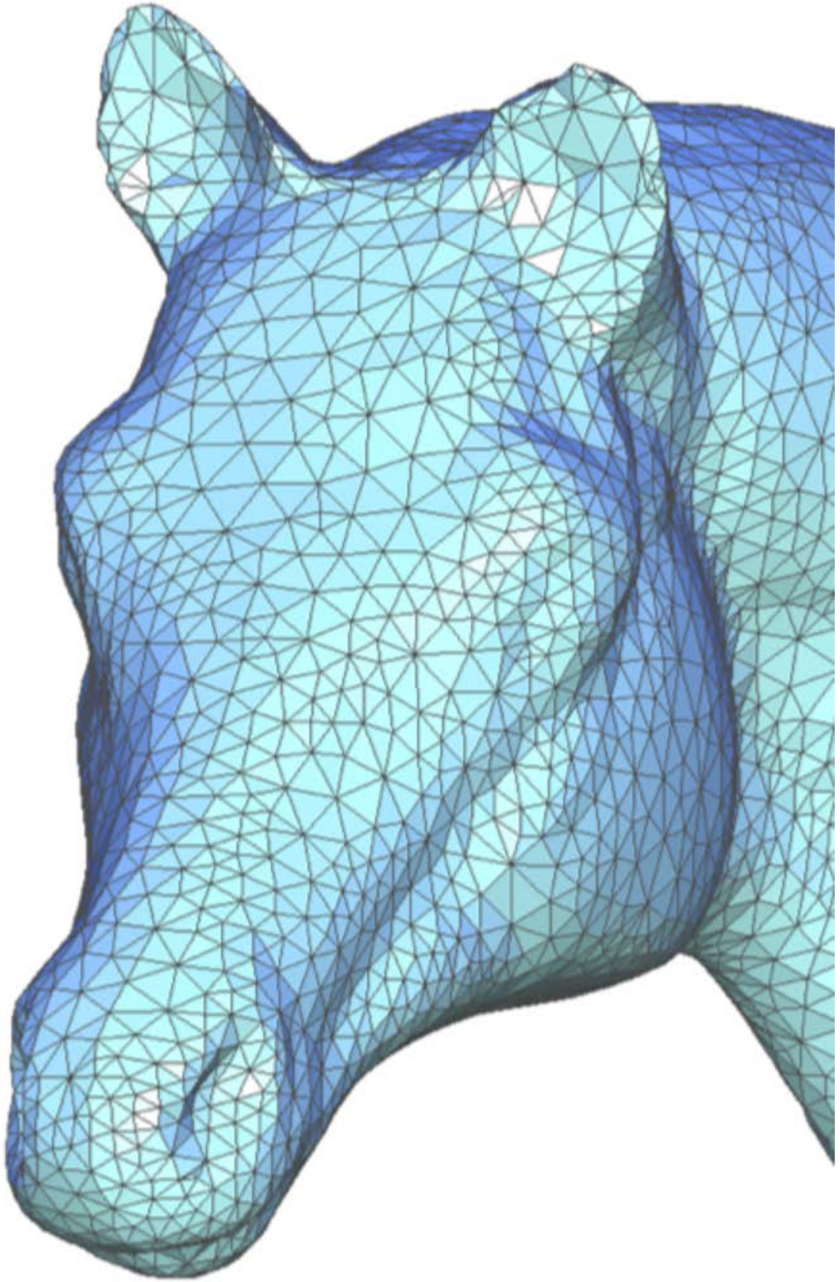
CSC418/2504 Computer Graphics

Some Slides/Images adapted from Marschner and Shirley

Meshes

Ottawa Convention Center



Meshes

Types of Surfaces

Triangles

Data Structures for Triangle Meshes

Normals for Meshes

Texture Mapping

Subdivision Surfaces

Announcements

Assignment 4 due Friday

Assignment 5 out soon

Any Questions ?

Surface Representations in Graphics

What are the two main types of surface representations ?

Surface Representations in Graphics

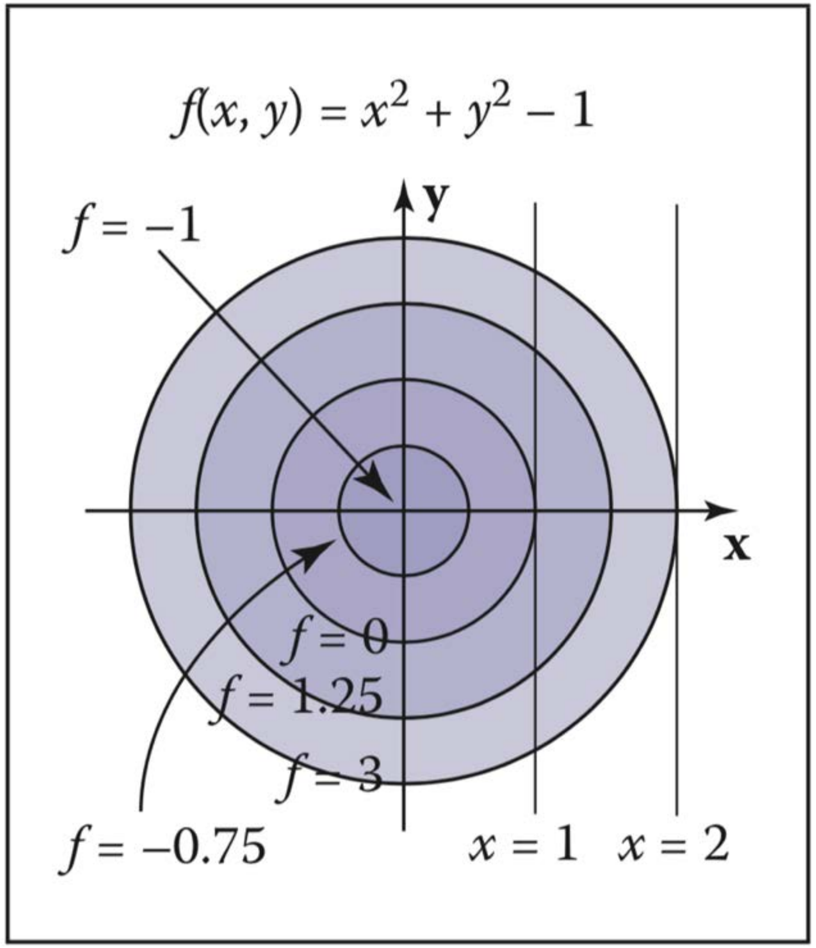
What are the two main types of surface representations ?

Implicit and Parametric

How do you define each type ?

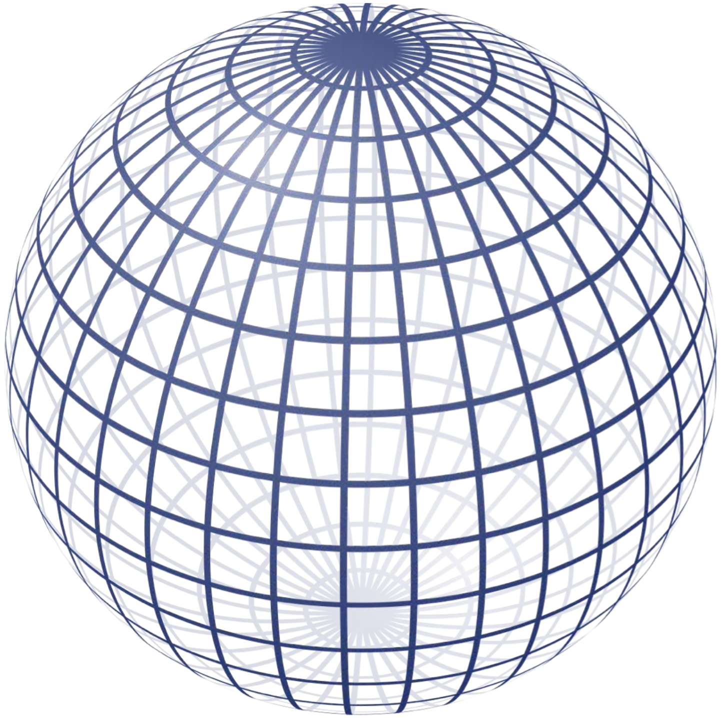
Surface Representations in Graphics

Implicit Surface

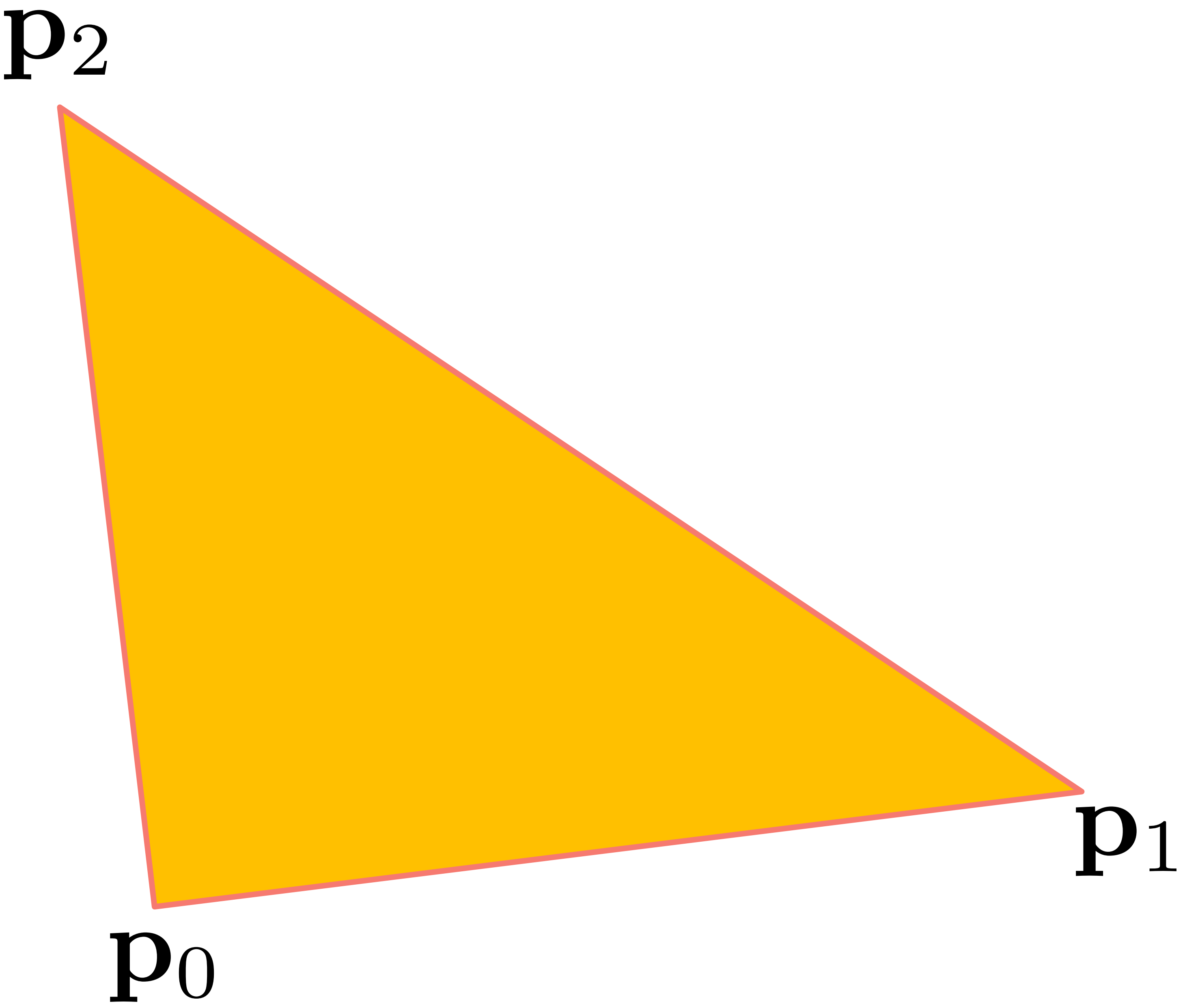


Parametric Surface

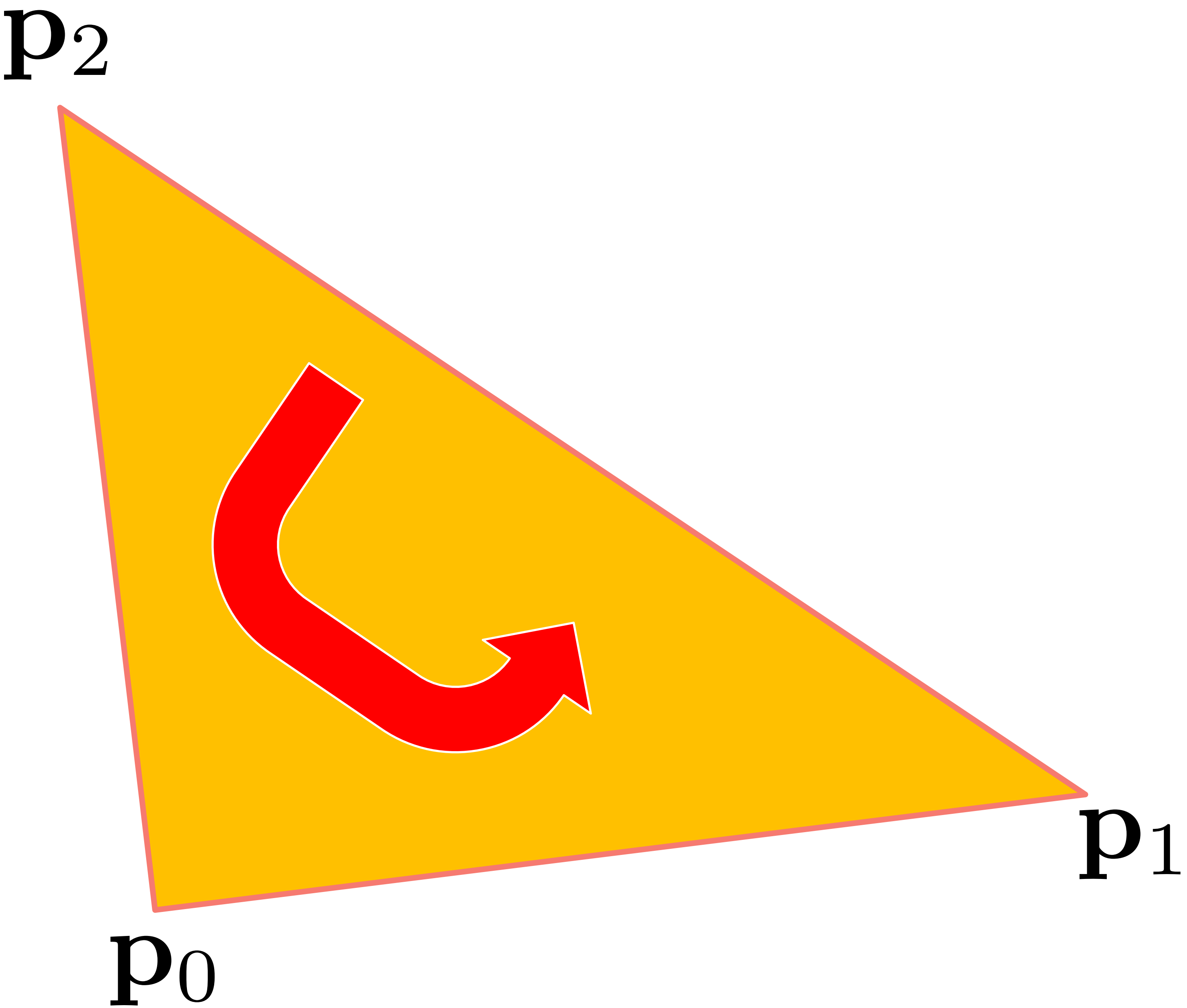
$$x = r\cos\phi\sin\theta,$$
$$y = r\sin\phi\sin\theta,$$
$$z = r\cos\theta.$$



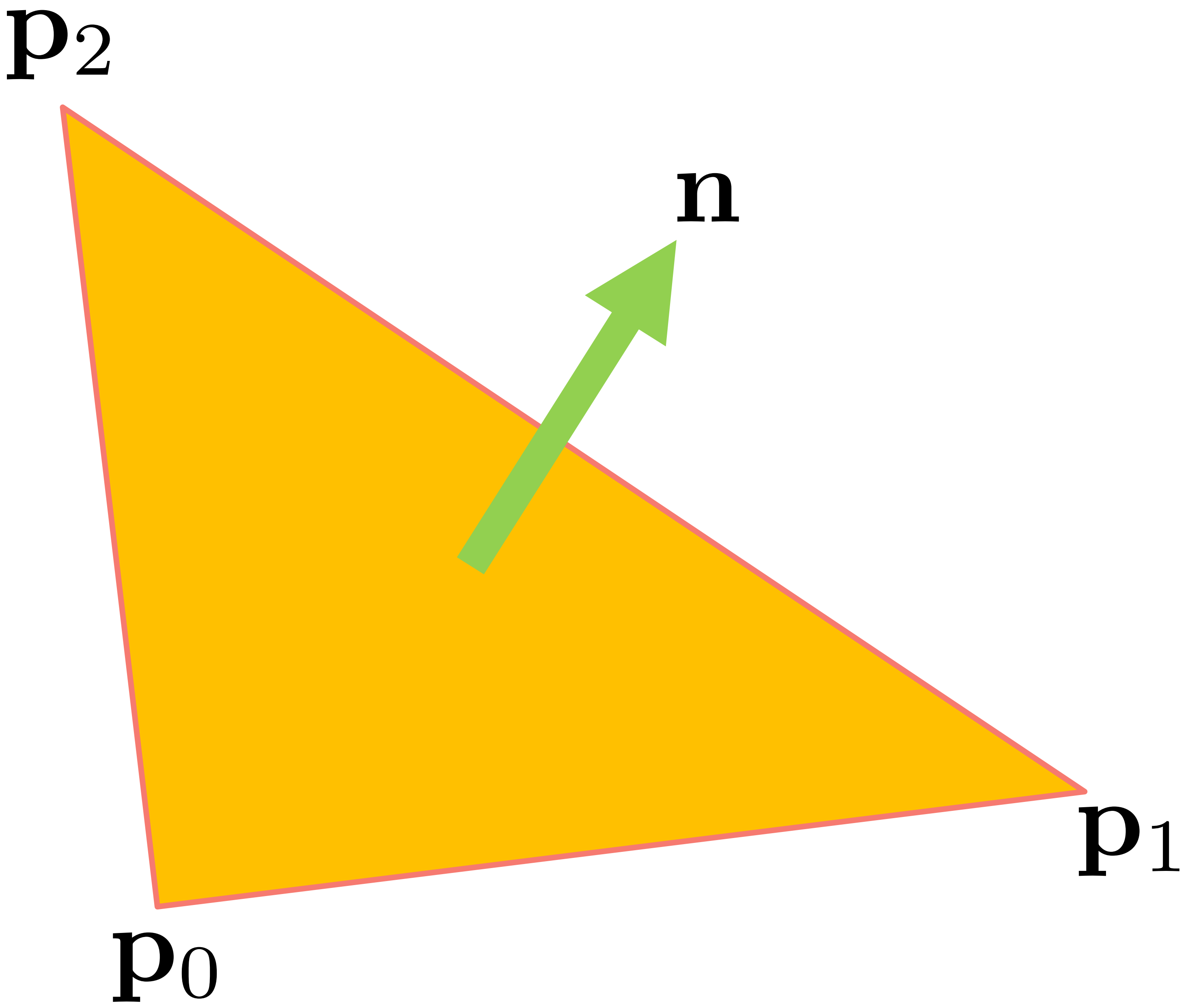
Triangles



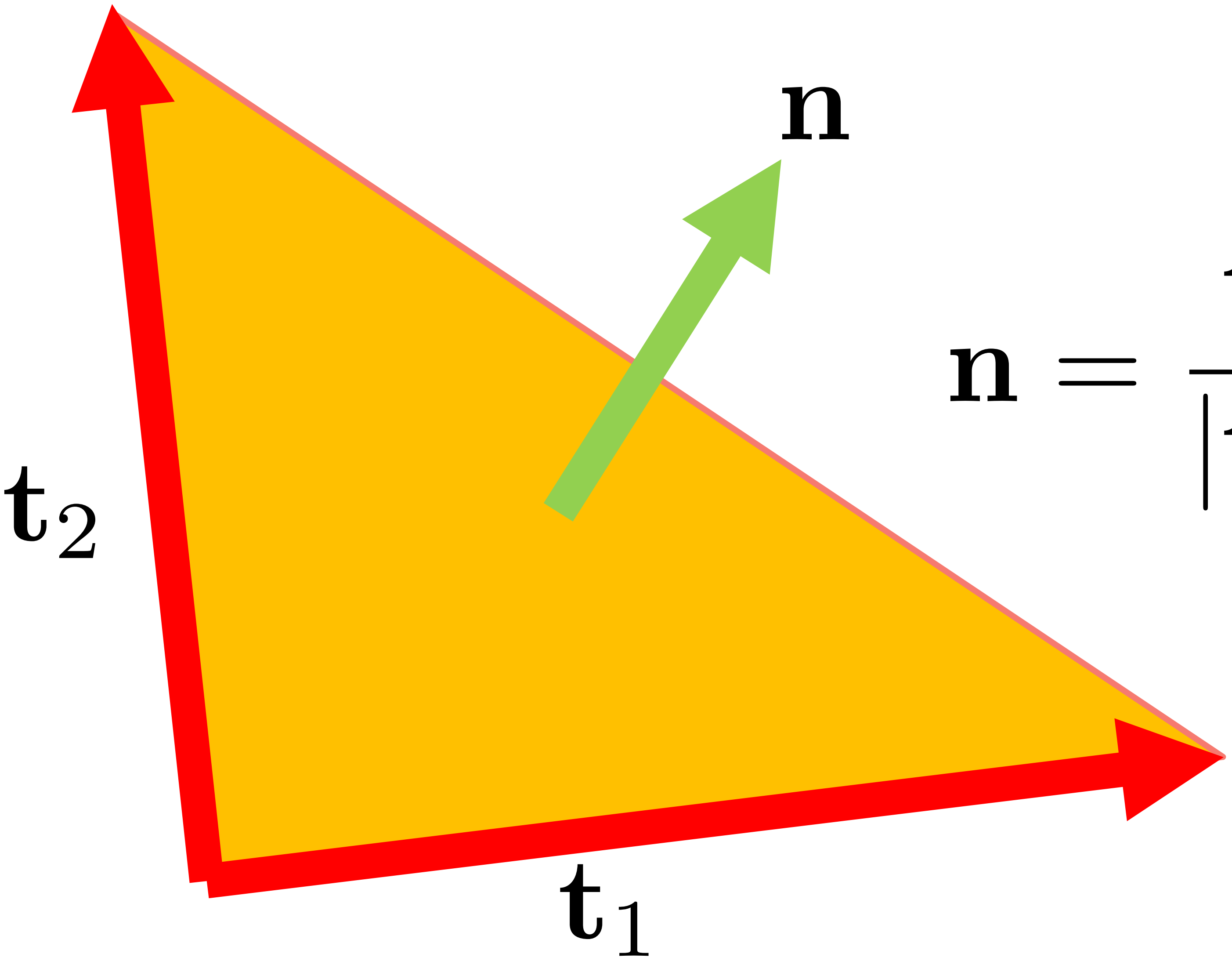
Triangles



Triangles



Triangles



$$\mathbf{n} = \frac{\mathbf{t}_1 \times \mathbf{t}_2}{|\mathbf{t}_1 \times \mathbf{t}_2|}$$

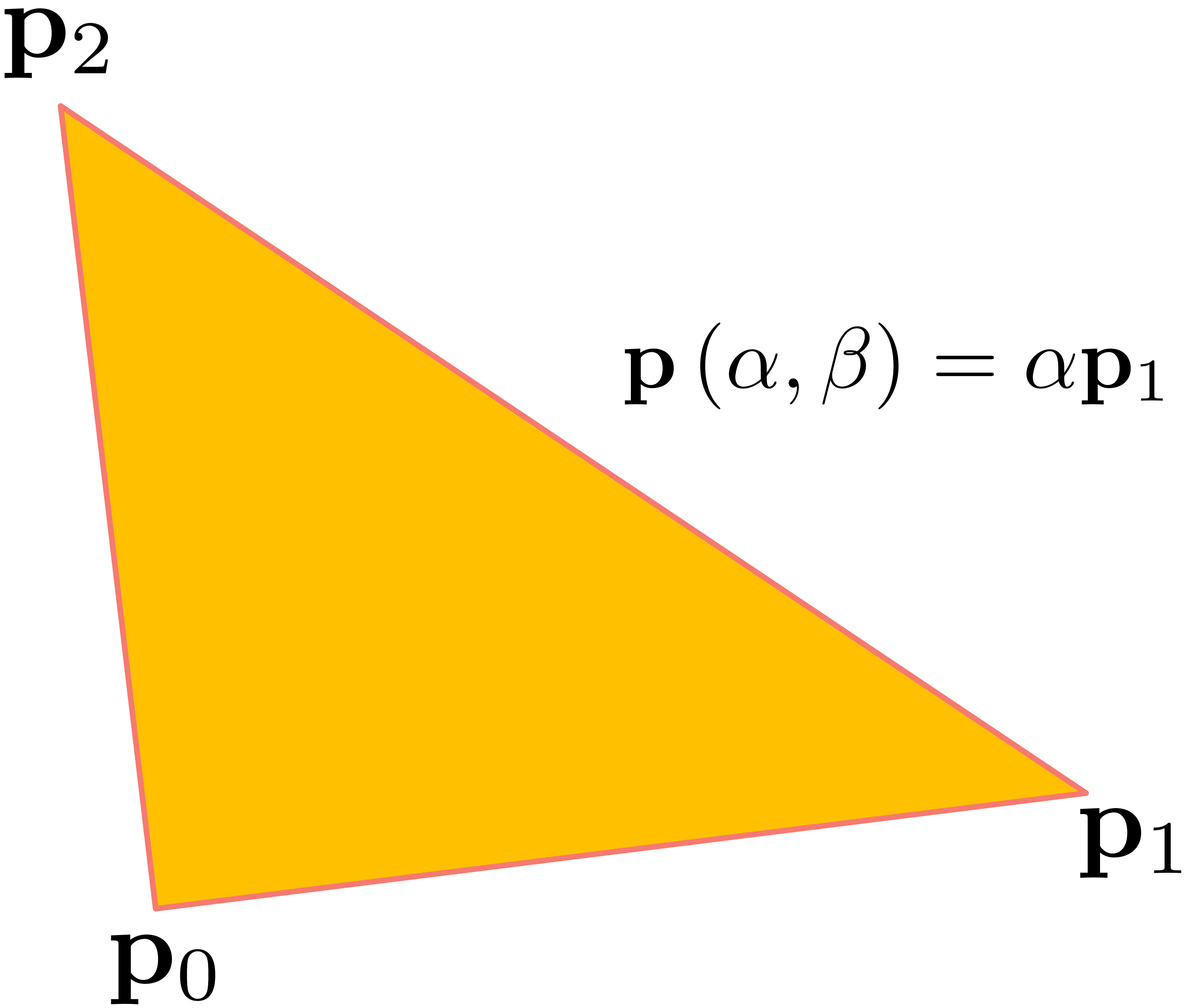
Barycentric Coordinates

$$\mathbf{p}_1 = \alpha\mathbf{t}_1 + \beta\mathbf{t}_2 + \mathbf{p}_0$$

$$\mathbf{p}\left(\alpha, \beta\right) = \alpha\left(\mathbf{p}_1 - \mathbf{p}_0\right) + \beta\left(\mathbf{p}_2 - \mathbf{p}_0\right) + \mathbf{p}_0$$

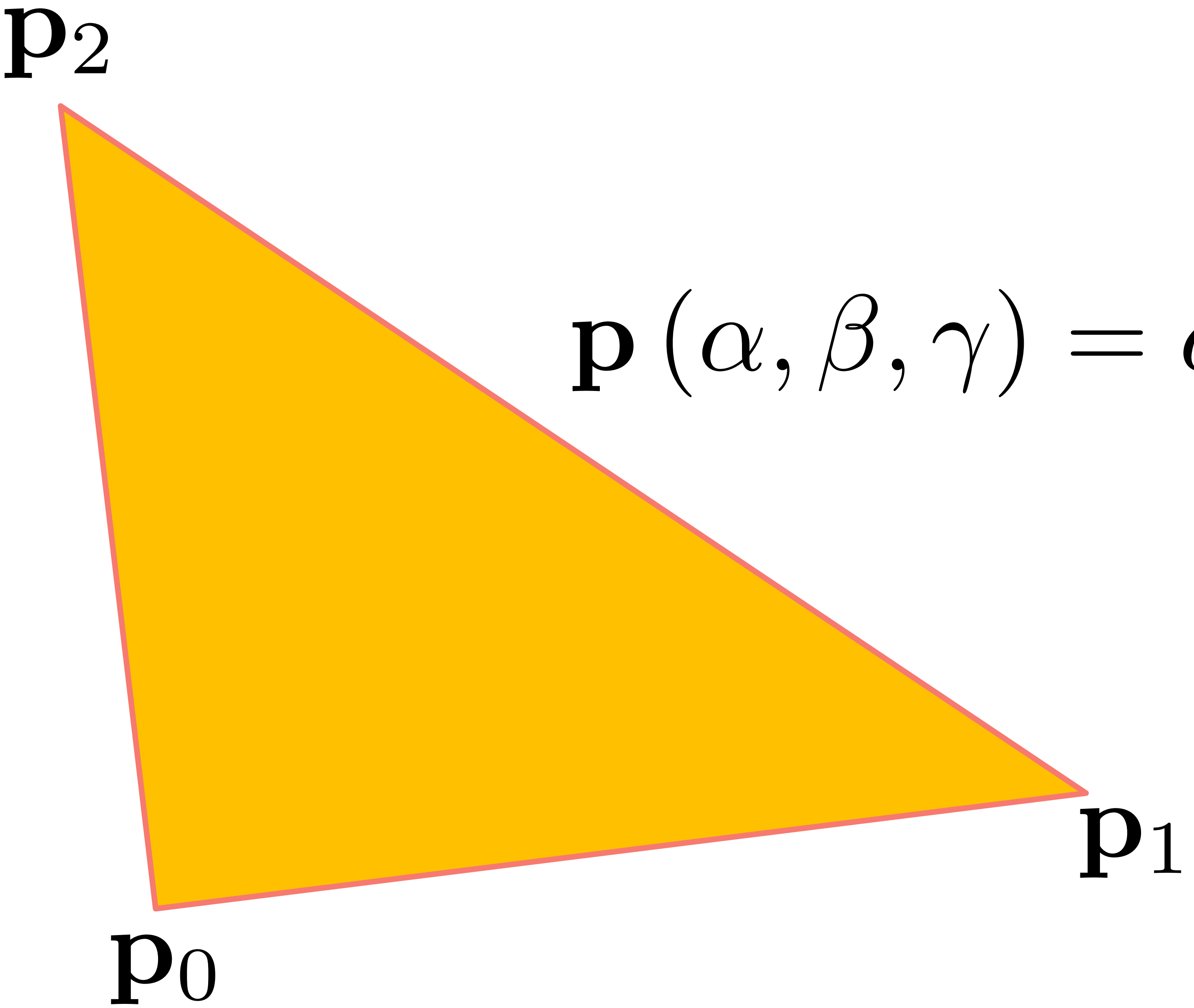
$$\mathbf{p}\left(\alpha, \beta\right) = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + (1 - \alpha - \beta)\mathbf{p}_0$$

Barycentric Coordinates



$$\mathbf{p}\,(\alpha, \beta) = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + (1 - \alpha - \beta)\mathbf{p}_0$$

Barycentric Coordinates



$$\mathbf{p}\left(\alpha, \beta, \gamma\right) = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_0$$

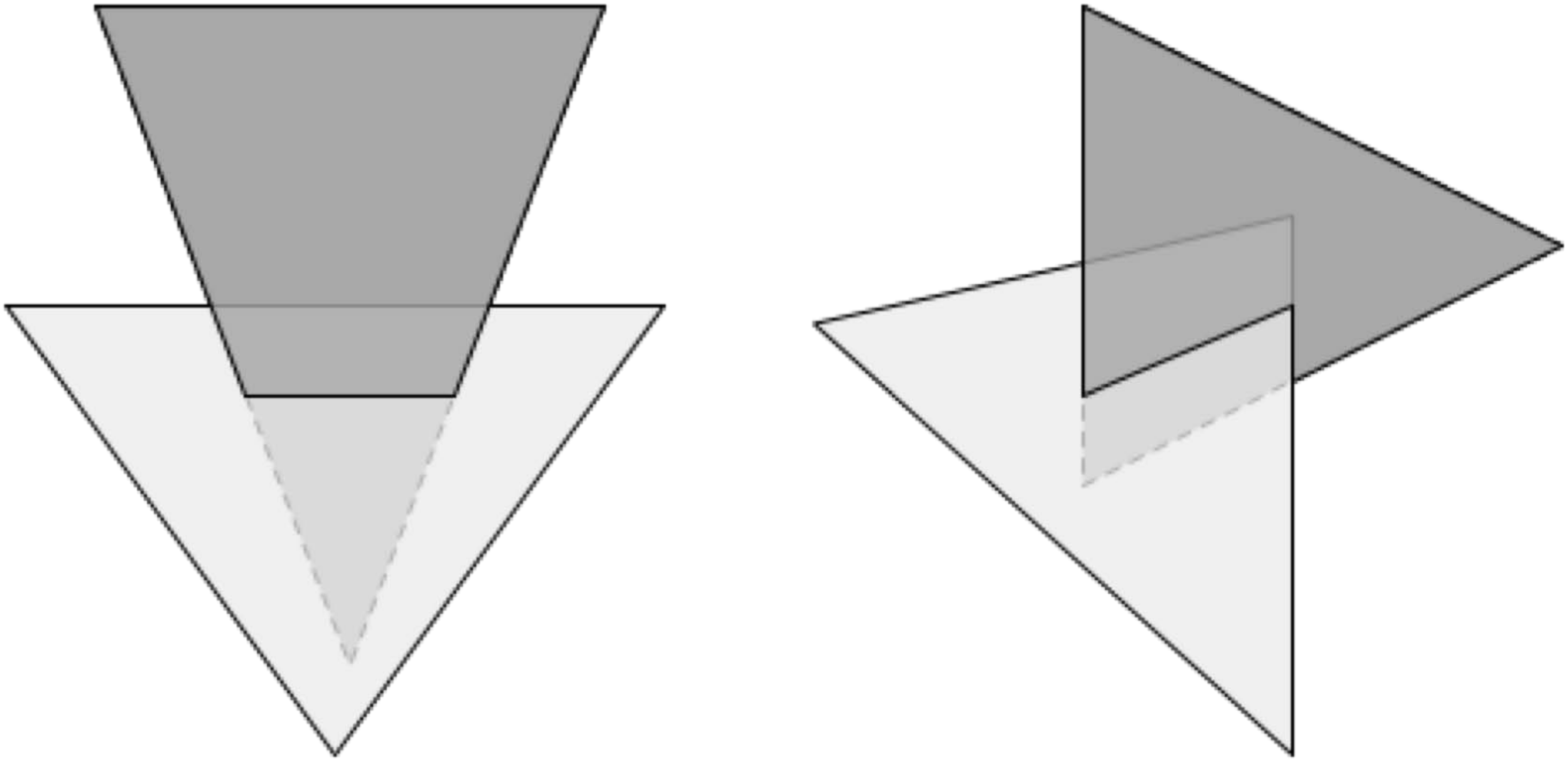
$$\alpha \geqslant 0$$

$$\beta \geqslant 0$$

$$\alpha + \beta \leqslant 1$$

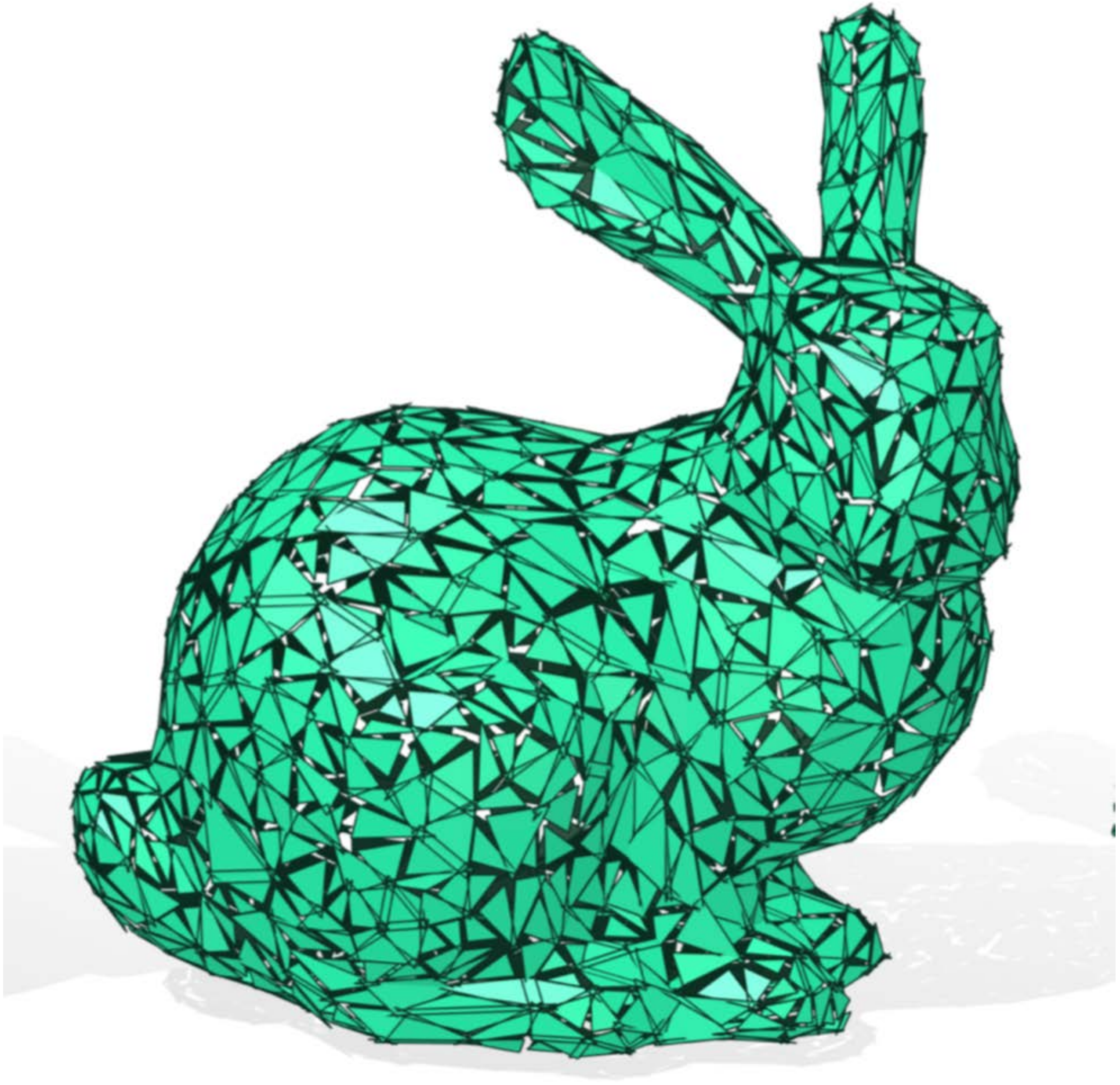
$$\gamma = 1 - \alpha - \beta$$

Triangle-Triangle Intersection Test

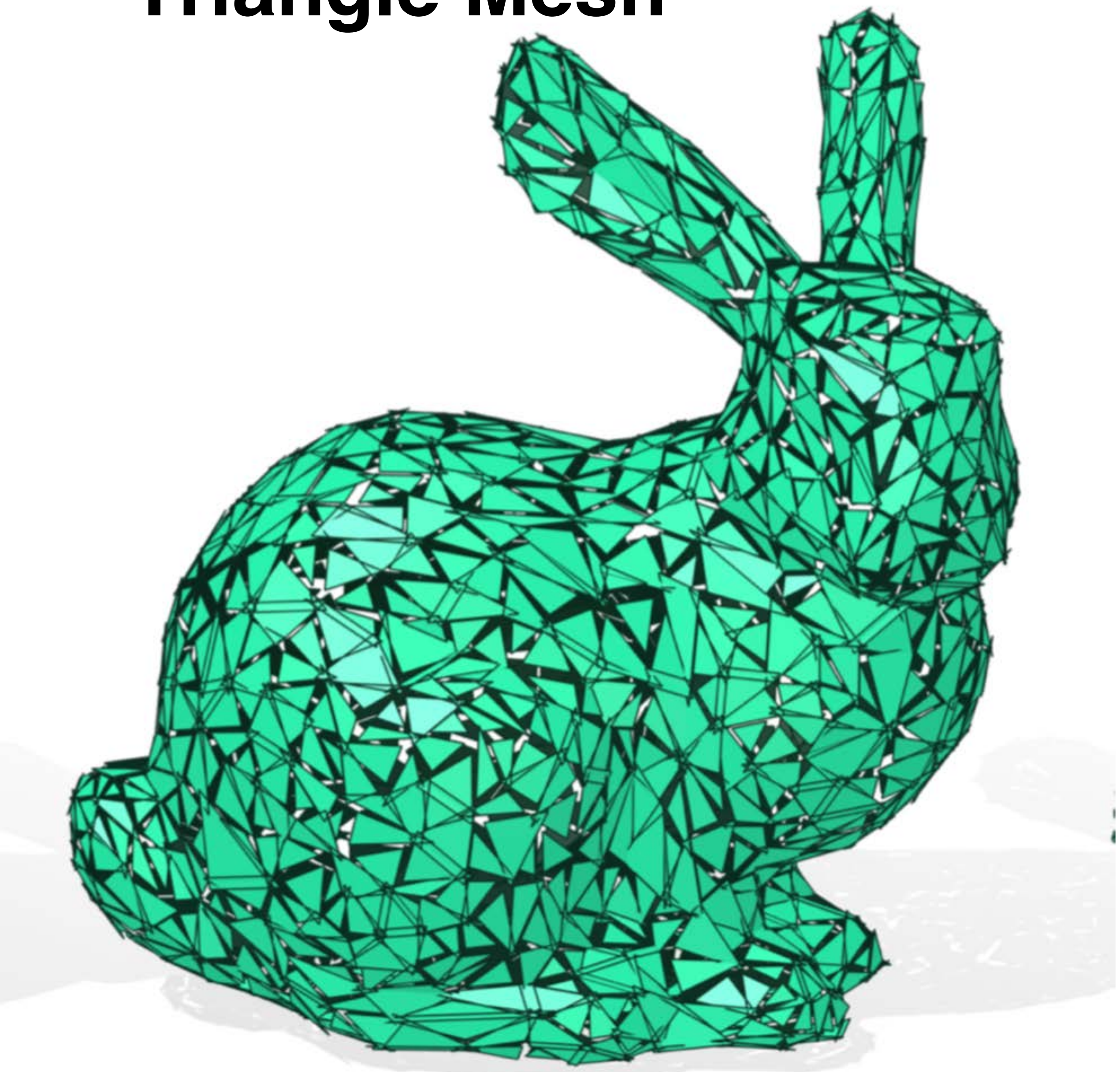


https://stackoverflow.com/questions/7113344/find-whether-two-triangles-intersect-or-not

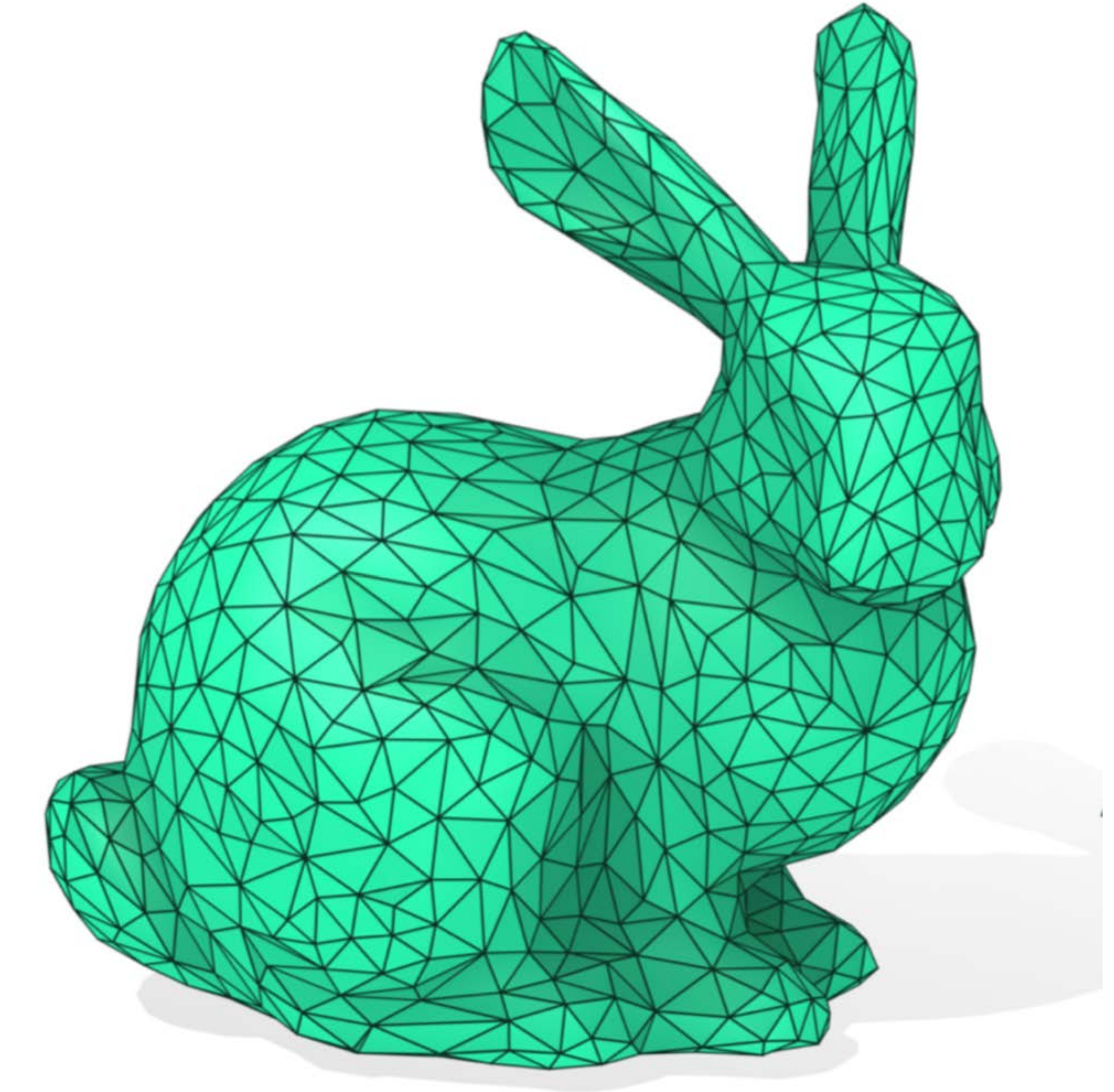
Triangle Soup



Triangle Mesh



Soup



Mesh

Topology

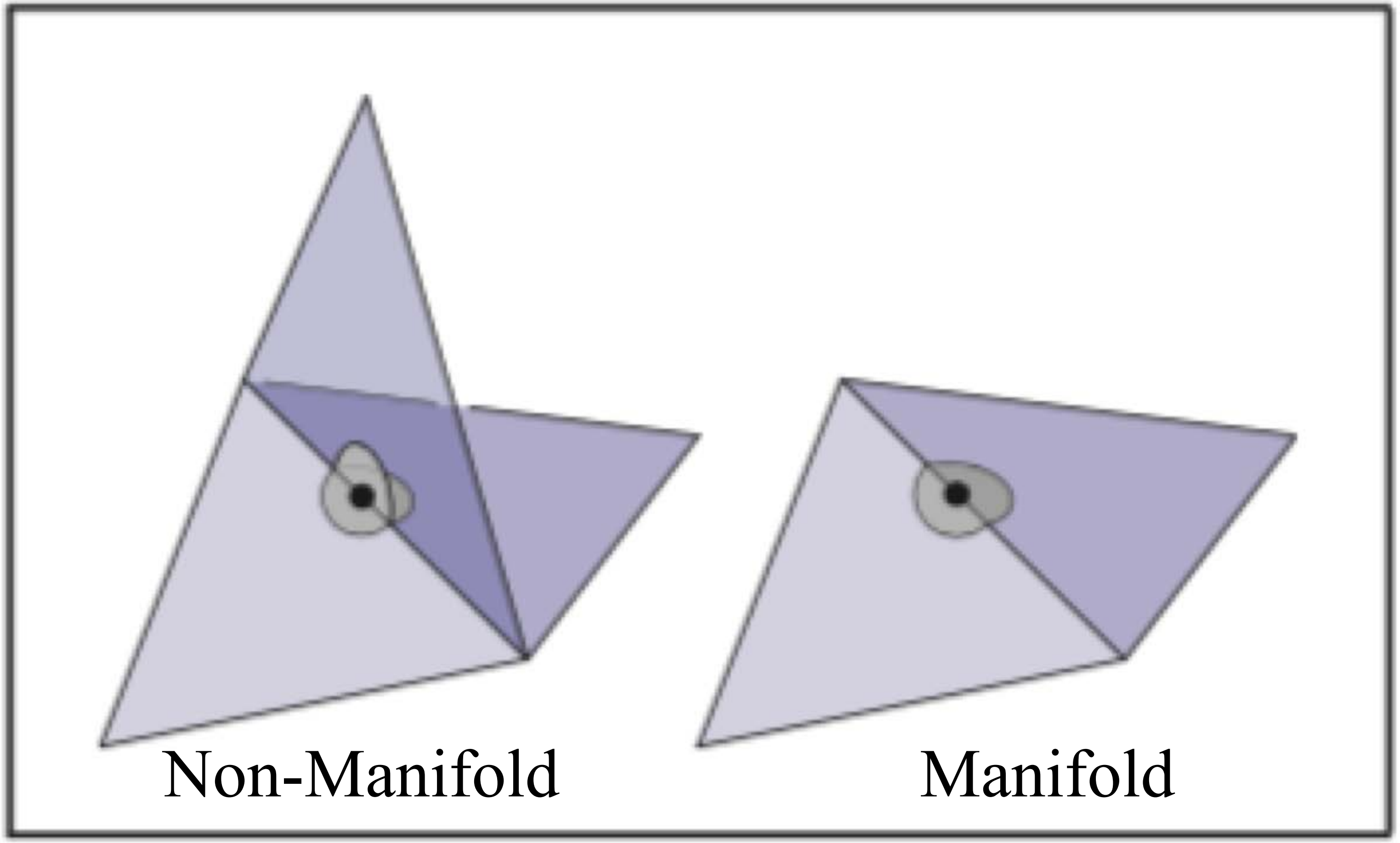
Topology is concerned with the connectivity of a mesh

Many algorithms are easier to implement or more efficient when connectivity data is available (we'll see an example of this later on).

We are going to assume that our meshes are *2-manifolds*

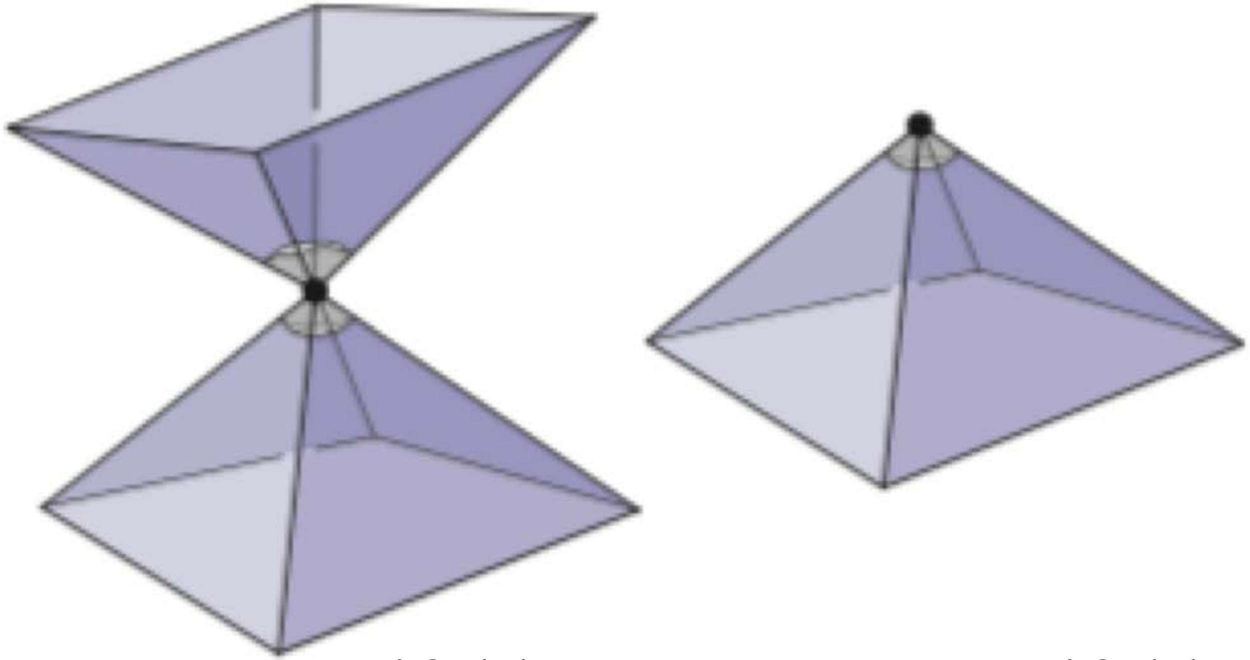
Manifold

A *2-manifold* is a surface for which the neighbourhood around any point can be flattened onto the plane



Manifold

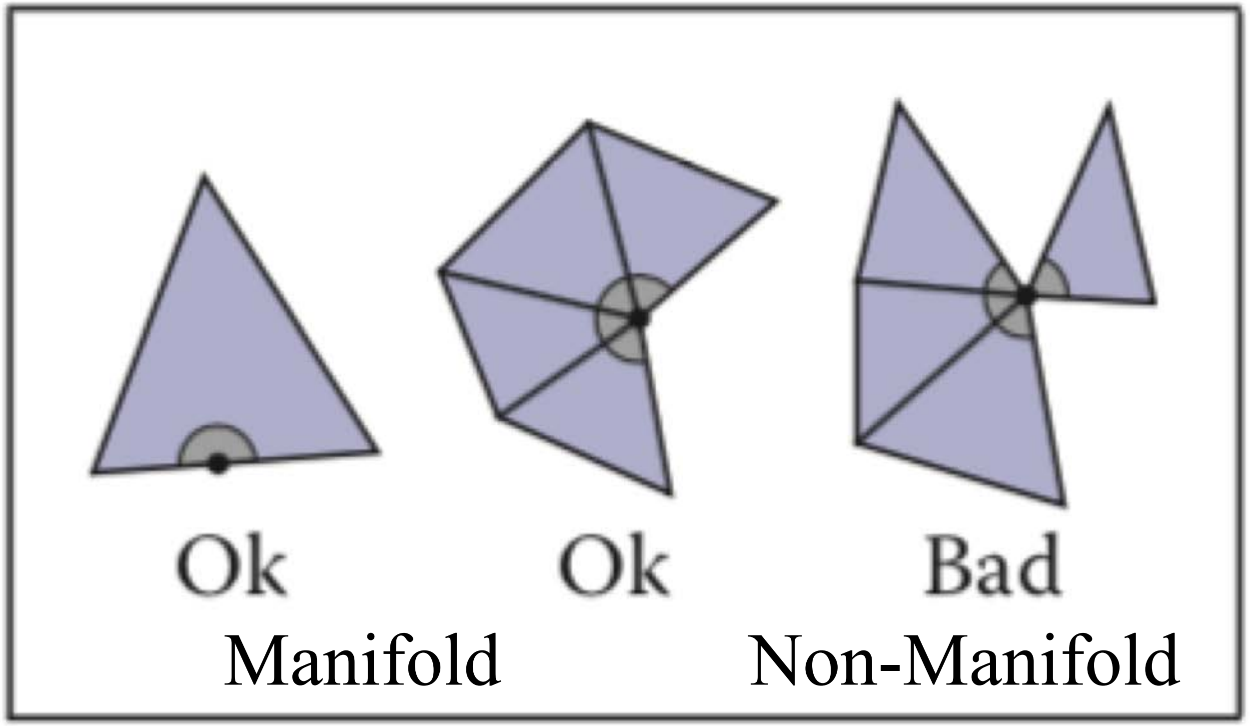
A *2-manifold* is a surface for which the neighbourhood around any point can be flattened onto the plane



Non-Manifold   Manifold

Manifold

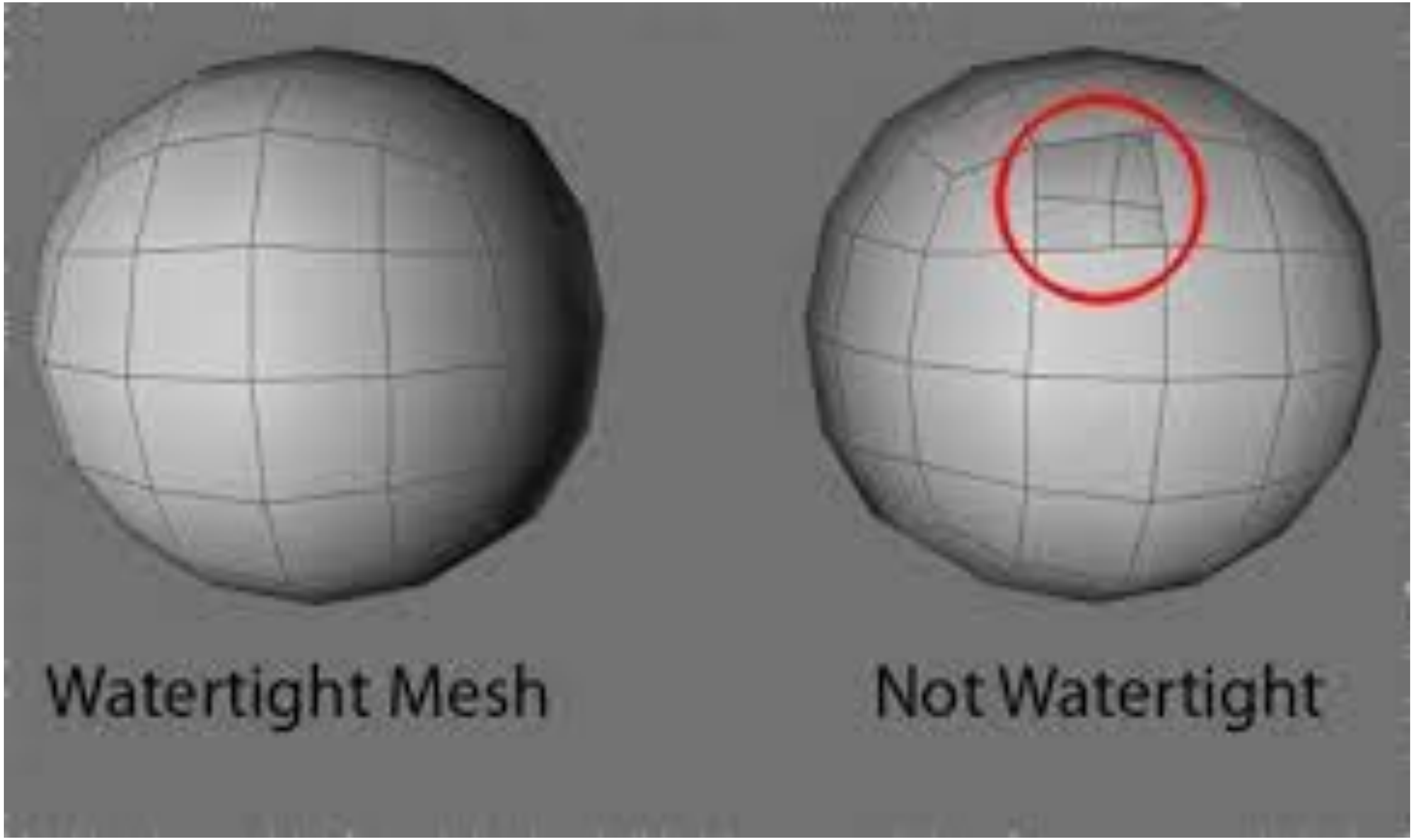
A *2-manifold* is a surface for which the neighbourhood around any point can be flattened onto the plane



Manifold Non-Manifold

Watertight

Watertight meshes have no holes

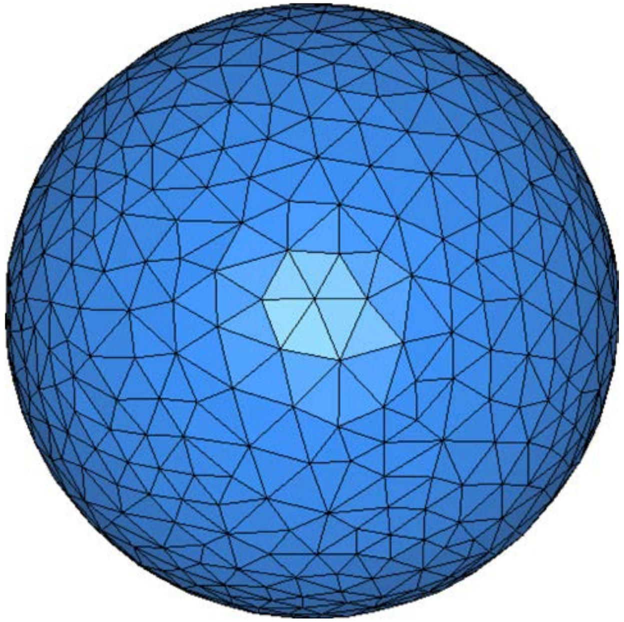


Geometry

Geometrically, a mesh is a piecewise planar surface

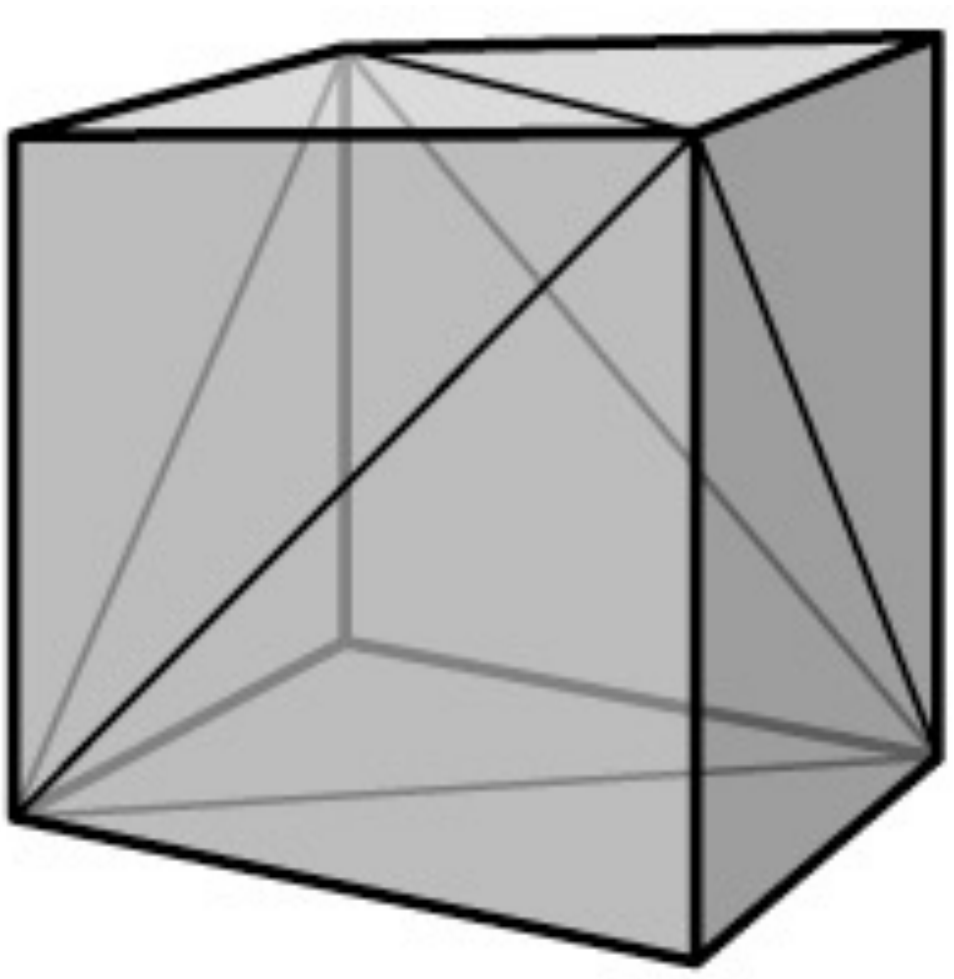
- almost everywhere, it is planar
- exceptions are at the edges where triangles join

Often, it's a piecewise planar approximation of a smooth surface



Rineau & Yvinec CGAL manual

Examples of Meshes



12 triangles, 8 vertices

Examples of Meshes

10 million triangles from a high-resolution 3D scan

Traditional Thai sculpture—scan by XYZRGB, inc., image by MeshLab project


About a trillion triangles from automatically
processed satellite and aerial photography.
Google earth

Storing Triangle Meshes

What do we care about ?

Storing Triangle Meshes

What do we care about ?

1. Compactness
2. Efficiency of queries
    - all vertices of a triangle
    - all triangles around a vertex
    - neighboring triangles of a triangles

Data Structures for Triangle Meshes

Separate Triangles

Indexed Triangle Set

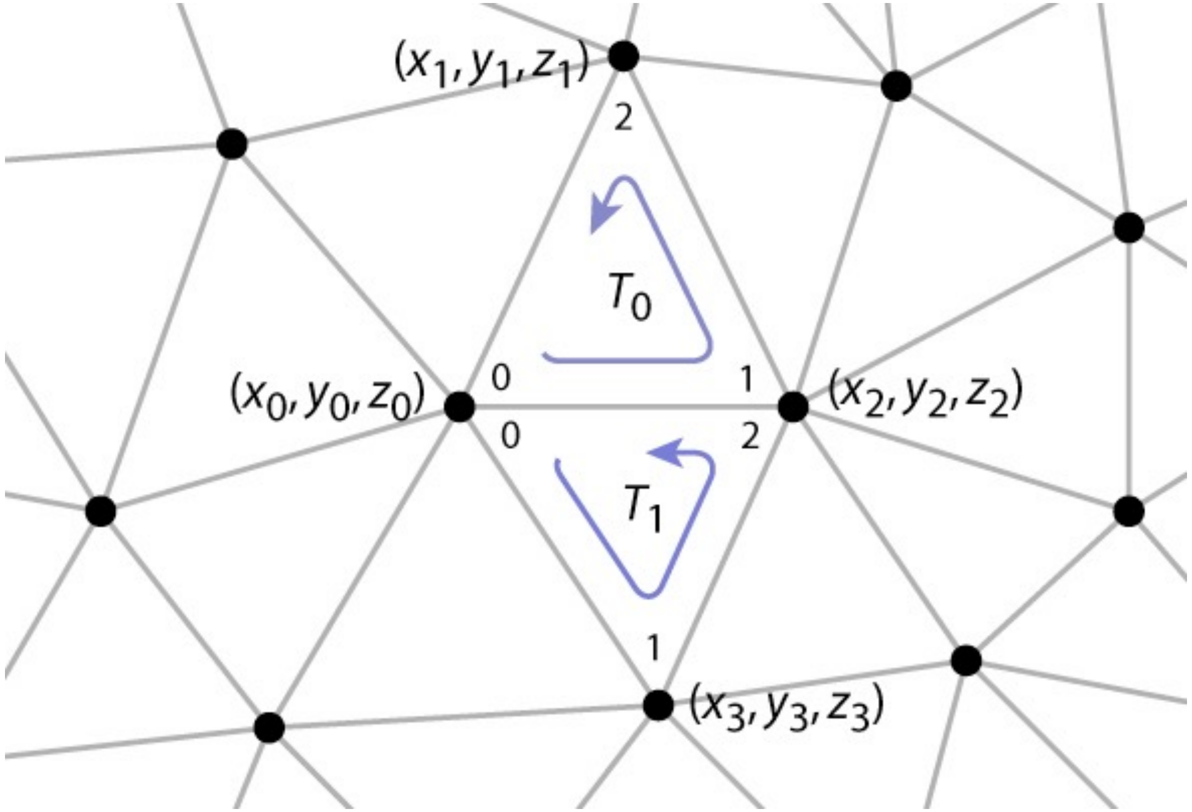
Triangle-Neighbour Data Structure

Winged-Edge Data Structure

Half-Edge Data Structure

Separate triangles

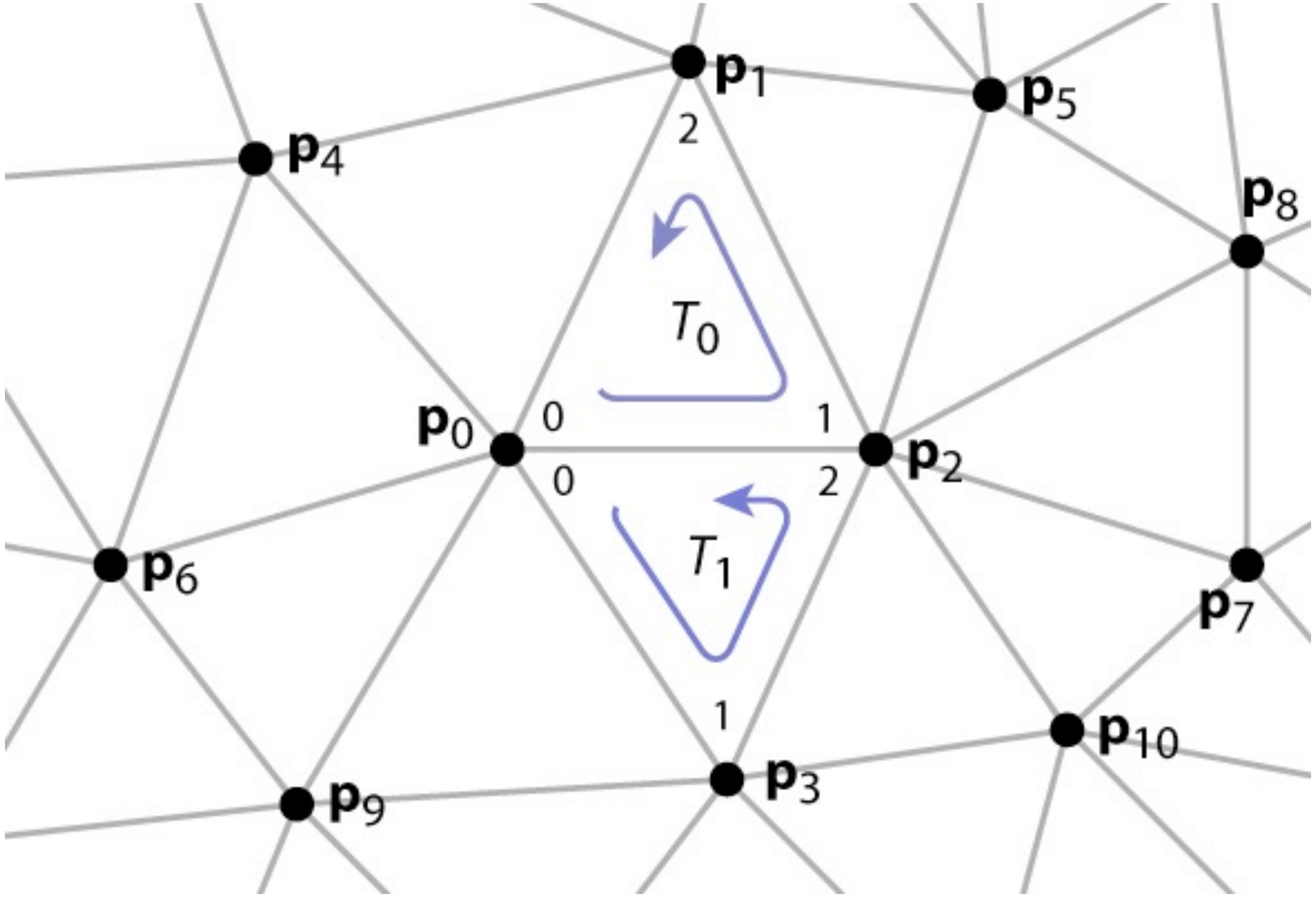
	[0]	[1]	[2]
tris[0]	x_0, y_0, z_0	x_2, y_2, z_2	x_1, y_1, z_1
tris[1]	x_0, y_0, z_0	x_3, y_3, z_3	x_2, y_2, z_2
⋮	⋮	⋮	⋮



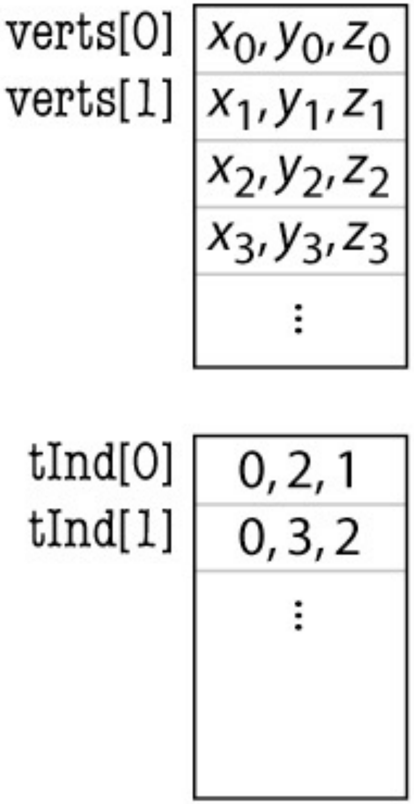
Indexed triangle set

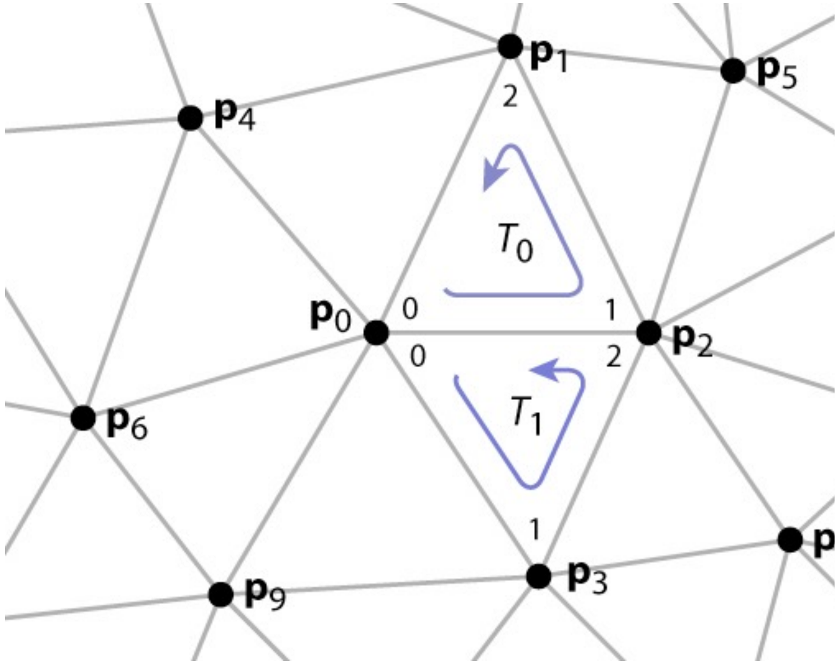
verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
	$\vdots$

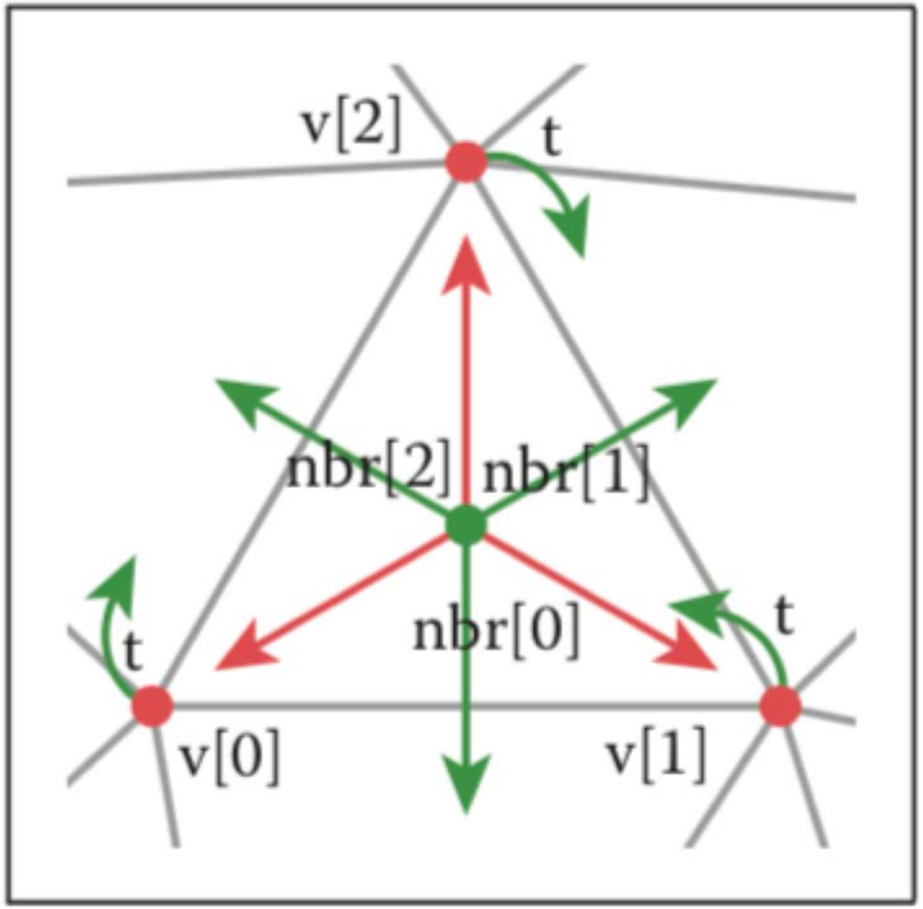
tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
	$\vdots$



Triangle-Neighbour Data Structure







Triangle-Neighbour Data Structure

vTri

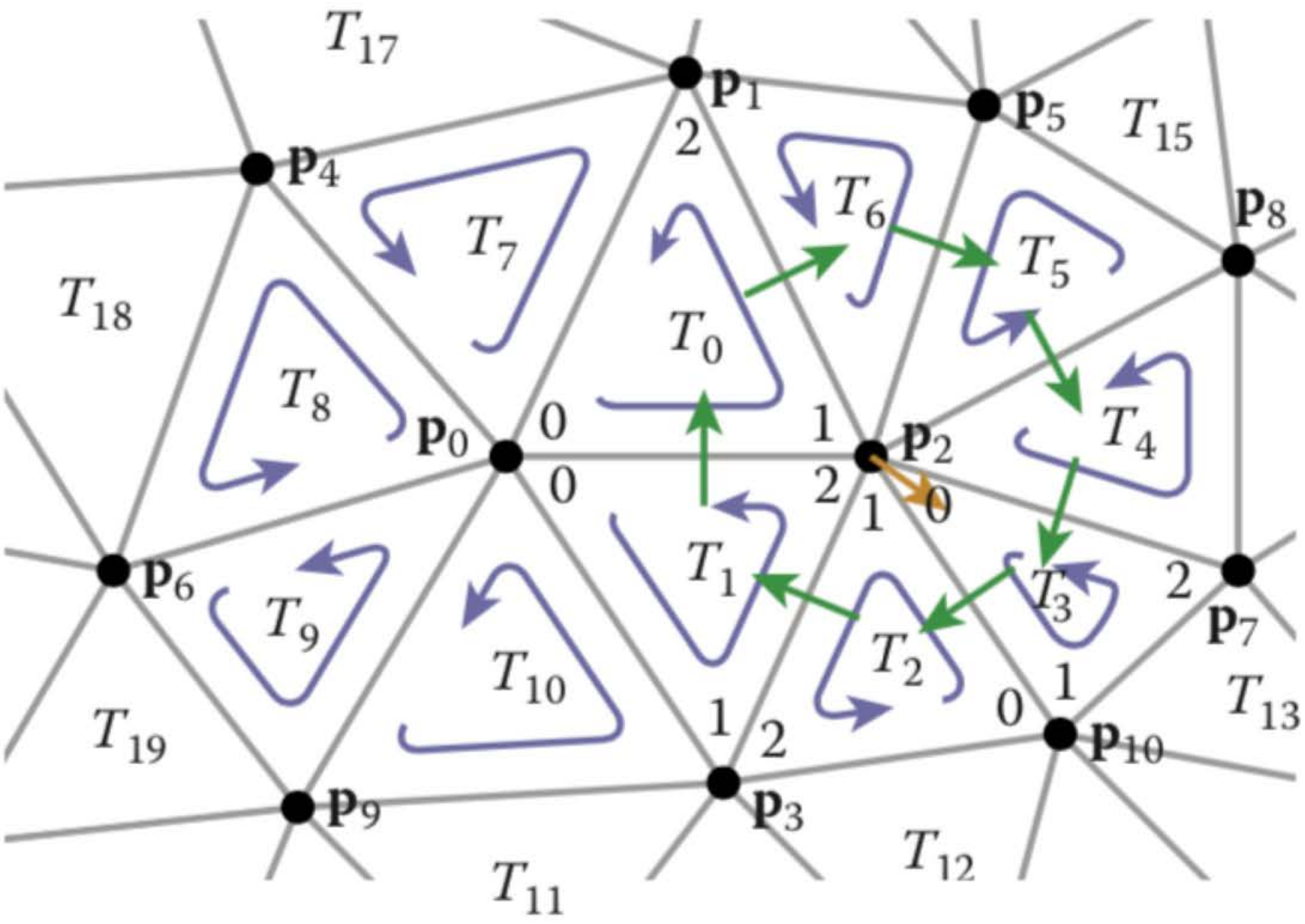
[0]	0
[1]	6
[2]	3
[3]	1
	⋮

tNbr

[0]	1, 6, 7
[1]	10, 2, 0
[2]	3, 1, 12
[3]	2, 13, 4
	⋮

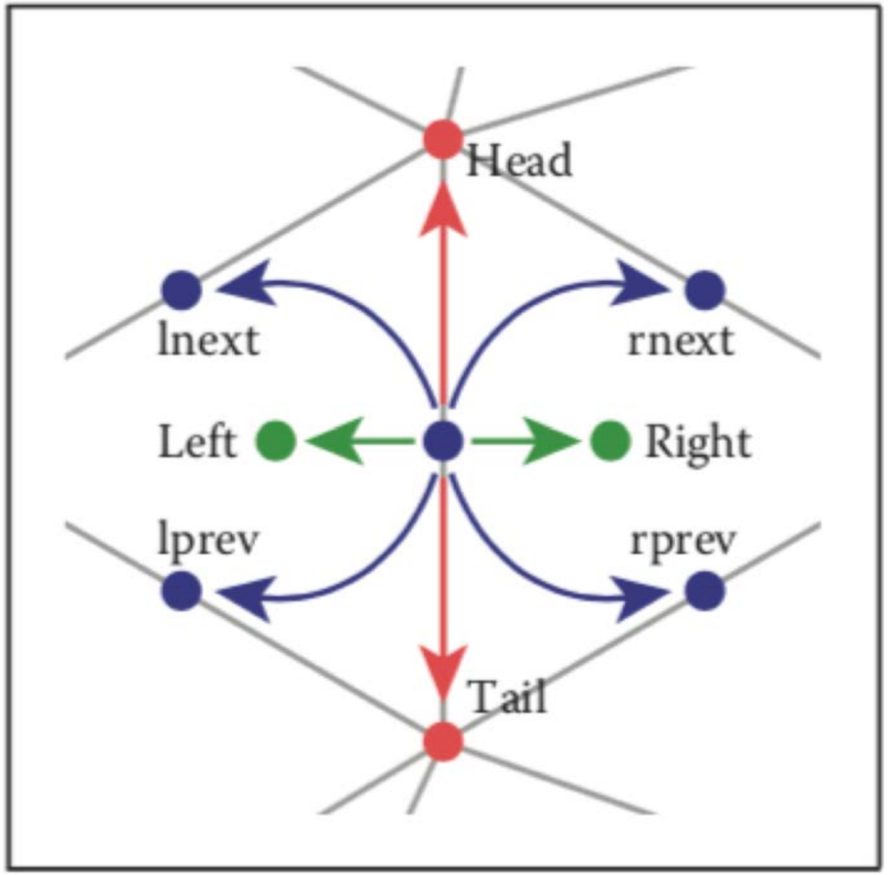
tInd

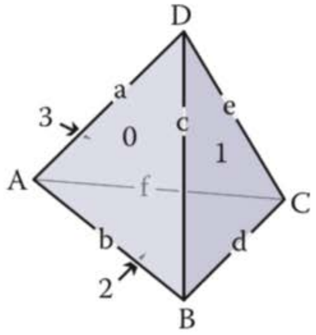
[0]	0, 2, 1
[1]	0, 3, 2
[2]	10, 2, 3
[3]	2,10,7
	⋮



The kth entry of tNbr points to the neighboring triangle that shares vertices k and k + 1

Winged-Edge Data Structure



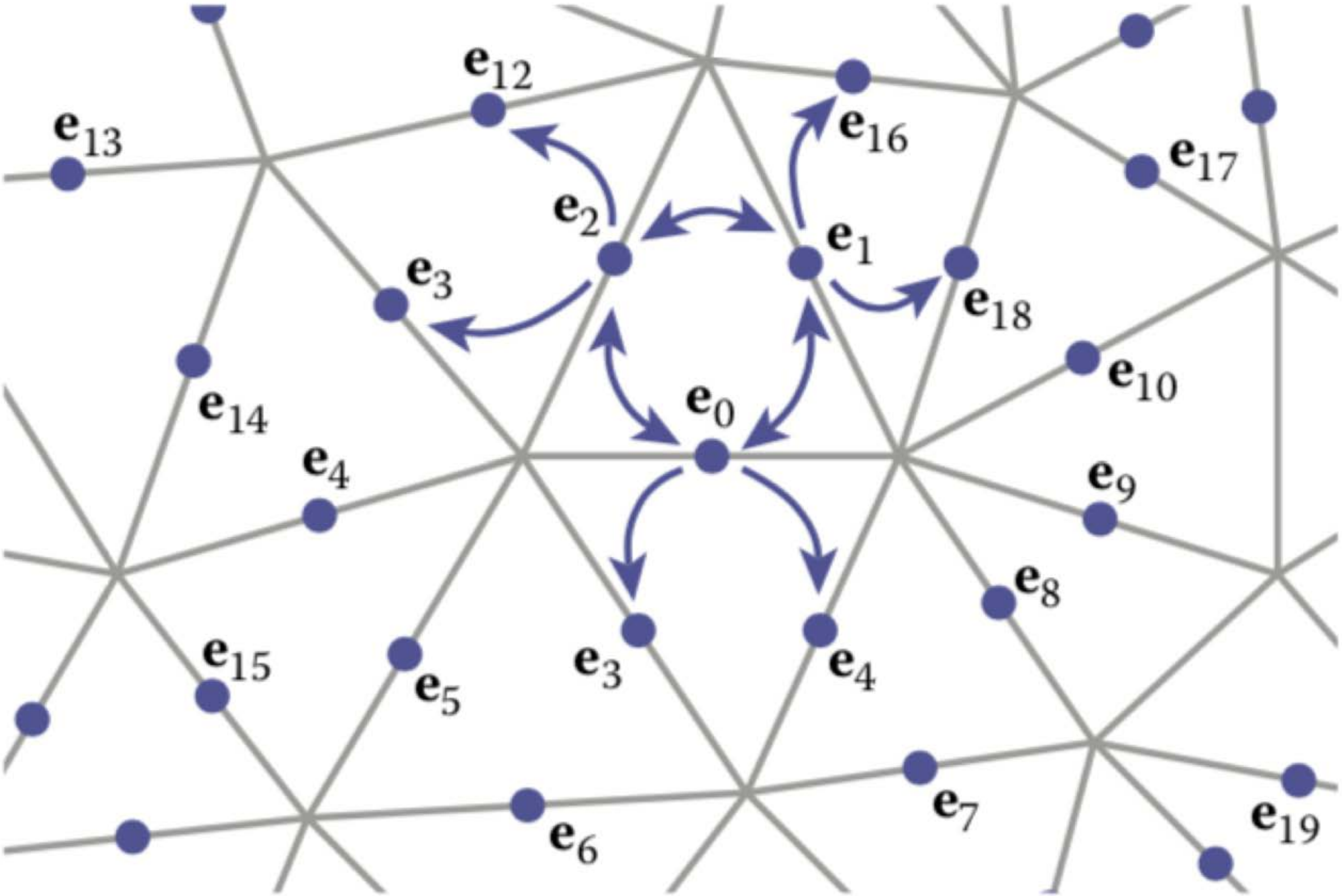


Edge	Vertex 1	Vertex 2	Face left	Face right	Pred left	Succ left	Pred right	Succ right
a	A	D	3	0	f	e	c	b
b	A	B	0	2	a	c	d	f
c	B	D	0	1	b	a	e	d
d	B	C	1	2	c	e	f	b
e	C	D	1	3	d	c	a	f
f	C	A	3	2	e	e	b	d

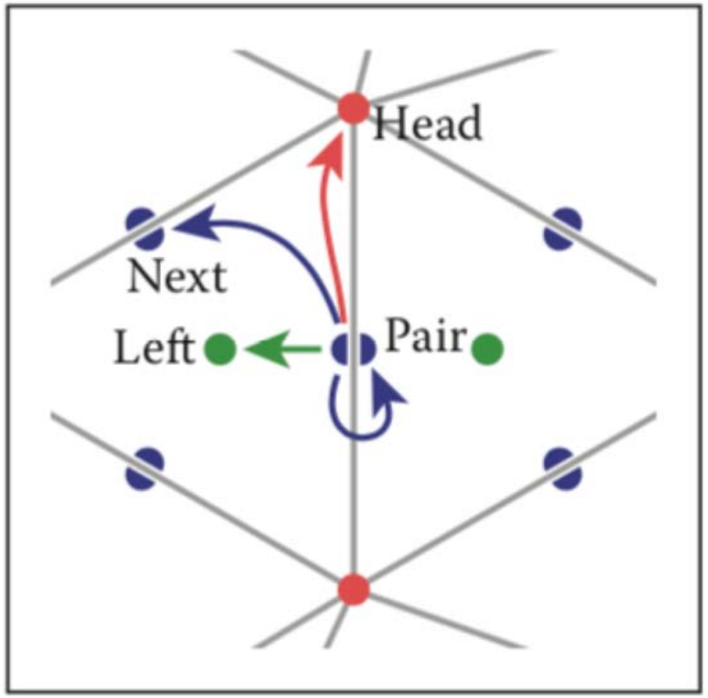
Vertex	Edge
A	a
B	d
C	d
D	e

Face	Edge
0	a
1	c
2	d
3	a

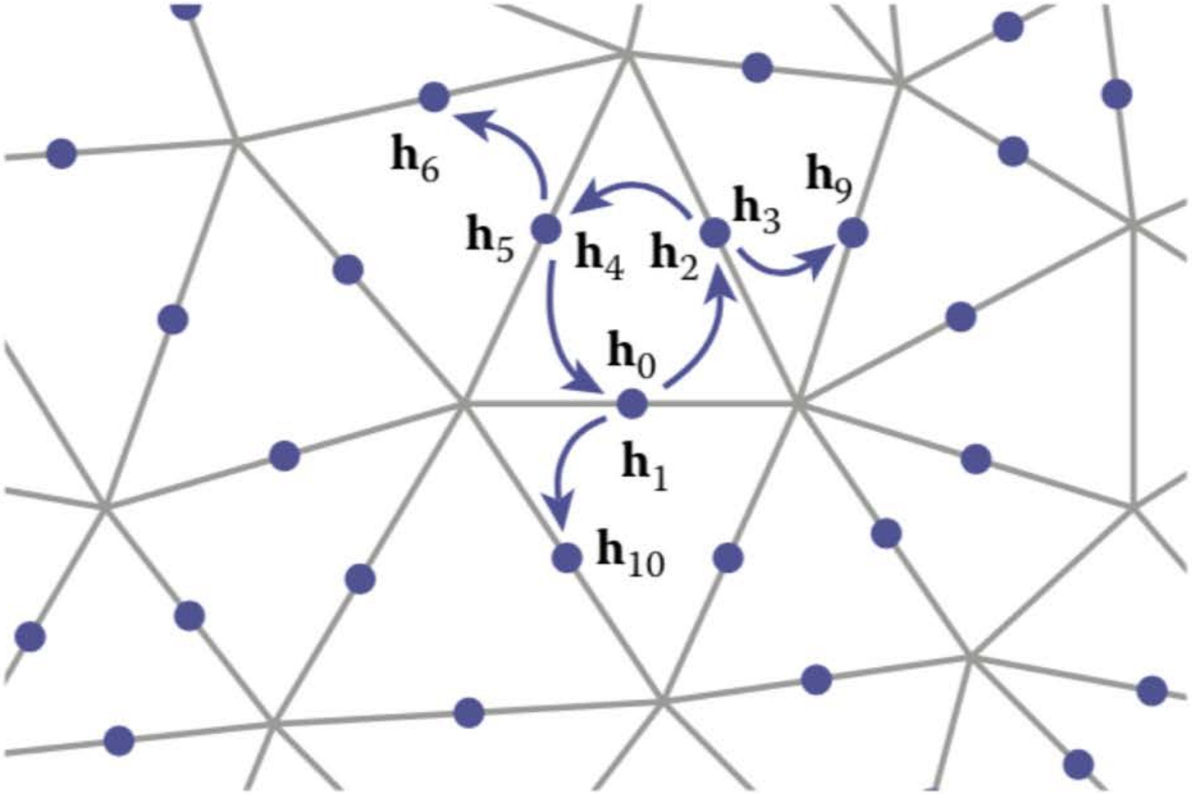
Winged-Edge Data Structure



Half-Edge Data Structure

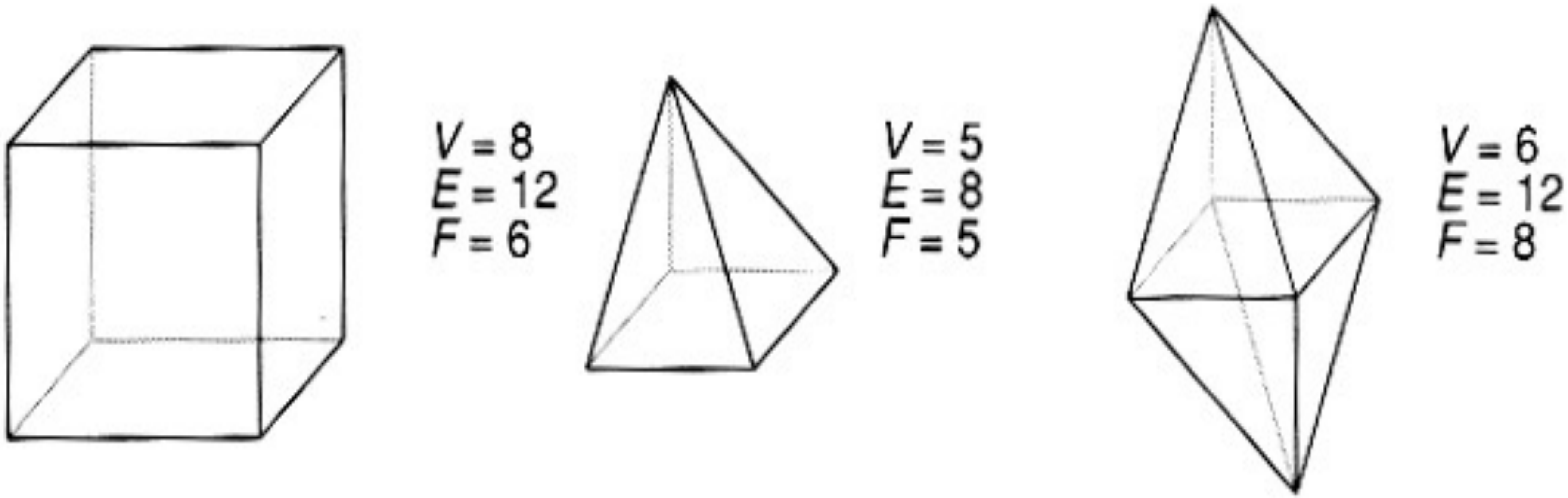


	Pair	Next
hedge[0]	1	2
hedge[1]	0	10
hedge[2]	3	4
hedge[3]	2	9
hedge[4]	5	0
hedge[5]	4	6
	⋮	



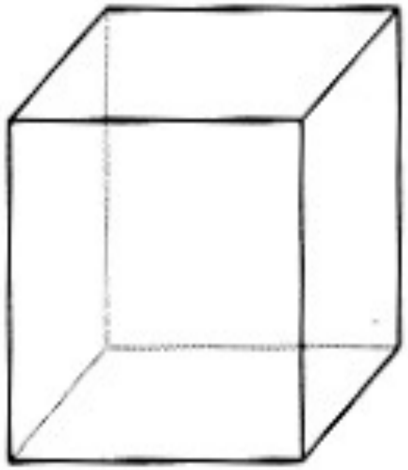
Relationships between primitive Types

What is the relationship between the number of vertices, the number of edges and the number of triangles in a mesh ?

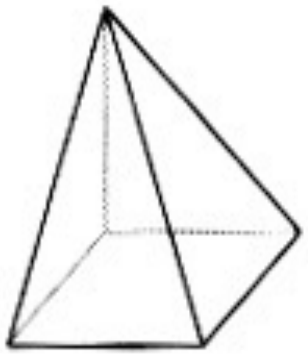


Relationships between primitive Types

Euler's Formula: $V + F - E = 2$
(closed triangle mesh)



$V = 8$
$E = 12$
$F = 6$



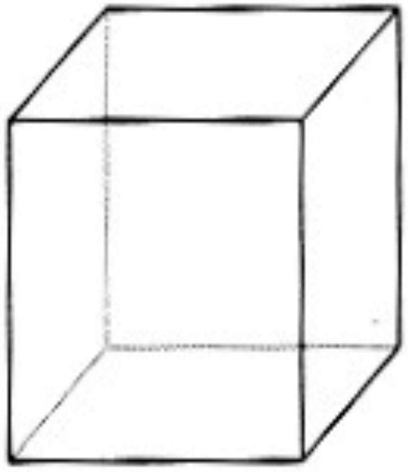
$V = 5$
$E = 8$
$F = 5$

$V = 6$
$E = 12$
$F = 8$

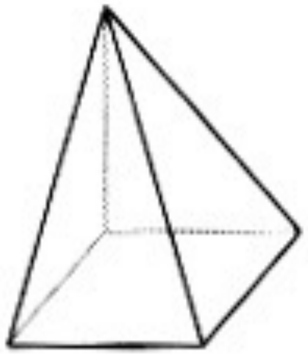
Relationships between primitive Types

$$3F = 2E$$

(closed triangle mesh)



$V = 8$
$E = 12$
$F = 6$

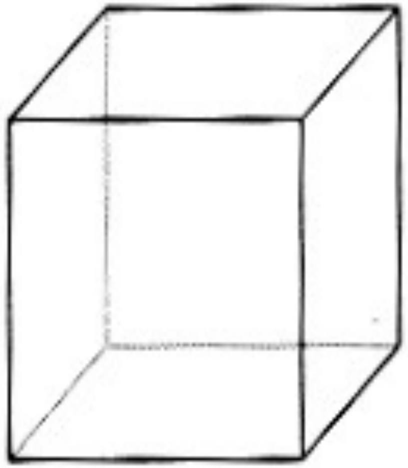


$V = 5$
$E = 8$
$F = 5$

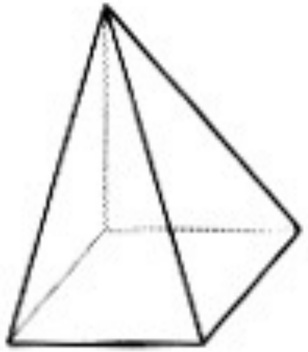
$V = 6$
$E = 12$
$F = 8$

Relationships between primitive Types

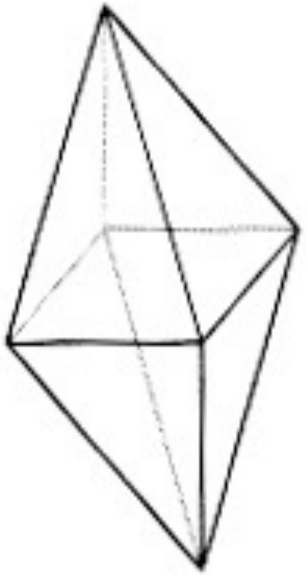
$$\text{number of half edges} = 2E$$



$V = 8$
$E = 12$
$F = 6$



$V = 5$
$E = 8$
$F = 5$

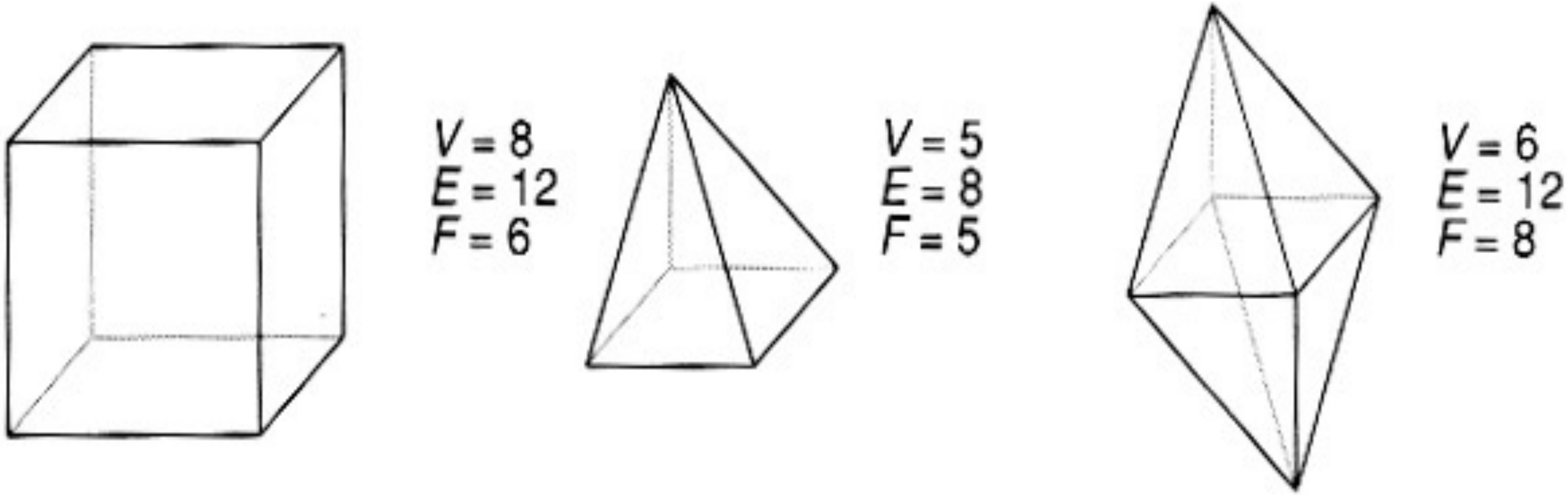


$V = 6$
$E = 12$
$F = 8$

Relationships between primitive Types

$$\text{number of half edges} = 2E$$

$$\text{number of half edges} = 3F$$



Data on meshes

Often need to store additional information besides just the geometry

Can store additional data at faces, vertices, or edges

Examples

- colours stored on faces, for faceted objects
- information about sharp creases stored at edges
- any quantity that varies continuously (without sudden changes, or discontinuities) gets stored at vertices

Key types of vertex data

Positions

- at some level this is just another piece of data
- position varies continuously between vertices

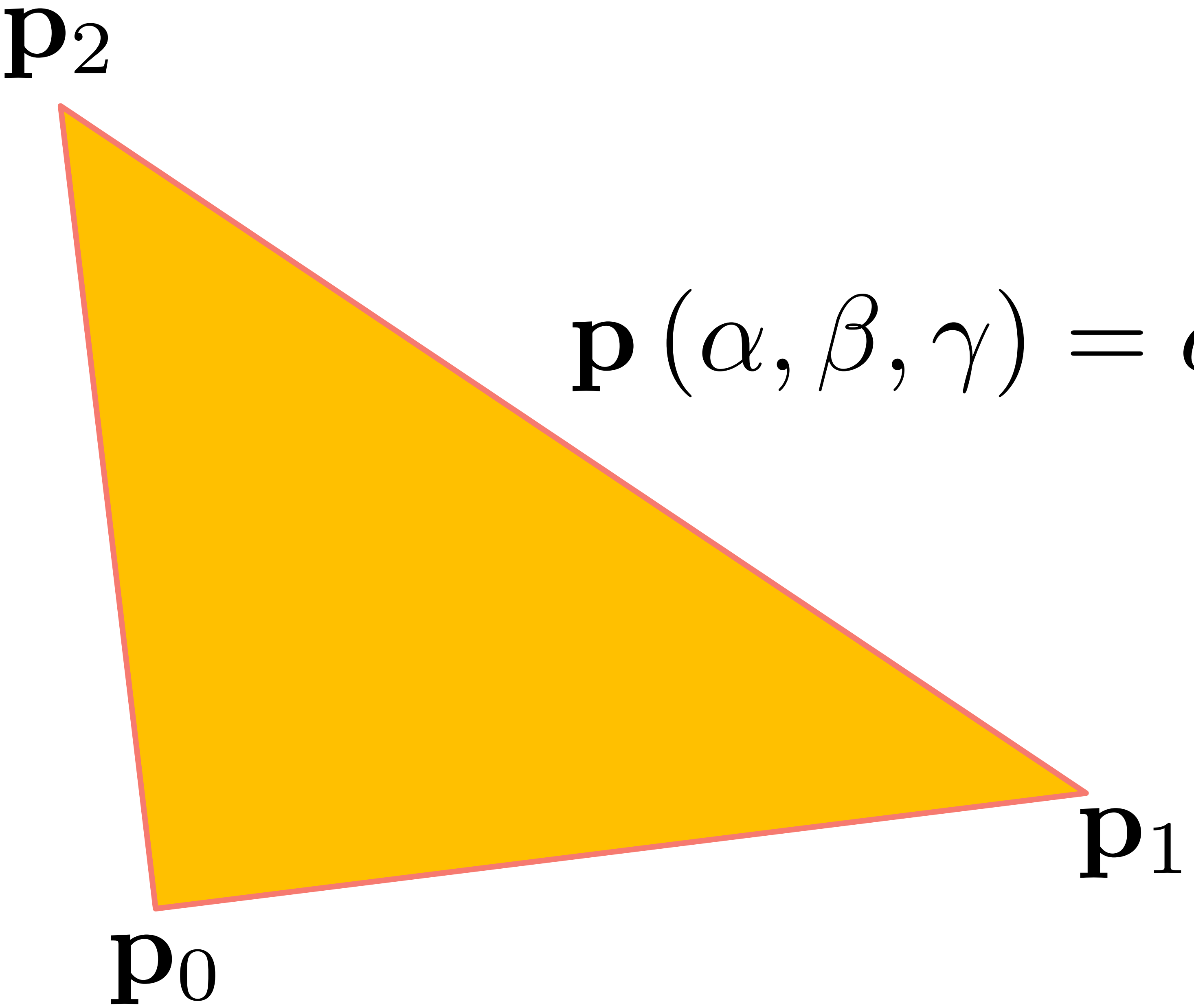
Surface normals

- when a mesh is approximating a curved surface, store normals at vertices

Texture coordinates

- 2D coordinates that tell you how to paste images on the surface

Barycentric Coordinates



$$\mathbf{p}\left(\alpha, \beta, \gamma\right) = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_0$$

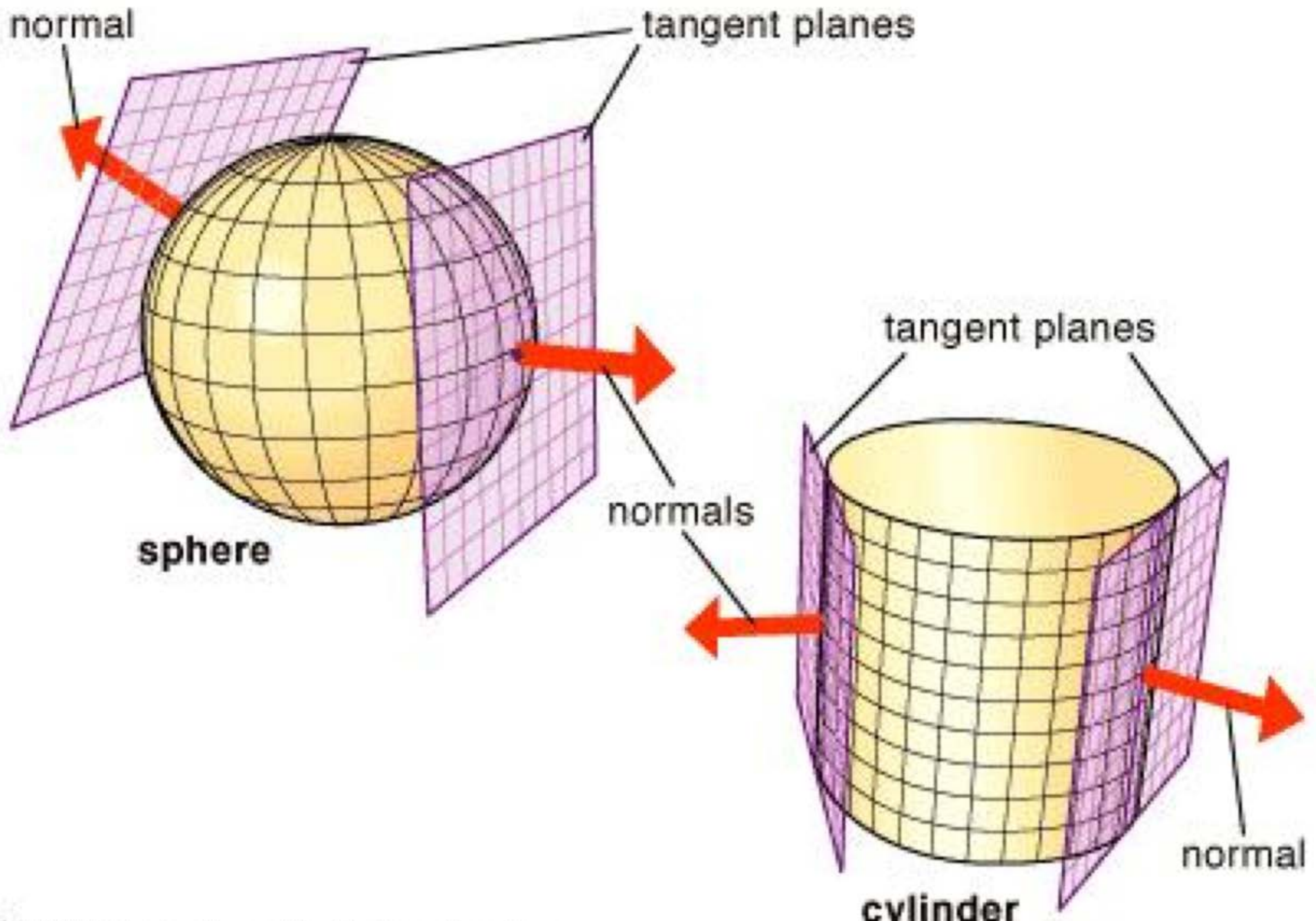
$$\alpha \geqslant 0$$

$$\beta \geqslant 0$$

$$\alpha + \beta \leqslant 1$$

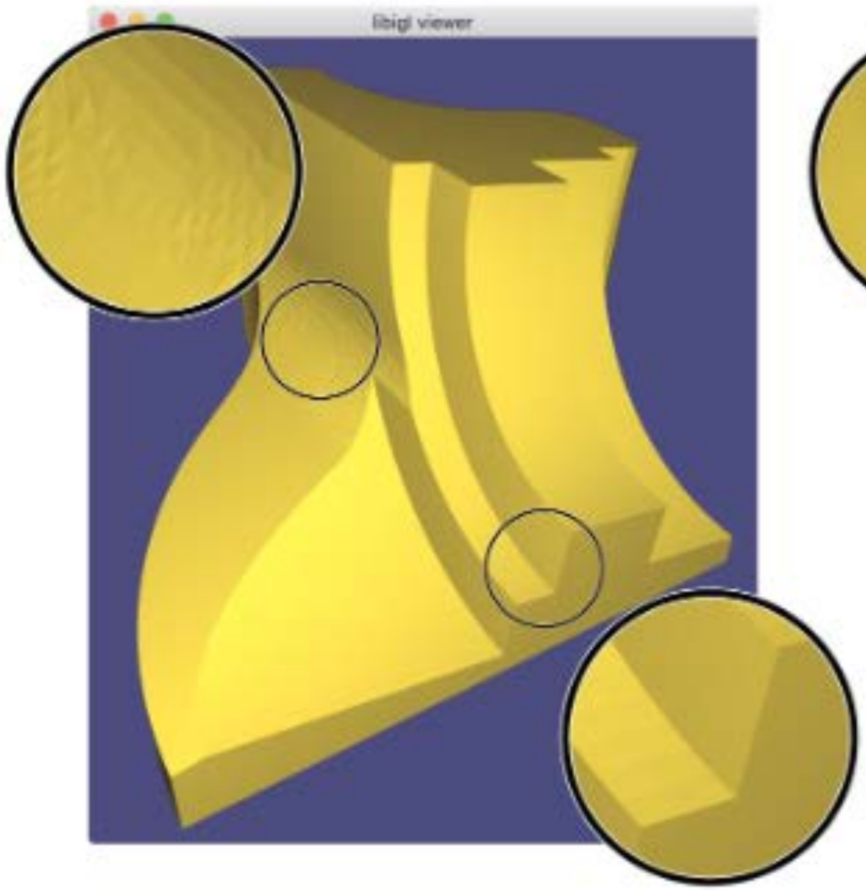
$$\gamma = 1 - \alpha - \beta$$

Normal Vectors

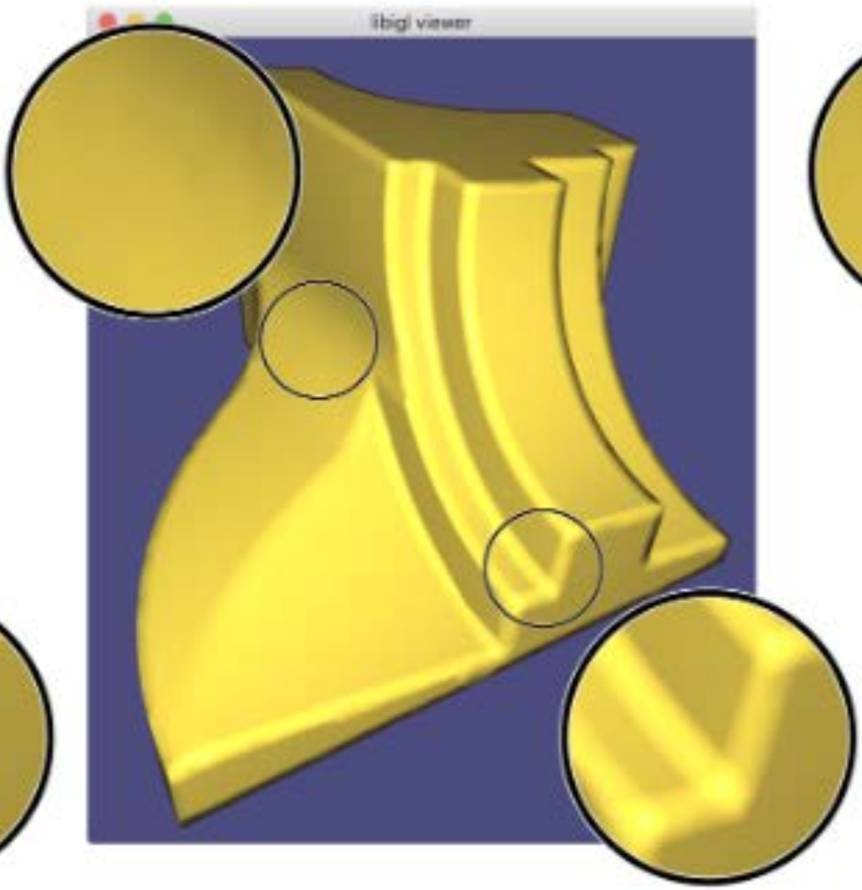


Computing a Per-vertex Normals

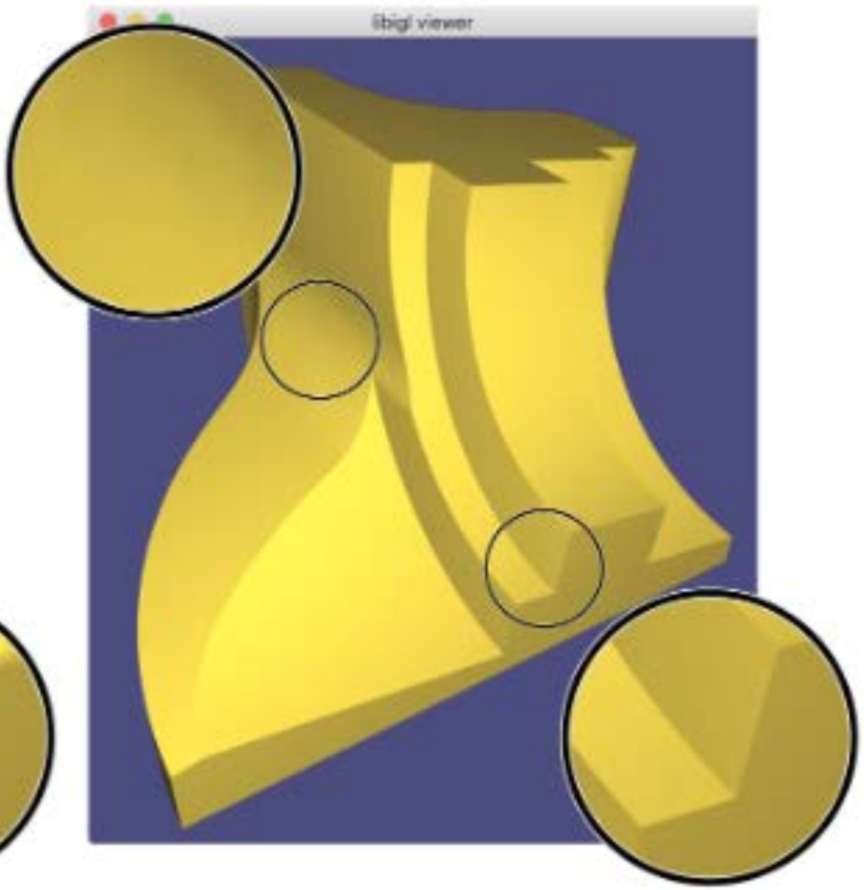
Per-Face Normals



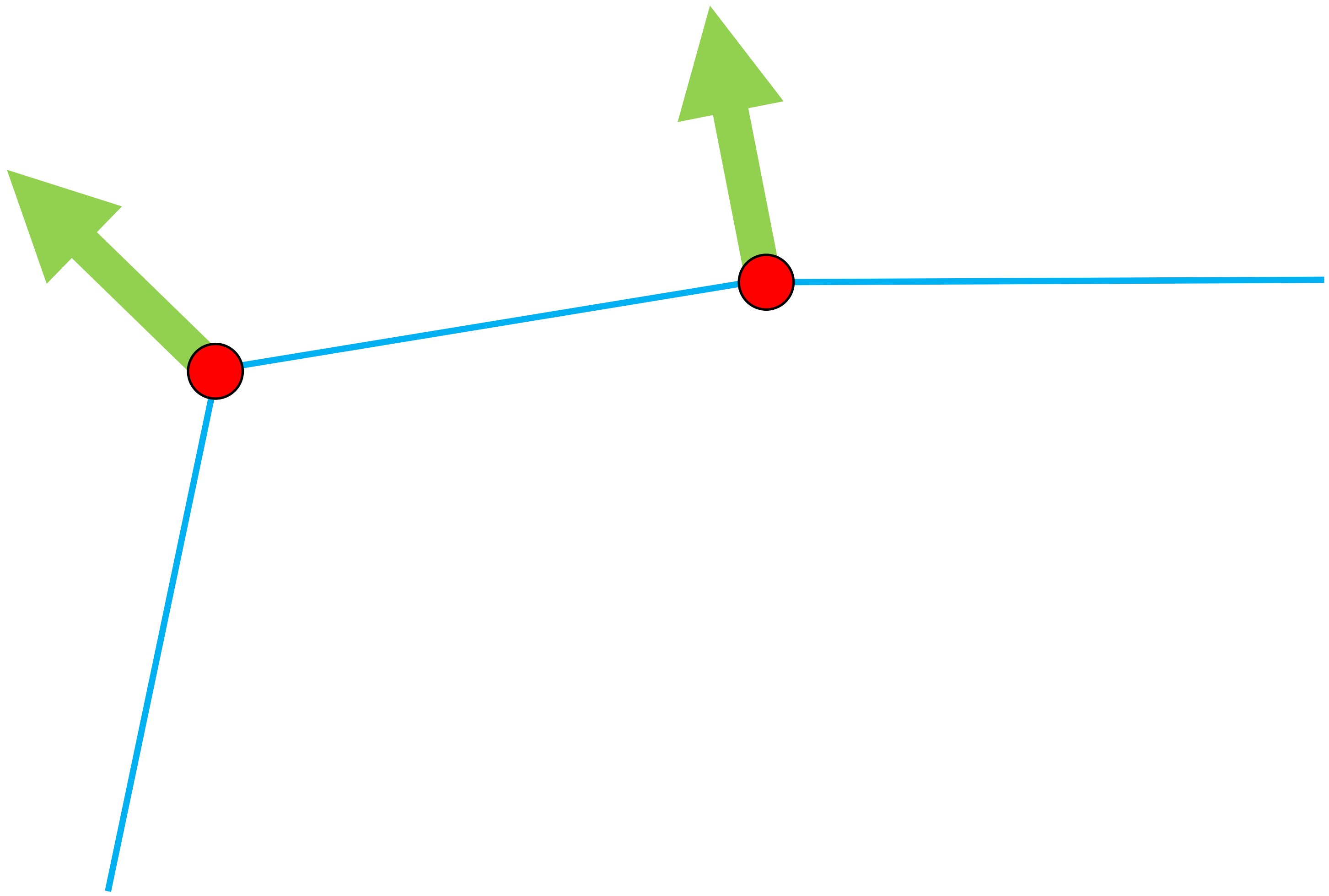
Per-Vertex Normals



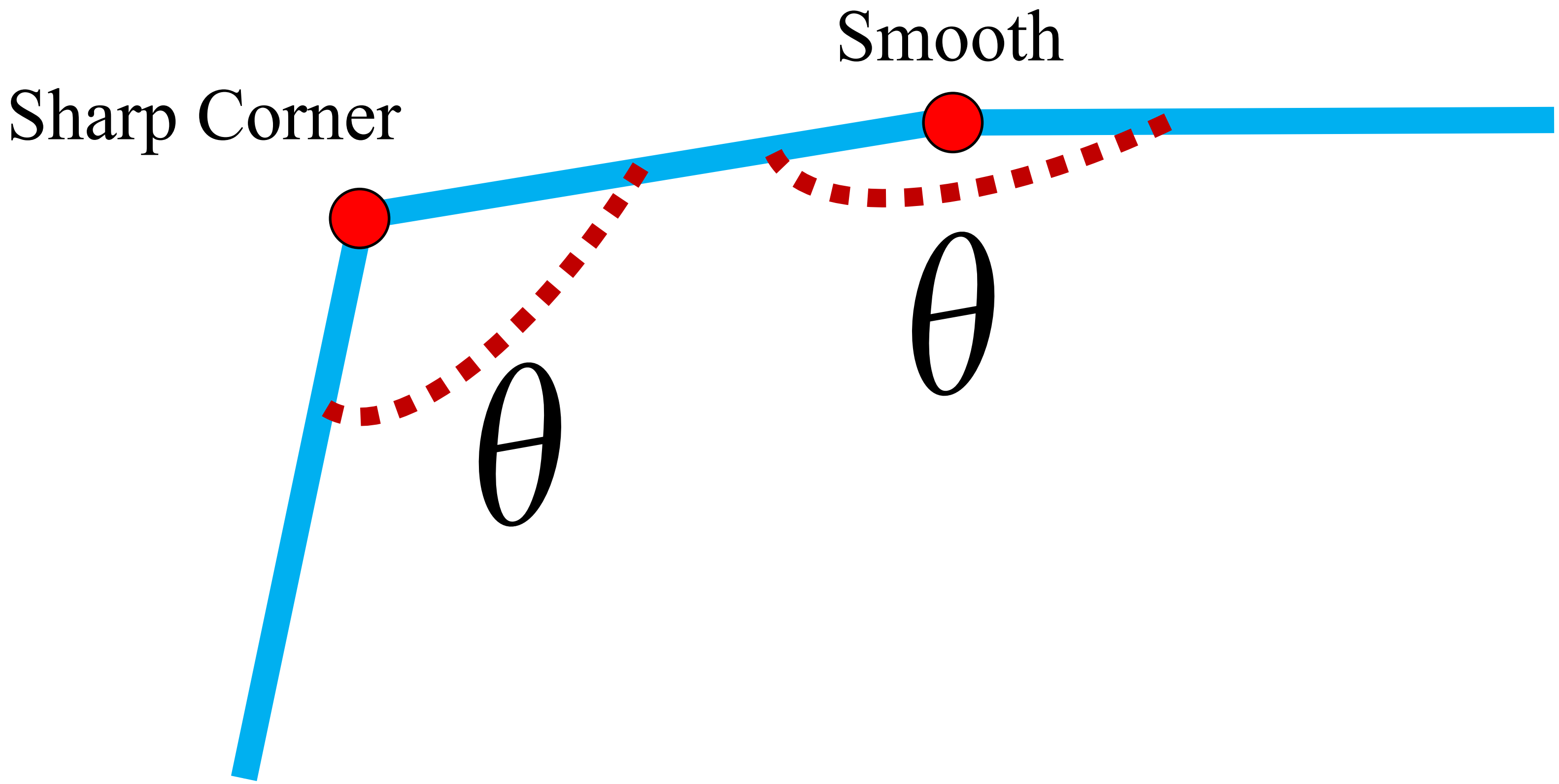
Per-Corner Normals



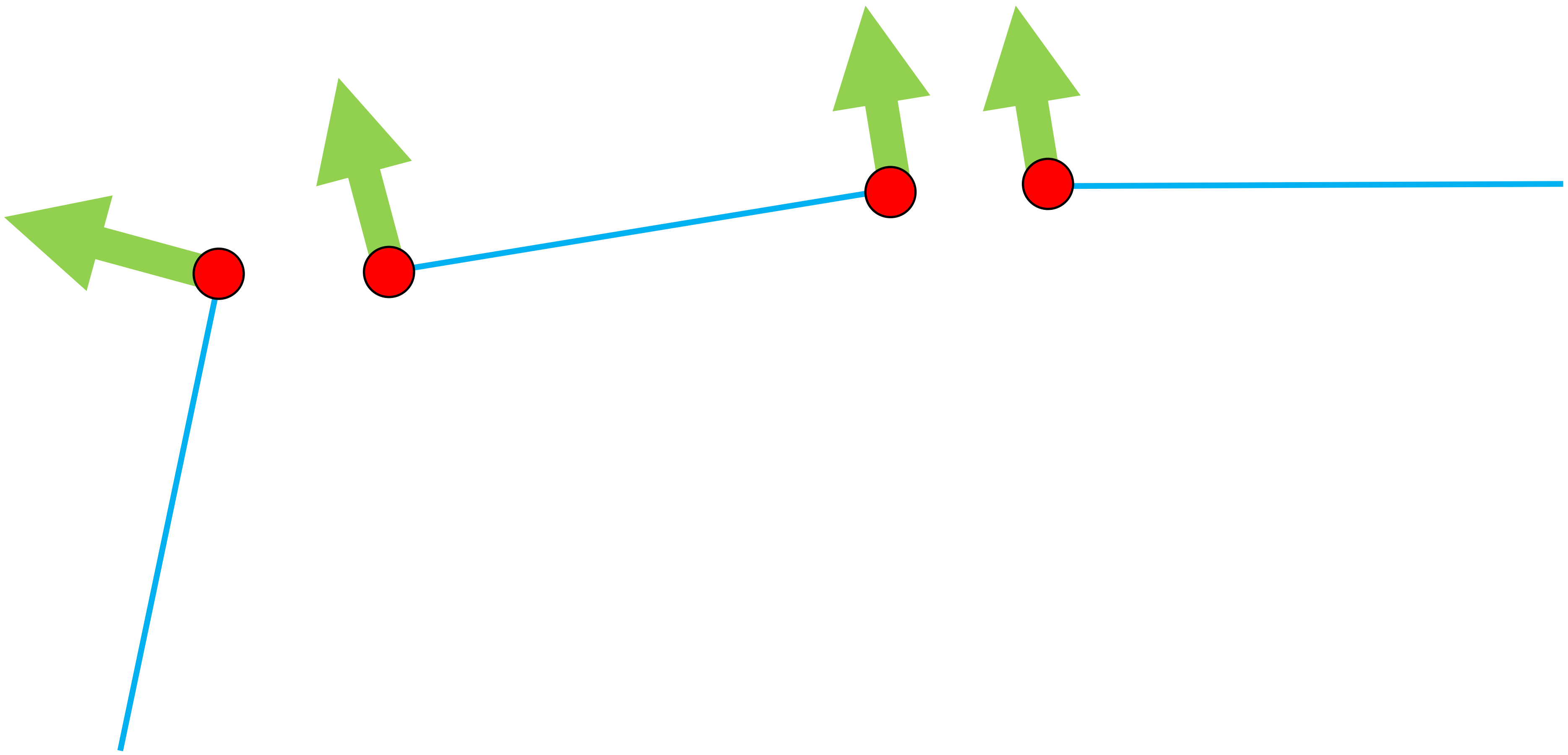
Per-Vertex Normals



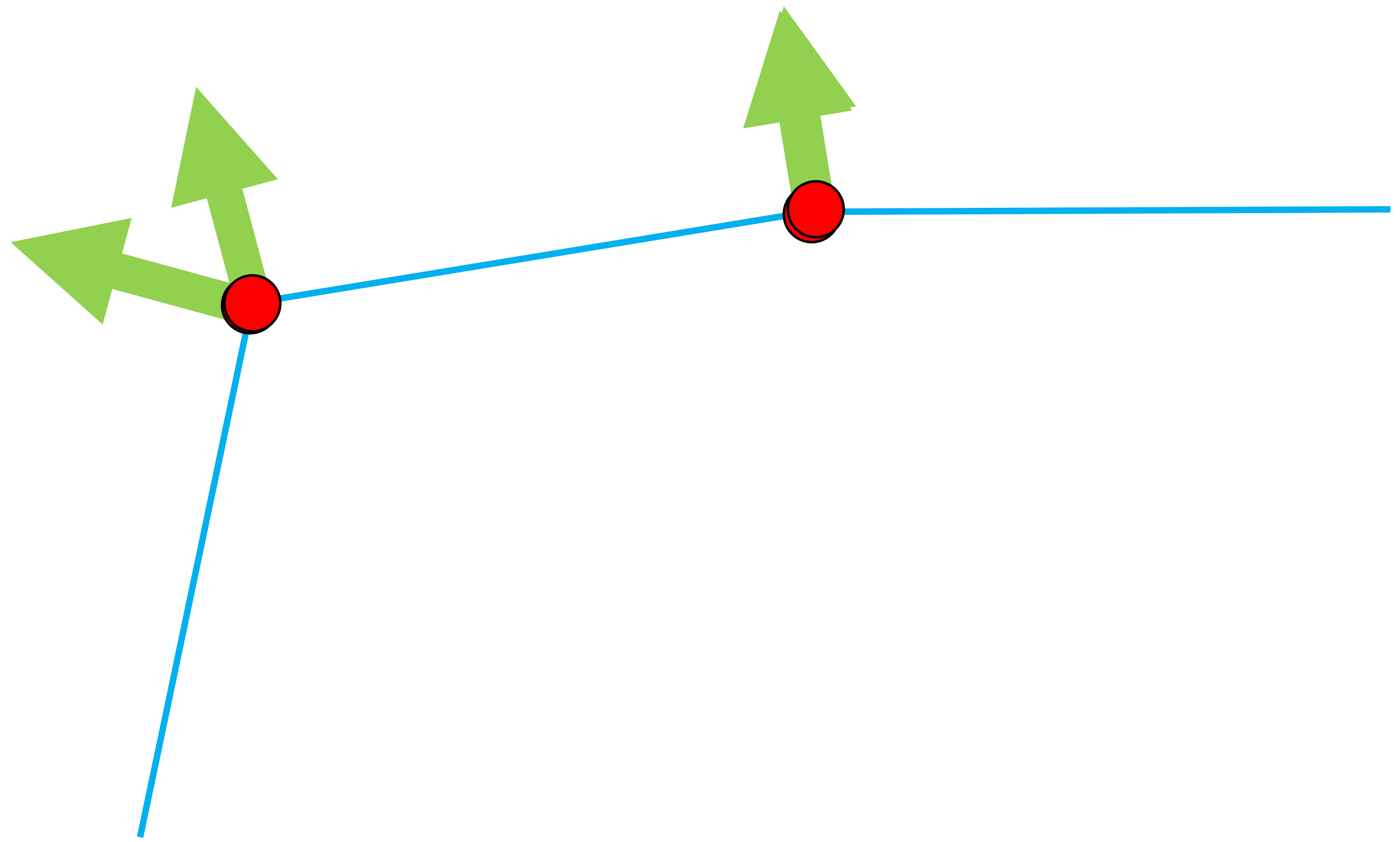
Per-Corner Normals



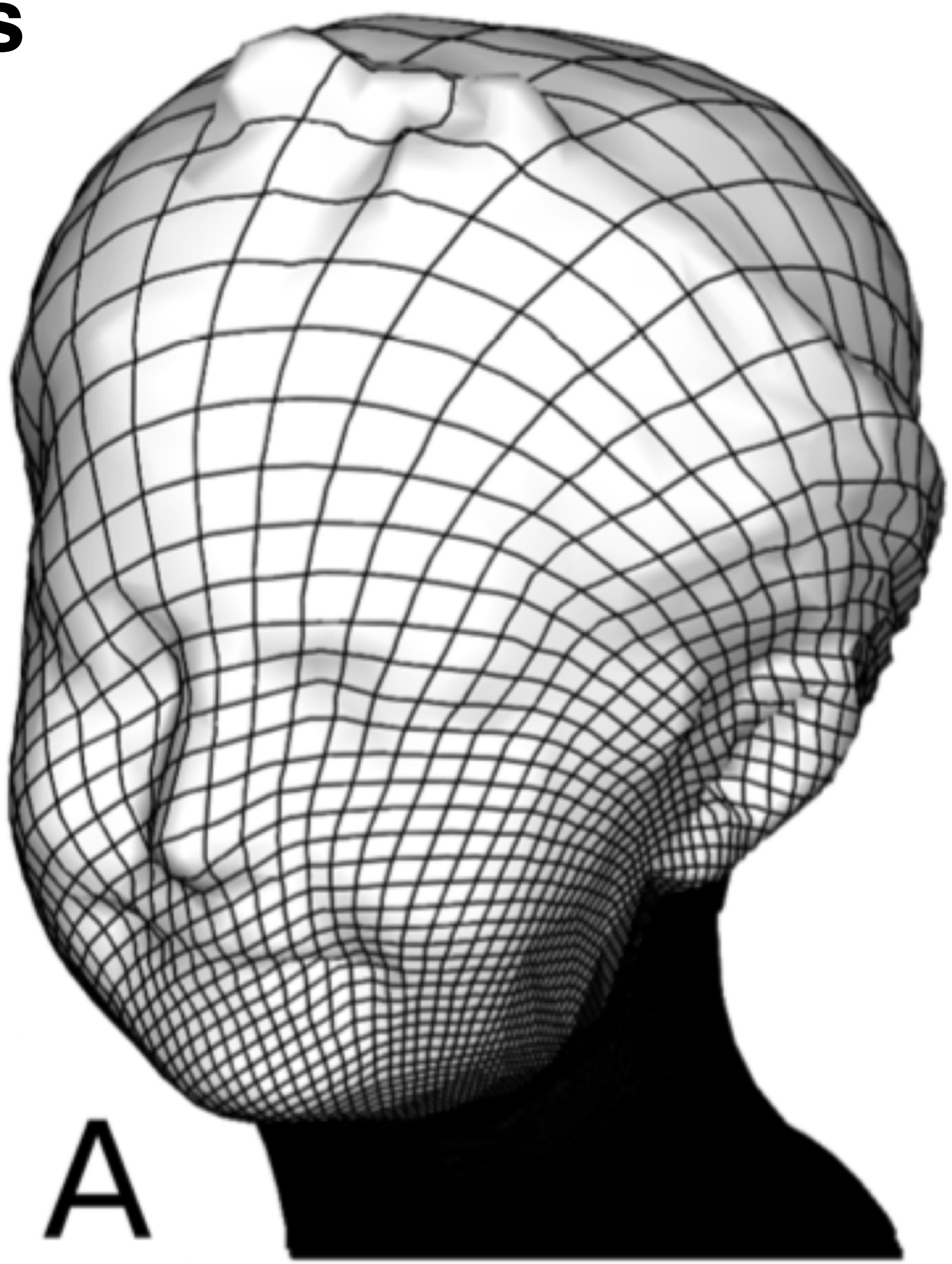
Per-Corner Normals



Per-Corner Normals

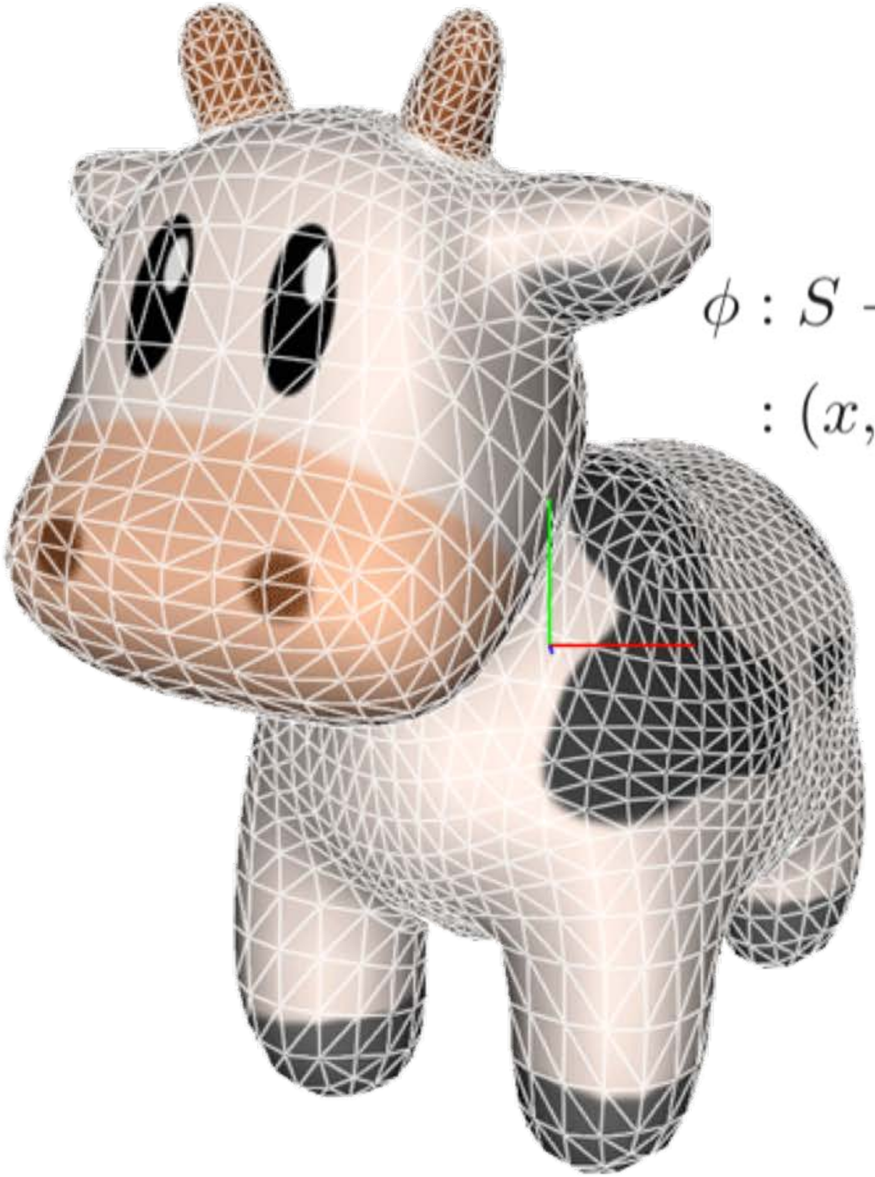


Quadrilateral (Quad) Meshes

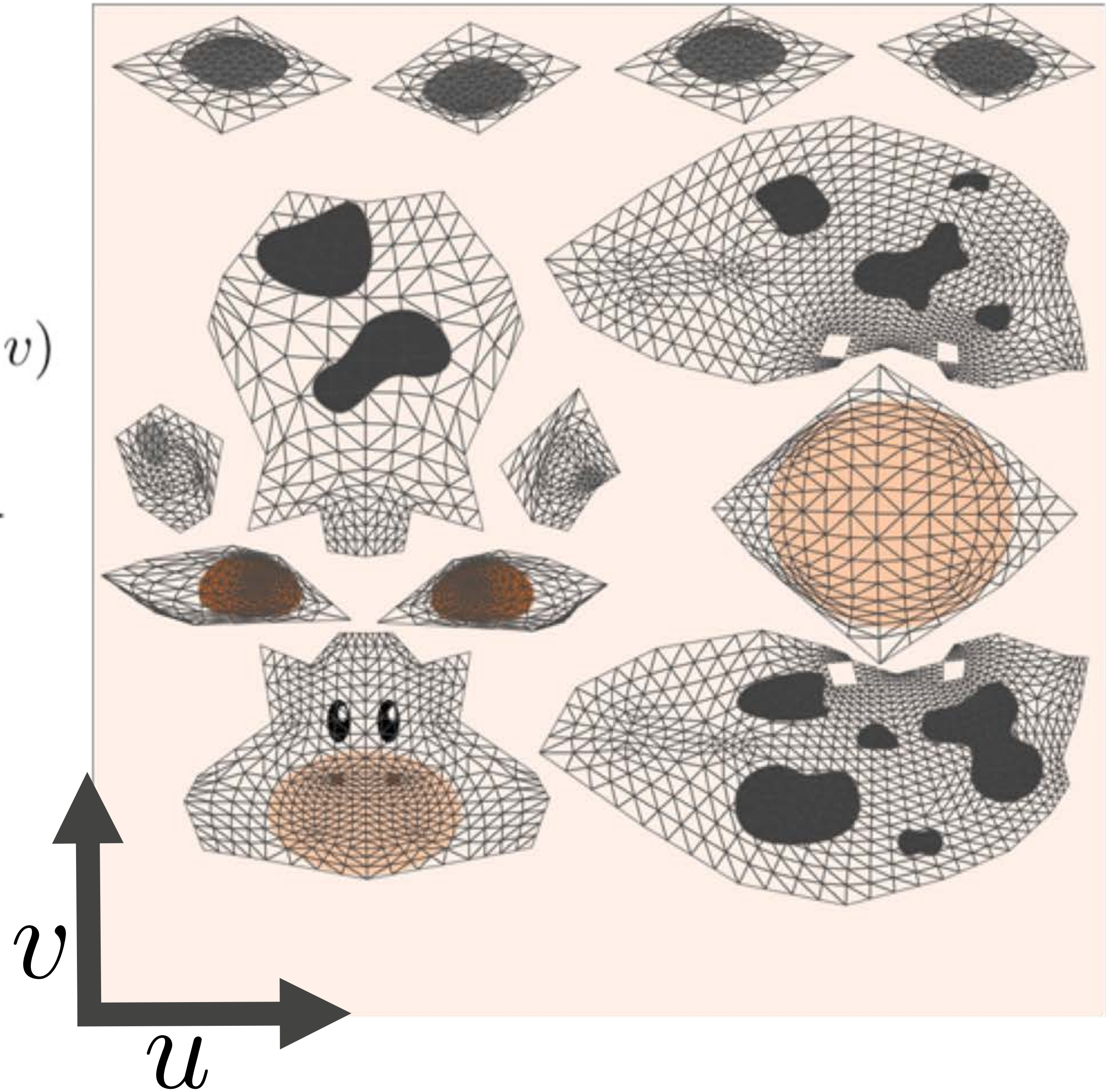


A

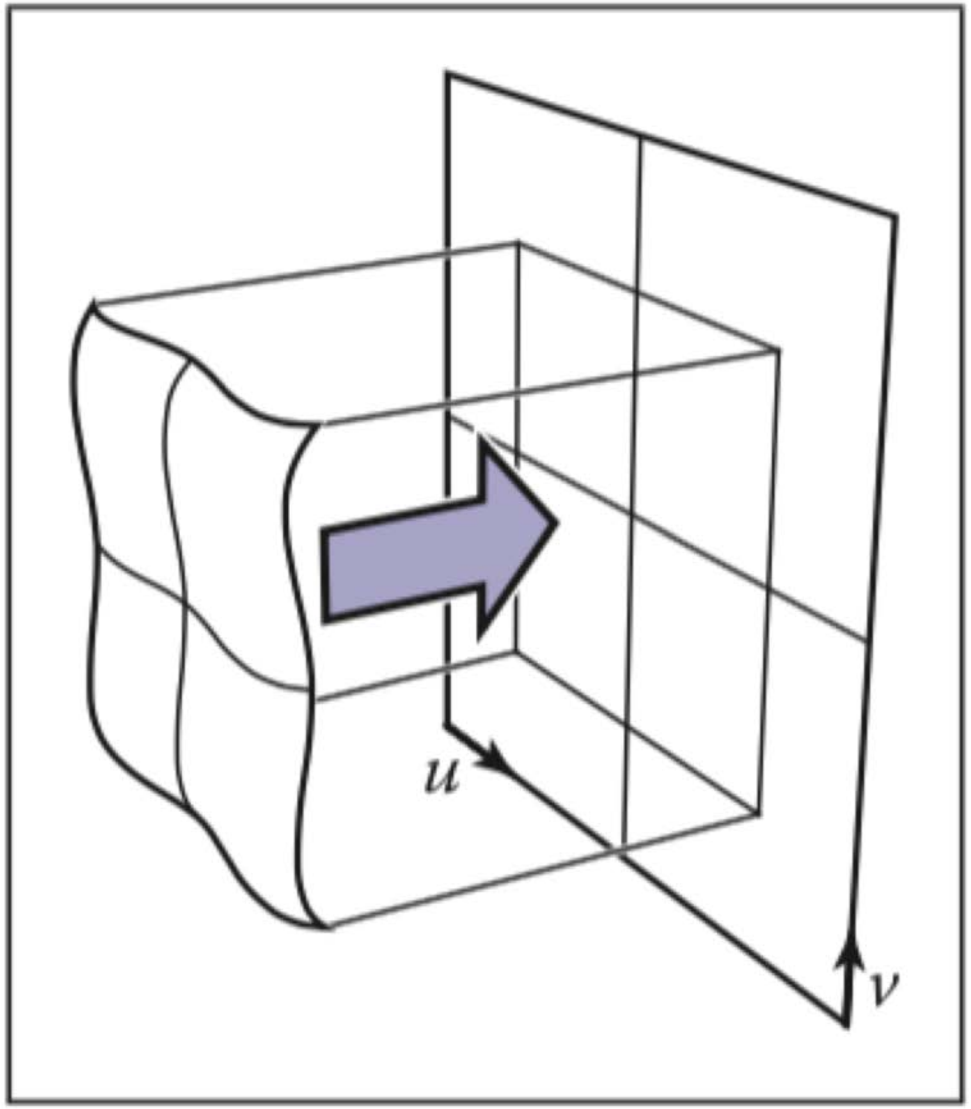
Texture coordinates



$$\phi : S \to T$$
$$: (x, y, z) \mapsto (u, v)$$



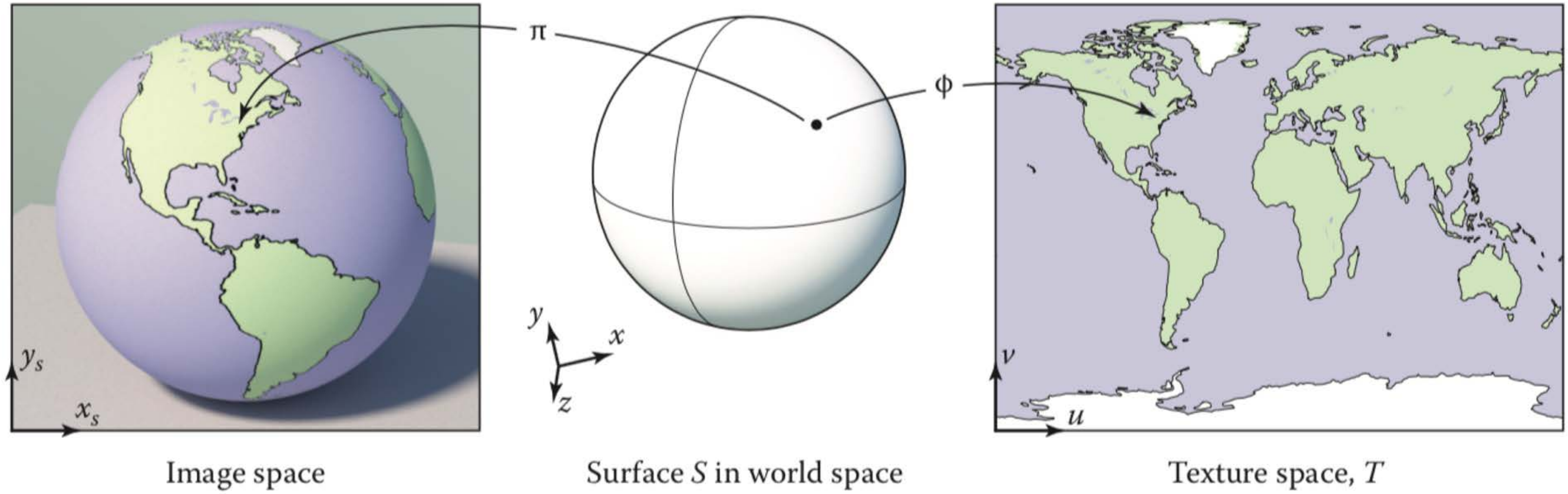
Planar Texture Map



$$\phi(x, y, z) = (u, v)$$

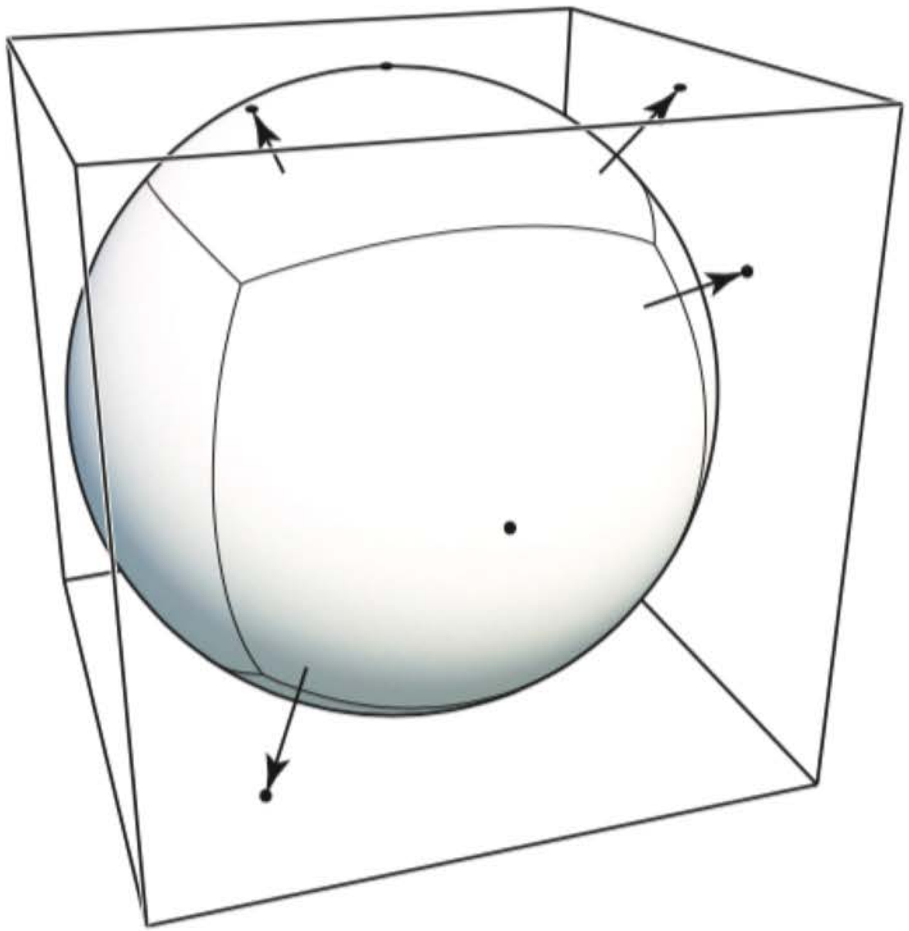
$$\text{where} \quad \begin{bmatrix} u \\ v \\ * \\ 1 \end{bmatrix} = M_t \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

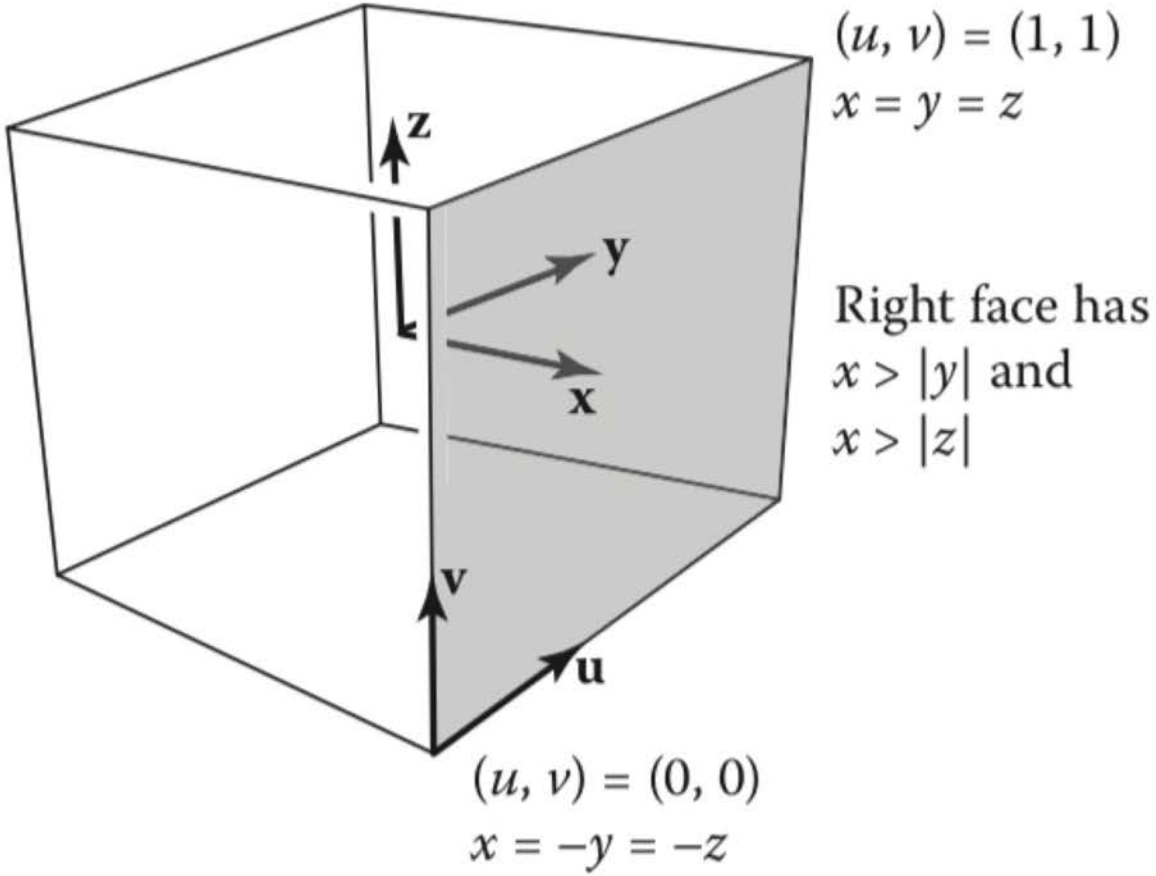
Spherical Texture Map



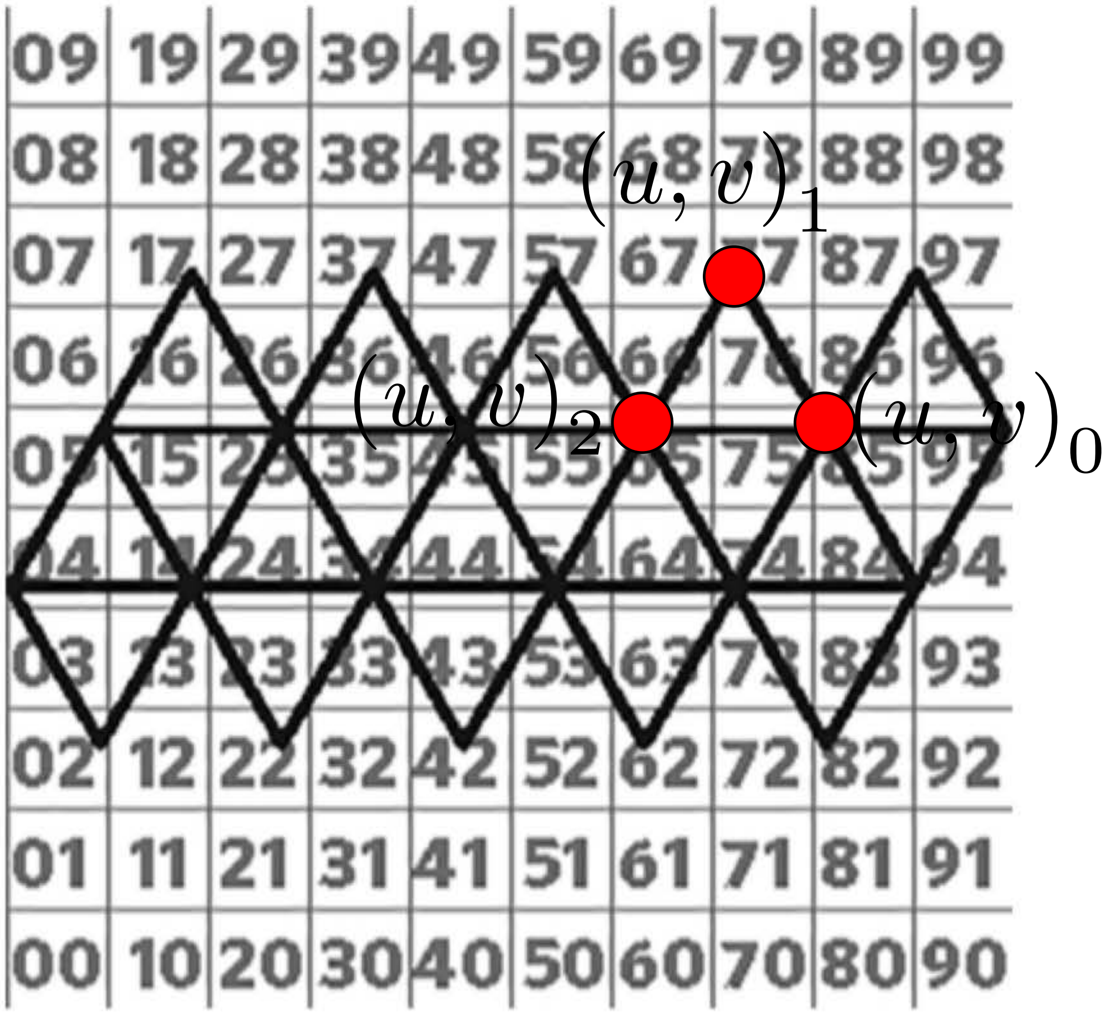
$$\phi(x, y, z) = ([\pi + \text{atan2}(y, x)]/2\pi, [\pi - \text{acos}(z/\|x\|)]/\pi)$$

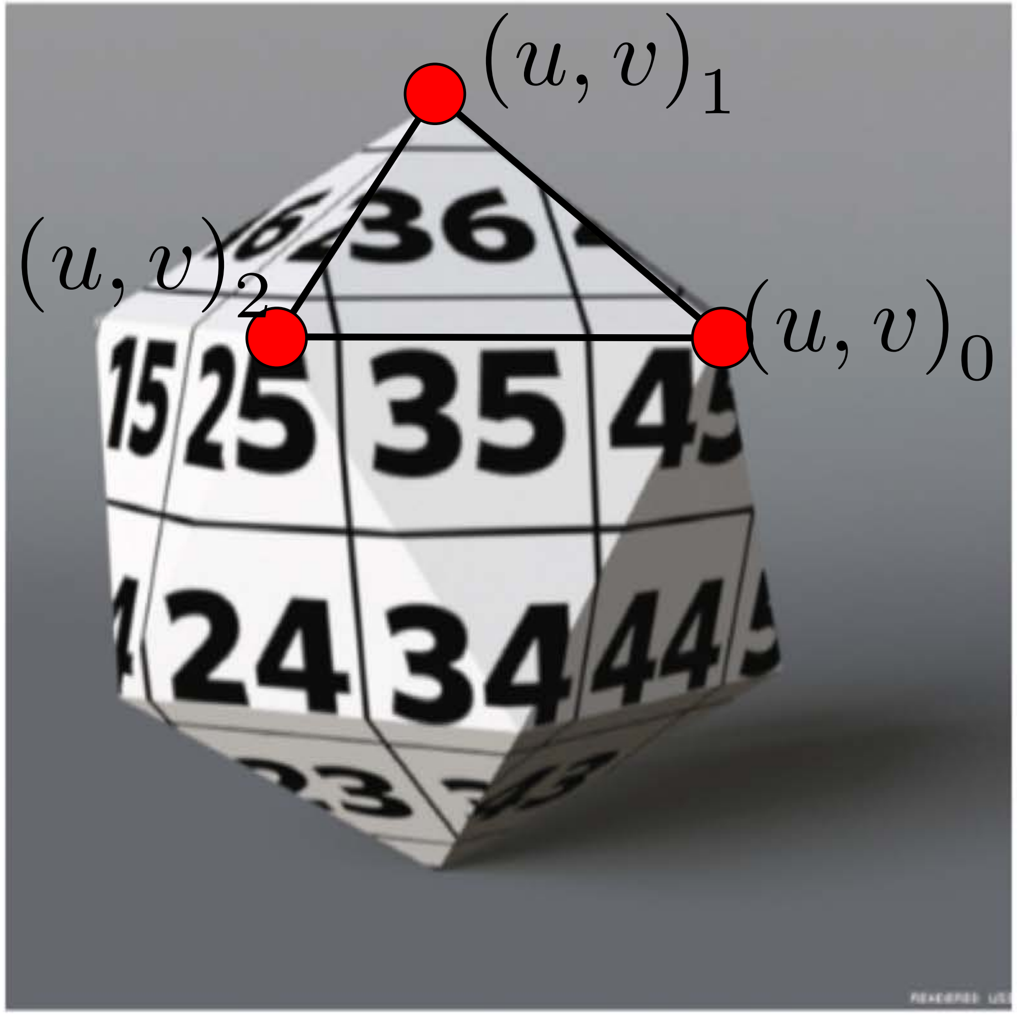
Cube Texture Map



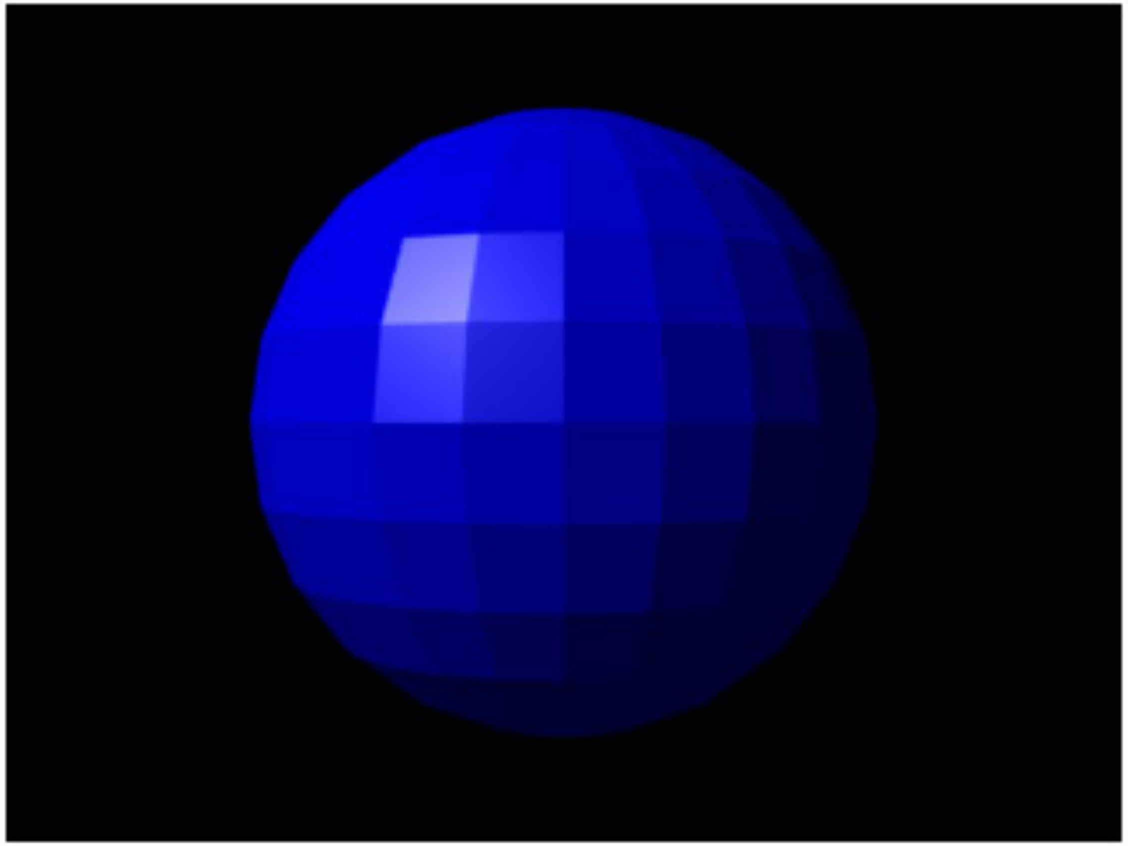


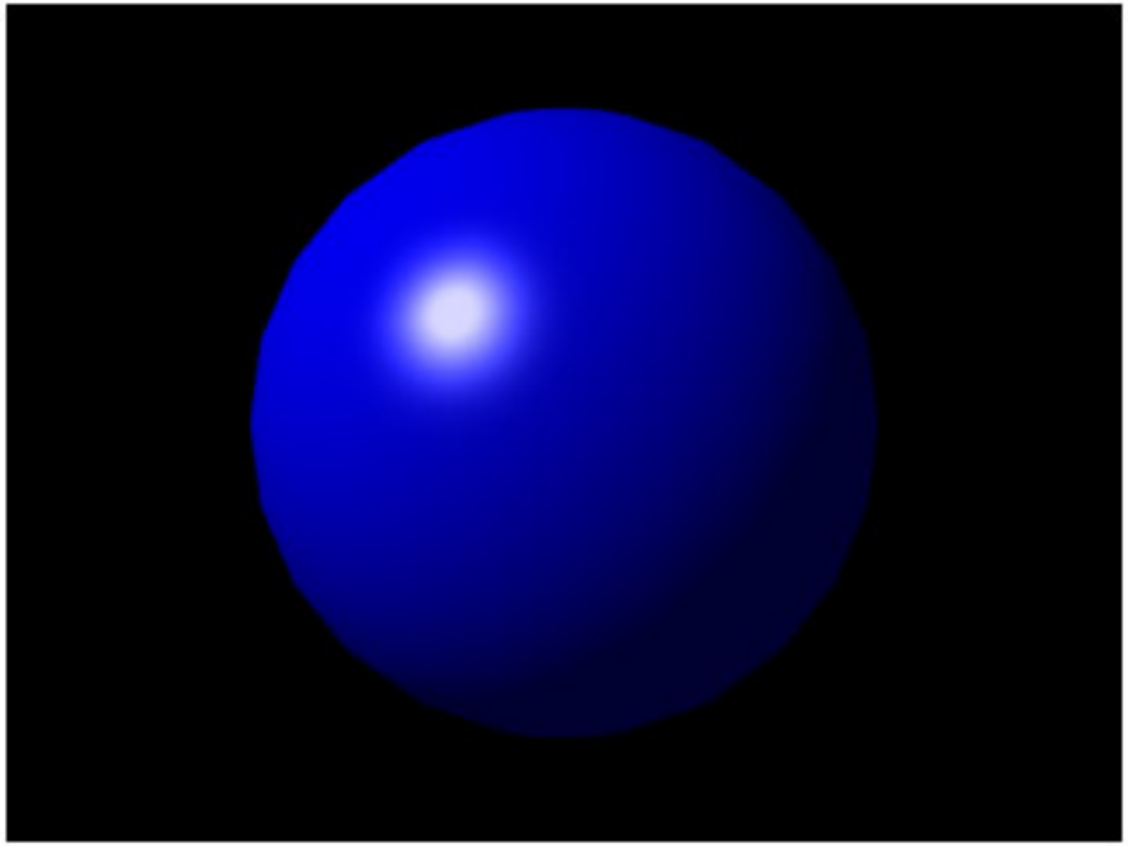
Interpolated Texture Coordinates



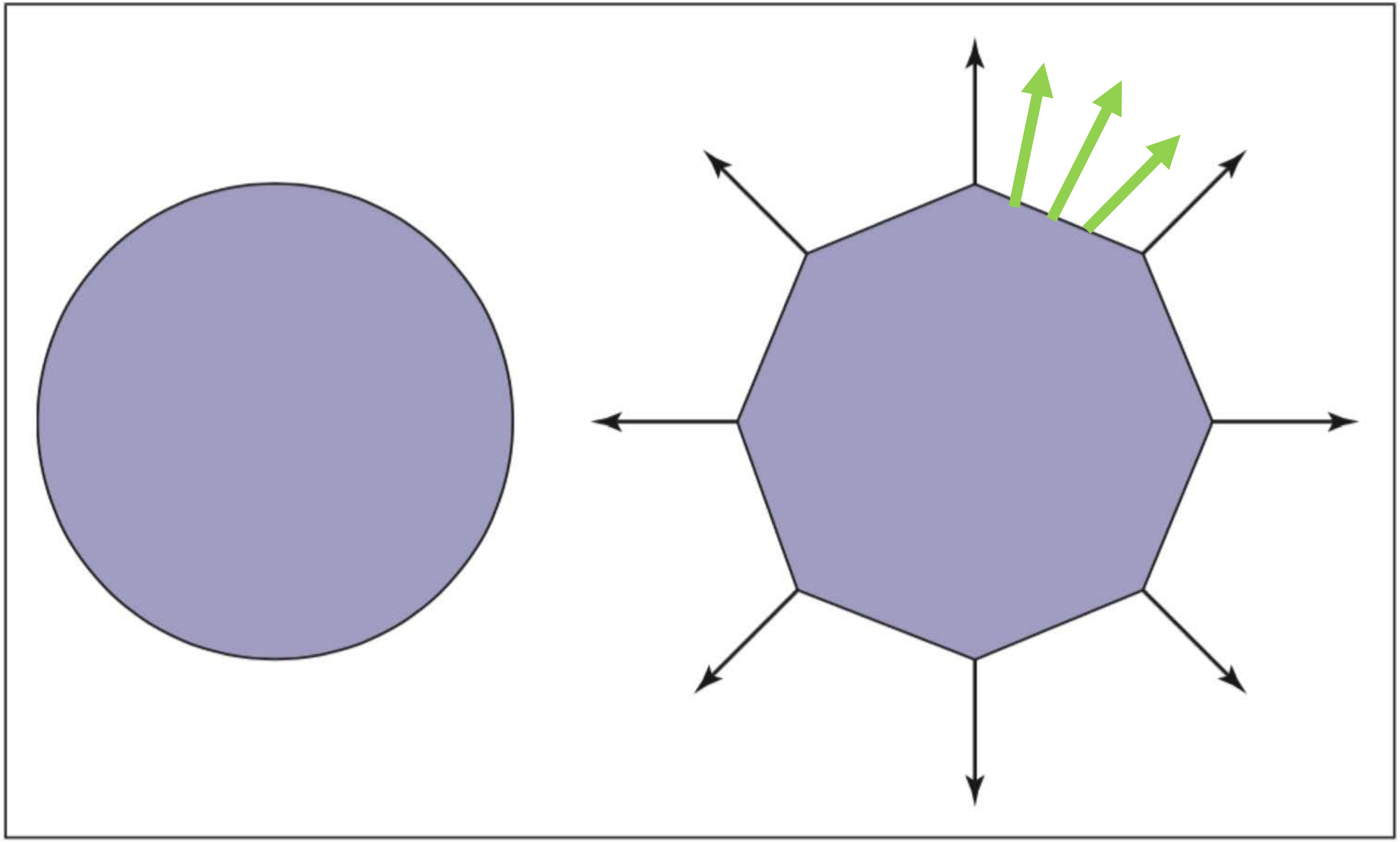


Smooth Surfaces in Computer Graphics

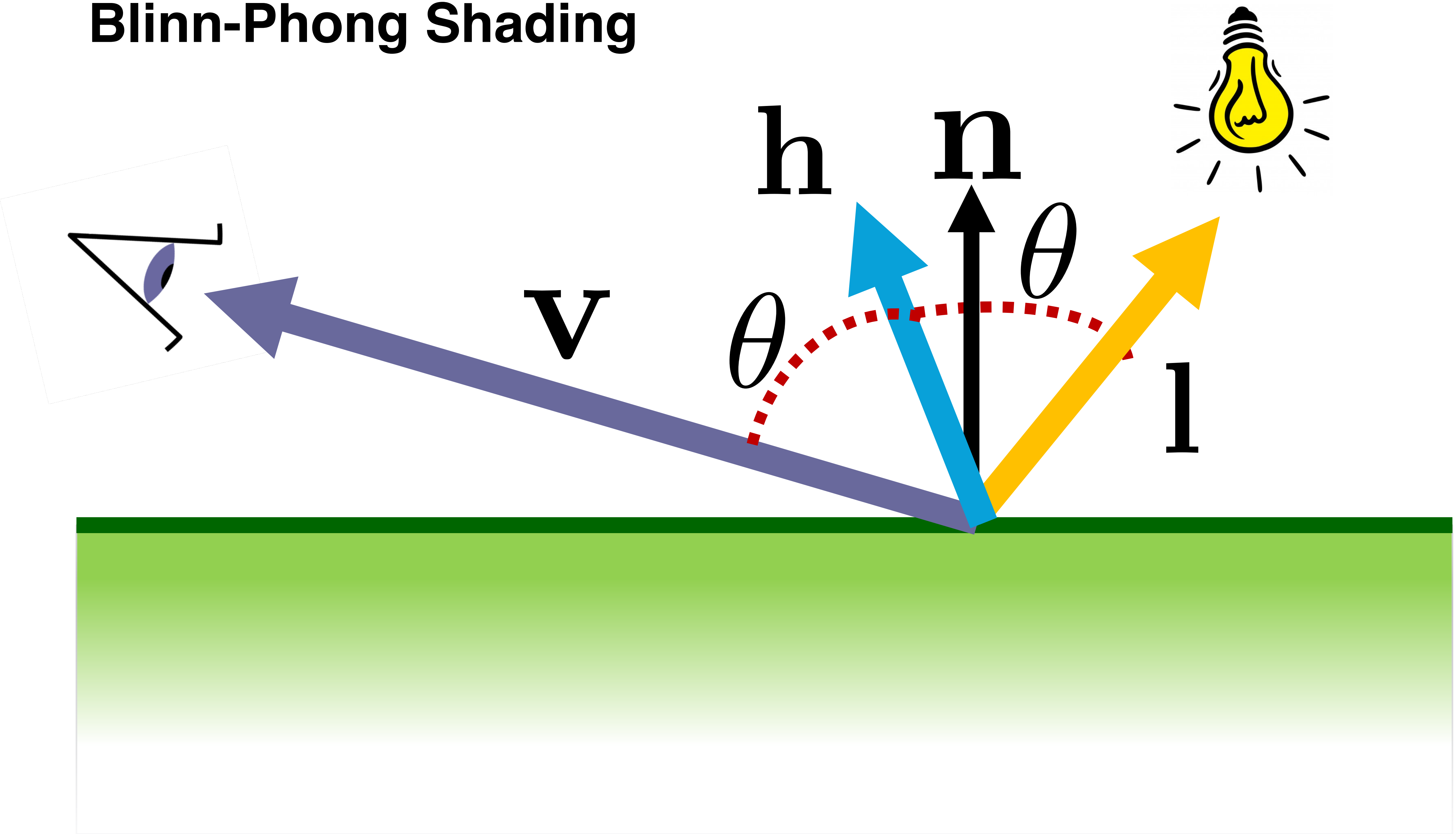




Phong Shading



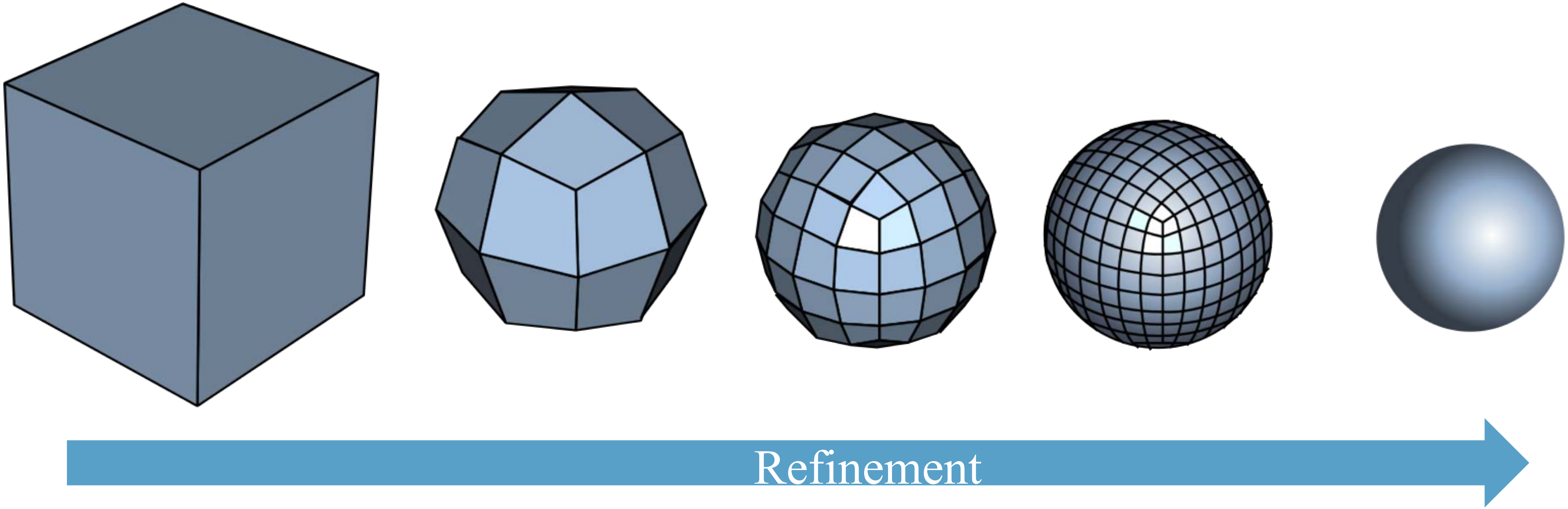
Blinn-Phong Shading



Subdivision Surfaces

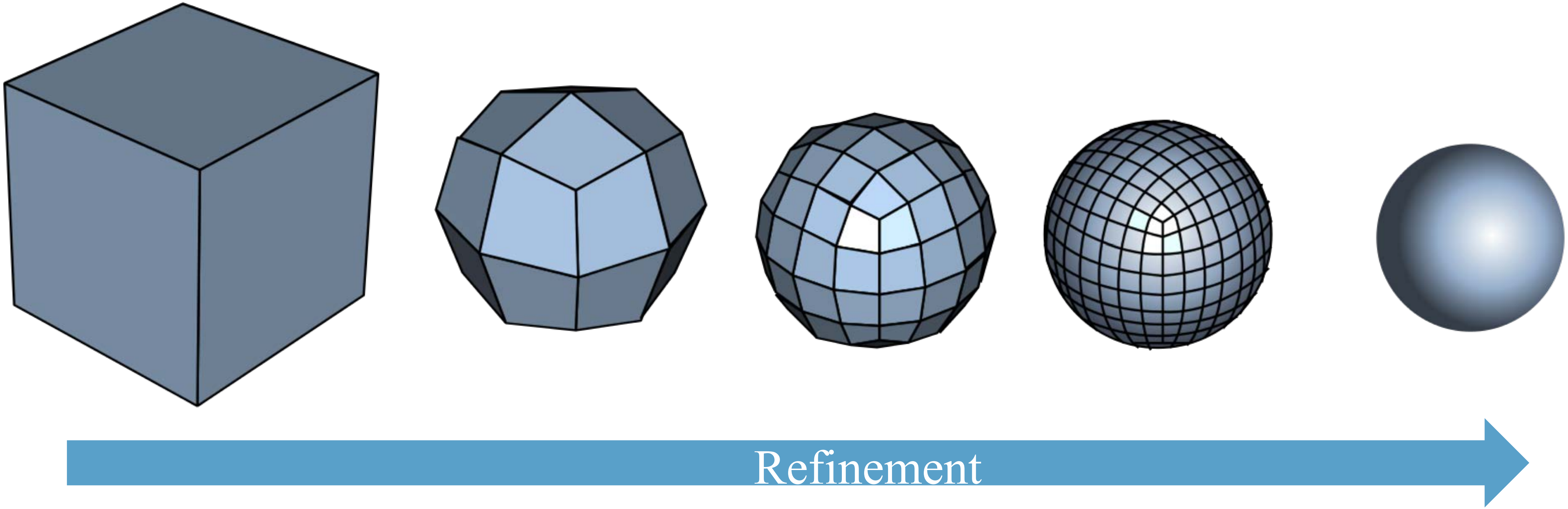
Recursive refinement of polygonal mesh

Results in a smooth "limit surface"



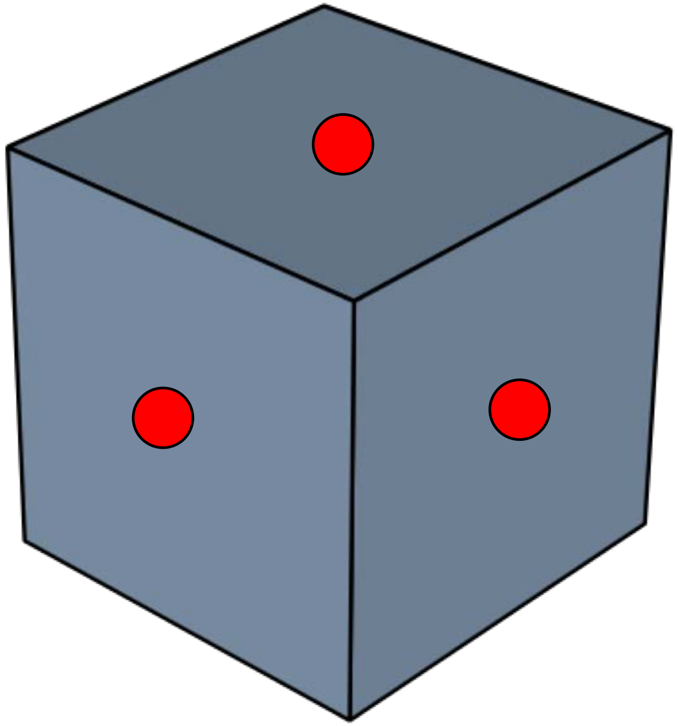
Catmull-Clark Subdivision

Particular type of subdivision scheme.



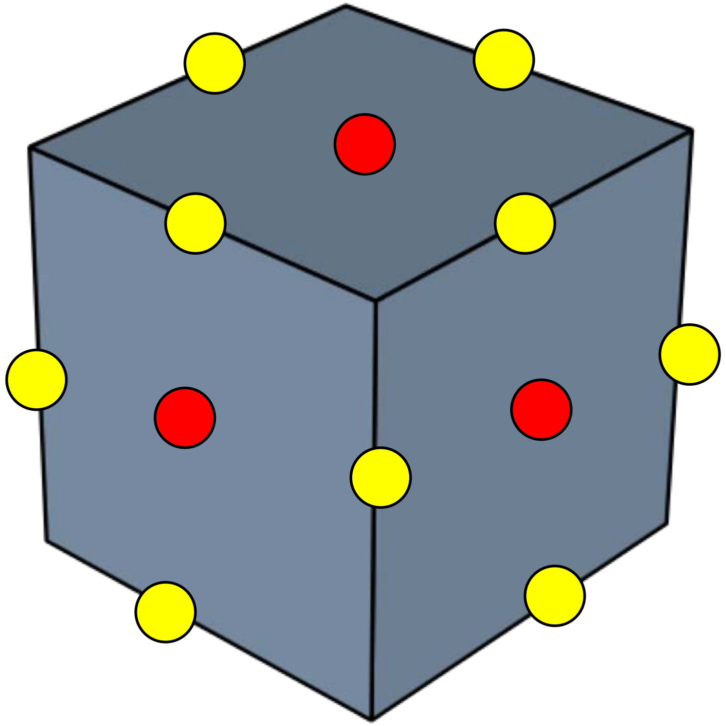
Catmull-Clark Subdivision

Step 1: Set the face point for each facet to be the average of its vertices



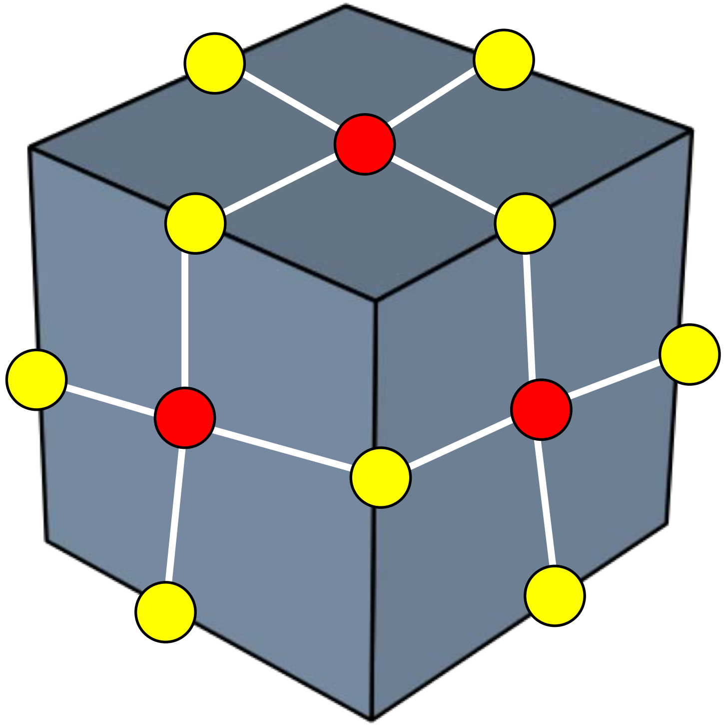
Catmull-Clark Subdivision

Step 2: Add edge points – average of two neighbouring face points and edge end points



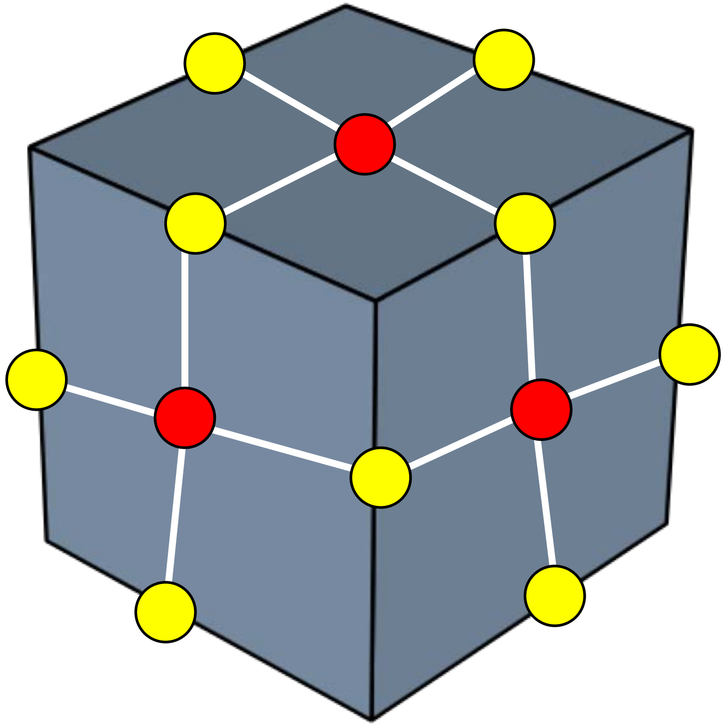
Catmull-Clark Subdivision

Step 3: Add edges between face points and edge points



Catmull-Clark Subdivision

Step 4: Move each original vertex according to new position given by:



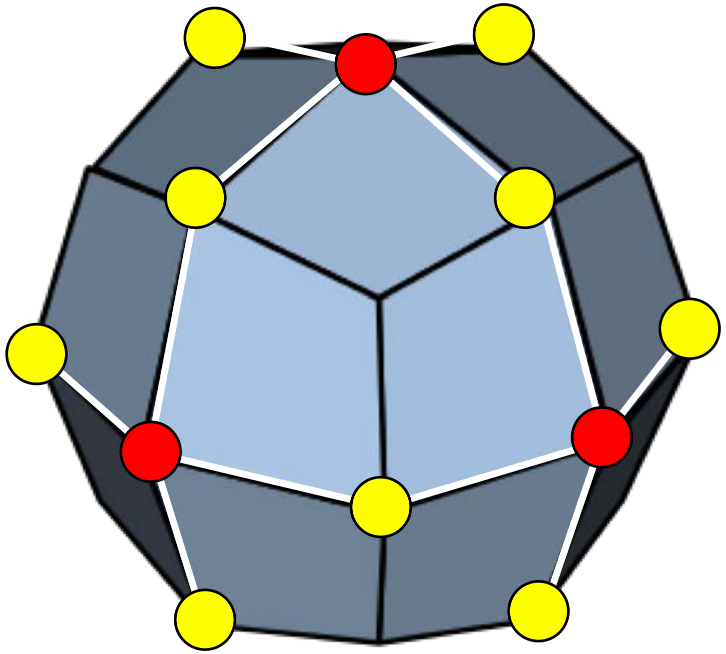
$$\frac{F + 2R + (n-3)P}{n}$$

F - Average of all n created face points adjacent to P

R - Average of all original edge midpoints touching P

Catmull-Clark Subdivision

Step 4: Move each original vertex according to new position given by:



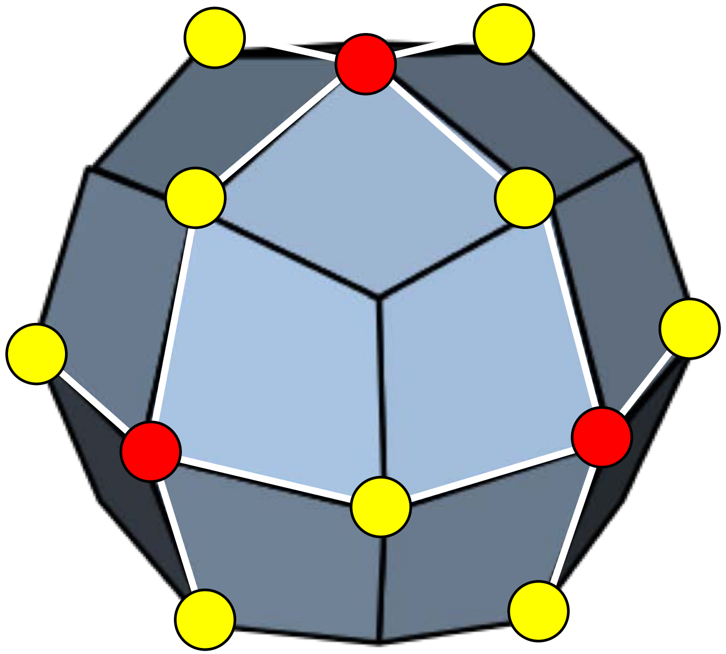
$$\frac{F + 2R + (n-3)P}{n}$$

F - Average of all n created face points adjacent to P

R - Average of all original edge midpoints touching P

Catmull-Clark Subdivision

Step 5: Connect up original points to make facets



Subdivision Surfaces in Action

http://graphics.pixar.com/opensubdiv/

Done

Assignment 4 due on Friday

Assignment 5 out soon

Office hours now BA5268