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.

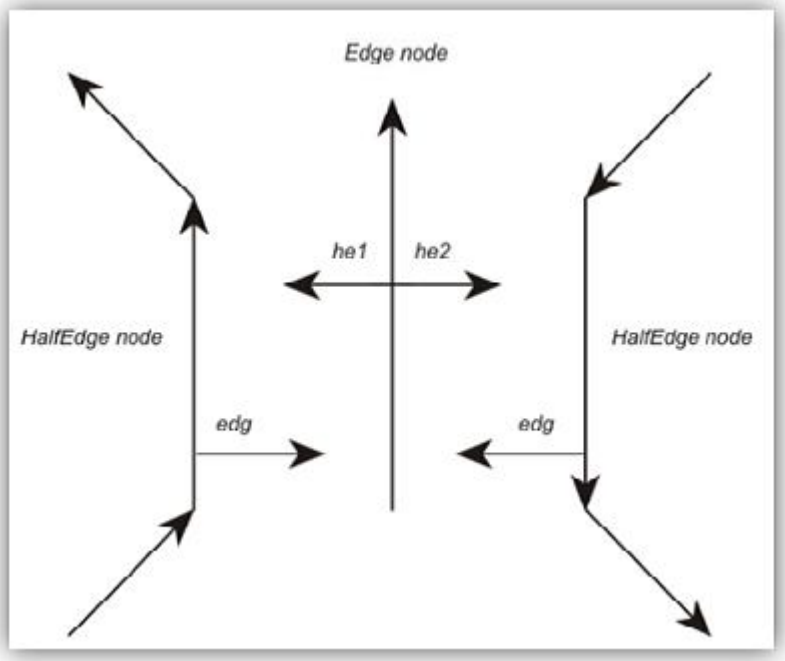


Figura – Relação entre a aresta e semi-aresta na estrutura de dados Half-Edge

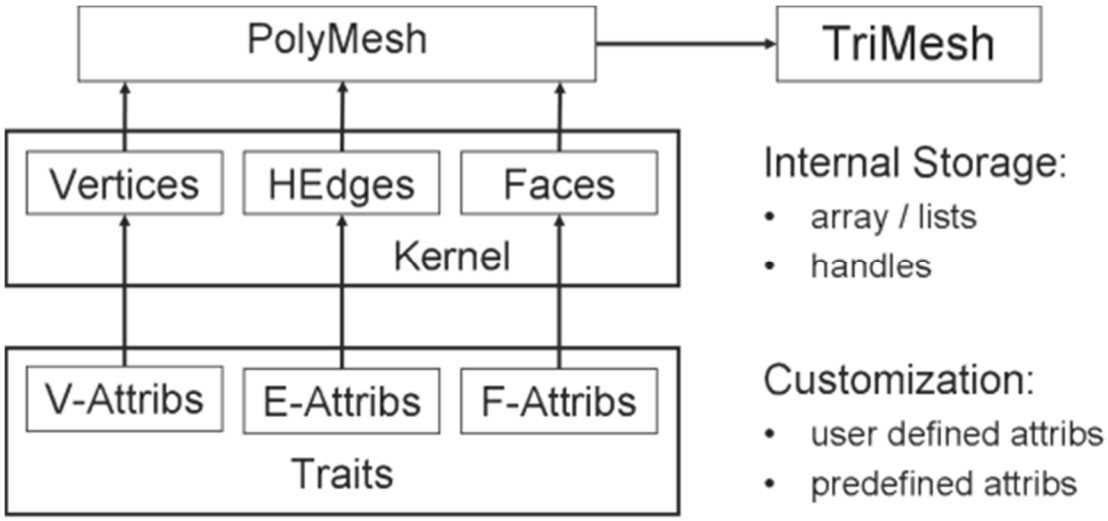


Figura – Modelo de estrutura e hierarquia entre os algoritmos e modelos

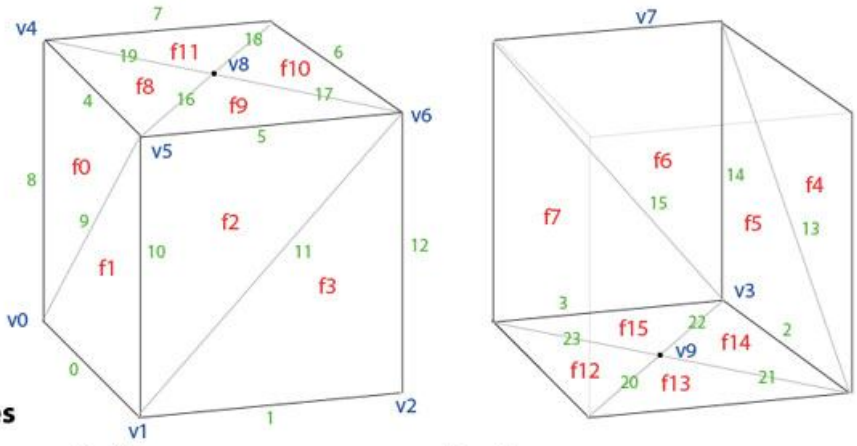
O algoritmo e as funções que devem ser informadas para seu funcionamento são:

I. Cada vértice referência um half-edge de saída, ou seja, um half-edge que começa neste vértice.
II. Cada face referência a um dos half-edges delimitam.
III. Cada half-edge fornece um identificador para
IV. 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.



Winged-Edge Meshes

Face List

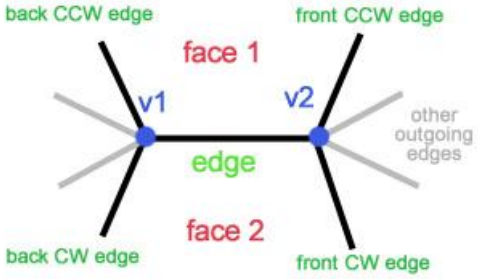
f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15
f8	4 16 19
f9	5 17 16
f10	6 18 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
f15	3 22 23

Edge List

e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 8 23
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19
e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

Vertex List

v0	0,0,0	8 9 0 23 3
v1	1,0,0	10 11 1 20 0
v2	1,1,0	12 13 2 21 1
v3	0,1,0	14 15 3 22 2
v4	0,0,1	8 15 7 19 4
v5	1,0,1	10 9 4 16 5
v6	1,1,1	12 11 5 17 6
v7	0,1,1	14 13 6 18 7
v8	.5,.5,0	16 17 18 19
v9	.5,.5,1	20 21 22 23



Winged Edge Structure

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