Chapter 6.1-3
Modeling Shapes with Polygonal Meshes

S. M. Lea
University of North Carolina at Greensboro
© 2007, Prentice Hall
3D Modeling

• Polygonal meshes capture the shape of complex 3D objects in simple data structures.
  – Platonic solids, the Buckyball, geodesic domes, prisms.
  – Extruded or swept shapes, and surfaces of revolution.
  – Solids with smoothly curved surfaces.

• Animated Particle systems: each particle responds to conditions.

• Physically based systems: the various objects in a scene are modeled as connected by springs, gears, electrostatic forces, gravity, or other mechanisms.
Particle Systems Example

- Particle system showing water droplets in a fountain. (Courtesy of Philipp Crocoll);
  Starfield simulation (Courtesy of Ge Wang)
Polygonal Meshes

- A polygonal mesh is a collection of polygons (faces) that approximate the surface of a 3D object.
  - Examples: surfaces of sphere, cone, cylinder made of polygons (Ch. 5); barn (below).
Polygonal Meshes (2)

• Polygons are easy to represent (by a sequence of vertices) and transform.
• They have simple properties (a single normal vector, a well-defined inside and outside, etc.).
• They are easy to draw (using a polygon-fill routine, or by mapping texture onto the polygon).
Polygonal Meshes (3)

• Meshes are a standard way of representing 3D objects in graphics.
• A mesh can approximate the surface to any degree of accuracy by making the mesh finer or coarser.
• We can also smooth the polygon edges using rendering techniques.
Polygonal Meshes (4)

• Meshes can model both solid shapes and thin skins.
  – The object is **solid** if the polygonal faces fit together to enclose space.
  – In other cases, the faces fit together without enclosing space, and so they represent an infinitesimally thin surface.

• In both cases we call the collection of polygons a **polygonal mesh** (or simply a **mesh**).
A polygonal mesh is described by a list of polygons, along with information about the direction in which each polygon is facing.

If the mesh represents a solid, each face has an inside and an outside relative to the rest of the mesh.

In such a case, the directional information is often simply the outward pointing normal vector to the plane of the face used in the shading process.
Polygonsal Meshes (6)

- The normal direction to a face determines its brightness.
Polygonal Meshes (7)

• For some objects, we associate a normal vector to each vertex of a face rather than one vector to an entire face.
  – We use meshes, which represent objects with smoothly curved faces such as a sphere or cylinder. We will refer to the faces of such objects, but with the idea that there is a “smooth-underlying surface”.
  – When we display such an object, we will want to de-emphasize the individual faces of the object in order to make the object look smooth.
Polygonal Meshes (8)

- Each vertex \( V_1, V_2, V_3, \) and \( V_4 \) defining the side wall of the barn has the *same* normal \( \mathbf{n}_1 \), the normal vector to the side wall.
- But vertices of the front wall, such as \( V_5 \), will use normal \( \mathbf{n}_2 \). (Note that vertices \( V_1 \) and \( V_5 \) are located at the same point in space, but use different normals.)
For the smoothly curved surface of the cylinder, both vertex $V_1$ of face $F_1$ and vertex $V_2$ on face $F_2$ use the same normal $n$, the vector perpendicular to the underlying smooth surface.
Defining a Polygonal Mesh

• A mesh consists of 3 lists: the vertices of the mesh, the outside normal at each vertex, and the faces of the mesh.

• Example: the basic barn has 7 polygonal faces and 10 vertices (each shared by 3 faces).
Defining a Polygonal Mesh (2)

- It has a square floor one unit on a side.
- Because the barn has flat walls, there are only 7 distinct normal vectors involved, the normal to each face as shown.
Defining a Polygonal Mesh (3)

- The vertex list reports the locations of the distinct vertices in the mesh.
- The list of normals reports the directions of the distinct normal vectors that occur in the model.
- The face list indexes into the vertex and normal lists.
## Vertex List for the Barn

<table>
<thead>
<tr>
<th>vertex</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>1.5</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Normal List for the Barn

- The normal list (as unit vectors, to the 7 basic planes or polygons).

<table>
<thead>
<tr>
<th>normal</th>
<th>n_x</th>
<th>n_y</th>
<th>n_z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-0.707</td>
<td>0.707</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.707</td>
<td>0.707</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>
## Face List for the Barn

<table>
<thead>
<tr>
<th>Face</th>
<th>Vertices</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (left)</td>
<td>0, 5, 9, 4</td>
<td>0,0,0,0</td>
</tr>
<tr>
<td>1 (roof left)</td>
<td>3, 4, 9, 8</td>
<td>1,1,1,1</td>
</tr>
<tr>
<td>2 (roof right)</td>
<td>2, 3, 8, 7</td>
<td>2, 2, 2,2</td>
</tr>
<tr>
<td>3 (right)</td>
<td>1, 2, 7, 6</td>
<td>3, 3, 3, 3</td>
</tr>
<tr>
<td>4 (bottom)</td>
<td>0, 1, 6, 5</td>
<td>4, 4, 4, 4</td>
</tr>
<tr>
<td>5 (front)</td>
<td>5, 6, 7, 8, 9</td>
<td>5, 5, 5, 5, 5</td>
</tr>
<tr>
<td>6 (back)</td>
<td>0, 4, 3, 2, 1</td>
<td>6, 6, 6, 6, 6</td>
</tr>
</tbody>
</table>
Defining a Polygonal Mesh (4)

• Only the indices of the vertices and normals are used.
• The list of vertices for a face begins with any vertex in the face, and then proceeds around the face vertex by vertex until a complete circuit has been made.
  – There are two ways to traverse a polygon: clockwise and counterclockwise. For instance, face #5 above could be listed as (5, 6, 7, 8, 9) or (9, 8, 7, 6, 5).
  – Convention: *Traverse the polygon counterclockwise as seen from outside the object.*
Defining a Polygonal Mesh (5)

- Using this order, if you traverse around the face by walking from vertex to vertex, the inside of the face is on your left.
- Using the convention allows algorithms to distinguish with ease the front from the back of a face.
- If we use an underlying smooth surface, such as a cylinder, normals are computed for that surface.
Properties of Meshes

• A closed mesh represents a solid object (which encloses a volume).
• A mesh is connected if there is an unbroken path along the edges of the mesh between any two vertices.
• A mesh is simple if it has no holes. Example: a sphere is simple; a torus is not.
• A mesh is planar if every face is a plane polygon. Triangular meshes are frequently used to enforce planarity.
Properties of Meshes (2)

• A mesh is convex if the line connecting any two interior points is entirely inside the mesh.
• Exterior connecting lines are shown for non-convex objects below (step and torus).
Meshes for Drawing Non-physical Objects

- The figure labeled IMPOSSIBLE looks impossible but is not.
- This object can be represented by a mesh.
- Gershon Elber’s web site (http://www.cs.technion.ac.il/~gershon/EscherForReal/) presents a collection of physically impossible objects, and describes how they can be modeled and drawn.
“Thin-skin” Meshes Representing Non-solid Objects
Working with Meshes in a Program

- We want an efficient Mesh class that makes it easy to create and draw the object.
- Since mesh data is frequently stored in a file, we also need simple ways to read and write mesh files.
- Code for classes VertexID, Face, and Mesh is in Fig. 6.15.
Meshes in a Program (2)

• The Face data type is a list of vertices and the normal vector associated with each vertex in the face.
• It is an array of index pairs; the normal to the $v^{th}$ vertex of the $f^{th}$ face has value $\text{norm[face[f].vert[v].normIndex]}$.
• This indexing scheme is quite orderly and easy to manage, and it allows rapid random access indexing into the pt[] array.
Example (tetrahedron & representation)
Drawing the Mesh Object

• **Mesh::draw()** (Fig. 6.17) traverses the array of faces in the mesh object, and for each face sends the list of vertices and their normals down the graphics pipeline.

• In OpenGL, to specify that subsequent vertices are associated with normal vector **m**, execute `glNormal3f(m.x, m.y, m.z)`.

• For proper shading, these vectors must be normalized. Otherwise, place `glEnable(GL_NORMALIZE)` in the `init()` function. This requests that OpenGL automatically normalize all normal vectors.
SDL and Meshes

• To use SDL, simply develop the Mesh class from the Shape class (as SDL does for you) and add the method `drawOpenGL()`. The book’s companion web site gives full details on the Shape class and SDL’s supporting classes.

• The Scene class that reads SDL files is already set to accept the keyword `mesh`, followed by the name of the file that contains the mesh description: e.g., `mesh pawn.3vn`
Using SDL to Create and Draw Meshes

• The mesh data are in a file with suffix .3vn.
• The first line of the file lists the number of vertices, number of normals, and number of faces, separated by whitespace.
• The second line begins the list of vertices, giving their x, y and z coordinates separated by whitespace.
  – Multiple vertex coordinates may be listed on a single line.
Using SDL (2)

• After all vertices have been listed, the list of normals begins. A normal is specified by nx, ny, and nz, separated by whitespace.
  – Multiple normal values may be listed on a single line.

• The list of faces follows. A face is specified by the number of vertices it has, the list of vertex indices (in counter-clockwise order from outside), and the list of normal indices (same order as the vertex indices).
Using SDL (3)

• We can also use the matrix manipulation functions of SDL to position and orient the mesh drawing.

• Example:
  – push translate 3 5 4 scale 3 3 3 mesh pawn.3vn pop
Meshes for Polyhedra

• Polyhedron: connected mesh of simple planar polygons that encloses a finite volume.
  – Every edge is shared by exactly 2 faces.
  – At least 3 edges meet at each vertex.
  – Faces do not interpenetrate. They touch either not at all, or only along their common edge.
  – Euler's formula: $V + F - E = 2$ for a simple polyhedron.
Schlegel Diagrams

• A **Schlegel diagram** reveals the structure of a polygon.

• It views the polyhedron from a point just outside the center of one of its faces.

• The front face appears as a large polygon surrounding the rest of the faces.
Schlegel Diagrams (2)

- Part a) shows the Schlegel diagram of a pyramid, and parts b) and c) show two different Schlegel diagrams for the basic barn. (Which faces are closest to the eye?).

a).  

b).  

c).
Prisms

- A prism is formed by moving a regular polygon along a straight line.
- When the line is perpendicular to the polygon, the prism is a right prism.
Platonic Solids

• All the faces are identical and each is a regular polygon.
Every Platonic solid P has a dual Platonic solid D. The vertices $v_i$ of D are the centers of faces of P, calculated as

$$c = \frac{1}{n} \sum_{i=0}^{n-1} v_i$$

The duals are tetrahedron-tetrahedron, hexahedron-octahedron, dodecahedron-icosahedron.
Flattened Models

- To keep track of vertex and face numbering, we use a flat **model**, which is made by cutting along certain edges of each solid and unfolding it to lie flat.
Normal Vectors for the Platonic Solids

- Normals can be found using Newell’s method.
- Also, because of the high degree of symmetry of a Platonic solid, *if the solid is centered at the origin*, the normal vector to each face is the vector from the origin to the *center* of the face (the average of the vertices of the face).
Vertex and Face lists for a Tetrahedron

- For the unit cube having vertices \((\pm 1, \pm 1, \pm 1)\), and the tetrahedron with one vertex at \((1,1,1)\), the tetrahedron has vertex and face lists given below.

<table>
<thead>
<tr>
<th>Vertex list</th>
<th>Face list</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertex</td>
<td>x y z</td>
</tr>
<tr>
<td>0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1 -1 -1</td>
</tr>
<tr>
<td>2</td>
<td>-1 -1 1</td>
</tr>
<tr>
<td>3</td>
<td>-1 1 -1</td>
</tr>
</tbody>
</table>
Icosahedron

- This figure shows that three mutually perpendicular golden rectangles inscribe the icosahedron. A vertex list may be read directly from this picture.
Icosahedron (2)

- We choose to align each golden rectangle with a coordinate axis. For convenience, we define one rectangle so that its longer edge extends from -1 to 1 along the x-axis, and its shorter edge extends from $-\varphi$ to $\varphi$, where $\varphi = 0.618$ is the reciprocal of the golden ratio $\Phi$. 
**Vertex List for the Icosahedron**

<table>
<thead>
<tr>
<th>Vertex</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>Vertex</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>φ</td>
<td>6</td>
<td>φ</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-φ</td>
<td>7</td>
<td>-φ</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>φ</td>
<td>0</td>
<td>8</td>
<td>φ</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-φ</td>
<td>0</td>
<td>9</td>
<td>-φ</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>-φ</td>
<td>10</td>
<td>-1</td>
<td>φ</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-1</td>
<td>φ</td>
<td>11</td>
<td>-1</td>
<td>-φ</td>
<td>0</td>
</tr>
</tbody>
</table>
Flattened Model for Icosahedron
Prism Model for Icosahedron

5 faces for pyramidal cap
10 faces make a prism
5 faces for pyramidal base
Flattened Model for Dodecahedron
Archimedean Solids

- Have more than one type of polygon as faces; semi-regular.
- Examples: truncated cube (octagon and triangle)
Truncated Cube

• Each edge of the cube is divided into three parts; the middle part of length $A = \frac{1}{1 + \sqrt{2}}$ and the middle portion of each edge is joined to its neighbors.

• Thus if an edge of the cube has endpoints $C$ and $D$, two new vertices, $V$ and $W$, are formed as the affine combinations

\[ V = \frac{1 + A}{2} C + \frac{1 - A}{2} D \quad W = \frac{1 - A}{2} C + \frac{1 + A}{2} D \]
Number of Archimedean Solids

• Given the constraints that faces must be regular polygons, and that they must occur in the same arrangement about each vertex, there are only 13 possible Archimedean solids.

• Archimedean solids have sufficient symmetry that the normal vector to each face is found using the center of the face.
Truncated Icosahedron

- The truncated icosahedron (soccer ball) consists of regular hexagons and pentagons.
- More recently this shape has been named the Buckyball after Buckminster Fuller, because of his interest in geodesic structures similar to this.
- Crystallographers have discovered that 60 atoms of carbon can be arranged at the vertices of the truncated icosahedron, producing a new kind of carbon molecule that is neither graphite nor diamond.
- The material has been named Fullerene.
The Buckyball and Flattened Version (Partial)