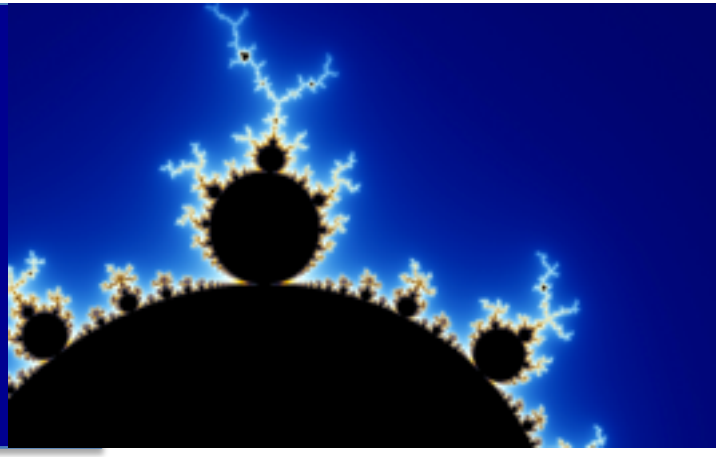
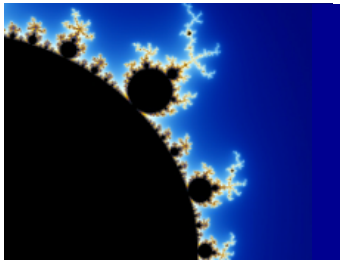
Computer Graphics

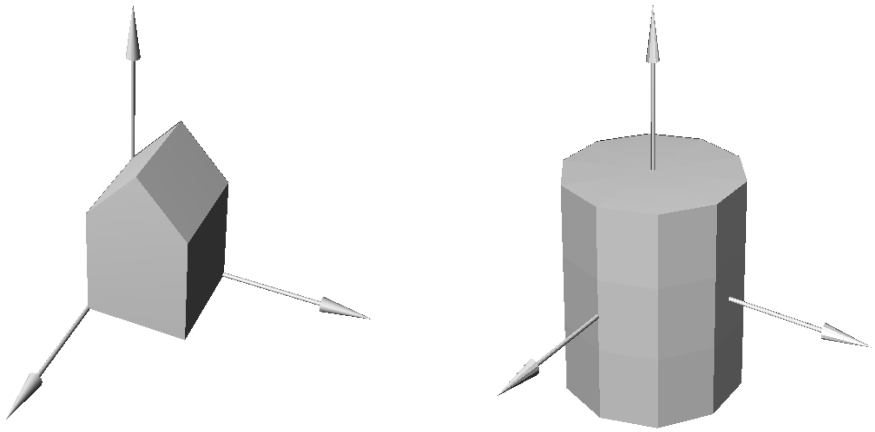


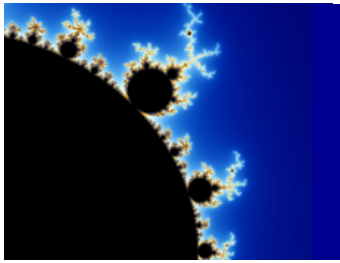
Modeling Shapes with Polygonal Meshes



Polygonal Meshes

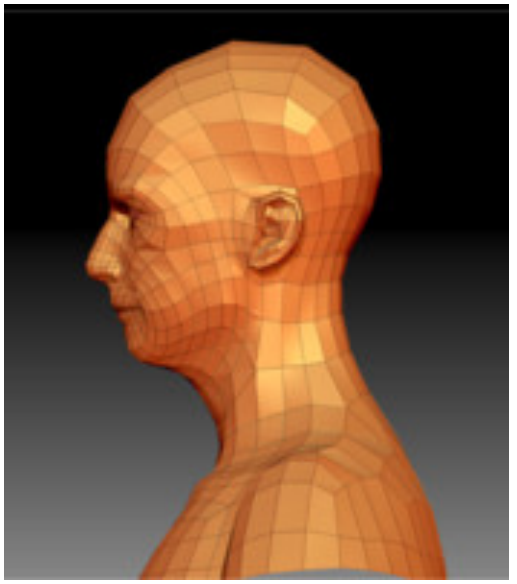
- A polygonal mesh is a collection of polygons (faces) that approximate the surface of a 3D object.
  - Examples: surfaces of sphere, cone, cylinder made of polygons; barn.

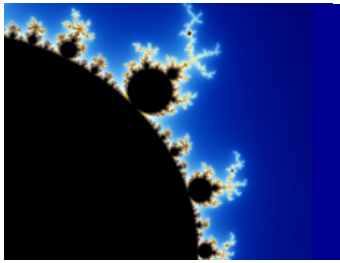




Polygonal Meshes

- 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**).



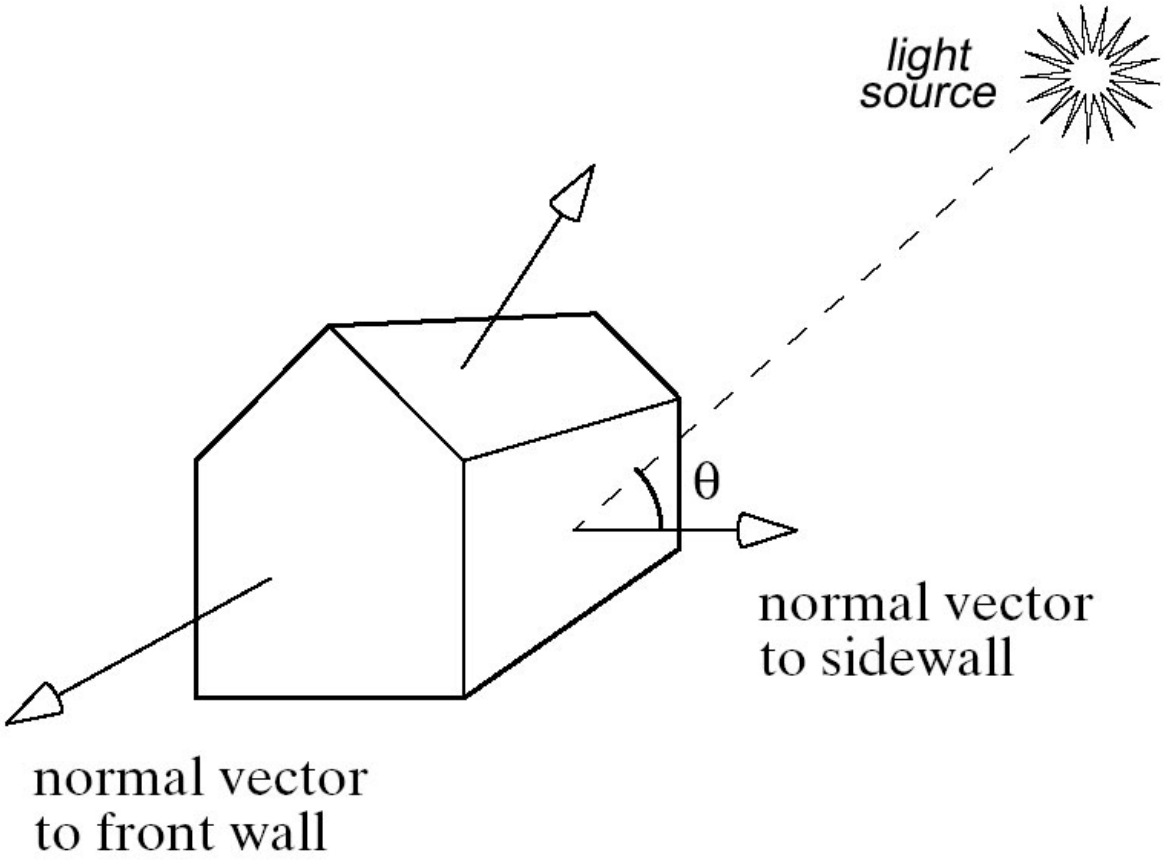


Polygonal Meshes

- 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.

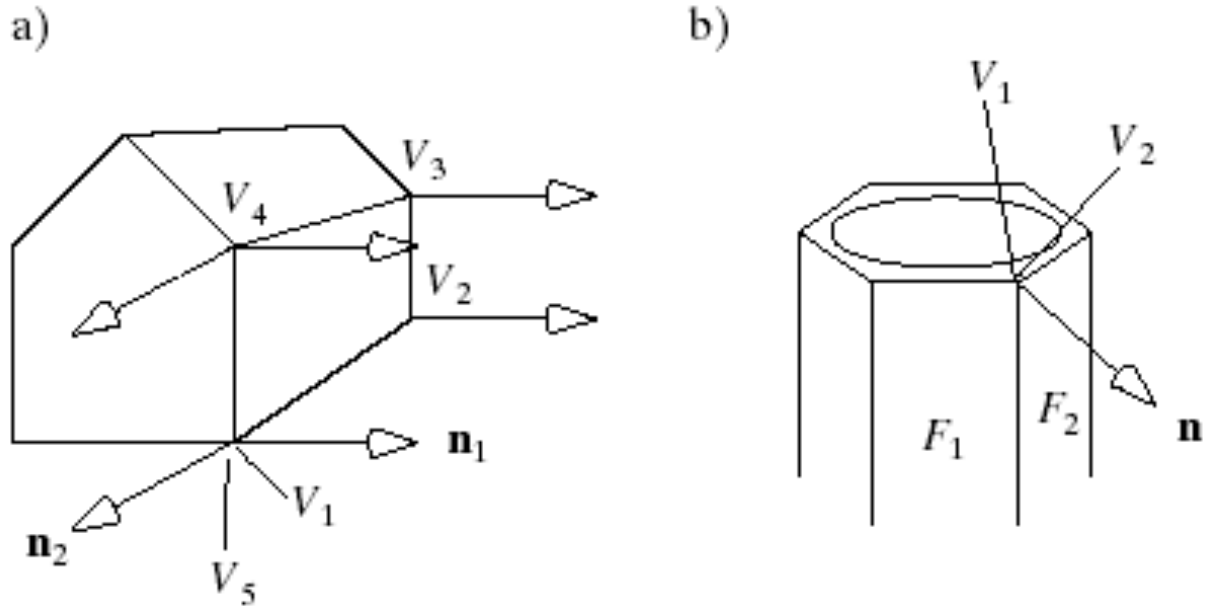
Polygonal Meshes (6)

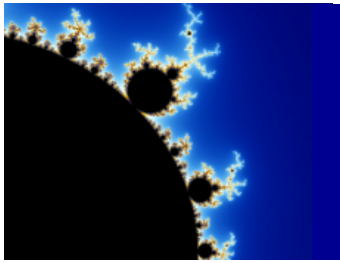
- The normal direction to a face determines its brightness.



Polygonal Meshes

- 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.)



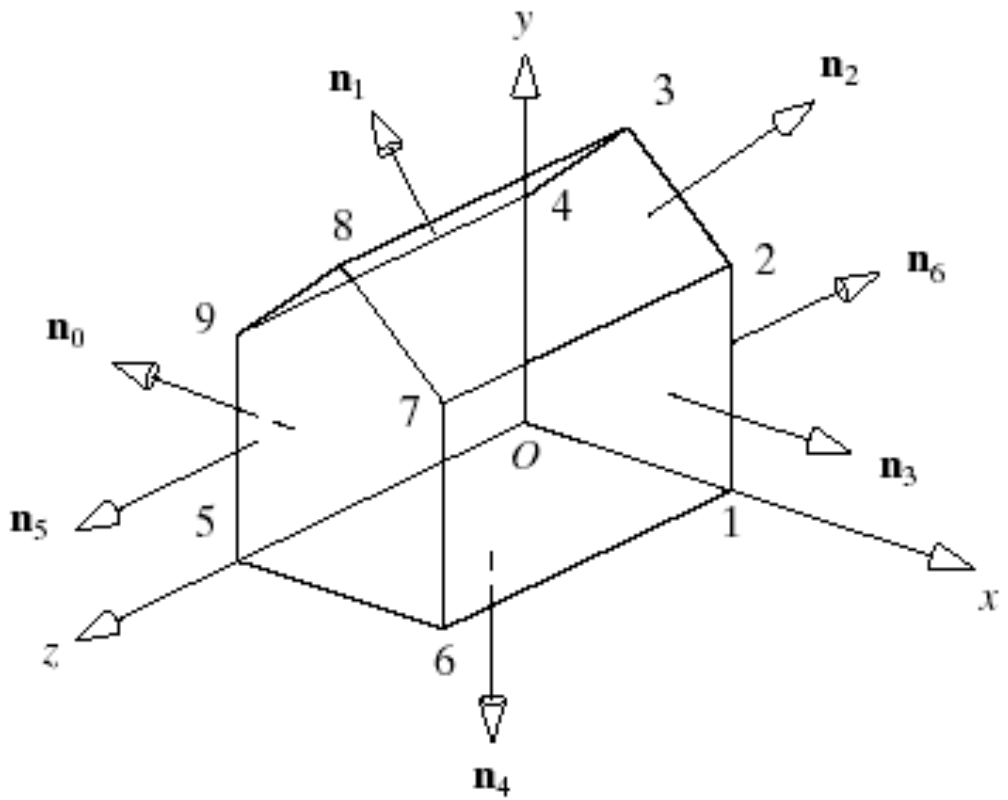


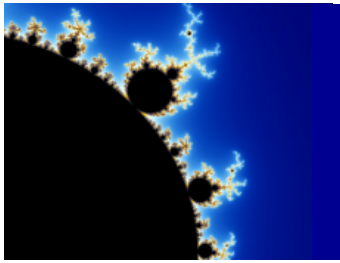
Defining a Polygonal Mesh

- 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
- 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.



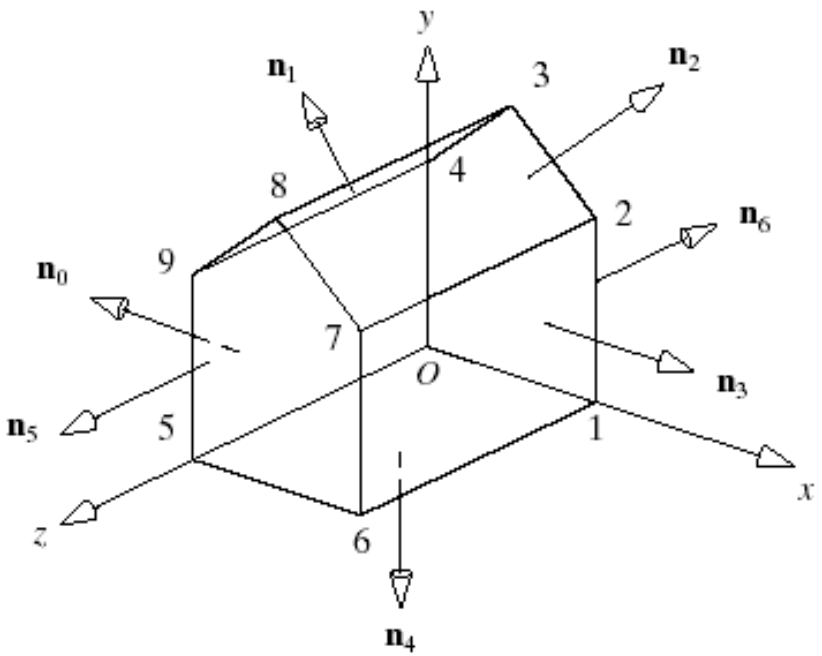


Defining a Polygonal Mesh

- 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.

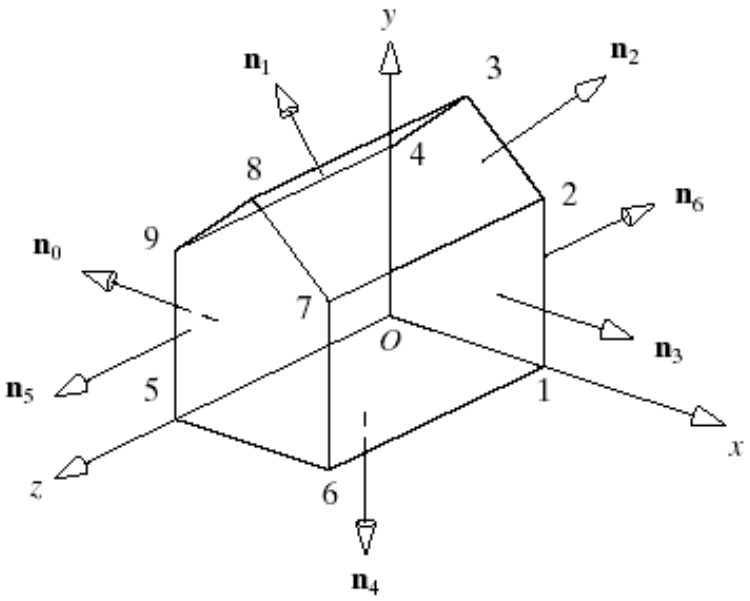
Vertex List for the Barn

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

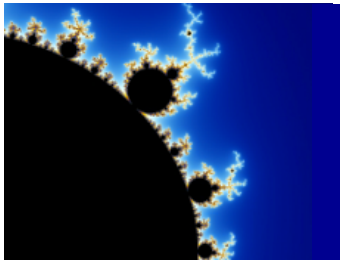


Normal List for the Barn

- 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

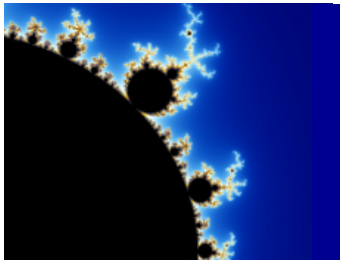


Face List for the Barn

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

3D File Formats

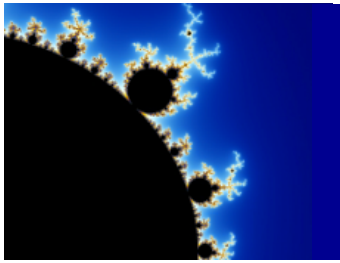
- There is no standard file format.
- Some formats have been found to be efficient and easy to use: for example, the .qs file format developed by the Stanford University Computer Graphics Laboratory. This particular mesh model has 2,748,318 points (about 5,500,000 triangles) and is based on 566,098 vertices.
  - http://graphics.stanford.edu/data/3Dscanrep/



3D File Formats (2)

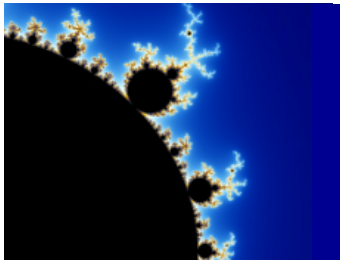
- There are a variety of 3D model formats such as (but not limited to) 3DS, VRML, PLY, MS3D, ASE and OBJ.

```
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 $\mathbf{m} = (V_1 - V_2) \times (V_3 - V_2)$ 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



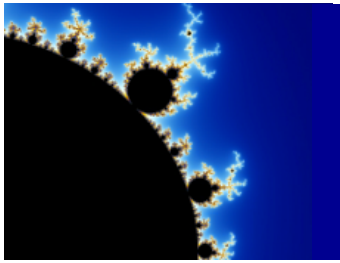
Newell's Method for Normals

- 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?

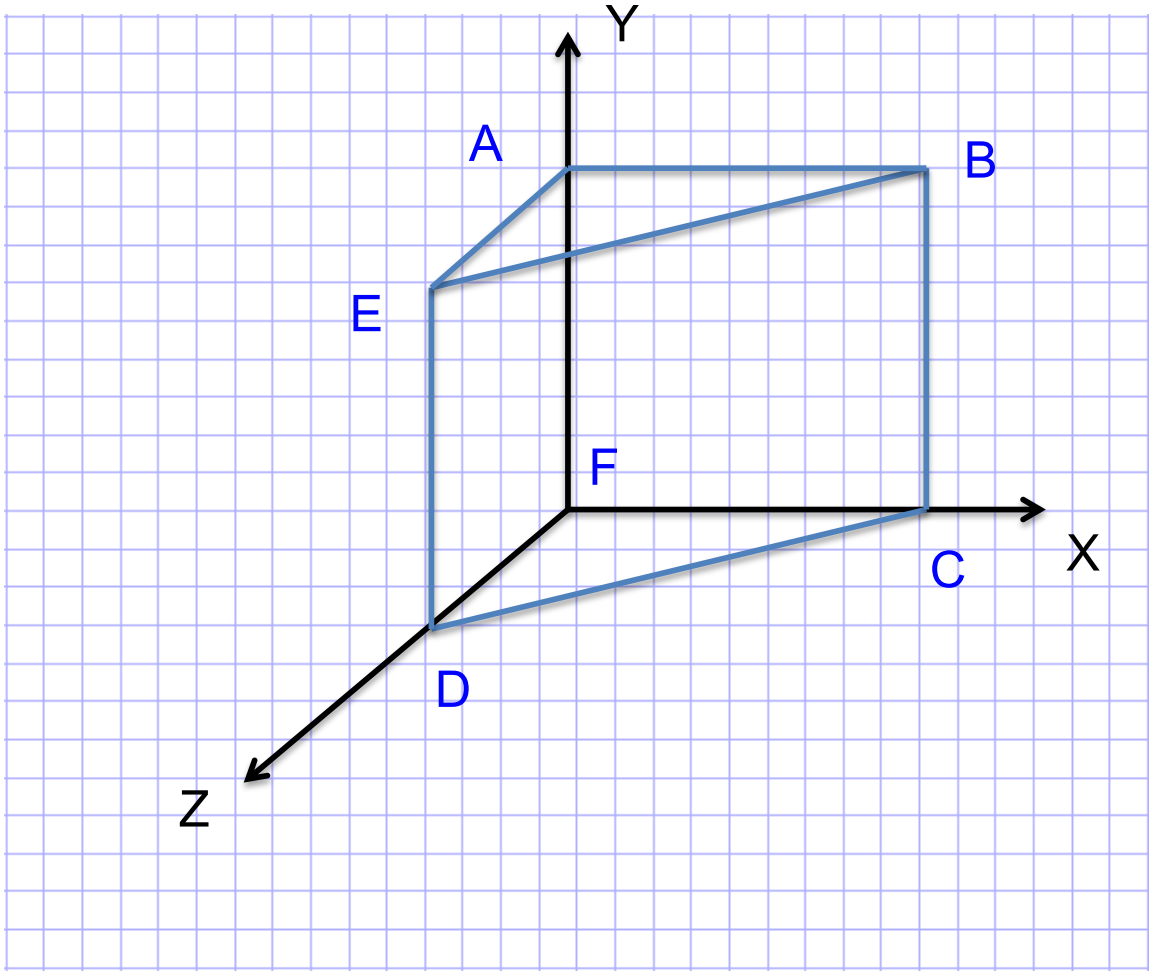
1. $P_0(6, 1, 4)$, $P_1=(7, 0, 9)$, and $P_2=(1, 1, 2)$
2. $P_0(0, 0, 3)$, $P_1=(-3, 1, 3)$, $P_2=(0, 0, 5)$, and $P_3=(3, 0, 0)$

Practice Question

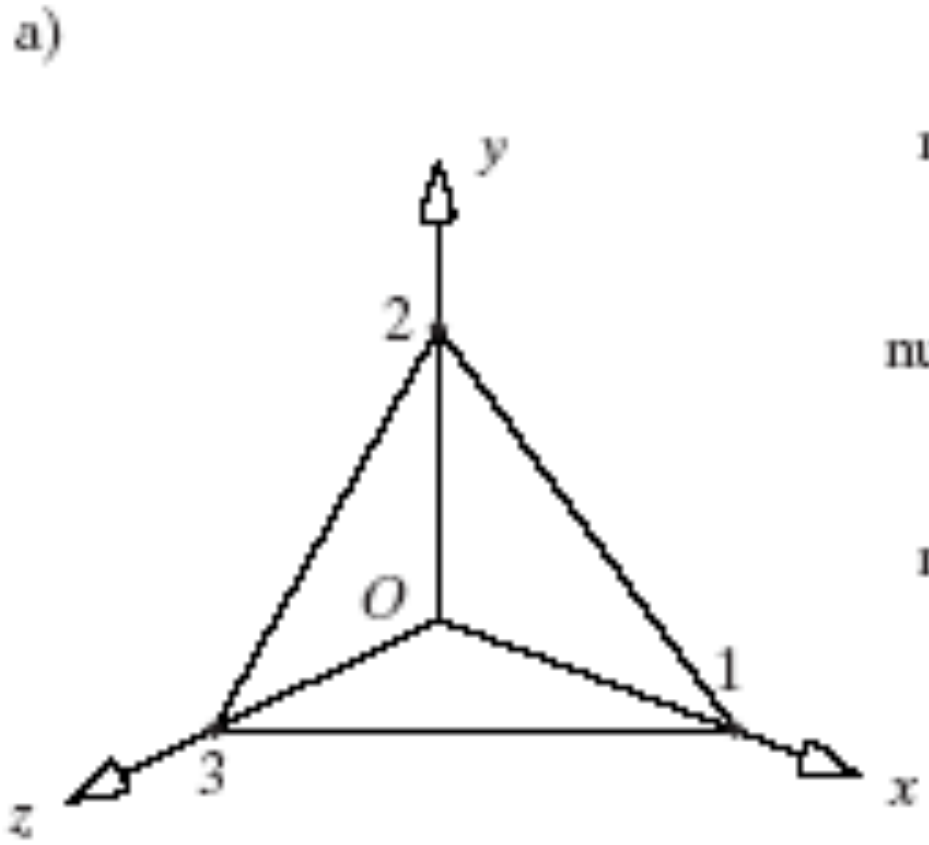
- Given a mesh with points, A=(0, 9, 0), B=(9, 9, 0), C=(9, 0, 0), D=(0, 0, 4), E=(0, 9, 4), F=(0, 0, 0)

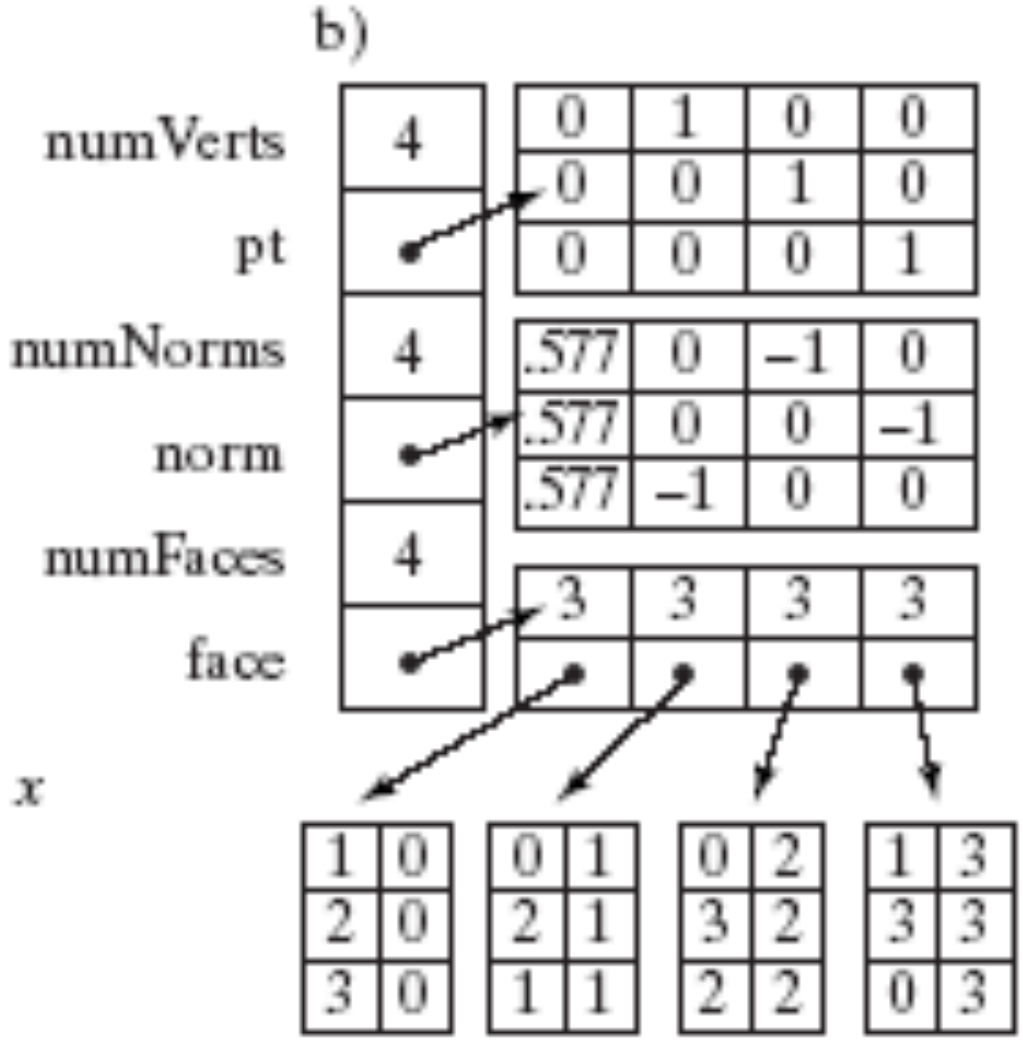
Show how to define this mesh using

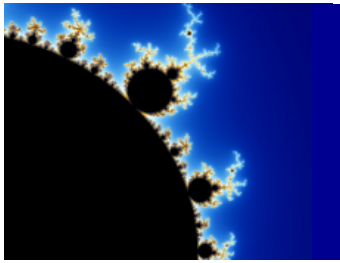
- Vertex list
- Normal list
- Face list



Example (tetrahedron & representation)







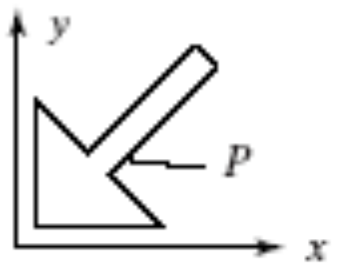
Extruded Shapes

- A large class of shapes can be generated by **extruding** or **sweeping** a 2D shape through space.
- In addition, surfaces of revolution can also be approximated by extrusion of a polygon once we slightly broaden the definition of extrusion.

Extruded Shapes

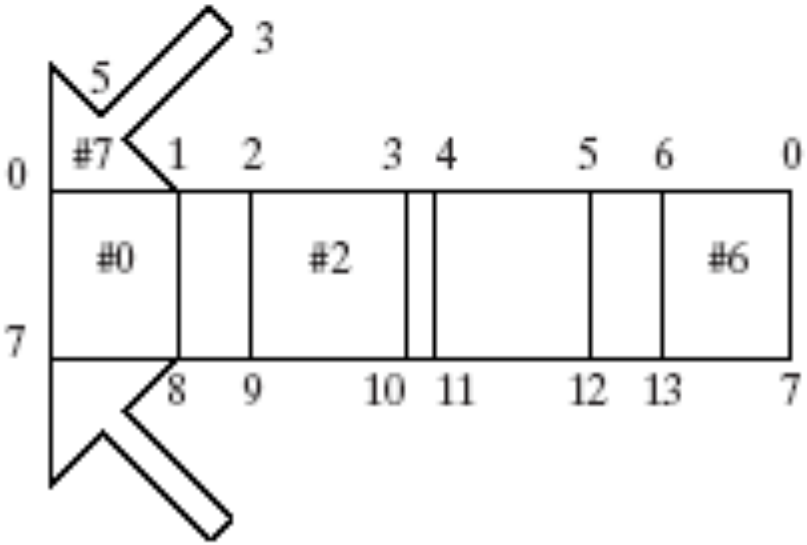
- Prism: formed by sweeping the arrow along a straight line.
- Flattened version.

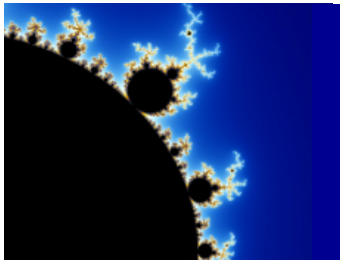
a) polygon base:



b) *P* swept in *z*-direction

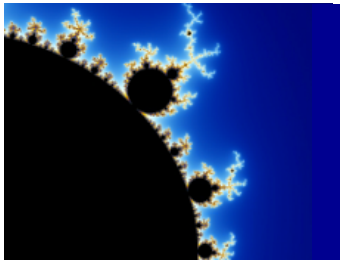
c) Model for ARROW prism





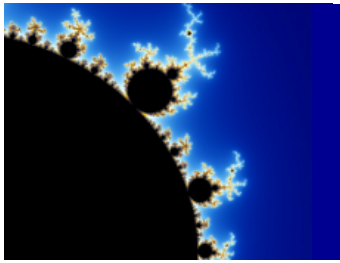
Extruded Shapes (2)

- Base has vertices (x_i, y_i, 0) and top has vertices (x_i, y_i, H).
- Each vertex (x_i, y_i, H) on the top is connected to corresponding vertex (x_i, y_i, 0) on the base.
- If the polygon has **n** sides, then there are **n** vertical sides of the prism plus a top side (cap) and a bottom side (base), or **n+2** faces altogether.
- The normals for the prism are the face normals. These may be obtained using the Newell method, and the normal list for the prism constructed.



Vertex List for the Prism

- Suppose the prism's base is a polygon with N vertices (x_i, y_i). We number the vertices of the base 0, . . . , N-1 and those of the cap N, . . ., 2N - 1, so that an edge joins vertices i and $i + N$.
- The vertex list is then easily constructed to contain the points $(x_i, y_i, 0)$ and (x_i, y_i, H), for i = 0, 1, ..., N-1.



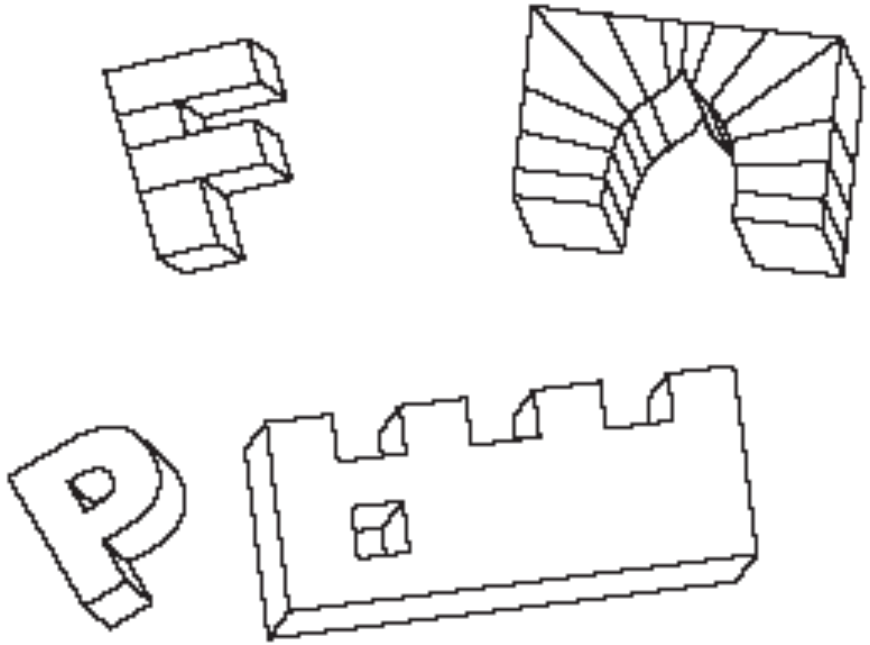
Face List for the Prism

- We first make the side faces and then add the cap and base.
- For the *j*-th wall (*j* = 0,...,*N*-1) we create a face with the four vertices having indices *j*, *j* + *N*, *next*(*j*)+ *N*, and *next*(*j*) where *next*(*j*) is (*j*+1) % N.

```
if (j < n-1)
    next = ++j;
else
    next = 0;
```

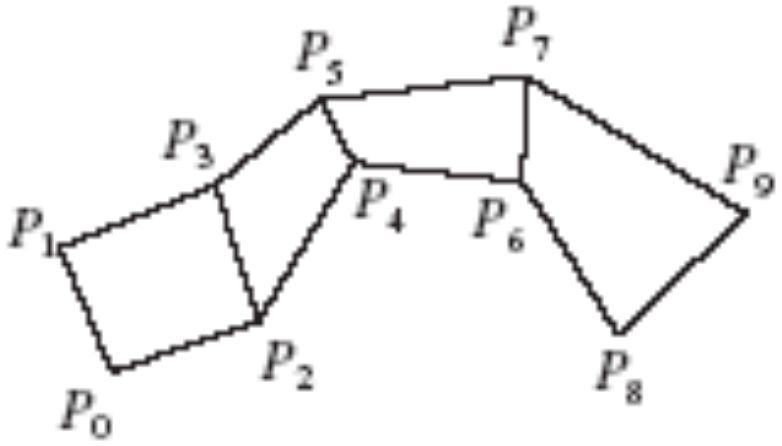
Arrays of Extruded Prisms

- WebGL/OpenGL can reliably draw only convex polygons (A **convex polygon** is defined as a **polygon** with all its interior angles less than 180°)
- For non-convex prisms, stack the parts.

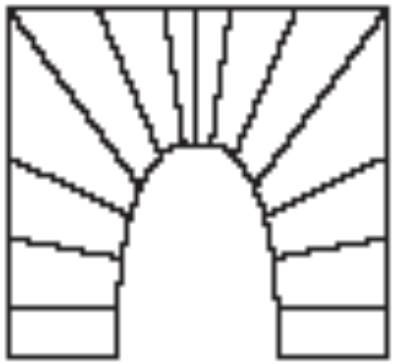


Special Case: Extruded Quadstrips

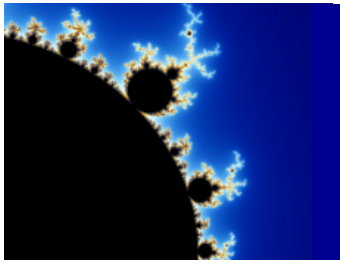
a)



b)



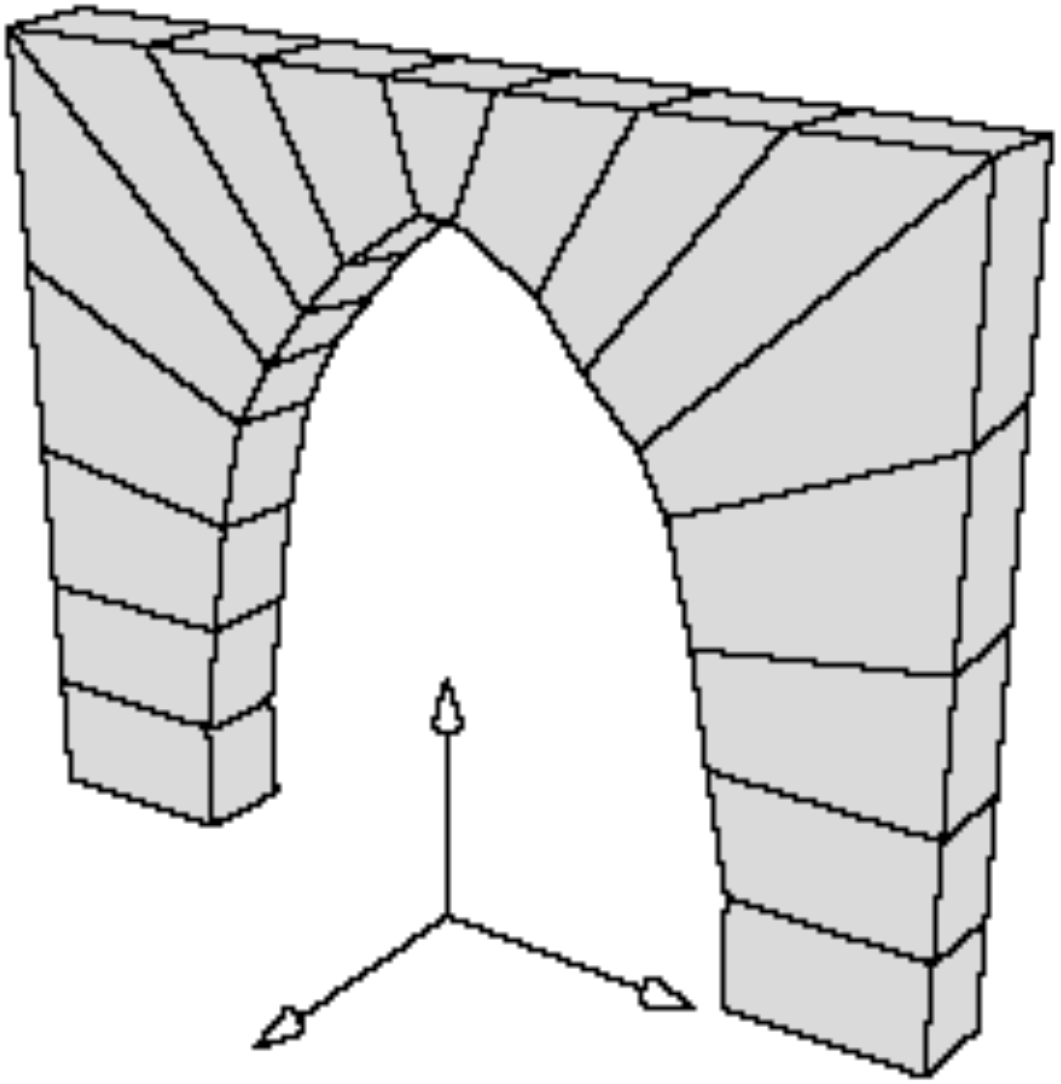
c)

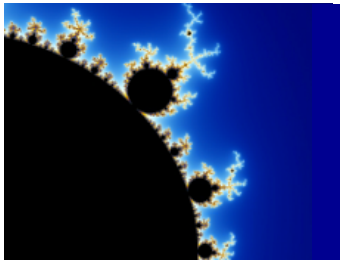


Quadstrip Data Structure

- quad-strip = $\{p_0, p_1, p_2, \ldots, p_{M-1}\}$
- The vertices are understood to be taken in pairs, with the *odd* ones forming one edge of the quad-strip, and the *even* ones forming the other edge.
- Not every polygon can be represented as a quad-strip.

Example: Arch



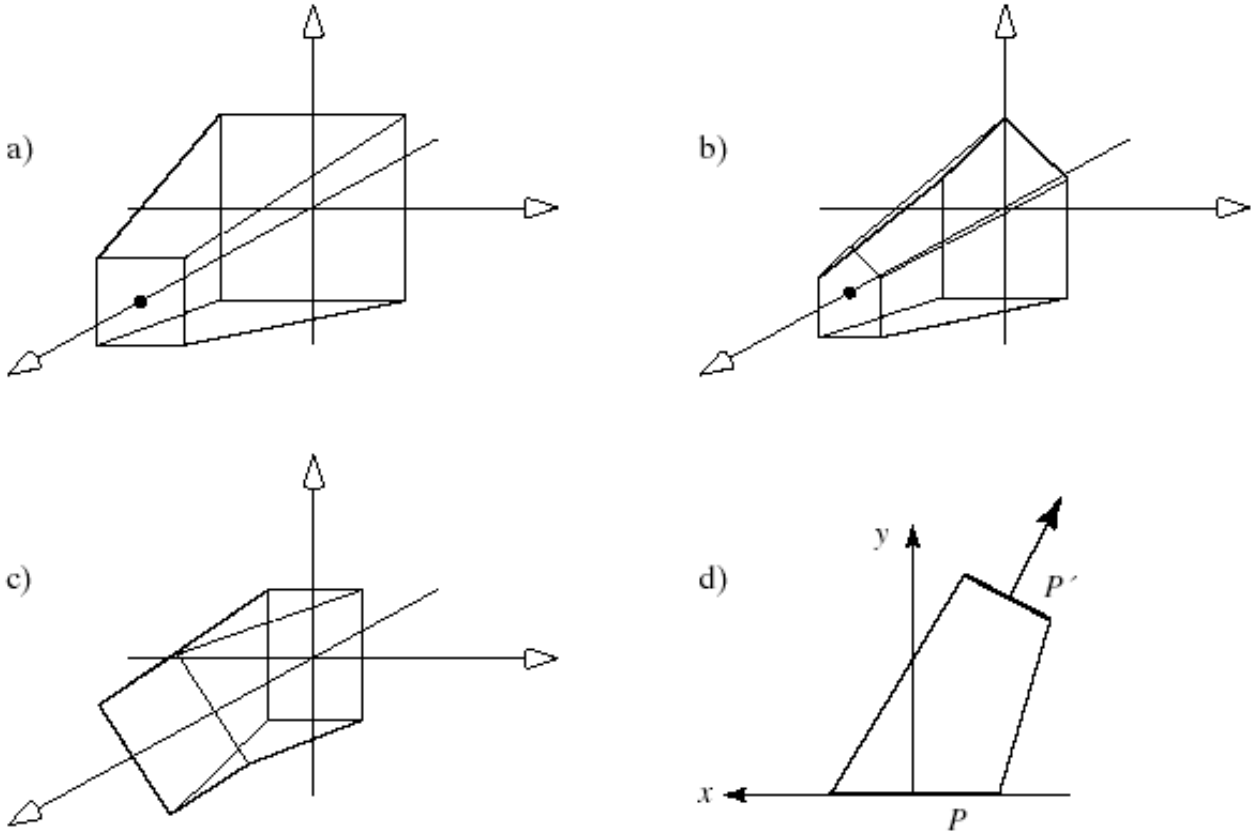


Special Case: Twisted Extrusions

- Base is n-gon, top is scaled, translated, and possibly rotated version of base.
- Specifically, if the base polygon is P, with vertices $\{p_0, p_1, \ldots, p_{N-1}\}$, the cap polygon has vertices $P' = \{ Mp_0, Mp_1, \ldots, Mp_{N-1}\}$ where M is some 4 by 4 affine transformation matrix.

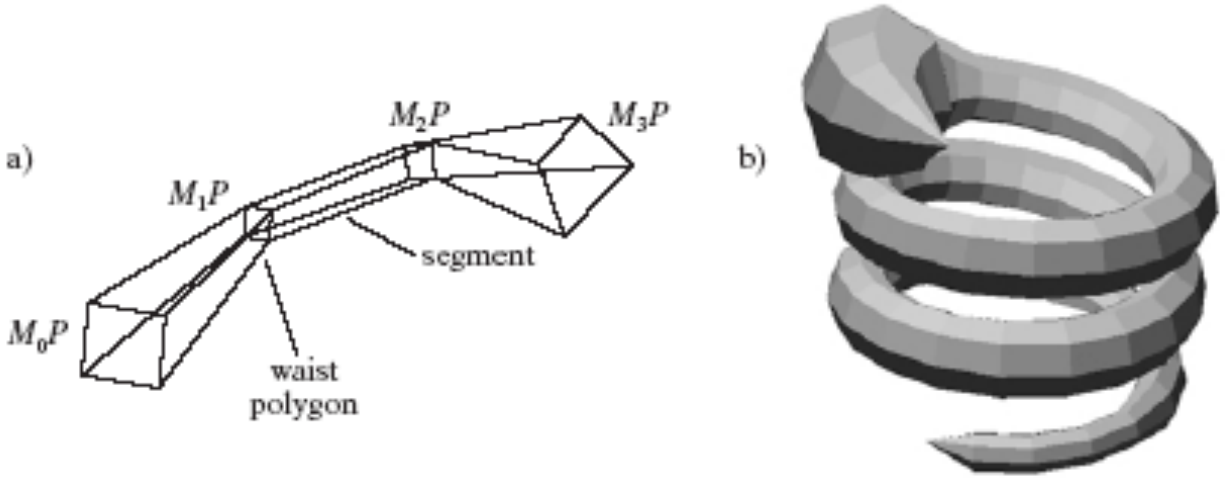
Examples

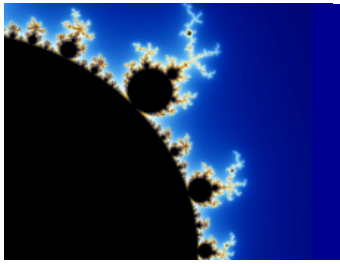
- A), B): cap is smaller version of base.
- C): cap is rotated through θ about *z*-axis before translation.
- D): cap *P*’ is scaled, and rotated arbitrarily before translation.



Segmented Extrusions

- Below: a square P extruded three times, in different directions with different tapers and twists. The first segment has end polygons M_0P and M_1P, where the initial matrix M_0 positions and orients the starting end of the tube. The second segment has end polygons M_1P and M_2P, etc.



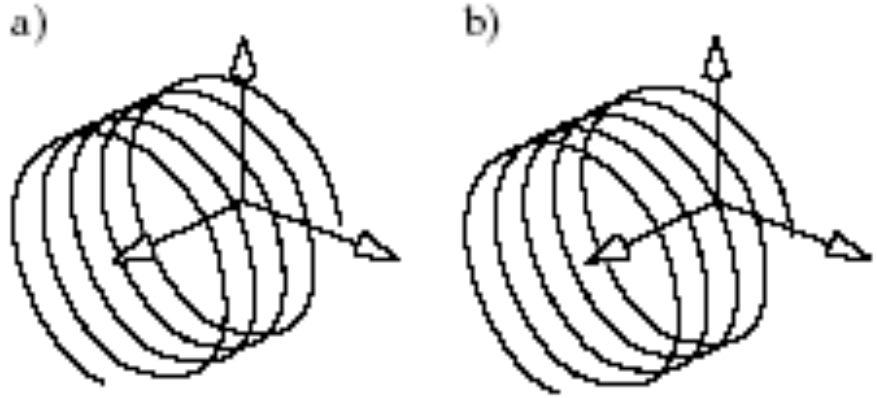


Special Case: Segmented Extrusions

- We shall call the various transformed squares the "**waists**" of the tube.
- In this example the vertex list of the mesh contains the 16 vertices
  - M_0p_0, $M_0\,p_1$, $M_0\,p_2$, $M_0\,p_3$,
  - M_1p_0, M_1p_1, M_1p_2, M_1p_3,
  - …,
  - M_3p0, M_3p_1, M_3p_2, M_3p_3.
- The "snake" used the matrices M_i to grow and shrink the tube to represent the body and head of a snake.

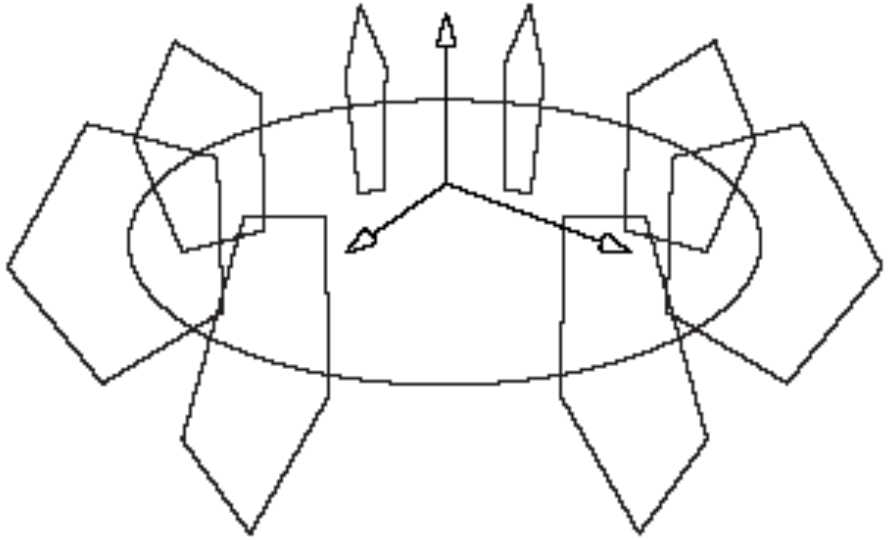
Methods for Twisted Extrusions

- Multiple extrusions used, each with its own transformation. The extrusions are joined together end to end.
- The extrusion tube is wrapped around a space curve C, the spine of the extrusion (e.g., helix $C(t) = (cos(t), sin(t), bt)$).



Discretely Swept Surfaces of Revolution

- Example: rotating a polyline around an axis to produce a 3D figure.

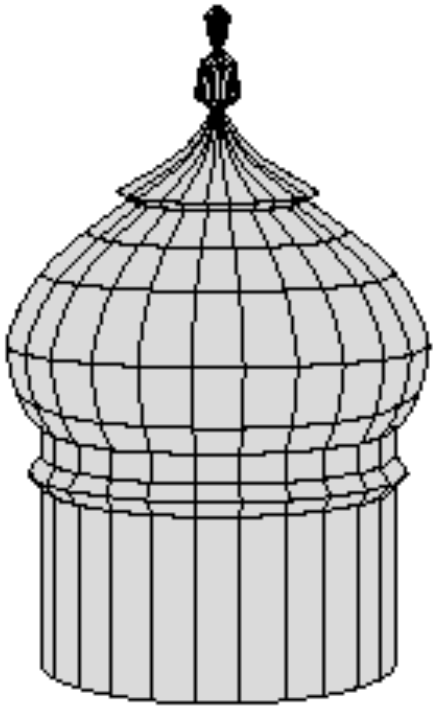


Example

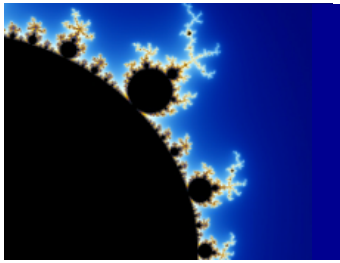
- A model of the dome of the Taj Mahal in Agra, India.

a)

b)

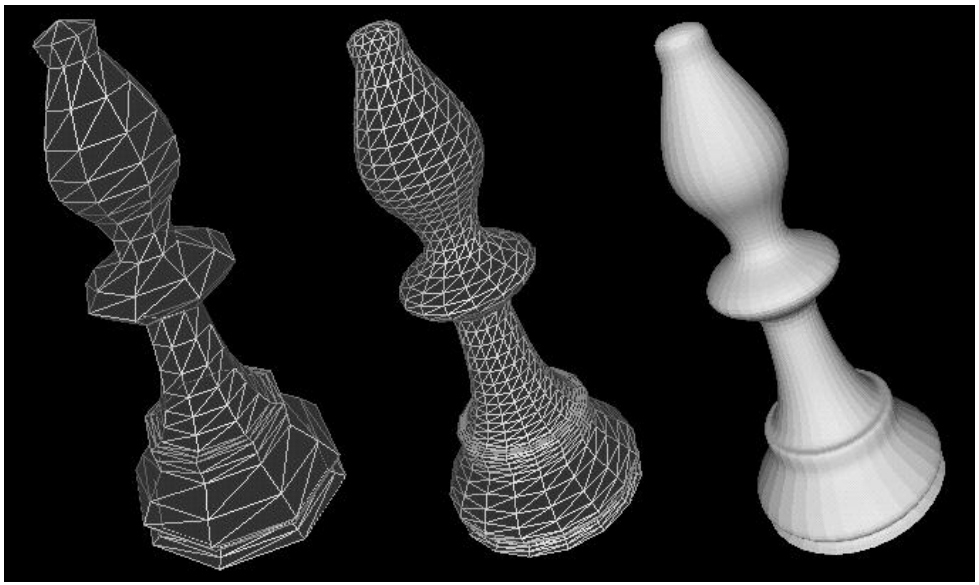


c)



Discretely Swept Surfaces of Revolution

- This is equivalent to **circularly sweeping** a shape about an axis.
- The resulting shape is often called a **surface of revolution**. Below: 3 versions of a pawn based on a mesh that is swept in discrete steps.

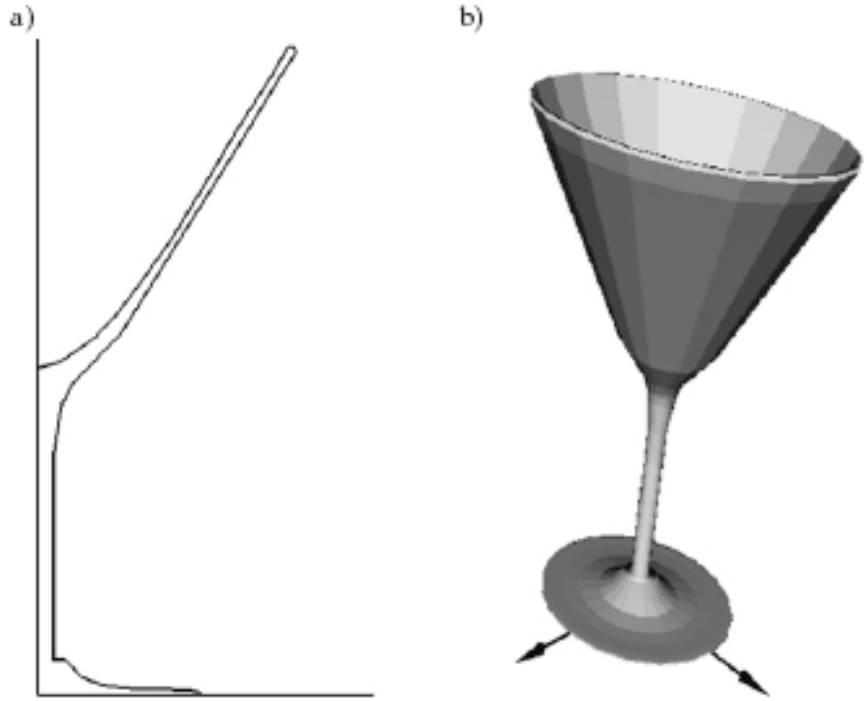


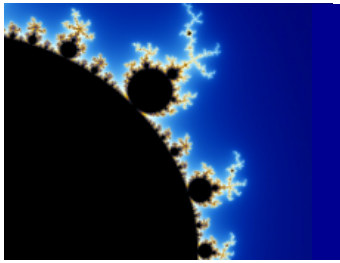
Discretely Swept Surfaces of Revolution

- Glass: polyline with P_j = (x_j, y_j, 0).
- To rotate the polyline to *K* equal-spaced angles about the *y*-axis:

$\theta_i = 2\pi * i/K$, i = 0, 1, 2, …, K, and

$$\tilde{M} = \begin{pmatrix} \cos(\vartheta_i) & 0 & \sin(\vartheta_i) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\vartheta_i) & 0 & \cos(\vartheta_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$





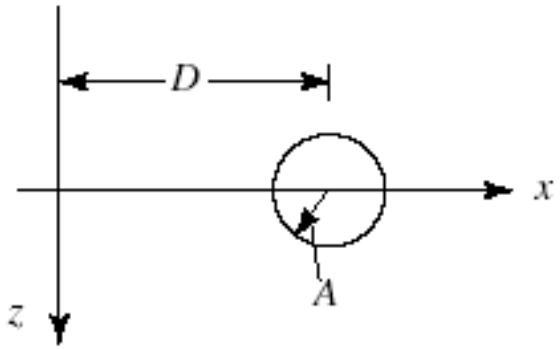
Surfaces of Revolution (2)

- The different positions of the curve *C* around the axis are called **meridians**.
- Sweeping C completely around generates a full circle, so contours of constant *v* are circles, called **parallels**.

Example

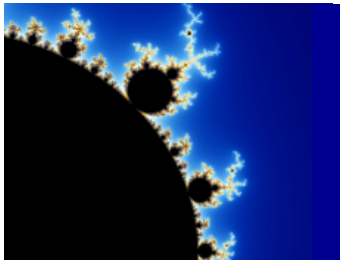
- The **torus** is generated by sweeping a circle displaced by a distance D along the x-axis about the *z*-axis.
- The circle has radius *A,* so its profile is C(v) = (*D* + *A cos*(*v*), *A sin*(*v*)).
- The torus has representation *P*(*u*, *v*) = ((*D* + *A cos*(*v*)) *cos*(*u*), (*D* + *A cos*(*v*)) *sin*(*u*), *A sin*(*v*))

a)



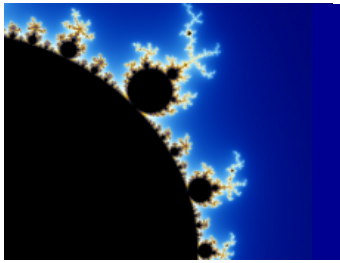
b)

V: angle of the points on the circle having radius=A
U: angle, as it sweeps around Z



Surfaces of Revolution as Mesh

- How to build a mesh for a surface of revolution ?
- We choose a set of u and v values, $\{u_i\}$ and $\{v_j\}$, and compute a vertex at each from $P(u_i, v_j)$, and a normal direction from $\mathbf{n}(u_i, v_j)$. Polygonal faces are built by joining four adjacent vertices with straight lines.



Mesh Approximations to Smooth Objects

- The faces have vertices that are found by evaluating the surface's parametric representation at discrete points.
- A mesh is created by building a vertex list and face list in the usual way, except now the vertices are computed from formulas.
- The vertex normal vectors are computed by evaluating formulas for the **normal to the smooth surface** at discrete points.