Geometry Processing
Lecture 3: OpenMesh Tutorial

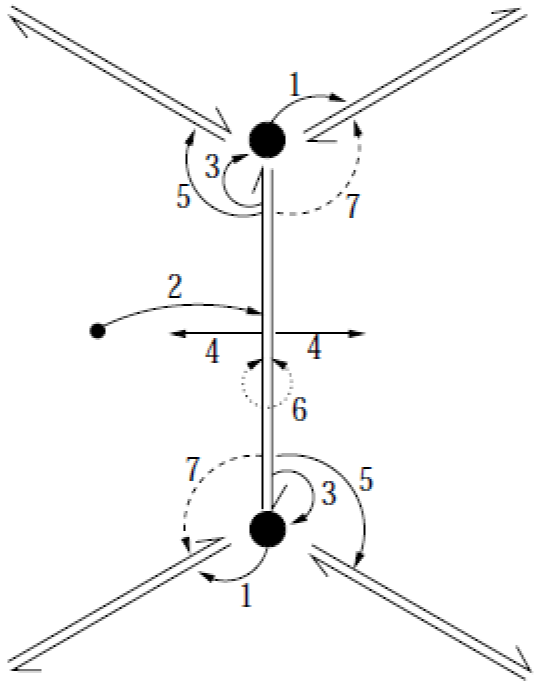
Jian Sun
Mathematical Sciences Center, Tsinghua University

March 11th 2013

OpenMesh

- ACG – RWTH Aachen http://openmesh.org/
- C++ library
- Implements half-edge data structure
- Integrated basic geometric operations
- 3-D model file reader/writer

Half Edge Data Structure



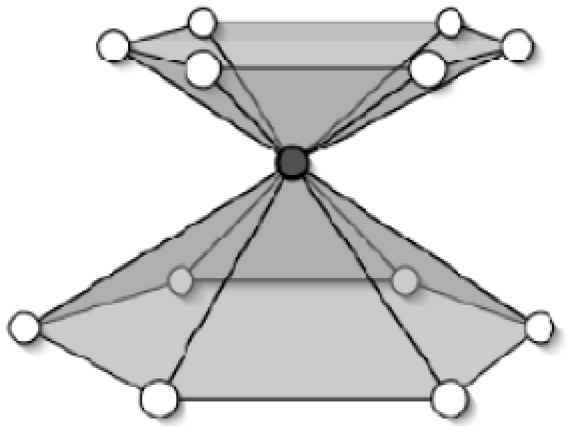
1. Vertex $\mapsto$ one outgoing halfedge,
2. Face $\mapsto$ one halfedge,
3. Halfedge $\mapsto$ target vertex,
4. Halfedge $\mapsto$ its face,
5. Halfedge $\mapsto$ next halfedge,
6. Halfedge $\mapsto$ opposite halfedge (implicit),
7. Halfedge $\mapsto$ previous halfedge (optional).

Open Mesh

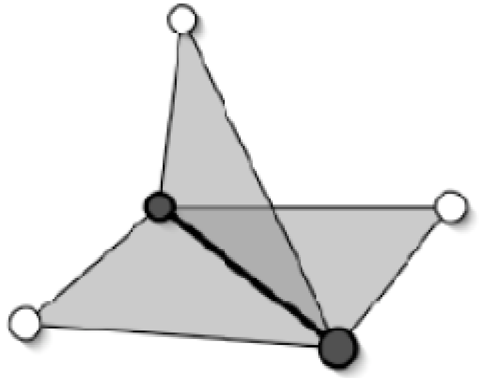
- Flexible
  - Random access to vertices, edges, and faces.
  - Support general polygonal meshes
  - Array or lists as underlying kernels
  - Arbitrary scalar types
  - Additional traits.

Open Mesh

- Efficient in space and time
  - Dynamic memory management for array-based meshes
  - Extendable to specialized kernels for non-manifold meshes



supported but how



Not supported so far

Mesh Definition

```
#include <OpenMesh/Core/IO/MeshIO.hh>
#include <OpenMesh/Core/Mesh/Types/TriMesh_ArrayKernelT.hh>

typedef Openmesh::TriMesh_ArrayKernelT<>  Mesh;
```

name space

mesh type:
- triangle mesh
- array kernel
- default traits

Mesh Definition

```
#include <OpenMesh/Core/IO/MeshIO.hh>
#include <OpenMesh/Core/Mesh/Types/TriMesh_ArrayKernelT.hh>

typedef Openmesh::TriMesh_ArrayKernelT<>  Mesh;
```

PolyMesh_ArrayKernelT

mesh type:
- triangle mesh
- array kernel
- default traits

name space

Properties

- Standard properties
  - Color, Normal, Position, Status, TexCoord
- Custom properties
- Extending mesh using traits

Standard Properties

- Add: e.g., request_vertex_normals()
- Remove: release_vertex_normals()
- Query: has_vertex_normals()
- Set: set_normal(VertexHandle, Normal&)
- Access: const Normal& normal(VertexHandle)

Custom Properties

- OpenMesh::VPropHandleT<MyMesh::Point> cogs;

  mesh.add_property(cogs);

  mesh.property(cogs,v_it) += mesh.point(vv_it);

  mesh.set_point(v_it, mesh.property(cogs,v_it));

Extending mesh using traits

```
#include <OpenMesh/Core/Mesh/TriMesh_ArrayKernelT.hh>
#include <OpenMesh/Core/Geometry/VectorT.hh>
struct MyTraits : public OpenMesh::DefaultTraits
{
typedef OpenMesh::Vec3d Point;
typedef OpenMesh::Vec3d Normal;
typedef OpenMesh::Vec4f Color;
VertexTraits
{
public:
const unsigned int valence() const { return valence; }
void set_valence(const unsigned int v) { valence = v; }
private:
unsigned int valence;
}; };

typedef OpenMesh::TriMesh_ArrayKernelT<MyTraits> MyMesh;
```

OpenMesh

- Iterating over vertices

```
typedef Openmesh::TriMesh_ArrayKernelT<> Mesh;
Mesh * myMesh;

Mesh::VertexIter vIt , vBegin , vEnd;

vBegin = myMesh->vertices_begin();
vEnd = myMesh->vertices_end();

for( vIt = vBegin ; vIt != vEnd ; ++vIt )
{
        doSomethingWithVertex(vIt.handle());
}
```

OpenMesh

- Iterating over faces

  Mesh::VertexIter → Mesh::FaceIter

  vertices_begin() → faces_begin()

  vertices_end() → faces_end()

OpenMesh

- Circulating over faces around a vertex

```
Mesh::VertexIter vIt , vBegin , vEnd;

vBegin = myMesh->vertices_begin();
vEnd = myMesh->vertices_end();

for( vIt = vBegin ; vIt != vEnd ; ++vIt )
{
        Mesh::VertexFaceIter vfIt , vfBegin;
        vfBegin = myMesh->vf_iter(vIt);

        for( vfIt = vfBegin ; vfIt ; ++vfIt)
        {
                doSomethingWithFace(vfIt.handle());
        }
}
```

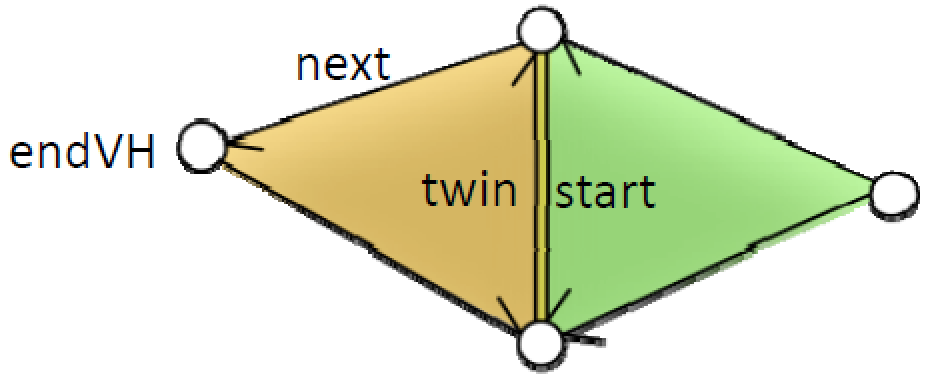
OpenMesh

- Neighbor Access in O(1)

```
OpenMesh::VertexHandle endVH;
OpenMesh::HalfEdgeHandle , startHEH , twinHEH , nextHEH;

startHEH = hehIt.handle();

twinHEH = myMesh->opposite_halfedge_handle(startHEH);
nextHEH = myMesh->next_halfedge_handle(twinHEH);
endVH =  myMesh->to_vertex_handle(nextHEH);
```



OpenMesh

- Modifying the geometry

```
for( vIt = vBegin ; vIt != vEnd ; ++vIt )
{
        scale(vIt.handle() , 2.0);
}

void scale(OpenMesh::VertexHandle & _vh , double _alpha)
{
        OpenMesh::Vec3f newCoordinate;
        newCoordinate = myMesh->point(_vh);
        myMesh->set_point(_vh , newCoordinate * _alpha);
}
```

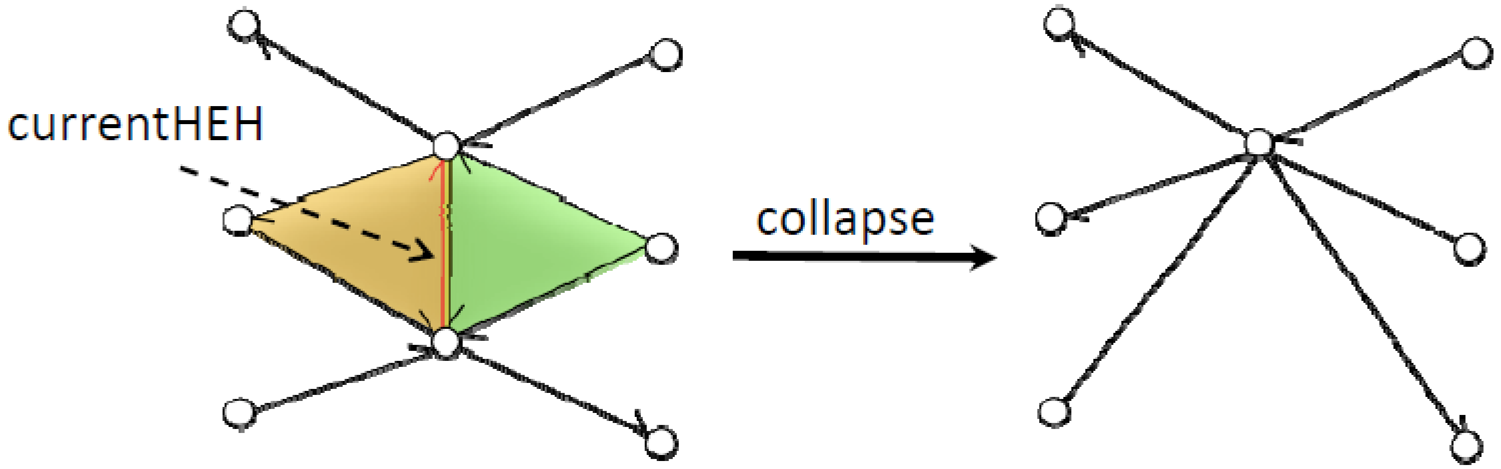
OpenMesh

- Modifying the topology

```
myMesh->request_vertex_status();
myMesh->request_edge_status();
myMesh->request_face_status();
```

```
OpenMesh::HalfedgeHandle currentHEH = helt.handle();
myMesh->collapse(currentHEH);
```

```
myMesh->garbage_collection();
```



OpenMesh

- Geometric Operations

```
OpenMesh::Vec3f x,y,n,crossproductXY;
...
l = (x-y).length();

n = x.normalize();
scalarProductXY = (x | y);
crossProductXY = x % y;
...
```

OpenMesh

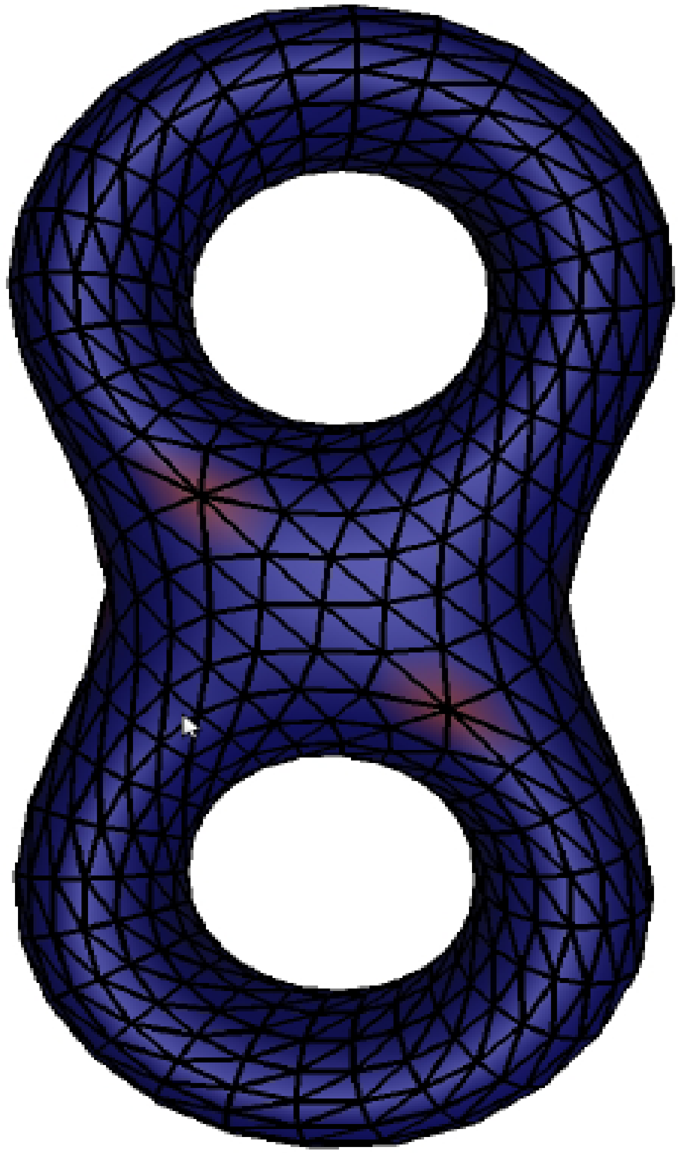
- Vertices, perimeter, area of a triangle

```
void analyzeTriangle(OpenMesh::FaceHandle & _fh)
{
        OpenMesh::Vec3f pointA , pointB , pointC;
        Mesh::ConstFaceVertexIter cfvIt;

        cfvIt = myMesh->cfv_iter(_fh);
        pointA = myMesh->point(cfvIt.handle());
        pointB = myMesh->point((++cfvIt).handle());
        pointC = myMesh->point((++cfvIt).handle());

        perimeter(pointA,pointB,pointC);
        area(pointA,pointB,pointC)
}
```

Project 1: Color code valence



References

- A similar course by Mirela Ben-Chen:
  http://graphics.stanford.edu/courses/cs468-10-fall/

- Slides from http://www.pmp-book.org/

Thank you for your attention

Questions?

Page Title

- first level bulletin
  - second level bulletin
    - third level bulletin