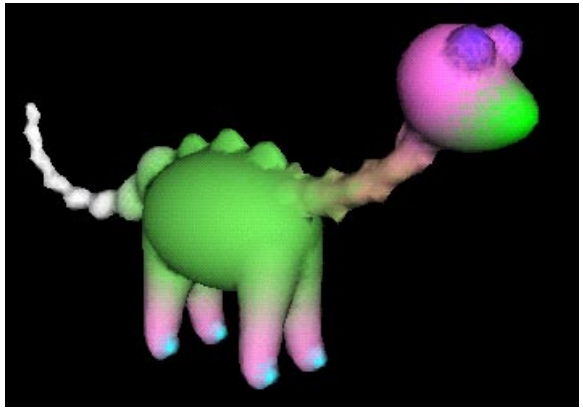
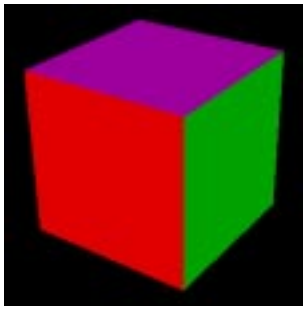
Computer Graphics



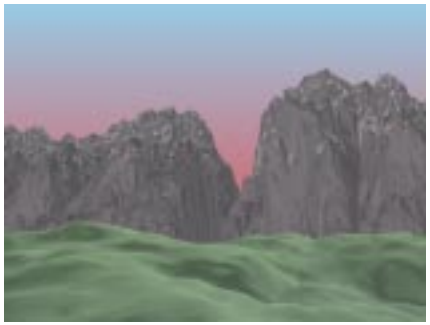
3D Modelling

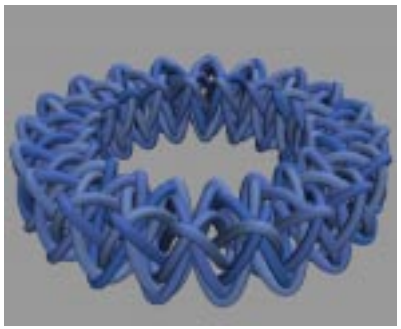


by

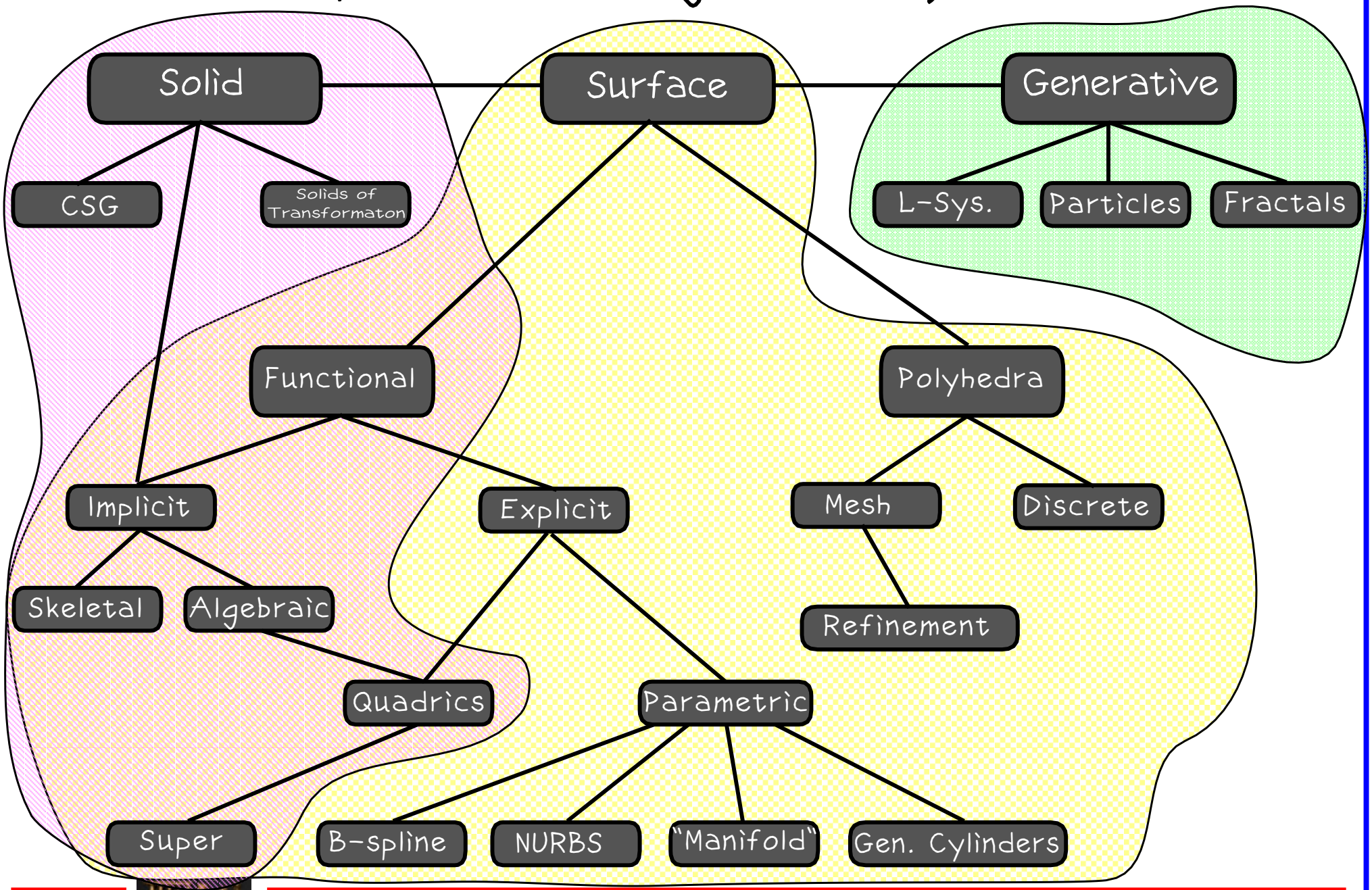
Brian Wyvill

University of Calgary



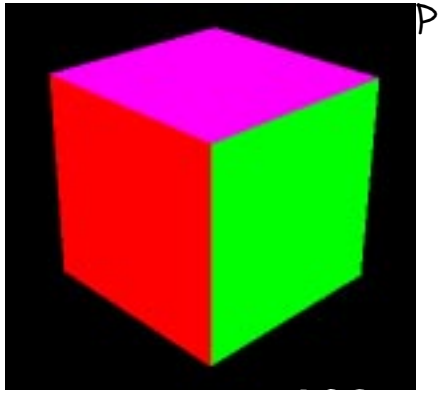


A Taxonomy of Modelling Techniques



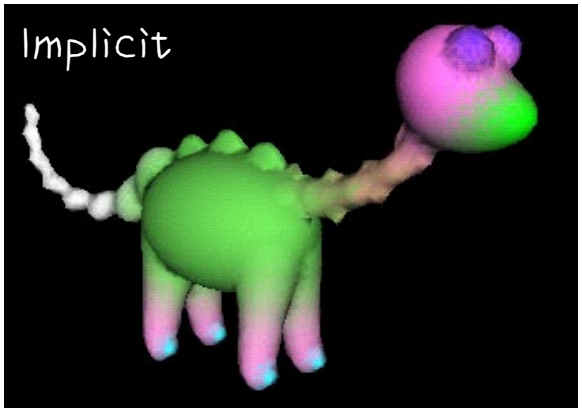
Some Examples

Polyhedra

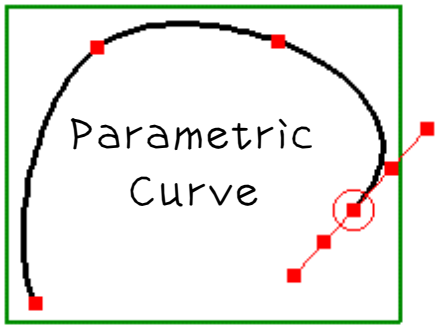


CSG

Implicit



Parametric Curve



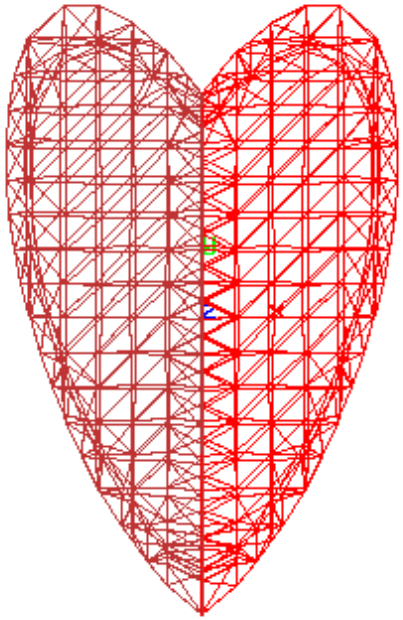
Parametric Patches

Fractal Terrain

L-System plant

Parametric Generalised cylinders

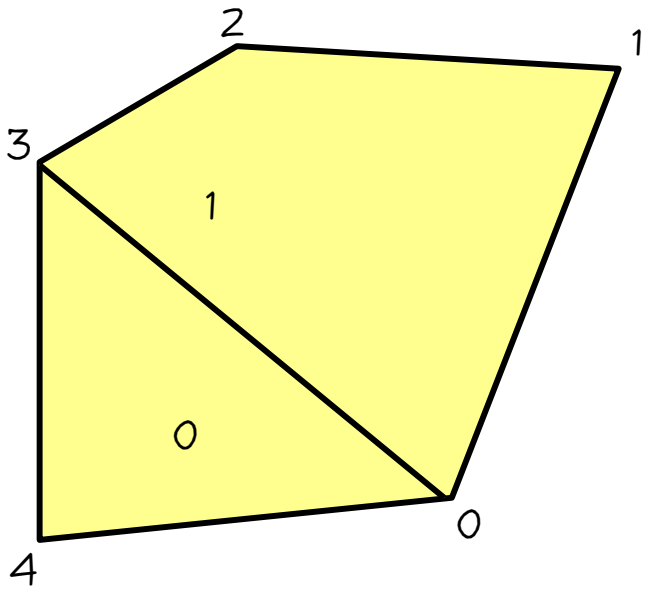
Polygon Meshes



Triangular Mesh

Mesh Consistencey

All polygons closed, all edges used once but less than n-times. Each vertex is references by at least two edges (closed mesh). Other applications have more stringent requirements, e.g. planarity or no holes.



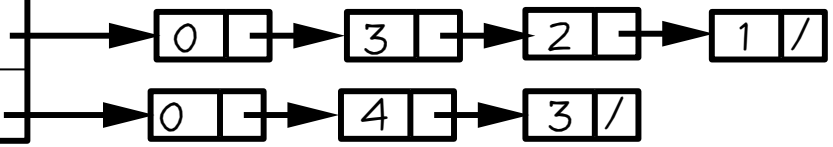
F&VD.1 p471
F&VD.2 p321

Vertex List

4	(x,y,z)
3	(x,y,z)
2	(x,y,z)
1	(x,y,z)
0	(x,y,z)

Polygon List

4		
3		
2		
1	4	
0	3	

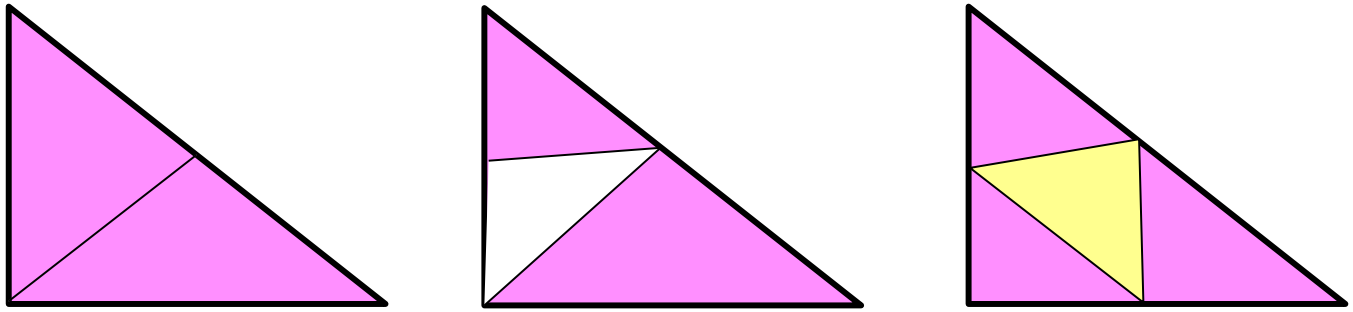


Edge List

5	0	3	1	0
4	1	0	1	/
3	2	1	1	/
2	3	2	1	/
1	4	3	0	/
0	0	4	0	/

Vertex1 vertex2 Polygon1 Polygon2

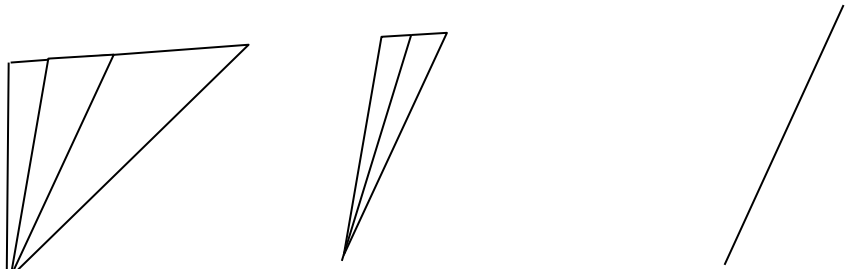
Subdivision



One edge

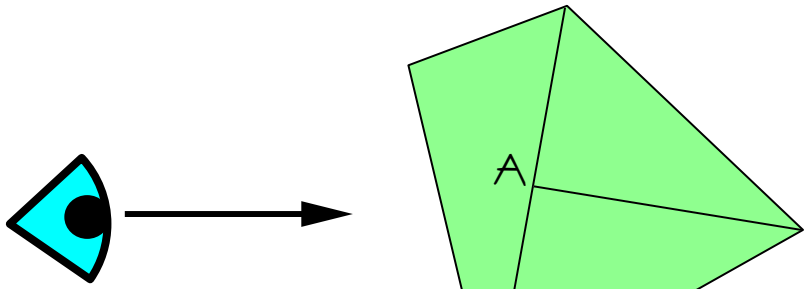
Two edge

Three edge

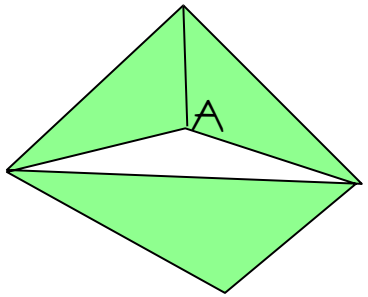


Sliver triangle

Degenerate

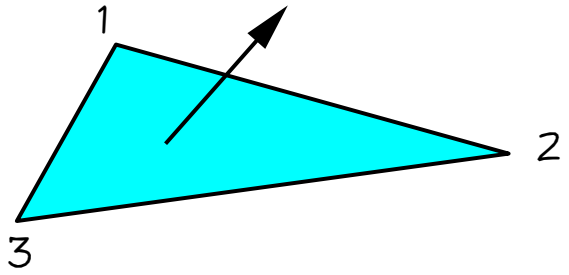


Viewing direction



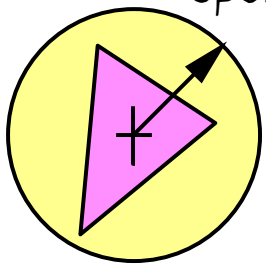
3D Subdivision Cracks Problem (t-intersections)

Using Triangle Meshes



1. Make sure triangles defined in a consistent direction, e.g. counter-clockwise indicates outwards normal.

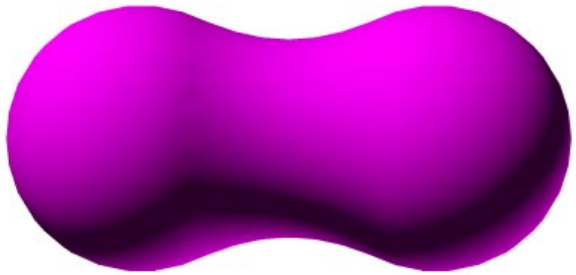
2. Check for co-linear vertices or all vertices within epsilon of each other.



circle of "confidence"

3. Rendering speed (few triangles) Vs. Smooth curved surfaces.

3016 polygons



408 polygons

4. Silhouette Edges.

From the OpenGl Manual

Constructing polygonal approximations to surfaces is an art, and there is no substitute for experience. This section, however, lists a few pointers that might make it a bit easier to get started.
Keep polygon orientations consistent. Make sure that when viewed from the outside, all the polygons on the surface are oriented in the same direction (all clockwise or all counterclockwise). Try to get this right the first time, since it's excruciatingly painful to fix the problem later.

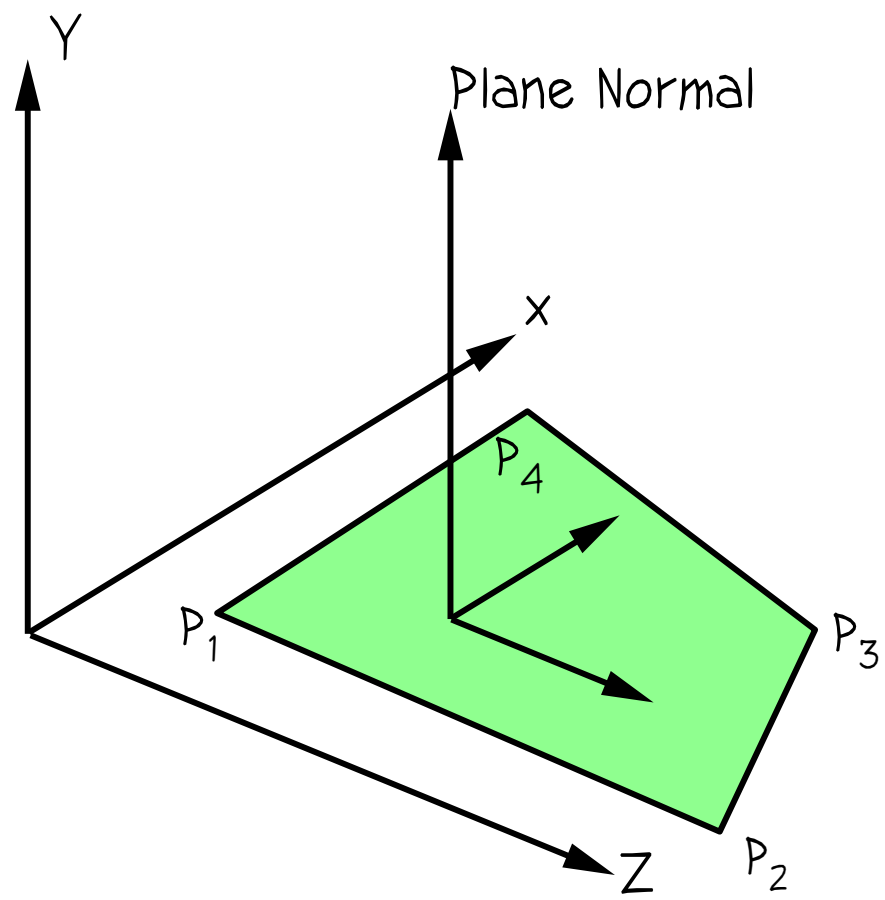
When you subdivide a surface, watch out for any non-triangular polygons. The three vertices of a triangle are guaranteed to lie on a plane; any polygon with four or more vertices might not. Nonplanar polygons can be viewed from some orientation such that the edges cross each other, and OpenGL might not render such polygons correctly.

There's always a trade-off between the display speed and the quality of the image. If you subdivide a surface into a small number of polygons, it renders quickly but might have a jagged appearance; if you subdivide it into millions of tiny polygons, it probably looks good but might take a long time to render. Ideally, you can provide a parameter to the subdivision routines that indicates how fine a subdivision you want, and if the object is farther from the eye, you can use a coarser subdivision. Also, when you subdivide, use relatively large polygons where the surface is relatively flat, and small polygons in regions of high curvature.

For high-quality images, it's a good idea to subdivide more on the silhouette edges than in the interior. If the surface is to be rotated relative to the eye, this is tougher to do, since the silhouette edges keep moving. Silhouette edges occur where the normal vectors are perpendicular to the vector from the surface to the viewpoint that is, when their vector dot product is zero. Your subdivision algorithm might choose to subdivide more if this dot product is near zero.

Try to avoid T-intersections in your models (see Figure 2-13). As shown, there's no guarantee that the line segments AB and BC lie on exactly the same pixels as the segment AC. Sometimes they do, and sometimes they don't, depending on the transformations and orientation. This can cause cracks to appear intermittently in the surface.

Plane Equation and Normal



A=C=0 for this plane

$$Ax + By + Cz + D = 0$$

A, B, C, D can be computed from 3 non-colinear points, 4th equation is the plane equation. (Write it as determinants and expand by cofactors).

Normal to the plane is given by coeficients [A B C]

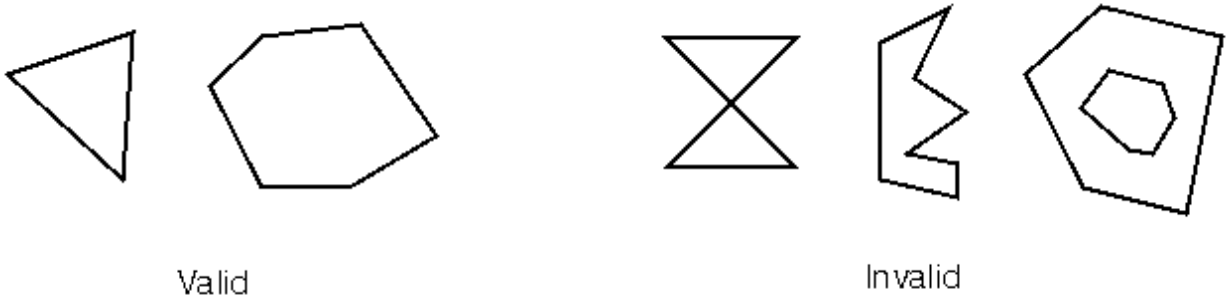
Can also compute normal as cross product of 2 edges:

$$P_1P_2 \times P_1P_4$$

Zero cross product indicates colinear vertices.

F&VD.1 p476
F&VD.2 p325

OpenGl Polygons and Triangle Meshes



OpenGL allows convex polygons to be specified:

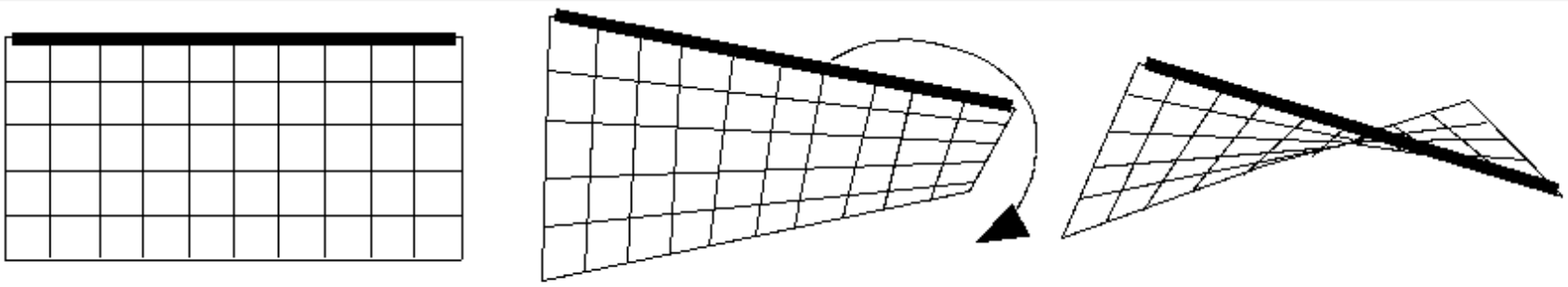


Figure 2–3 Nonplanar Polygon Transformed to Nonsimple Polygon

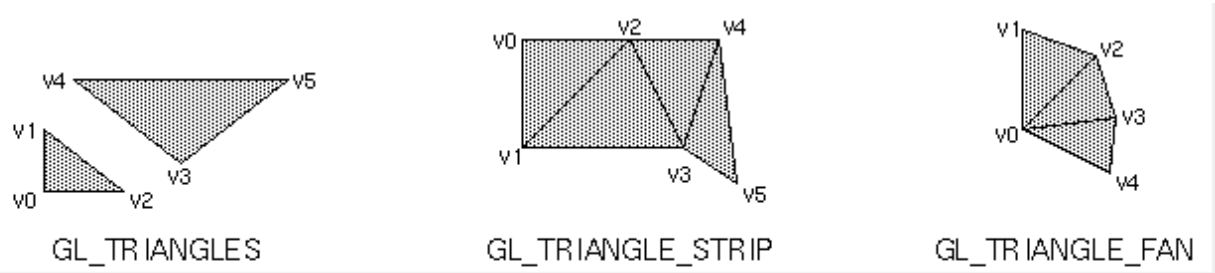
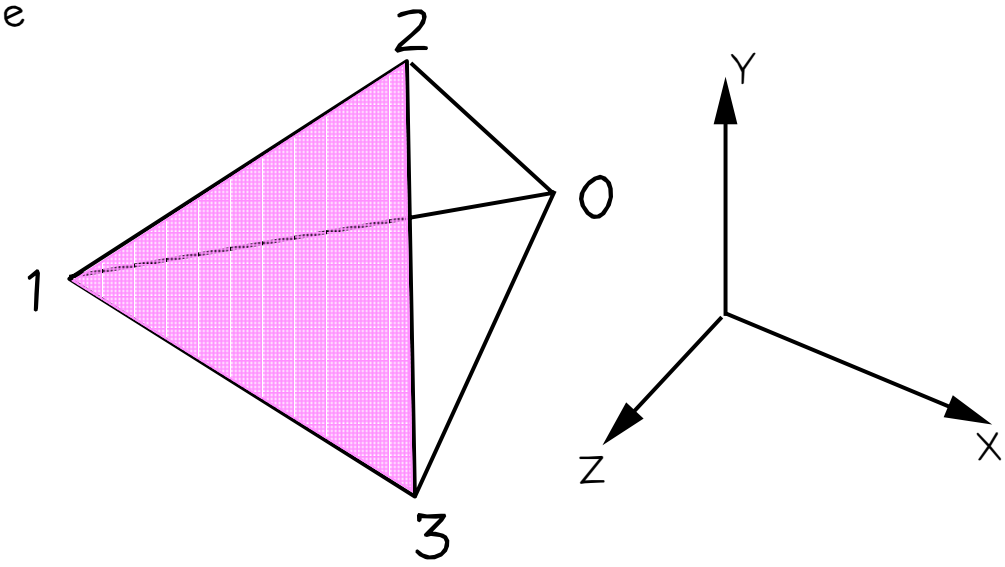


Figure 2–6 Geometric Primitive Types

A Primitive Mesh file format

```
*       opcode    data
*
*       v   x y z
*       n   nx ny nz
*       c   r g b       # n or c commands apply to the preceeding v command
*       u   u v         * uv coordinates for texturing
*       t   i1 i2 i3    # i1 i2 i3 are addresses in table of vertices from 0
*       # ignore rest of line
```

```
v 0.000000 0.000000 -1.500000
n 0.000000 -0.219275 -0.975663
c 1.000000 0.000000 0.000000
v -1.500000 0.000000 1.500000
n -0.930914 0.131520 0.340738
c 1.000000 0.000000 1.000000
v 0.000000 1.500000 0.000000
n 0.000000 0.997432 -0.071612
c 0.000000 1.000000 0.000000
v 1.500000 0.000000 1.500000
n 0.930914 0.131520 0.340738
c 1.000000 1.000000 0.000000
t 0 1 2
t 3 0 2
t 1 3 2
t 0 3 1
```



Only colour and texture coordinates
No other model/surface attributes