

Triangle Mesh: Halfedge Data Structure

David Gu

Yau Mathematics Science Center
Tsinghua University

gu@cs.stonybrook.edu

July 7, 2020

Triangle Mesh: Halfedge Data Structure

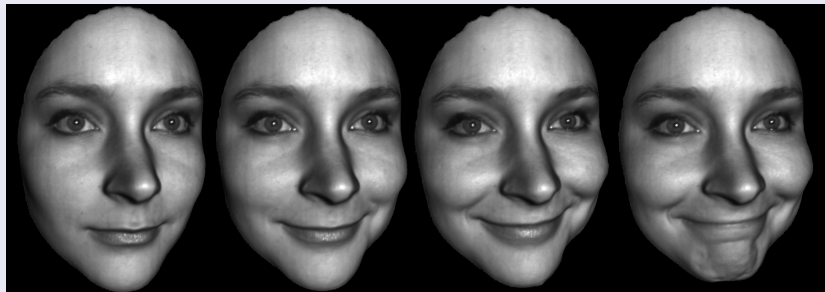
Discrete Surfaces

Acquired using 3D scanner.



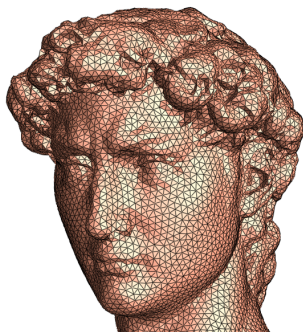
Discrete Surfaces

Our group has developed high speed 3D scanner, which can capture facial surfaces with dynamic expressions.



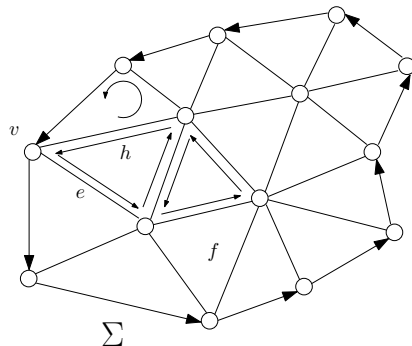
Generic Surface Model - Triangular Mesh

- Surfaces are represented as polyhedron triangular meshes.
- Isometric gluing of triangles in \mathbb{E}^2 .
- Isometric gluing of triangles in $\mathbb{H}^2, \mathbb{S}^2$.



- Topology - Simplicial Complex , combinatorics
- Conformal Structure - Corner angles (and other variant definitions)
- Riemannian metrics - Edge lengths
- Embedding - Vertex coordinates

Generic Surface Model - Triangular Mesh



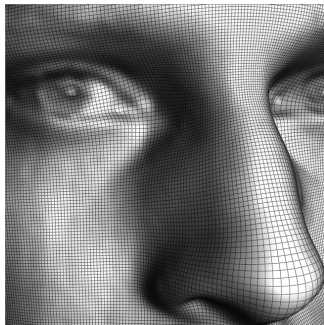
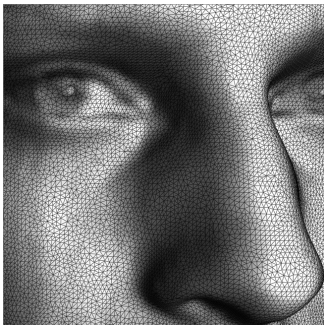
Triangle mesh

Definition (Triangle Mesh)

A triangle mesh is a oriented two dimensional simplicial complex, generally embedded in \mathbb{R}^3 .

Our goal is to design a data structure to efficiently represent general meshes.

Generic Surface Model - Triangular Mesh



fundamental classes

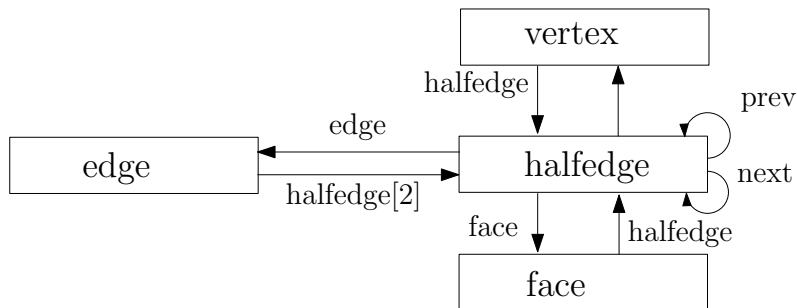
- Vertex
- Halfedge, oriented edge
- Edge, non-oriented edge
- Face, oriented

Links

All objects are linked together through pointers, such that

- 1 The local Euler operation can be easily performed
- 2 The memory cost is minimized

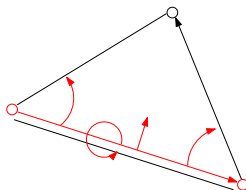
Generic Surface Model - Triangular Mesh



Halfedge class

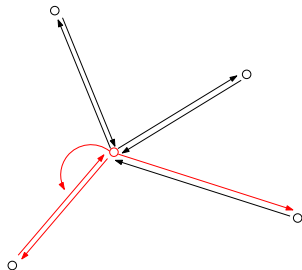
Pointers

- Halfedge pointers: prev, next halfedge;
- Vertex pointers: target vertex, source vertex;
- Edge pointer: the adjacent edge;
- face pointer: the face it belongs to;



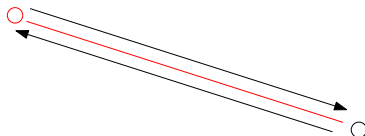
Pointers

- Halfedge pointers: the first in halfedge



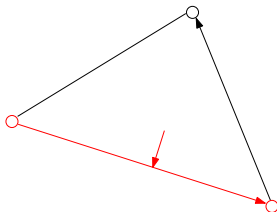
Pointers

- Halfedge pointers: to the adjacent two halfedges.
- if the edge is on the boundary, then the second halfedge pointer is null.



Pointers

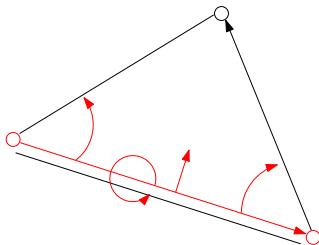
- Halfedge pointers: to the first halfedge.



Mesh class

Data members

- A list of vertices;
- A list of halfedges;
- A list of edges;
- A list of faces;



Euler Operation

circulating neighbors of a vertex $v \rightarrow v/e/f/h$

- iterate out-halfedges counter-clock-wisely
- iterate in-halfedges counter-clock-wisely
- iterate neighboring faces CCWly
- iterate neighboring vertices CCWly

Rotate a halfedge about its target vertex clwly:

$$he = he \rightarrow next() \rightarrow dual();$$

Rotate a halfedge about its target vertex ccwly:

$$he = he \rightarrow dual() \rightarrow prev();$$

Euler Operation

circulating neighbors of a face $f \rightarrow v/e/f/h$

- iterate halfedges ccwly
- iterate edges ccwly
- iterate vertices ccwly
- iterate faces ccwly

Circulate halfedges of a face ccwly:

$$he = he \rightarrow next()$$

circulate halfedge of a face clwly:

$$he = he \rightarrow prev();$$

Attributes

Each object stores attributes (traits) which defines other structures on the mesh:

- metric structure: edge length
- angle structure: halfedge
- curvature : vertex
- conformal factor: vertex
- Laplace-Beltrami operator: edge
- Ricci flow edge weight; edge
- holomorphic 1-form: halfedge

Coding Procedure

Define Mesh Class

- define vertex, face, edge, halfedge classes;
- define mesh class with template, including all types of iterators;
- instantiate the mesh class, with the vertex, face, edge, halfedge classes;
- define different methods for the mesh class.

Design Algorithm

- Use the mesh as the main data structure;
- Update the attributes of vertex, edge, halfedge and face;
- Update the connectivity;
- Form numerical linear systems, use linear package to solve it.

```

1  class CMyVertex : public CVertex
2  {
3  public:
4      CMyVertex() : m_rgb(1, 1, 1) {};
5      ~CMyVertex() {};
6      void _from_string();
7      CPoint & rgb() { return m_rgb; };
8  protected:
9      CPoint m_rgb;
10 };
11 inline void CMyVertex::_from_string()
12 {
13     CParser parser(m_string);
14     for (std::list<CToken*>::iterator iter = parser.tokens()
15         .begin(); iter != parser.tokens().end(); ++iter)
16     {
17         CToken * token = *iter;
18         if (token->m_key == "rgb") // CPoint
19             token->m_value >> m_rgb;
20     }
21 }

```

Listing 1: Vertex Class

```

1  class CMyEdge : public CEdge
2  {
3  public:
4      CMyEdge() :m_sharp(false) {};
5      ~CMyEdge() {};
6      void _from_string();
7      bool & sharp() { return m_sharp; };
8  protected:
9      bool m_sharp;
10 };
11 inline void CMyEdge::_from_string()
12 {
13     CParser parser(m_string);
14     for (std::list<CToken*>::iterator iter = parser.tokens()
15         .begin(); iter != parser.tokens().end(); ++iter)
16     {
17         CToken * token = *iter;
18         if (token->m_key == "sharp") // bool
19             m_sharp = true;
20     }
21 }

```

Listing 2: Edge Class

```

1  class CMyFace : public CFace
2  {
3  public:
4
5      CPoint & normal() { return m_normal; };
6      double & area()   { return m_area;   }
7  protected:
8      CPoint m_normal;
9      double m_area;
10 };
11
12 class CMyHalfEdge: public CHalfEdge
13 {
14 public:
15     double angle() { return m_angle; }
16 protected:
17     double m_angle;
18 };

```

Listing 3: Face and HalfEdge Class

```

1  template<typename V, typename E, typename F, typename H>
2  class MyMesh : public CDynamicMesh<V, E, F, H>
3  {
4  public:
5      typedef V V;
6      typedef E E;
7      typedef F F;
8      typedef H H;
9
10     typedef CBoundary<V, E, F, H>          CBoundary;
11     typedef CLoop<V, E, F, H>              CLoop;
12
13     typedef MeshVertexIterator<V, E, F, H>
14     MeshVertexIterator;
15     typedef MeshEdgeIterator<V, E, F, H>
16     MeshEdgeIterator;
17     typedef MeshFaceIterator<V, E, F, H>
18     MeshFaceIterator;
19     typedef MeshHalfEdgeIterator<V, E, F, H>
20     MeshHalfEdgeIterator;
21
22     typedef VertexVertexIterator<V, E, F, H>
23     VertexVertexIterator;

```



```

19     typedef VertexEdgeIterator<V, E, F, H>
VertexEdgeIterator;
20     typedef VertexFaceIterator<V, E, F, H>
VertexFaceIterator;
21     typedef VertexInHalfedgeIterator<V, E, F, H>
VertexInHalfedgeIterator;
22     typedef VertexOutHalfedgeIterator<V, E, F, H>
VertexOutHalfedgeIterator;
23
24     typedef FaceVertexIterator<V, E, F, H>
FaceVertexIterator;
25     typedef FaceEdgeIterator<V, E, F, H>
FaceEdgeIterator;
26     typedef FaceHalfedgeIterator<V, E, F, H>
FaceHalfedgeIterator;
27
28     void outputMeshInfo();
29     void testIterator();
30 };
31 typedef MyMesh<CMyVertex, CMyEdge, CMyFace, CMyHalfEdge>
CMyMesh;

```

Listing 4: Mesh Class

```

1 template<typename V, typename E, typename F, typename H>
2 void MyMesh<V, E, F, H>::testIterator()
3 {
4     for (MeshVertexIterator viter(this); !viter.end(); ++
5         viter)
6     {
7         V * pV = *viter;
8         // you can do something to the vertex here
9         // ...
10
11         for (VertexVertexIterator vviter(pV); !vviter.end();
12             ++vviter)
13         {
14             V * pW = *vviter;
15             // you can do something to the neighboring vertices
16             with CCW
17             // ...
18         }
19
20         for (VertexEdgeIterator veiter(pV); !veiter.end(); ++
21             veiter)
22         {
23             E * pE = *veiter;

```

```

20         // you can do something to the neighboring edges
with CCW
21         // ...
22     }
23
24     for (VertexFaceIterator vfiter(pV); !vfiter.end(); ++
vfiter)
25     {
26         F * pF = *vfiter;
27         // you can do something to the neighboring faces
with CCW
28         // ...
29     }
30
31     for (VertexInHalfedgeIterator whiter(this, pV); !
whiter.end(); ++whiter)
32     {
33         H * pH = *whiter;
34         // you can do something to the incoming halfedges
with CCW
35         // ...
36     }
37 }

```

```

38   for (MeshEdgeIterator eiter(this); !eiter.end(); ++eiter
39   )
40   {
41       E * pE = *eiter;
42       // you can do something to the edge here
43       // ...
44   }
45
46   for (MeshFaceIterator fiter(this); !fiter.end(); ++fiter
47   )
48   {
49       F * pF = *fiter;
50       // you can do something to the face here
51       // ...
52   }
53
54   //there are some other iterators which you can find them
55   in class MyMesh
56 }

```

Listing 5: Test different iterators