COSC 422 Assignment 2

Non-Photorealistic Rendering

Important Dates

- Due date:

  ~~**Friday, 24th September 11:55pm**~~

  ● **Monday, 27th September 11:55pm**
- Drop-dead date with 15% penalty (3 marks):

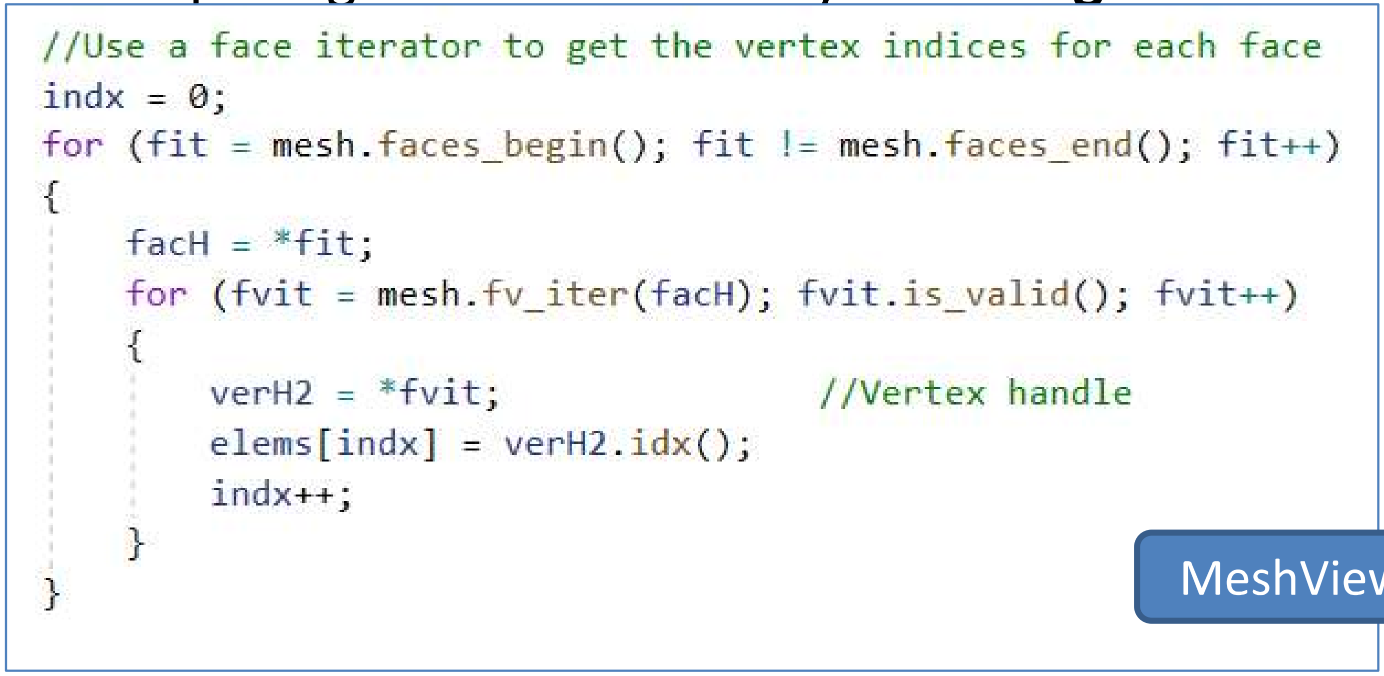
  Friday, 1st October

● Corrections/Clarifications highlighted

OpenMesh

- You may use MeshViewer.cpp (Programming Exercise 11) to load mesh files using the OpenMesh library.
- Please use mesh models containing only triangles. Please do not use highly complex models (> 50,000 triangles)
- https://free3d.com/   (Use only files in OBJ format)

Computing the element array for **triangles:**



```
//Use a face iterator to get the vertex indices for each face
indx = 0;
for (fit = mesh.faces_begin(); fit != mesh.faces_end(); fit++)
{
    facH = *fit;
    for (fvit = mesh.fv_iter(facH); fvit.is_valid(); fvit++)
    {
        verH2 = *fvit;                  //Vertex handle
        elems[indx] = verH2.idx();
        indx++;
    }
}
```

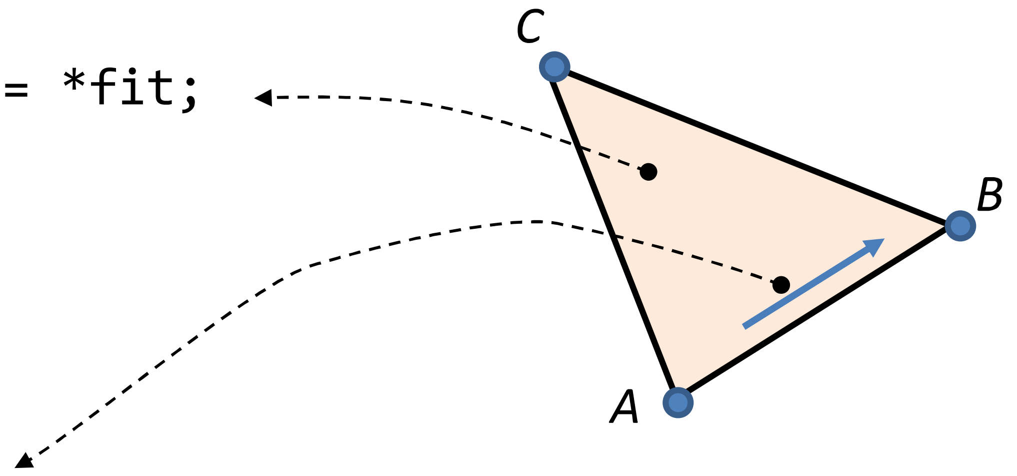
MeshViewer.cpp

Extending MeshViewer.cpp

- Modify the method shown on the previous slide to generate the element array for triangle adjacency primitives
- Render using GL_TRIANGLE_ADJACENCY primitive

  glDrawElements(**GL_TRIANGLES_ADJACENCY**, num_Elems, GL_UNSIGNED_SHORT, NULL);
- Include a function to load textures (loadTGA or DevIL)
- Add a geometry shader to the shader program object (NOTE: A geometry shader must be used for highlighting edges of the model. The two-pass algorithm (slide [6]-18) is not required)
- Include the necessary event callback functions

Processing a Triangle



```
OpenMesh::FaceHandle faH  = *fit;
```

```
OpenMesh::HalfedgeHandle heH = mesh.halfedge_handle(faH);
```

Handle for Vertex A:

```
    OpenMesh::VertexHandle veH1 = mesh.from_vertex_handle(heH);
```

Handle for Vertex B:

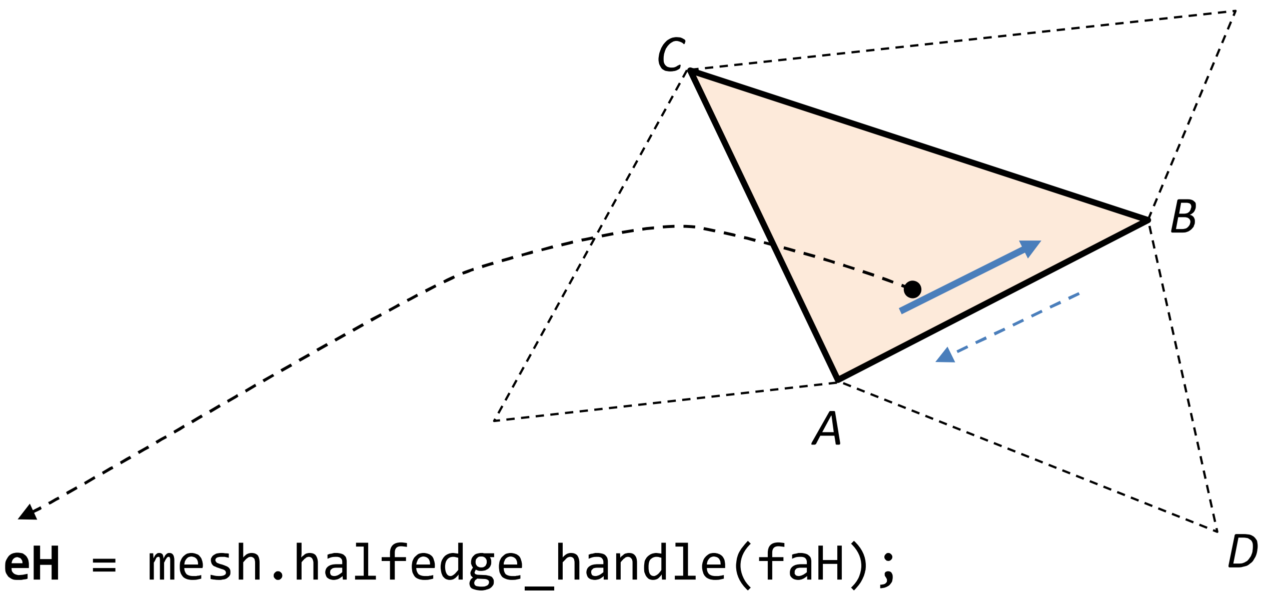
```
    OpenMesh::VertexHandle veH2 = mesh.to_vertex_handle(heH);
```

Handle for Vertex C:

```
    OpenMesh::VertexHandle veH3 = mesh.opposite_vh(heH);
```

Using the .idx() function on a vertex handle gives the index of that vertex.

Triangle Adjacency Primitive



```
OpenMesh::HalfedgeHandle heH = mesh.halfedge_handle(faH);
```

Handle for Vertex A:

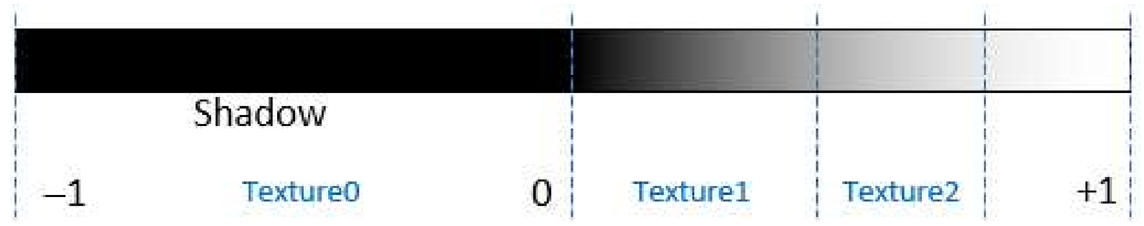
```
  OpenMesh::VertexHandle veH1 = mesh.from_vertex_handle(heH);
  int elem = veH1.idx();
```

Handle for Vertex D:

```
  OpenMesh::VertexHandle veH4 = mesh.opposite_he_opposite_vh(heH);
  elem = ver4.idx();
```

Texture

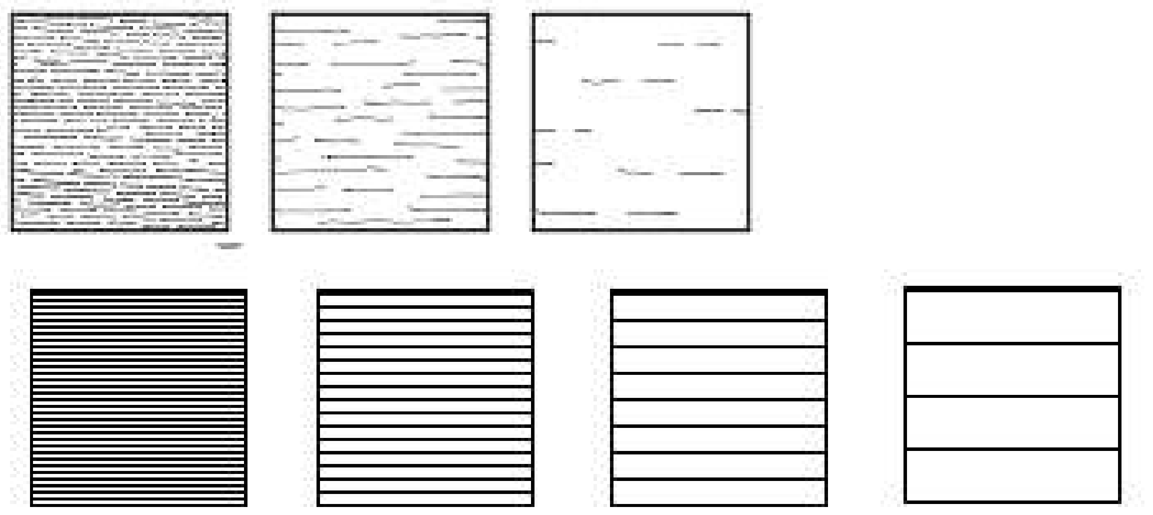
- Use three or four textures corresponding to a discretized set of shade levels



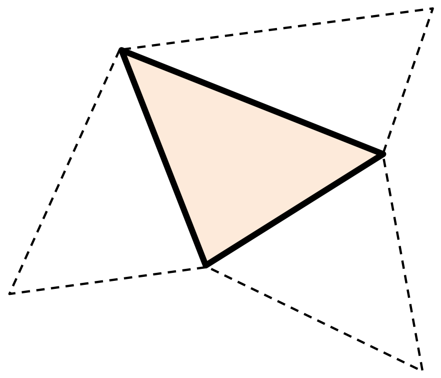
- Preferred size: 64x64

(comparable to the max projected area of a triangle)

- You may use any pattern (cross hatch, pen-and-ink, charcoal etc) suitable for a stylistic rendering of a mesh model. The textures could be procedurally generated or images of hand drawn patterns.



Geometry Shader



```
layout (triangles_adjacency) in;
layout (triangle_strip, max_vertices = 27) out;
```

- Input: 6 vertices in world coordinates (do not multiply vertex position by mvpMatrix in vertex shader)
- Assigns texture coordinates to the vertices of the main triangle.
- Outputs the diffuse term for each vertex of the main triangle
- Creates a triangle strip for silhouette edges and crease edges of the current primitive.

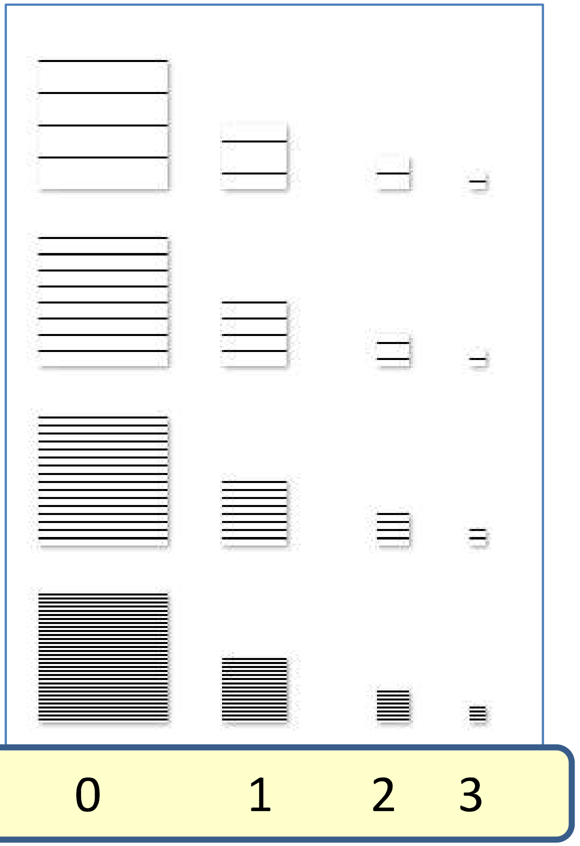
Extra Features

Texture Mipmaps

- Triangles on a mesh model can have varying sizes.
- Suggested mipmap sizes:

  64x64,   32x32,   16x16,   8x8.

  (Max mipmap level = 3)
- loadTGA.h will need to be modified to include mipmap level

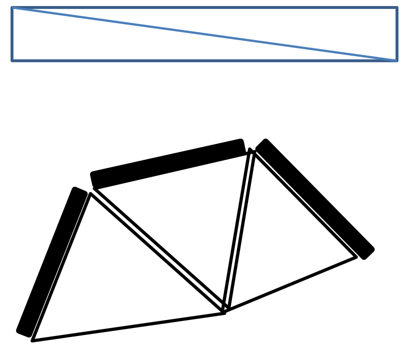


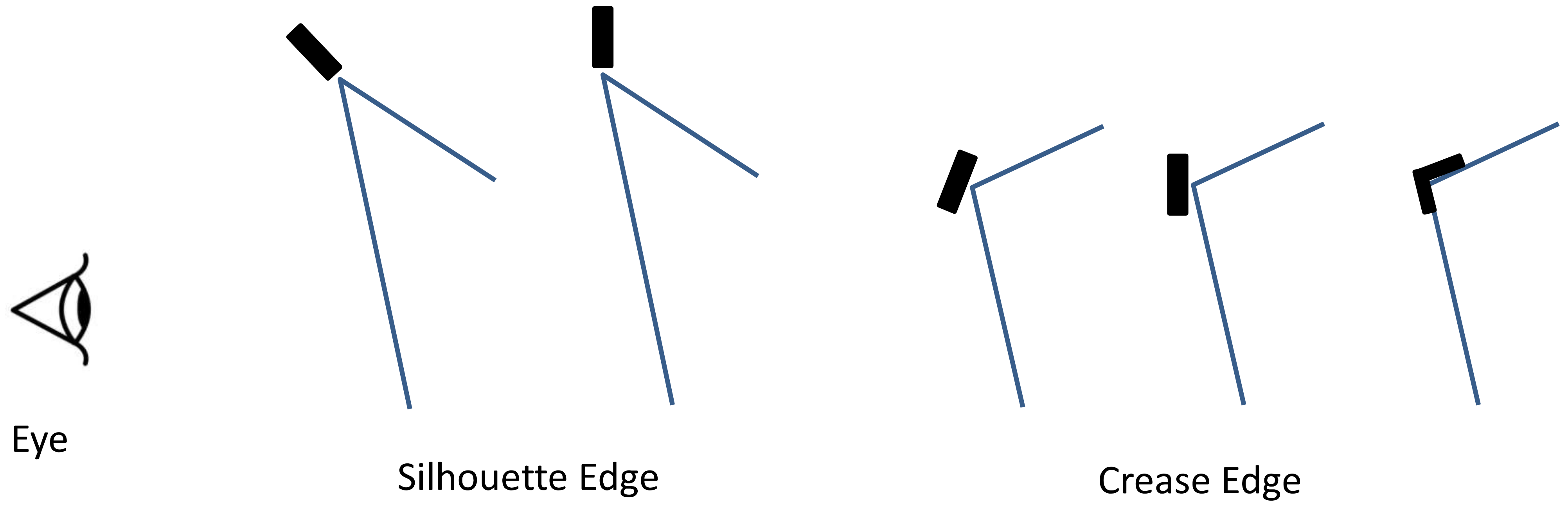
```
void loadTGA_mipmap(string filename, int level)
{
...
...
glTexImage2D(GL_TEXTURE_2D, level, 3, wid, hgt, 0, GL_RGB, GL_UNSIGNED_BYTE, imageData);

}
```

Edge Enhancement

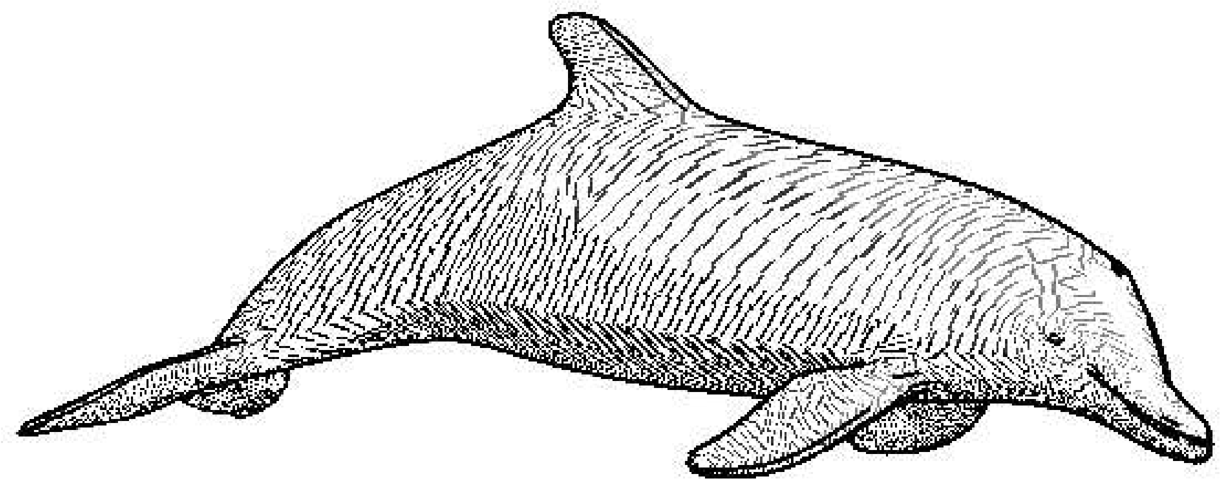
- Methods for rendering silhouette and crease edges are discussed on Slides [6]:25-28. You may suggest suitable improvements of these methods or alternatives for enhancing rendering quality.





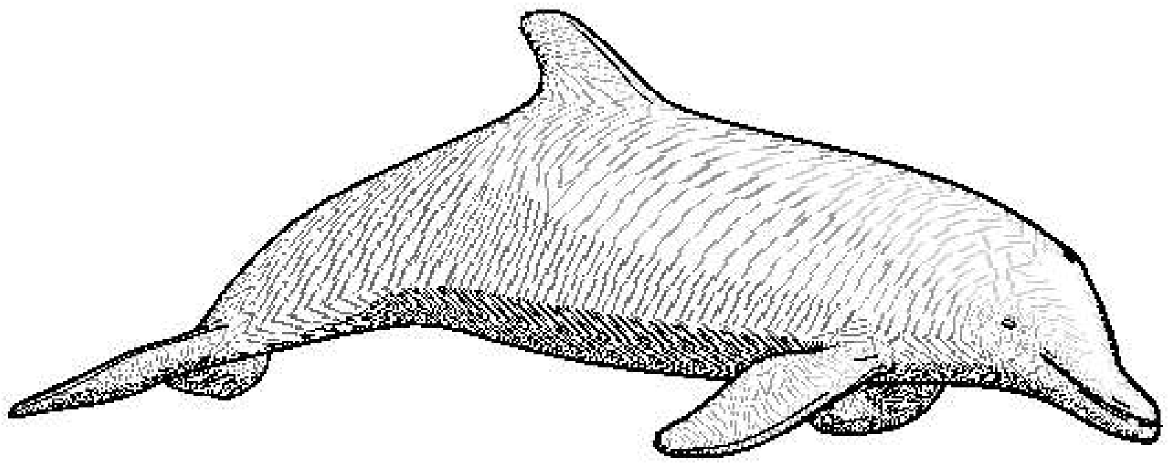
Local Curvature Estimation

- Approximate alignment of pencil stroke lines with the direction of local curvature by identifying the edge with the largest dihedral angle between adjacent triangles (Slide [6]:30).

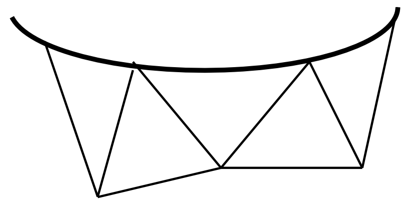


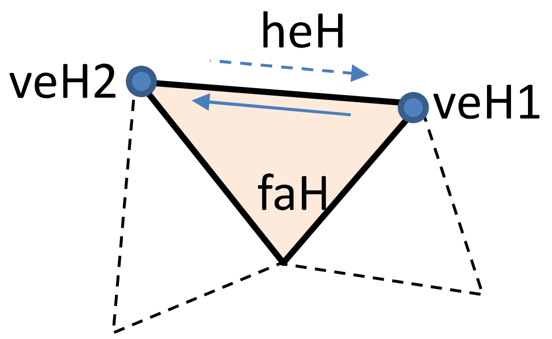
Texture Blending

The appearance of texture lines could be changed by methods such as intensity scaling, modulation and blending in the fragment shader.

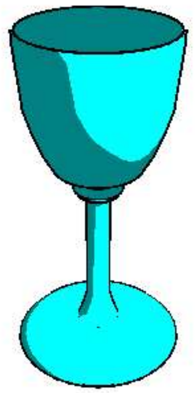


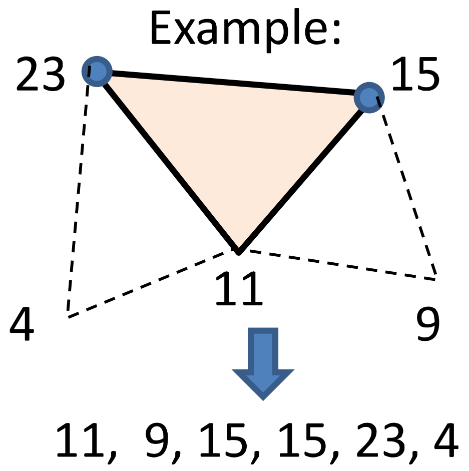
Border Edges





Triangle Adjacency Primitive





- In the above example,

```
mesh.is_boundary(heH)   == true
mesh.is_boundary(veH1)   == true
mesh.is_boundary(veH2)   == true
mesh.is_boundary(faH)   == true
```

- Use a repeated index to represent the index of the missing vertex in the triangle adjacency primitive