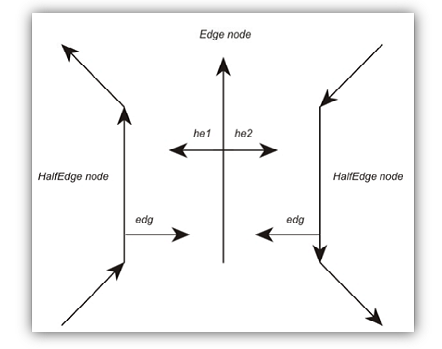


Figura – Relação entre a aresta e semi-aresta na estrutura de dados Half-Edge

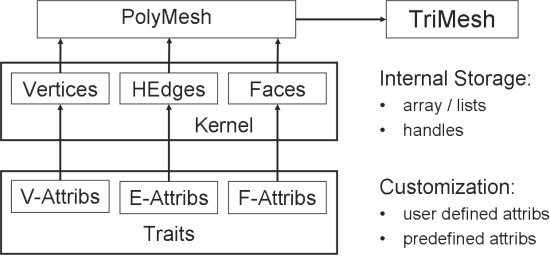


Figura – Modelo de estrutura e hierarquia entre os algoritmos e modelos

O algoritmo e as funções que devem ser informadas para seu funcionamento são:

1. Cada vértice referência um half-edge de saída, ou seja, um half-edge que começa neste vértice.
2. Cada face referência a um dos half-edges delimitam.
3. Cada half-edge fornece um identificador para
4. O vértice ele aponta, a face que pertence a próxima half-edge dentro da face (ordenado sentido anti-horário), o halfedge oposto, (opcionalmente: a half-edge anterior na cara).

Segue abaixo um código pego pelo projeto opensource que utilizei para o estudo desta estrutura para exemplificação:

struct HalfEdge

{

HalfEdge \* oppositeHalfEdge;

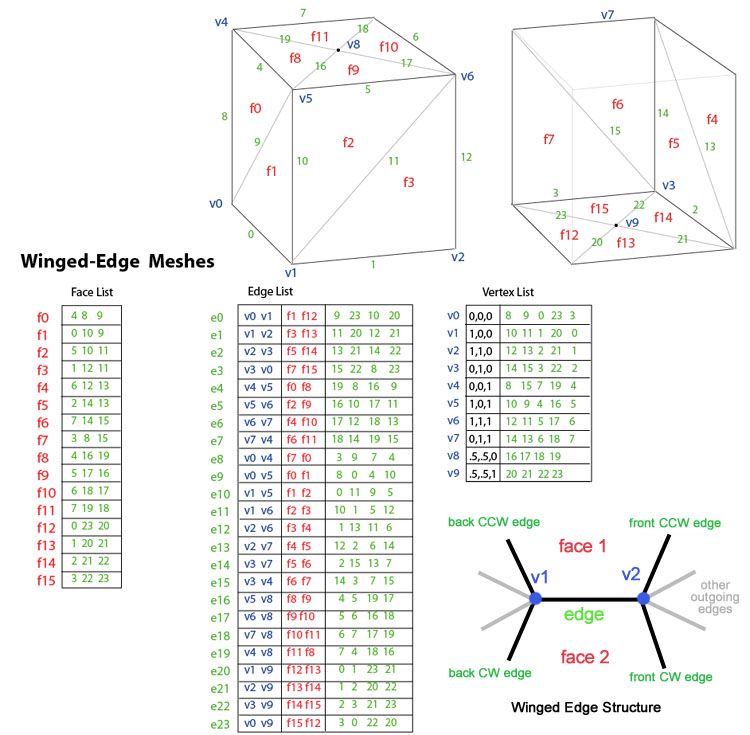
HalfEdge \* nextHalfEdge;

Vertex \* vertex;

Face \* face;

}

Como pode-se observar, cada vértice tem como estrutura no Half-Edge um ponto anterior, um posterior, o vértice e que face ele aponta. Se for observar com relação ao Winged-Edge, pode-se notar que no winged-edge tem a estrutura da outra face que o ponto compõem, o que no Half-Edge não possui. A imagem logo abaixo ilustra muito bem a diferença do Half-edge para o Winged-Edge.



Em seguida, para percorrer todos os vértices é preciso de um laço que informe cada ponto e este será preenchido com estas informações da estrutura acima para realizar o mesh(arresta) de cada vértice e sua respectiva face. Algo parecido com o código abaixo:

for cada face F //Face

{

for cada edge (u,v) //de Face

{

Edges[ par(u,v) ] = novo HalfEdge();

Edges[ par(u,v) ]->face = F;

}

for cada edge (u,v) //de F

{

set Edges[ par(u,v) ]->nextHalfEdge //para proximo half-edge de F

if ( Edges.search( par(v,u) ) != Edges.end() )

{

Edges[ par(u,v) ]->oppositeHalfEdge = Edges[ par(v,u) ];

Edges[ par(v,u) ]->oppositeHalfEdge = Edges[ par(u,v) ];

}

}

}

Referências

OpenMesh:

<http://openmesh.org/Documentation/OpenMesh-Doc-Latest/mesh_hds.html>

FlipCode:

<http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml>

CS-Virginia:

<http://www.cs.virginia.edu/~gfx/Courses/2008/AdvancedGraphics/lectures/lecture02_mesh.pdf>