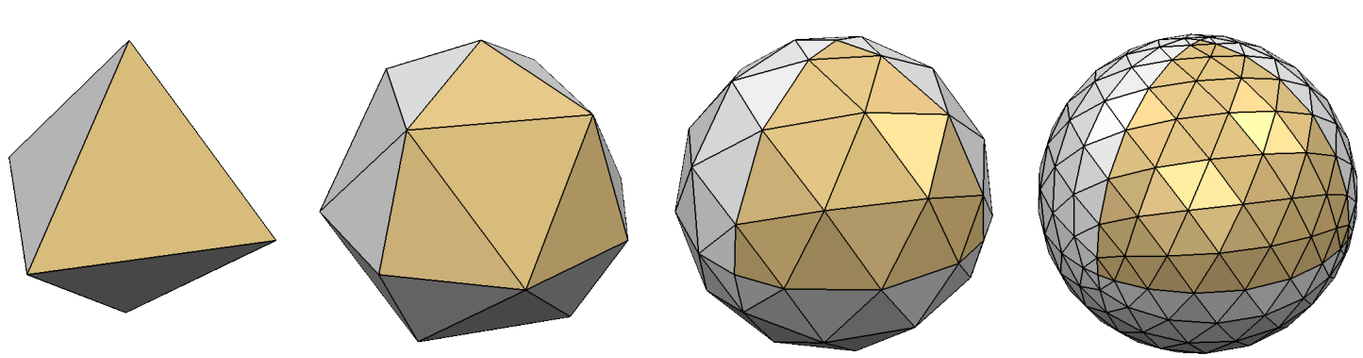
Geometric Modeling Based on Polygonal Meshes:

OpenMesh



Prof. Dr. Mario Botsch

Computer Graphics & Geometry Processing

Universität Bielefeld

CITEC

OpenMesh 1.1.0

- Developed at RWTH Aachen
- C++ library for polygonal / triangle meshes
- Implements halfedge data structure
- Includes
  - basic geometric operations
  - mesh processing algorithms
  - file IO

Why OpenMesh?

- Flexibility
  - Vertices, (half-)edges, faces
  - Arbitrary scalar types
  - Customizable mesh kernels
  - Dynamic properties for V, E, HE, F

Why OpenMesh?

- Already includes several algorithms
  - Mesh smoothing
  - Mesh decimation
  - Subdivision
  - Progressive meshes
  - OpenGL rendering
- And: It’s free!

Geometry: 2D, 3D Vectors

```
typedef OpenMesh::VectorT<float, 3>  Vec3f;
Vec3f x, y;

float length = (x-y).norm();
```

```
Vec3f n = (x%y);              // cross product
n.normalize();                // n.norm()==1
n /= n.norm();                // this would do the same...

float dot = (x | y);          // dot product
float angle = acos( dot / x.norm() / y.norm() );
```

Mesh Types & Customization

has to be included first

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


struct MyMeshTraits : public OpenMesh::DefaultTraits
{
  typedef OpenMesh::Vec3f    Point;
  typedef OpenMesh::Vec3f    Normal;
  typedef OpenMesh::Vec3uc   Color;
  typedef OpenMesh::Vec2f    TexCoord2D;
}

typedef OpenMesh::TriMesh_ArrayKernelT<MyMeshTraits>  Mesh;
```

define types

TriMesh
PolyMesh

File IO

```
Mesh mesh;

if (OpenMesh::IO::read_mesh(mesh, "bunny.off"))
{
   if (!mesh.has_face_normals())
      mesh.request_face_normals();

   if (!mesh.has_vertex_normals())
      mesh.request_vertex_normals();

   mesh.update_face_normals();
   mesh.update_vertex_normals();
}

if (!OpenMesh::IO::write_mesh(mesh, "output.off"))
   std::cerr << "writing failed\n";
```

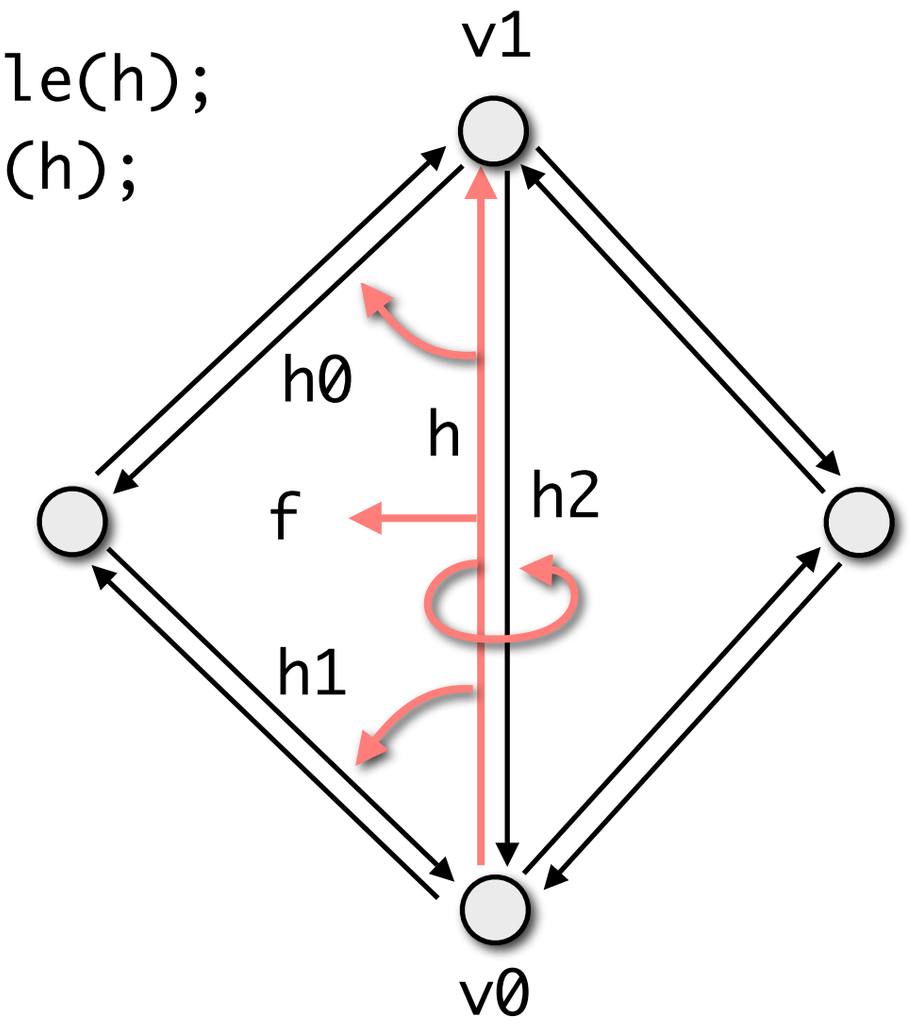
Can read OFF, OBJ, STL, IV

allocate the properties you need

Can write OFF, OBJ, STL, IV

Halfedge Connectivity

```
Mesh::HHandle h;
Mesh::HHandle h0 = mesh.next_halfedge_handle(h);
Mesh::HHandle h1 = mesh.prev_halfedge_handle(h);
Mesh::HHandle h2 = mesh.opposite_halfedge_handle(h);
Mesh::FHandle  f = mesh.face_handle(h);
Mesh::VHandle v0 = mesh.from_vertex_handle(h);
Mesh::VHandle v1 = mesh.to_vertex_handle(h);
```



What are Handles?

```
class BaseHandle
{
public:

  BaseHandle(int _idx=-1) : idx_(_idx) {}

  int idx() const { return idx_; }
  bool is_valid() const { return idx_ != -1; }

private:

  int idx_;
};
```

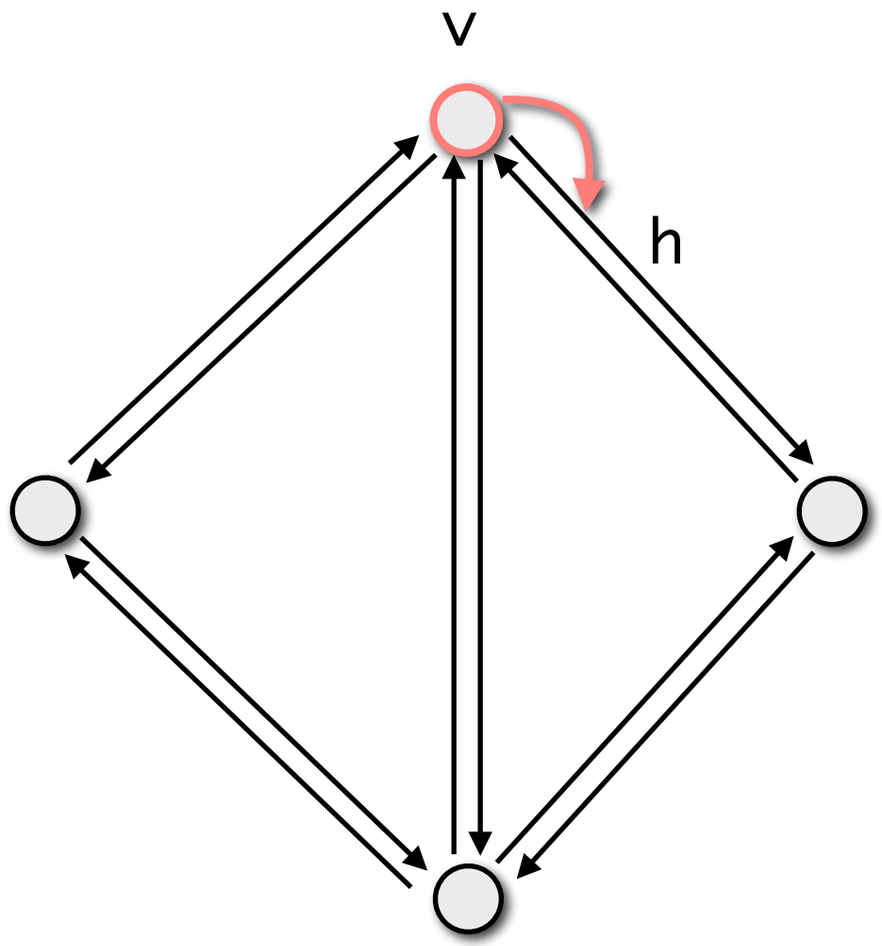
What are Handles?

```
struct VertexHandle   : public BaseHandle { ... };

struct HalfedgeHandle : public BaseHandle { ... };

struct EdgeHandle     : public BaseHandle { ... };

struct FaceHandle     : public BaseHandle { ... };
```

Vertex Connectivity

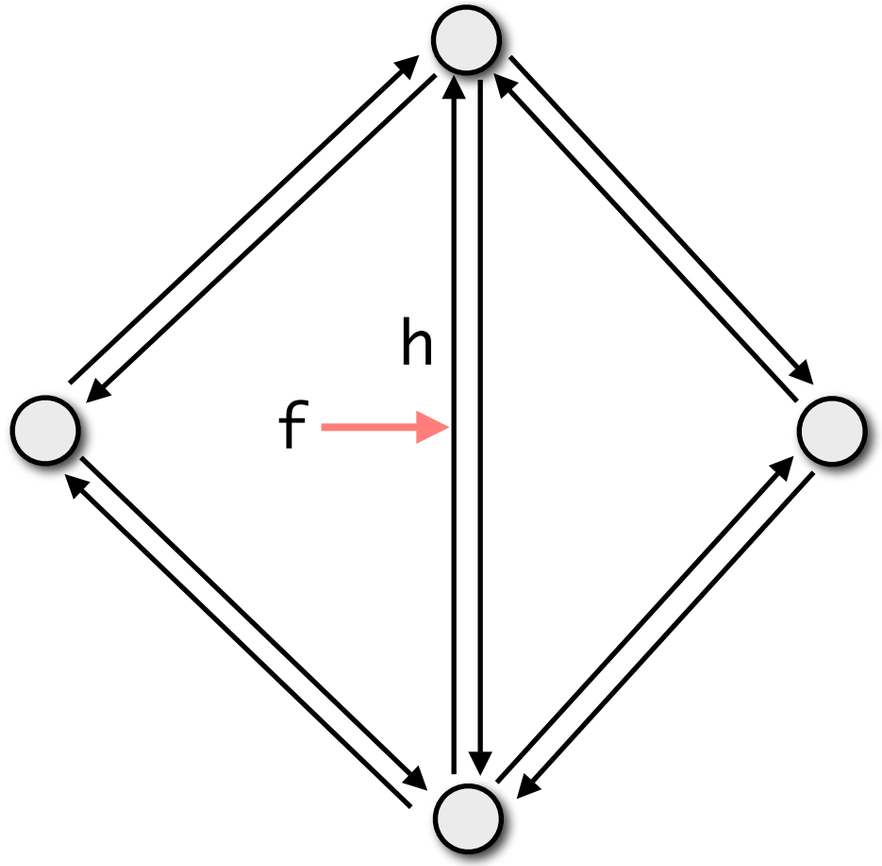
```
Mesh::VHandle v;
Mesh::HHandle h = mesh.halfedge_handle(v);
```

for boundary vertices the outgoing halfedge is a boundary halfedge



Face Connectivity

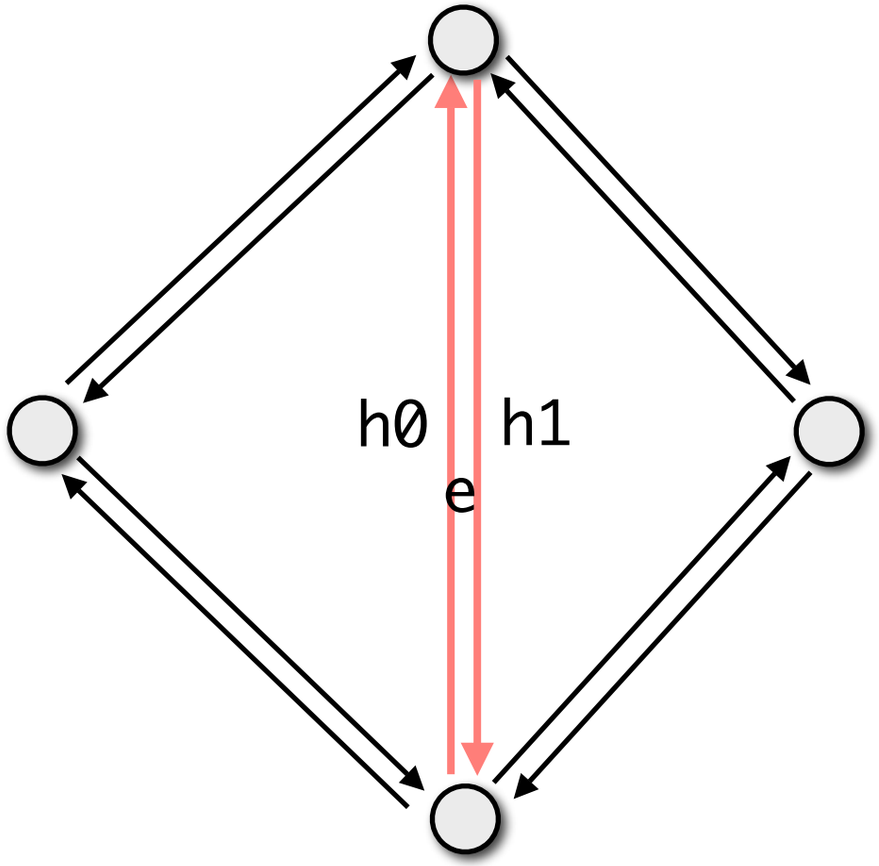
```
Mesh::FHandle f;
Mesh::HHandle h = mesh.halfedge_handle(f);
```



Edge Connectivity

```
Mesh::EHandle e;
Mesh::HHandle h0 = mesh.halfedge_handle(e,0);
Mesh::HHandle h1 = mesh.halfedge_handle(e,1);
```

edges simply cluster two halfedges



Iterators

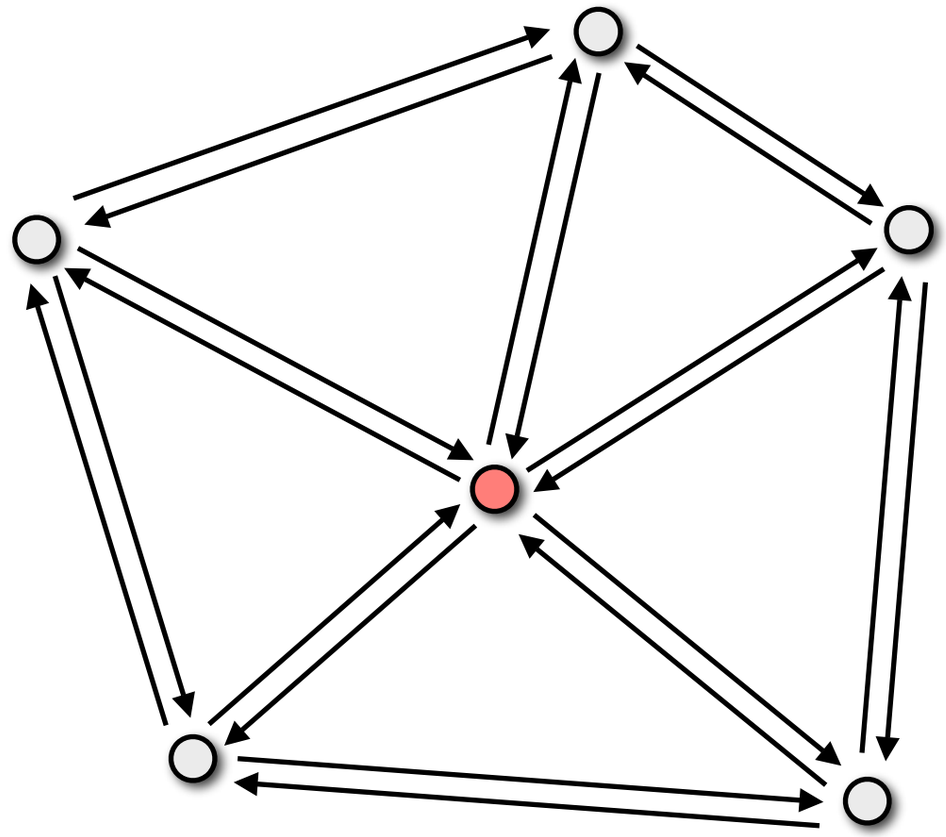
- Iterator over all vertices

```
Mesh::Point cog(0,0,0);
Mesh::VertexIter v_it;

for (v_it = mesh.vertices_begin();
     v_it != mesh.vertices_end();
     ++v_it)
  cog += mesh.point(v_it);

cog /= mesh.n_vertices();
```

- Analogous for Halfedgeiter, Edgeiter, Faceiter...

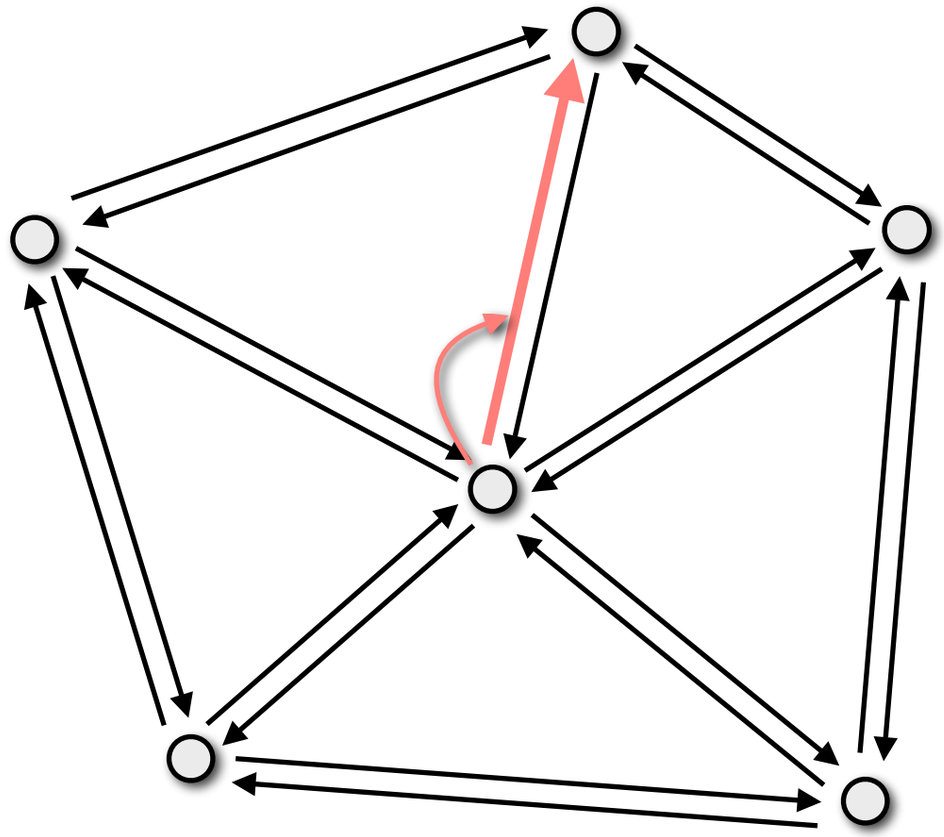
One-Ring Traversal

1. Start at vertex



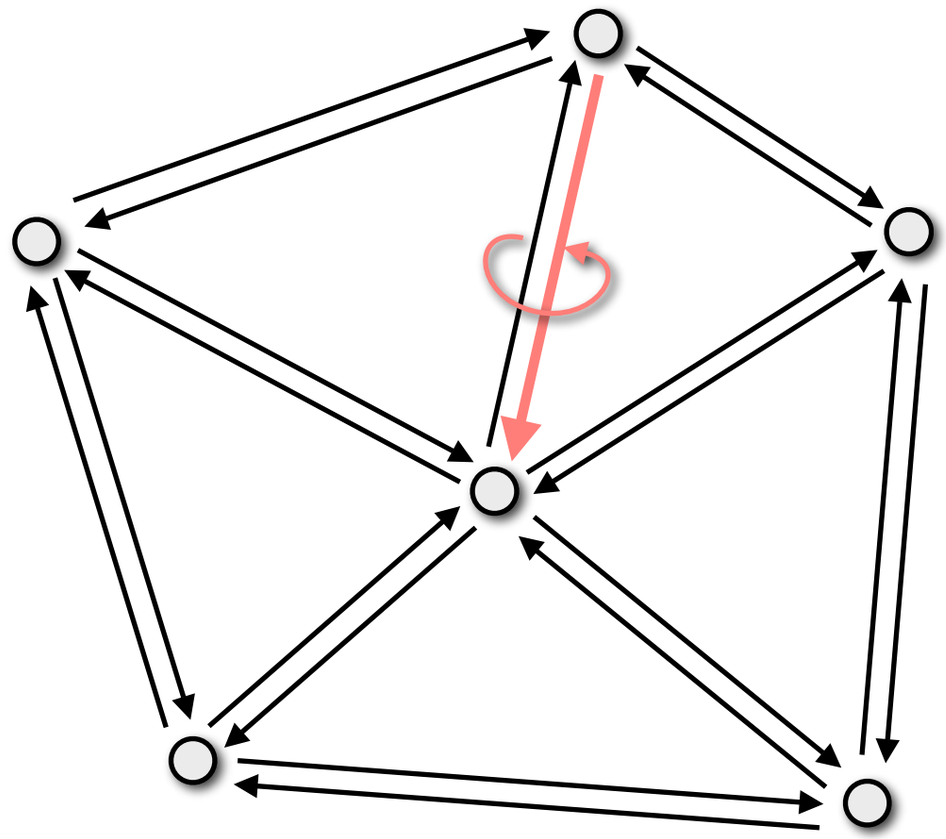
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



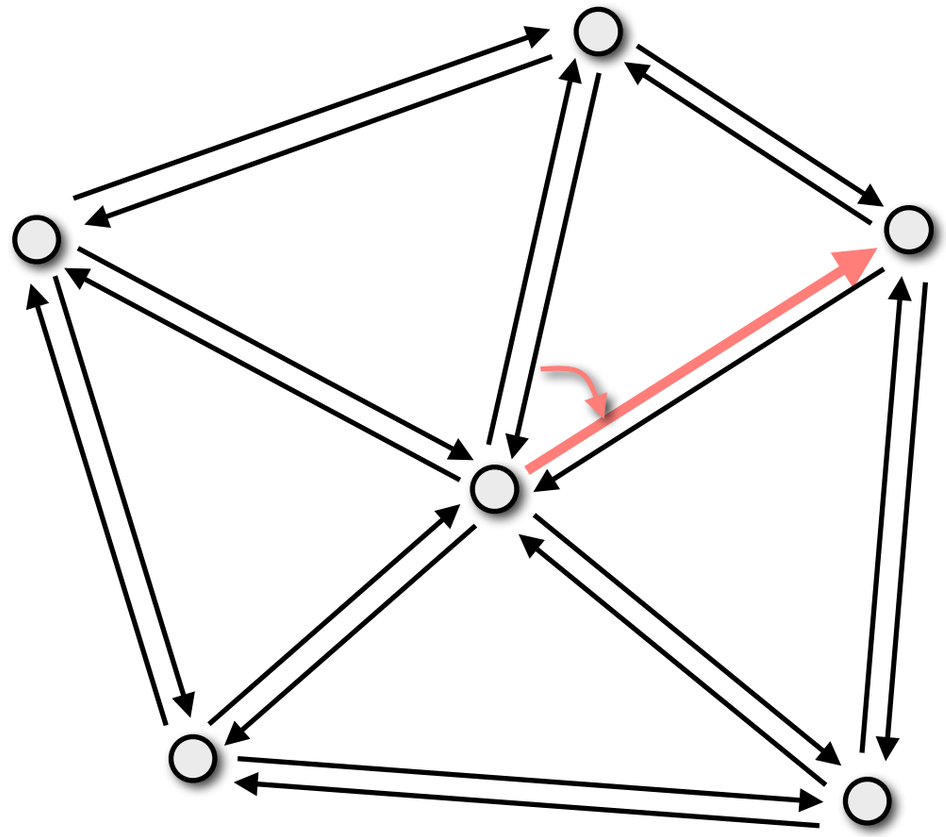
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



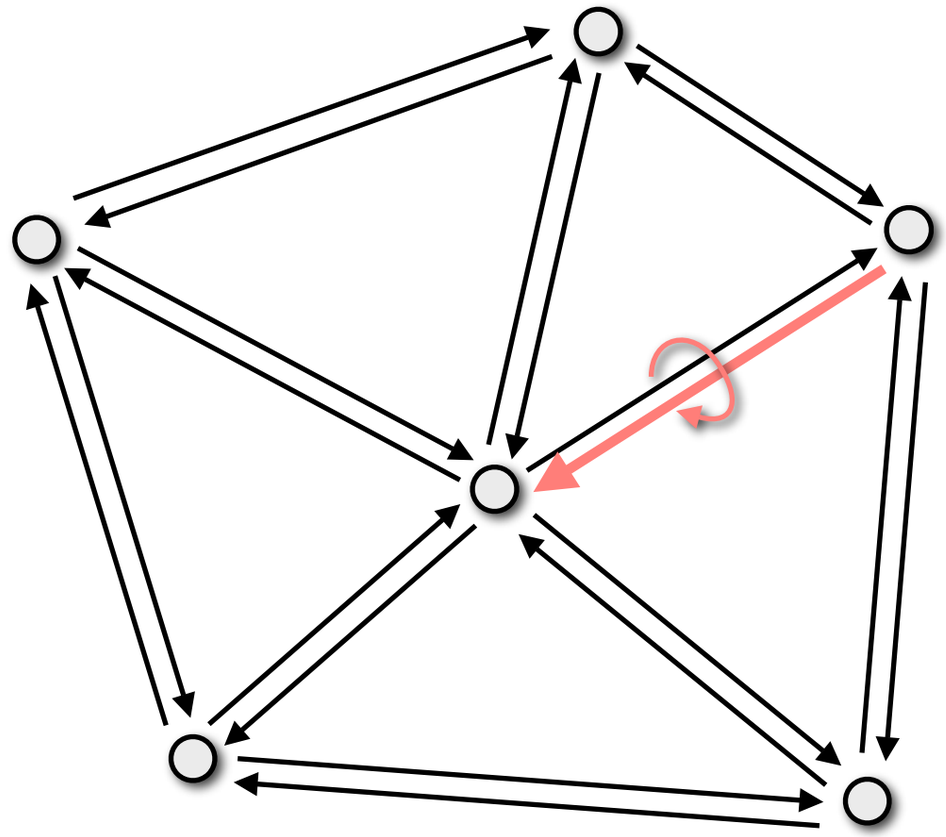
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



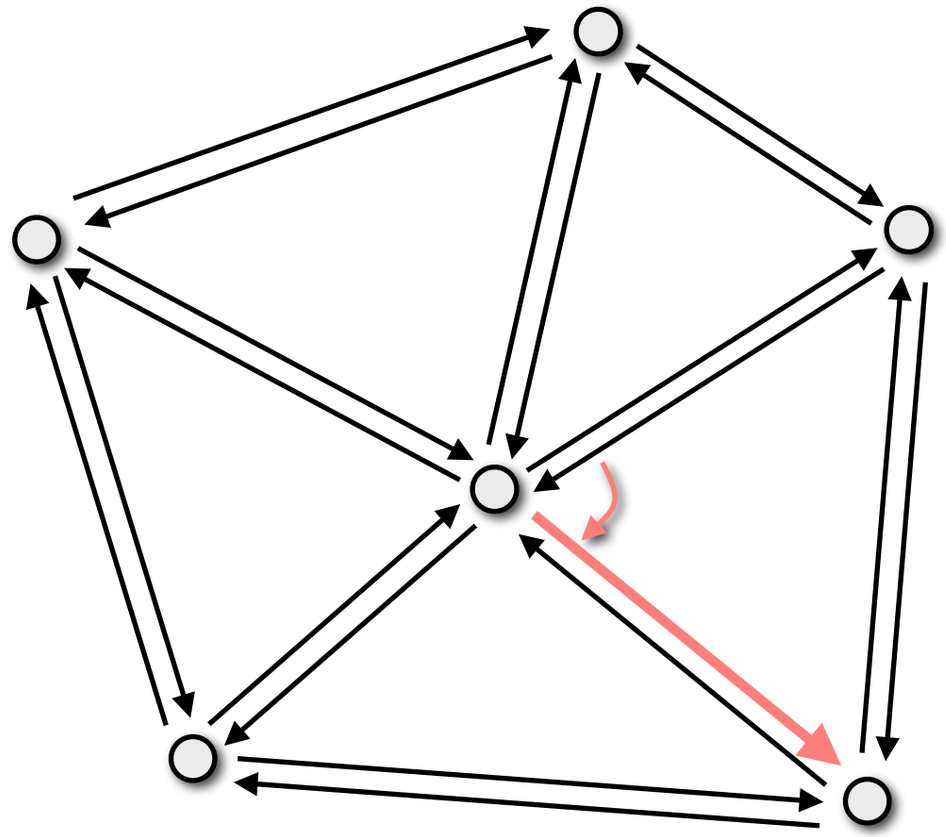
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...



Circulators

- Circulate around a vertex == traverse one-ring

```
int compute_valence(Mesh::VHandle _vh)
{
   int val(0);
   Mesh::VertexVertexIter vv_it;

   for (vv_it=mesh.vv_iter(_vh); vv_it; ++vv_it)
      ++val;

   return val;
}
```

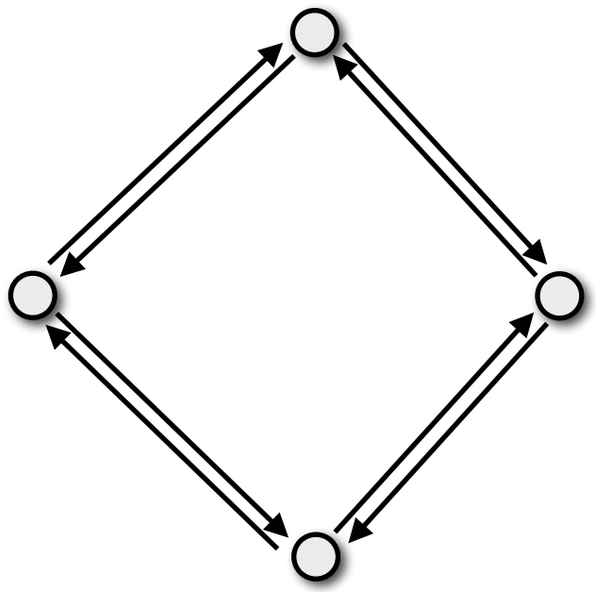
returns false after a complete circulation

- Analogous for VertexFaceIter, VertexEdgeIter, ...

Circulators

- Circulate around a face, enumerate its vertices

```
Mesh::Point compute_center(Mesh::FaceHandle _f)
{
   Mesh::Point    p(0,0,0);
   Mesh::Scalar   count(0);
   Mesh::FaceVertexIter fv_it;

   for (fv_it=mesh.fv_iter(_f); fv_it; ++fv_it)
   {
      p += mesh.point(fv_it);
      ++count;
   }

   return p/count;
}
```



Predefined Properties

```
Mesh::VertexHandle v;
Mesh::FaceHandle   f;

mesh.request_face_normals();
Mesh::Normal n = mesh.normal(f);
mesh.set_normal(f, n);

mesh.request_vertex_colors();
Mesh::Color c = mesh.color(v);
mesh.set_color(v, c);

mesh.request_vertex_status();
bool b = mesh.status(v).locked();
mesh.status(v).set_locked(b);
```

Predefined Properties

- Request, check, and release properties

```
mesh.request_PROPERTY();
mesh.has_PROPERTY();
mesh.release_PROPERTY();
```

- With PROPERTY being one of

```
vertex_{normals,colors,status,texcoords{1D,2D,3D}}
face_{normals,colors,status}
edge_status
halfedge_status
```

Dynamic Custom Properties

- Attach arbitrary types to mesh entities

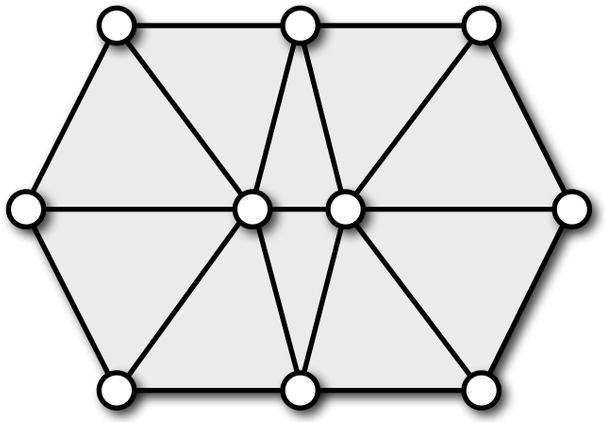
```
OpenMesh::VPropHandleT<int> vvalence;
mesh.add_property(vvalence);

for (v_it=mesh.vertices_begin(); v_it!=vertices_end(); ++v_it)
  mesh.property(vvalence, v_it) = compute_valence(v_it);


OpenMesh::FPropHandleT<Mesh::Point> fcenter;
mesh.add_property(fcenter);

for (f_it=mesh.faces_begin(); f_it!=faces_end(); ++f_it)
  mesh.property(fcenter, f_it) = compute_center(f_it);
```

Modify Geometry

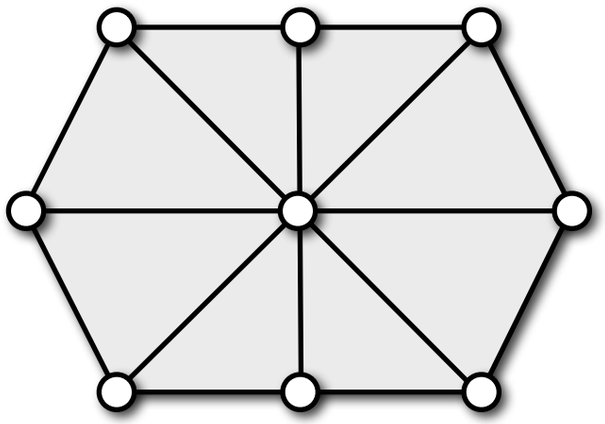
- Change vertex coordinates

```
for (v_it = mesh.vertices_begin();
     v_it != mesh.vertices_end();
     ++v_it)
{
   Mesh::Point p = mesh.point(v_it);
   p = some_transformation(p);
   mesh_.set_point(v_it, p);
}

mesh_.update_normals(); // updates vertex and face normals
```

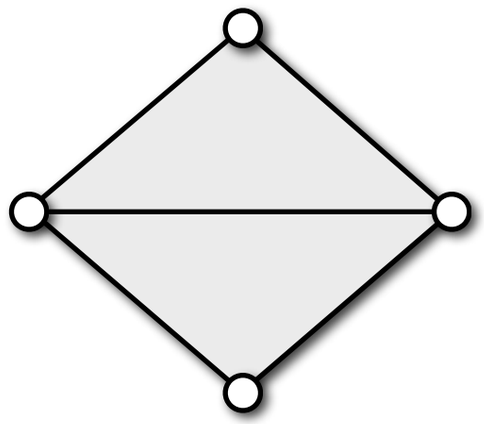
Change Topology



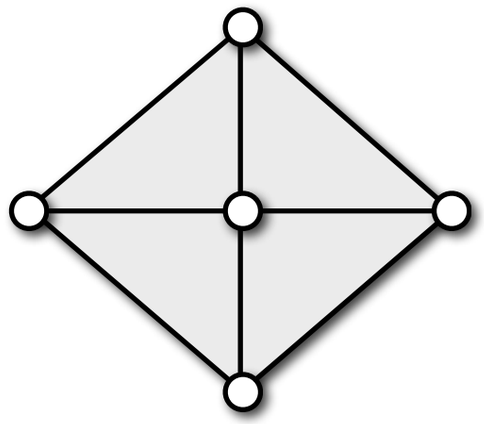
Edge Collapse



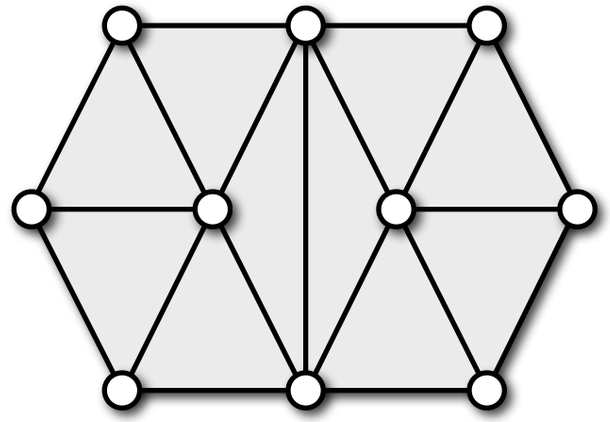
`mesh.collapse(HHandle)`



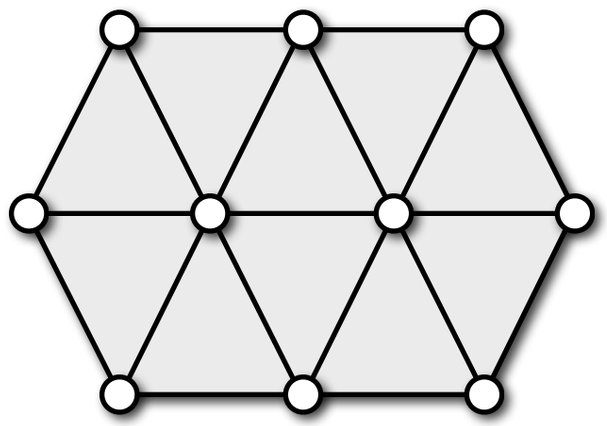
Edge Split



`mesh.split(EHandle,Point)`



Edge Flip



`mesh.flip(EHandle)`

Further Information

- OpenMesh website:
  - http://www.openmesh.org
- Included documentation
  - file://src/OpenMesh/Doc/html/index.html
- Course notes
  - http://graphics.uni-bielefed.de/teaching/ws08/modeling
  - Chaper 3.3 on OpenMesh & CGAL