Open Mesh Reference

OpenMesh

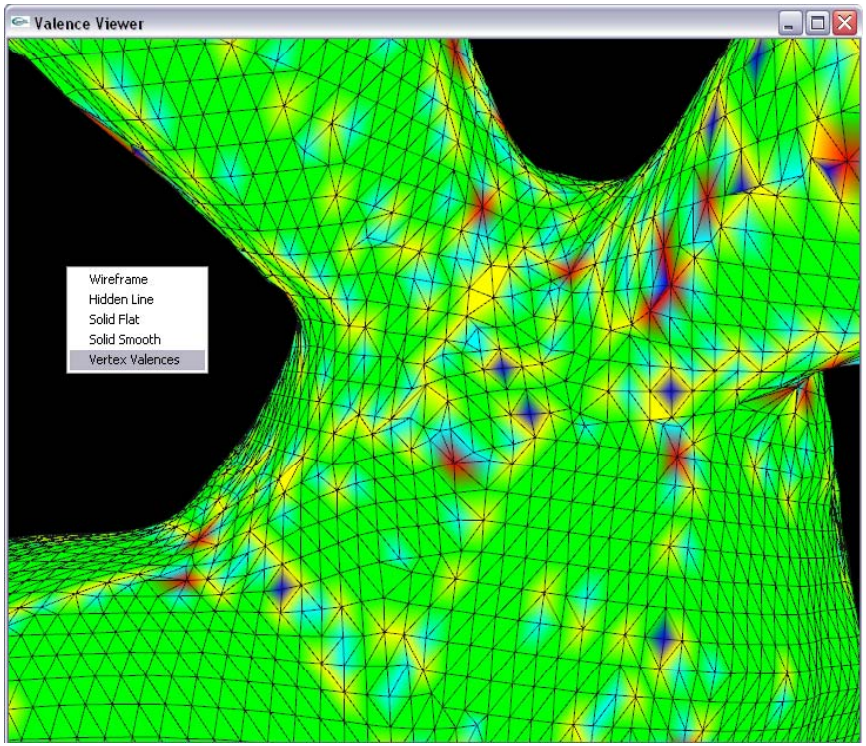
- ACG – RWTH Aachen
- C++ library
- Implements half-edge data structure
- Integrated basic geometric operations
- 3-D model file reader/writer

OpenMesh

- Flexible
  - Random access to vertices, edges, and faces
  - Arbitrary scalar types
  - Arrays or lists as underlying kernels
- Efficient in space and time
  - Dynamic memory management for array-based meshes
  - Extendable to specialized kernels for non-manifold meshes

Exercise 1

- Valence Viewer



Exercise 1

- Microsoft Visual Studio 2008

- Solution file has two projects:
  - OpenMesh library
    - compile it once
    - never need to edit
  - GLUT based mesh viewer
    - extend with your code

Exercise 1

- Classes

GlutViewer	GLUT window, popup menu
GlutExaminer	Trackball, basic rendering of teapot
MeshViewer	Loading and rendering mesh
ValenceViewer	Custom processing and rendering mode

Exercise 1

- Compute valences in a custom mesh property
- Compute colors out of valences and store them in the predefined property

Exercise 1

- Send zip of your source files, project files and solution files
- Describe your solution in readme.txt.
- Don’t send binaries and other intermediary files
- Can use the clearSolution.bat
  - ! deletes recursively all debug and release directories and intermediary solution files

OpenMesh

- Geometric Operations

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

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

OpenMesh

- 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

OpenMesh

- Loading/Writing a Mesh

```
Mesh * myMesh;

OpenMesh::IO::Options readOptions;

OpenMesh::IO::read_mesh(*myMesh,"/path/to/bunny.off",readOptions)
```

reader/writer settings:
– enable vertex normals/colors / texture coordinates?
– enable face normals/colors?

OpenMesh

- Adding Attributes

```
Mesh * myMesh;
OpenMesh::IO::Options readOptions;
OpenMesh::IO::read_mesh(*myMesh, ”/path/to/bunny.off” , readOptions)

if(!readOptions.check(OpenMesh::IO::Options::FaceNormal))
{
        myMesh->update_face_normals();
}

if(! readOptions.check(OpenMesh::IO::Options::VertexNormal))
{
        myMesh->update_vertex_normals();
}
```

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

- 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)
}
```

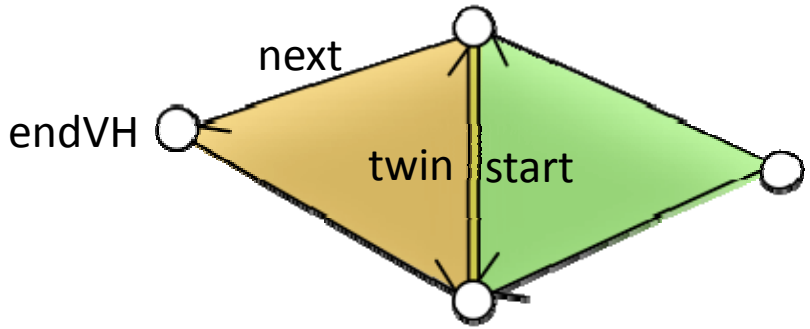
OpenMesh

- 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);
}
```

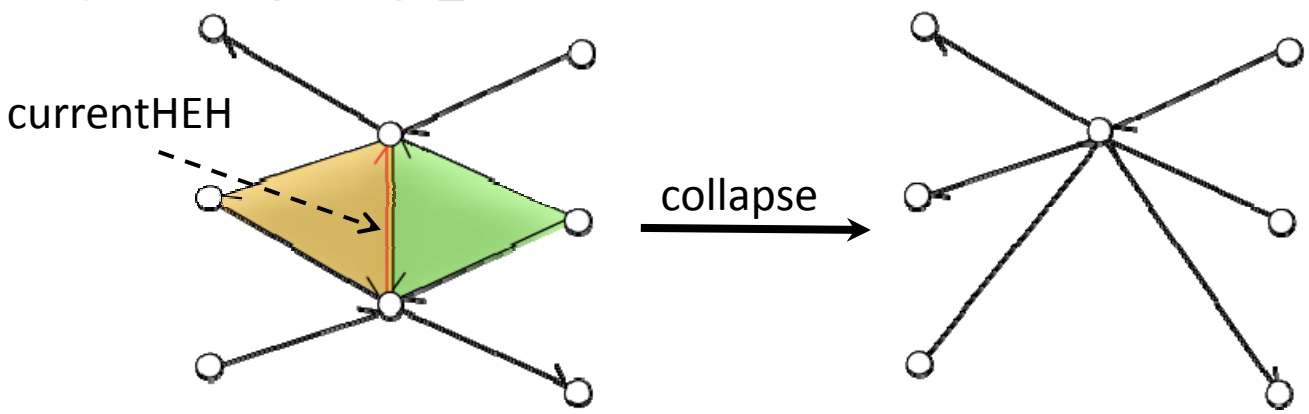
OpenMesh

- Modifying the topology

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

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

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



OpenMesh

- Adding Custom Traits

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

struct myMeshTraits : public OpenMesh::DefaultTraits
{
        typedef OpenMesh::Vec4f Color;

        VertexAttributes (
        OpenMesh::Attributes::Normal |
        OpenMesh::Attributes::Color);

        FaceAttributes (
        OpenMesh::Attributes::Normal |
        OpenMesh::Attributes::Color);

}
```

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

OpenMesh

- Setting/Getting Predefined Attributes

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

Mesh * myMesh;

... // load file into myMesh

myMesh->request_vertex_normals();
myMesh->request_vertex_colors();
myMesh->request_face_normals();

...

myMesh->set_color(currentVH,Mesh::Color(0,0,255));

blueColor = myMesh->color(currentVH);
```

OpenMesh

- Setting/Getting Custom Attributes

```
OpenMesh::FPropHandleT<bool> marked;
myMesh->add_property(marked);

for(fIt = fBegin; fIt != fEnd; ++fIt)
{
        if(shouldMark(fIt))
                myMesh->property(marked,fIt) = true;
        else
                myMesh->property(marked,fIt) = false;
}

for(fIt = fBegin; fIt != fEnd; ++fIt)
{
        if(myMesh->property(marked,fIt))
        doSomething(fIt);
}
```

OpenMesh

- For more examples, tutorials, documentation: www.openmesh.org