Fundamentals of Computer Graphics

Lecture 4. 3D modeling using polygonal meshes

Yong-Jin Liu
liuyongjin@tsinghua.edu.cn

Outline

- 3D models
- Polygonal meshes
- OpenGL rendering pipeline
- OpenGL transformation

3D modeling

Polygonal meshes capture the shape of complex 3D objects in simple data structures

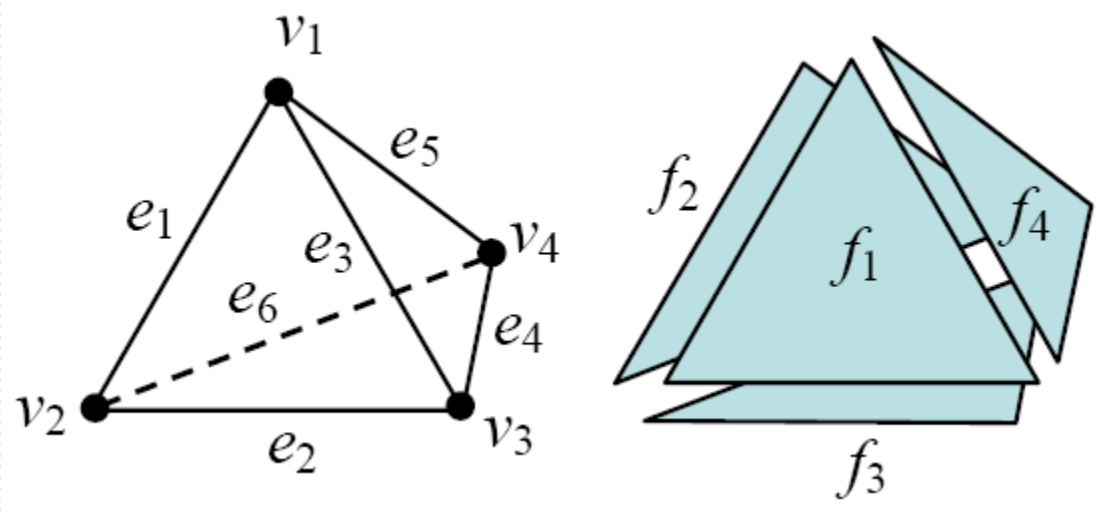
- Quad/tri meshes
- Approximation of solids with smoothly curved surface

Polygonal Meshes

A polygonal mesh is a collection of polygons (faces) that approximate the surface of a 3D object

- Polygons are easy to represent (by a sequence of vertices) and transform

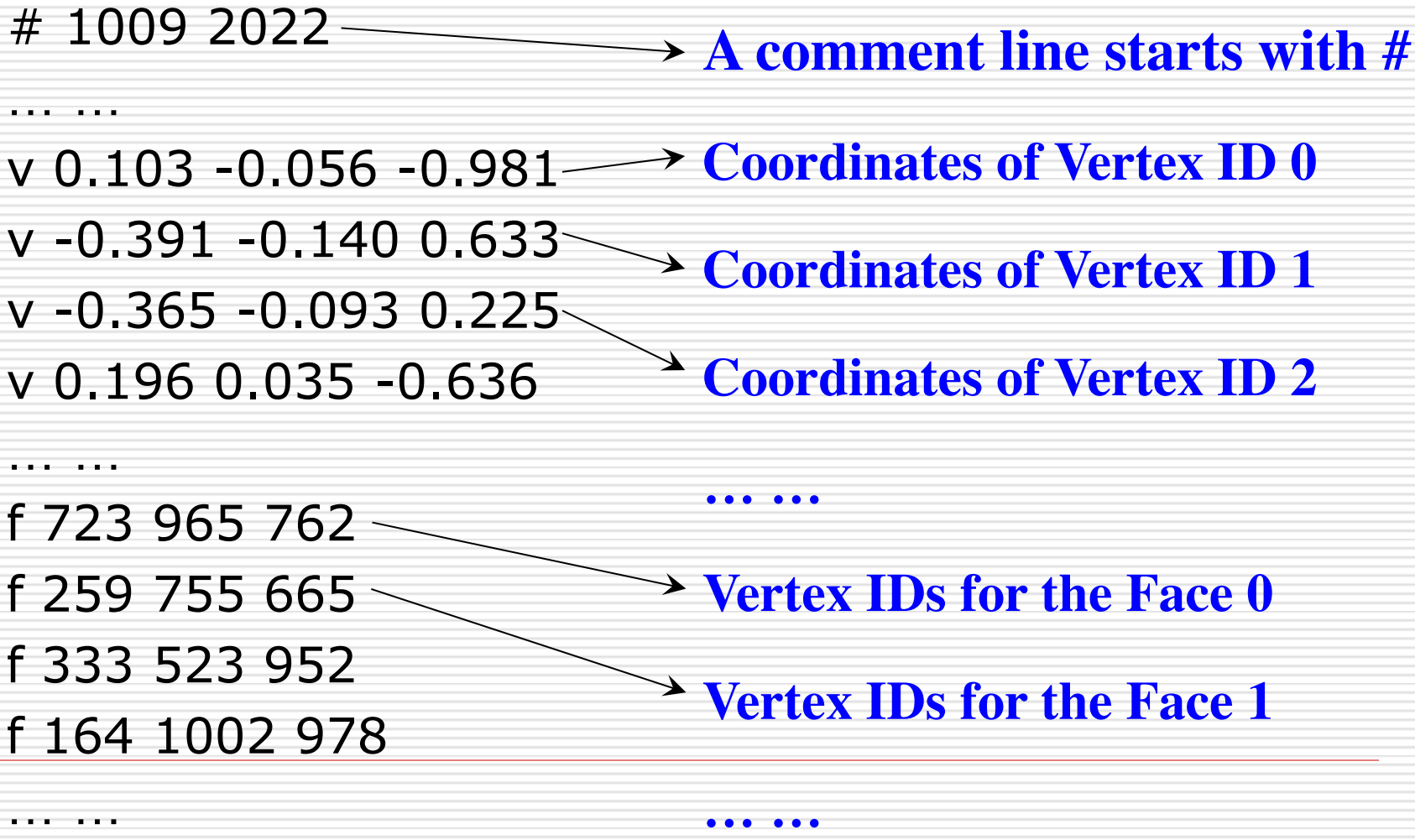
An example



Vertex position list	
v_1	(x_1, y_1, z_1)
v_2	(x_2, y_2, z_2)
v_3	(x_3, y_3, z_3)
v_4	(x_4, y_4, z_4)

Face list	
f_1	(v_1, v_2, v_3)
f_2	(v_1, v_4, v_2)
f_3	(v_2, v_4, v_3)
f_4	(v_1, v_3, v_4)

An example


1009 2022
A comment line starts with #
… …
v 0.103 -0.056 -0.981
Coordinates of Vertex ID 0
v -0.391 -0.140 0.633
Coordinates of Vertex ID 1
v -0.365 -0.093 0.225
Coordinates of Vertex ID 2
v 0.196 0.035 -0.636
… …
… …
f 723 965 762
Vertex IDs for the Face 0
f 259 755 665
Vertex IDs for the Face 1
f 333 523 952
f 164 1002 978
… …
… …

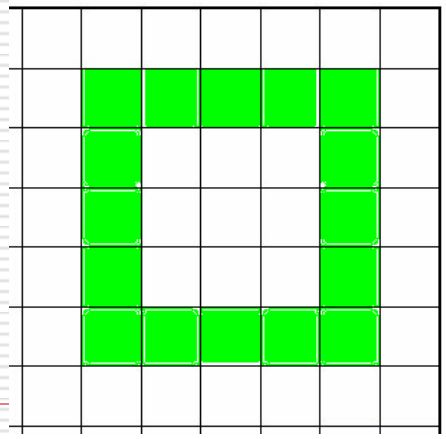
Polygonal Meshes

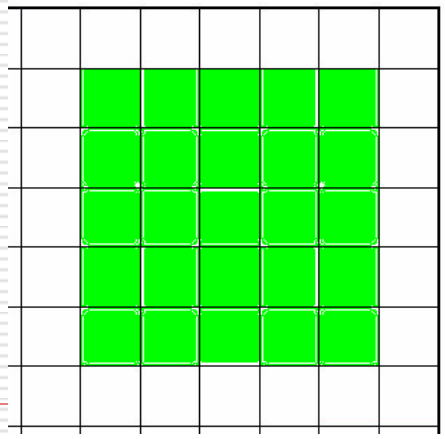
- Polygons are easy to represent (by a sequence of vertices) and transform
- They have simple properties (a single normal vector, a well-defined inside and outside, etc.)
- They are easy to draw (using a polygon-fill routine, or by mapping texture onto the polygon)

```
glPolygonMode(parameter1, parameter2);
glBegin(GL_POLYGON);
        glVertex3f(-0.5,-0.5,0.0);
        … … …
glEnd();
```

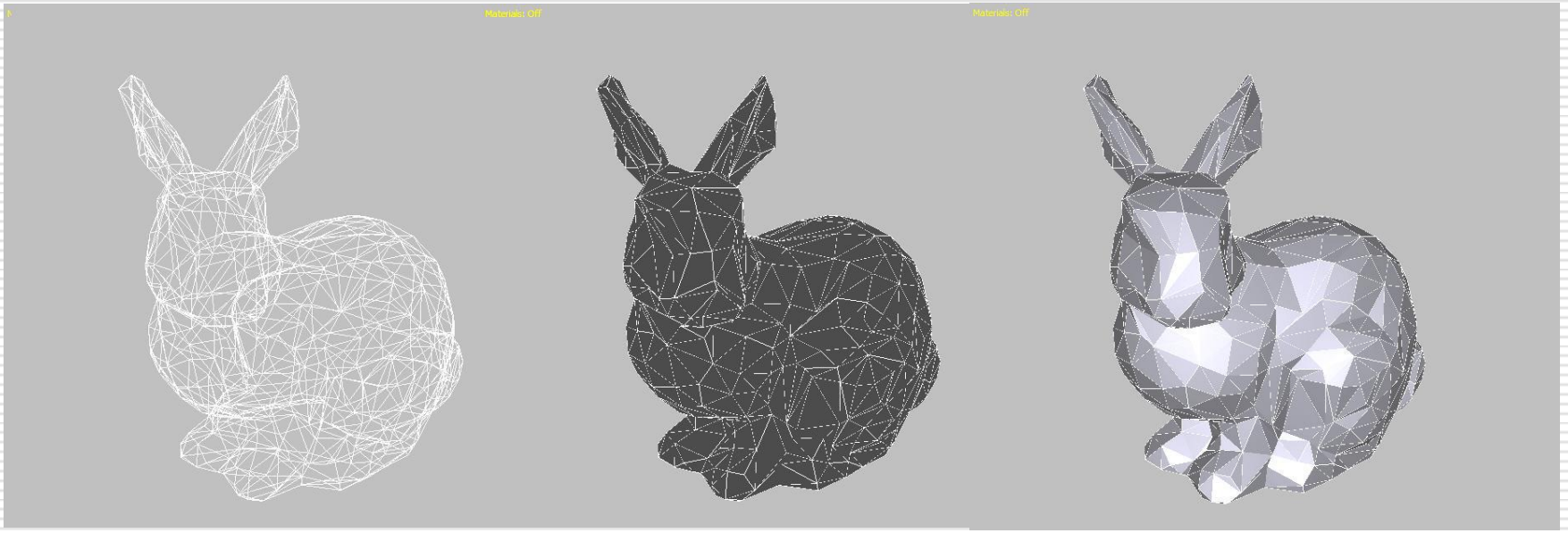
parameter2: GL_LINE, GL_FILL

parameter1: GL_FRONT, GL_BACK





Polygonal Meshes



GL_LINE

GL_FILL without lighting

GL_FILL with lighting

Polygonal Meshes

- Meshes are a standard way of representing 3D objects in graphics

  One more example with details

Example II

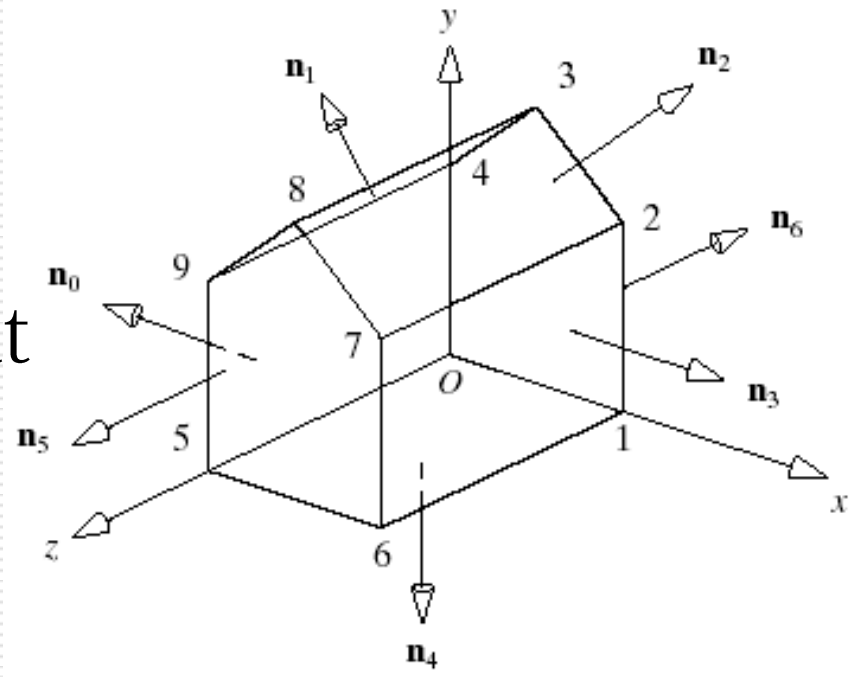
Defining a Polygonal Mesh

- 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)

Example II

Barn model

- 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



Example II

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

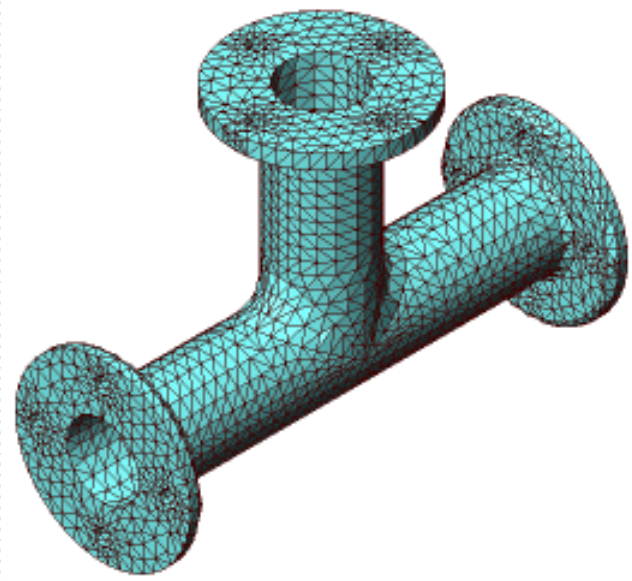
Face List for the Barn

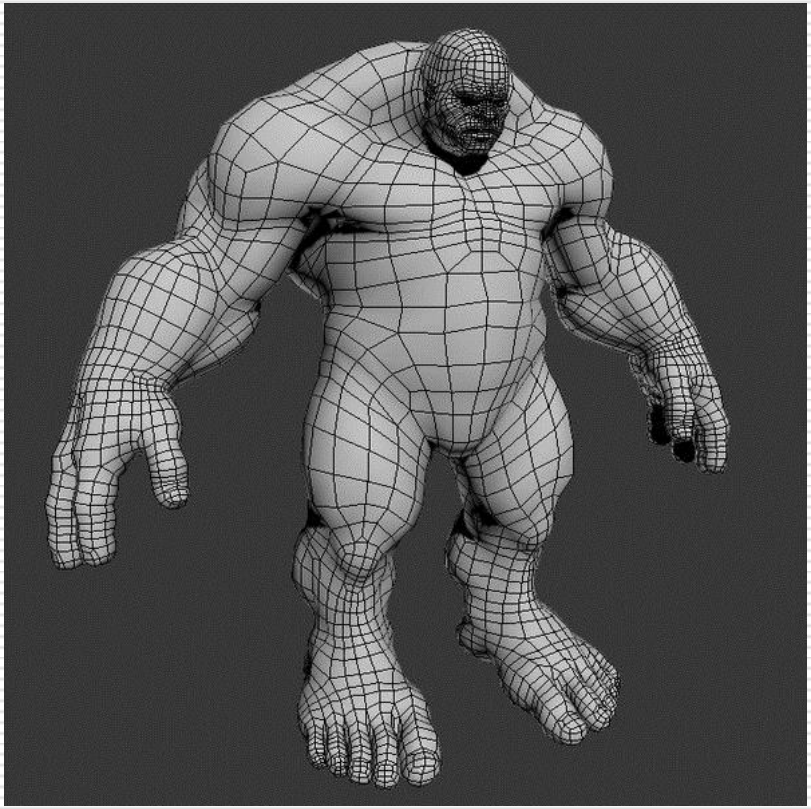
Face	Vertices	Normal
0 (left)	0, 5, 9, 4	0,0,0,0
1 (roof left)	3, 4, 9, 8	1,1,1,1
2 (roof right)	2, 3, 8, 7	2, 2, 2,2
3 (right)	1, 2, 7, 6	3, 3, 3, 3
4 (bottom)	0, 1, 6, 5	4, 4, 4, 4
5 (front)	5, 6, 7, 8, 9	5, 5, 5, 5, 5
6 (back)	0, 4, 3, 2, 1	6, 6, 6, 6, 6

Polygonal Meshes

- 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





Polygonal Meshes

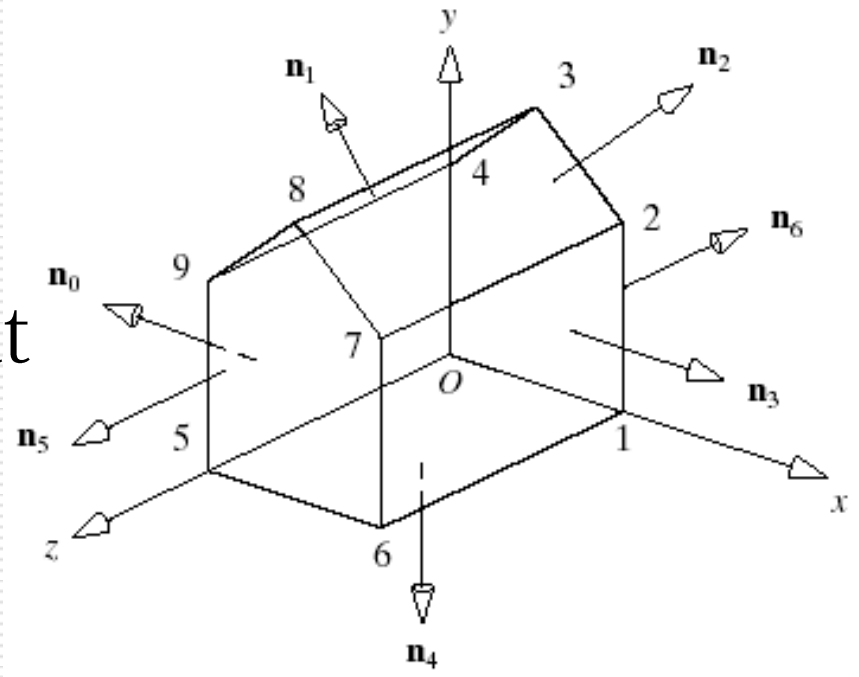
Polygonal Meshes

- 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

Example II

Barn model

- 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



Polygonal Meshes

- 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

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

Example II

A mesh consists of 3 lists

- The vertices of the mesh
- The outside normal at each vertex
- The faces of the mesh

Calculating Normals

Take any three non-collinear points on the face, V1, V2, and V3, and compute the normal as their cross product m = (V1 - V2) $\times$ (V3 - V2) and normalize it to unit length

- If the two vectors V1 - V2 and V3 - V2 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 Normals

- Given N vertices, define next(i) = ni = (i+1) mod N
- Traverse the vertices for the face in counter-clockwise order from the outside
- The normal given by the values on the next slide points to the outside (front) of the face

Newell's Method for Normals

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

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

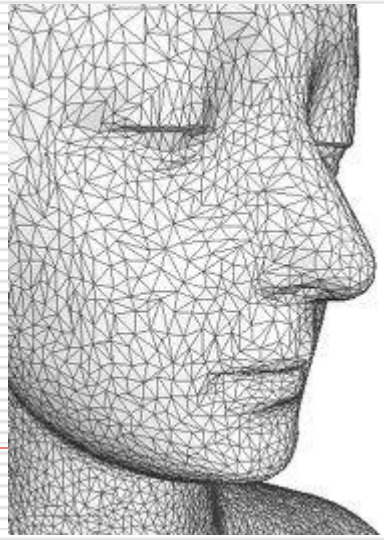
$$n_z = \sum_{i=0}^{N-1} (x_i - x_{ni})(y_i + y_{ni})$$

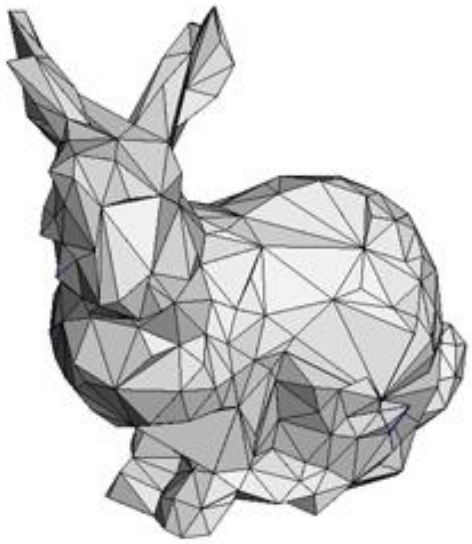
Properties of Meshes

- 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.

Polygonal Meshes

- Meshes are a standard way of representing 3D objects in graphics
- Where the 3D polygonal meshes come from?

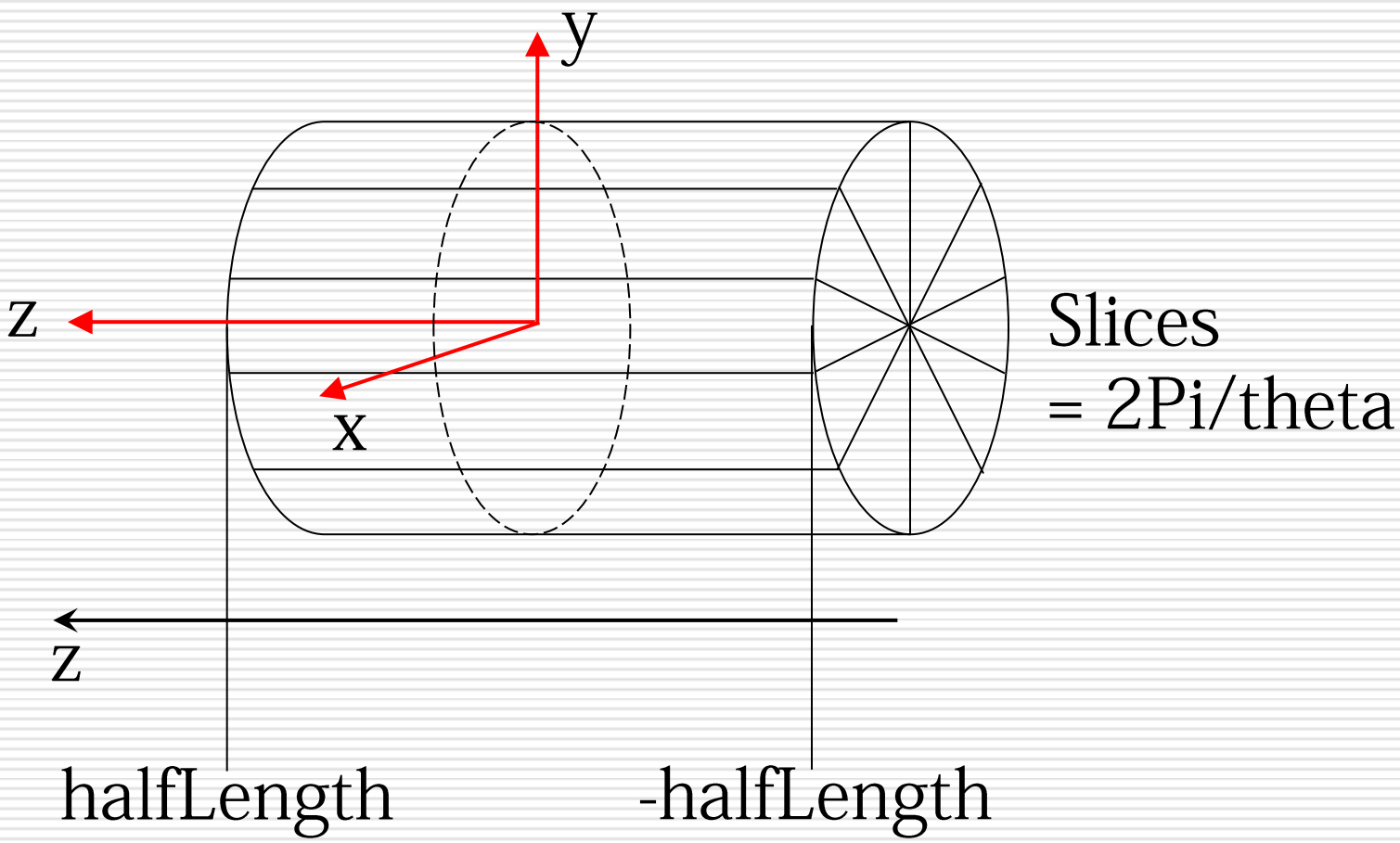




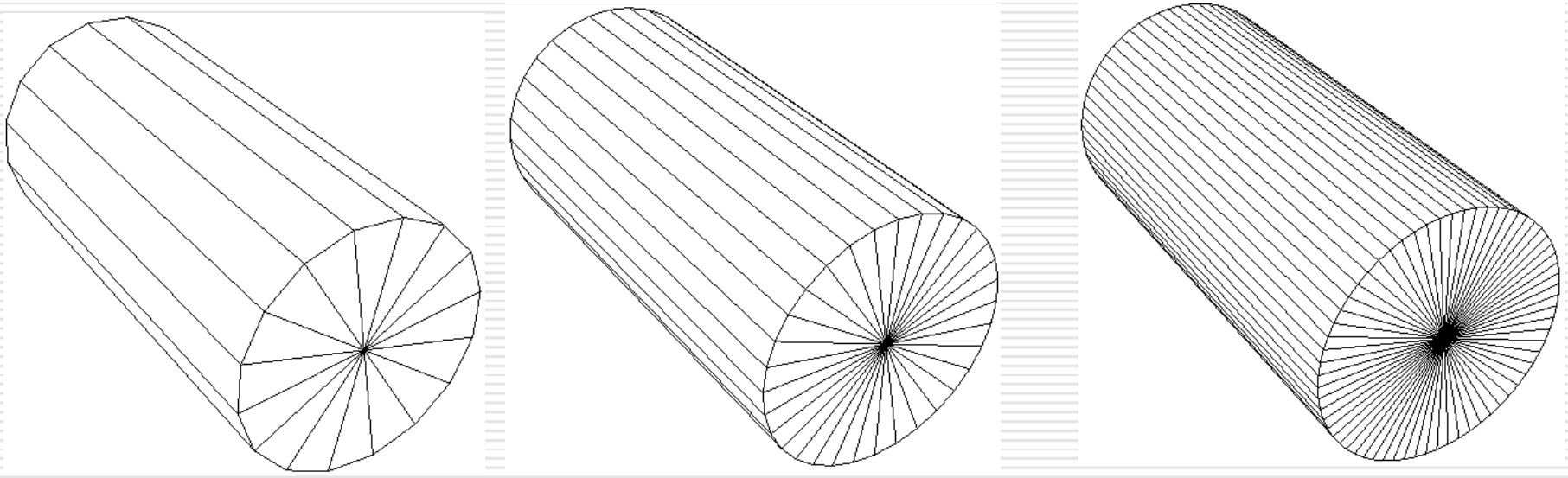
Sources of polygonal Meshes

- Input a mesh file from storage
- Using modeling software
  3DS/MAX, Maya, SolidWorks, etc
- Procedure modeling:
  Programming codes to general 3D shape with analytic forms

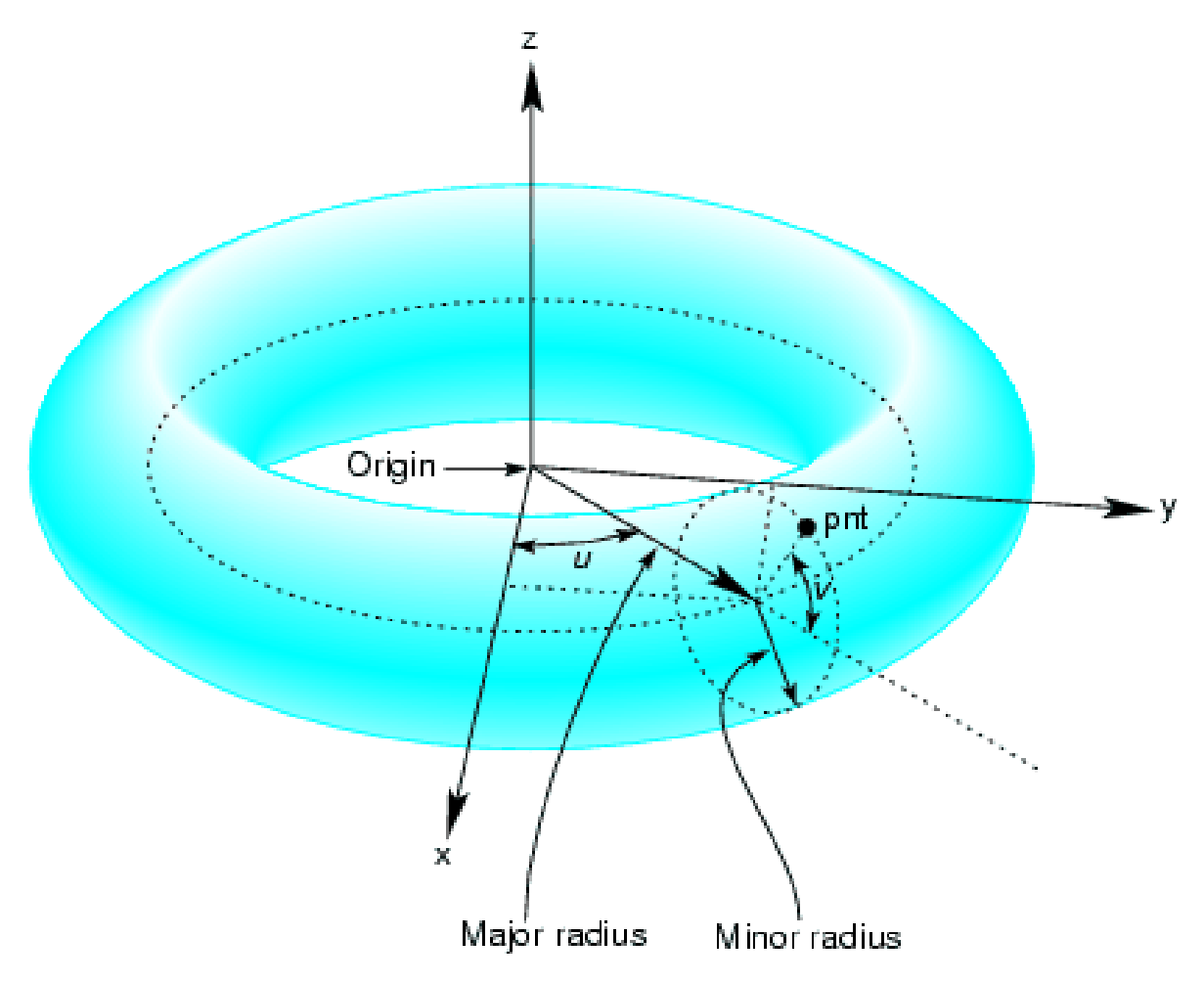
radius = 1.0; length = 2.0; slices = 32



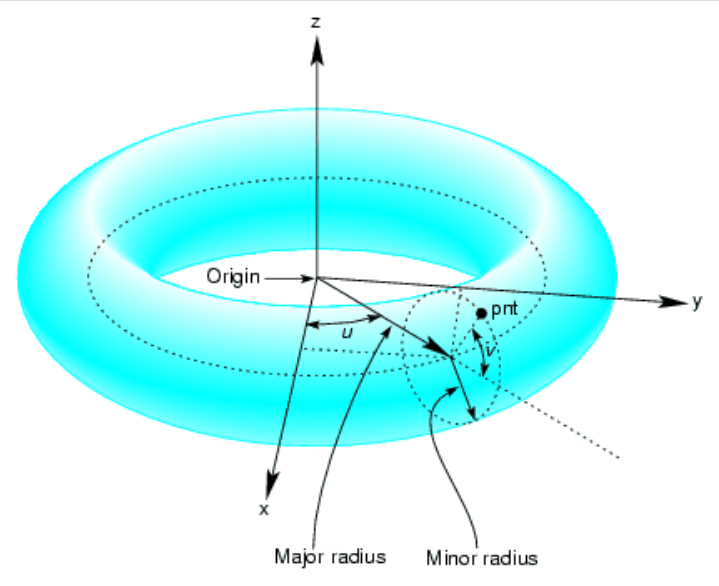
Parameterized cylinders

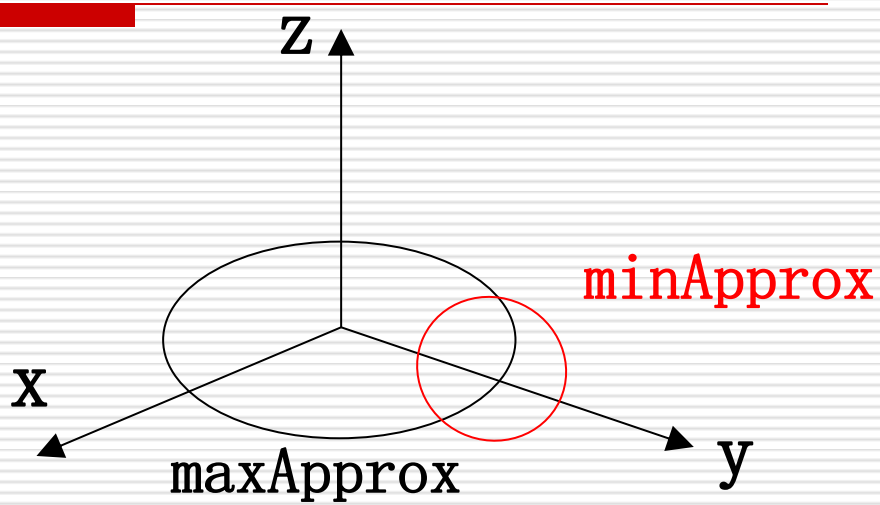


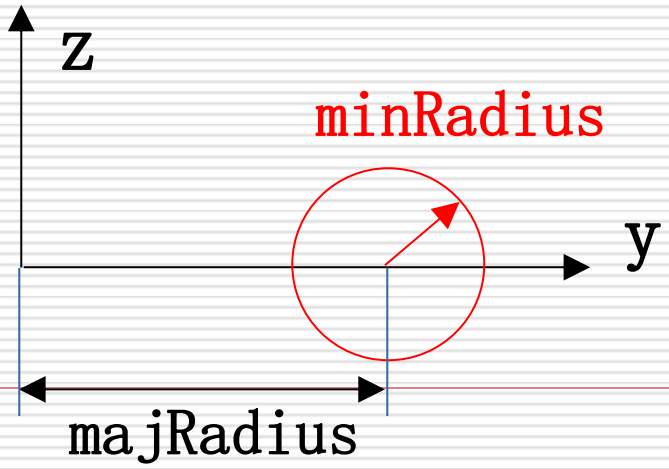
Parameterized torus

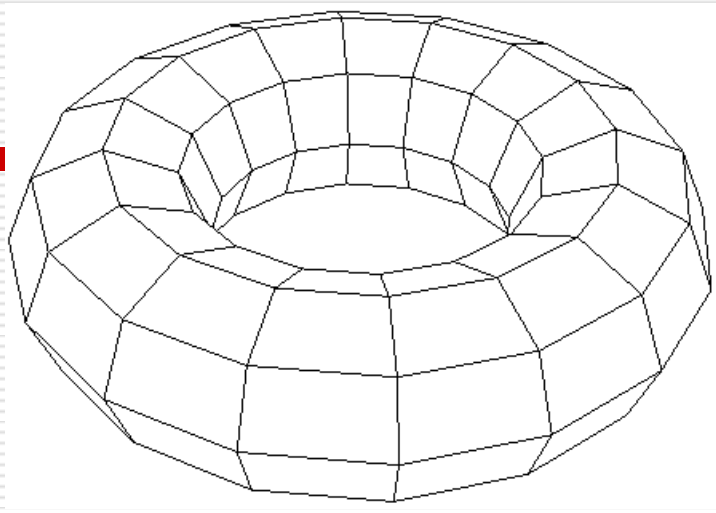


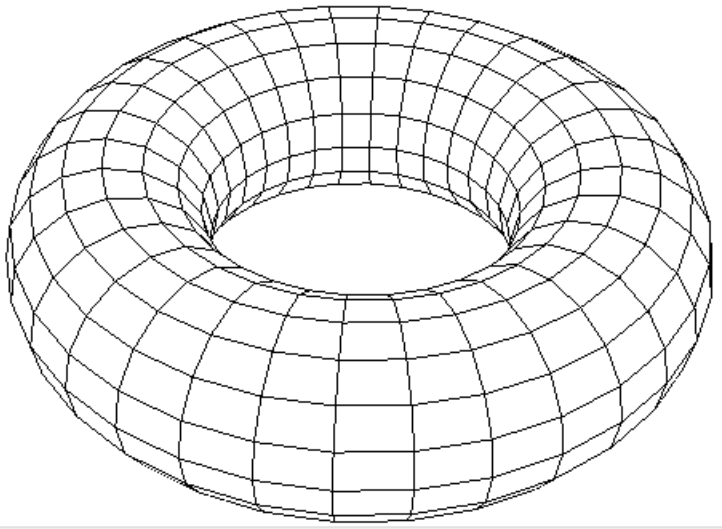
Parameterized torus

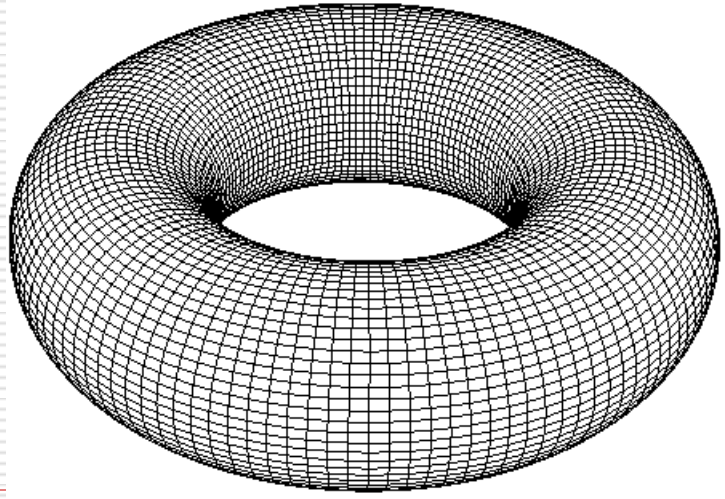


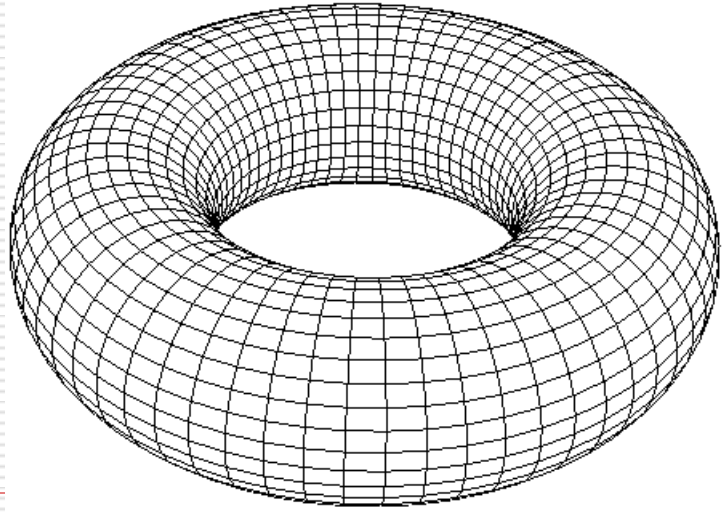


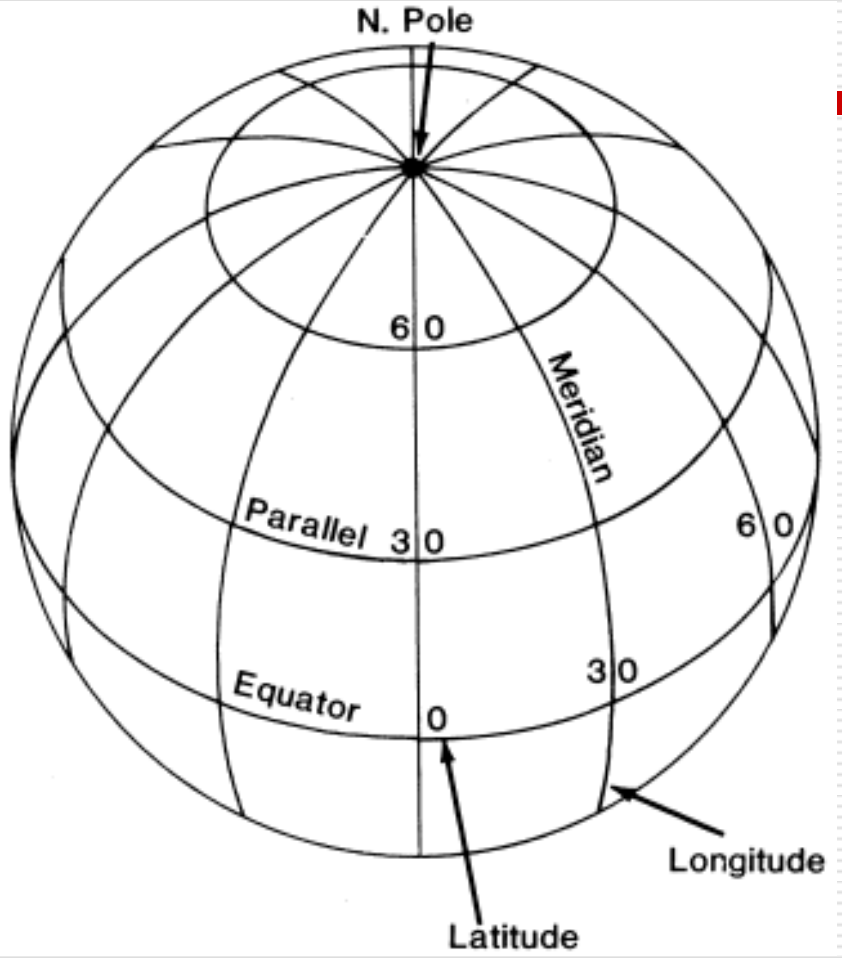


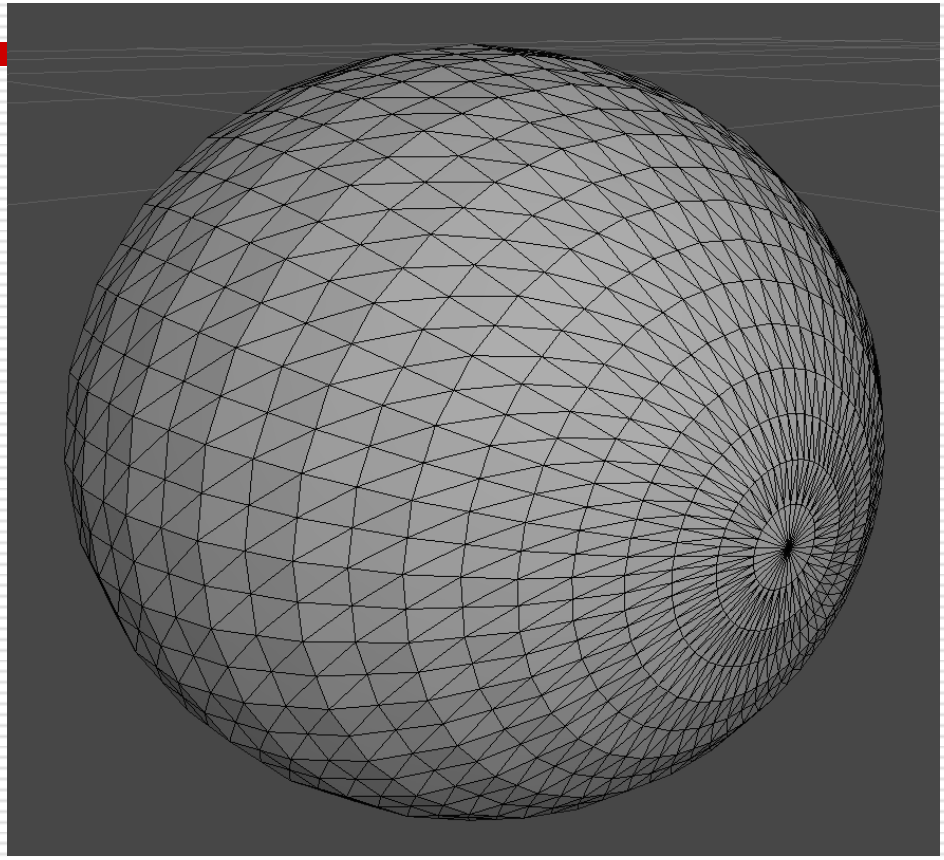




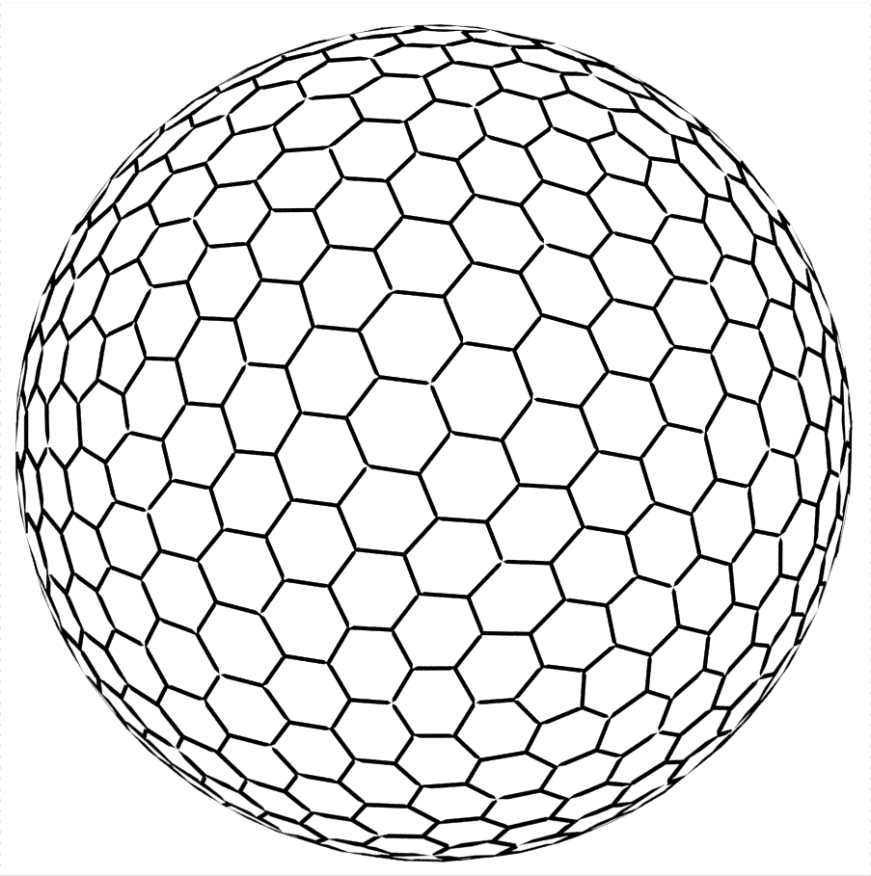




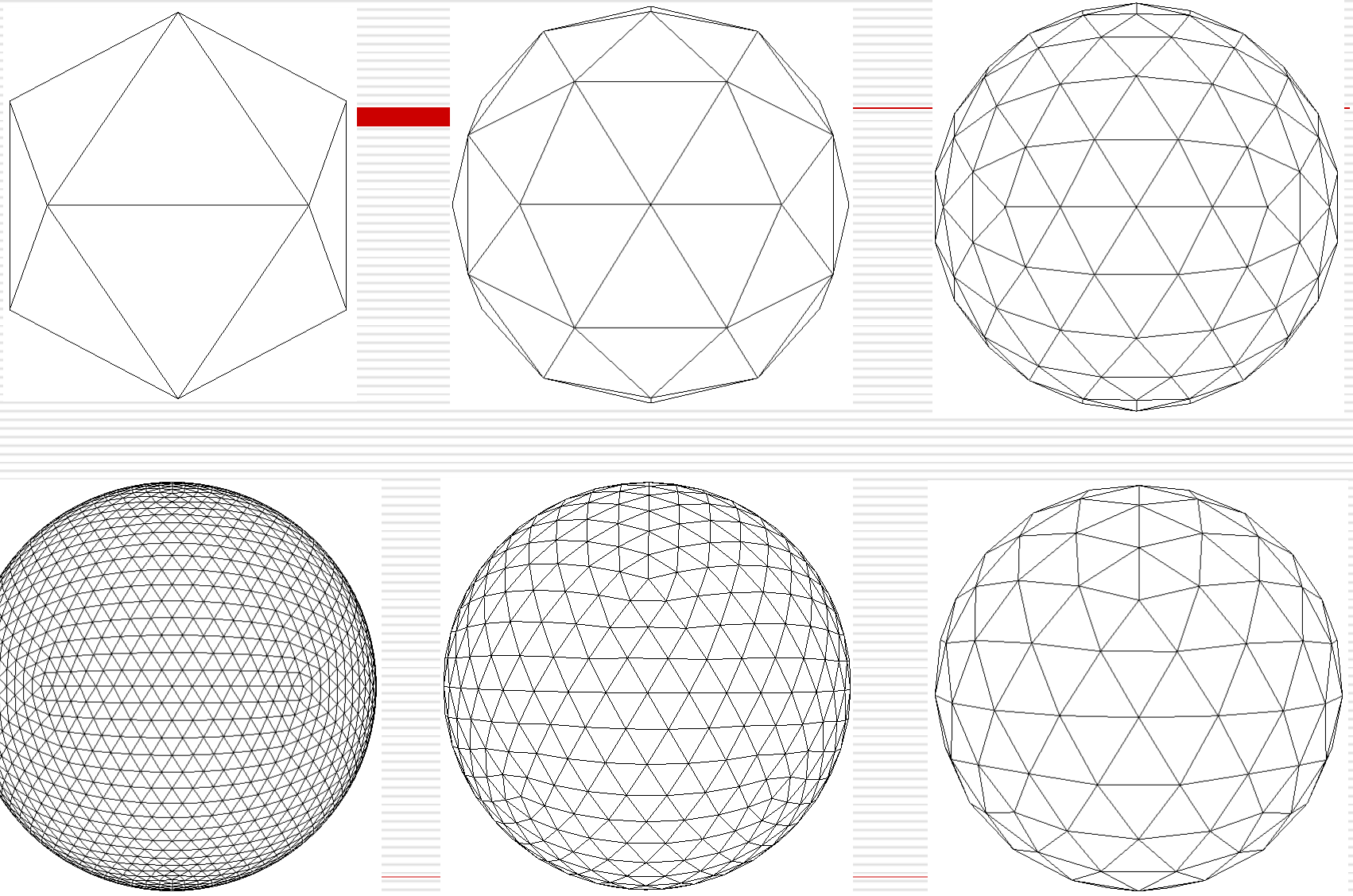

N. Pole
60
Meridian
Parallel 30
60
30
Equator
0
Longitude
Latitude

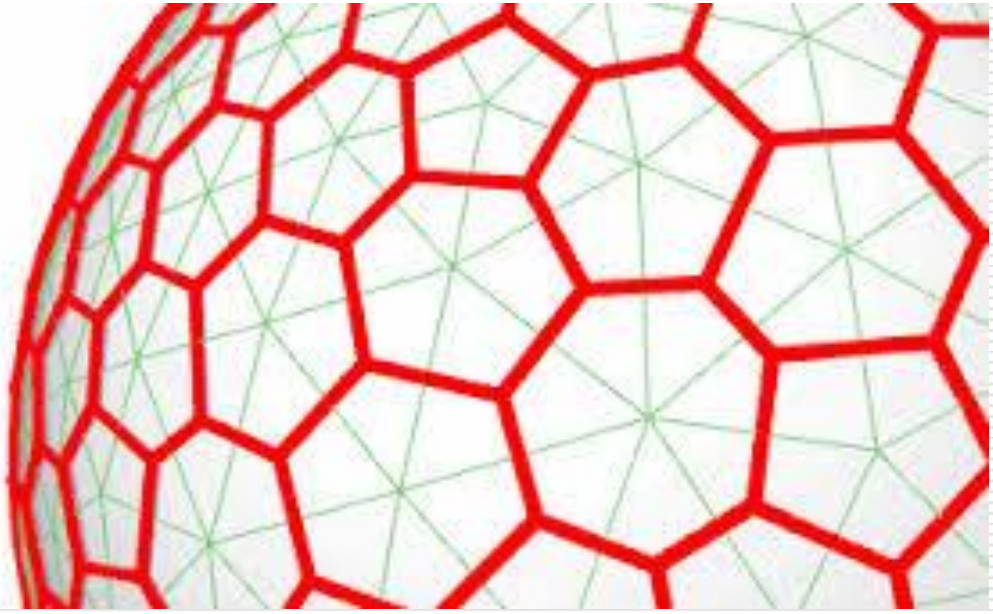


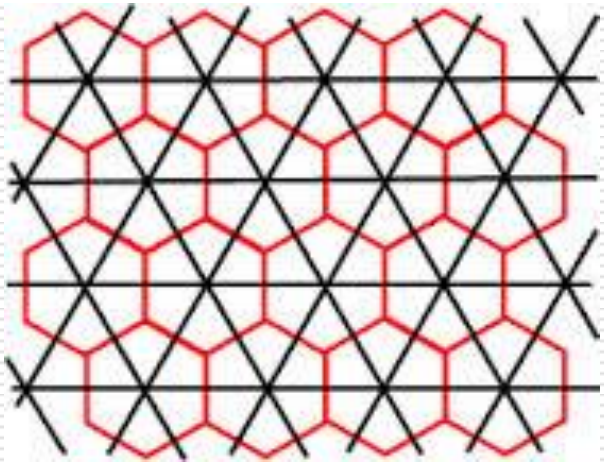
Procedure modeling



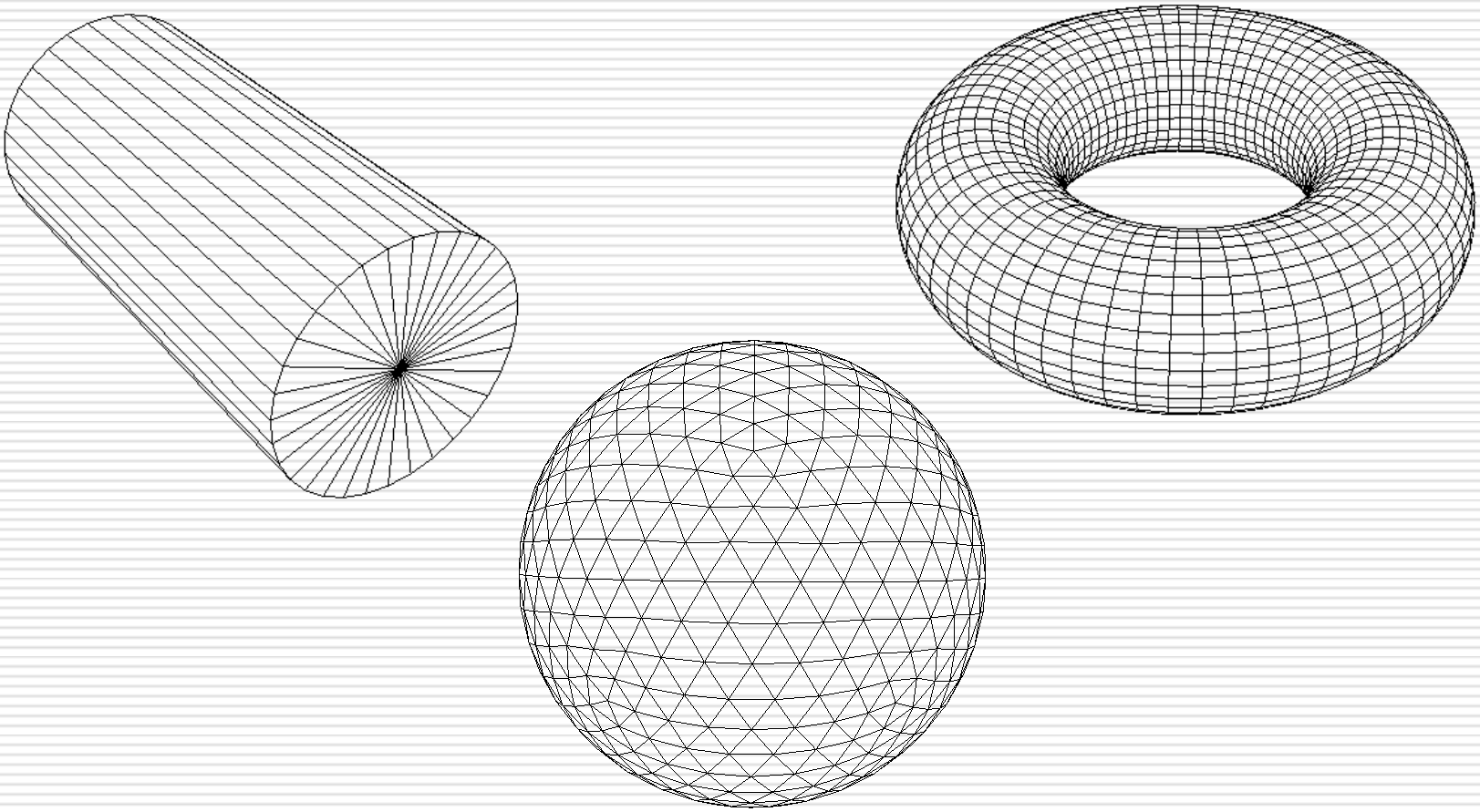
Regular icosahedron



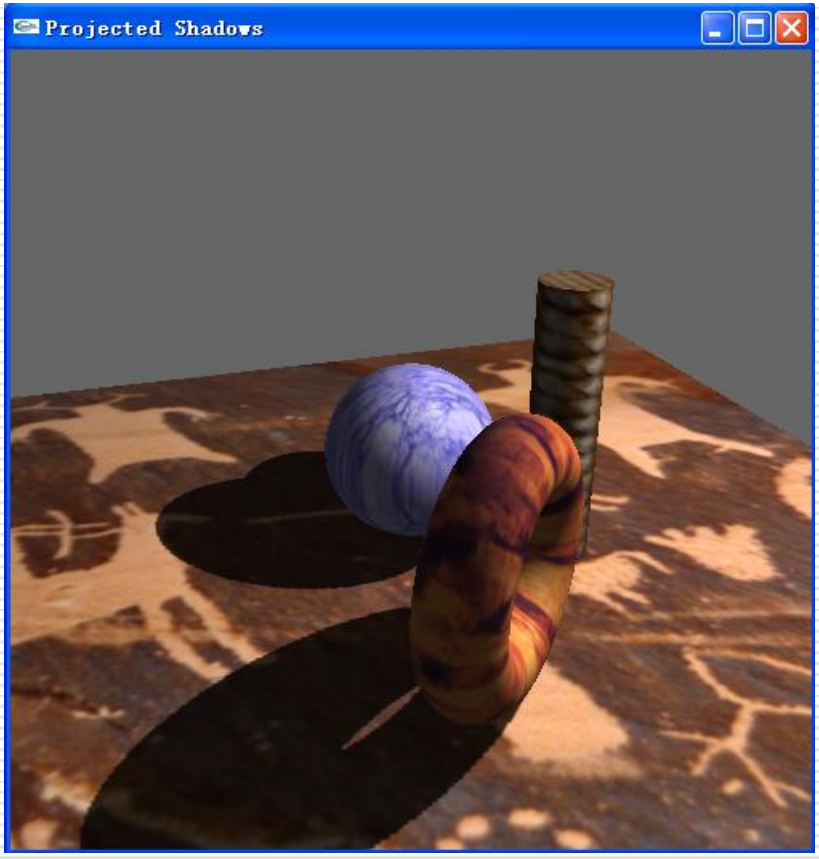


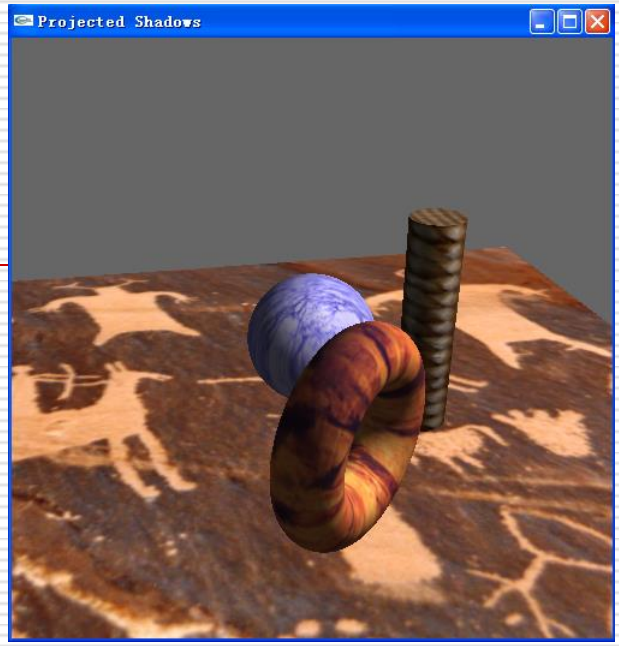


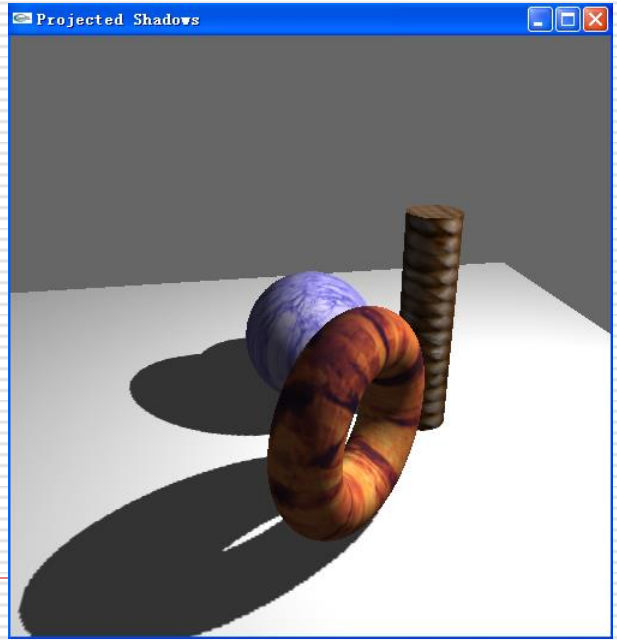
Procedure modeling



Texture mapping



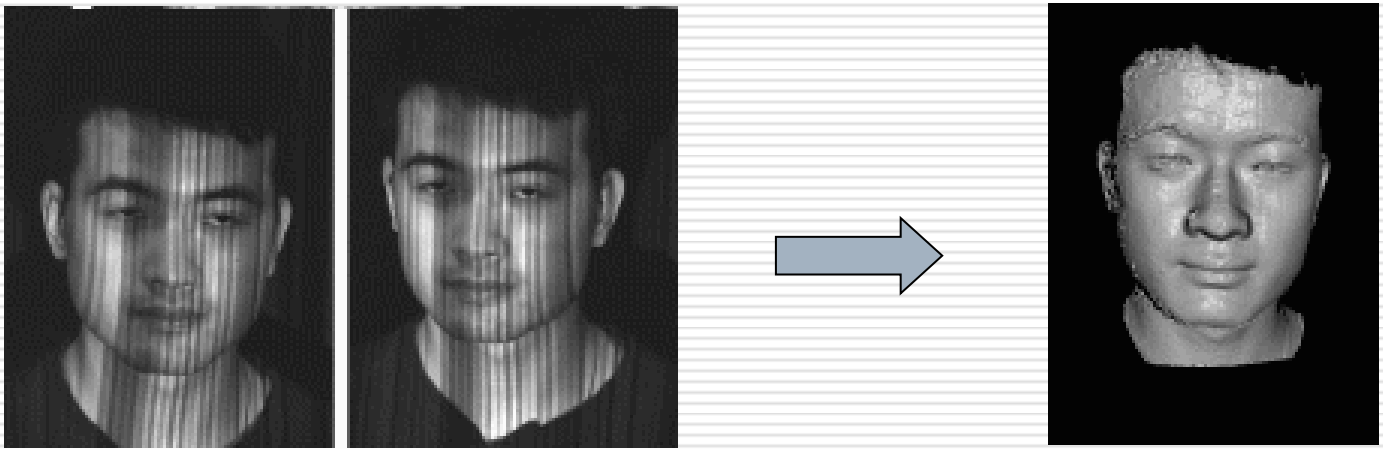


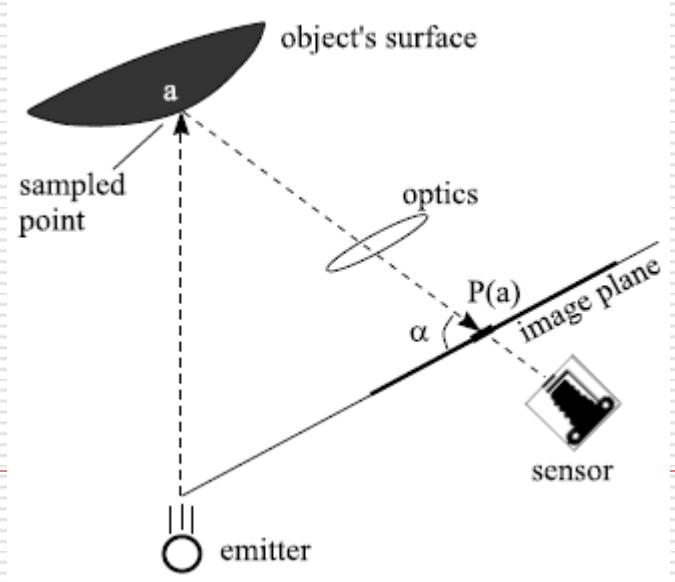


Sources of polygonal Meshes

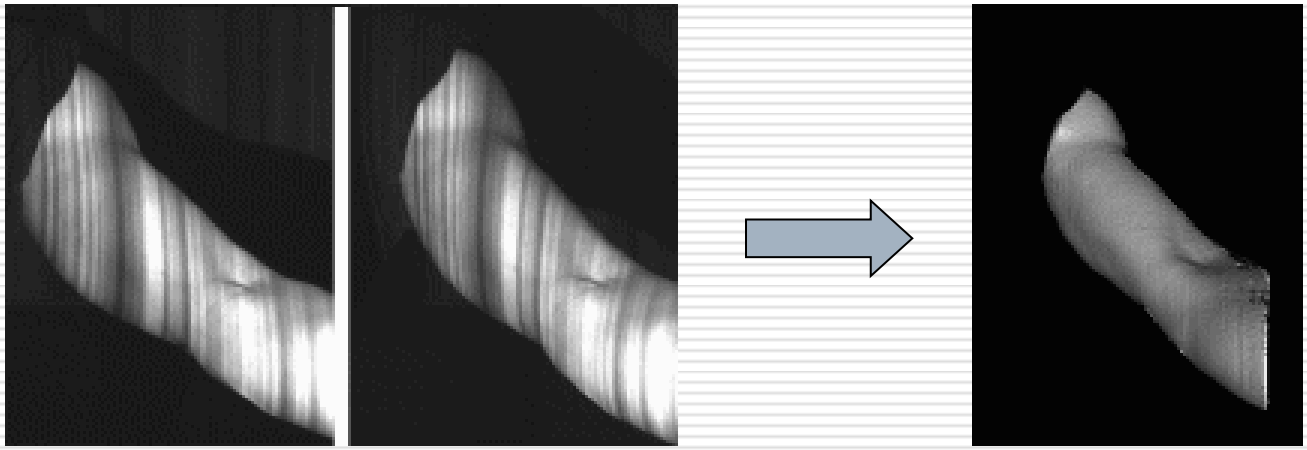
- Input a mesh file from storage
- Using modeling software
  3DS/MAX, Maya, SolidWorks, etc
- Procedure modeling:
  Programming codes to general 3D shape with analytic forms
- 3D scanning
  photogrammetry, Range/laser scanning

Range scanning with a projector

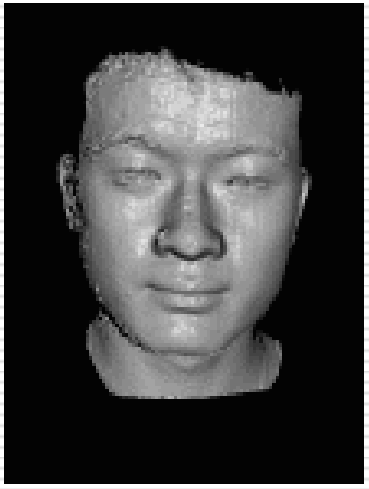




Range scanning with a projector



Range scanning with a projector

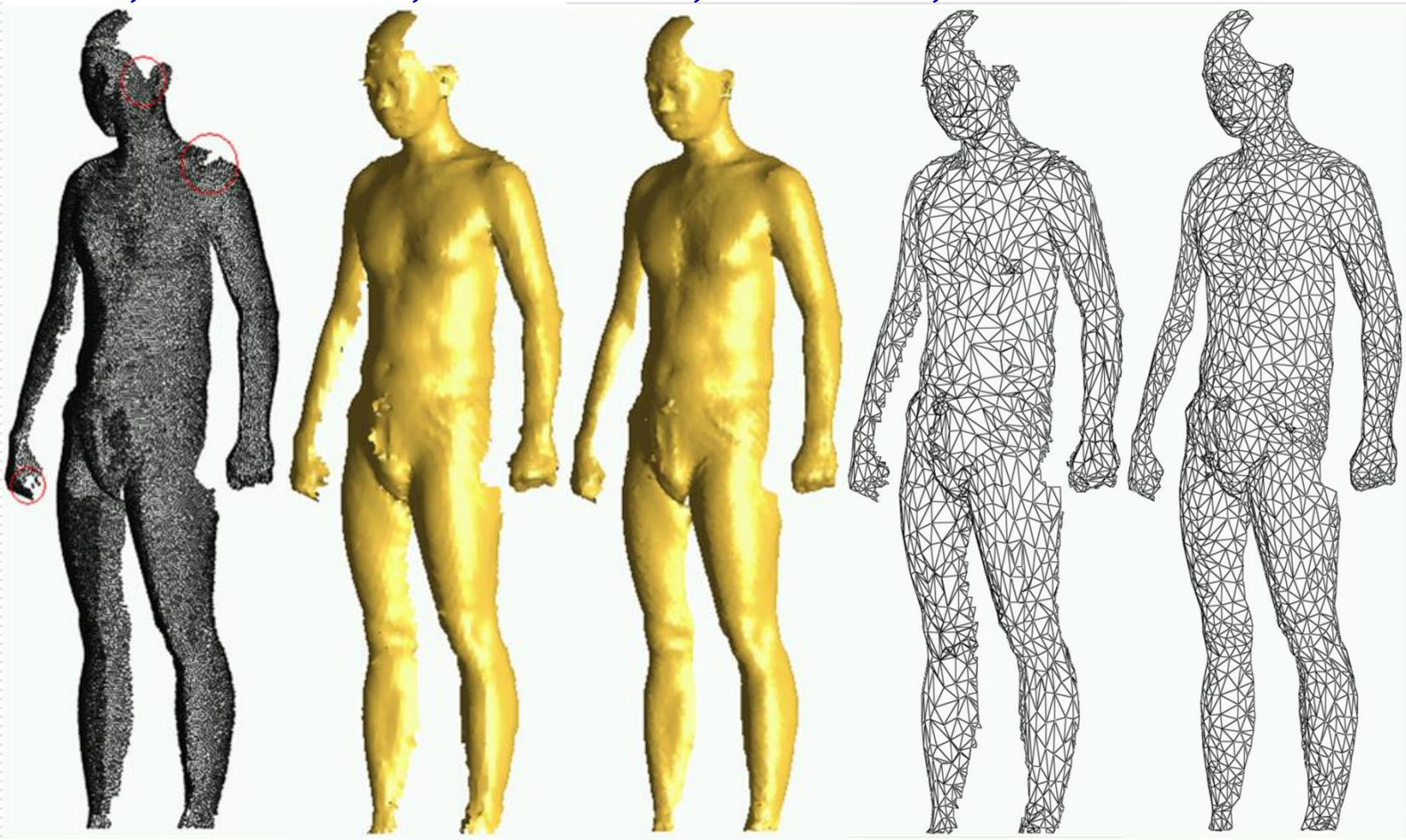


Singal processing
Smooth/denoise

Small smart scanning

Laser scanning

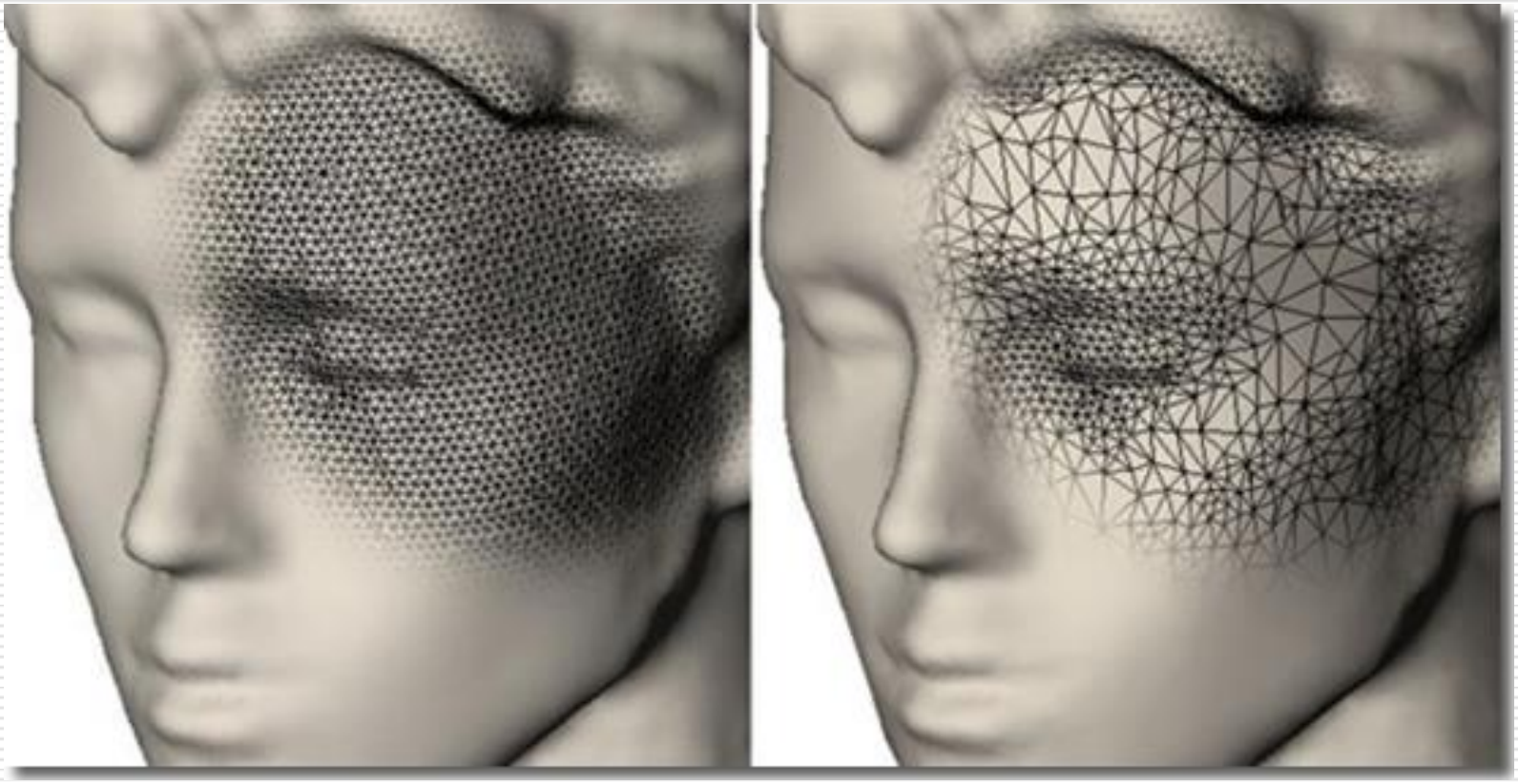
Mesh optimization



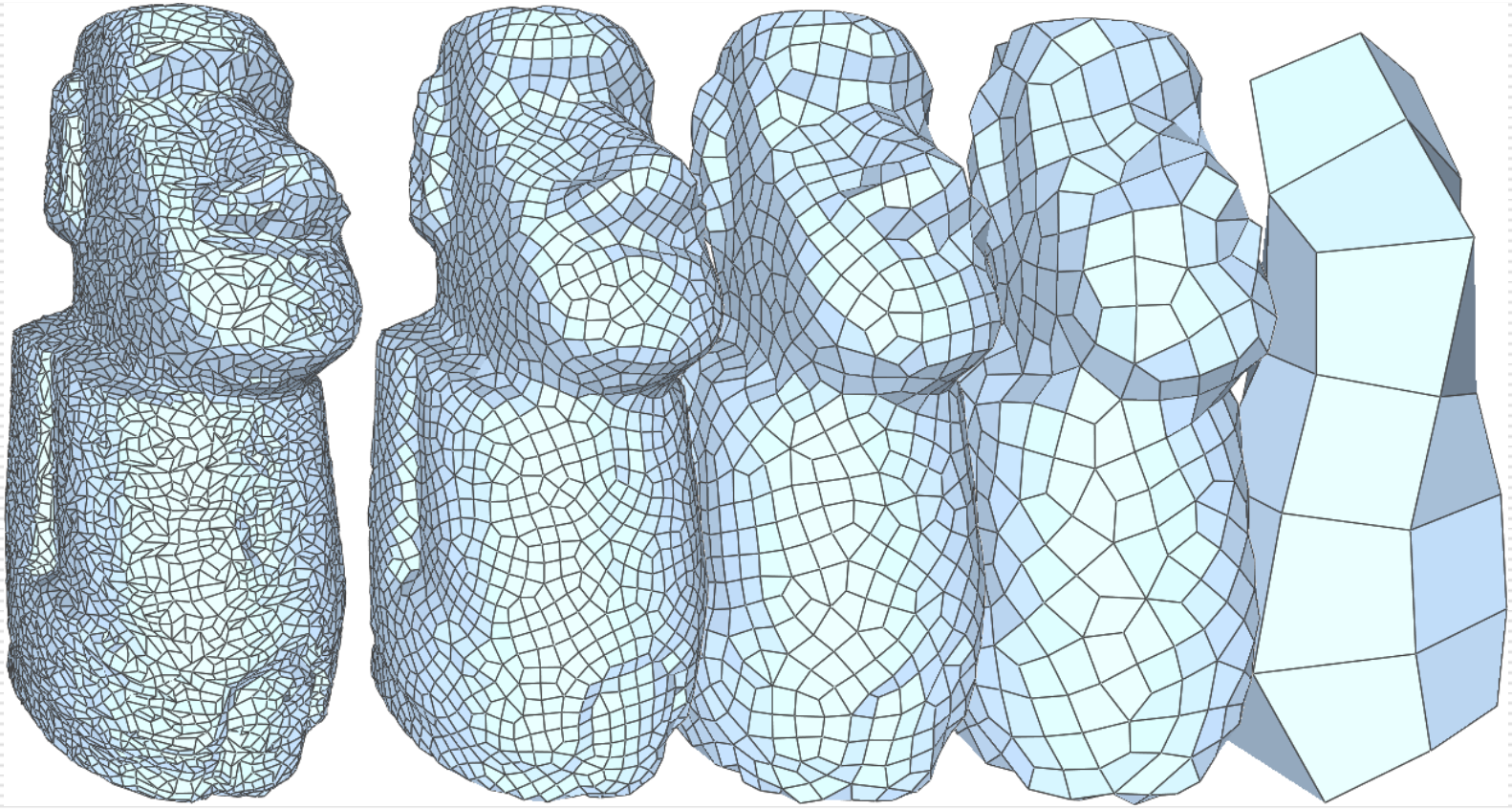
Polygonal Meshes

- 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

Mesh simplification

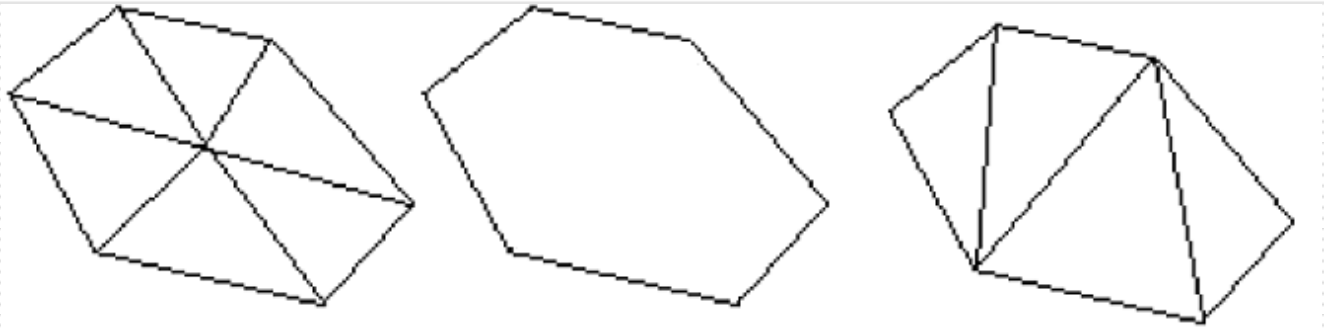


Mesh simplification



Mesh simplification

- Vertex decimation
  Every time, remove one vertex from the mesh



Mesh simplification

- Edge contraction
  Every time, remove one edge from the mesh

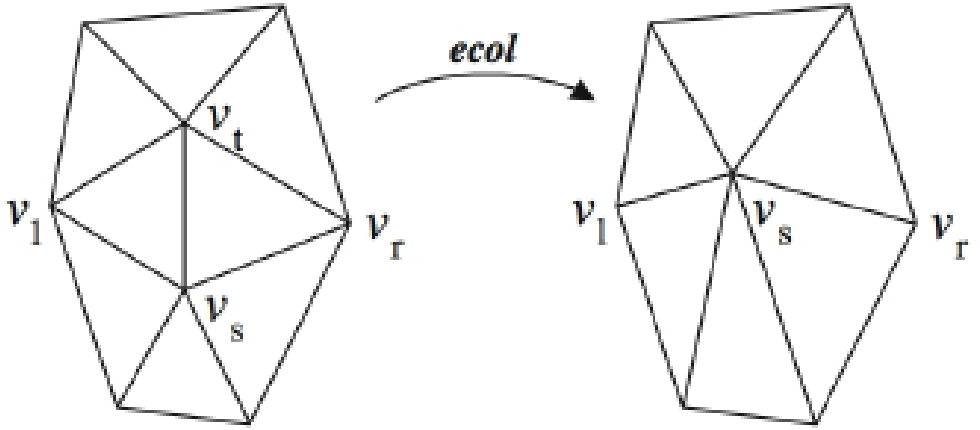
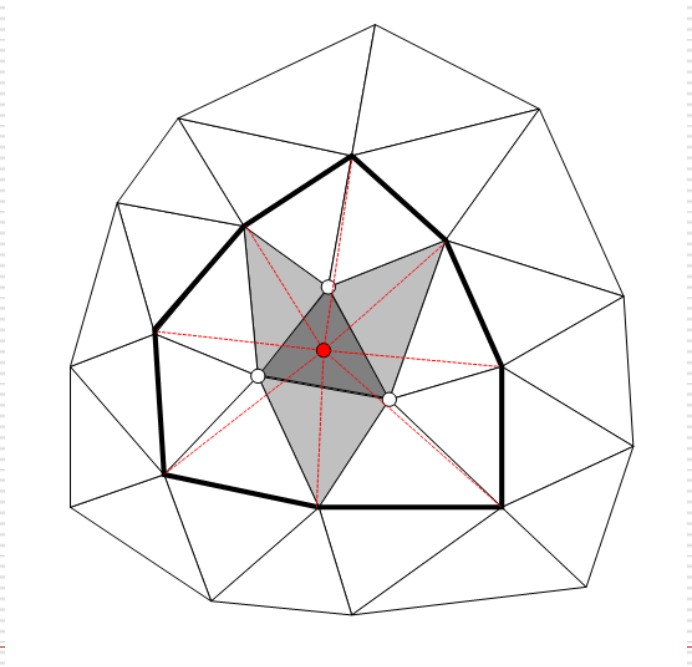


Fig. 1. The edge contraction operation [Hoppe 1993].

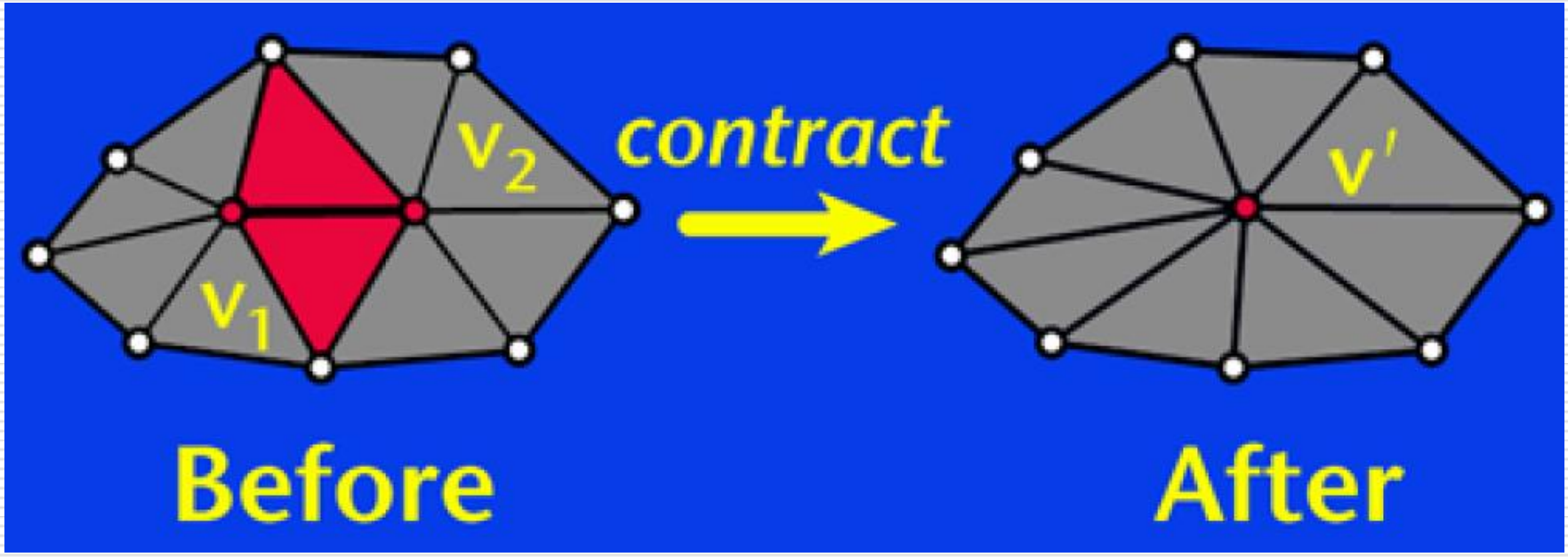
Mesh simplification

- Face contraction
  Every time, remove one face from the mesh



Simplification - Edge contraction

- Every time, remove one edge from the mesh



How We Measure Error

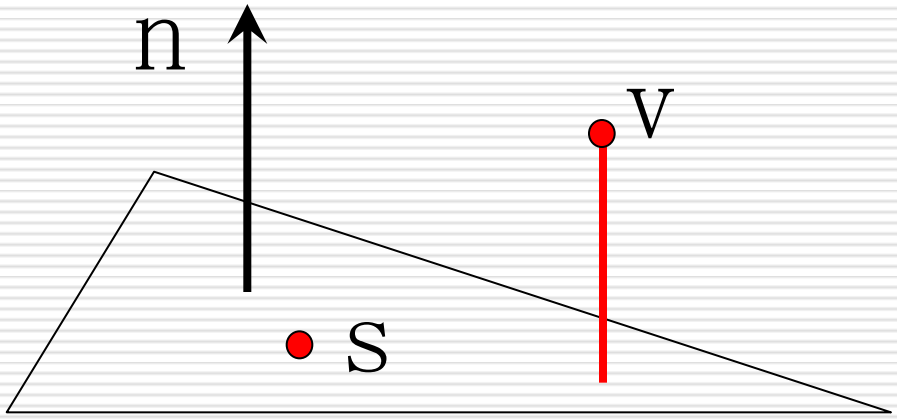
Measure error at current vertices

For a given point v, measure sum of squared distances to associated set of planes

- Each vertex v has an associated set of planes
  - *Initialize with planes of incident faces in original*
  - *Merge sets when contracting pairs*
  - *Initial error of each vertex is 0*

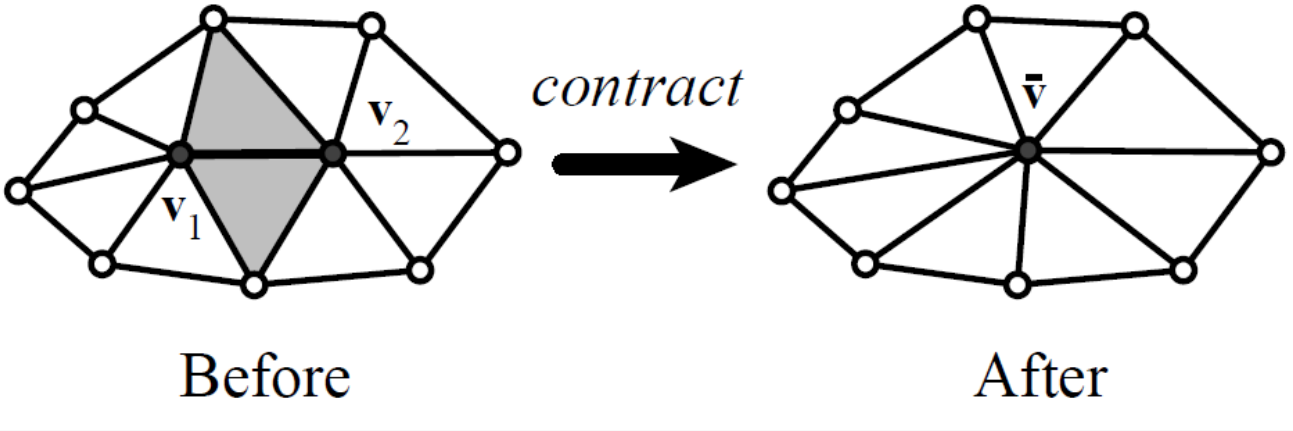
$\mathbf{p} = [a\ b\ c\ d]^T$ represents the plane defined by the equation

$ax + by + cz + d = 0$ where $a^2 + b^2 + c^2 = 1$



Error of vertex:

Sum of squared distances



$$\text{planes}(\bar{\mathbf{v}}) = \text{planes}(\mathbf{v}_1) \cup \text{planes}(\mathbf{v}_2)$$

$$\Delta(\mathbf{v}) = \Delta([v_x\ v_y\ v_z\ 1]^\mathsf{T}) = \sum_{\mathbf{p} \in \text{planes}(\mathbf{v})} (\mathbf{p}^\mathsf{T}\mathbf{v})^2$$

Rewrite as a quadratic form:

$$\begin{aligned}\Delta(\mathbf{v}) &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{v}^\top\mathbf{p})(\mathbf{p}^\top\mathbf{v}) \\ &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{v}^\top(\mathbf{p}\mathbf{p}^\top)\mathbf{v} \\ &= \mathbf{v}^\top\left(\sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{K}_\mathbf{p}\right)\mathbf{v}\end{aligned}$$

where $\mathbf{K}_\mathbf{p}$ is the matrix:

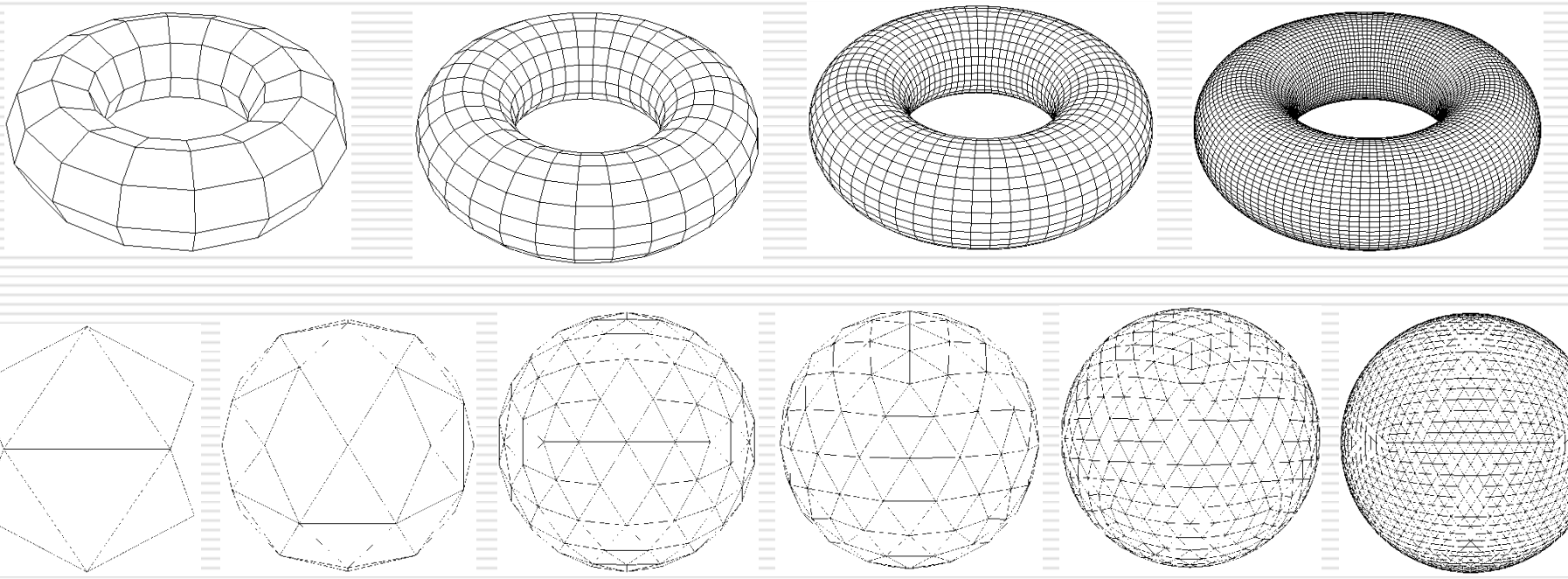
$$\mathbf{K}_\mathbf{p} = \mathbf{p}\mathbf{p}^\top = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Polygonal Meshes

- 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

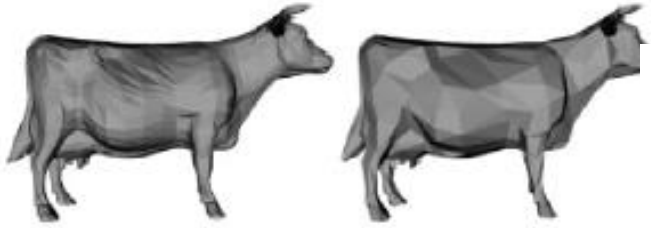
Mesh refinement

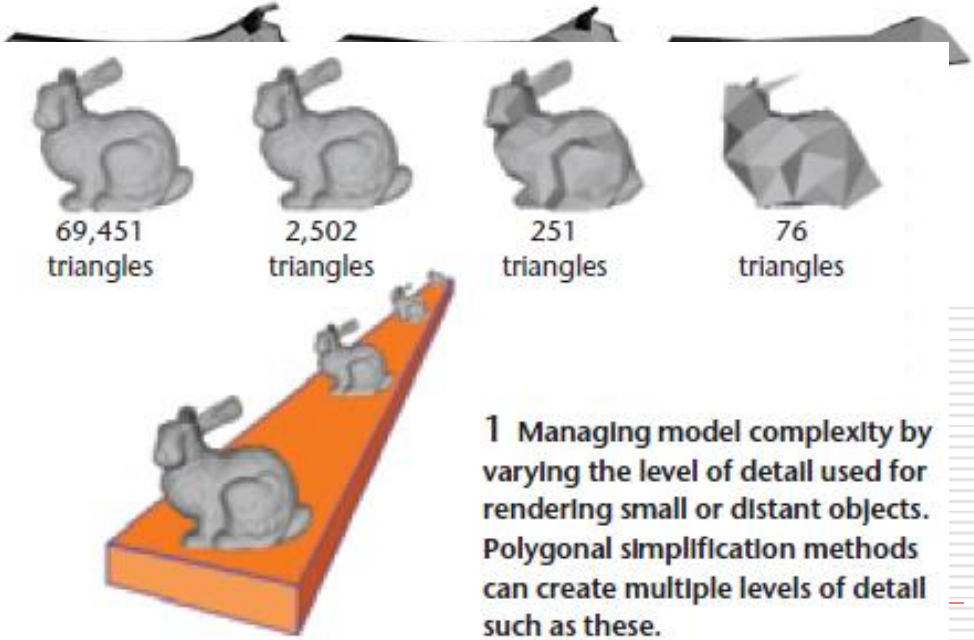
- Insert more faces into the mesh



Example

- Level-of-Details (LOD)

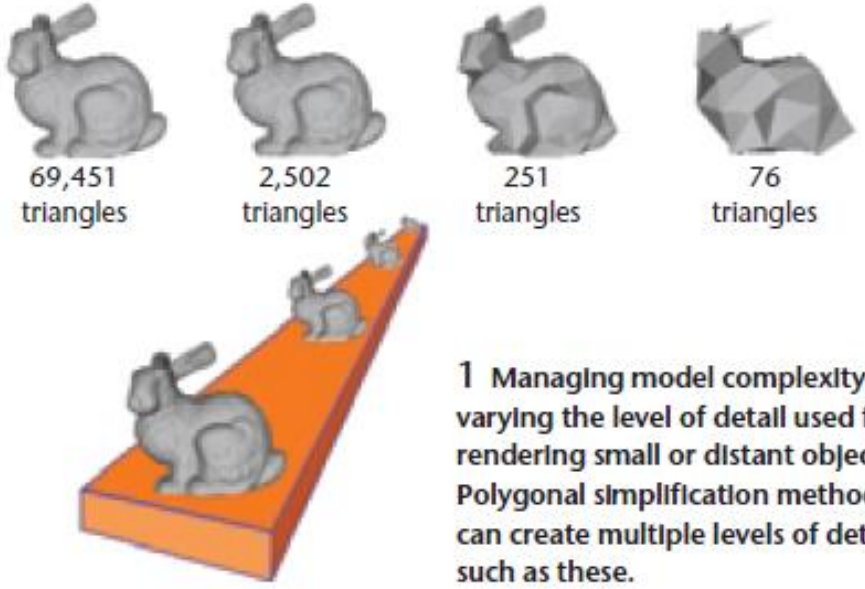




1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

Example

- Level-of-Details (LOD)
  When the object is far
  from the viewpiont,
  using the low-resolution;
  When the object is near,
  using the fine resolution



1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

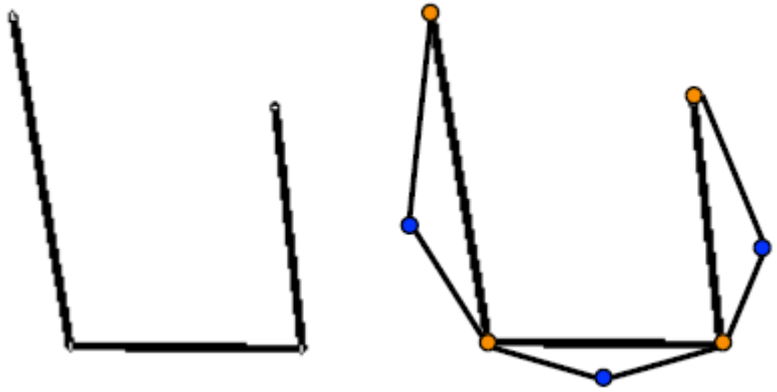
Subdivision

- How do you make a smooth curve?

We want to “smooth out” severe angles

Subdivision

- How do you make a smooth curve?



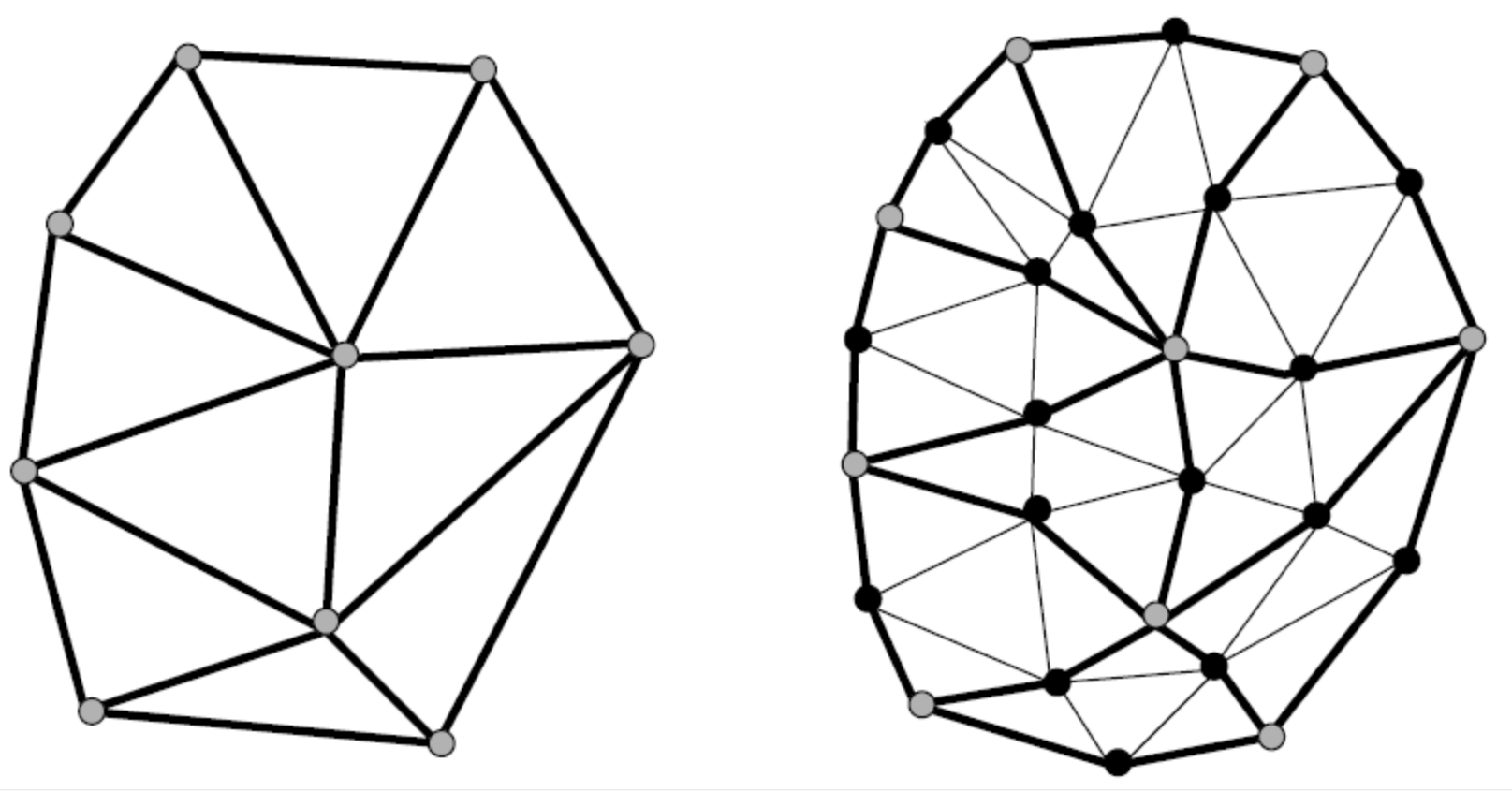
We want to “smooth out” severe angles

Subdivision

- How do you make a smooth curve?

We want to “smooth out” severe angles

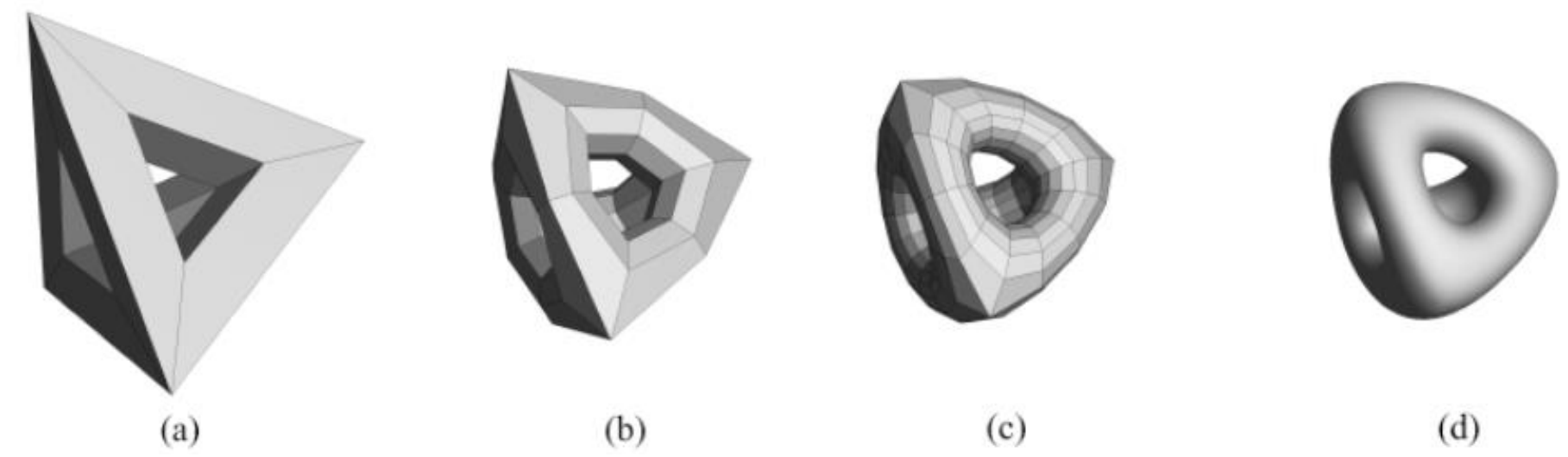
Subdivision surface



Subdivision surfaces

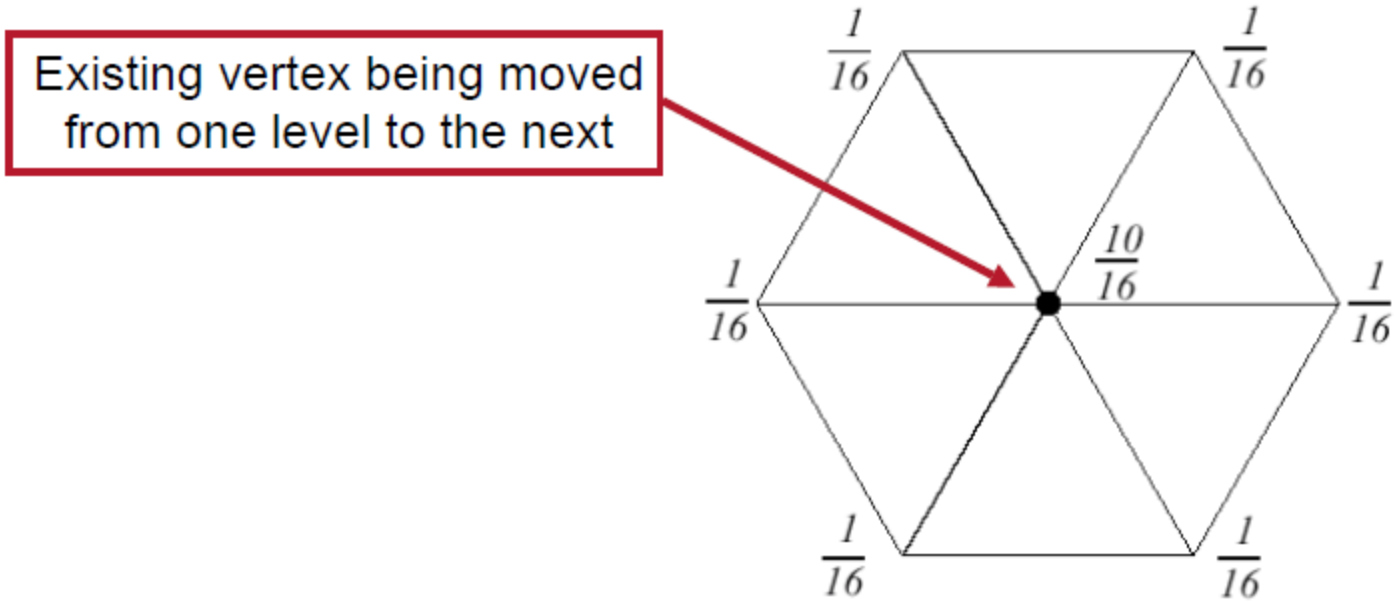
- Interpolation subdivision surface
  - Old vertices do not move
  - Only the new splitting vertices move
- Approximation subdivision surface
  - All the vertices (including old and new ones) are moved at every iteration

Loop subdivision surfaces



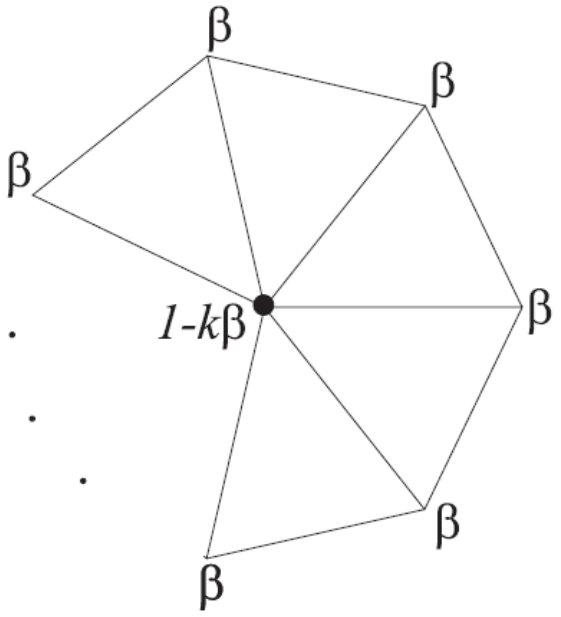
Loop Subdivision Scheme

- How to subdivide the mesh:
  - Refinement
  - Smoothing:
    - » Existing Vertices: Choose *new* location as weighted average of *original* vertex and its neighbors



Loop Subdivision Scheme

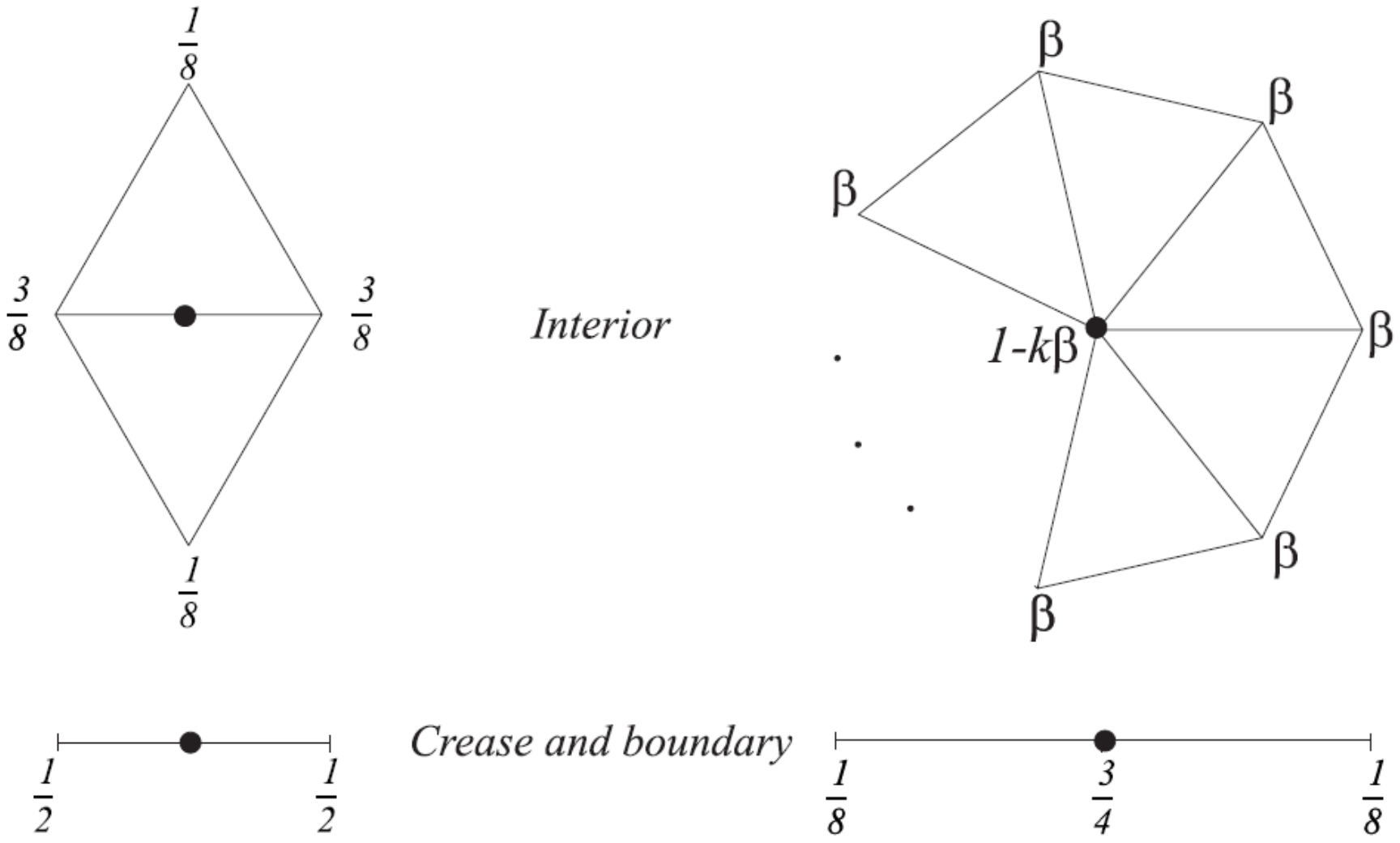
- General rule for moving existing *interior vertices*:



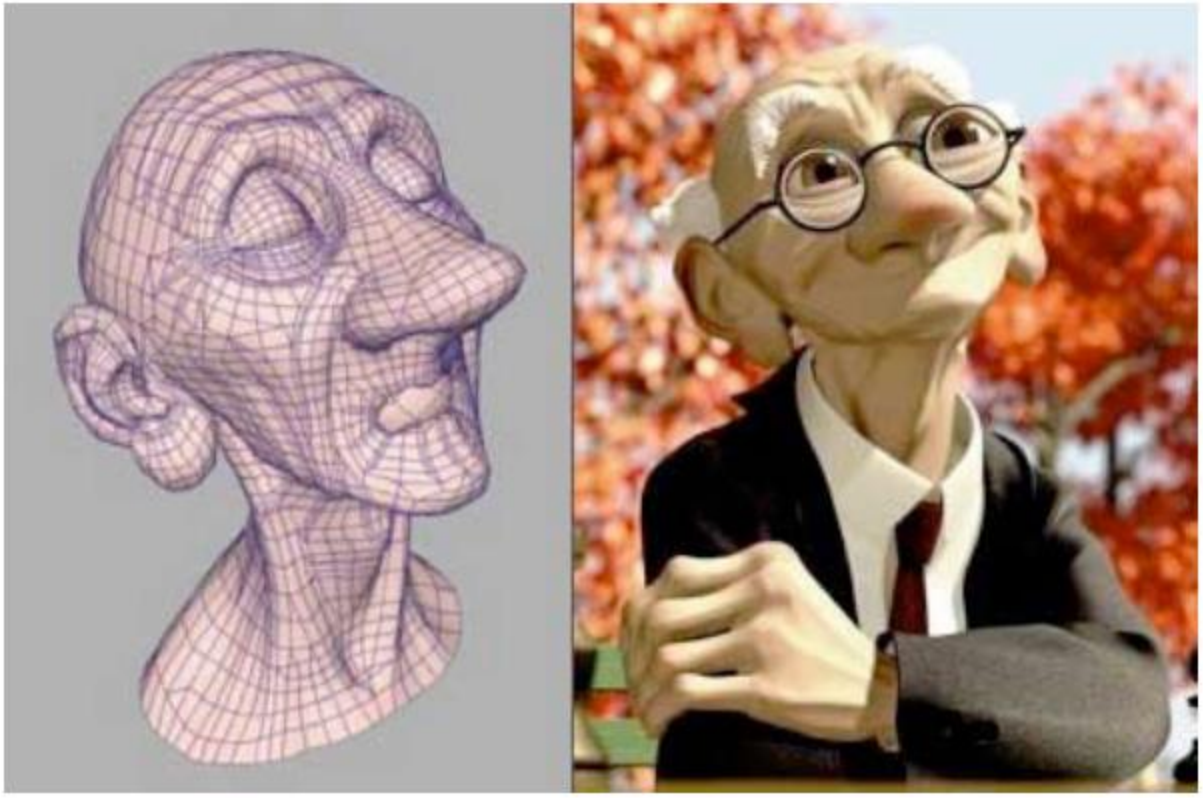
What about vertices that have more
Or less than 6 neighboring faces?

New_position = (1 - *k*β)original_position + sum(β * *each_original_vertex*)

New splitting vertices **(Old) existing vertices**

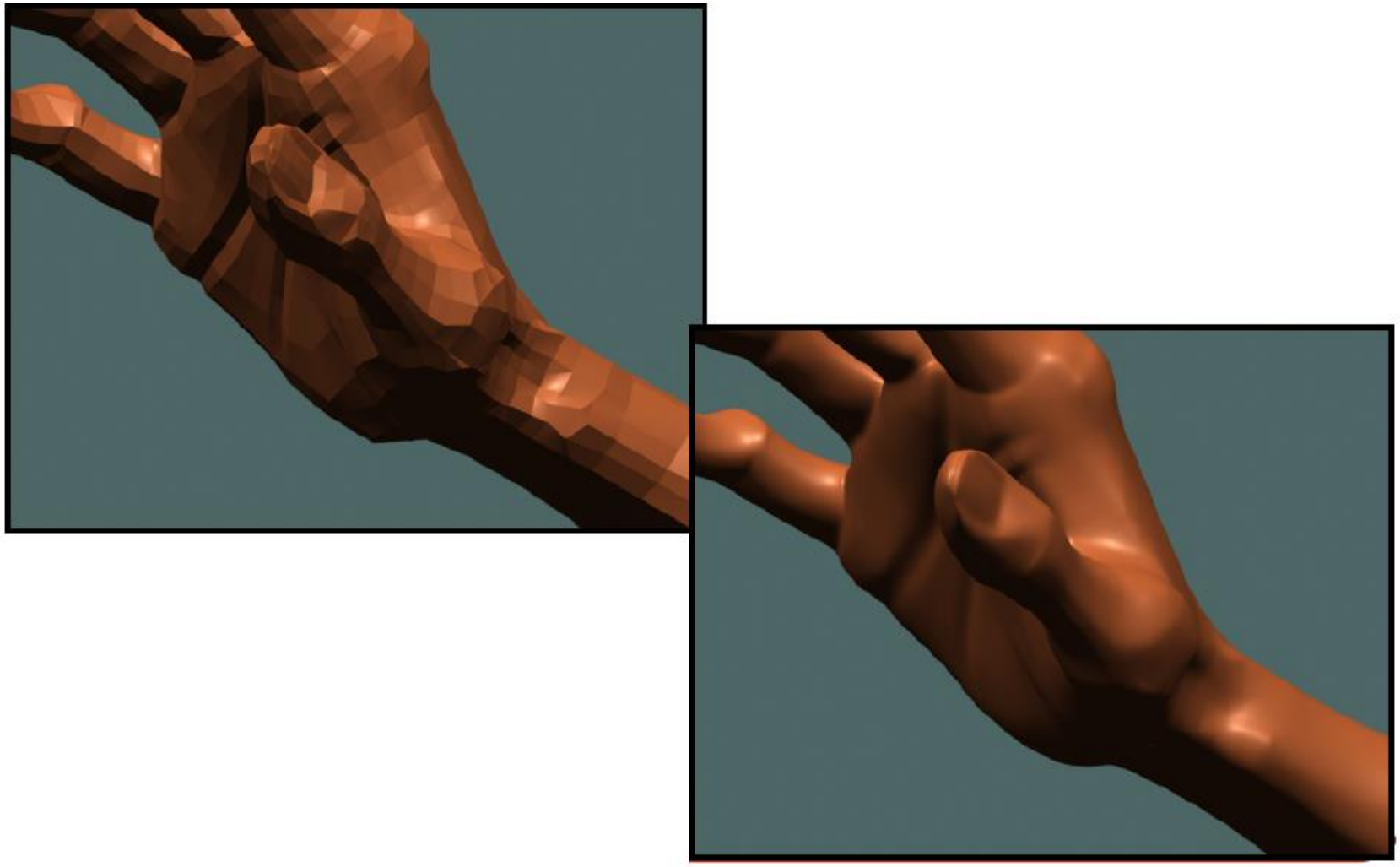


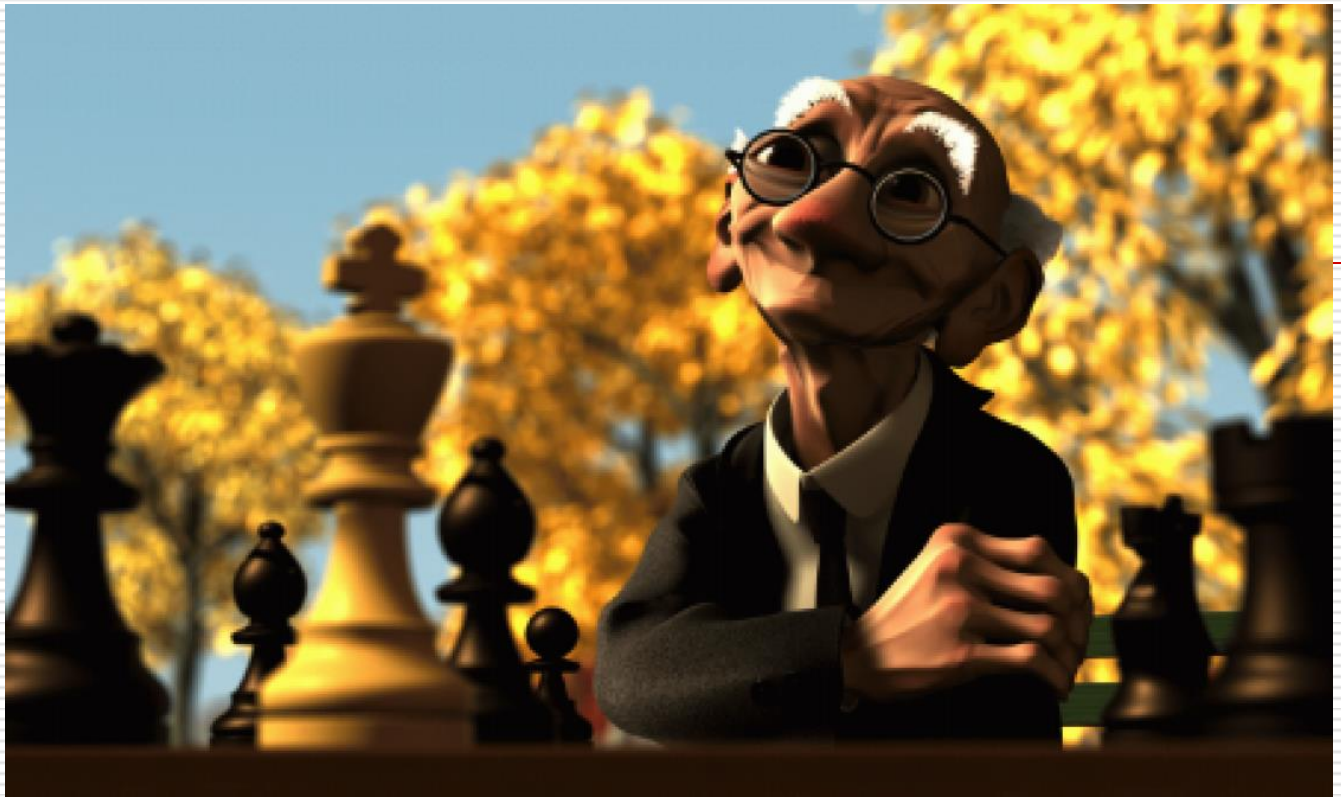
Loop Subdivision Scheme

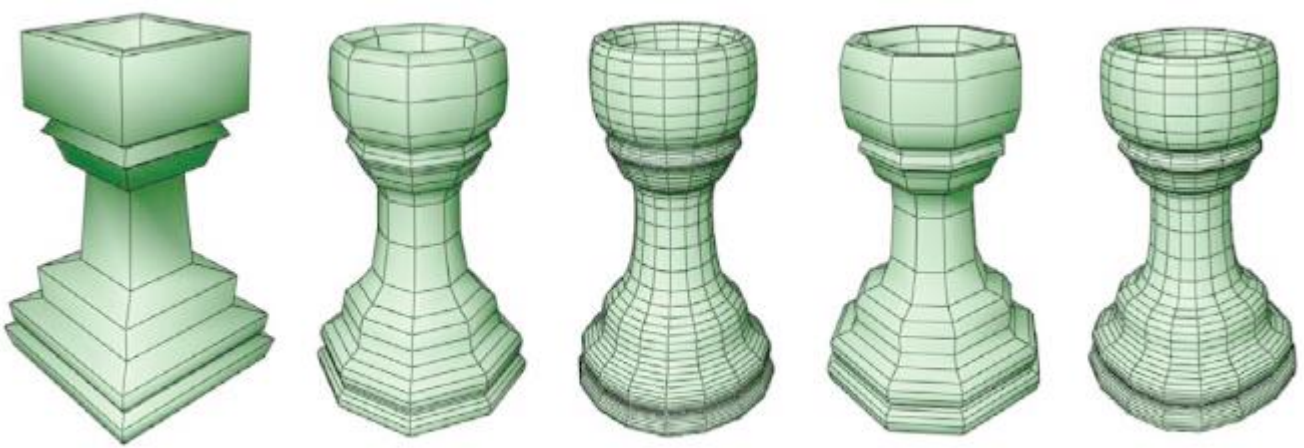


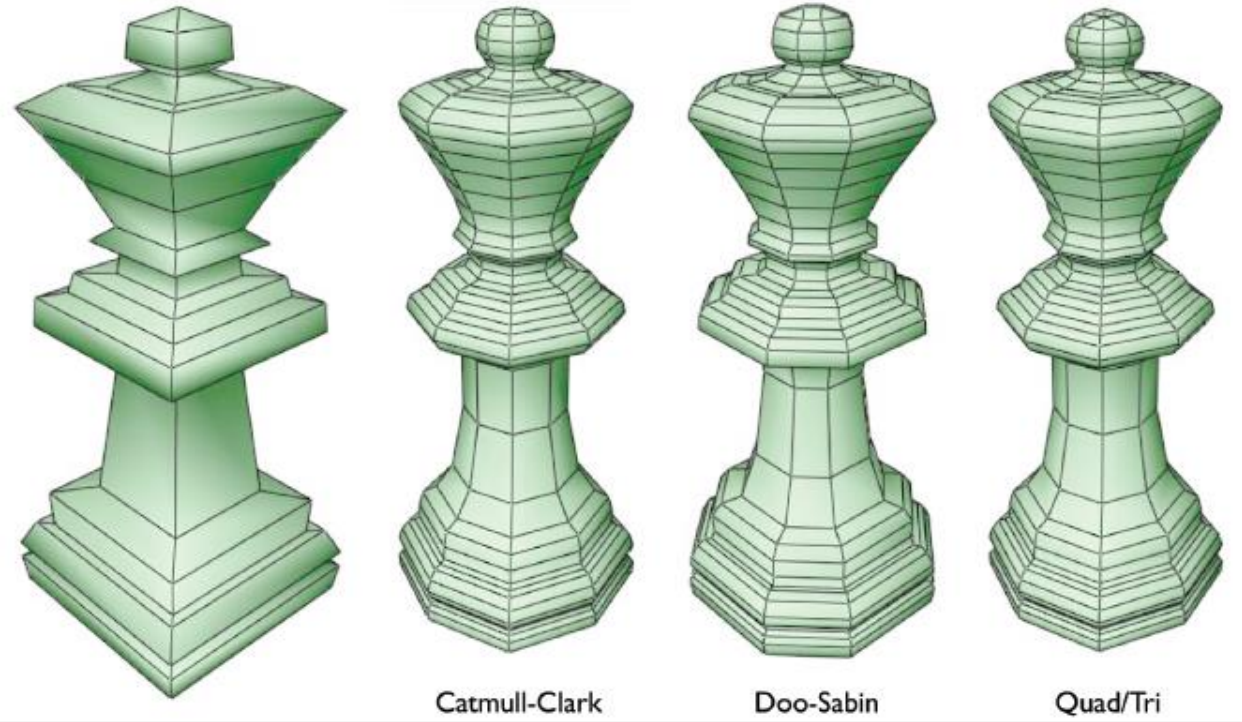
Pixar

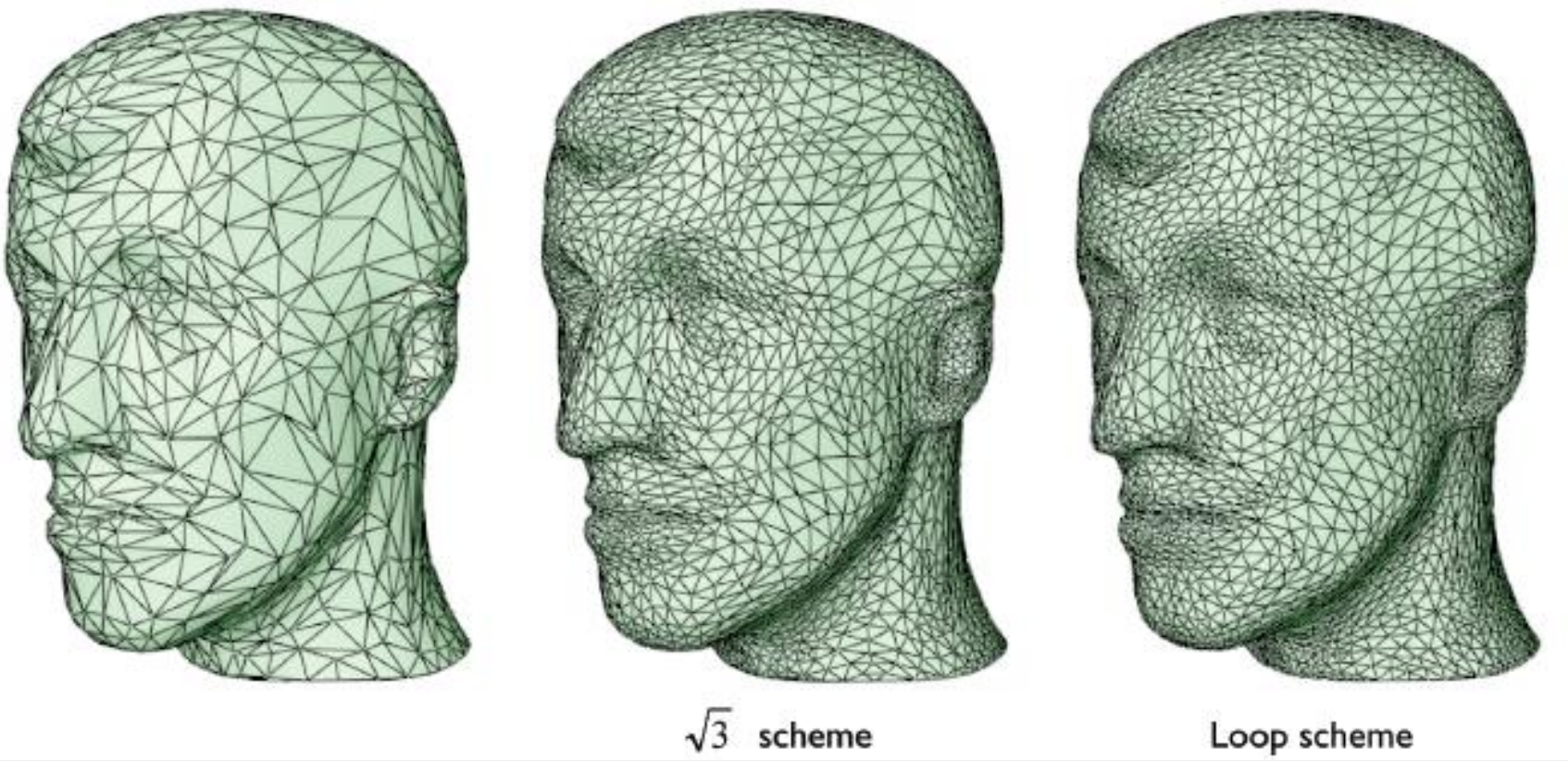
Loop Subdivision Scheme






Catmull-Clark scheme
Doo-Sabin scheme


Catmull-Clark
Doo-Sabin
Quad/Tri


$\sqrt{3}$ scheme
Loop scheme

Outline

- 3D models
- Polygonal meshes
- Mesh simplification
- Mesh subdivision/refinement

End.
Thanks