3D Object Representations

Introduction

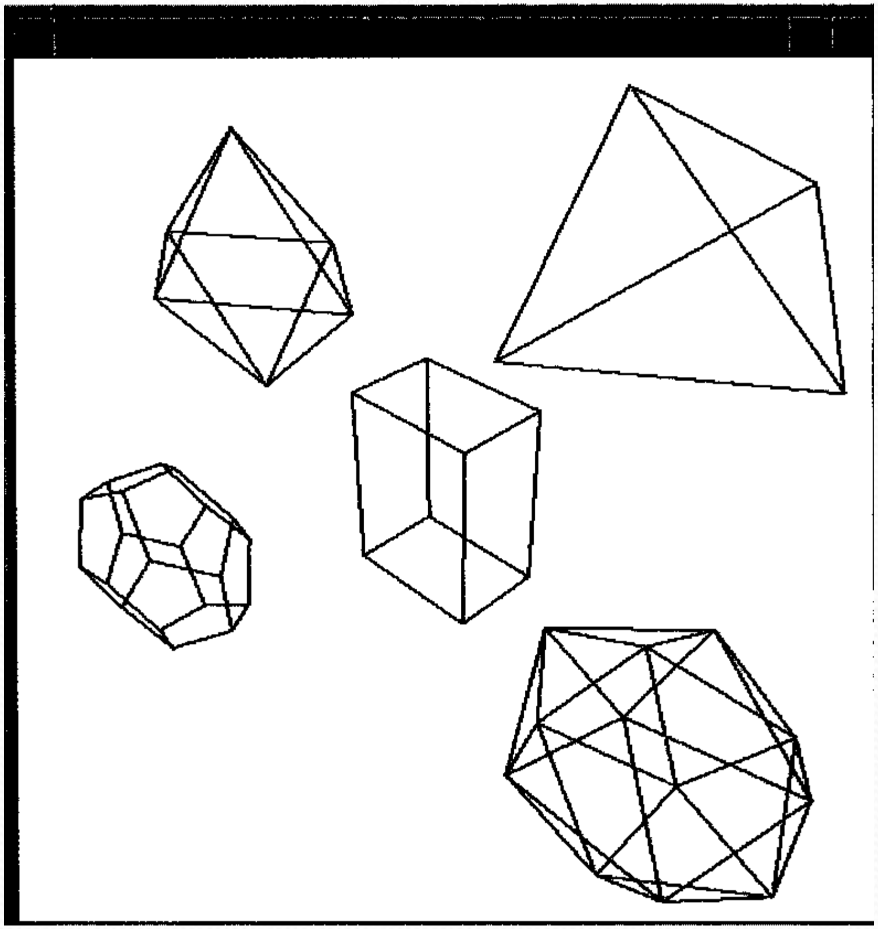
3D Representations

- Basic boundary representations
- Blobby objects
- Various solid modeling principles
- BSP-trees
- Shape Grammars
- L-Grammars (Graftals)
- Particle systems
- Physically based modeling principles

Basic boundary representations

- Polyhedra (a set of surface polygons)
  - triangles, quadrilaterals
- Quadric surfaces (second degree equations)
  - sphere, ellipsoid, torus
- Superquadrics (additional parameters)
  - superellipse (2D), superellipsoid (3D)
- Spline surfaces
  - Bézier, B-spline, rational splines (NURBS)

Polyhedra examples

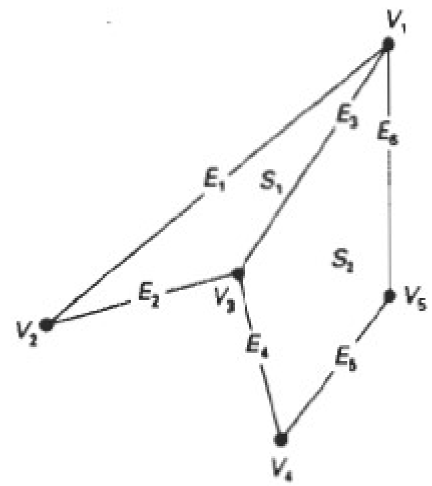


Polygon Surfaces

- Object is represented as a set of surface polygons.
- Simplifies and speeds up surface rendering as surfaces are described as linear equations.
- Polyhedron can be represented
  - By precise surface features
  - Polygon mesh

Polygon Table

- Specify polygon surface as
  - Set of vertices & associated attributes
- Polygon info is stored as data tables
  1. Geometric tables – vertex & orientation
  2. attribute tables – transparency, reflectivity
- Geometric data is stored as 3 lists
  - Vertex table, edge table & polygon table



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

EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Figure 10-2
Geometric data table representation for two adjacent polygon surfaces, formed with six edges and five vertices.

E_1:	V_1, V_2, S_1
E_2:	V_2, V_3, S_1
E_3:	V_3, V_1, S_1, S_2
E_4:	V_3, V_4, S_2
E_5:	V_4, V_5, S_2
E_6:	V_5, V_1, S_2

Figure 10-3
Edge table for the surfaces of Fig. 10-2 expanded to include pointers to the polygon table.

Plane Equations

- Equation of a plane surface
  - $Ax + By + Cz + D = 0$
- (x,y,z) – any point on the plane
- Coefficients A,B,C,D – constants
- To find A,B,C,D solve sets of plane eqns.
- (x_1,y_1,z_1) (x_2,y_2,z_2) , (x_3,y_3,z_3)
  - $(A/D)x_k + (B/D)\, y_k + (C/D)z_k = -1 \quad k=1,2,3$

The solution for this set of equations can be obtained in determinant form, using Cramer's rule, as

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{10-3}$$

Expanding the determinants, we can write the calculations for the plane coefficients in the form

$$\begin{aligned} A &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\ B &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\ C &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\ D &= -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1) \end{aligned} \tag{10-4}$$

Orientation of a plane surface

- Can be described with the normal vector, which has Cartesian components(A,B,C)
- Need to distinguish between two sides of the polygon surface (inside and outside)
- Normal vector will be from inside to outside if
  - polygon vertices are specified in counterclockwise direction &
  - Viewing from the outer side of the plane in a right handed coordinate system.

Normal Vector N Calculations using unit cube

- Method 1:
  - Select 3 vertices in counterclockwise direction
  - Compute A=1,B=0,C=0,D=-1 by substituting these vertices in determinant eqns.
- Method 2:
  - Use cross product
    - N = (V2-V1) × (V3-V1)

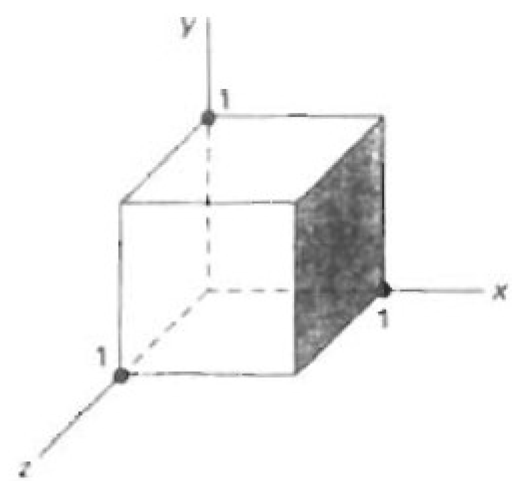


Figure 10-5
The shaded polygon surface of the unit cube has plane equation $x - 1 = 0$ and normal vector $\mathbf{N} = (1, 0, 0)$.

Inequalities of Plane Eqns.

- Plane eqns. are also used to find the position of the spatial points relative to the plane surfaces.

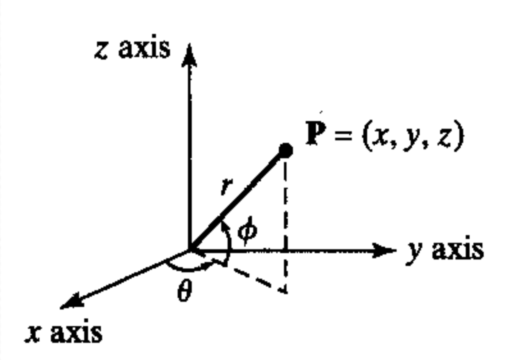
- If (x,y,z) not on surface,
  - $Ax + By + Cz + D \neq 0$

- $Ax + By + Cz + D < 0$ point lies inside the surface
- $Ax + By + Cz + D > 0$ point lies outside the surface

Quadric - Sphere

A point on the sphere surface satisfies:

$x^2 + y^2 + z^2 = r^2$

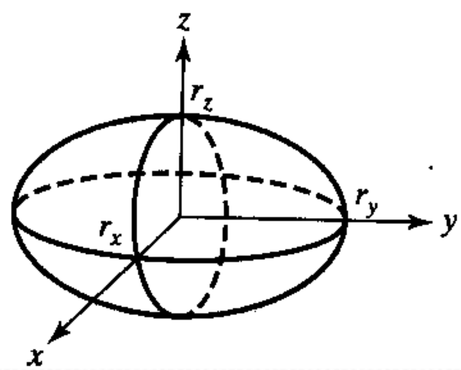


$$x = r\cos\phi\cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r\cos\phi\sin\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r\sin\phi$$

Quadric - ellipsoid



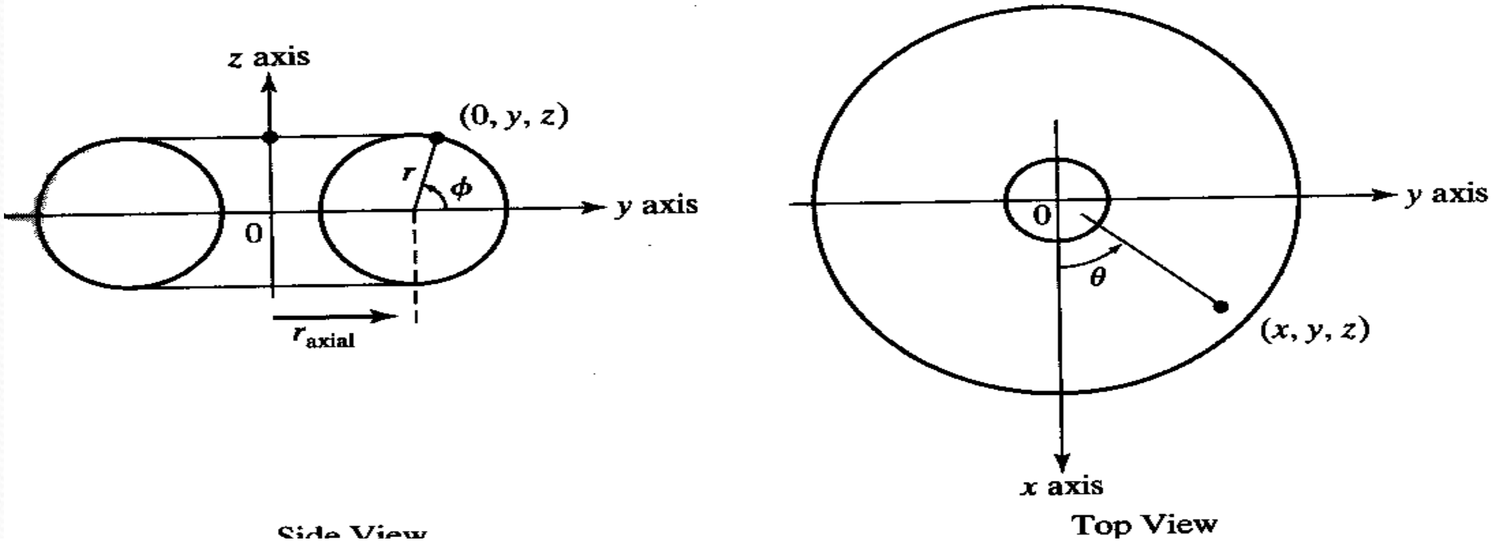
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

$$x = r_x \cos\phi \cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r_y \cos\phi \sin\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r_z \sin\phi$$

Quadric - torus



$$\left[r - \sqrt{\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2}\right]^2 + \left(\frac{z}{r_z}\right)^2 = 1 \qquad (10\text{-}11)$$

where r is any given offset value. Parametric representations for a torus are similar to those for an ellipse, except that angle ϕ extends over 360°. Using latitude and longitude angles ϕ and θ, we can describe the torus surface as the set of points that satisfy

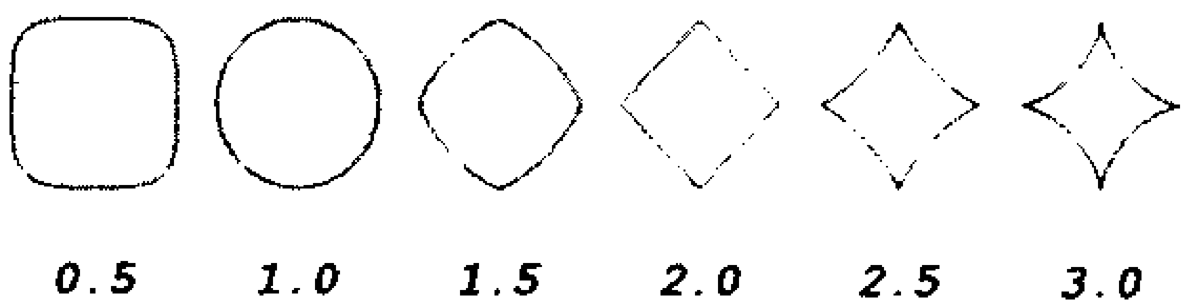
$$\begin{aligned} x &= r_x(r + \cos\phi)\cos\theta, && -\pi \le \phi \le \pi \\ y &= r_y(r + \cos\phi)\sin\theta, && -\pi \le \theta \le \pi \qquad (10\text{-}12) \\ z &= r_z \sin\phi \end{aligned}$$

Superquadric - superellipse

$$\left(\frac{x}{r_x}\right)^{2/s} + \left(\frac{y}{r_y}\right)^{2/s} = 1$$

$$x = r_x \cos^s \theta, \qquad -\pi \le \theta \le \pi$$

$$y = r_y \sin^s \theta$$



Superquadric - Superellipsoid

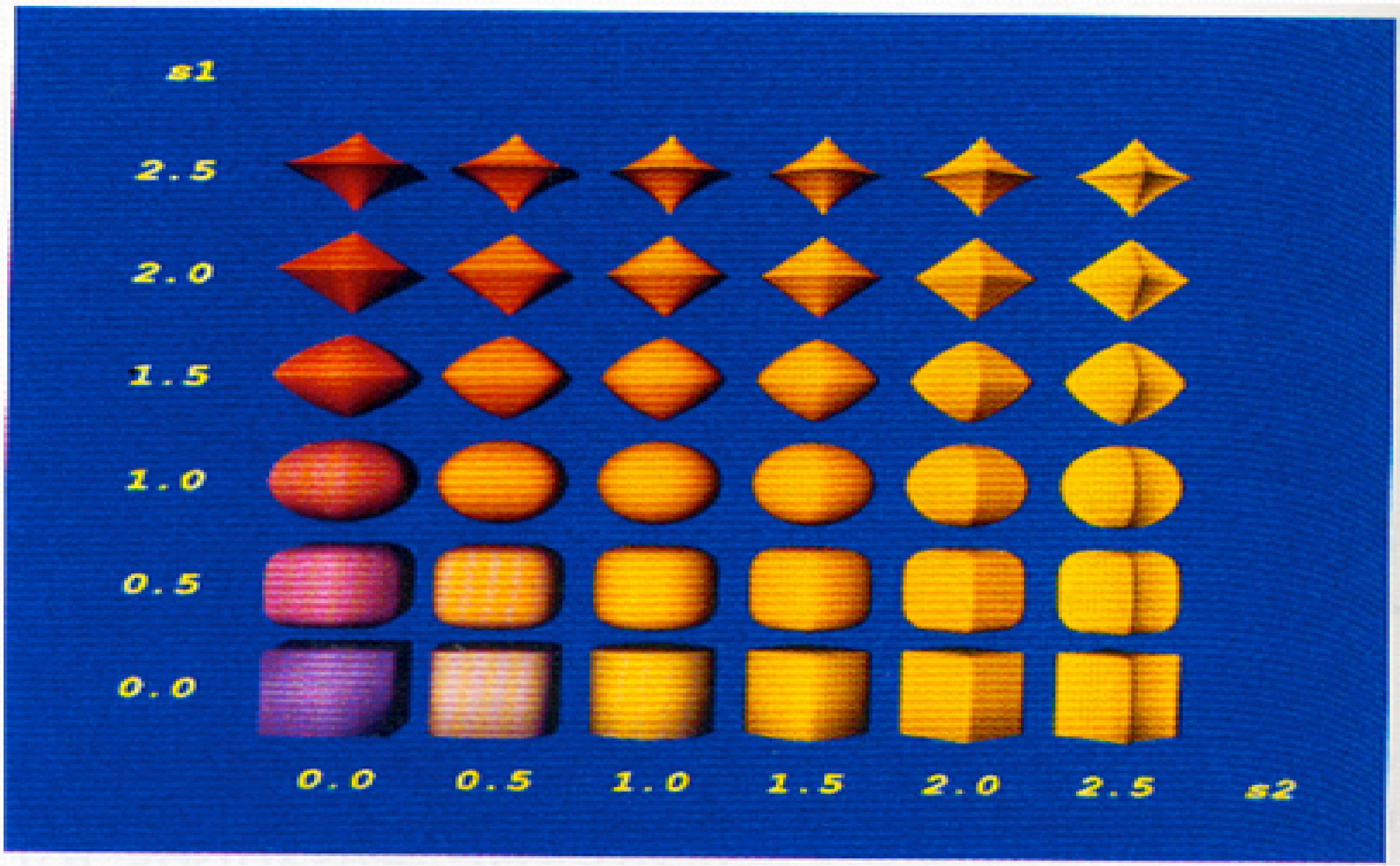
$$\left[\left(\frac{x}{r_x}\right)^{2/s_2} + \left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{z}{r_z}\right)^{2/s_1} = 1$$

$$x = r_x \cos^{s_1}\phi \cos^{s_2}\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r_y \cos^{s_1}\phi \sin^{s_2}\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r_z \sin^{s_1}\phi$$

Superellipsoids



Blobby objects

Objects with changing surface shapes, e.g.

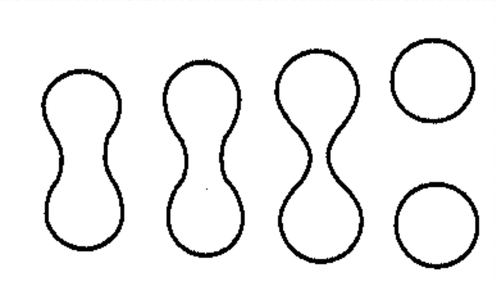
- in certain motions
- in contact with other objects

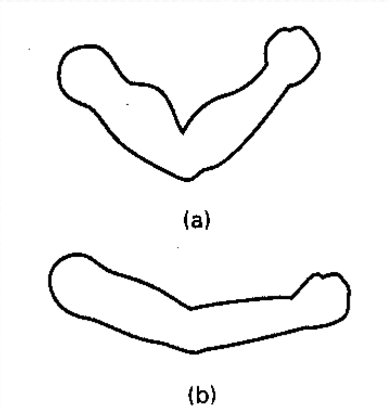
Shape is not fixed, for instance, water droplets and melting objects

Also, various bumps and dents are often used

Usual principle

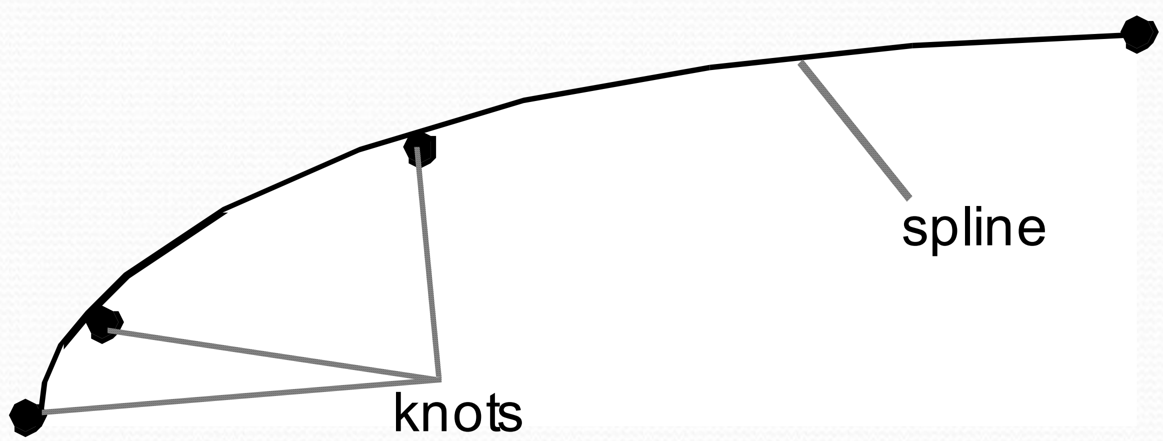
- Fixed volume while shape is changed, e.g. molecules moving apart from each other and human muscles





Splines

- Spline curve – described as a piecewise cubic polynomial function whose 1st and 2nd derivatives are continuous across the various curve sections.
- Used in graphics to design curve and surface shapes, to specify animation path for objects, CAD applications for design of automobile parts.



Interpolation and Approximation Splines

- Control Points – Set of coordinate points that specifies the spline (indicates the general shape of the curve)
- These points are fitted with polynomial functions in one of 2 ways
  - Interpolation- curve passes thro each control point
  - Approximation – does not pass thro any control point.

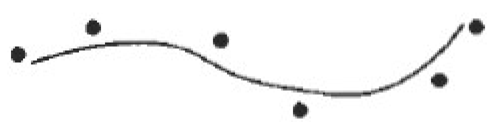


Figure 10-20
A set of six control points approximated with piecewise continuous polynomial sections

Figure 10-19
A set of six control points interpolated with piecewise continuous polynomial

Spline Manipulation

- Initial curve is designed and then manipulated using control points.
- Convex Hull – Convex polygon boundary that encloses a set of control points

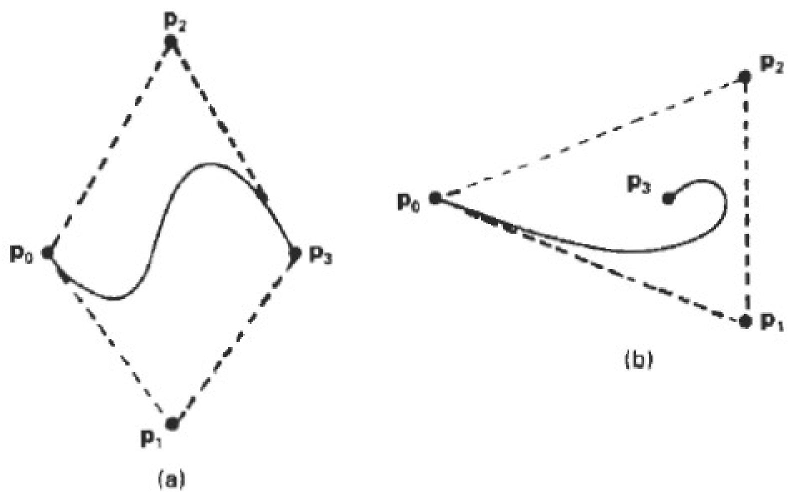


Figure 10-22
Convex-hull shapes (dashed lines) for two sets of control points.

Spline Manipulation

- Control Graph – Polyline connecting a sequence of control points

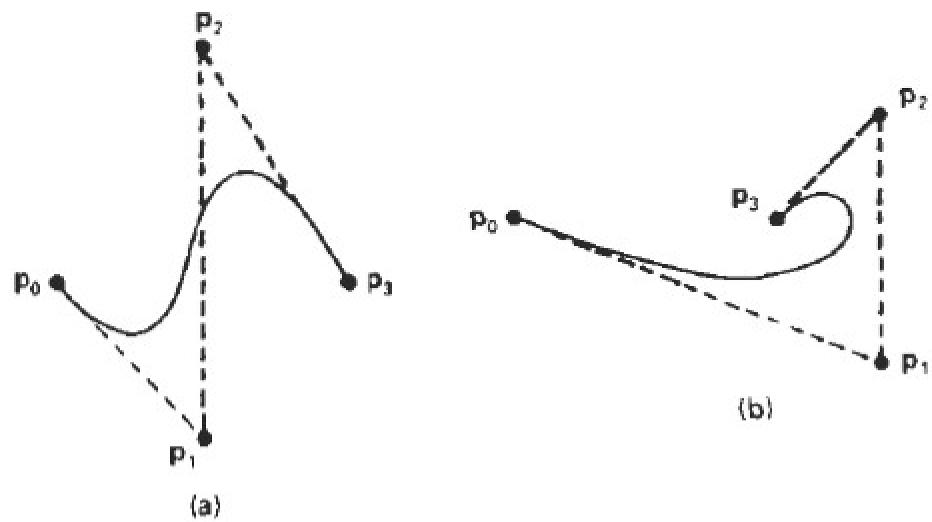


Figure 10-23
Control-graph shapes (dashed lines) for two different sets of control points.

Continuity

- To ensure smooth transition between curve sections
- Each section of the spline is described with a a set of parametric coordinate functions
  - X = x(u)   y = y(u) z = z(u)
- We set continuity by matching the parametric derivates of adjoining curve sections
- Parametric continuity Cx
  - Only P is continuous: C_0
    - Positional continuity – i.e values of x,y,z evaluated at u_2 of first section = values of x,y,z evaluated at u_1 of next section
  - P and first derivative dP/du are continuous: C_1
    - Tangential continuity
  - P + first + second: C_2
    - Curvature continuity

Continuity

- Geometric continuity Gx
  - Parametric derivates should be proportional.
- Zero Order G0
  - Two sections must have the same coordinate position at boundary point
- First Order G1
- Second order G2

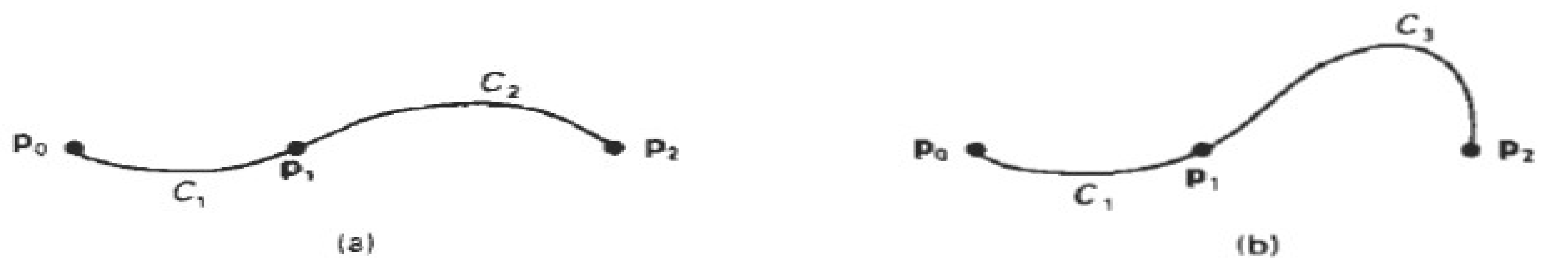
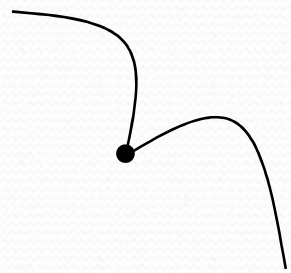


Figure 10-25
Three control points fitted with two curve sections joined with (a) parametric continuity and (b) geometric continuity, where the tangent vector of curve C_3 at point $\mathbf{p}_1$ has a greater magnitude than the tangent vector of curve C_1 at $\mathbf{p}_1$.

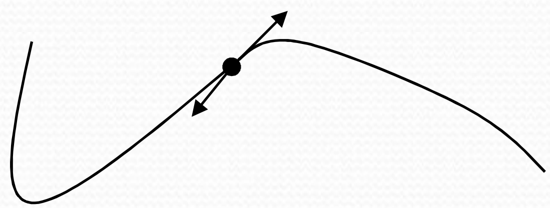
Order of continuity

$G^{(0)}$ $C^{(0)}$ 0^{th} order continuity



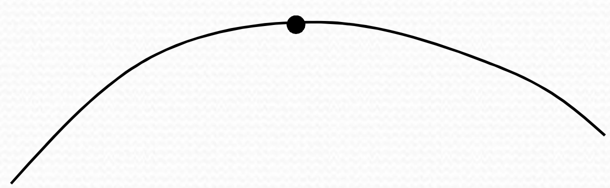
(a)

$G^{(1)}$ $C^{(1)}$ 1^{st} order continuity



(b)

$G^{(2)}$ $C^{(2)}$ 2^{nd} order continuity



(c)

Spline Specifications

- 3 Methods for specifying spline representation
  - State set of boundary conditions imposed on the spline
  - State the matrix of the spline
  - State the set of blending functions that determine how specified geometric constraints are combined to calculate positions along the curve path.
  - Suppose, the x coordinate of a spline section has
    - $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 <= u <= 1$
  - Boundary conditions may be set on endpoints $x(0)$ and $x(1)$
  - Using that determine a_x, b_x, c_x, d_x.

- From the boundary conditions, obtain the matrix

$$x(u) = [u^3\ u^2\ u\ 1]\begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

$$= \mathbf{U} \cdot \mathbf{C}$$

We can write the boundary conditions in the matrix form

$$\mathbf{C} = \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}}$$

Where M geom is a four element col. Matrix containing the geometric constrints

- So we can say,

$$x(u) = \mathbf{U} \cdot \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}}$$

We can expand this to obtain a polynomial representation for Coordinate x in terms of geometric constraint parameters.

$$x(u) = \sum_{k=0}^{3} g_k \cdot BF_k(u)$$

Constructive Solid Geometry (CSG)

A new object is constructed by using the following set operations

- *union*
- *intersection*
- *difference*

between any pair of objects

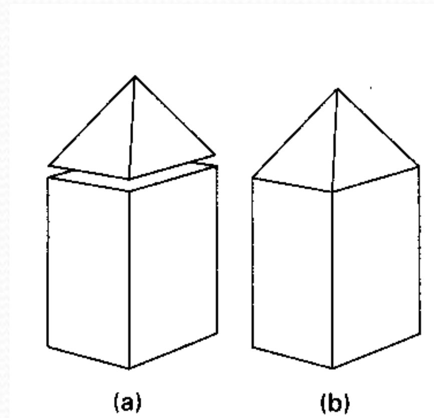
Basic 3D objects

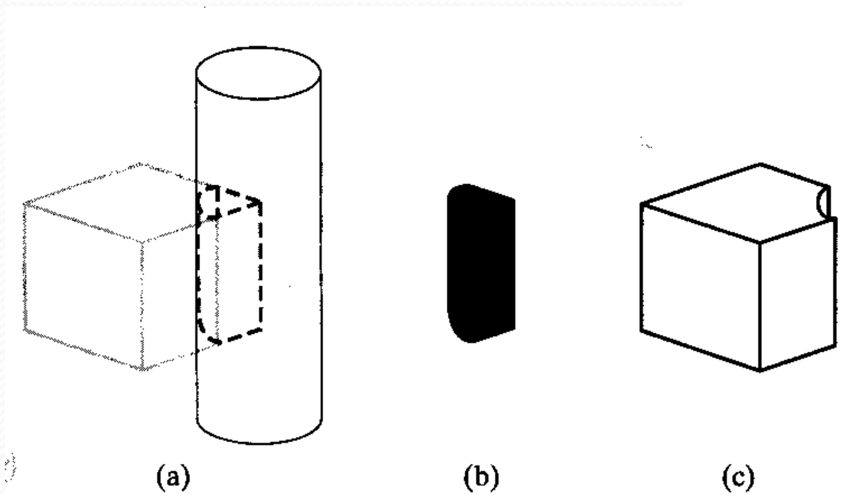
Typical 3D primitives (predefined or modelled):

- block (box, parallelepiped)
- cylinder
- cone
- sphere
- various closed spline surfaces

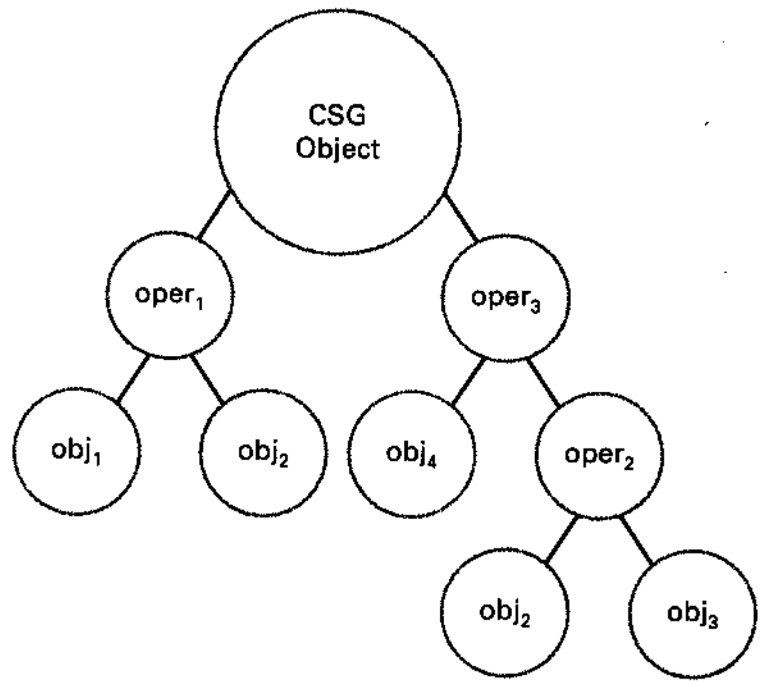
Operation examples

Select any two objects and a set operation => a new object





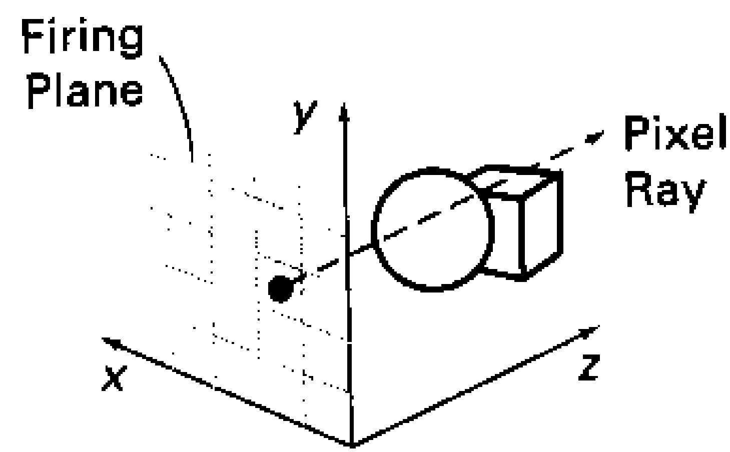
CSG - Binary tree representation



Ray-casting

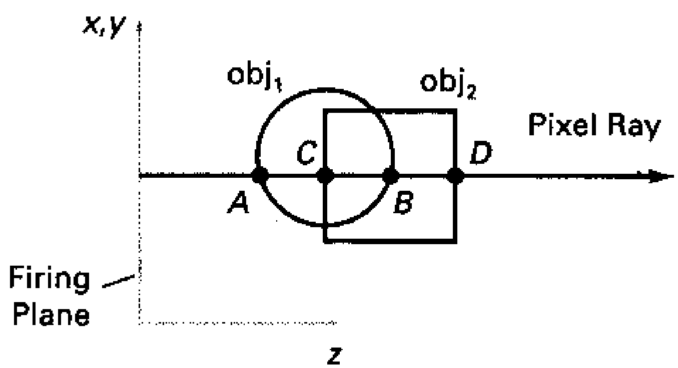
A technique to implement CSG operations:

- an xy-plane (cp the pixel plane) in the display, called *firing plane*, is defined
- parallel rays from the pixel positions are fired in the z-direction



Ray-casting,cont'd

- intersection points with object surfaces are calculated
- these points are sorted on increasing distances from the firing plane
- surface limits are then determined by the current set operation



(a)

Operation	Surface Limits
Union	A, D
Intersection	C, B
Difference ($obj_2 - obj_1$)	B, D

(b)

Octrees

A simple technique to represent also the interior structure of a 3D object

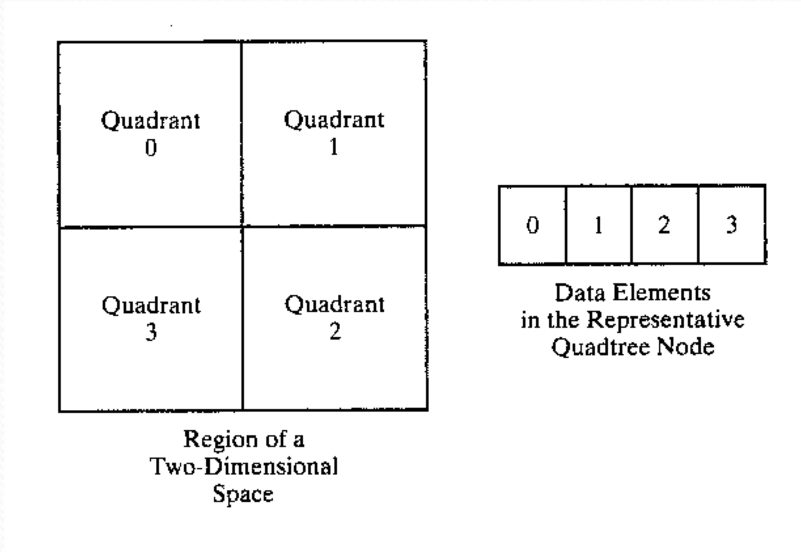
Each node of the octree represents a particular 3D section of an object; coherence will be used to save memory

Octrees - an extension of *quadtrees* from 2D, but quadtrees are easier to illustrate, although still based on the same principle

Quadtrees

Assume that the area to represent is quadratic

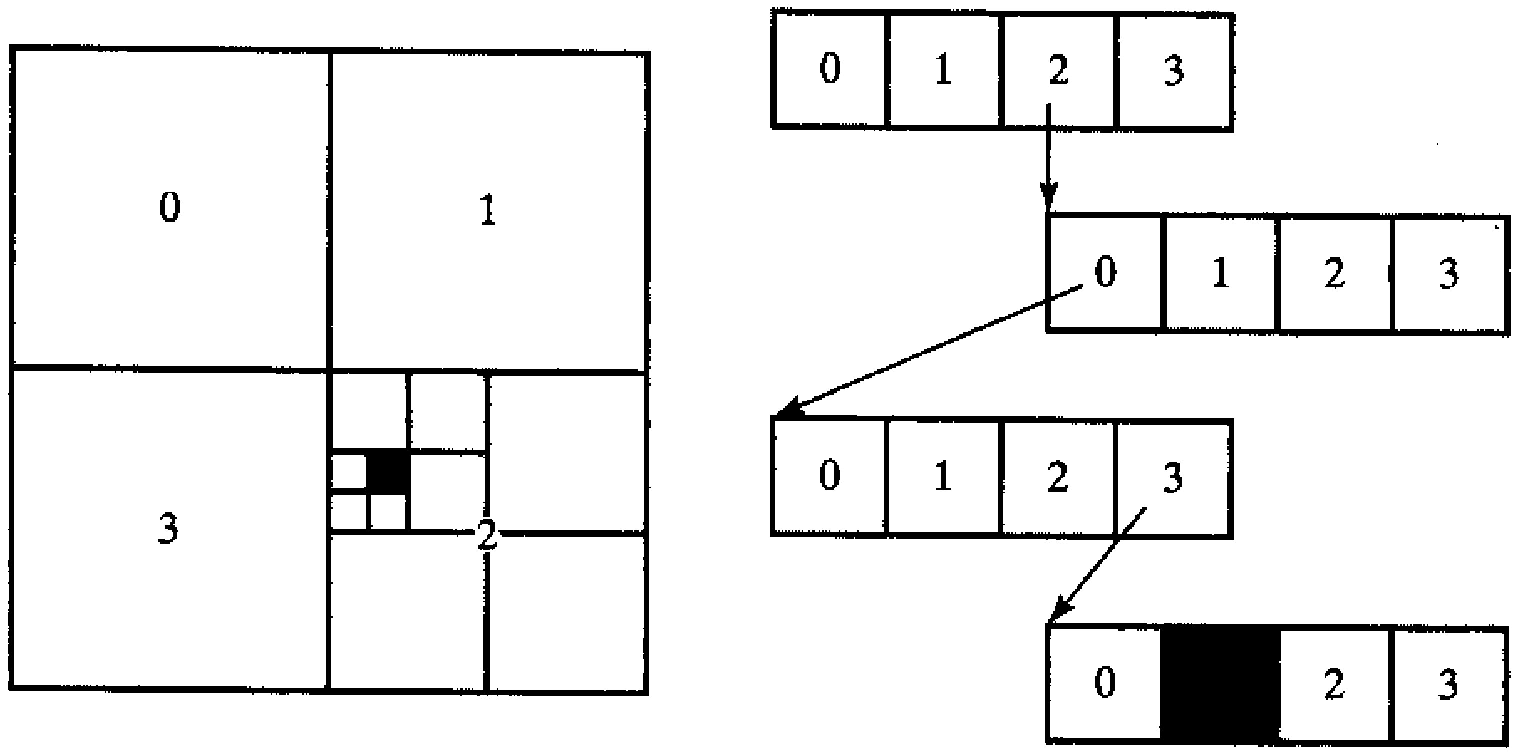
- a quadtree is created by successively dividing the given area into four quadrants
- each node of the tree has four elements; one for each quadrant



Quadtrees, cont'd

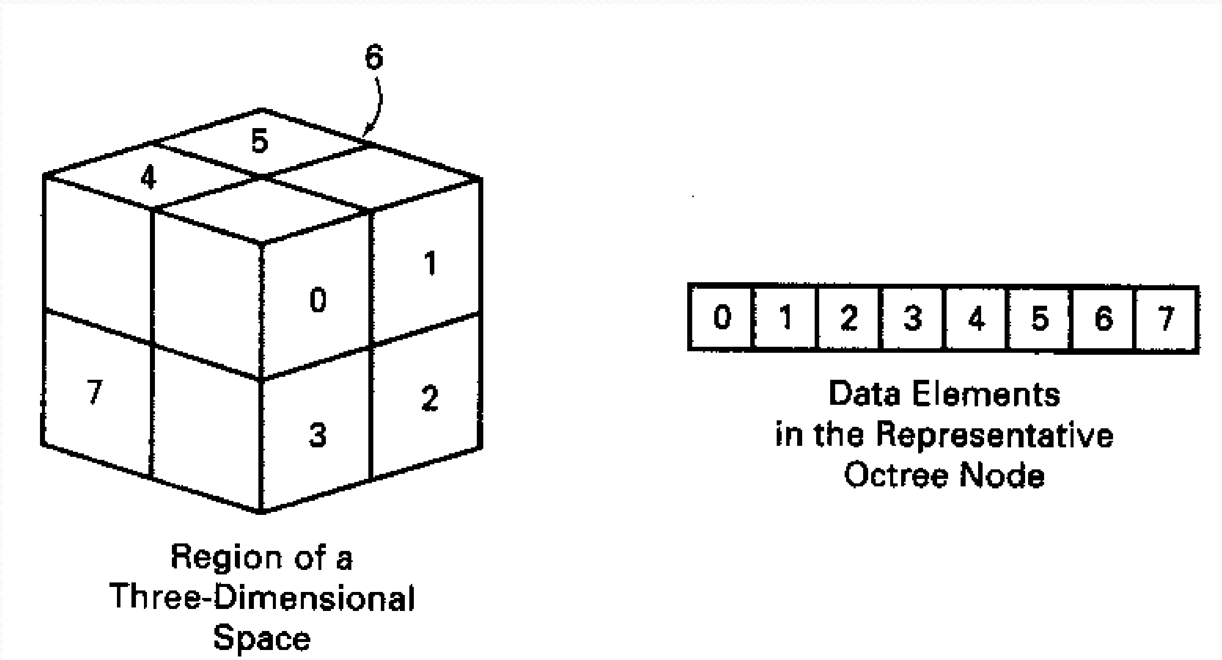
- if a homogeneous quadrant (same intensity on all pixels in it), the intensity is stored in the corresponding data element of the node, and a flag is set to indicate a homogeneous quadrant
- if a heterogeneous quadrant, it is divided into 4 subquadrants, the flag is set to indicate a heterogeneous quadrant, and instead of the intensity, a pointer to the next node in the quadtree is stored

Quadtree - example



Octree

A 3D space (assumed to be a cube) is divided into eight octants and stores eight data elements in each node of the tree



Octrees, cont'd

Individual elements in a 3D space are called volume elements, *voxels*, in which other properties than intensity such as material and density, can be included

When all voxels in an octant have the same intensity (homogeneous), this intensity is stored in the corresponding data element of the node

Each heterogeneous octant is divided into 8 new suboctants with the corresponding data element pointing at the next node of the tree

Comments on octrees

Empty regions are represented by ”void”

If the current space is not a cube, it can easily be enclosed by one

To locate the visible parts of an object, first possible front octants are examined; if visible, the hidden octants don’t need to be examined (cp VSD)

BSP trees

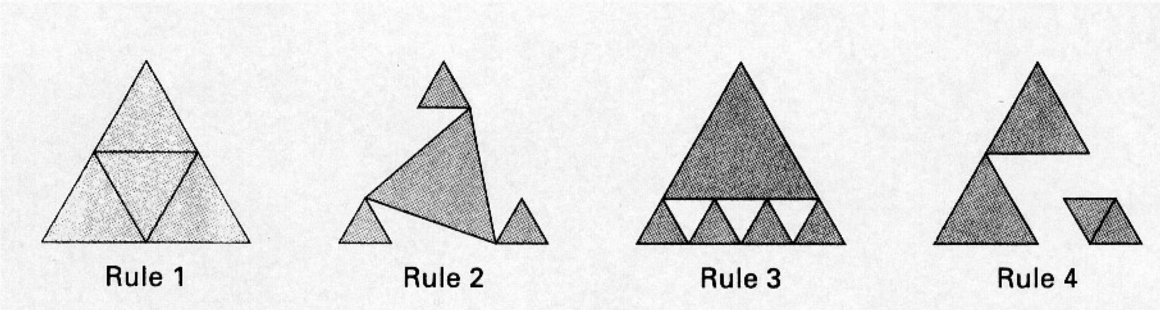
BSP - *Binary Space Partitioning*

Similar to octrees, but successive subdivision into two partitions in each step => a binary tree representation

In each subdivision, the scene is divided into two parts by a plane that can be defined anywhere and with arbitrary orientation

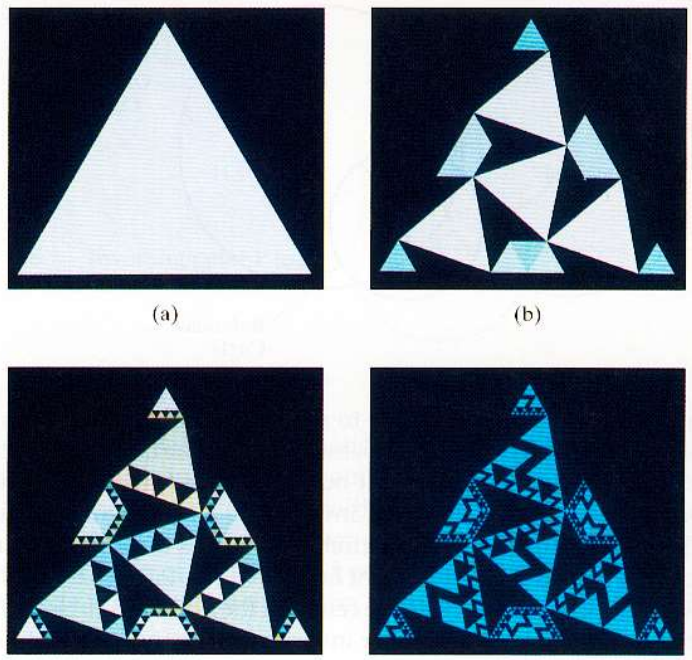
Shape Grammars

- Sets of production rules that can be used on an initial object to add details based on certain rules
- Transformations can be used to
  - change the shape of an object
  - add surface details/textures of an object



Grammars,cont'd

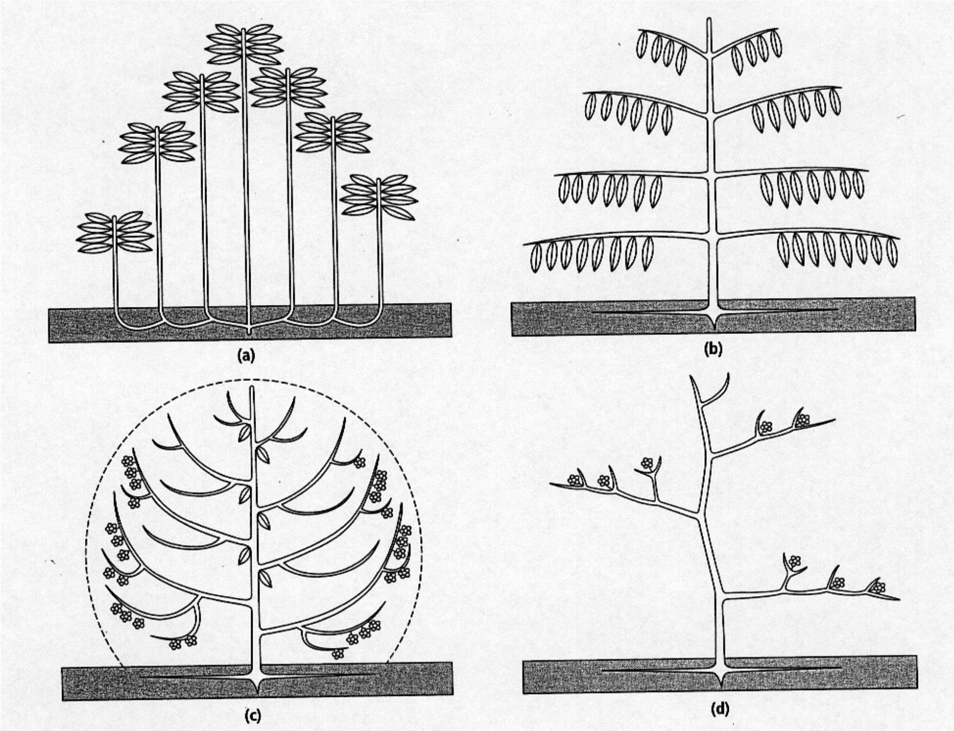
- Use the set of production rules at each step to change a given initial object
- Transformations corresponding t the rules can be generated automatically based on a picture with a special production-rule editor
- Each rule can be described by giv the initial and final shapes

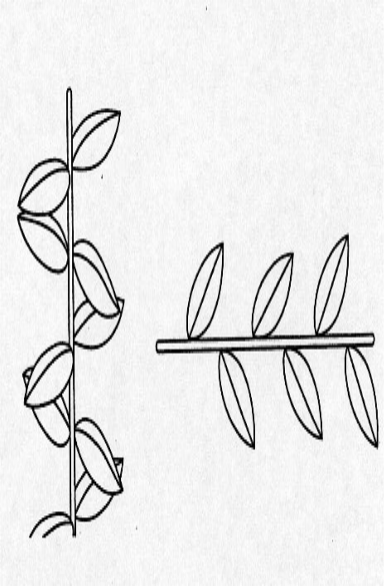

(a) (b) (c) (d)

L-Grammars, Graftals

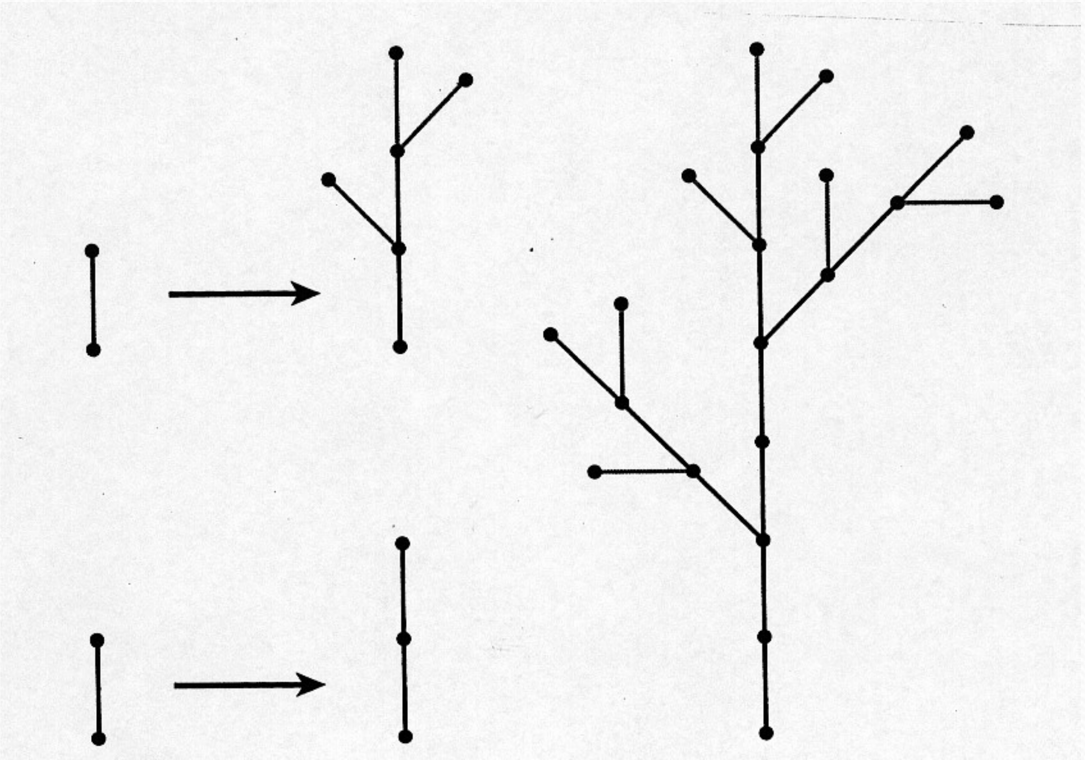
- Used to generate plants, trees and similar objects
- For instance, a tree:
  - a trunk with branches and leaves
  - can be modified with rules for
    - particular connections of the branches
    - leaves on individual branches
    - placing in suitable positions
- There are special plant-generator packages/editors

Graftals examples





Graftal rules,example



Ex with Plant Editor

Physically Based Modeling

Non-rigid objects can be represented by methods that describe the behaviour as a mix of external and internal forces

Typical objects are ropes, cloth material, softballs.

A strict description consider, for example:

* how a stiff object effects a soft material, i.e. textile material

* the interaction between the threads in the textile

Textile on chair

Cloth examples

Models of cotton, wool and polyester using energy-function minimization

In animation

Physically based modeling also improves the realism in animation providing a better description of motion paths.

Instead of spline paths and kinematics, dynamical equations with forces and accelerations give much more realistic motions.

Final example