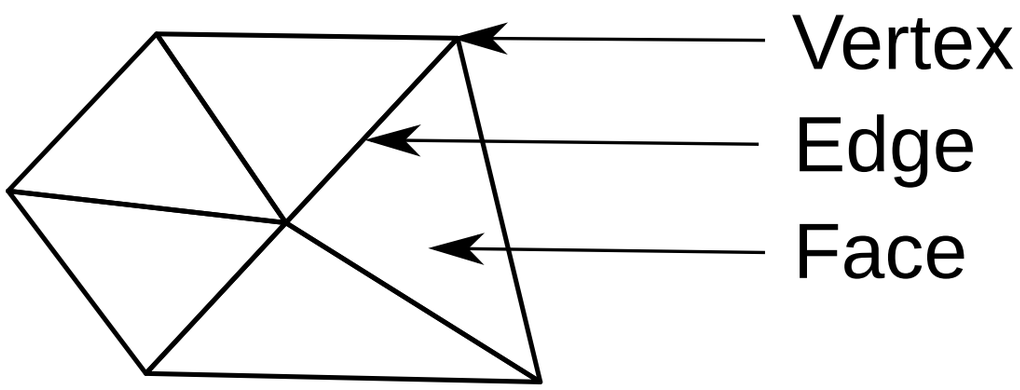
Surface Mesh Intro

Martin Komaritzan

Computer Graphics Group, TU Dortmund

Basics




```
// define Mesh somewhere
SurfaceMesh mesh;
mesh.read("bunny.off");

// loop through all vertices
for(auto v : mesh.vertices()) {...}

// loop through all faces
for(auto f : mesh.faces()) {...}

// loop through all edges
for(auto e : mesh.edges()) {...}

// loop through all halfedges
for(auto h : mesh.halfedges()) {...}
```

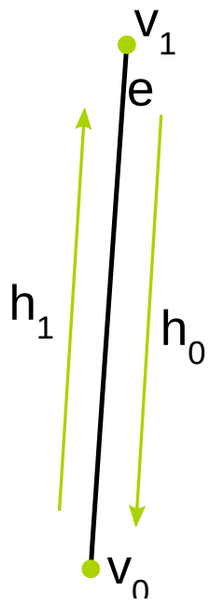
SurfaceMesh Properties

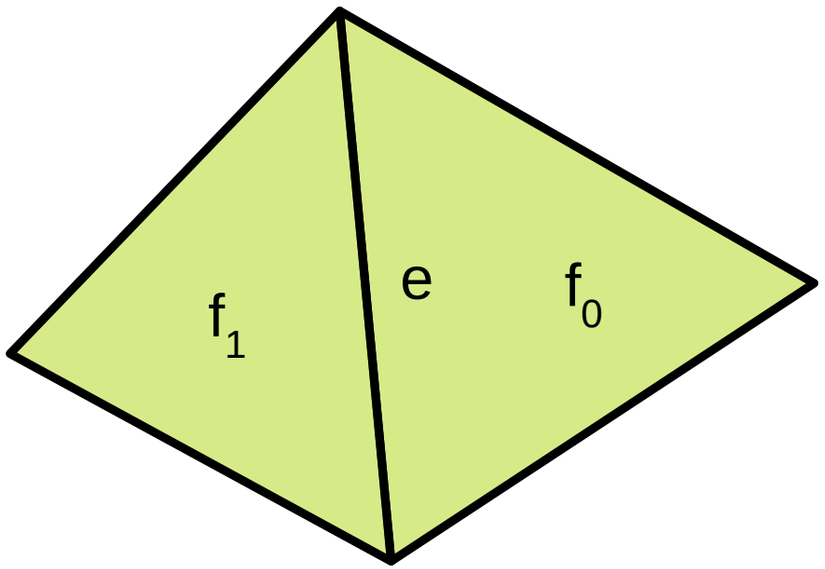
```cpp
// You know std::vector?
// define a vector of integers of size 12
std::vector<int> vec(12);
// set an element
vec[4] = 2;

// properties are very similar but we can define them per Vertex/Edge/Face/Halfedge
// define a vertex property that can strore a float per vertex
// the string at the end is a unique key to store the property in the mesh
auto vvalence = mesh.add_vertex_property<int>("v:valence");

// set the elements - instead of an integer, we use a Vertex
for(auto v : mesh.vertices())
{
    vvalence[v] = mesh.valence(v);
}

// another example, but this time a face property
auto farea = mesh.add_face_property<float>("f:area");
for(auto f : mesh.faces())
{
    farea[f] = triangle_area(mesh,f);
}
```

Vertex ≠ Point

```
// Vertex is basically an index

// this will print out numbers from 0 to n_vertices()
for(auto v : mesh.vertices())
{
    std::cout << v.idx() << std::endl;
}

// define the 100th vertex of any mesh
Vertex v(100);

// get 3D-position of a vertex
Point p100 = mesh.position(v);

// vertex property "v:point" always stores 3D positions
auto points = mesh.get_vertex_property<Point>("v:point");
Point p100 = points[v];
```

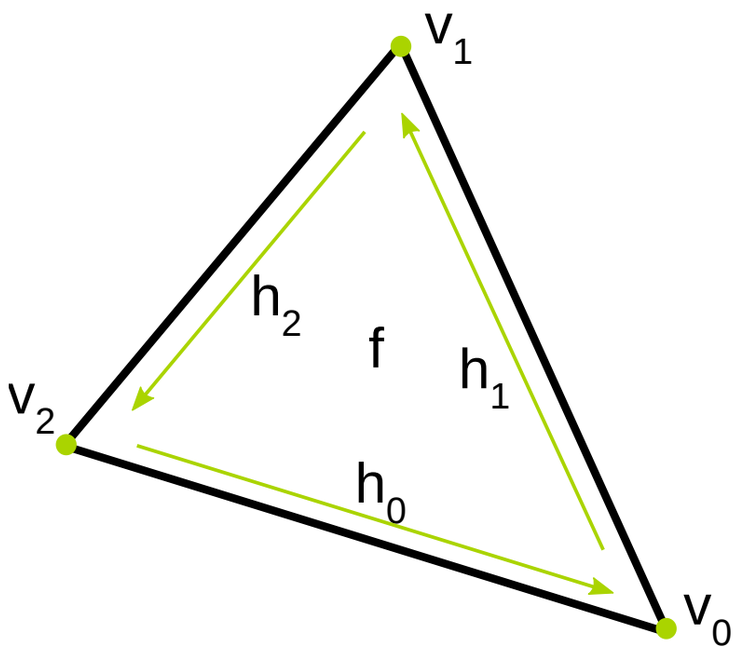
Edges





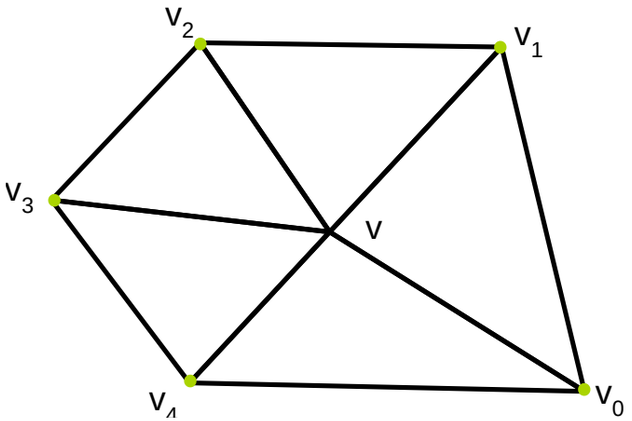

```
// every edge has two corresponding halfedges
Halfedge h0 = mesh.halfedge(e,0);
Halfedge h1 = mesh.halfedge(e,1);
h1 = mesh.opposite_halfedge(h0);

// every edge has two corresponding vertices
Halfedge v0 = mesh.vertex(e,0);
Halfedge v1 = mesh.vertex(e,1);
v0 = mesh.to_vertex(h0);
v1 = mesh.to_vertex(h1);
v0 = mesh.from_vertex(h1);
v1 = mesh.from_vertex(h0);

// every edge has one or two corresponding faces
Face f0 = mesh.face(e,0);
Face f1 = mesh.face(e,1);
```

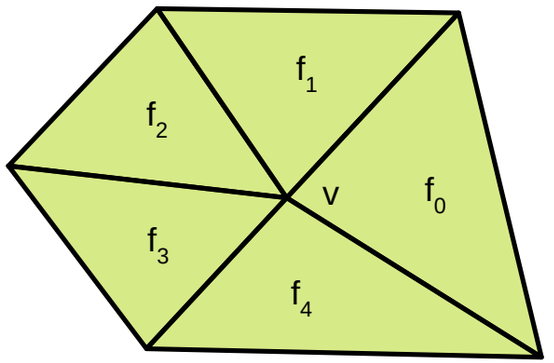
Faces

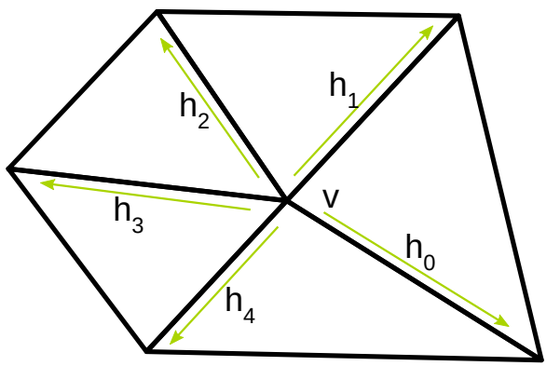



```
// we can get all vertices of a face
for(auto v : mesh.vertices(f))
{
    // loops through v0,v1,v2
}

// sometimes easier but with pointer magic
auto iterator = mesh.vertices(f);
Vertex v0 = *iterator;
Vertex v1 = *(++iterator);
Vertex v2 = *(++iterator);

// we can get all halfedges of a face
for(auto h : mesh.halfedges(f))
{
    // loops through h0,h1,h2
}

// we can get the next halfedge of a face/boundary
Halfedge h1 = mesh.next_halfedge(h0);
Halfedge h2 = mesh.prev_halfedge(h0);

// we can get the face corresponding to a halfedge
Face f = mesh.face(h0);
```

Vertices








```
// loop through the one-ring vertex neighbors
for(auto vv : mesh.vertices(v))
{
    // loops through v0,v1,v2,v3,v4,v5
}
```

```
// loop through all faces sharing a vertex
for(auto fv : mesh.faces(v))
{
    // loops through f0,f1,f2,f3,f4,f5
}
```

```
// loop through all outgoing halfedges of a vertex
for(auto hv : mesh.halfedges(v))
{
    // loops through h0,h1,h2,h3,h4,h5
}
```

Useful Functions

```
// number of vertices in one ring neighborhood to v
int nv = mesh.valence(v);

// total number of vertices, edges, faces, halfedges
int nv = mesh.n_vertices();
int ne = mesh.n_edges();
int nf = mesh.n_faces();
int nh = mesh.n_halfedges();

// check if something is boundary
if(mesh.is_boundary(v)){...}
if(mesh.is_boundary(e)){...}

// note: just one of the edge's halfedges will be boundary if edge is boundary
if(mesh.is_boundary(h)){...}
```