CS 543 - Computer Graphics: Polygonal Meshes

by

Robert W. Lindeman

gogo@wpi.edu

(with help from Emmanuel Agu ;-)

3D Modeling

- We talked about basic geometric objects in GLUT
- Wireframe vs. solid models
- Composite objects using SDL file format
- Basic objects
  - Cylinder: **`glutWireCylinder( ), glutSolidCylinder( )`**
  - Cone: **`glutWireCone( ), glutSolidCone( )`**
  - Sphere: **`glutWireSphere( ), glutSolidSphere( )`**
  - Cube: **`glutWireCube( ), glutSolidCube( )`**
  - Newell Teapot
  - Dodecahedron, Torus, *etc.*

Polygonal Meshes

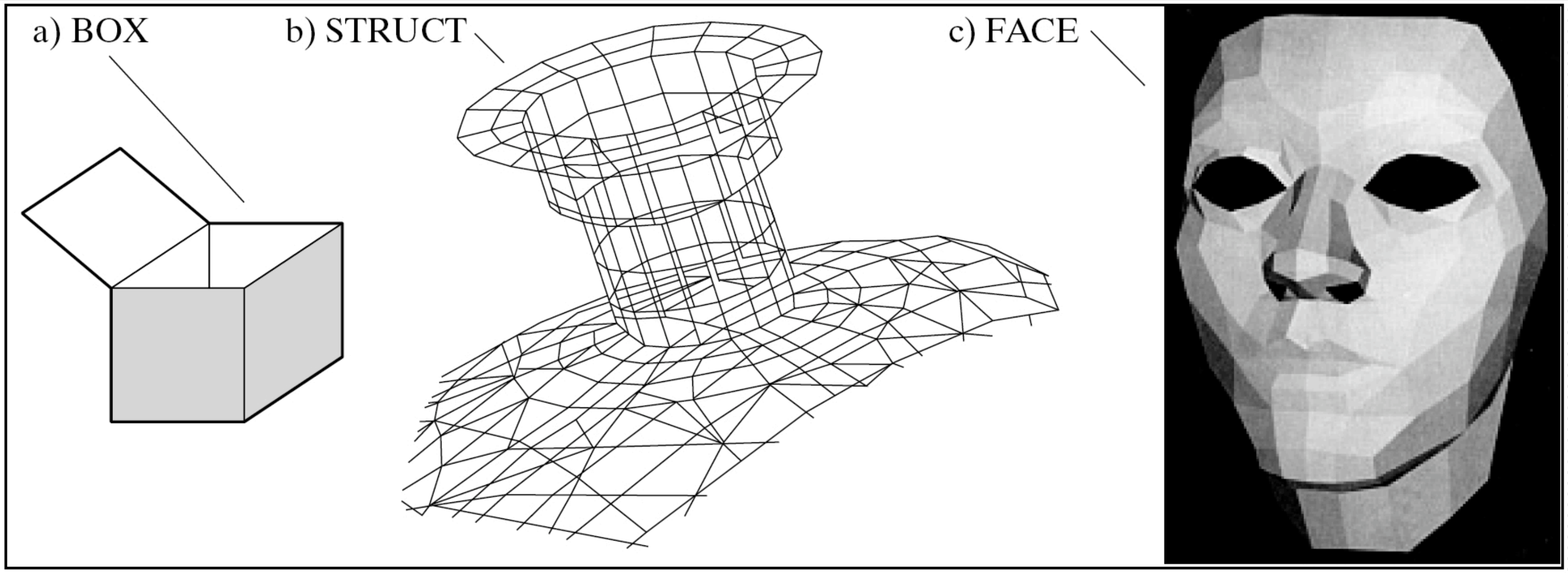
- Modeling with basic shapes (cube, cylinder, sphere, *etc.*) is too primitive
- Difficult to approach realism
- Polygonal meshes
  - Collection of polygons, or faces, that form "skin" of object
  - Offer more flexibility
  - Model complex surfaces better
  - Examples
    - Human face
    - Animal structures
    - Arbitrary curves, *etc.*

Polygonal Meshes (cont.)

- Have become standard in CG
- OpenGL
  - Good at drawing polygons
  - Mesh = sequence of polygons
- Simple meshes are exact (*e.g.*, barn)
- Complex meshes are approximate (*e.g.*, human face)
- Later
  - Use shading technique to smoothen the appearance

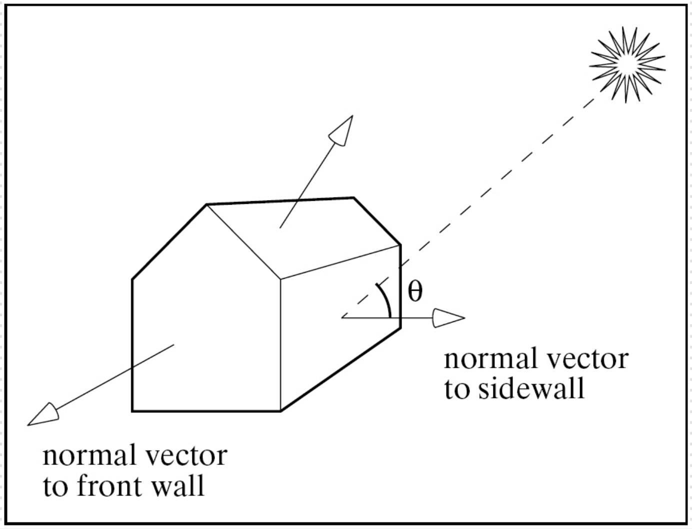
Non-Solid Objects

- □ Examples: box, face
- □ Visualize as infinitely thin *skin*
- □ Meshes to approximate complex objects
- □ Shading used later to smoothen
- □ Non-trivial: creating mesh for complex objects (CAD)



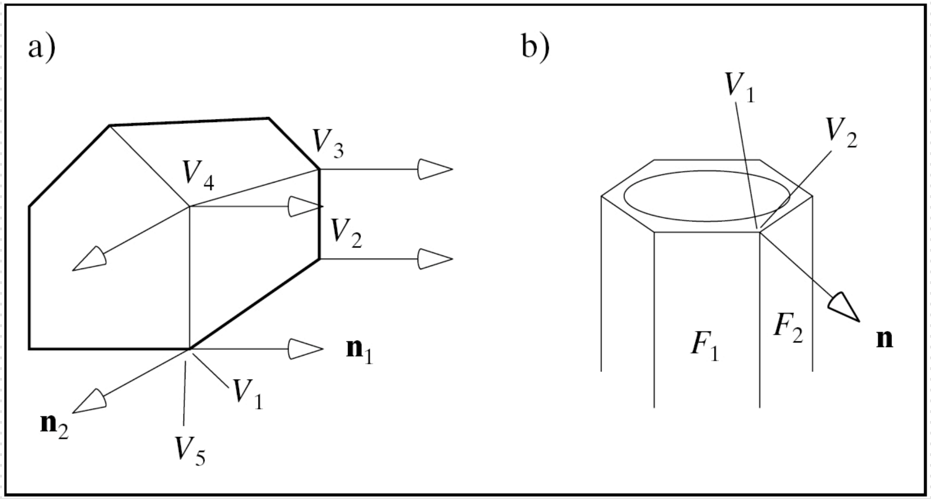
What is a Polygonal Mesh?

- Polygonal mesh defined by
  - List of polygons
  - *Normal* of each polygon
  - Normal vectors used in shading
    - Normal & light vectors determine shading



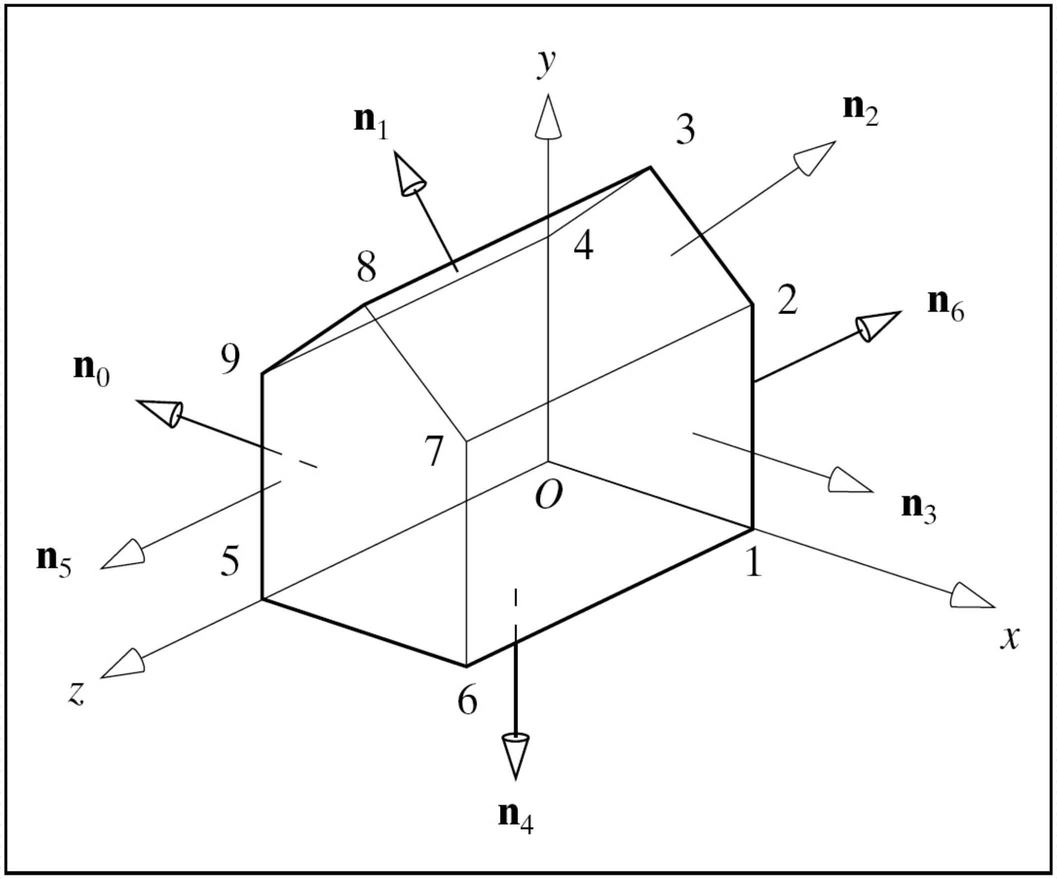
Vertex Normals

- □ Use vertex normal instead of face normal
- □ See advantages later
  - ■ Facilitates clipping / culling
  - ■ Shading of smoothly curved shapes
  - ■ Flat surfaces
    - □ All vertices associated with same **n**
  - ■ Smoothly curved surfaces
    - □ V_1, V_2 with common edge share **n**



Defining a Polygonal Mesh

- Barn example



Defining a Polygonal Mesh

- Three lists:
  - Vertex list
    - Distinct vertices (vertex number, V_x, V_y, V_z)
  - Normal list
    - Normals to faces (normalized n_x, n_y, n_z)
  - Face list
    - Indices into vertex and normal lists. *i.e.*, vertices and normals associated with each face
- Face list convention
  - Traverse vertices *counter-clockwise*
  - Interior on left, exterior on right

Newell Method for Normal Vectors

- Martin Newell at Utah
  - Yeah, the "teapot" guy
- Normal vector
  - Calculation is difficult by hand
  - Given formulae, it is suitable for the computer
  - Compute during mesh generation
- Simple approach used previously
  - Start with any three vertices V_1, V_2, V_3
  - Form two vectors, say V_1-V_2, V_3-V_2
  - Normal = cross product (perpendicular) of vectors

Newell Method for Normal Vectors (cont.)

- Problems with simple approach
  - If two vectors are almost parallel, cross product is small
  - Numerical inaccuracy may result
  - Newell method is more robust
  - Formulae: Normal N = (m_x, m_y, m_z)

$$m_x = \sum_{i=0}^{N-1} \left(y_i - y_{next(i)}\right)\left(z_i + z_{next(i)}\right)$$

$$m_y = \sum_{i=0}^{N-1} \left(z_i - z_{next(i)}\right)\left(x_i + x_{next(i)}\right)$$

$$m_z = \sum_{i=0}^{N-1} \left(x_i - x_{next(i)}\right)\left(y_i + y_{next(i)}\right)$$

Newell Method Example

- Example: Find normal of polygon with vertices

  P_0 = (6,1,4), P_1=(7,0,9) and P_2 = (1,1,2)

- Solution:

  Using simple cross product:

  ((7,0,9)-(6,1,4)) X ((1,1,2)-(6,1,4)) = (2,-23,-5)

  Using Newell method, plug in values, result is the same:

  Normal is (2, -23, -5)

Meshes in Programs

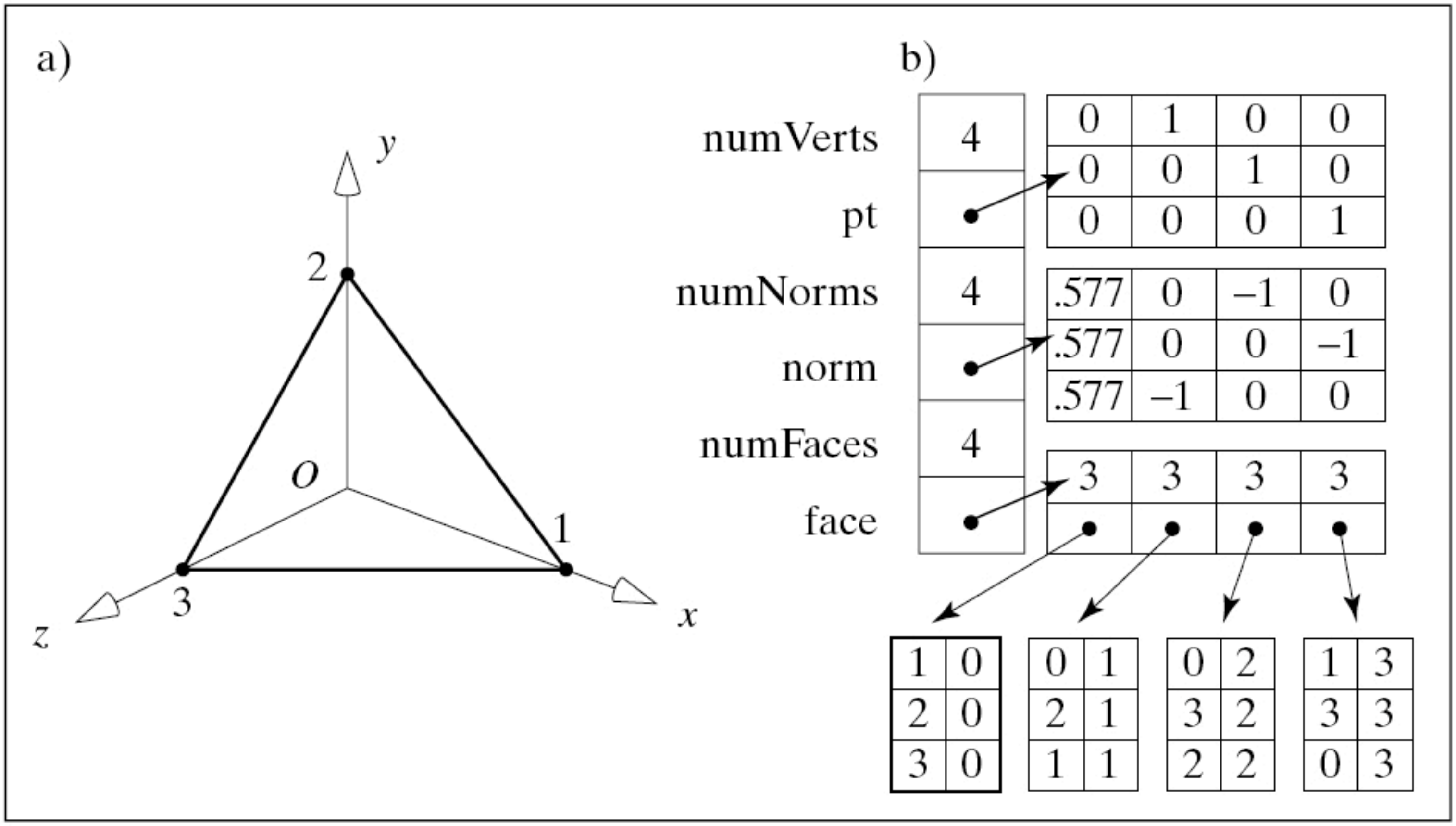
- Class *Mesh*
- Helper classes
  - VertexID
  - Face
- Mesh Object
  - Normal list
  - Vertex list
  - Face list
- Use arrays of pt, norm, face
- Dynamic allocation at runtime
- Array lengths
  - numVerts, numNormals, numFaces

Meshes in Programs (cont.)

- Face
  - Vertex list
  - Normal vector associated with each face
  - Array of index pairs (vertex, normal)
- Example, v^{th} vertex of f^{th} face:
  - Position: pt[face[f].vert[v].vertIndex]
  - Normal vector: norm[face[f].vert[v].normIndex]
- Organized approach, permits random access

Meshes in Programs (cont.)

- Tetrahedron example



numVerts	4
pt	•
numNorms	4
norm	•
numFaces	4
face	•

0	1	0	0
0	0	1	0
0	0	0	1

.577	0	–1	0
.577	0	0	–1
.577	–1	0	0

3	3	3	3
•	•	•	•

1	0
2	0
3	0

0	1
2	1
1	1

0	2
3	2
2	2

1	3
3	3
0	3

Meshes in Programs (cont.)

- Data structures

```
// Vertex ID
class VertexID  {
  public:
    int vertIndex; // index of this vertex in the vertex list
    int normIndex; // index of this vertex’s normal
};
// Face
class Face  {
  public:
    int nVerts;     // number of vertices in this face
    VertexID *vert; // the list of vertex and normal indices
    Face( ){ nVerts = 0; vert = NULL; };     // constructor
    ~Face( ){ nVerts = 0; delete[ ] vert; }; // destructor
};
```

Meshes in Programs (cont.)

```
// Mesh
class Mesh  {
  private:
    int numVerts;   // number of vertices in the mesh
    Point3 *pt;     // array of 3D vertices
    int numNormals; // number of normal vertices for the mesh
    Vector3 *norm;  // array of normals
    int numFaces;   // number of faces in the mesh
    Face *face;     // array of face data
    // ... others to be added later
  public:
    Mesh( );        // constructor
    ~Mesh( );       // destructor
    int readFile( char *fileName ); // to read in mesh file.
    // ... other methods ...
}
```

Drawing Meshes Using OpenGL

□ Pseudo-code

```
for( each face f in Mesh )  {
  glBegin( GL_POLYGON );
    for( each vertex v in face f )  {
      glNormal3f( normal at vertex v );
      glVertex3f( position of vertex v );
    }
  glEnd( );
}
```

Drawing Meshes Using OpenGL (cont.)

- Actual code

```
void Mesh::draw( void )  {  // use openGL to draw this mesh
  for( int f = 0; f < numFaces; f++ )  {
    glBegin( GL_POLYGON );
      // for each vertex of this polygon
      for( int  v = 0; v < face[f].nVerts; v++ )  {
        // index of the normal for this vertex
        int in = face[f].vert[v].normIndex;
        glNormal3f( norm[in].x, norm[in].y, norm[in].z );
        // index of this vertex
        int iv = face[f].vert[v].vertIndex;
        glVertex3f( pt[iv].x, pt[iv].y, pt[iv].z );
      }
    glEnd( );
  }
}
```

Drawing Meshes Using SDL

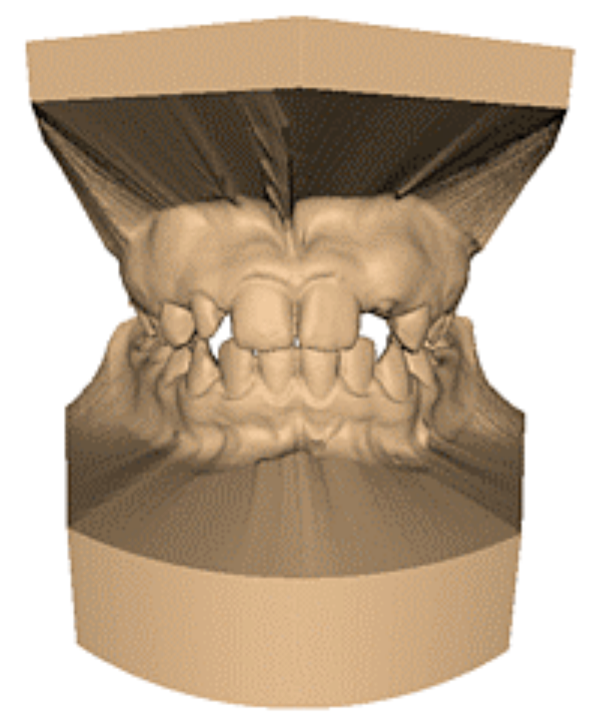
- Scene class reads SDL files
- Accepts keyword *mesh*
- Example
  - Pawn stored in mesh file "pawn.3vn"
  - Add line

```
push translate 3 5 4 scale 3 3 3 mesh pawn.3vn pop
```

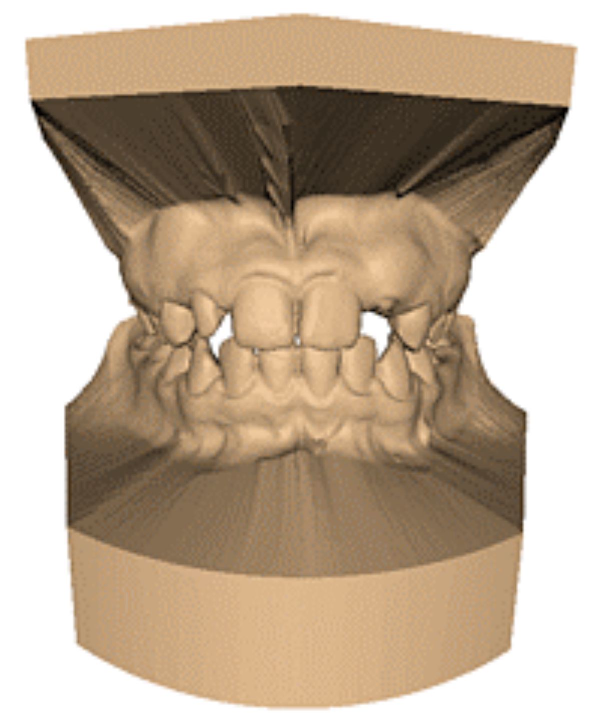
More on Meshes

- Simple meshes are easy to create by hand
- Complex meshes
  - Mathematical functions
  - Algorithms
  - Digitize real objects
- Libraries of meshes available
- Mesh trends
  - 3D scanning
  - Mesh Simplification

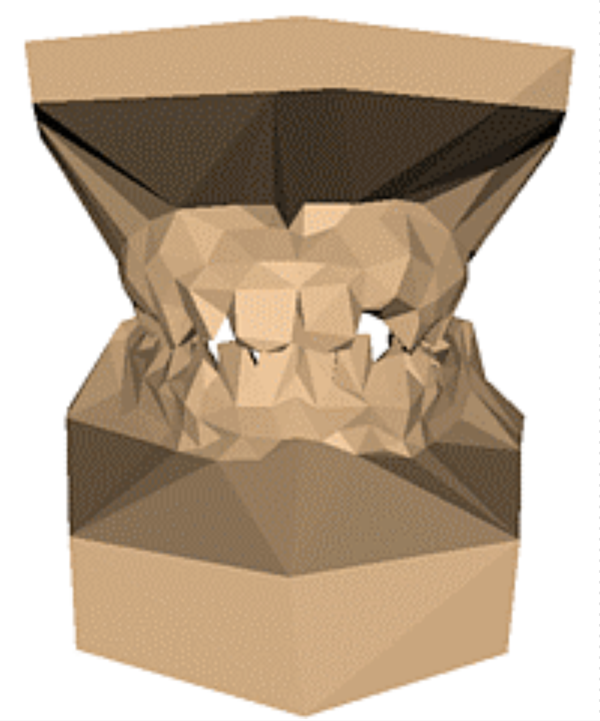
3D Simplification Example



Original: 424,000 triangles



60,000 triangles (14%)



1000 triangles (0.2%)

(courtesy of Michael Garland and Data courtesy of Iris Development.)

References

- Hill: 6.1-6.2