Computer Graphics

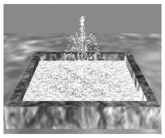
Modeling Shapes with Polygonal Meshes 6.1-3

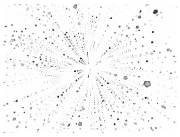
3D Modeling

- 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)



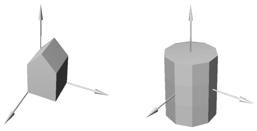


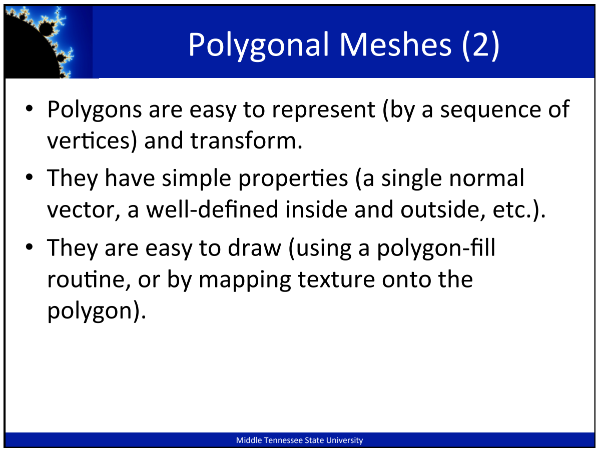
Particle system animations

- http://www.youtube.com/watch?v=DYp9NptNYdY&feature=related
- http://www.youtube.com/watch?v=9B9i4vjj5D4&feature=related

Polygonal Meshes

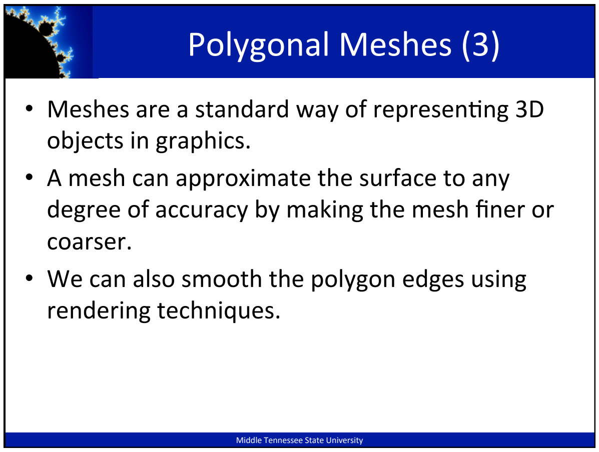
- A polygonal mesh is a collection of polygons (faces) that approximate the surface of a 3D object.
  - Examples: surfaces of sphere, cone, cylinder made of polygons; barn.





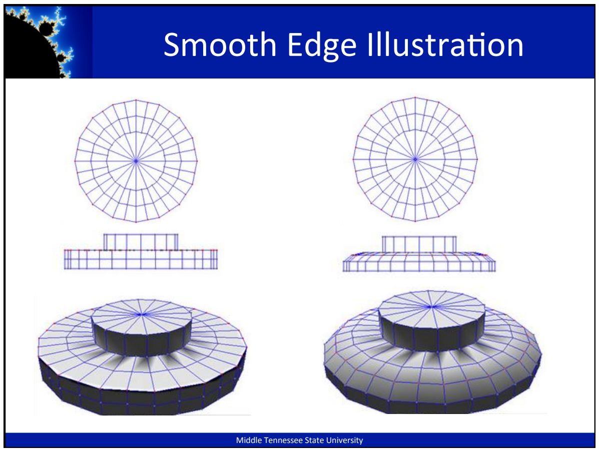
Polygonal Meshes (2)

- 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).

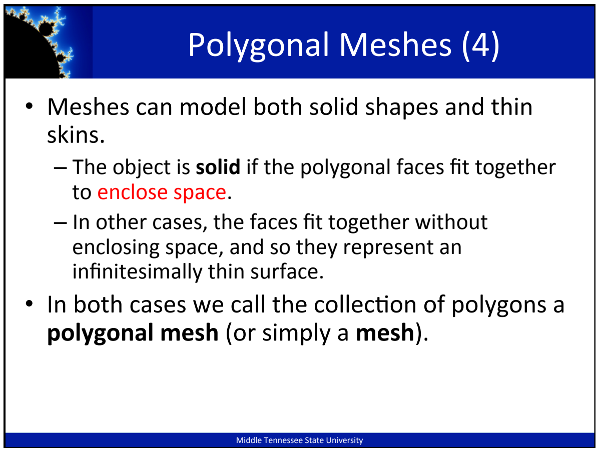


Polygonal Meshes (3)

- 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.



Smooth Edge Illustration



Polygonal Meshes (4)

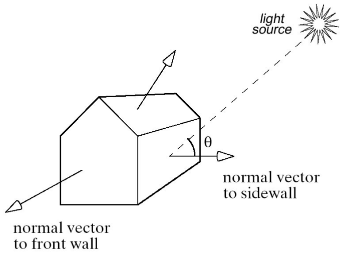
- Meshes can model both solid shapes and thin skins.
  - The object is **solid** if the polygonal faces fit together to enclose space.
  - In other cases, the faces fit together without enclosing space, and so they represent an infinitesimally thin surface.
- In both cases we call the collection of polygons a **polygonal mesh** (or simply a **mesh**).

Polygonal Meshes (5)

- 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.

Polygonal Meshes (6)

- The normal direction to a face determines its brightness.

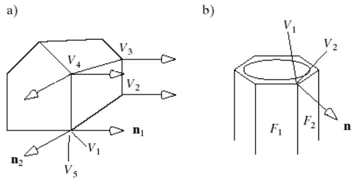


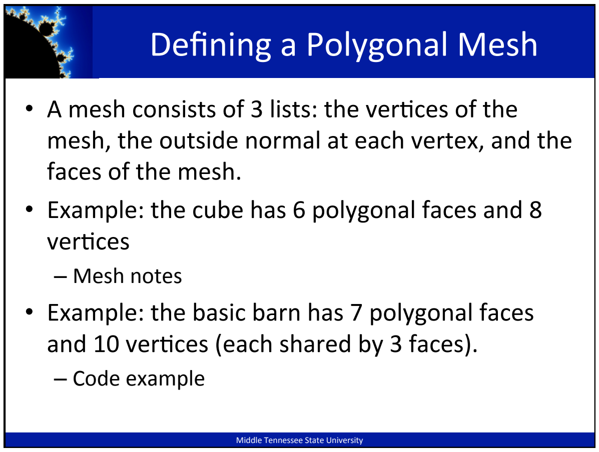
Polygonal Meshes (7)

- 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)

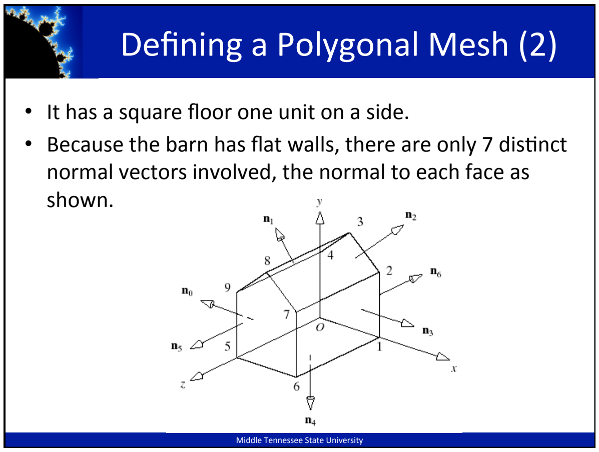
- Each vertex V_1, V_2, V_3, and V_4 defining the side wall of the barn has the *same* normal $\mathbf{n}_1$, the normal vector to the side wall.
- But vertices of the front wall, such as V_5, will use normal $\mathbf{n}_2$. (Note that vertices V_1 and V_5 are located at the same point in space, but use different normals.)





Defining a Polygonal Mesh

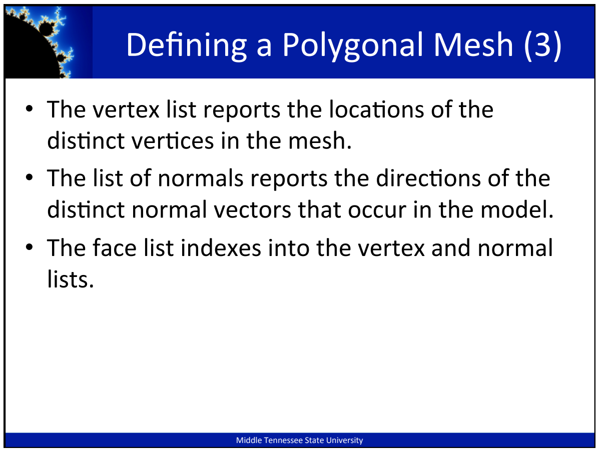
- A mesh consists of 3 lists: the vertices of the mesh, the outside normal at each vertex, and the faces of the mesh.
- Example: the cube has 6 polygonal faces and 8 vertices
  - Mesh notes
- Example: the basic barn has 7 polygonal faces and 10 vertices (each shared by 3 faces).
  - Code example



Defining a Polygonal Mesh (2)

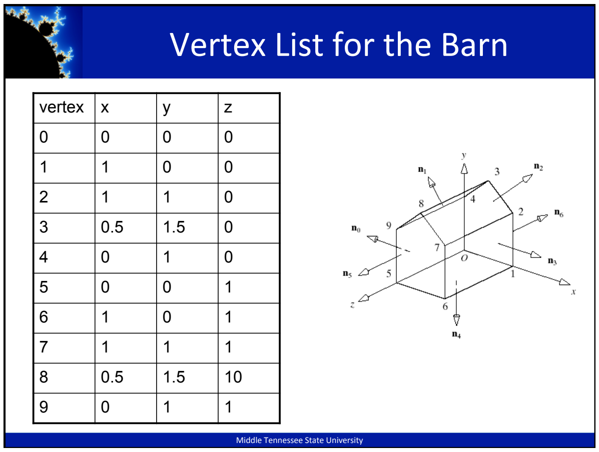
- It has a square floor one unit on a side.
- Because the barn has flat walls, there are only 7 distinct normal vectors involved, the normal to each face as shown.

y
n_1
3
n_2
8
4
2
n_6
n_0
9
7
O
n_3
n_5
5
1
x
z
6
n_4



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.



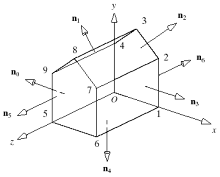
Vertex List for the Barn

vertex	x	y	z
0	0	0	0
1	1	0	0
2	1	1	0
3	0.5	1.5	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0.5	1.5	10
9	0	1	1

y
n_1
3
n_2
8
4
2
n_6
n_0
9
7
O
n_3
n_5
5
1
x
z
6
n_4

Normal List for the Barn

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



normal	n_x	n_y	n_z
0	-1	0	0
1	-0.707	0.707	0
2	0.707	0.707	0
3	1	0	0
4	0	-1	0
5	0	0	1
6	0	0	-1

Face List for the Barn

Face	Vertices	Normal
0 (left)	0, 5, 9, 4	0,0,0,0
1 (roof left)	3, 4, 9, 8	1,1,1,1
2 (roof right)	2, 3, 8, 7	2, 2, 2,2
3 (right)	1, 2, 7, 6	3, 3, 3, 3
4 (bottom)	0, 1, 6, 5	4, 4, 4, 4
5 (front)	5, 6, 7, 8, 9	5, 5, 5, 5, 5
6 (back)	0, 4, 3, 2, 1	6, 6, 6, 6, 6

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.***
  - Traverse the vertex so that the interior of the face lies to the left
  - Using the convention allows algorithms to distinguish with ease the front from the back of a face.

3D File Formats

- There is no standard file format.
- Some formats have been found to be efficient and easy to use: for example, the .qs file format developed by the Stanford University Computer Graphics Laboratory. This particular mesh model has 2,748,318 points (about 5,500,000 triangles) and is based on 566,098 vertices.
  - http://lodbook.com/models/

3D File Formats (2)

- OpenGL has the capability to load a variety of 3D model formats such as (but not limited to) 3DS, VRML, PLY, MS3D, ASE and OBJ.

```
ply
format ascii 1.0
comment generated by ply_writer
element vertex 543652
property float x
property float y
property float z
element face 1087716
property list uchar int vertex_indices
end_header
-0.00709987 0.064825 -0.0472725
-0.0046435 0.064825 -0.0472796
-0.00423929 0.064825 -0.0472725
-0.0078746 0.065075 -0.0472725
-0.0076435 0.0650297 -0.0472725
…..
3 543641 543645 543651
3 543641 543651 543650
3 543651 543645 543602
3 543645 543595 543601
3 543648 543644 543613
3 543649 543650 543625
```

Calculating Normals

- Take any three non-collinear points on the face, V_1, V_2, and V_3, and compute the normal as their cross product **m** = (V_1 - V_2) × (V_3 - V_2) and normalize it to unit length.
  - If the two vectors V_1 - V_2 and V_3 - V_2 are nearly parallel, the cross product will be very small and numerical inaccuracies may result.
  - The polygon may not be perfectly planar. Thus the surface represented by the vertices cannot be truly flat. We need to form some average value for the normal to the polygon, one that takes into consideration all of the vertices.
    - Newell's Method for Normal

Newell's Method for Normals

- Given N vertices, define next(i) = n_i = (i+1) mod N.
- Traverse the vertices for the face in counter-clockwise order from the outside.
- The normal computed points to the outside (front) of the face:

$$n_x = \sum_{i=0}^{N-1} (y_i - y_{ni})(z_i + z_{ni})$$

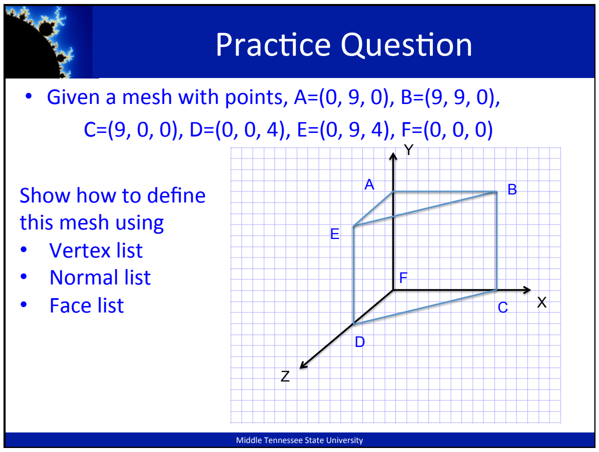
$$n_y = \sum_{i=0}^{N-1} (z_i - z_{ni})(x_i + x_{ni})$$

$$n_z = \sum_{i=0}^{N-1} (x_i - x_{ni})(y_i + y_{ni})$$

Practice Question

- For the following polygons,
  - Are they planar?
  - What is its normal computed based on cross product method?
  - What is the normal computed using the Newell's method?

1. P_0(6, 1, 4), P_1=(7, 0, 9), and P_2=(1, 1, 2)
2. P_0(0, 0, 3), P_1=(-3, 1, 3), P_2=(0, 0, 5), and P_3=(3, 0, 0)

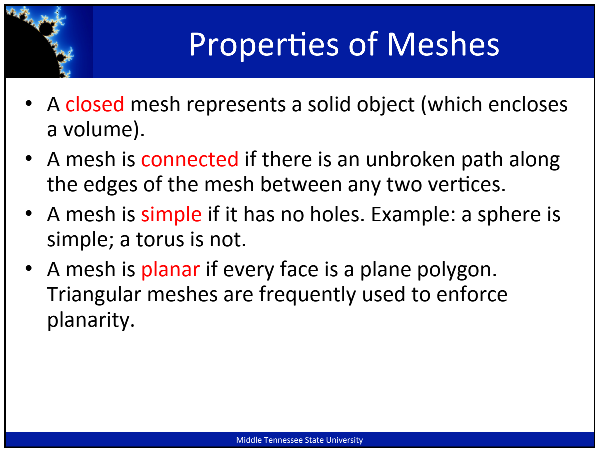


Practice Question

- Given a mesh with points, A=(0, 9, 0), B=(9, 9, 0), C=(9, 0, 0), D=(0, 0, 4), E=(0, 9, 4), F=(0, 0, 0)

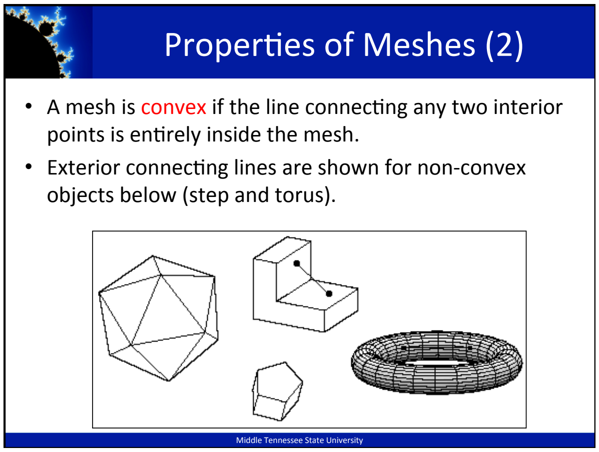
Show how to define this mesh using

- Vertex list
- Normal list
- Face list



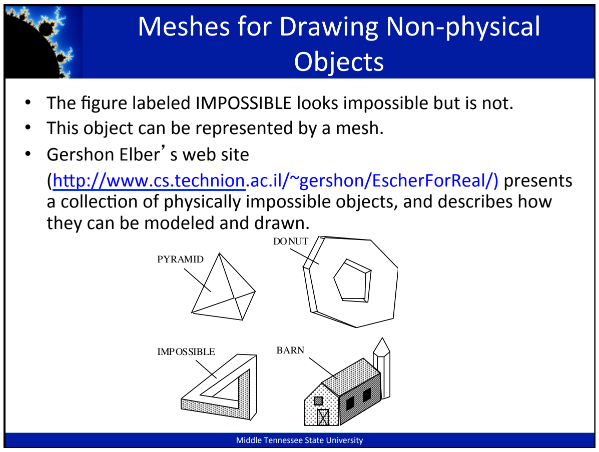
Properties of Meshes

- 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)

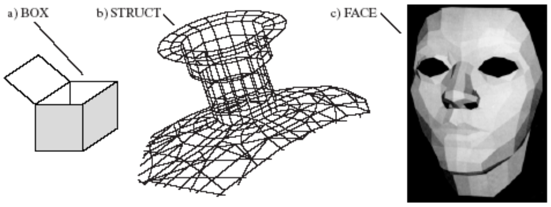
- A mesh is convex if the line connecting any two interior points is entirely inside the mesh.
- Exterior connecting lines are shown for non-convex objects below (step and torus).



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



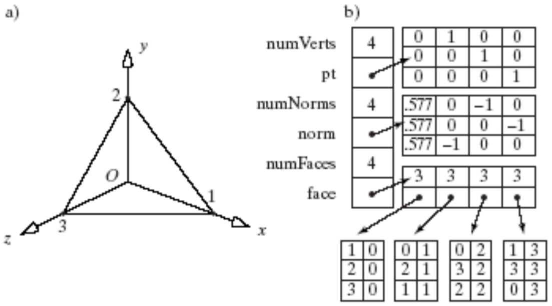
Working with Meshes in a Program

- We want an efficient Mesh class that makes it easy to create (in terms of reading mesh data into special data structure) and draw the object.
- Since mesh data is frequently stored in a file, we also need simple ways to read and write mesh files.
  - Refer to Mesh notes on cube

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 *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)



numVerts	4			
pt	0	1	0	0
	0	0	1	0
	0	0	0	1
numNorms	4			
norm	.577	0	–1	0
	.577	0	0	–1
	.577	–1	0	0
numFaces	4			
face	3	3	3	3

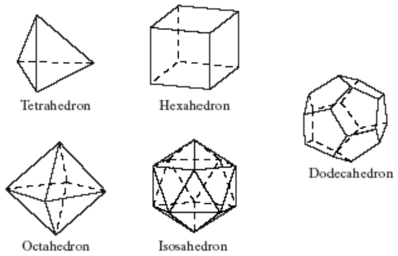
1	0	0	1	0	2	1	3
2	0	2	1	3	2	3	3
3	0	1	1	2	2	0	3

Drawing the Mesh Object

- Mesh::draw() 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.

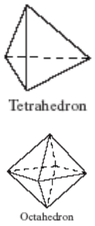
Platonic Solids

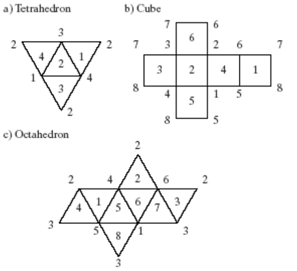
- All the faces are identical and each is a regular polygon.



Flattened Models

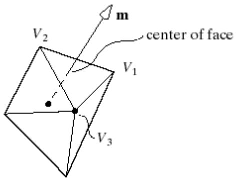
- To keep track of vertex and face numbering, we use a flat **model**, which is made by cutting along certain edges of each solid and unfolding it to lie flat.



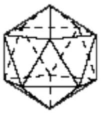


Normal Vectors for the Platonic Solids

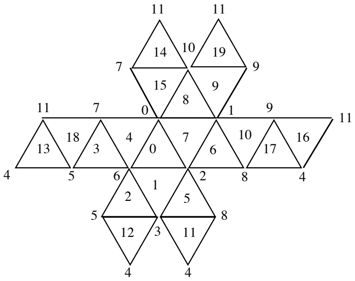
- Normals can be found using Newell's method.
- Also, because of the high degree of symmetry of a Platonic solid, **if the solid is centered at the origin**, the normal vector to each face is the vector from the origin to the *center* of the face (the average of the vertices of the face).



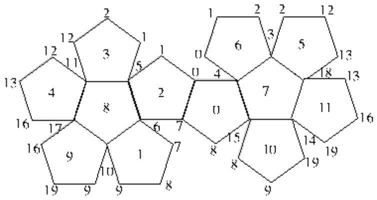
Flattened Model for Icosahedron

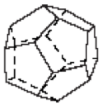


Isosahedron



Flattened Model for Dodecahedron

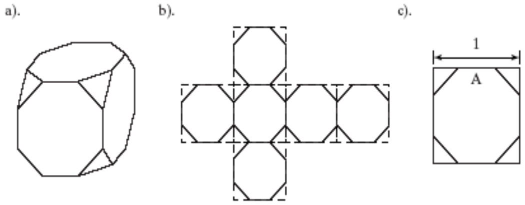




Dodecahedron

Archimedean Solids

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

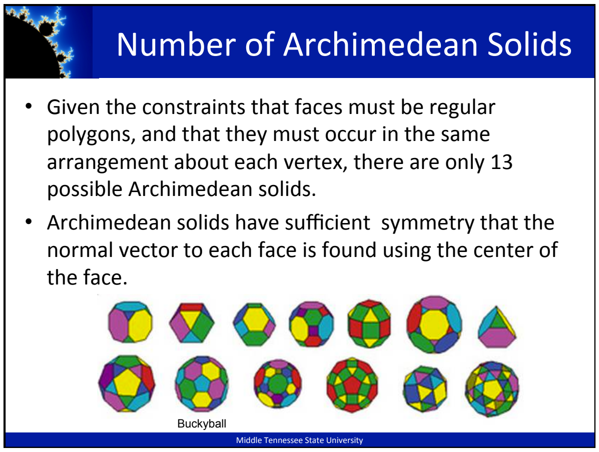


Truncated Cube

- Each edge of the cube is divided into three parts; the middle part of length 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

$$V = \frac{1+A}{2}C + \frac{1-A}{2}D \qquad W = \frac{1-A}{2}C + \frac{1+A}{2}D$$

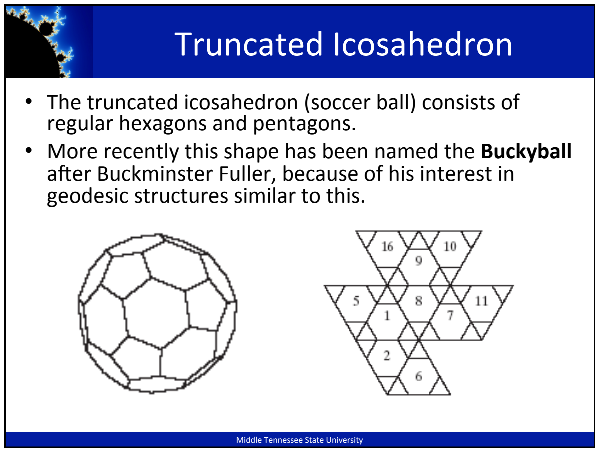
$$A = \sqrt{2} - 1$$



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.

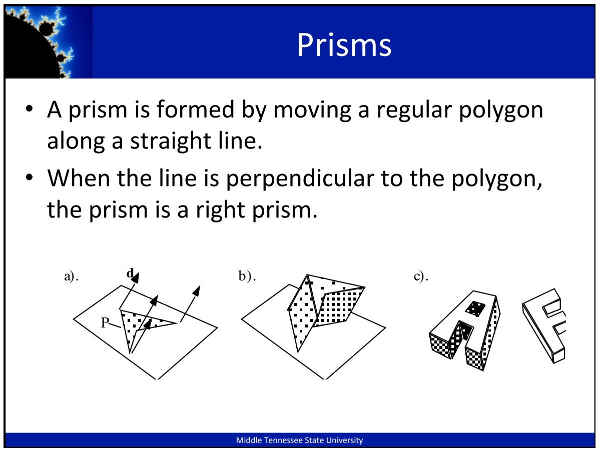
Middle Tennessee State University



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.

Middle Tennessee State University



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.

Middle Tennessee State University