Subdivision Surfaces

Connelly Barnes

CS 4810: Graphics

Acknowledgment: slides by Jason Lawrence, Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin

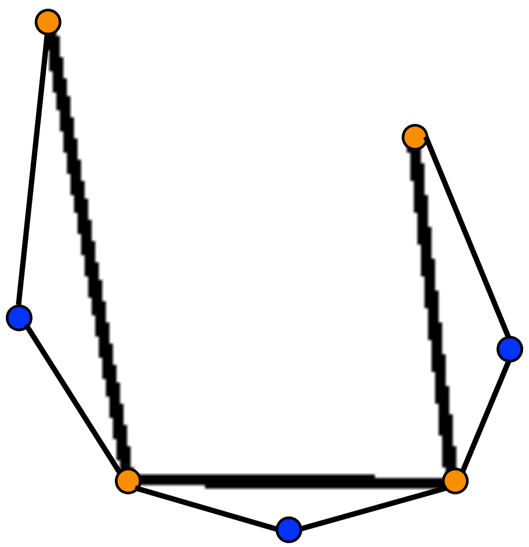
Subdivision

- 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

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision

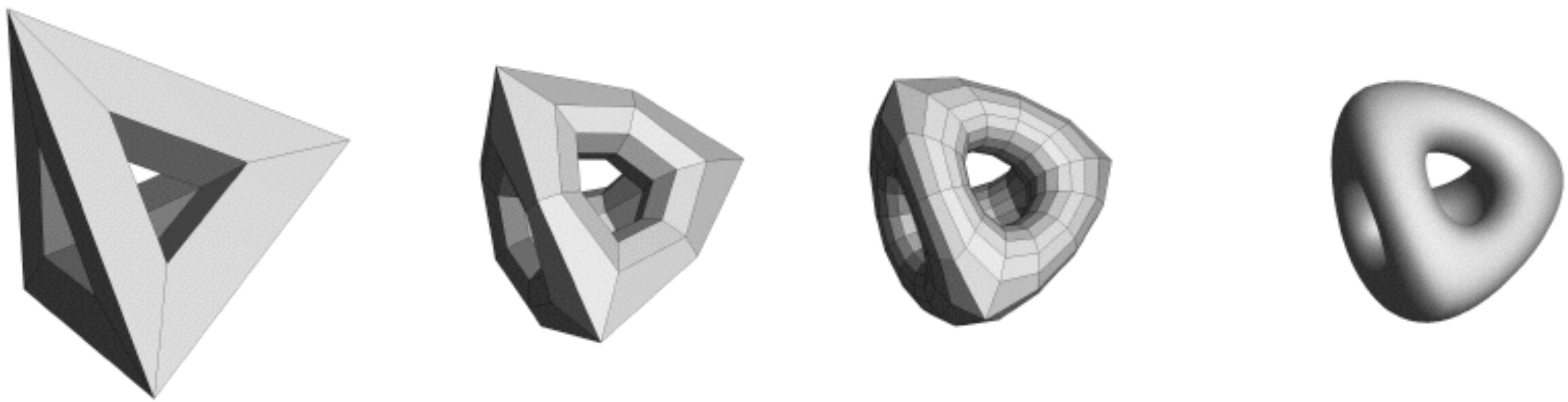
- How do you make a smooth curve?

We want to "smooth out" severe angles

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces

- Coarse mesh & subdivision rule
  - o Define smooth surface as limit of sequence of refinements

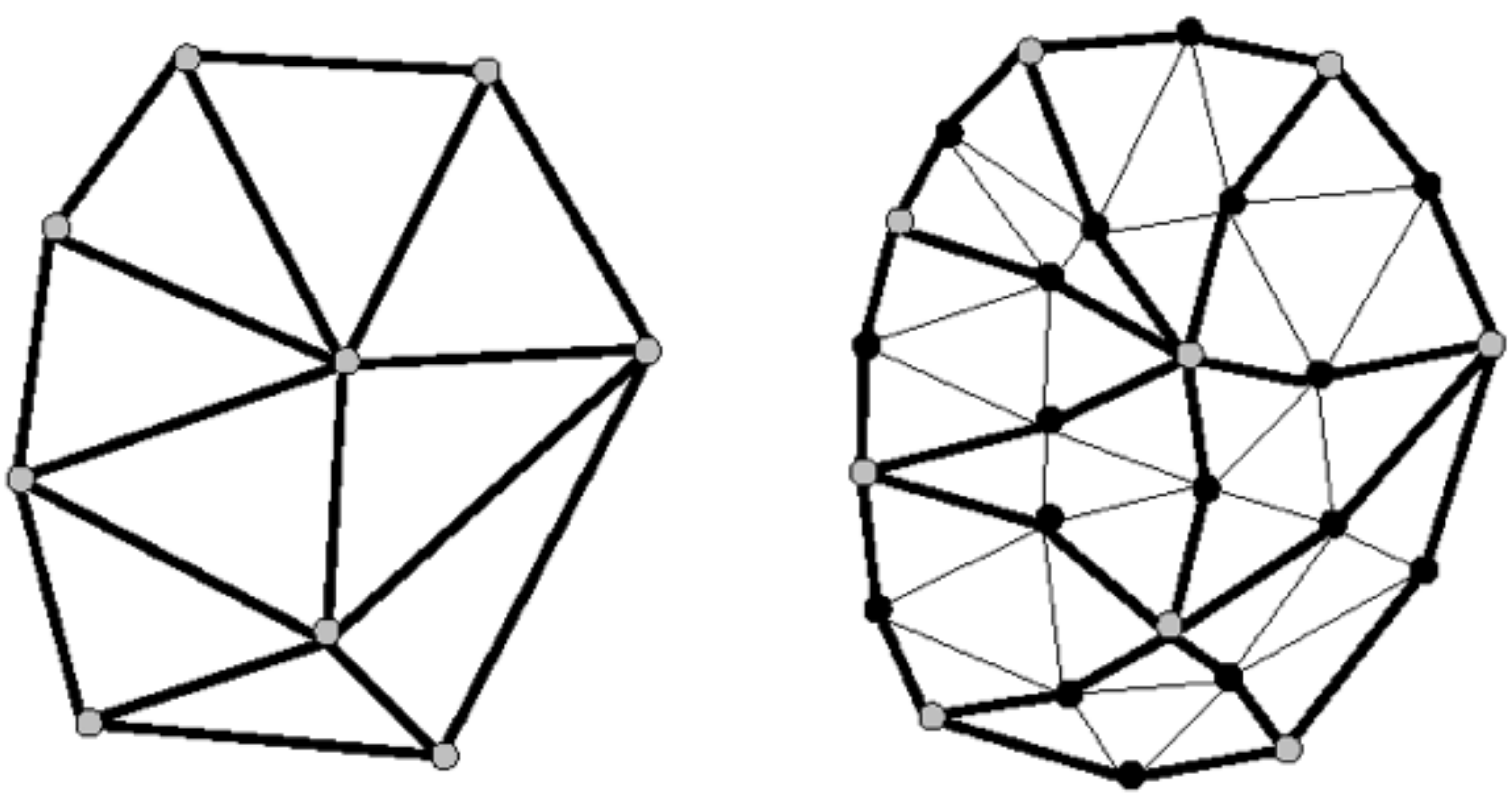


(a) (b) (c) (d)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Key Questions

- How to subdivide the mesh?
  - Aim for properties like smoothness
- How to store the mesh?
  - Aim for efficiency of implementing subdivision rules



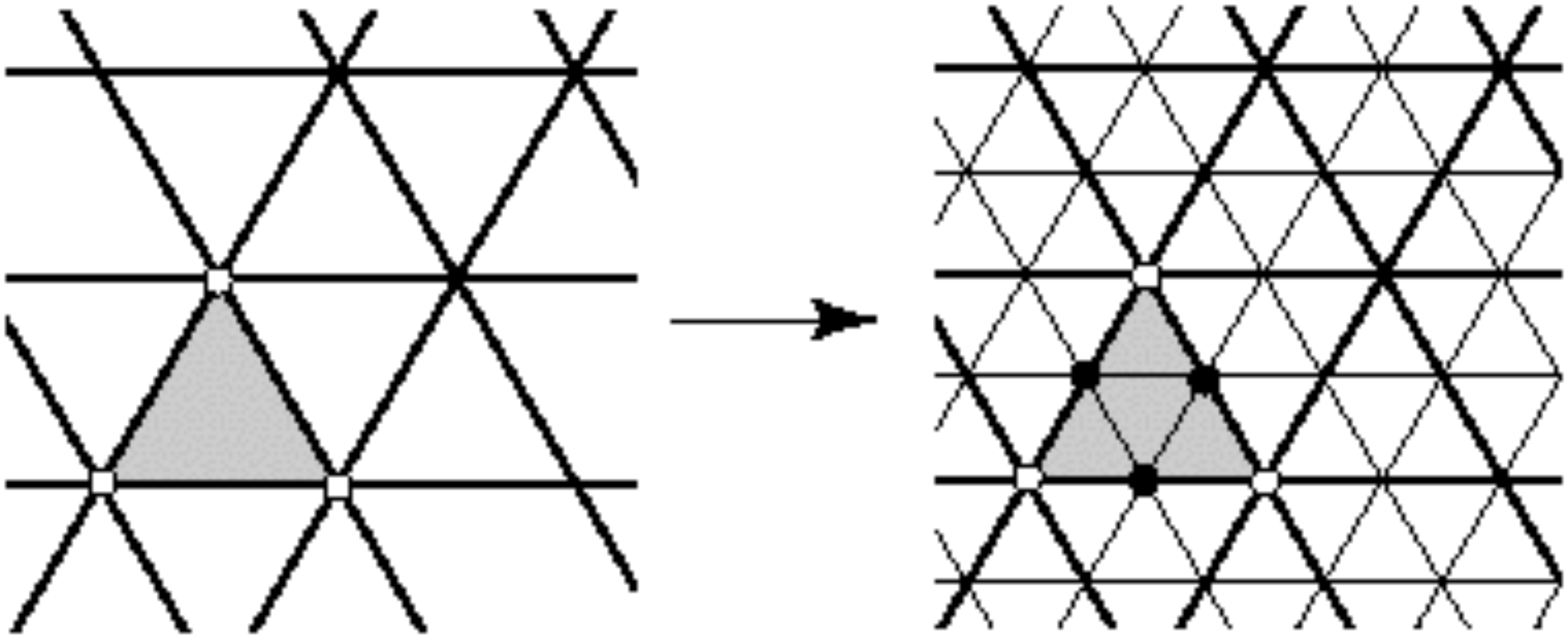
Zorin & Schroeder
SIGGRAPH 99
Course Notes

General Subdivision Scheme

- How to subdivide the mesh?
  - Two parts:
    - » Refinement:
      - – Add new vertices and connect (topological)
    - » Smoothing:
      - – Move vertex positions (geometric)

Loop Subdivision Scheme

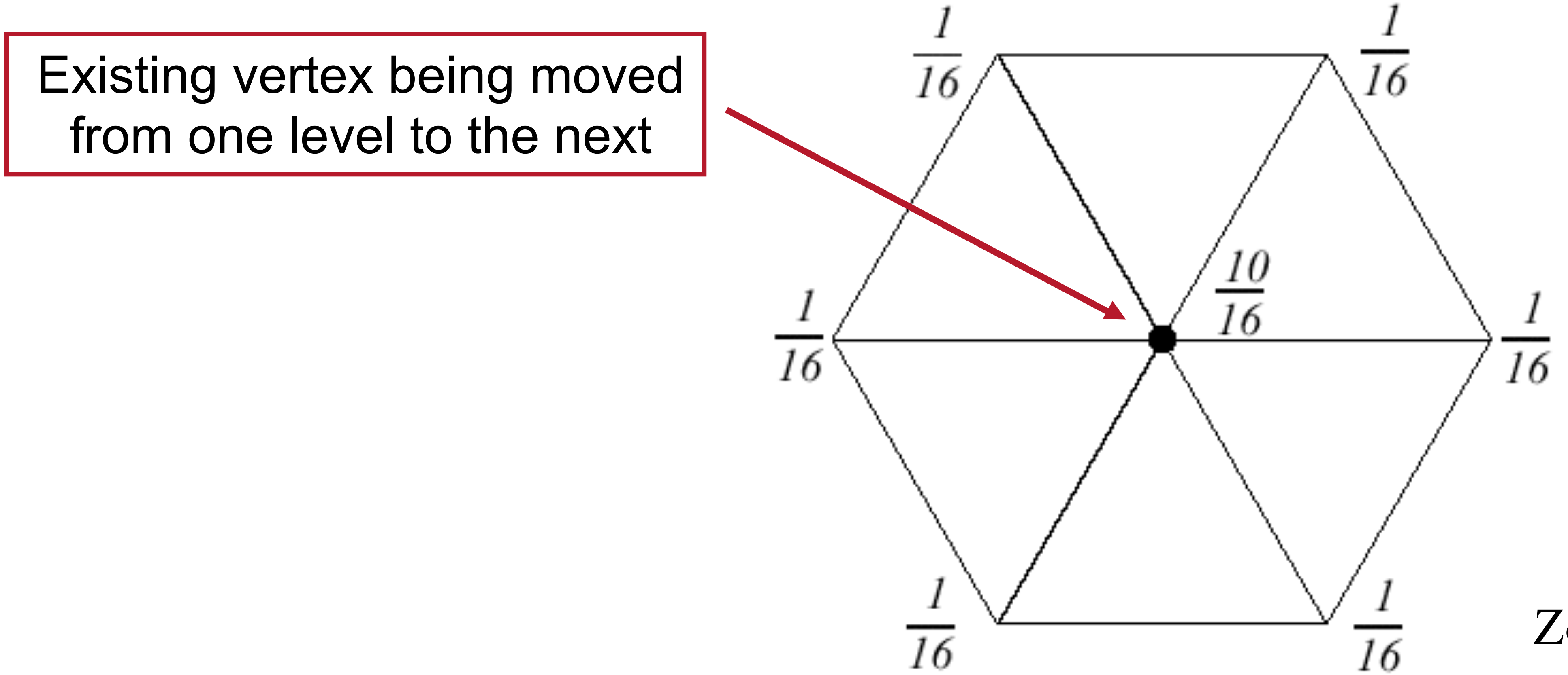
- How to subdivide the mesh?
  - Refinement:
    - » Subdivide each triangle into 4 triangles by splitting each edge and connecting new vertices



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

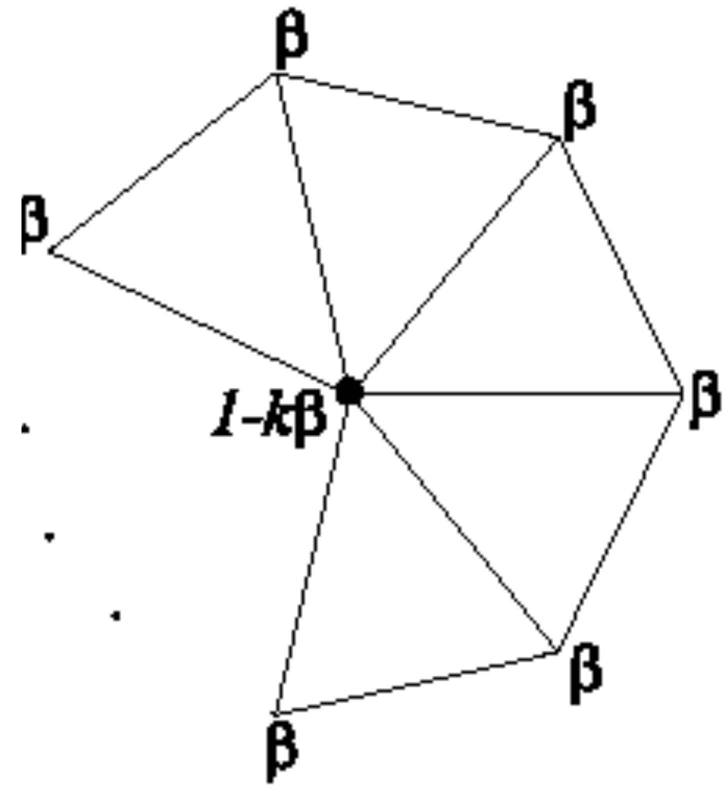
- How to subdivide the mesh:
  - Refinement
  - Smoothing:
    - » Existing Vertices: Choose *new* location as weighted average of *original* vertex and its neighbors



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

- General rule for moving existing *interior vertices*:



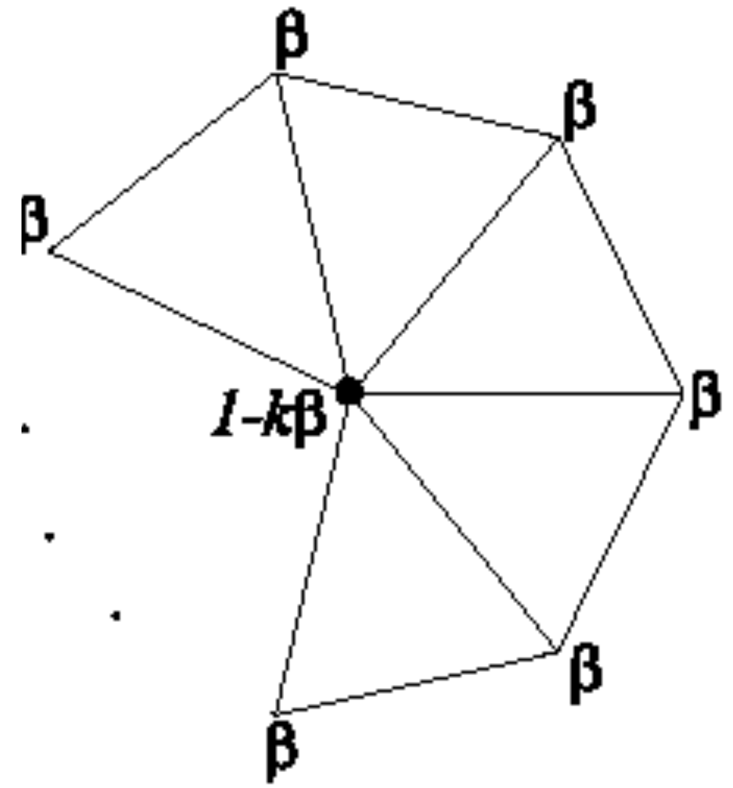
What about vertices that have more or less than 6 neighboring faces?

*new_position = (1-kβ)original_position + sum(β*each_original_vertex)*

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

- General rule for moving existing *interior vertices*:



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

*new_*

<u>0≤ β ≤1/k</u>:

• As β increases, the contribution from adjacent vertices plays a more important role.

hroeder
SIGGRAPH 99
Course Notes

Where do existing vertices move?

- How to choose β?
  - Analyze properties of limit surface
  - Interested in continuity of surface and smoothness
  - Involves calculating eigenvalues of matrices
    - Original Loop

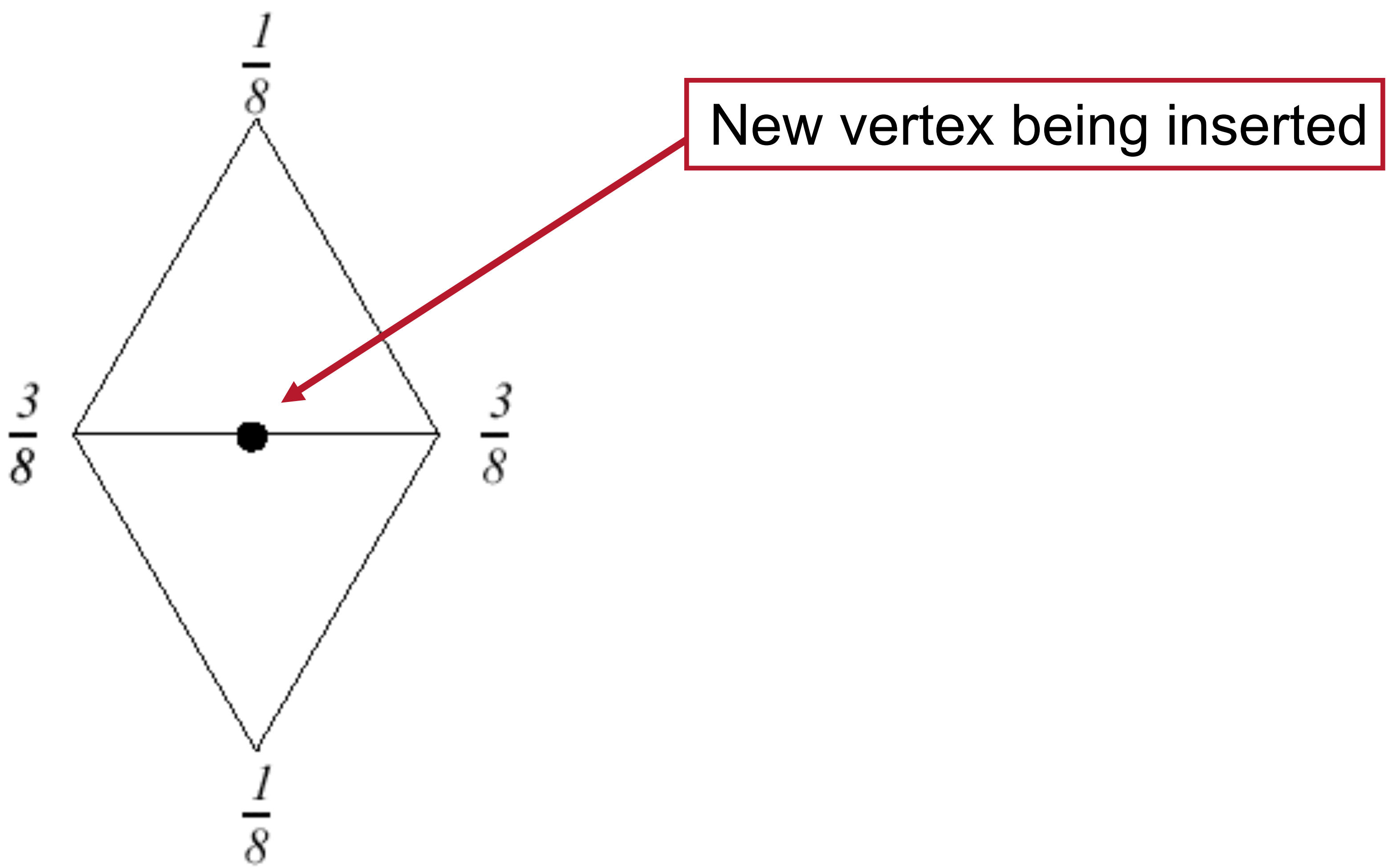
$$\beta = \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k}\right)^2\right)$$

    - Warren

$$\beta = \begin{cases} \frac{3}{8k} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

Loop Subdivision Scheme

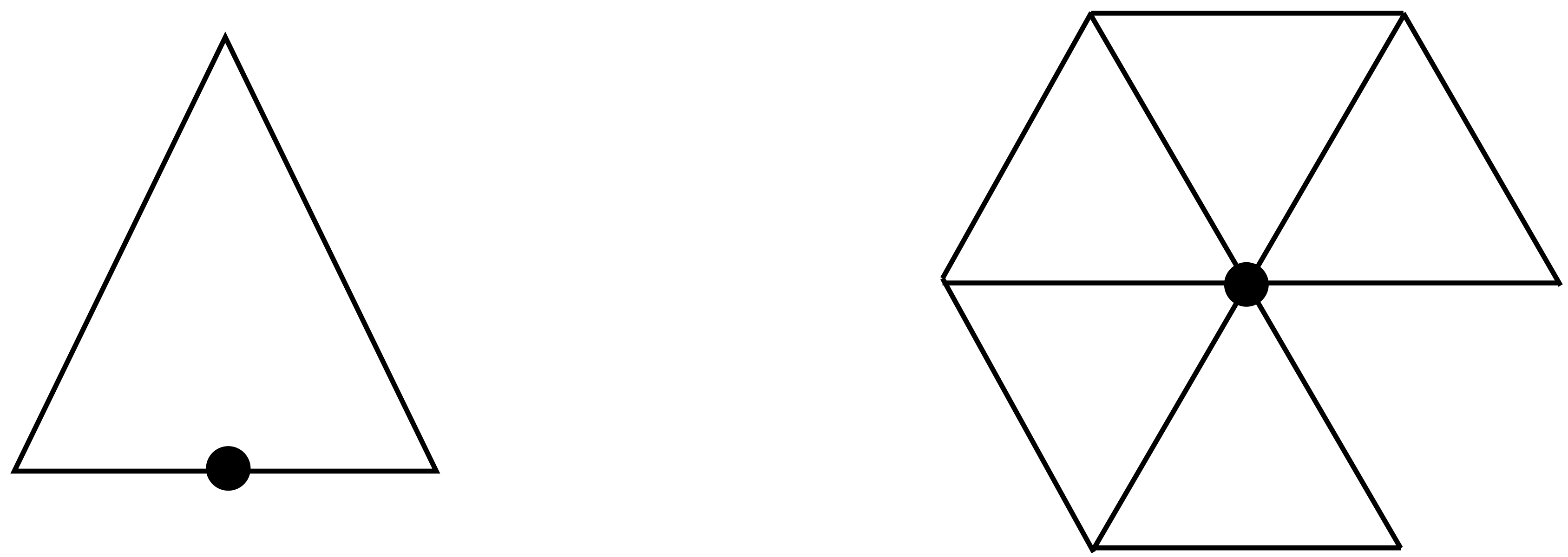
- How to subdivide the mesh:
  - Refinement
  - Smoothing:
    - » Inserted Vertices: Choose location as weighted average of *original* vertices in local neighborhood



Zorin & Schroeder
SIGGRAPH 99
Course Notes

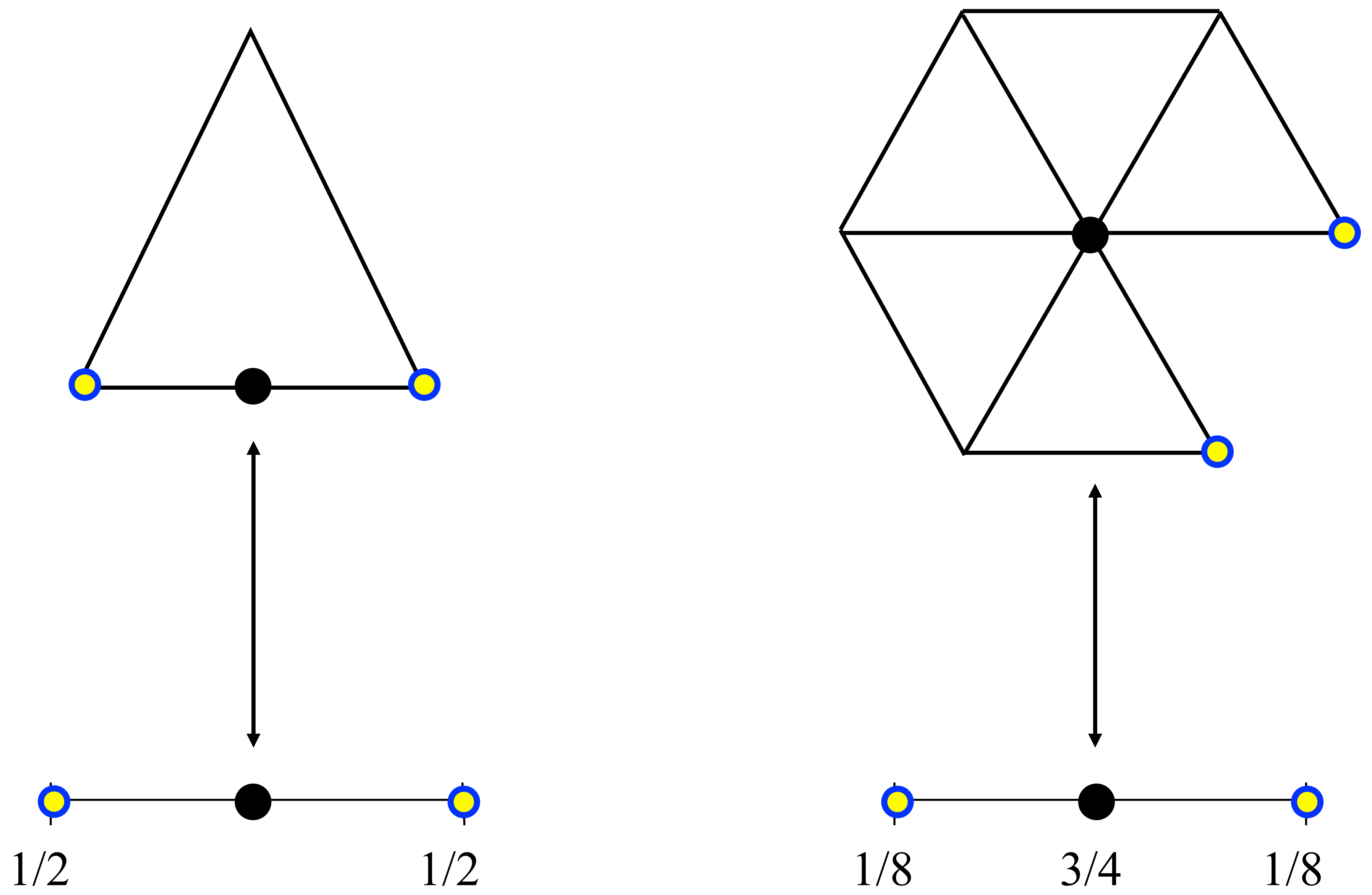
Boundary Cases?

- What about *extraordinary vertices* and *boundary edges?*:
  - **o**Existing vertex adjacent to a missing triangle
  - **o**New vertex bordered by only one triangle



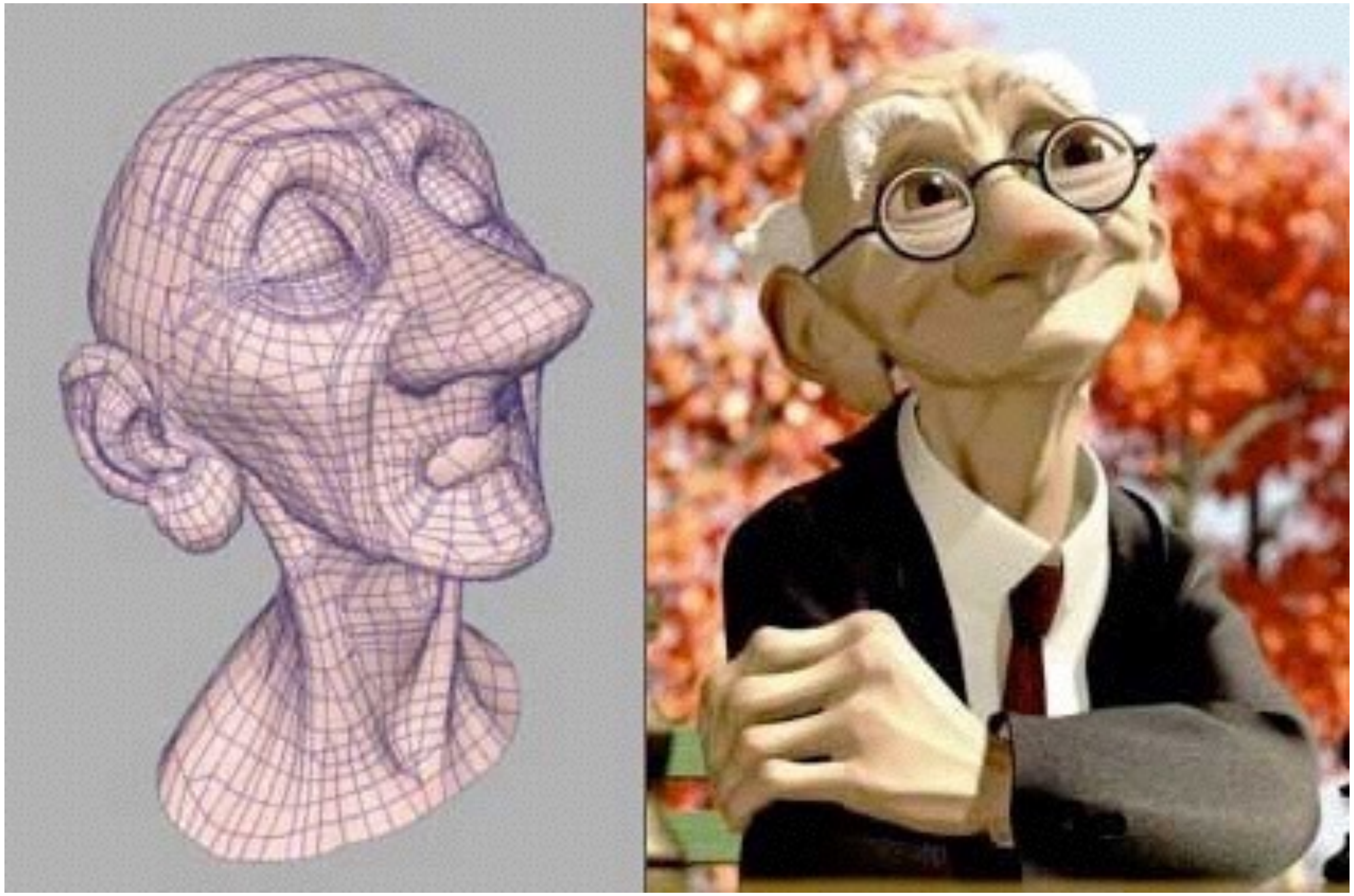
Boundary Cases?

- Rules for *extraordinary vertices* and *boundaries*:



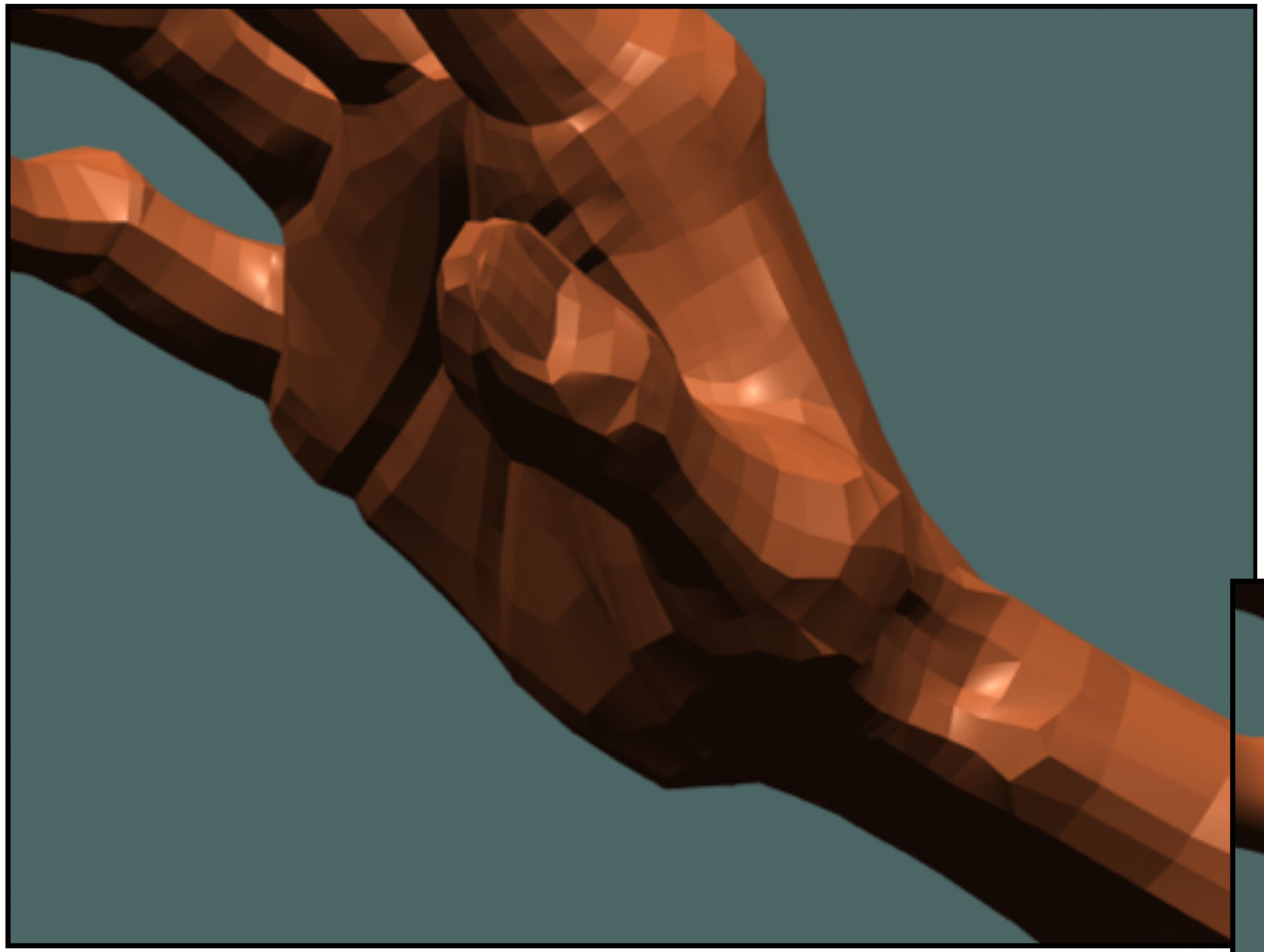
Zorin & Schroeder
SIGGRAPH 99
Course Notes

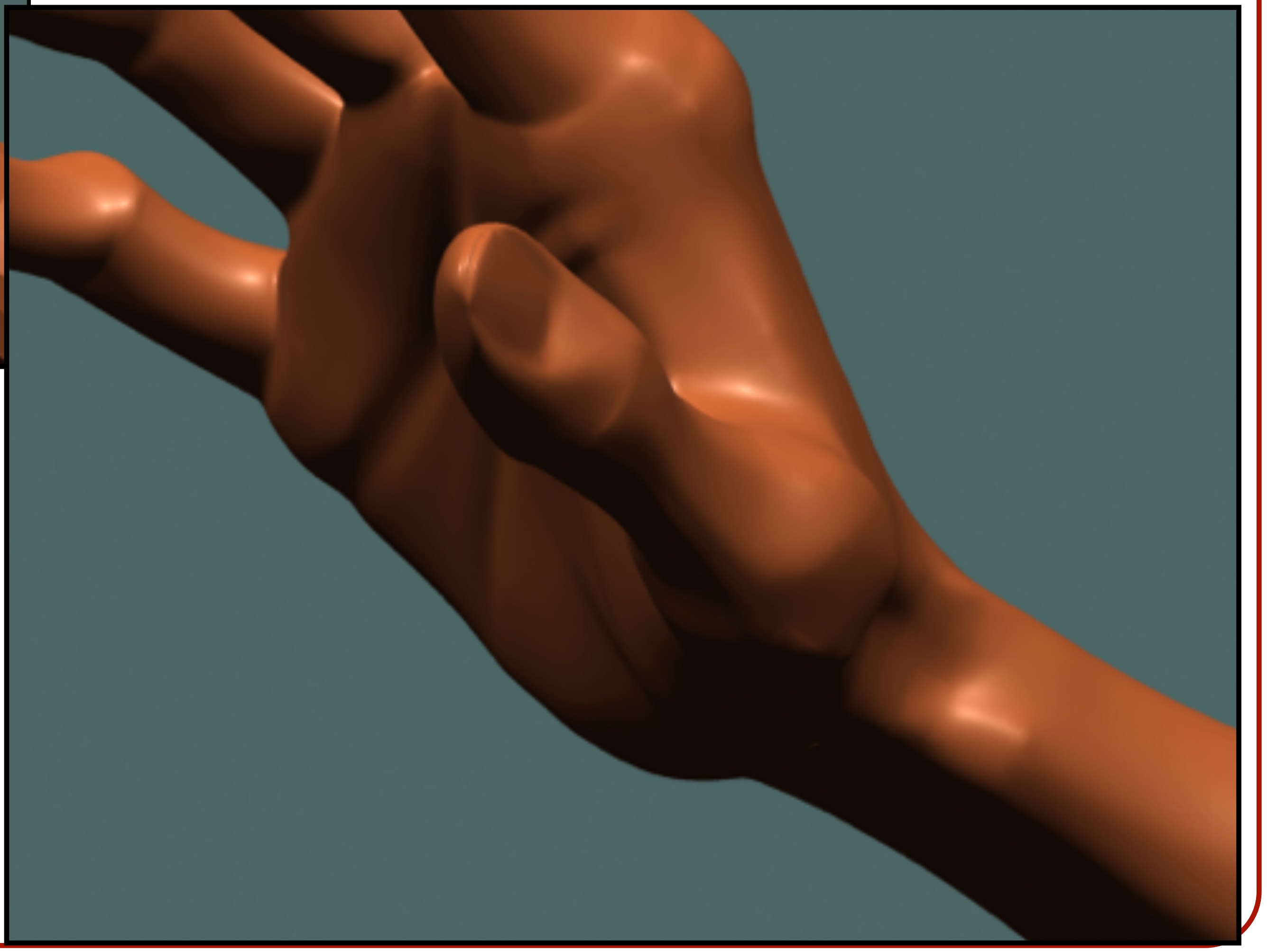
Loop Subdivision Scheme



Pixar

Loop Subdivision Scheme





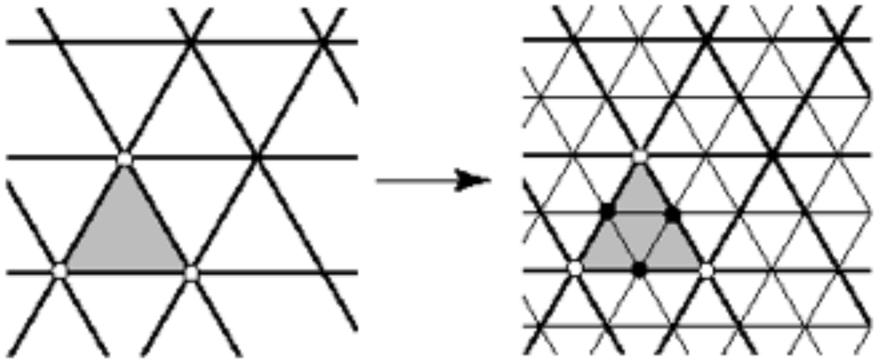
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

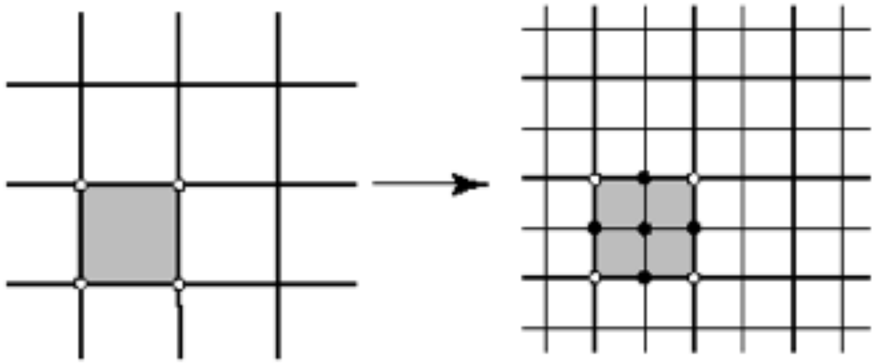
Geri's Game, *Pixar*

Subdivision Schemes

- There are different subdivision schemes
  - Different methods for refining topology
  - Different rules for positioning vertices
    - Interpolating versus approximating



Face split for triangles



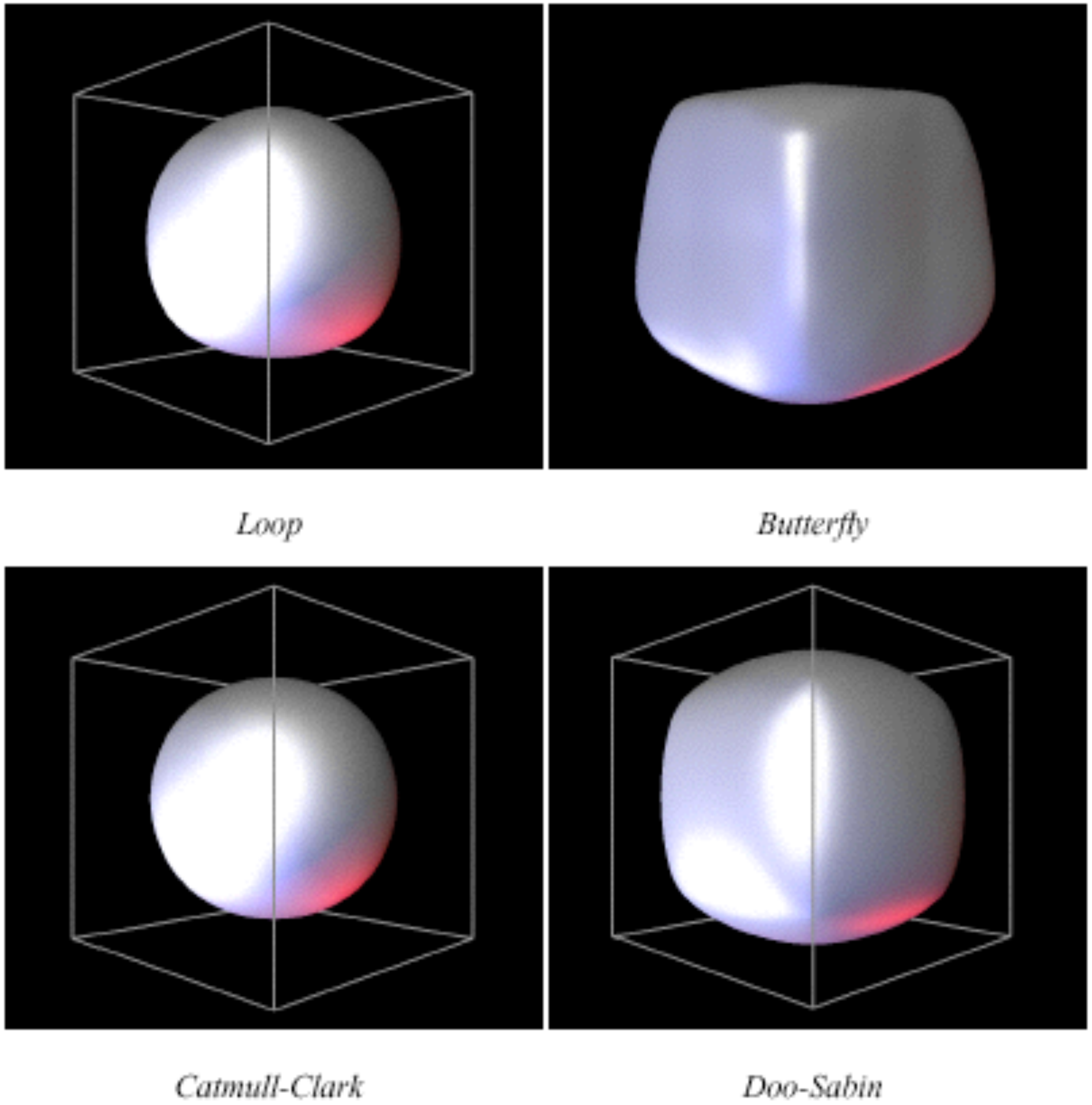
Face split for quads

Face split		
	Triangular meshes	*Quad. meshes*
Approximating	Loop (C^2)	Catmull-Clark (C^2)
Interpolating	Mod. Butterfly (C^1)	Kobbelt (C^1)

Vertex split
Doo-Sabin, Midedge (C^1)
Biquartic (C^2)

Zorin & Schroeder, SIGGRAPH 99 , Course Notes

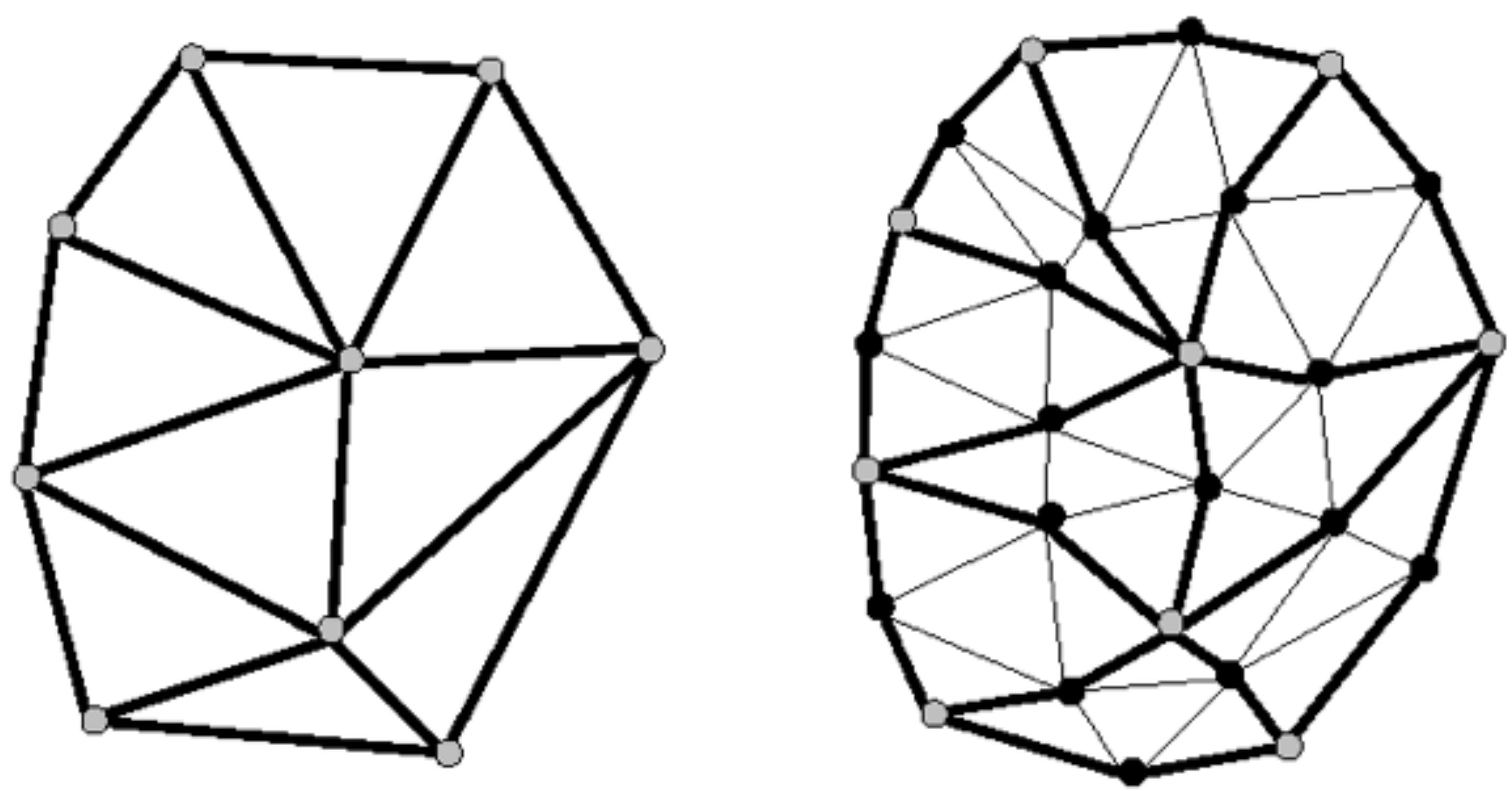
Subdivision Schemes



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Key Questions

- How to refine the mesh?
  - **Aim for properties like smoothness**
- **How to store the mesh?**
  - Aim for efficiency for implementing subdivision rules

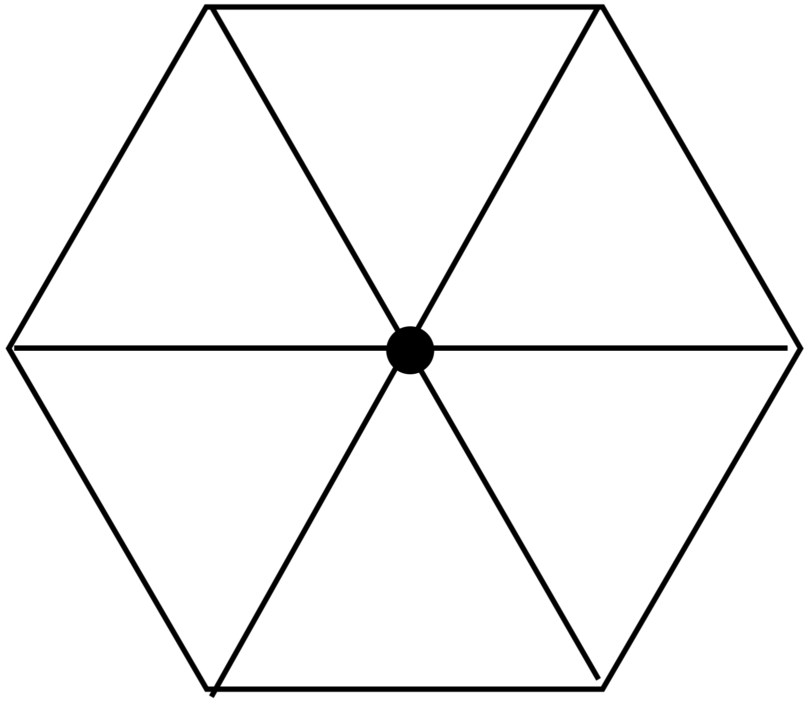


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Smoothness

To determine the smoothness of the subdivision:

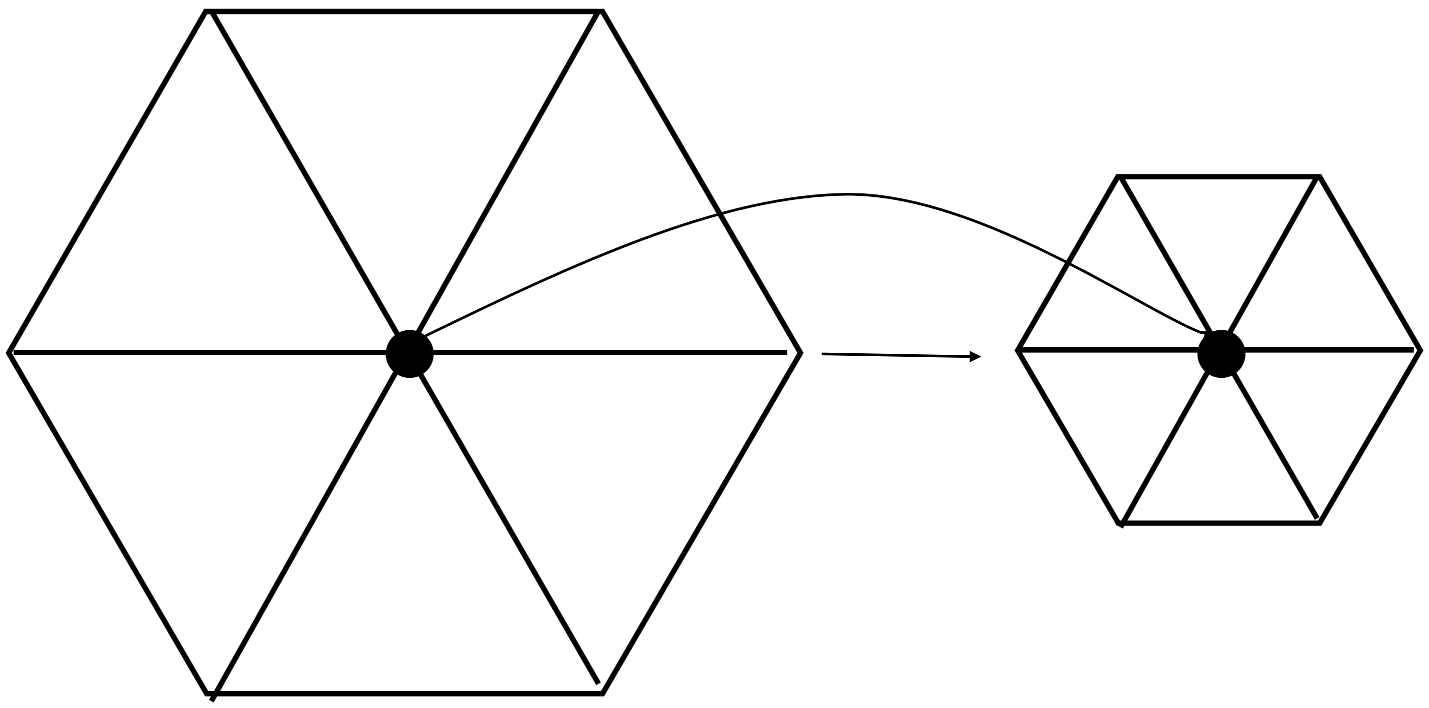
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

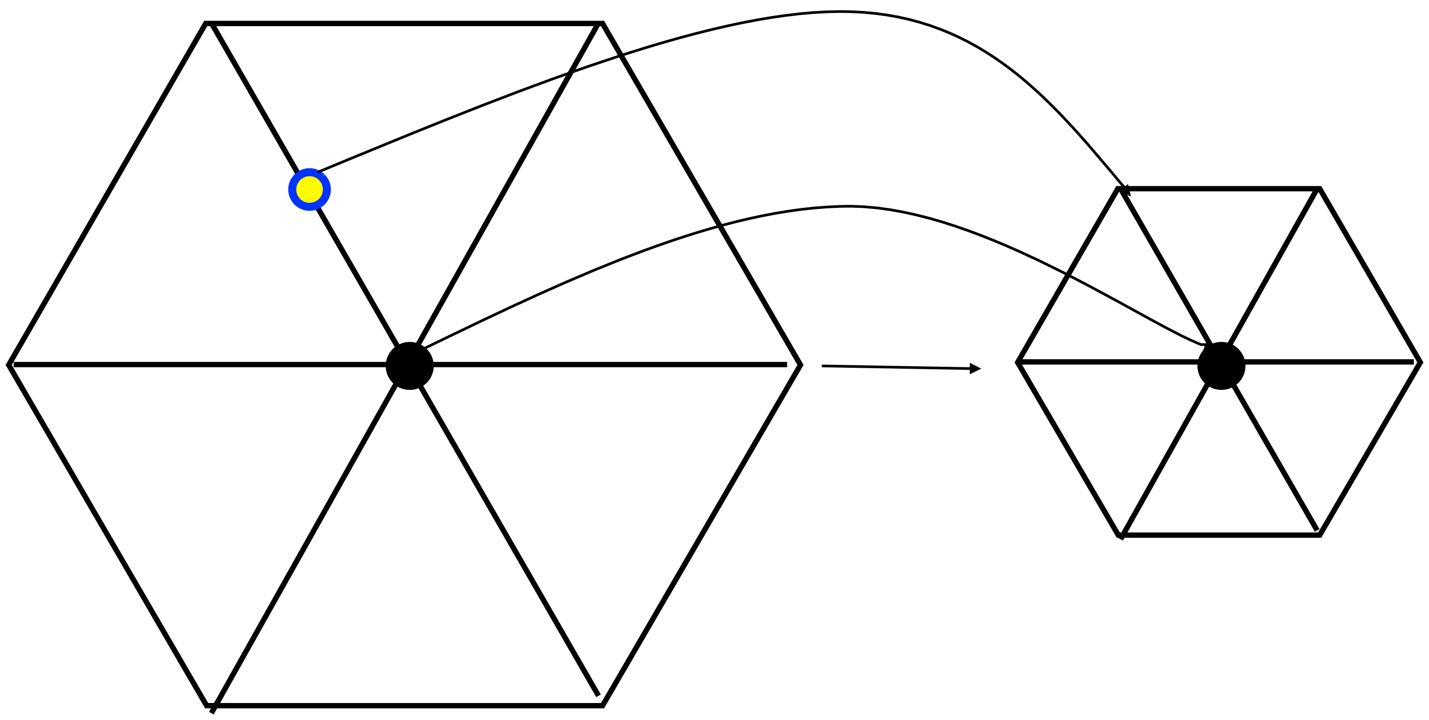
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

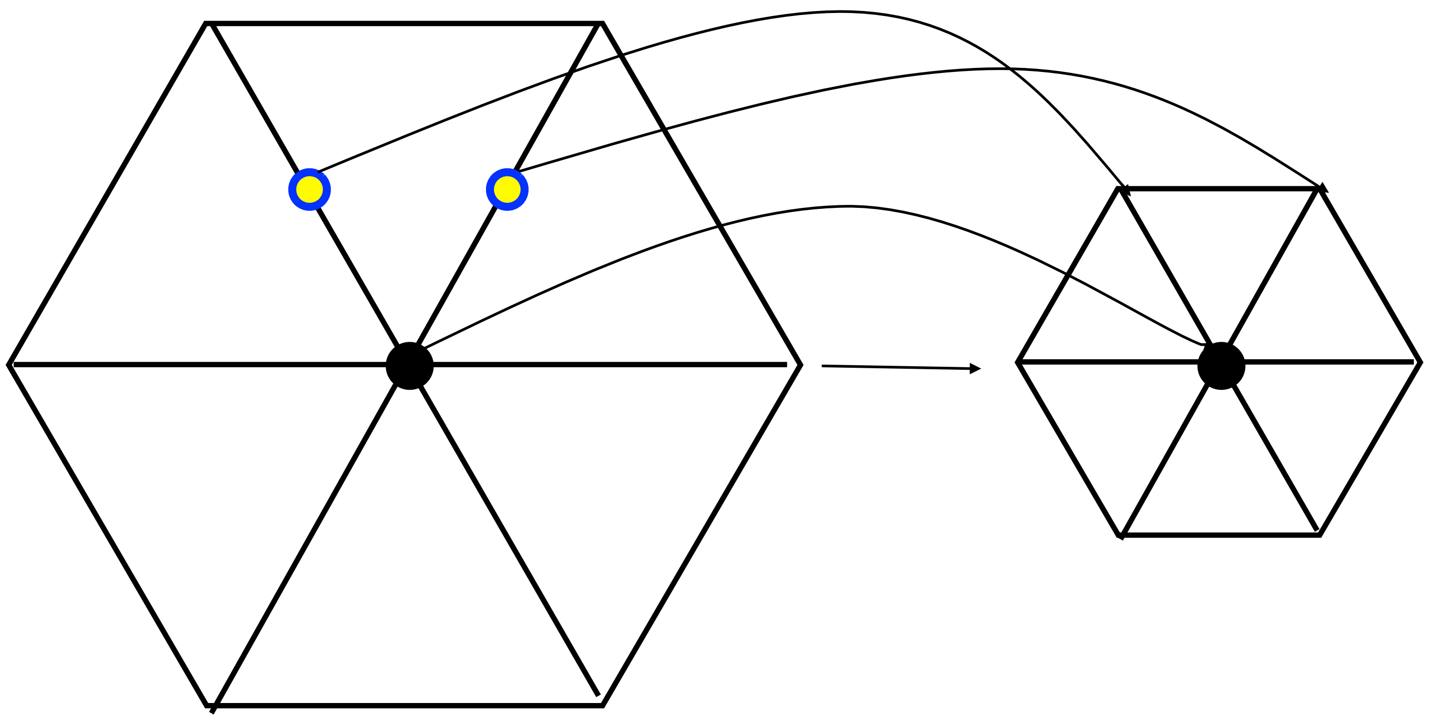
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

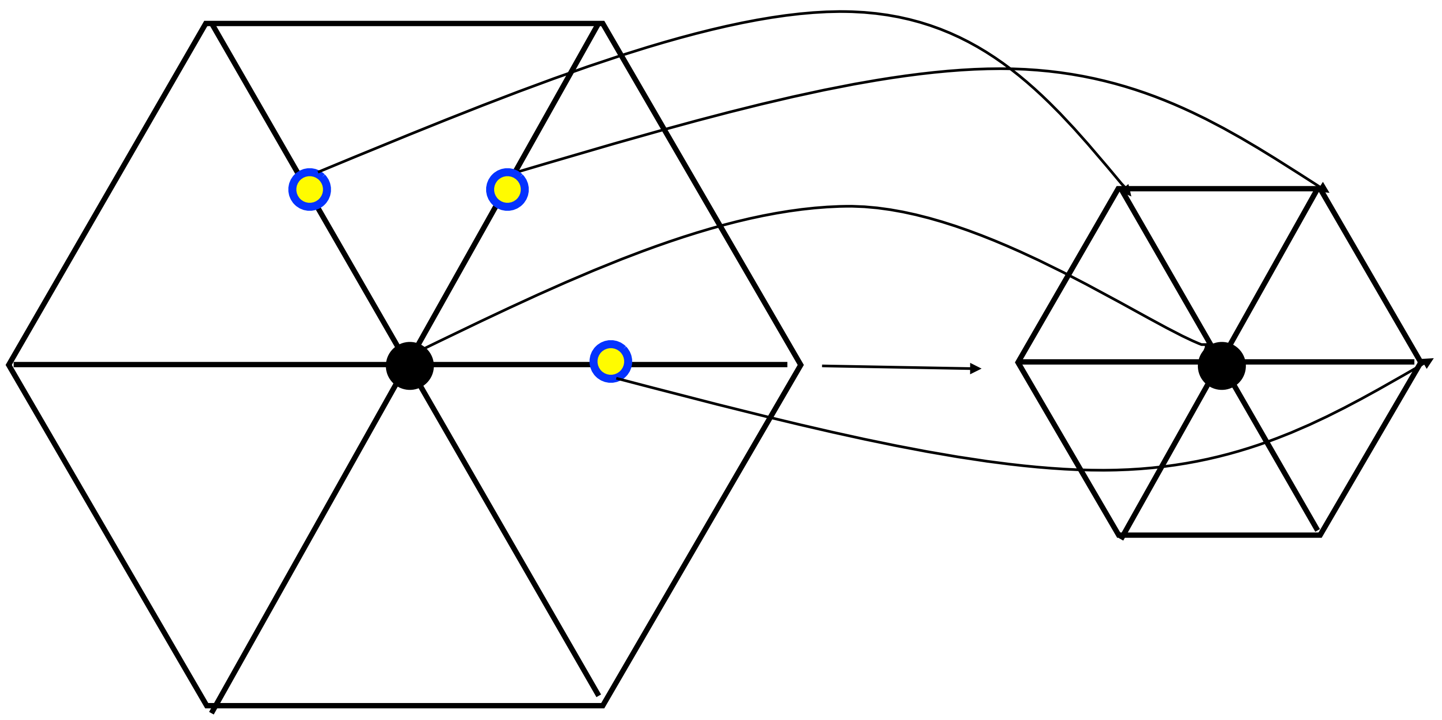
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

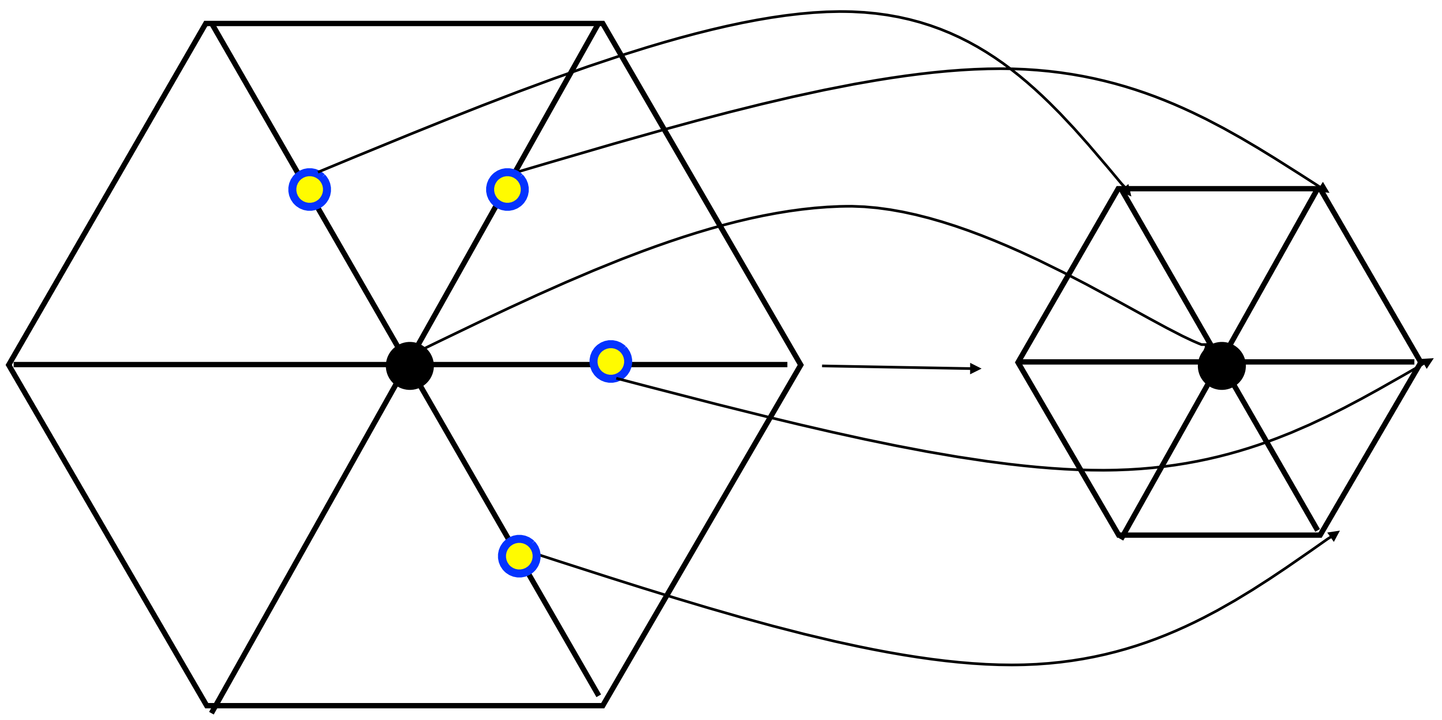
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

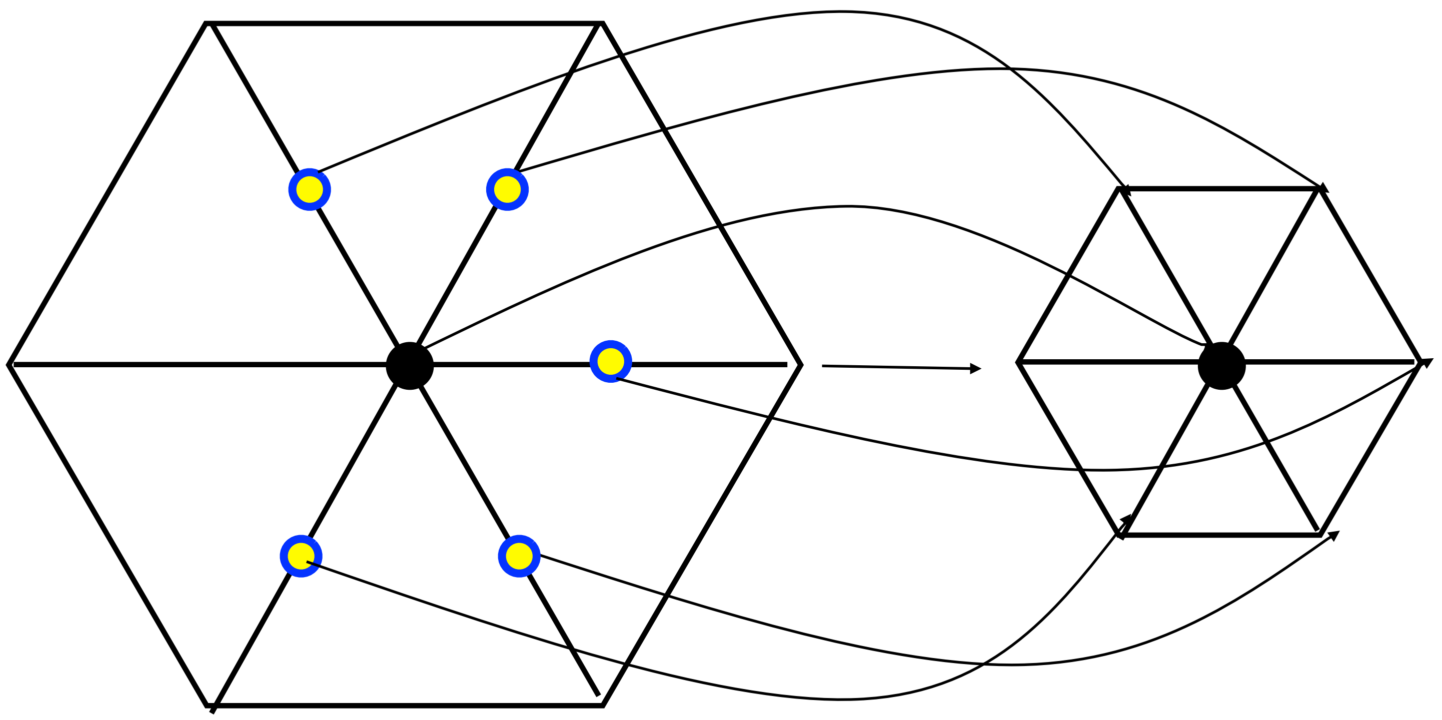
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

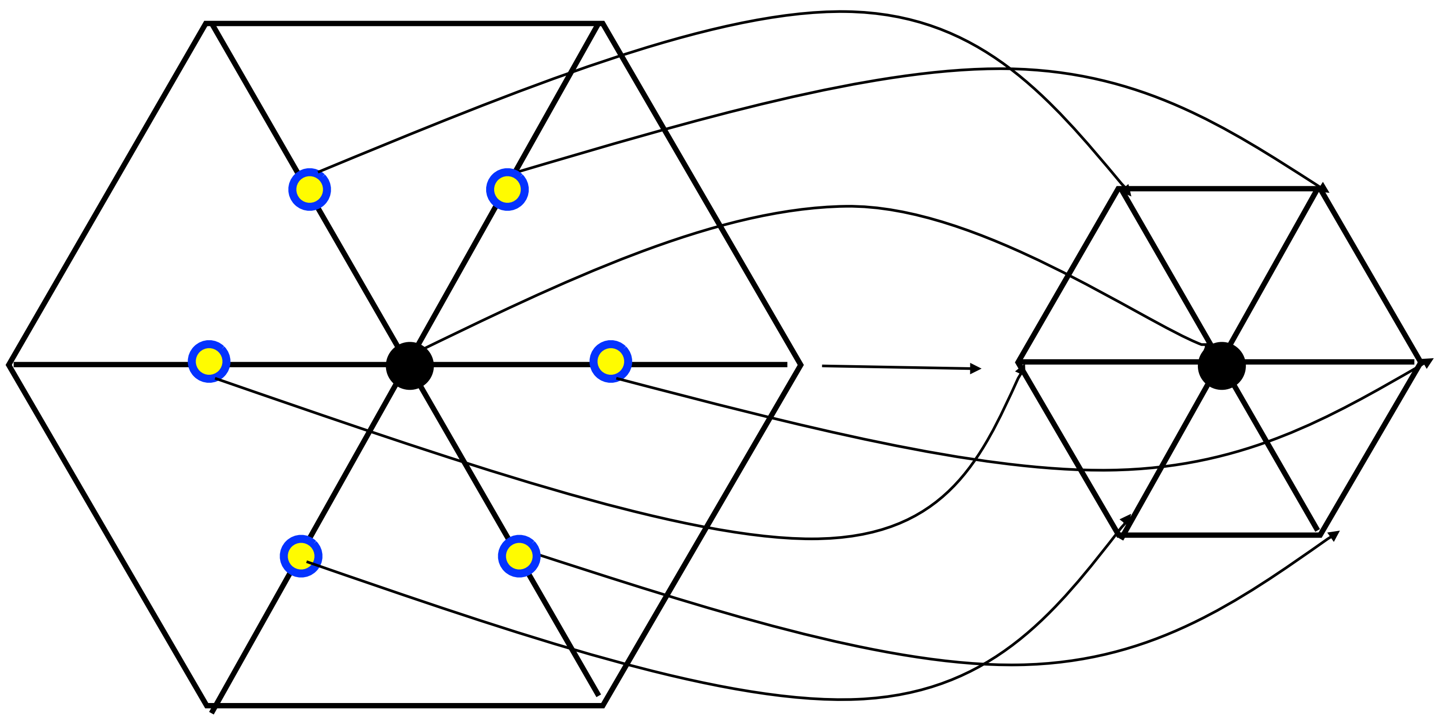
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

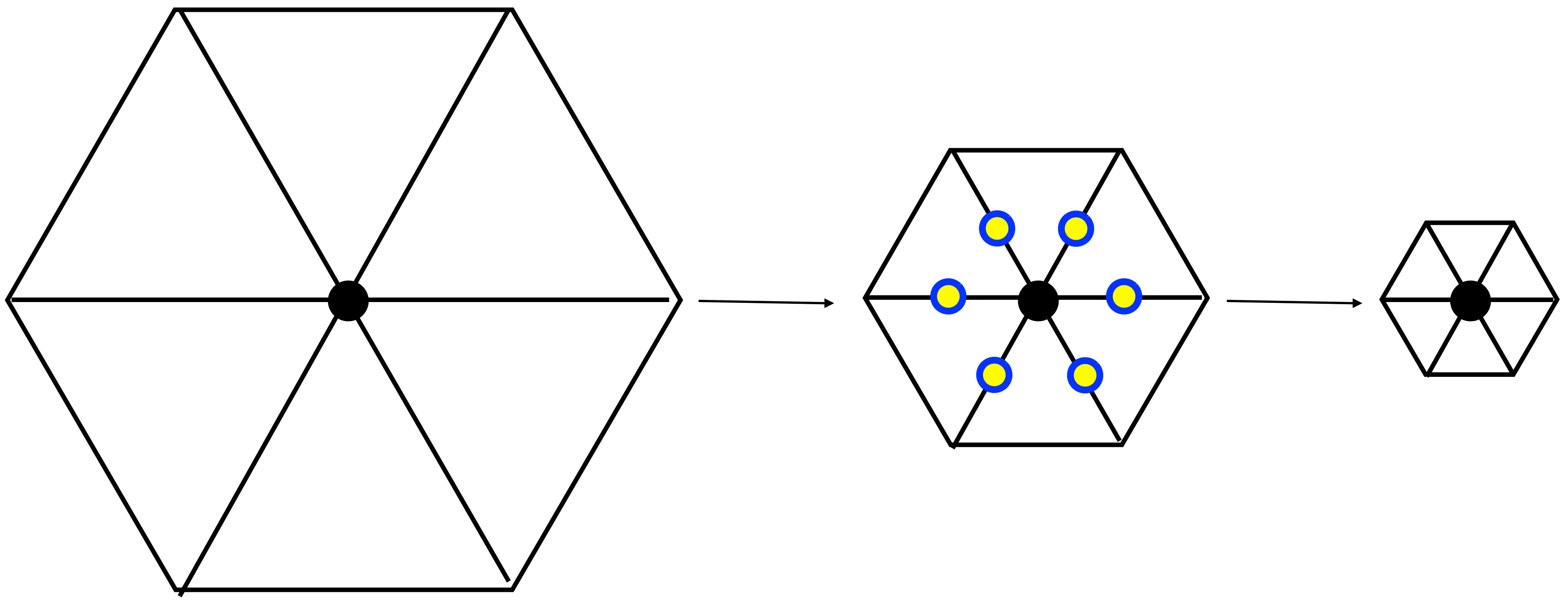
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

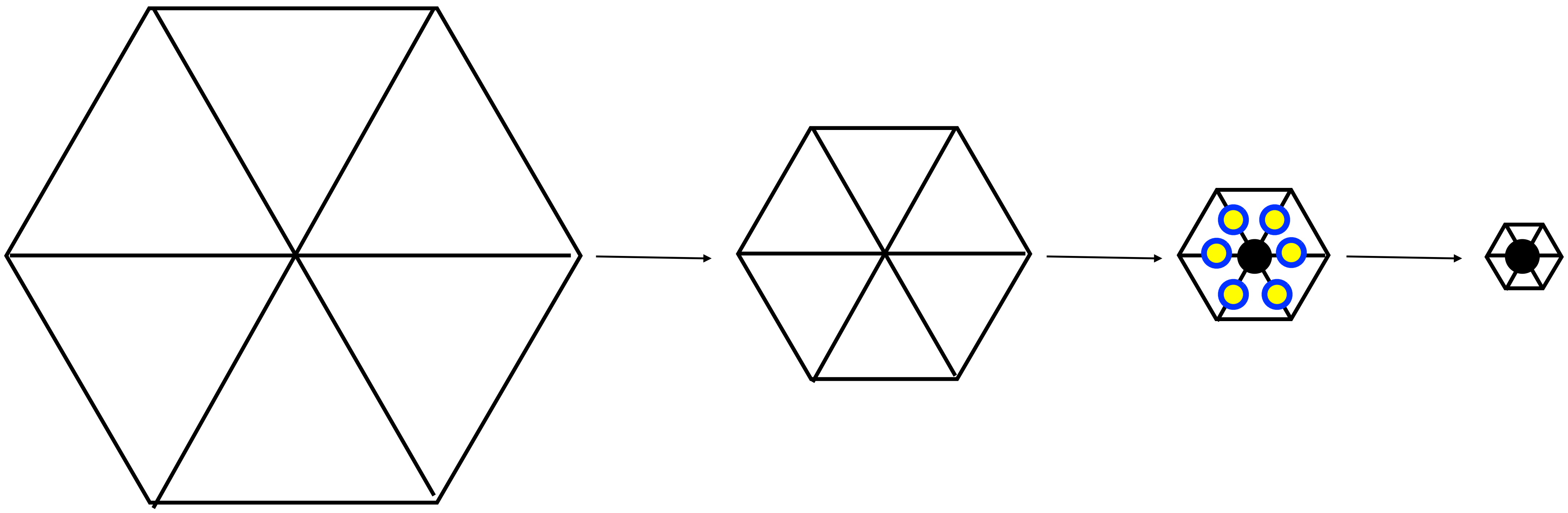
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.



Subdivision Smoothness

To determine the smoothness of the subdivision:

- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.

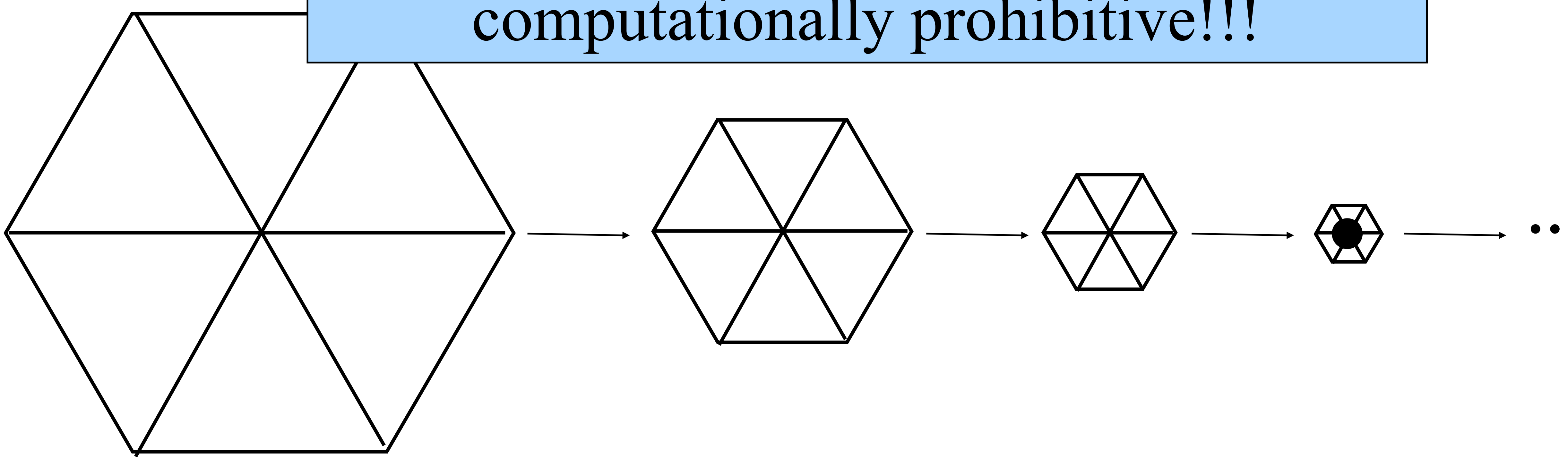


Subdivision Smoothness

To determine the smoothness of the subdivision:

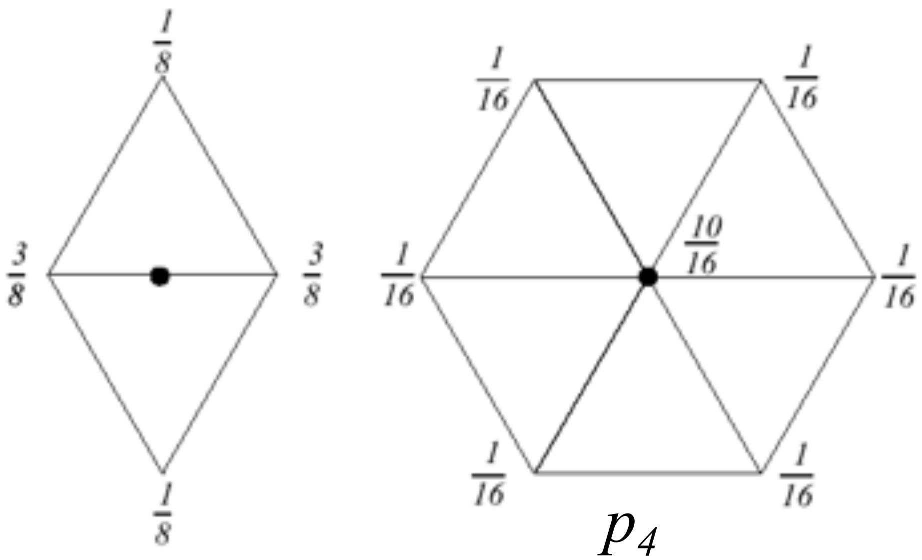
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.

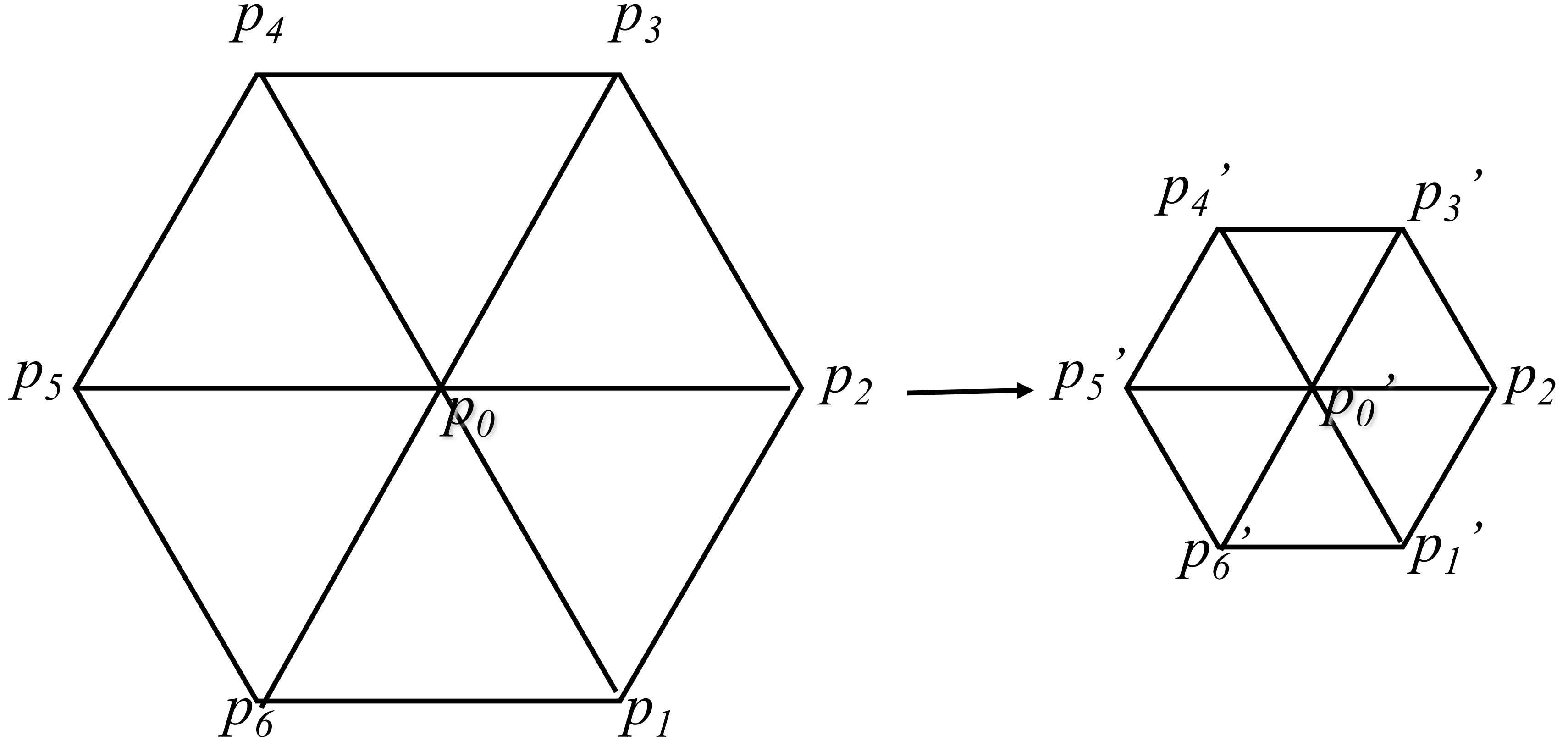
Computing infinitely many iterations is computationally prohibitive!!!



Subdivision Matrix

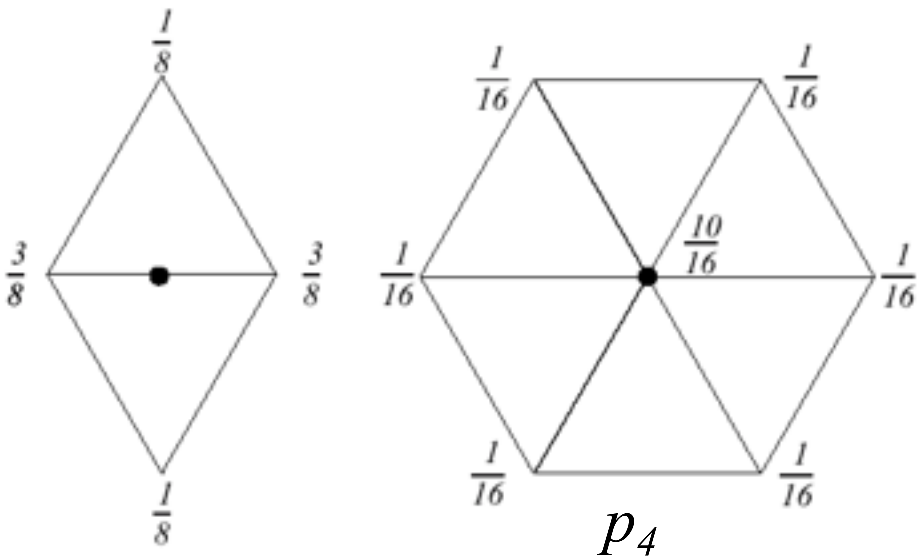
- Compute the new positions/vertices as a linear combination of previous ones.





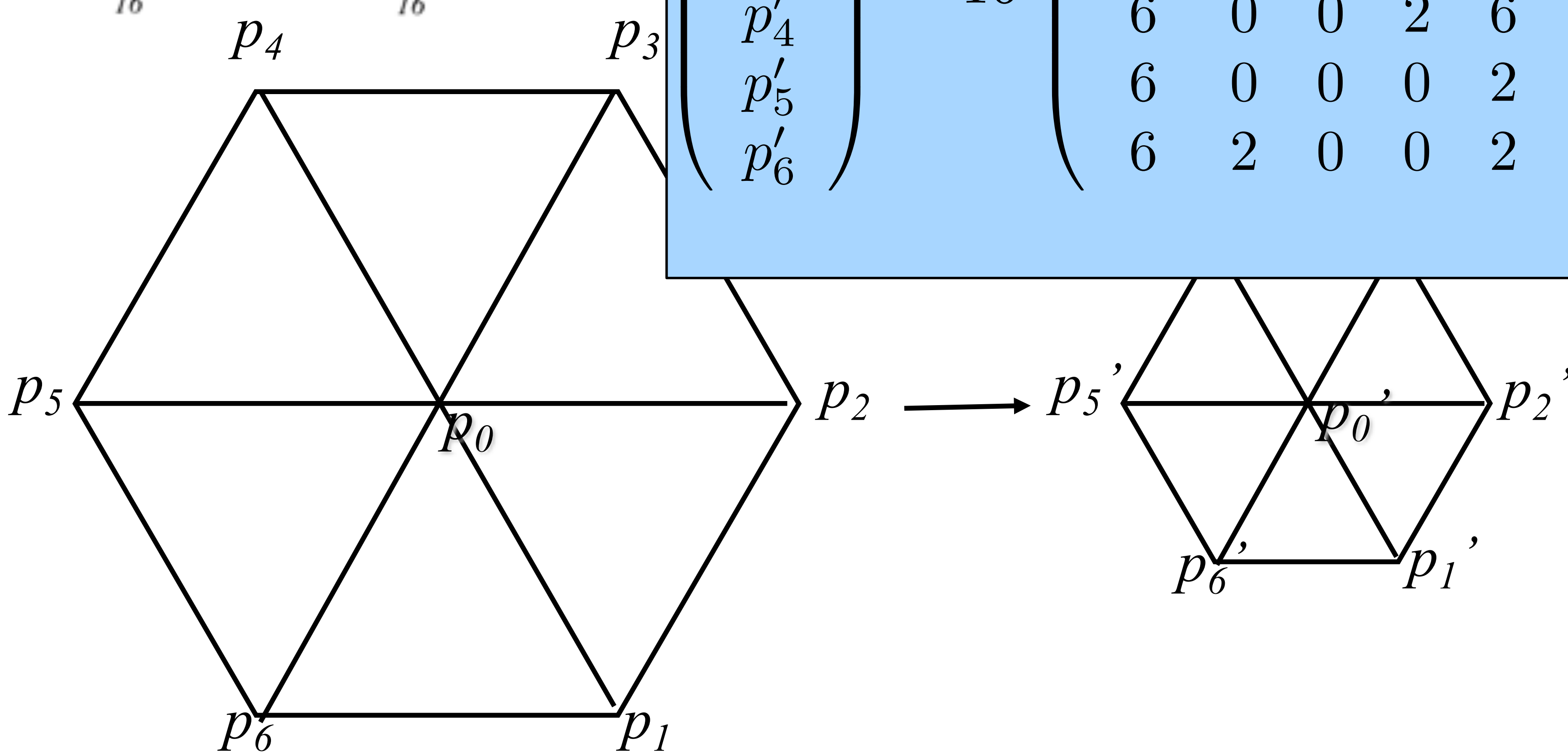
Subdivision Matrix

- Compute the new positions/vertices as a linear combination of previous ones.



Subdivision Matrix

$$\begin{pmatrix} p'_0 \\ p'_1 \\ p'_2 \\ p'_3 \\ p'_4 \\ p'_5 \\ p'_6 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 6 & 2 & 0 & 0 & 0 & 2 \\ 6 & 2 & 6 & 2 & 0 & 0 & 0 \\ 6 & 0 & 2 & 6 & 2 & 0 & 0 \\ 6 & 0 & 0 & 2 & 6 & 2 & 0 \\ 6 & 0 & 0 & 0 & 2 & 6 & 2 \\ 6 & 2 & 0 & 0 & 2 & 2 & 6 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix}$$



Subdivision Matrix

- Compute the new positions/vertices as a linear combination of previous ones.
- To find the limit position of p_0, repeatedly apply the subdivision matrix.

$$\begin{pmatrix} p_0^{(n)} \\ p_1^{(n)} \\ p_2^{(n)} \\ p_3^{(n)} \\ p_4^{(n)} \\ p_5^{(n)} \\ p_6^{(n)} \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 6 & 2 & 0 & 0 & 0 & 2 \\ 6 & 2 & 6 & 2 & 0 & 0 & 0 \\ 6 & 0 & 2 & 6 & 2 & 0 & 0 \\ 6 & 0 & 0 & 2 & 6 & 2 & 0 \\ 6 & 0 & 0 & 0 & 2 & 6 & 2 \\ 6 & 2 & 0 & 0 & 2 & 2 & 6 \end{pmatrix}^n \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix}$$

Subdivision Matrix

- Compute the new positions/vertices as a linear combination of previous ones.
- To find the limit position of p_0, repeatedly apply the subdivision matrix.

$$\begin{pmatrix} p_0^{(n)} \\ p_1^{(n)} \\ \vdots \end{pmatrix} \quad \begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 6 & 2 & 0 & 0 & 0 & 2 \\ \vdots \end{pmatrix}^n \begin{pmatrix} p_0 \\ p_1 \\ \vdots \end{pmatrix}$$

If, after a change of basis we have $M=A^{-1}DA$, where D is a diagonal matrix, then:

$$M^n=A^{-1}D^nA,$$

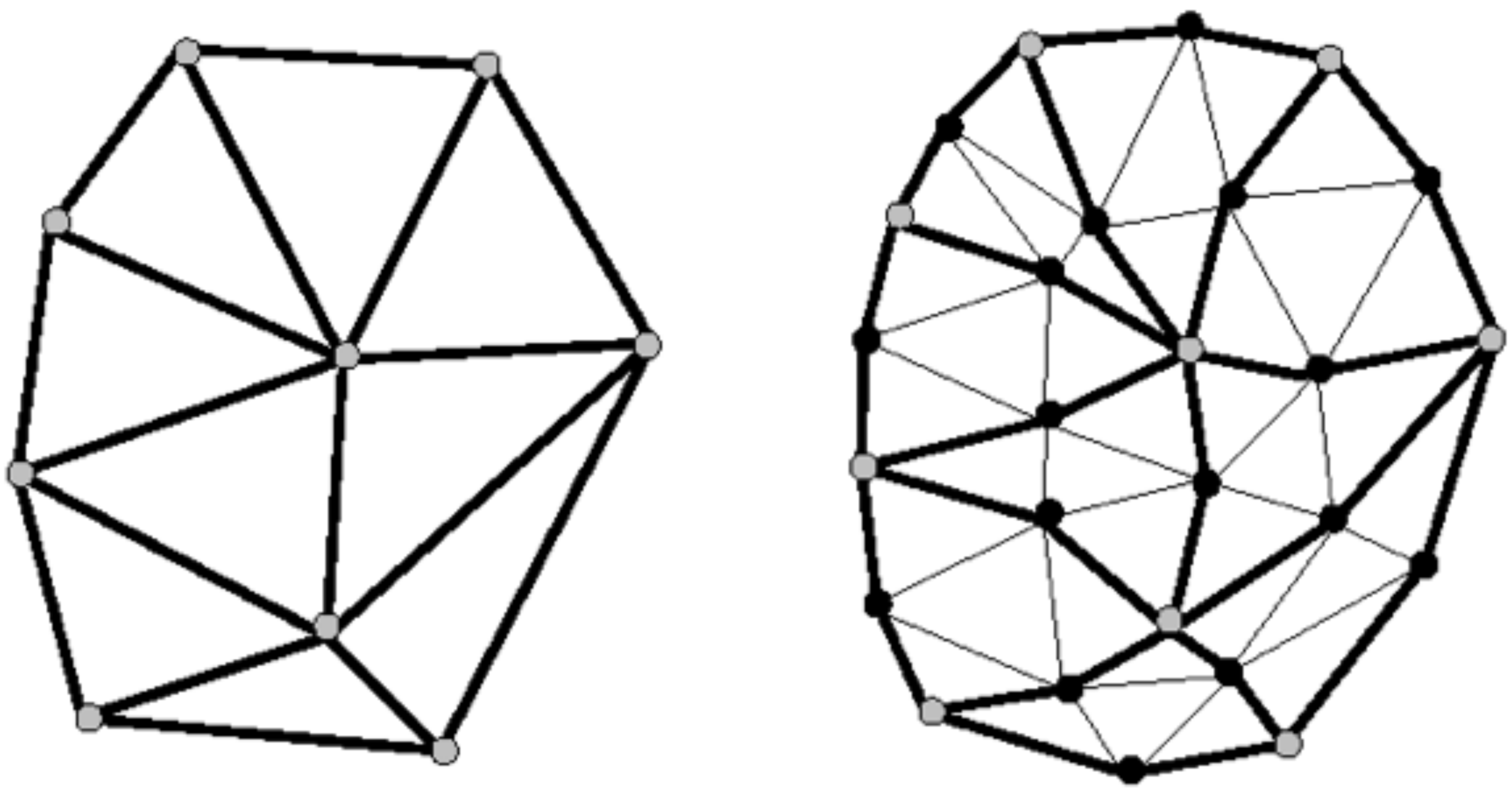
Since D is diagonal, raising D to the n-th power just amounts to raising each of the diagonal entries of D to the n-th power.

Subdivision Modeling

- ZBrush Modeling Session

Key Questions

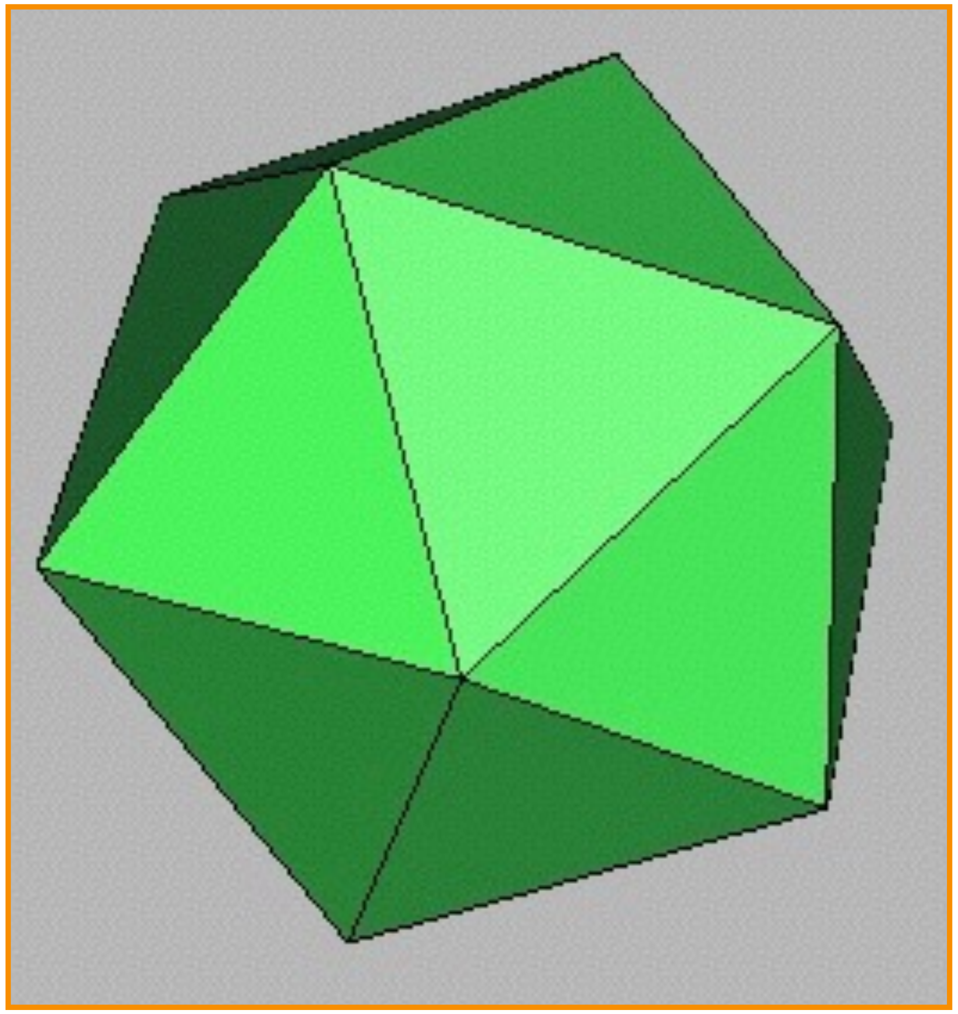
- How to refine the mesh?
  - o Aim for properties like smoothness
- **How to store the mesh?**
  - o Aim for efficiency for implementing subdivision rules



Zorin & Schroeder
SIGGRAPH 99
Course Notes

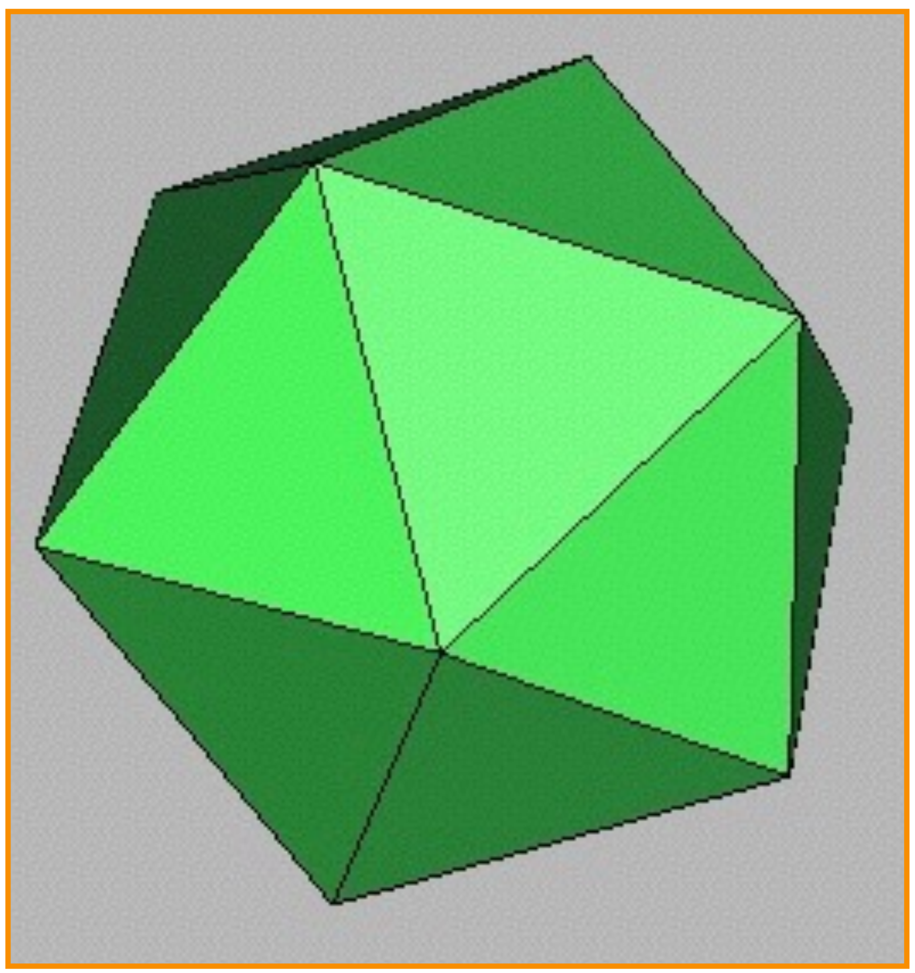
Polygon Meshes

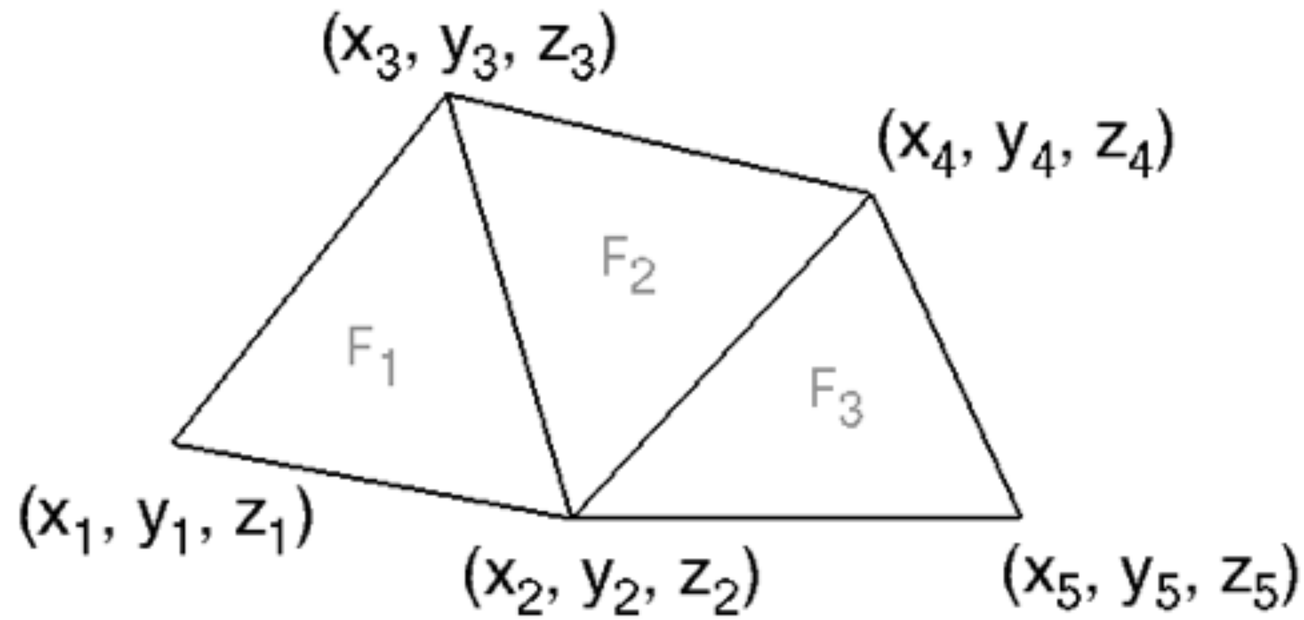
- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge



Independent Faces

- Each face lists vertex coordinates

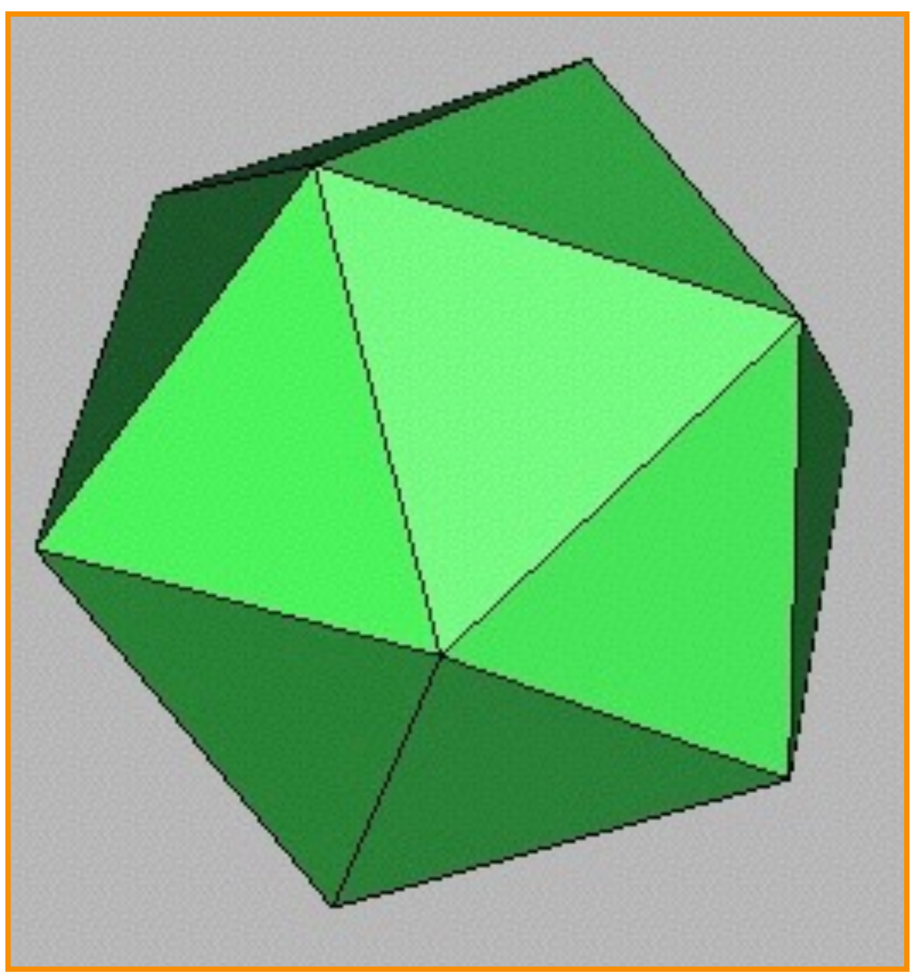


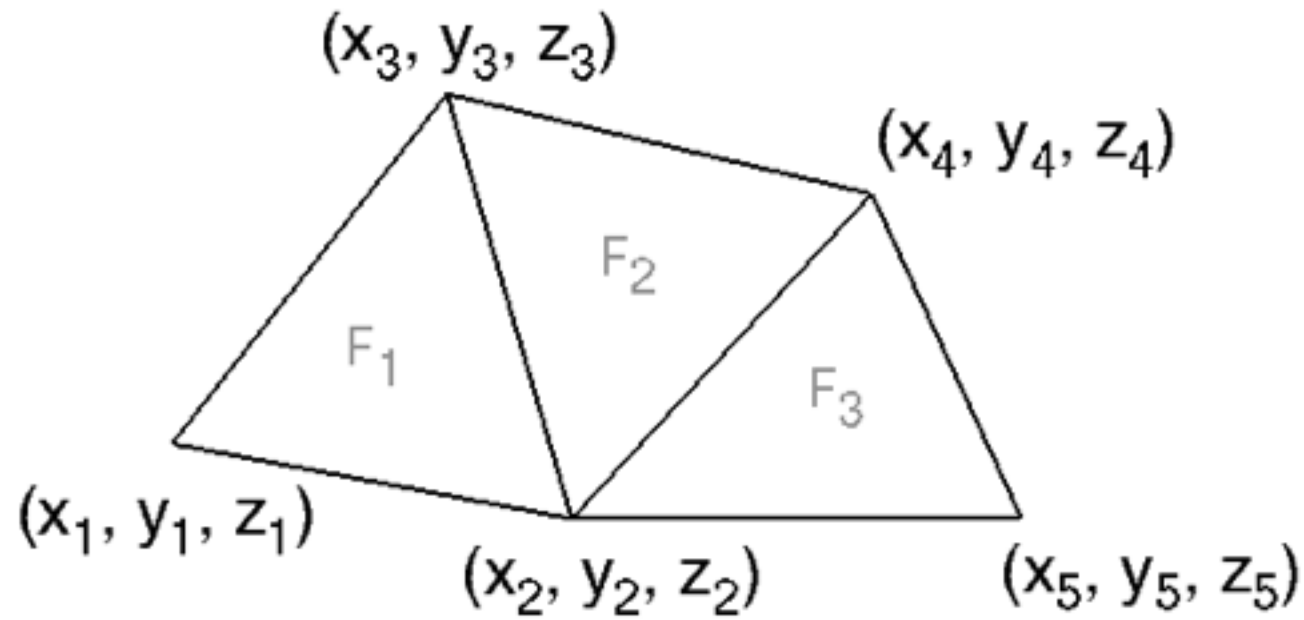


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Independent Faces

- Each face lists vertex coordinates
  - ✗ Redundant vertices
  - ✗ No topology information

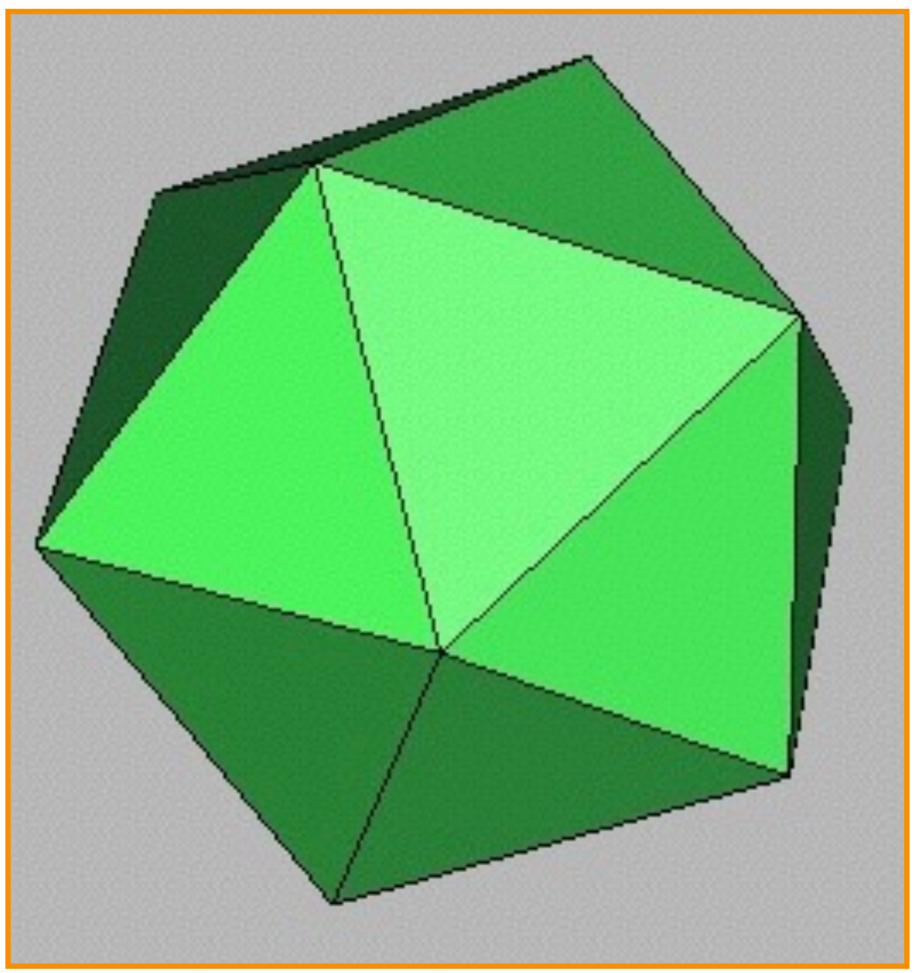


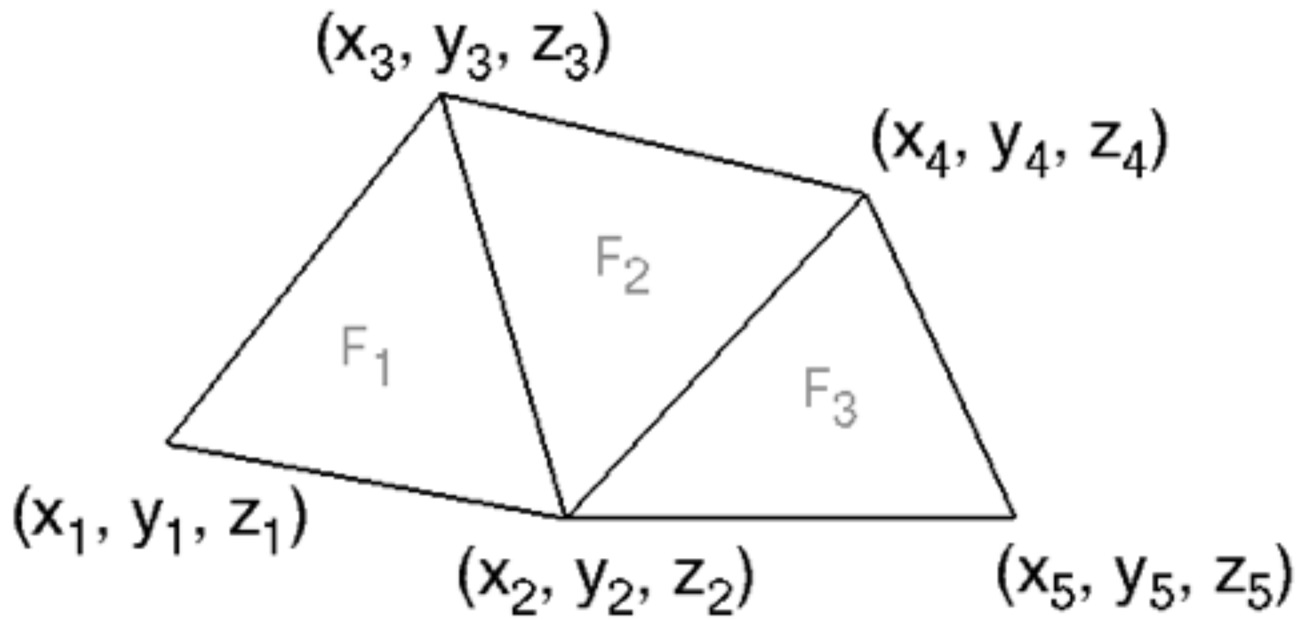


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

- Each face lists vertex references



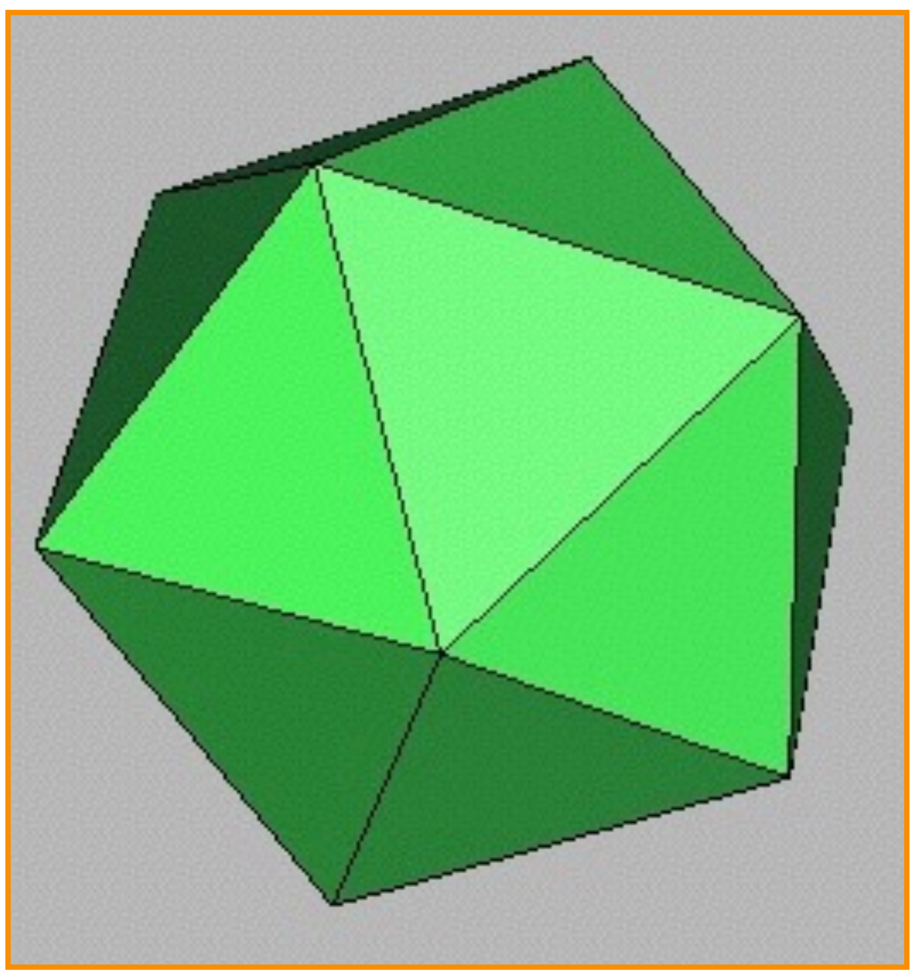


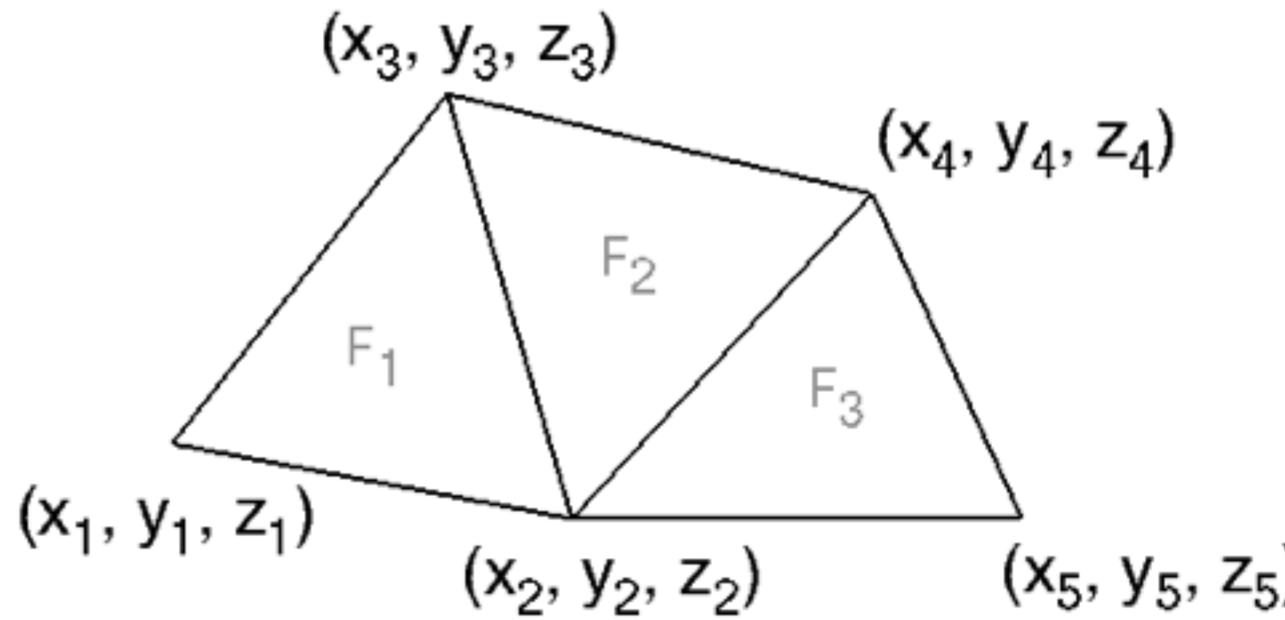
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Vertex and Face Tables

- Each face lists vertex references
  - ✓Shared vertices



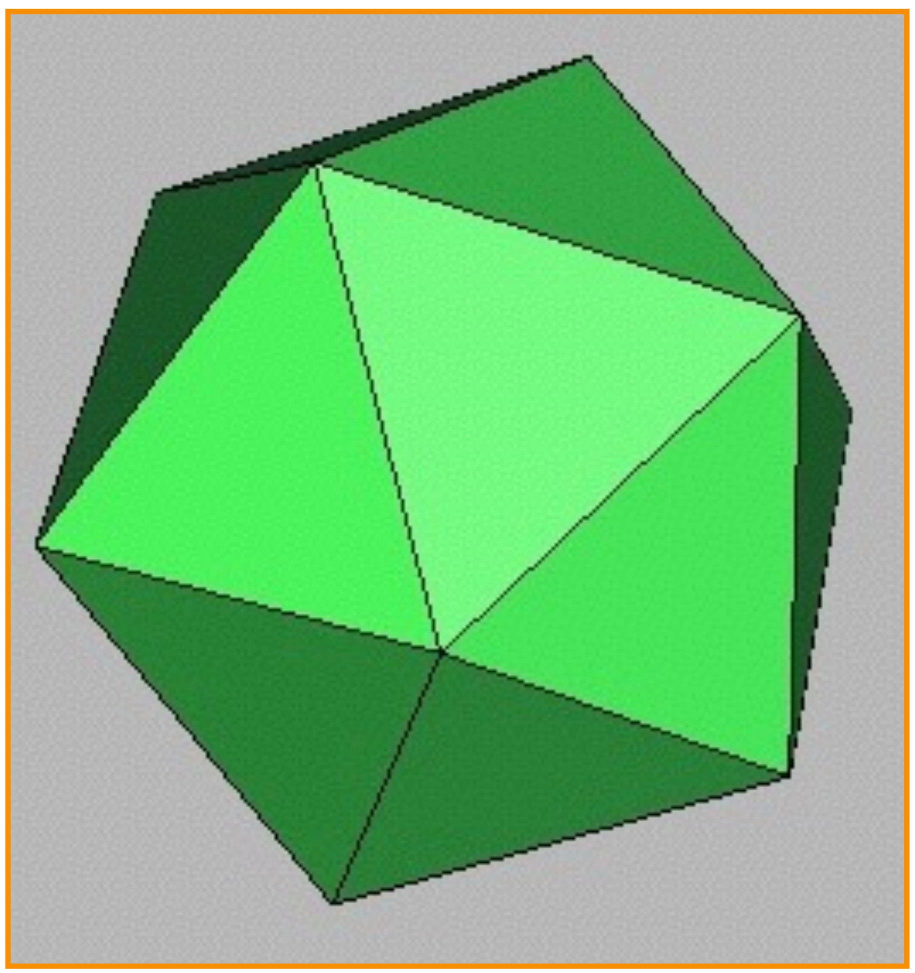


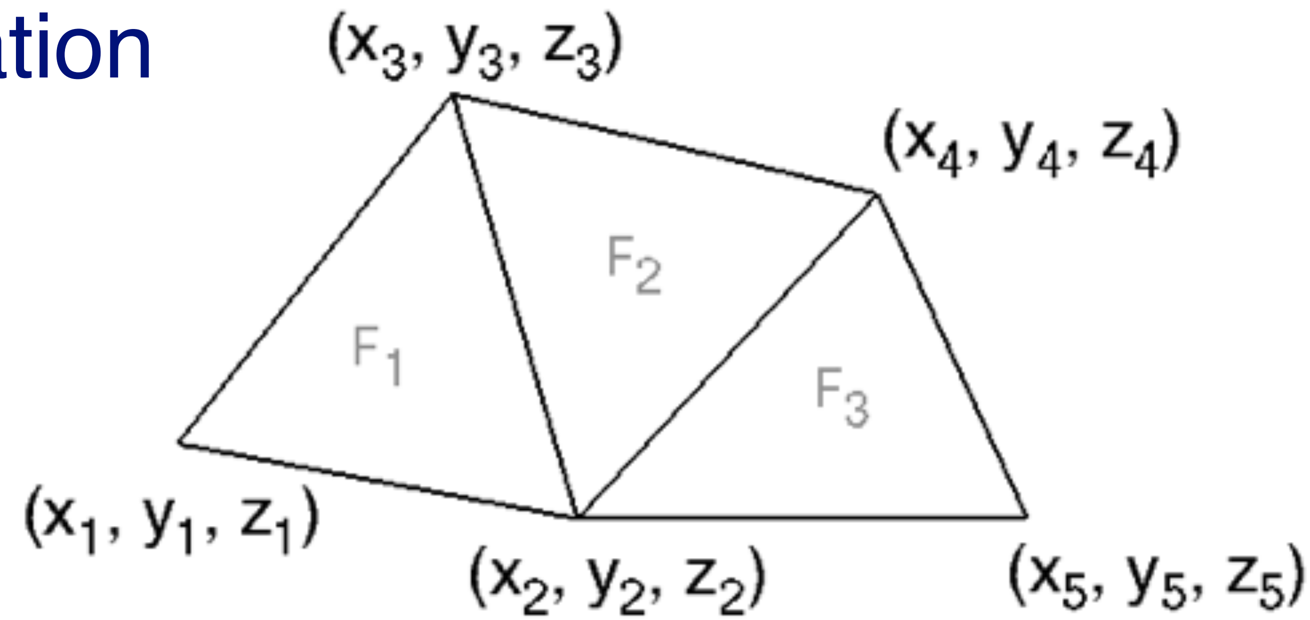
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Vertex and Face Tables

- Each face lists vertex references
  - ✓Shared vertices
  - ✗Still no topology information



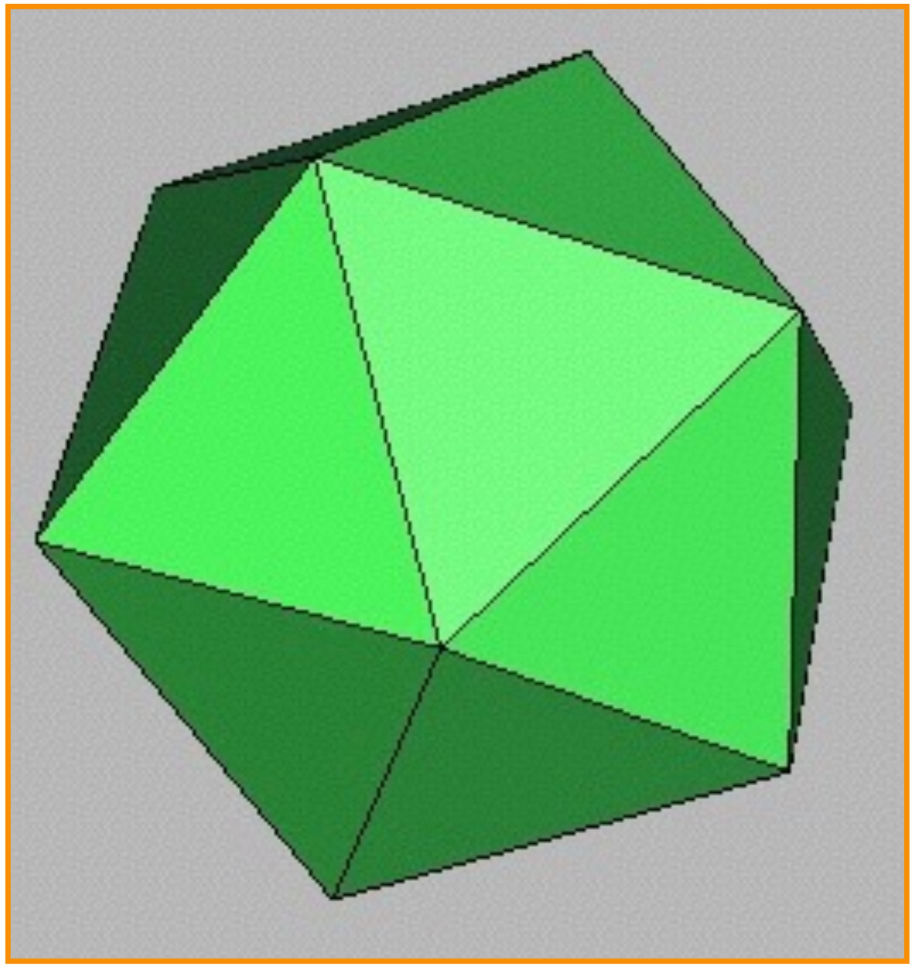


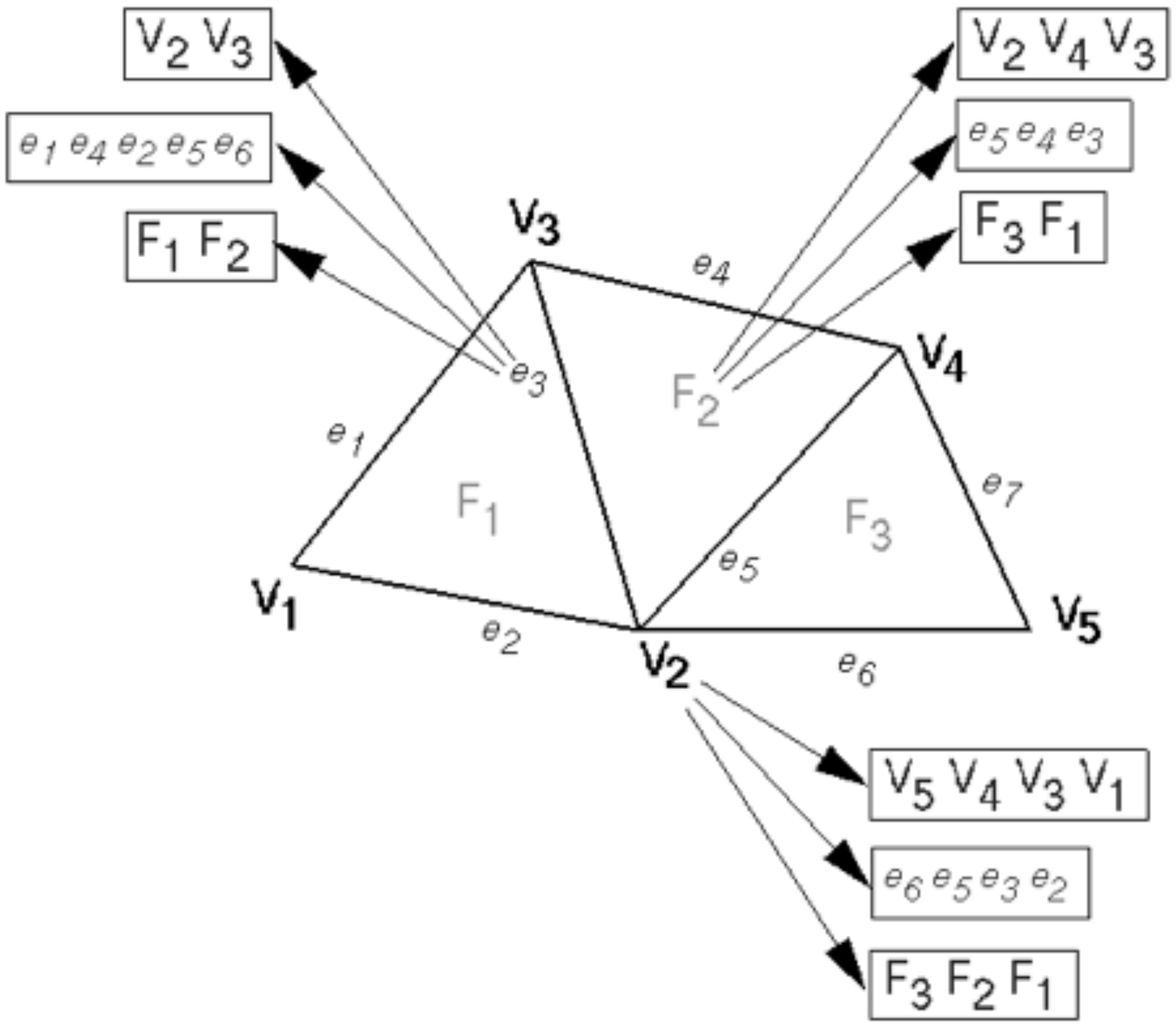
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Adjacency Lists

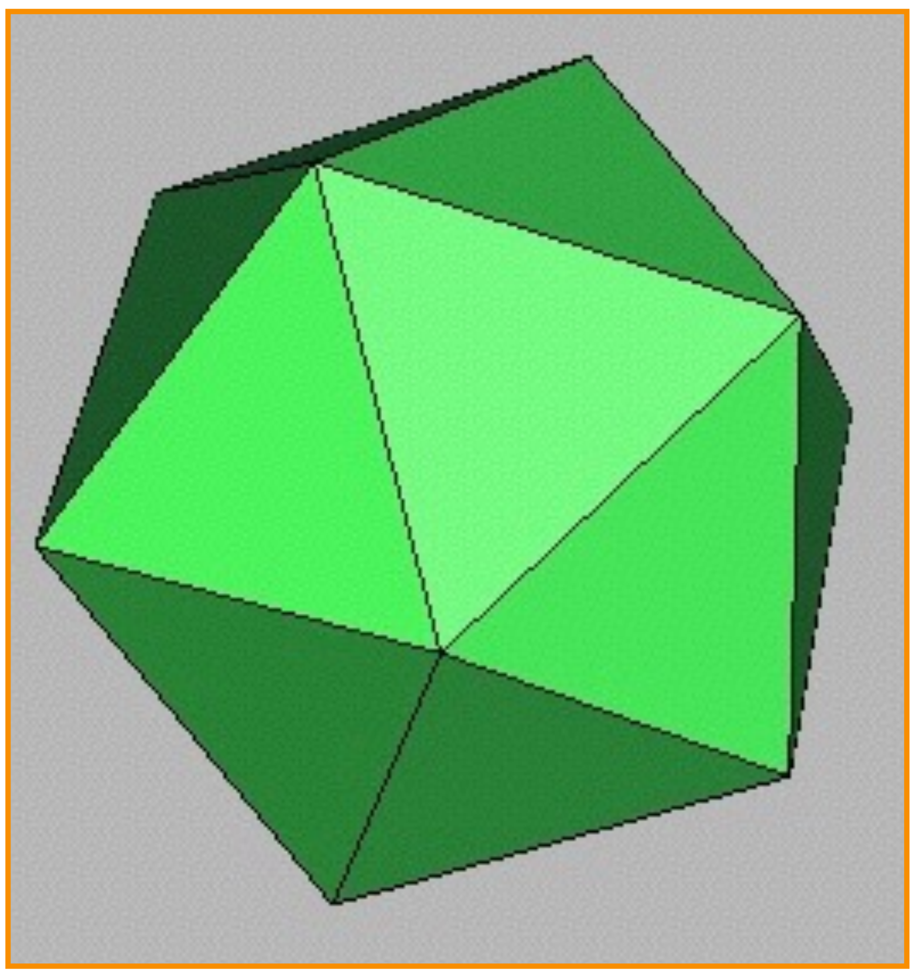
- Store all vertex, edge, and face adjacencies

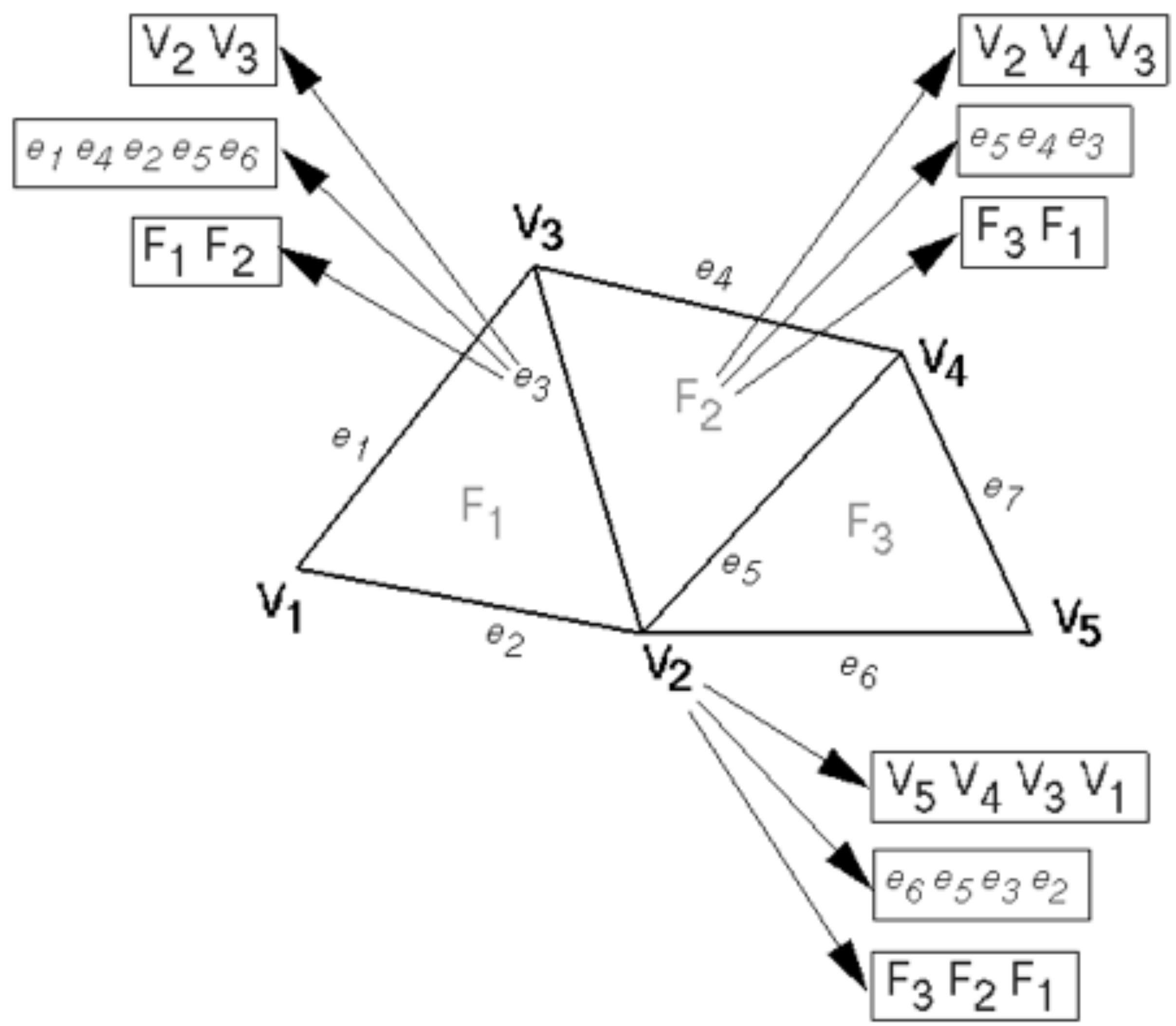




Adjacency Lists

- Store all vertex, edge, and face adjacencies
  - ✓Efficient topology traversal





Adjacency Lists

- Store all vertex, edge, and face adjacencies
  - ✓ Efficient topology traversal
  - ✗ Extra storage
  - ✗ Variable size arrays