Object Representations

Object Representation

What Objects?

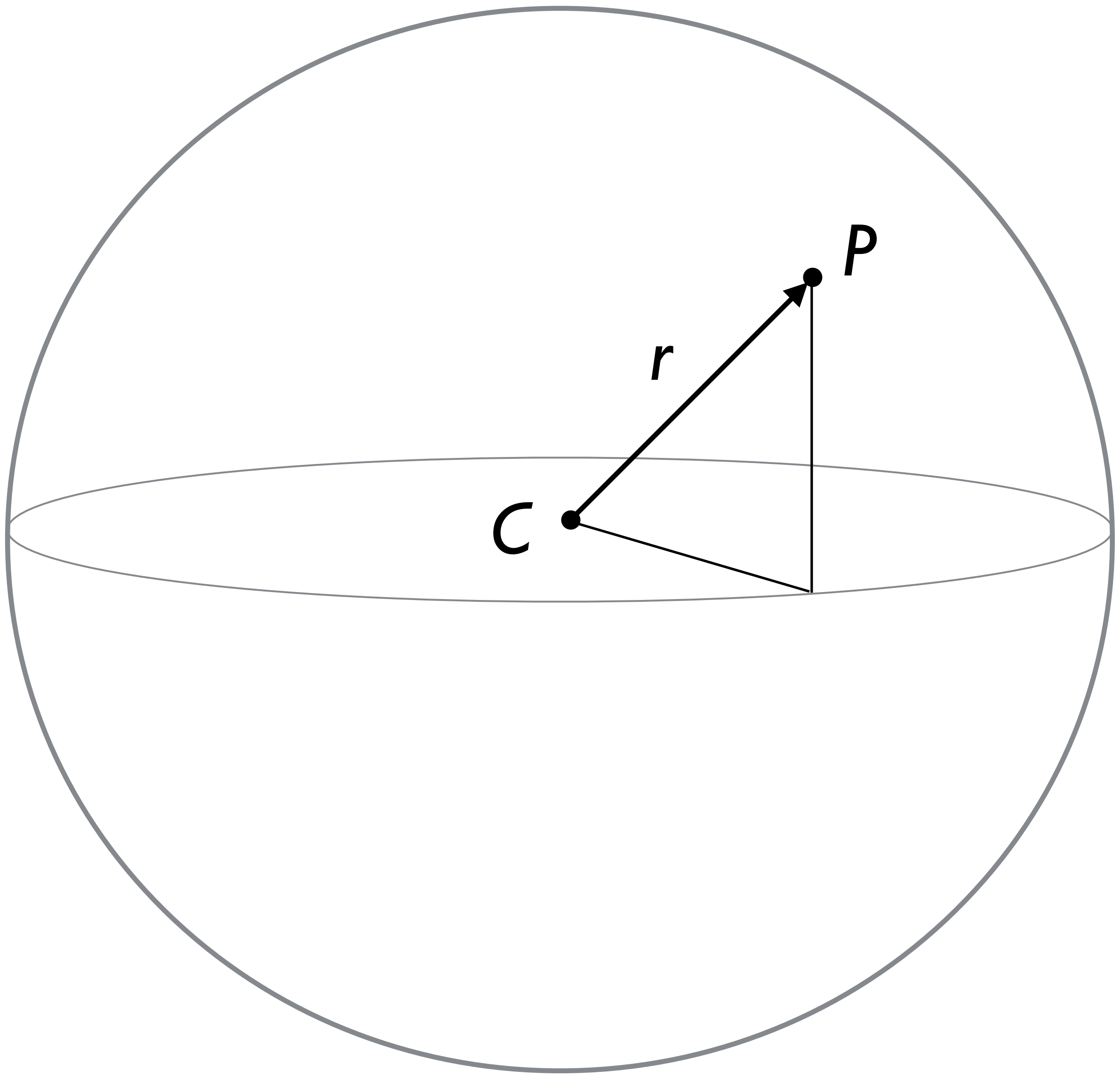
- Solids (sphere, cube, cone, torus, …)
- Flat Surfaces (plane, polygon, discs, …)
- Curved Surfaces (paraboloid, hiperboloid, bicubic, nurbs, …)
- Soft or deformable (liquids, smoke, cloth, hair)

What Operations?

- Rendering on a raster engine
- Rendering on a ray tracing engine
- Compute features such as volumes, areas, etc.
- intersection, difference, union
- Distinguish between inside, outside and surface

Example

Consider a sphere centred in $C = (c_x, c_y, c_z)$, with radius r.



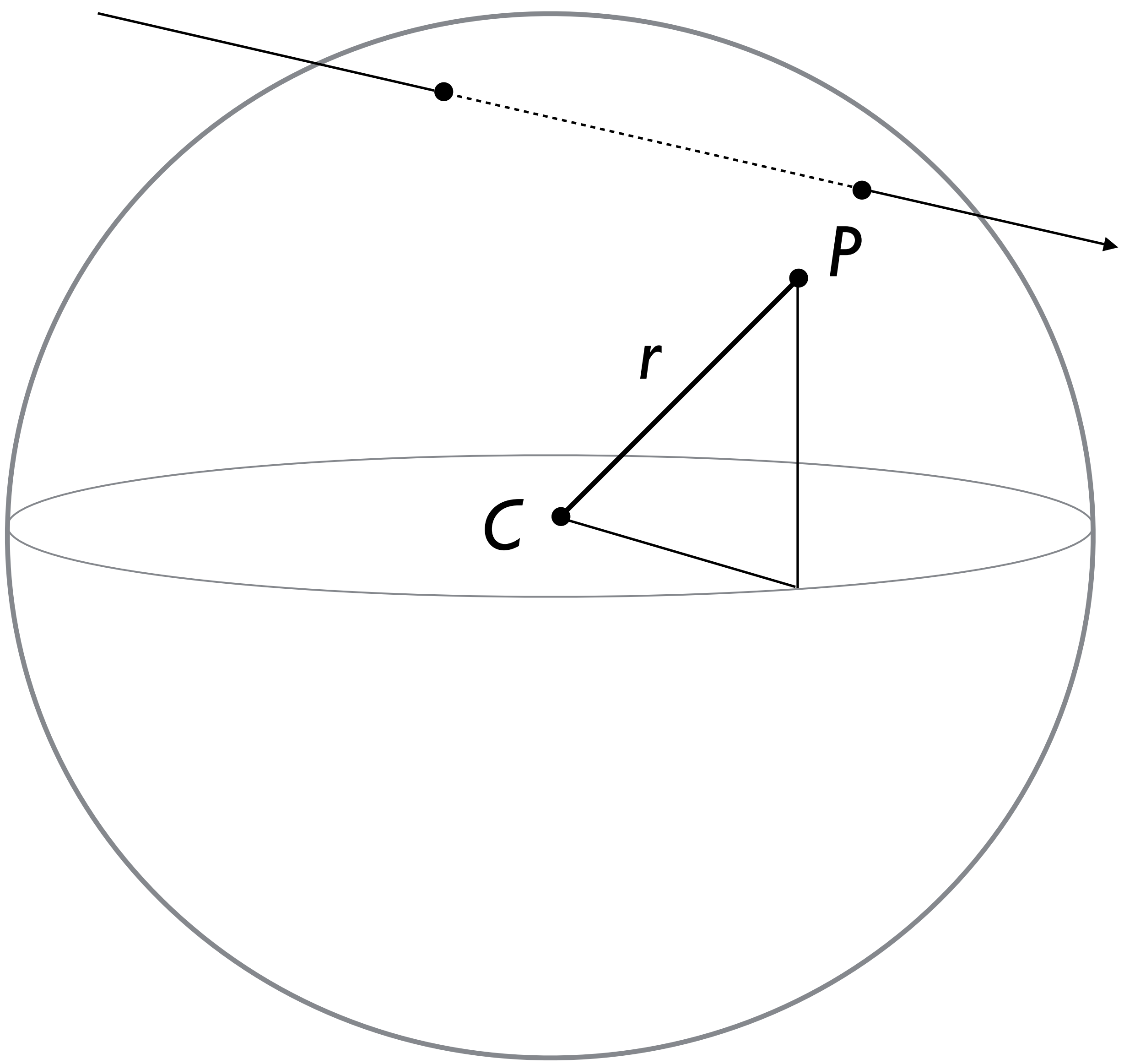
Point $P = (x, y, z)$ can be:

- inside:
- outside
- on the surface

$$\sqrt{(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2} \begin{matrix} < \\ > \\ = \end{matrix} r$$

Algebraic Representations

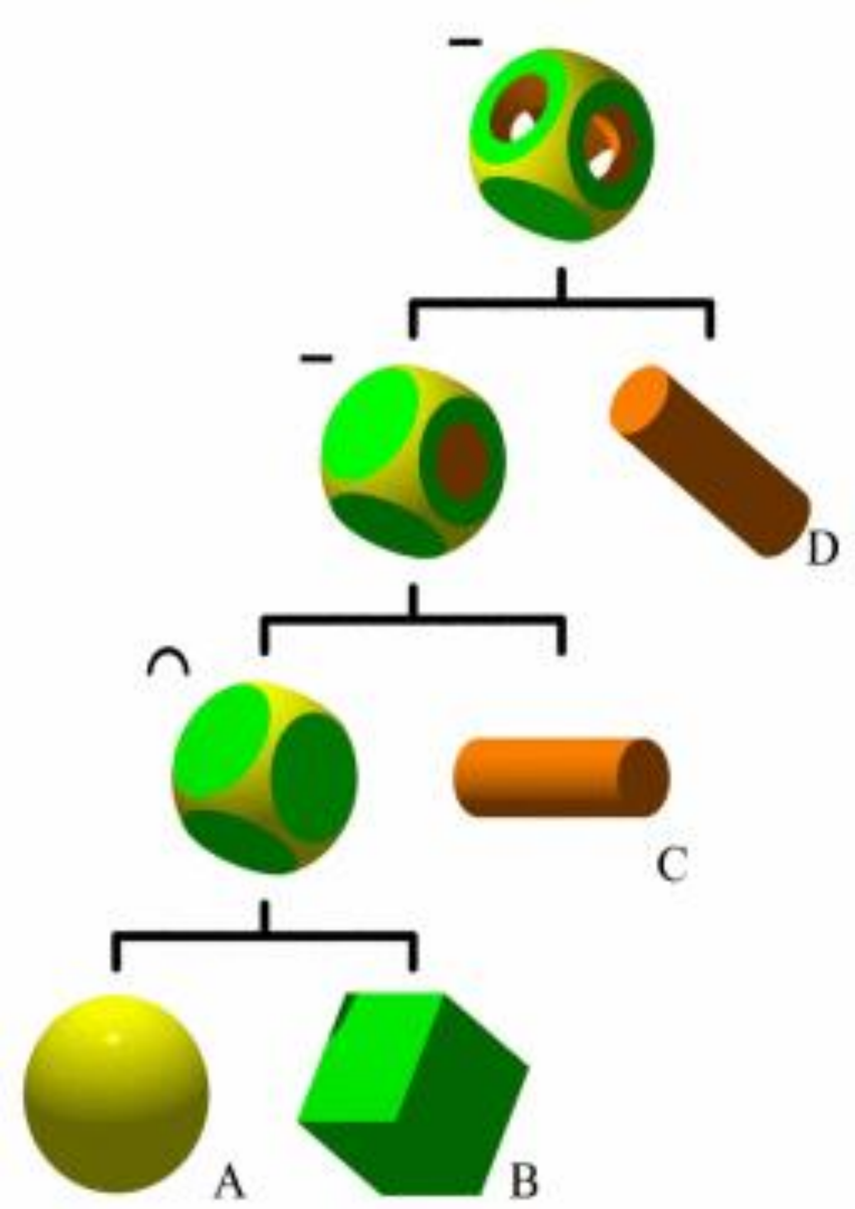
Example



Appropriate for:

- Ray tracing: compute intersections of rays with object's surface to determine entry and exit points

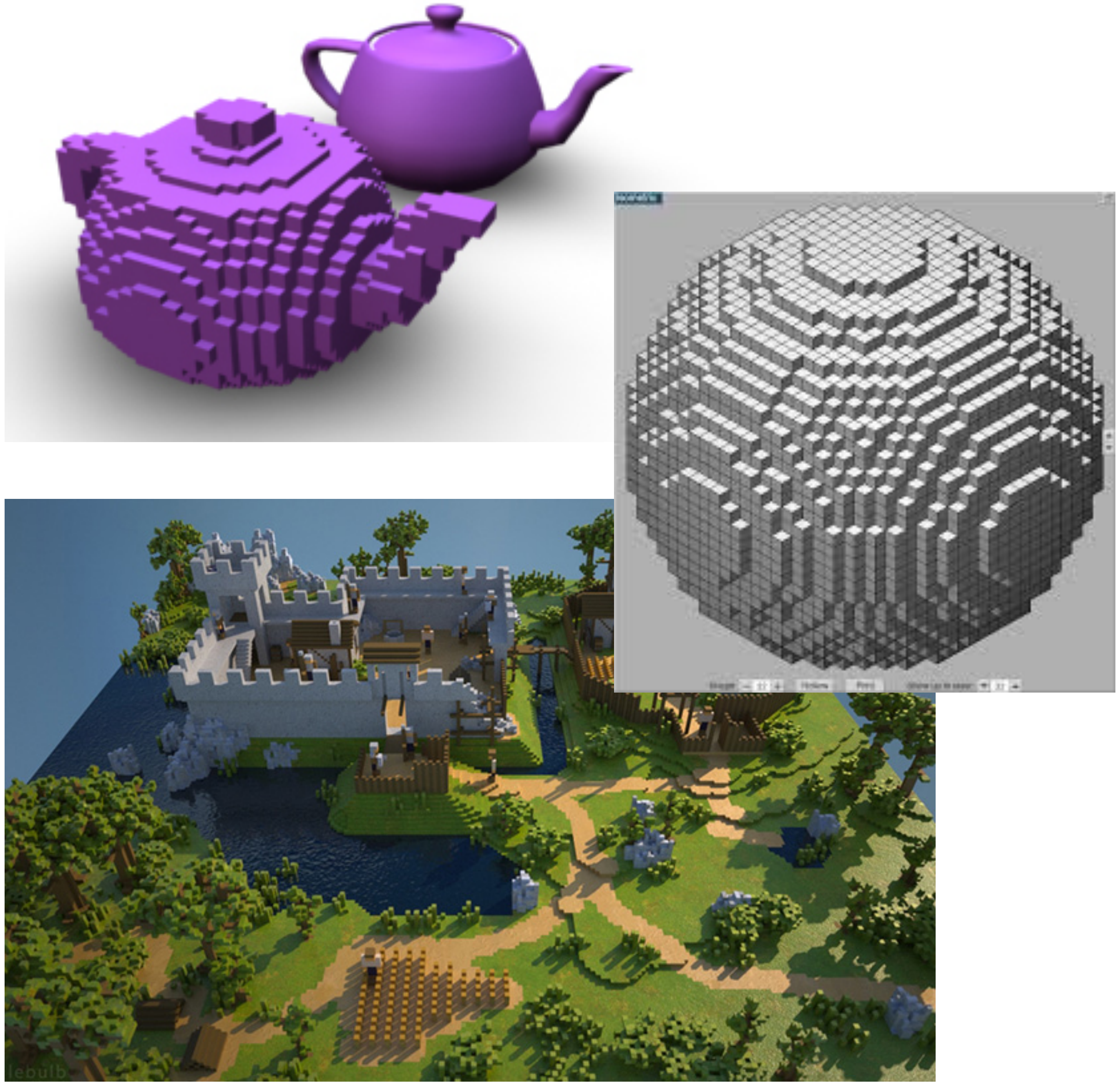
Example



Appropriate for:

- Boolean set operations to build solids from others using Constructive Solid Geometry (CSG).

Spatial-Occupancy Enumeration Representation



An object is a list of voxels (3D pixels).

Each voxel may be occupied (belongs to some object) or empty

Appropriate in CAT/MRI visualisation applications

+ adjacency test

+ inside/outside test

+ boolean set operations

- no partial occupancy

- accuracy

- data structure size

Spatial-Occupancy Enumeration Representation

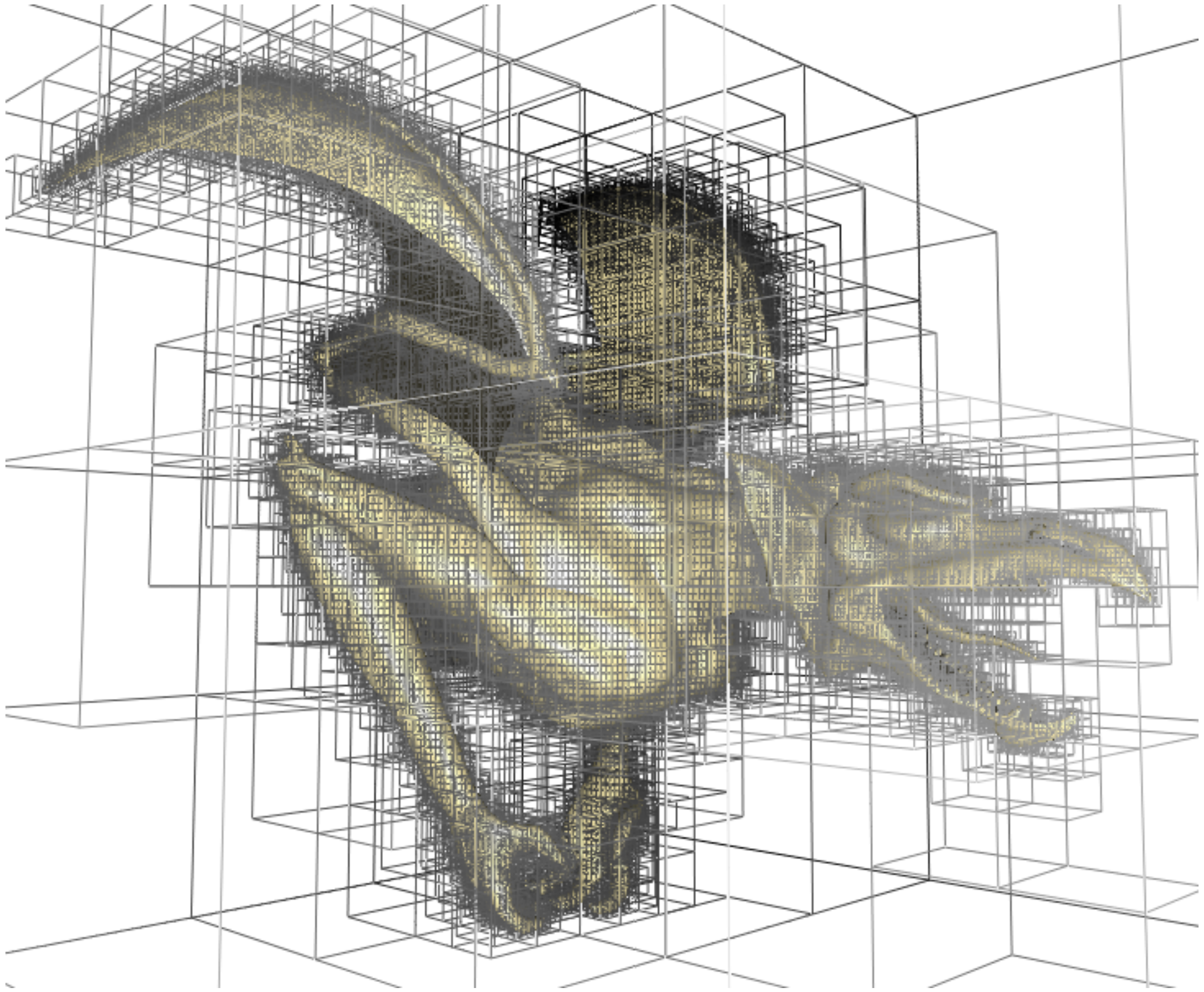
Octree based representations were introduced to reduce storage needed by regular grids of voxels.

Extension to 3D from 2D quad trees.

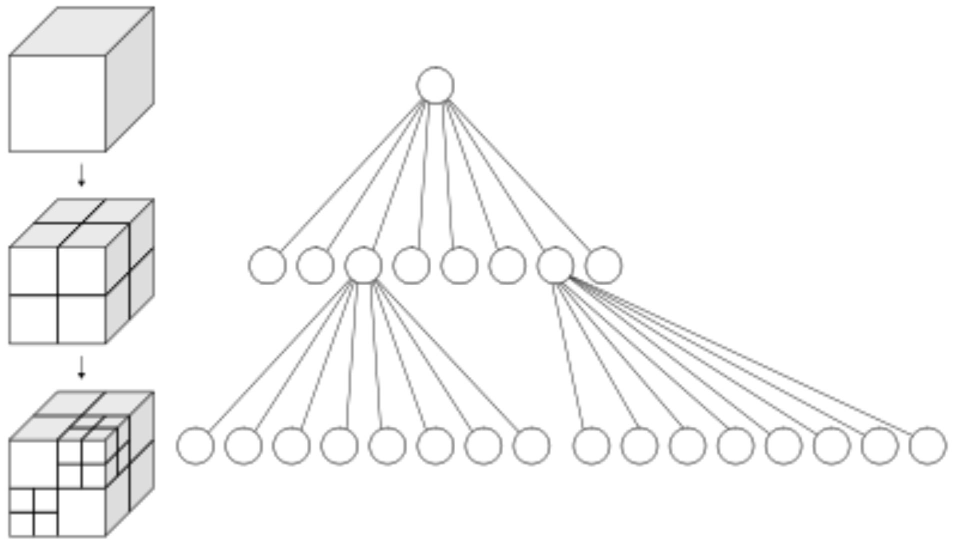
Binary subdivision of a cell along 3 axis gives 8 sub-cells.

Repeated recursively until we get an homogeneous cell or reach recursion limit.

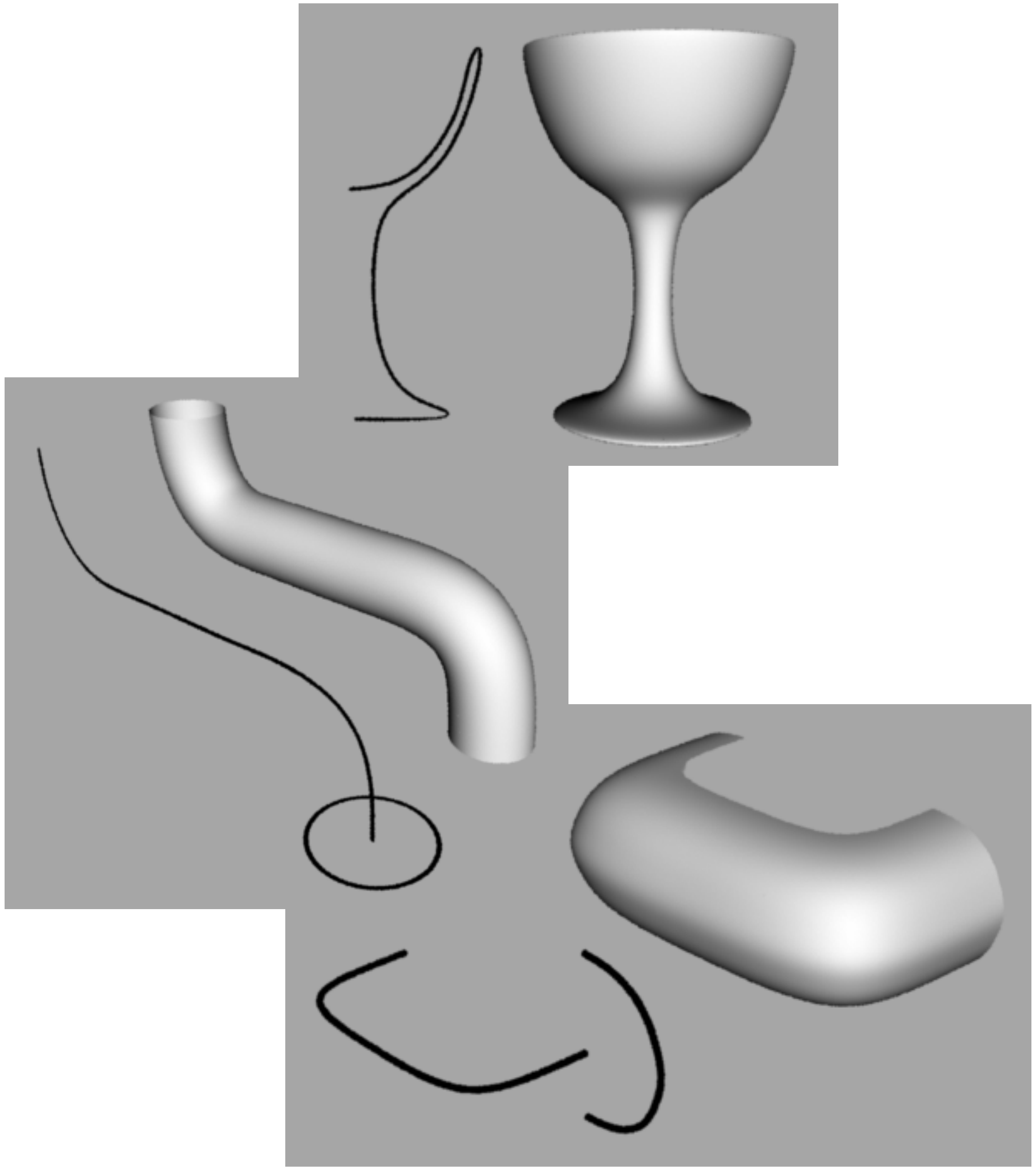
Trees with internal nodes having 8 children.



* from GPU Gems 2



Sweep Representations



Obtained by sweeping an object along a trajectory.

Translational and rotational sweeps.

Object may change in size while being swept.

- Not adequate for boolean set operations since the result is normally not a swept object.

+ Volume/Area calculation may be easy

+ Appropriate for some Computer Aided Manufacturing applications

* from http://ayam.sourceforge.net

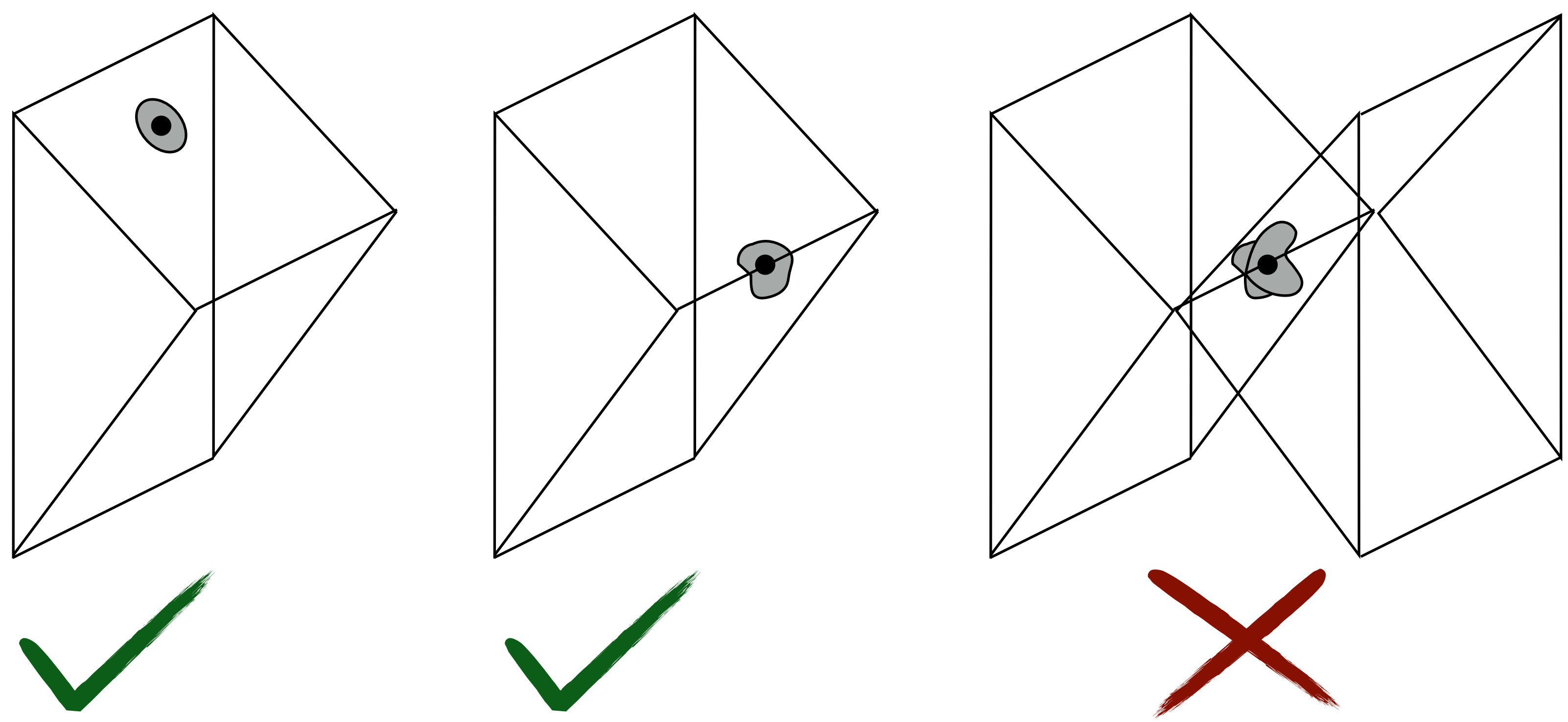
Boundary Representations

Boundary Representations (B-REPS)

- Objects are described in terms of their surface boundaries: vertices, edges and faces
- Curved surfaces are allowed, but they are mostly approximated with polygons or with patches (bicubic or nurbs)
- Convex polygons are the most common type of face with B-REPs.
- Some systems reduce general polygons to triangles by splitting the faces.
- Many b-rep systems only support 2-manifold objects.

Boundary Representations

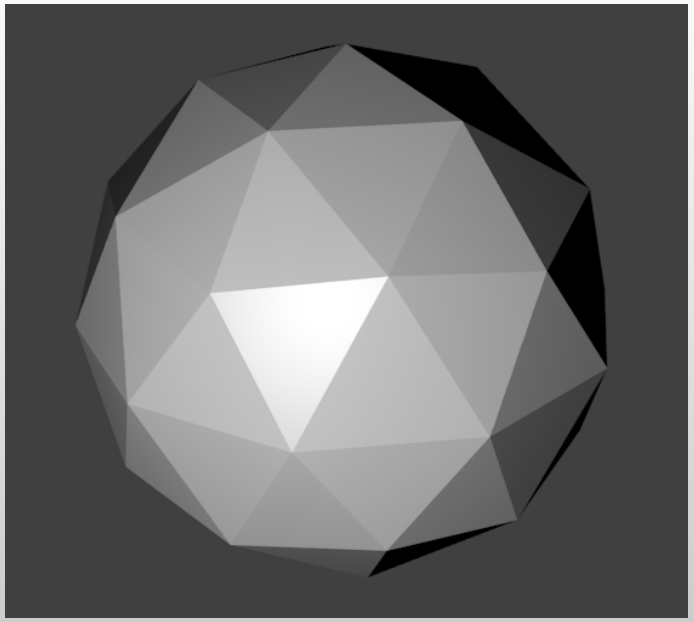
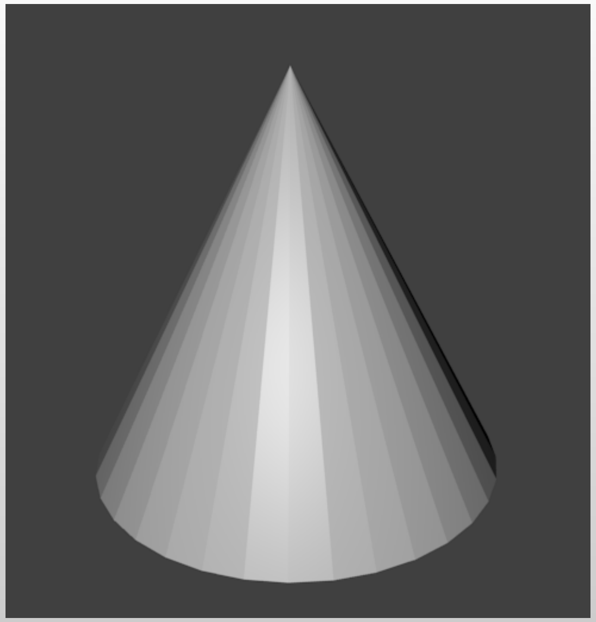
2-Manifold: every point on a 2-manifold has some arbitrarily small neighbourhood of points around it that can be considered topologically the same as a disk in the plane.



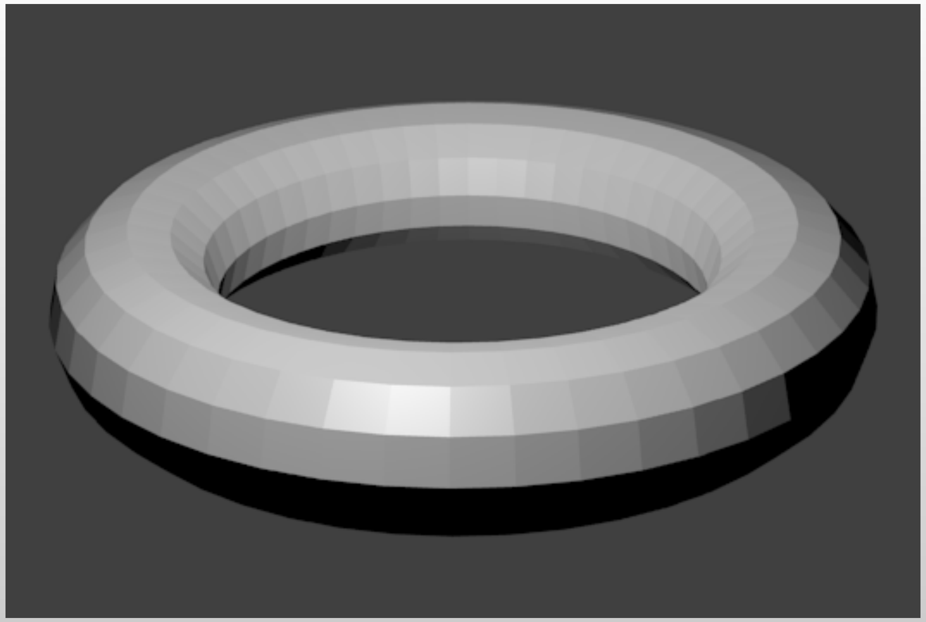
there is a continuous one-to-one correspondence between the neighbourhood of points and the disc.

Boundary Representations

Polyhedron: solid bounded by a set of polygons whose edges belong to an even number of polygons (2 for 2-manifolds).

simple

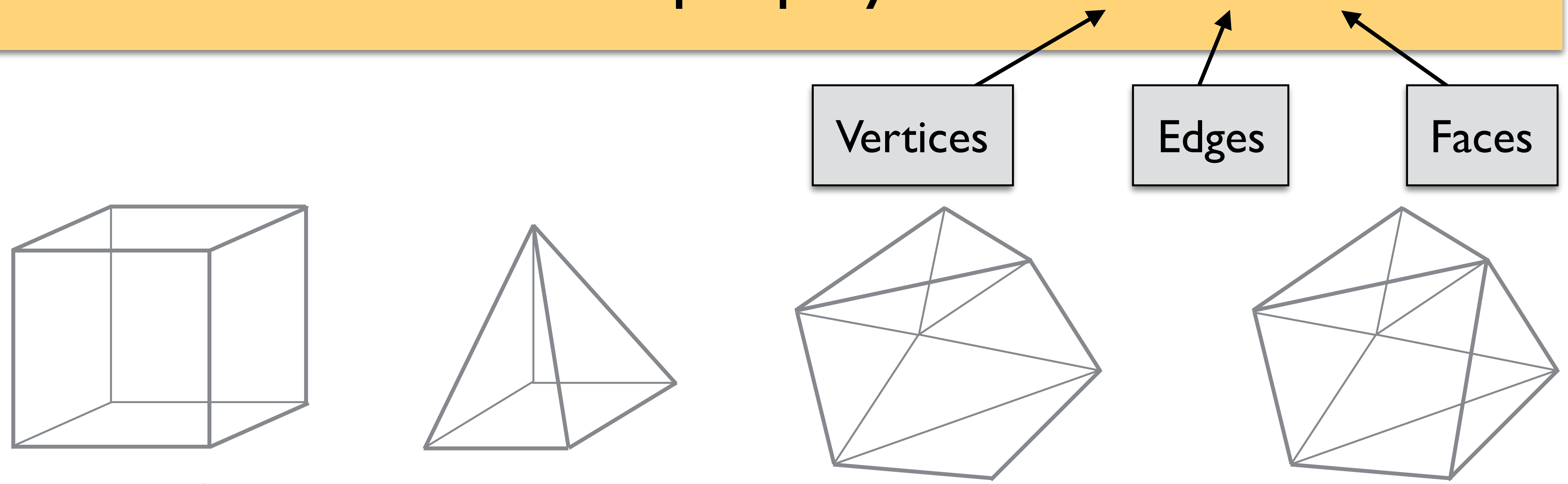


non simple

Polyhedra have flat faces, straight edges and sharp vertices. A simple polyhedron can be deformed into a sphere (no holes).

Boundary Representations

Euler's Formula for simple polyhedra: $V - E + F = 2$



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

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

$V = 7$, $E = 13$, $F = 8$

$V = 7$, $E = 14$, $F = 9$

Euler's formula is **necessary** but **not sufficient** for an object to be a simple polyhedron.
Other requirements:

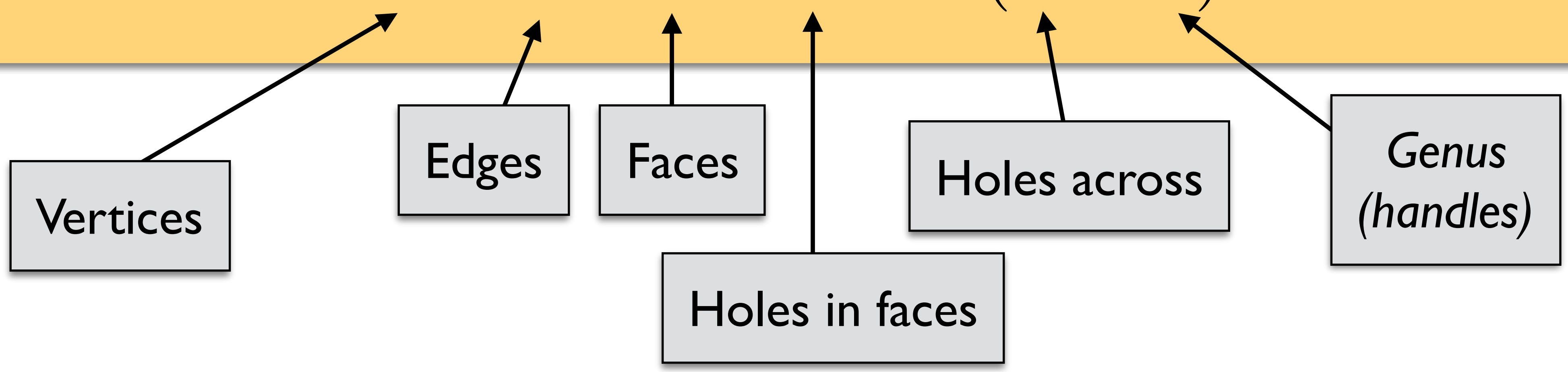
- Each edge must connect two vertices and be shared by exactly two faces;
- At least three edges must meet at a vertex.
- Faces cannot cross each other.

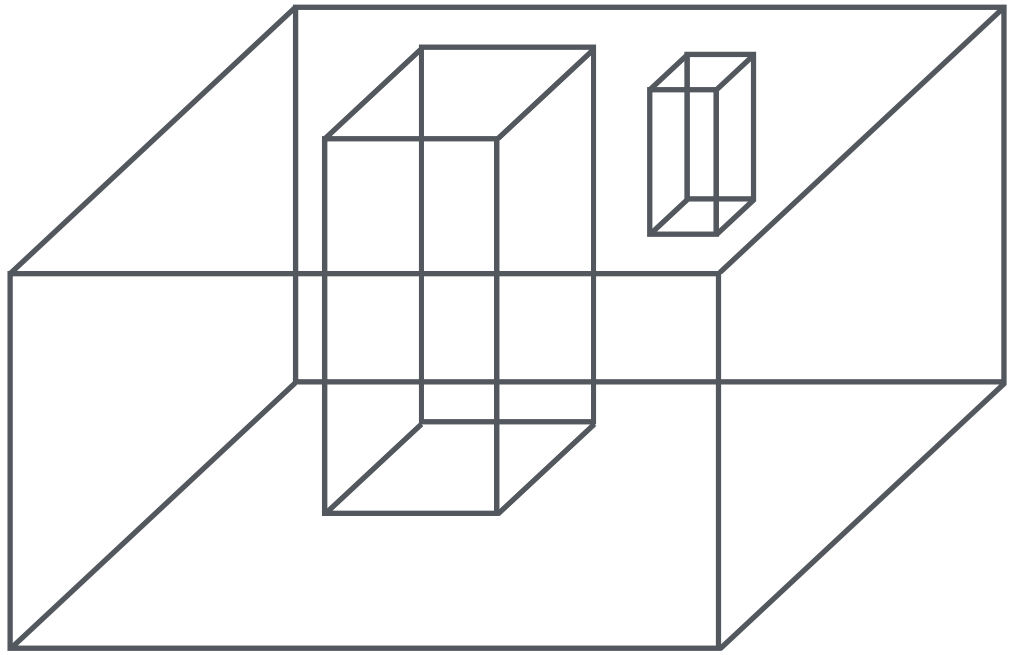
Euler's formula is also valid for curved edges and nonplanar faces (non polyhedra solids)

Boundary Representations

Generalisation of Euler's formula for 2-manifolds with holes

$$V - E + F - H = 2(C - G)$$



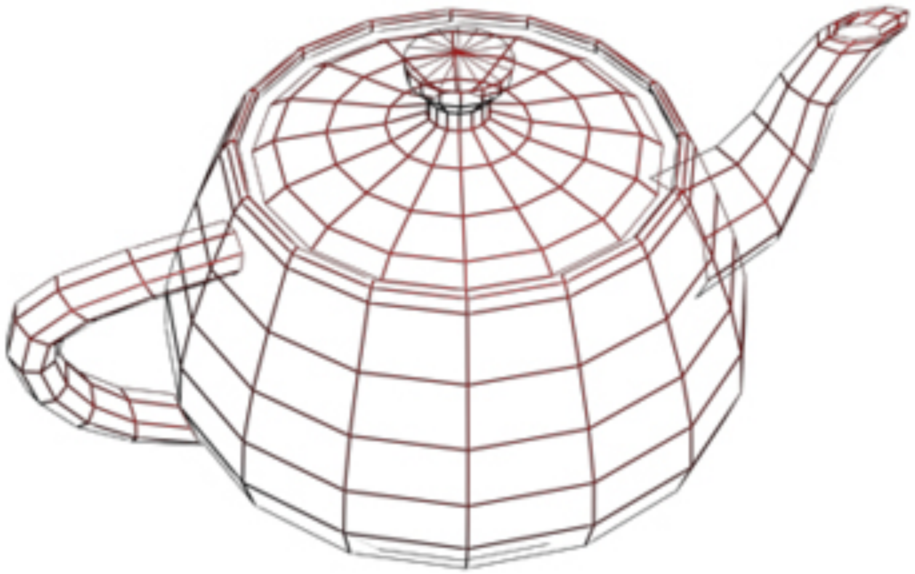


$$V = 24$$
$$E = 36$$
$$F = 15$$
$$H = 3$$
$$C = 1$$
$$G = 1$$

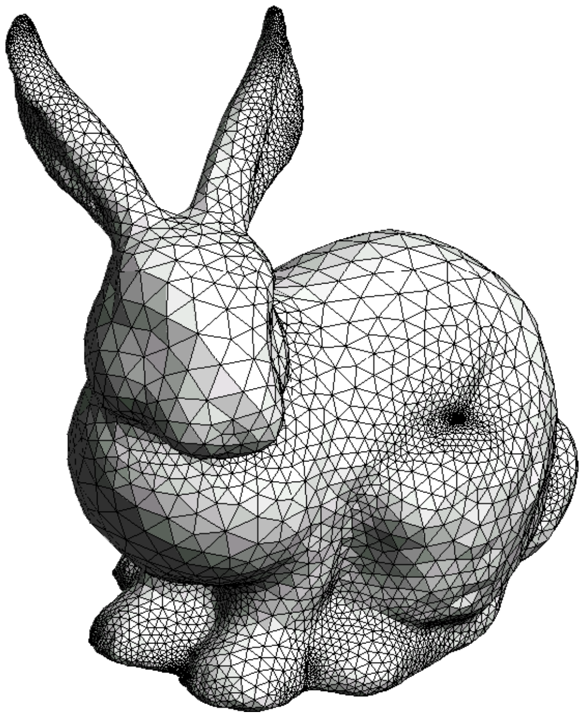
Mesh Data Structures

Mesh Data Structures

- To describe a mesh we need:
  - the locations of all vertices (vertex table)
  - all the edges connecting vertices (explicitly or implicitly defined)
  - All the faces that make up the model (built using vertices or edges)



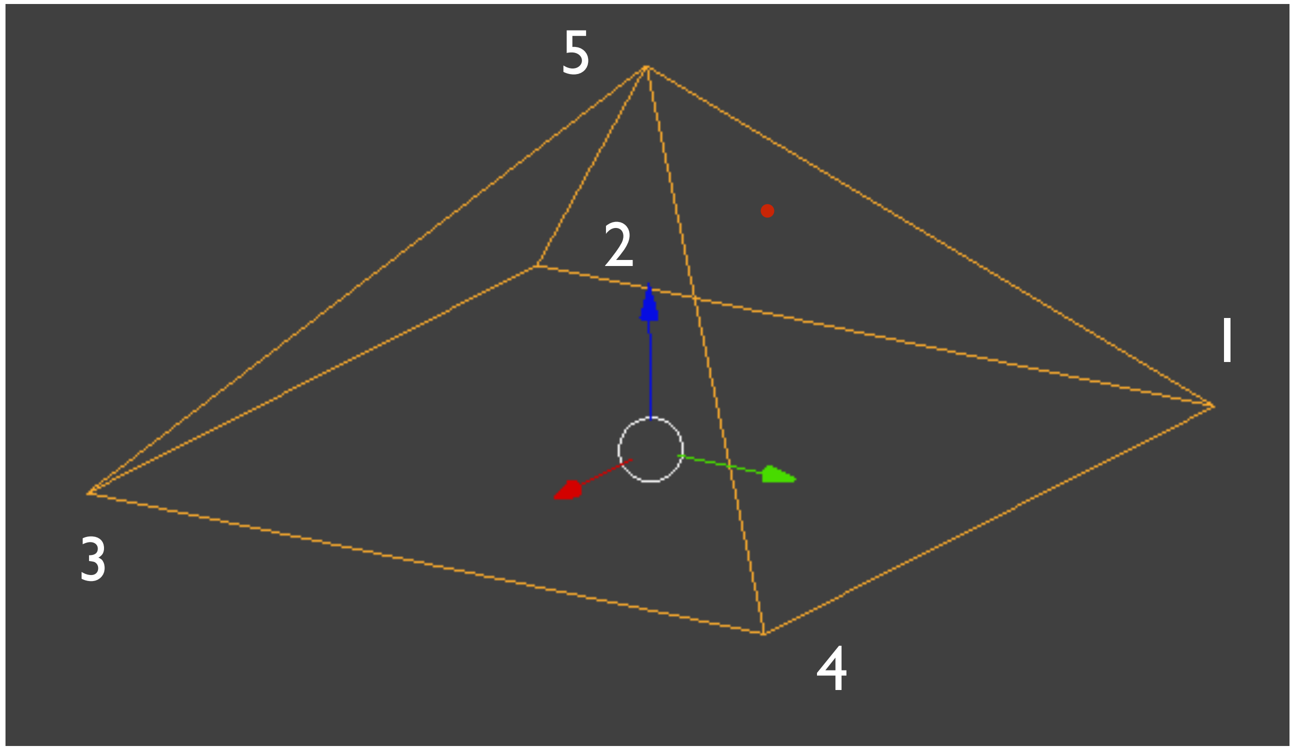
Original teapot by Martin Newell 1975



Original Stanford bunny by Greg Turk and Marc Levoy 1994

Pyramid example (explicit edges)

- Faces point to edges and edges point to vertices. Each vertex has two different ways to be reached, coming from the faces.
- This representation doesn't prevent dangling edges (edges not belonging to any face).
- Also, isolated vertices are allowed.



Topology

Faces	
1	5, 8, 4
2	6, 1, 5
3	6, 2, 7
4	7,3,8
5	4,3,2,1

Edges	
1	(1, 2)
2	(2,3)
3	(3,4)
4	(4,1)
5	(1,5)
6	(5,2)
7	(5,3)
8	(5,4)

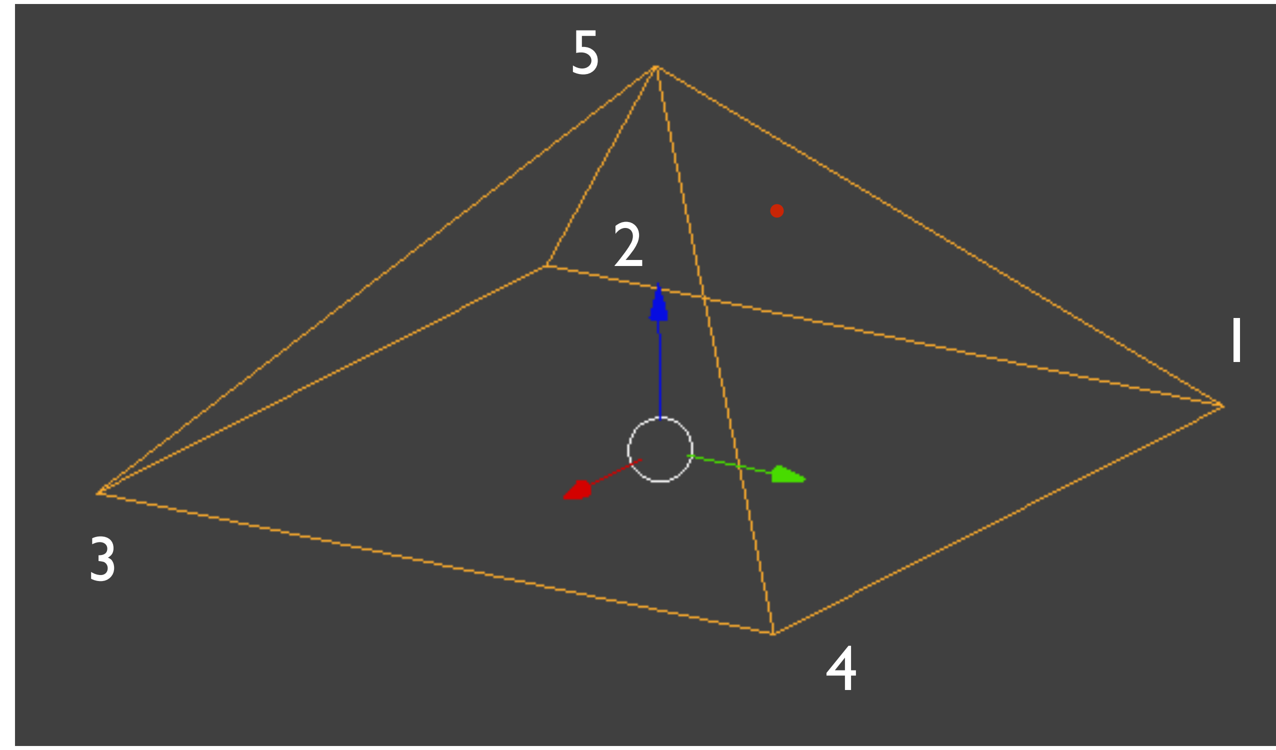
Geometry

Vertices	
1	(1,0,-1)
2	(-1,0,-1)
3	(-1,0,1)
4	(1,0,1)
5	(0,1,0)
6	(1,1,1)

Mesh Data Structures

Pyramid example (implicit edges)

- Faces point to vertices. There is no explicit edge information.
- Edges can be retrieved from faces.
- No dangling edges are possible: all edges are those encoded in faces!
- Still, it is possible to have isolated vertices.



Topology

Faces	
1	1, 5, 4
2	5, 2, 1
3	5, 2, 3
4	5, 3, 4
5	1, 4, 3, 2

Geometry

Vertices	
1	(1,0,-1)
2	(-1,0,-1)
3	(-1,0,1)
4	(1,0,1)
5	(0,1,0)
6	(1,1,1)

edges: (1,4); (4,3); (3,2); (2,1)

Mesh Data Structures

Faces	
1	1, 5, 4
2	5, 2, 1
3	5, 2, 3
4	5, 3, 4
5	1, 4, 3, 2

Faces	
1	1, 5, 4
2	5, 2, 1
3	5, 2, 3
4	5, 3, 4
5	1, 4, 3
6	1, 3, 2

Triangle Mesh!

- Storing faces with an arbitrary number of vertices is not very "computer" friendly.
- faces with more than 3 vertices can present problems (non planar, non convex)

+ Every face can be subdivided into triangles (no ambiguity for planar faces)
+ The triangle is the most simple polygon (always planar)
+ Triangles are usually supported in hardware
+ Triangles are always convex