3

OBJECT REPRESENTATION & MODELLING [15HR]

3.1 3D object representation

Graphical scenes can contain many different kinds of objects like trees, flowers, rocks, waters...etc. No single method can be used to describe objects that will include all features of those different materials. To produce realistic display of scenes, we need to use representations that accurately model object characteristics.

- Simple Euclidean objects like polyhedrons and ellipsoids can be represented by polygon and quadric surfaces.
- Spline surface are useful for designing aircraft wings, gears and other engineering objects.
- Procedural methods and particle systems allow us to give accurate representation of clouds, clumps of grass, and other natural objects.
- Octree encodings are used to represent internal features of objects such as medical CT images.

Representation schemes for solid objects are often divided into two broad categories:

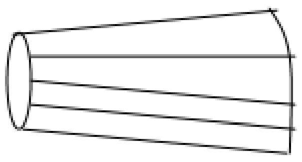
1. **Boundary representation:** It describes a 3D object as a set of polygonal surfaces, separate the object interior from environment.
2. **Space-partitioning representation:** It is used to describe interior properties, by partitioning the spatial region, containing an object into a set of small, non-overlapping, contiguous solids. E.g. 3D object as octree representation.

Boundary representation

Each 3D object is supposed to be formed its surface by collection of polygon facets and spline patches. Some of the boundary representation methods for 3D surface are:

1. Polygon Surfaces:

It is most common representation for 3D graphics object. In this representation, a 3D object is represented by a set of surfaces that enclose the object interior. Many graphics systems use this method. Set of polygons are stored for object description. This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.



A 3D object represented by polygons

The polygon surfaces are common in design and solid-modeling applications, since wire frame display can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.

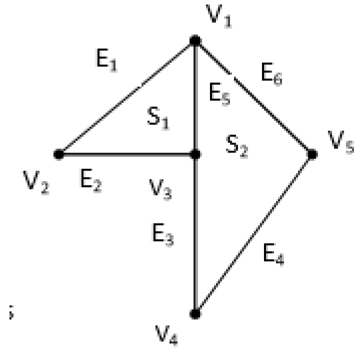
Polygon Table:

A polygon surface is specified with a set of vertex co-ordinates and associated attribute parameters. A convenient organization for storing geometric data is to create 3 lists:

- A vertex table : Vertex table stores co-ordinates of each vertex in the object.

- An edge table : The edge table stores the edge information of each edge of polygon facets.
- A polygon surface table : The polygon surface table stores the surface information for each surface i.e. each surface is represented by edge lists of polygon.

Consider the surface contains polygonal facets as shown in figure (only two polygons are taken here)



S_1 and S_2 are two polygon surface that represent the boundary of some 3D object. Foe storing geometric data, we can use following three tables:

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_4
E_4: V_4,V_5
E_5: V_1,V_3
E_6: V_5, V_1

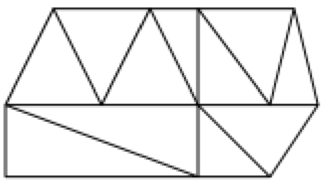
POLYGON SURFACE TABLE
S_1: E_1,E_2,E_3
S_2: E_3,E_4,E_5,E_6

The object can be displayed efficiently by using data from tables and processing them for surface rendering and visible surface determination.

Polygon Meshes:

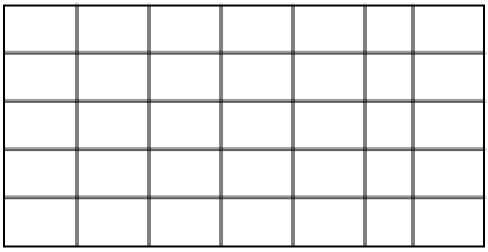
A polygon mesh is collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices and a polygon is a closed sequence of edges. An edge can be shared by two polygons and a vertex is shared by at least two edges.

When object surface is to be tiled, it is more convenient to specify the surface facets with a mesh function. One type of polygon mesh is triangle strip. This function produce n-2 connected triangles.



Triangular Mesh

Another similar function is the quadrilateral mesh, which generates a mesh of (n-1) by (m-1) quadrilaterals, given the co-ordinates for an nxm array of vertices.



6 by 8 vertices array , 35 element quadrilateral mesh

If the surface of 3D object is planer, it is comfortable to represent surface with meshes.

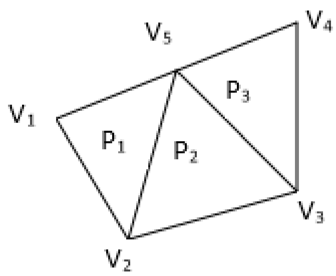
Representing polygon meshes:

In explicit representation, each polygon is represented by a list of vertex co-ordinates.

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots , (x_n, y_n, z_n))$$

The vertices are stored in order traveling around the polygon. There are edges between successive vertices in the list and between the last and first vertices. For a single polygon it is efficient but for polygon mesh it is not space efficient since no of vertices may duplicate.

So another method is to define polygon with pointers to a vertex list. So each vertex is stored just once, in vertex list $V = \{v_1, v_2, \ldots, v_n\}$. A polygon is defined by list of indices (pointers) into the vertex list e.g. A polygon made up of vertices 3, 5, 7, 10 in vertex list be represented as $P_1 = \{3, 5, 7, 10\}$



Representing polygon mesh with each polygon as vertex list.

$$P_1 = \{v_1, v_2, v_5\}$$
$$P_2 = \{v_2, v_3, v_5\}$$
$$P_3 = \{v_3, v_4, v_5\}$$

Here most of the vertices are duplicated so it is not efficient.

Representation with indexes into a vertex list:

$$V = \{v_1, v_2, v_3, v_4, v_5\} = \{(x_1, y_1, z_1), \ldots\ldots (x_5, y_5, z_5)\}$$
$$P_1 = \{1,2,3\}$$
$$P_2 = \{2,3,5\}$$
$$P_3 = \{3,4,5\}$$

Defining polygons by pointers to an edge list:

In this method, we have vertex list V, represent the polygon as a list of pointers not to the vertex list but to an edge list. Each edge in edge list points to the two vertices in the vertex list. Also to one or two polygon, the edge belongs.

Hence we describe polygon as:

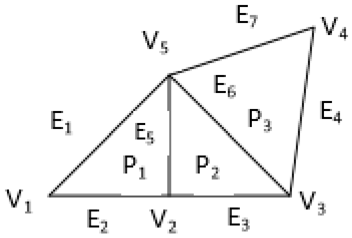
$P = (E_1, E_2, \ldots , E_n)$

and edge as:

$E = (V_1, V_2, P_1, P_2)$

Here, if edge belongs to only one polygon, either then P_1 or P_2 is null.

For the mesh given below,



$$V = \{v_1, v_2, v_3, v_4, v_5\} = \{(x_1, y_1, z_1), \ldots\ldots(x_5, y_5, z_5)\}$$

$$E_1 = (V_1, V_5, P_1, N)$$
$$E_2 = (V_1, V_2, P_1, N)$$
$$E_3 = (V_2, V_3, P_2, N)$$
$$E_4 = (V_3, V_4, P_3, N)$$
$$E_5 = (V_2, V_5, P_1, P_2)$$
$$E_6 = (V_3, V_5, P_1, P_3)$$

Here N represents Null.

$$E_7 = (V_4, V_5, P_3, N)$$
$$P_1 = (E_1, E_2, E_3)$$
$$P_2 = (E_3, E_6, E_5)$$
$$P_3 = (E_4, E_7, E_6)$$

Polygon Surface: Plane Equation Method:

Plane equation method is another method for representation the polygon surface for 3D object. The information about the spatial orientation of object is described by its individual surface, which is obtained by the vertex co-ordinates and the equation of each surface. The equation for a plane surface can be expressed in the form:

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

Where (x, y, z) is any point on the plane, and A, B, C, D are constants describing the spatial properties of the plane. The values of A, B, C, D can be obtained by solving a set of three plane equations using coordinate values of 3 noncollinear points on the plane.

Let (x_1, y_1, z_1), (x_2, y_3, z_2) and (x_3, y_3, z_3) are three such points on the plane, then,

$Ax_1 + By_1 + Cz_1 + D = 0$

$Ax_2 + By_2 + Cz_2 + D = 0$

$Ax_3 + By_3 + Cz_3 + D = 0$

The solution of these equations can be obtained in determinant from using Cramer's rule as:-

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

For any points (x, y, z)

If $Ax + By + Cz + D \neq 0$, then (x, y, z) is not on the plane.
If $Ax + By + Cz + D < 0$, then (x, y, z) is inside the plane i. e. invisible side
If $Ax + By + Cz + D > 0$, then (x, y, z) is lies out side the surface.

2. Quadric Surface

Quadric Surface is one of the frequently used 3D objects surface representation. The quadric surface can be represented by a second degree polynomial. This includes:

1. Sphere: For the set of surface points (x, y, z) the spherical surface is represented as: $x^2+y^2+z^2 = r^2$, with radius r and centered at co-ordinate origin.
2. Ellipsoid: $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$, where (x, y, z) is the surface points and a, b, c are the radii on X, Y and Z directions respectively.
3. Elliptic parboiled: $\frac{x^2}{a^2}+\frac{y^2}{b^2}=z$
4. Hyperbolic parboiled : $\frac{x^2}{a^2}-\frac{y^2}{b^2}=z$
5. Elliptic cone : $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=0$
6. Hyperboloid of one sheet: $\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$
7. Hyperboloid of two sheet: $\frac{x^2}{a^2}-\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$

3. Wireframe representation:

In this method 3D objects is represented as a list of straight lines, each of which is represented by its two end points (x_1, y_1, z_1) and (x_2, y_2, z_2). This method only shows the skeletal structure of the objects. It is simple and can see through the object and fast method. But independent line data structure is very inefficient i.e. don't know what is connected to what. In this method the scenes represented are not realistic.

4. Blobby objects:

Some objects don't maintain a fixed shape but change their surface characteristics in certain motions or when proximity to another objects e.g. molecular structures, water droplets, other liquid effects, melting objects, muscle shaped in human body etc. These objects can be described as exhibiting "blobbiness" and are referred as blobby objects.

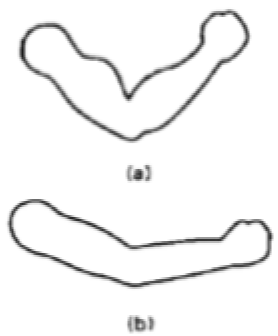


Fig: Blobby muscle shapes in a human arm

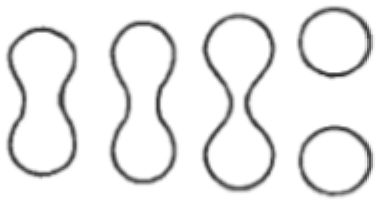


Fig: Molecular bonding (stretching and contracting into spheres)

Several models have been developed for representing blobby objects as distribution functions over a region of space. One way is to use Gaussian density function or bumps. A surface function is defined as:

$$f(x, y, z) = \sum_k b_{k0} e^{-a_k r_k^2} - T = 0$$

Where $r_k = \sqrt{x_k^2 + y_k^2 + z_k^2}$, T = Threshold, a and b are used to adjust amount of blobbiness.

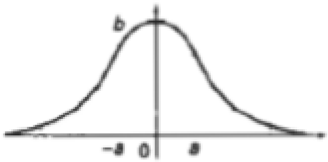


Fig: A three-dimensional Gaussian bump centered at position 0, with height band standard deviation a.

Fig: A composite blobby object formed with four Gaussian bumps

Other method for generating blobby objects uses quadratic density function as:

$$f(r) = \begin{cases} b(1 - 3r^2/d^2), & \text{if } 0 < r \leq d/3 \\ \frac{3}{2}b(1 - r/d)^2, & \text{if } d/3 < r \leq d \\ 0, & \text{if } r > d \end{cases}$$

<u>Advantages:</u>

- Can represent organic, blobby or liquid line structures.
- Suitable for modelling natural phenomenon like water, human body.
- Surface properties can be easily derived from mathematical equations.

<u>Disadvantages:</u>

- Requires expensive computation.
- Requires special rendering engine.
- Not supported by most graphics hardware.

5. Spline Representation:

A Spline is a flexible strips used to produce smooth curve through a designated set of points. A curve drawn with these set of points is spline curve. Spline curves are used to model 3D object surface shape smoothly.

Mathematically, spline are described as piece-wise cubic polynomial functions. In computer graphics, a spline surface can be described with two set of orthogonal spline curves. Spline is used in graphics application to design and digitalize drawings for storage in computer and to specify animation path. Typical CAD application for spline includes the design of automobile bodies, aircraft and spacecraft surface etc.

Interpolation and approximation spline:

- Given the set of control points, the curve is said to interpolate the control point if it passes through each points.
- If the curve is fitted from the given control points such that it follows the path of control point without necessarily passing through the set of point, then it is said to approximate the set of control point.

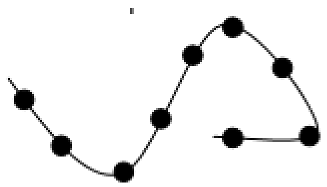


Fig: *A* set of nine control point **interpolated** with piecewise continuous polynomial sections.

Fig: *A* set of nine control points **approximated** with the piecewise continuous polynomial sections

Cubic Spline:

It is most often used to set up path for object motions o tot provide a representation for an existing object or drawing. To design surface of 3D object any spline curve can be represented by piece-wise cubic spline.

Cubic polynomial offers a reasonable compromise between flexibility and speed of computation. Cubic spline requires less calculations with comparison to higher order polynomials and require less memory. Compared to lower order polynomial cubic spline are more flexible for modeling arbitrary curve shape.

Given a set of control points, cubic interpolation spines are obtained by fitting the input points with a piecewise cubic polynomial curve that passes through every control points.

Suppose we have n+1 control points specified with co-ordinates

$p_k = (x_k, y_k, z_k)$, k = 0, 1, 2, 3, ……….. n

Fig: A piecewise continuous cubic-spline interpolation of n+1 control points

we can describe the parametric cubic polynomial that is to be fitted between each pair of control points with the following set of parametric equations:

$x(u) = a_xu^3 + b_xu^2 + c_xu + d_x$

$y(u) = a_yu^3 + b_yu^2 + c_yu + d_y$

$z(u) = a_zu^3 + b_zu^2 + c_zu + d_z$

Where (0 < = u < = 1)

There are three equivalent methods for specifying a particular spline representation.

1. **Set of boundary conditions:**
   For the parametric cubic polynomial for the x – coordinate along the path of spline section,
   $x(u) = a_xu^3 + b_x^2 + c_xu + d_x$, where (0<=u<=1)
   Boundary condition for this curve be the set on the end point coordinate x(0) and x(1) and in the first derivatives at end points x'(0) and x'(1). These four boundary condition are sufficient to determine the four coefficient a_x, b_x, c_x, d_x.
2. **Characterizing Matrix:** From the boundary condition we can obtain the characterizing matrix for spline. Then the parametric equation can be written as:

$$x(u) = [\ u3 \ \ u2 \ \ u \ \ 1] \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

3. **Blending function** that determines how specified geometric constraints on the curve are combined to calculate position along the curve path.

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

Where g_k are the geometric constraint parameters such as control points co-ordinate and slope of the curve at control point. $BF_k(u)$ are the polynomial blending functions.

Bezier curve and surface:

This is spline approximation method, developed by the French Engineer Pierre Bezier for use in the design of automobile body. Beziers spline has a number of properties that make them highly useful and convenient for curve and surface design. They are easy to implement. For this reason, Bezier spline is widely available in various CAD systems.

Beziers Curves:

In general, Bezier curve can be fitted to any number of control points. The number of control points to be approximated and their relative position determine the degree of Bezier polynomial. The Bezier curve can be specified with boundary condition, with characterizing matrix or blending functions. But for general Bezier curves, blending function specification is most convenient.

Suppose we have n+1 control points: $p_k(x_k, y_k, z_k)$, $0 \le k \le n$. These co-ordinate points can be blended to produce the following position vector P(u) which describes path of an approximating Bezier polynomial function p_0 and p_n.

$$P(u) = \sum_{k=0}^{n} p_k . BEZ_{k,n}(u), \qquad 0 \le u \le 1 \text{ ---------------------- } 1$$

The Bezier blending function $BEZ_{k,n}(u)$ are the Bernstein polynomial.

$$BEZ_{k,n}(u) = c(n,k)u^k(1-u)^{n-k}$$

Where, c(n,k) are the binomial coefficients:

$$C(n,k) = n! / k!(n-k)!$$

The vector equation (1) represents a set of three parametric equations for individual curve coordinates.

$$x(u) = \sum_{k=0}^{n} x_k . BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k . BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k . BEZ_{k,n}(u)$$

As a rule, a Bezier curve is a polynomial of degree one less than the number of control points used: Three points generate a parabola, four points a cubic curve, and so forth.

Fig below demonstrates the appearance of some Bezier curves for various selections of control points in the xy-plane (z = 0), with certain control-point placements:

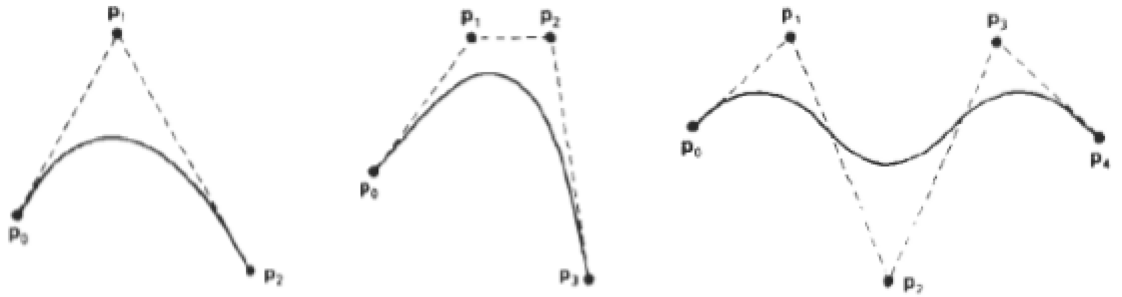


Fig: Examples of two-dimensional Bezier curves generated from three, four and five control points.

Properties of Bezier Curves:

1. It always passes through initial and final control points, i.e. $P(0) = p_0$ and $P(1) = p_n$.
2. Values of the parametric first derivatives of Bezier curve at the end points can be calculated from control points as:
   $P'(0) = -np_0 + np_1$
   $P'(1) = -np_{n-1} + np_n$
3. The slope at the beginning of the curve is along the line joining the first two points and slope at the end of curve is along the line joining last two points.
4. Parametric second derivative of a Bezier curve at end points are calculated as:
   $P''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$
   $P''(1) = n(n-1)[(p_{n-2} - p_{n-1}) – (p_{n-1} - p_n)]$
5. It lies with in the convex hull of the control points. This follows from the properties of Bezier blending functions: they are all positive and their sum is always 1

$$\sum_{k=0}^{n} BEZ_{k,n}(u) = 1$$

Bezier Surfaces:

Two sets of orthogonal Bezier curves can be used to design an object surface by specifying by an input mesh of control points. The parametric vector function for the Bezier surface is formed as the Cartesian product of Bezier blending functions:

$$\mathbf{P}(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} \mathbf{p}_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

With $p_{j,k}$ specifying the location of the (m + 1) by (n + 1) control points.

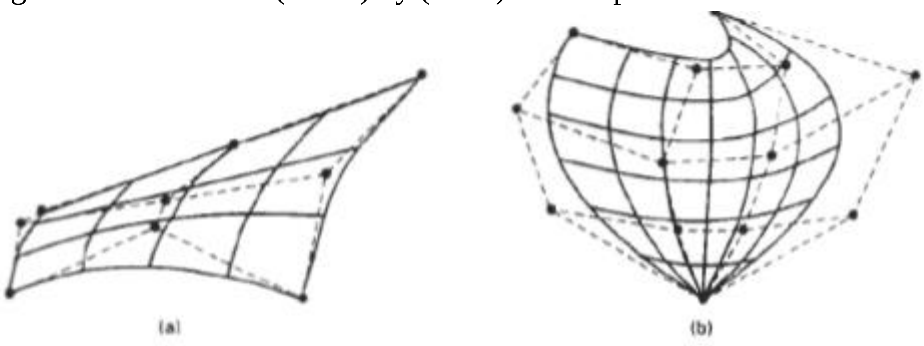


Fig: Bezier surfaces constructed tor
(a) m = 3, n = 3, and (b) m = 4, n = 4. Dashed lines connect the control points

Octree Representation (Solid – object representation)

This is the space-partitioning method for 3D solid object representation. This is hierarchical tree structures (octree) used to represent solid object in some graphical system. Medical imaging and other applications that require displays of object cross section commonly use this method. E.g. CT-scan

The octree encoding procedure for a three-dimensional space is an extension of an encoding scheme for two-dimensional space, called quadtree encoding. Quadtrees are generated by successively dividing a two-dimensional region (usually a square) into quadrants. Each node in the quadtree has four data elements, one for each of the quadrants in the region. If all pixels within a quadrant have the same color (a homogeneous quadrant) the corresponding data element in the node stores that color. In addition, a flag is set in the data element to indicate that the quadrant is homogeneous. Otherwise, the quadrant is said to be heterogeneous, and that quadrant is itself divided into quadrants.

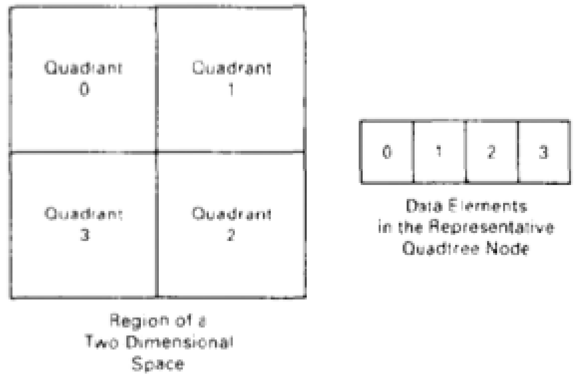


Fig: Region of a two – dimensional space divided into numbered quadrants and the associated quadtree node with four data elements.

It provides a convenient representation for storing information about object interiors. An octree encoding scheme divides region of 3D space into octants and stores 8 data elements in each node of the tree. Individual elements are called volume element or voxels. When all voxels in an octant are of same type, this type value is stored in corresponding data elements. Any heterogeneous octants are subdivided into octants again and the corresponding data element in the node points to the next node in the octree. Procedures for generating octrees are similar to those for quadtrees: Voxels in each octant are tested, and octant subdivisions continue until the region of space contains only homogeneous octants. Each node in the octree can now have from zero to eight immediate descendants.

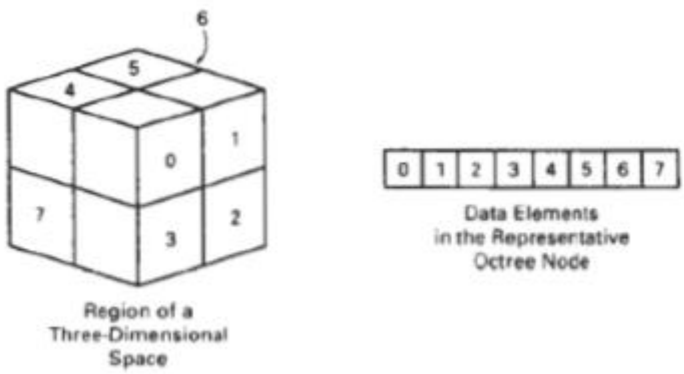


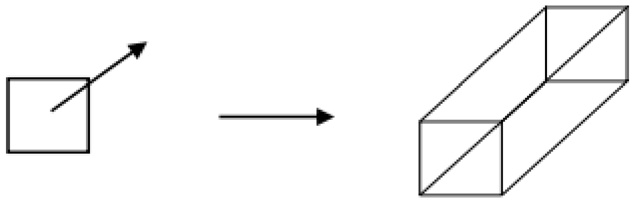
Fig: Region of a three – dimensional space divided into numbered octants and the associated octree node with eight data elements.

Solid modelling

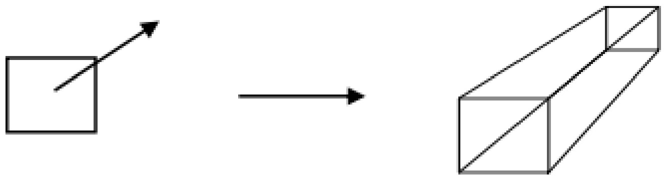
3.2.1 Sweep

Sweep Volume: Sweeping a 2D area along a trajectory creates a new 3D object.

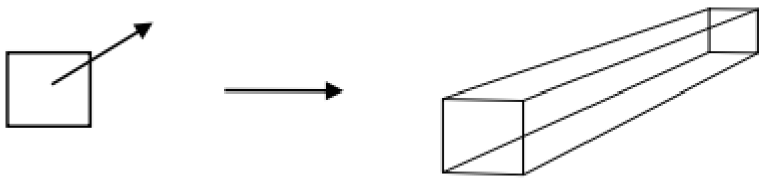
- **Translational sweep:** 2D area swept along a linear trajectory normal to the 2D Plane.



- **Tapered Sweep:** Scale area while sweeping.



- **Slanted Sweep:** Trajectory is not normal to the 2D plane.

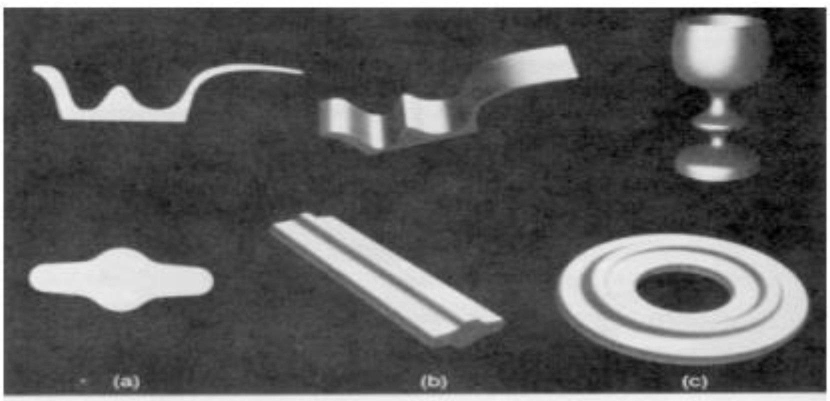


- **Rotational Sweep:** 2D area is rotated about an axis.

- **General sweep:** Object swept along any trajectory and transformed along the sweep.

Sweep Volume:

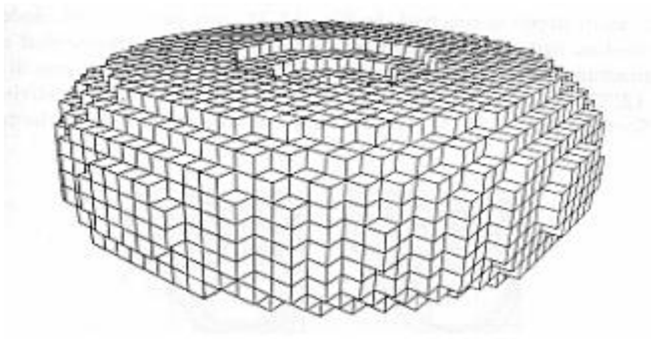
Translational and rotational sweep volumes.



Spatial Occupancy Enumeration:

- Space is described as a regular array of cells (usually cubes). Each cell is called a voxel.

- A 3D object is represented as a list of filled voxels.



Pros:

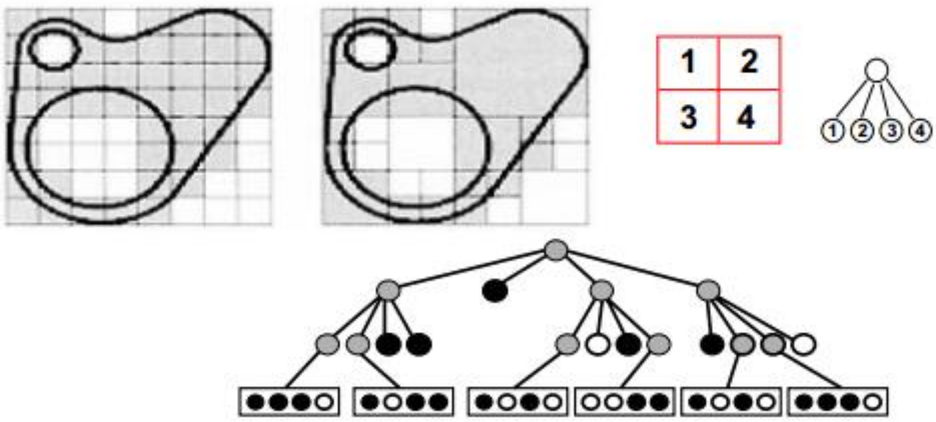
- Easy to verify if a point (a voxel) is inside or outside an object.
- Boolean operations are easy to apply.

Cons:

- Memory costs are high.
- Resolution is limited to size and shape of voxel.

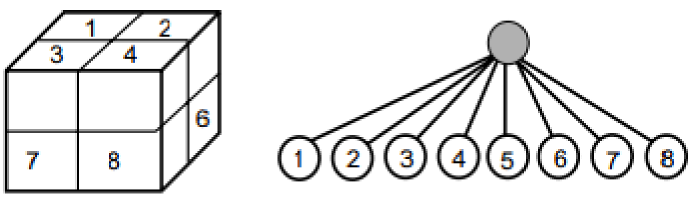
Quadtrees:

- A Quadtree is a data structure enabling efficient storage of 2D data.
- Completely full or empty regions are represented by one cell; recursive subdivision is used on the others.



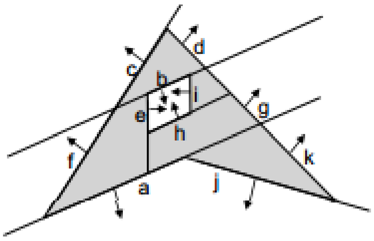
Octrees:

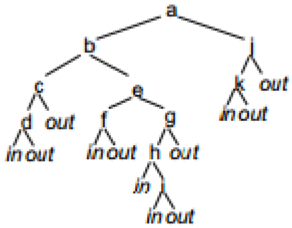
- An Octree is a 3D generalization of a Quadtree.
- Each node in an Octree has eight children rather than four.
- Describes a recursive partitioning of a volume into cells that are completely full or empty.



Binary Space Partition Tree - BSP

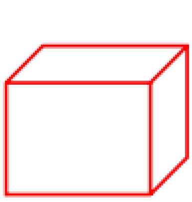
- Each internal node represents a plane in 3D space.
- Each node has 2 children pointers one for each side of the plane.
- A leaf node represents a homogeneous portion of space – either "in" or "out".
- Easy to determine if a point is inside or outside an object (recurse down the BSP tree).

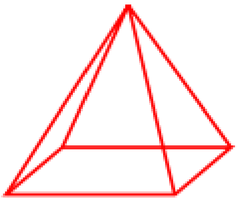




Boundary Representation

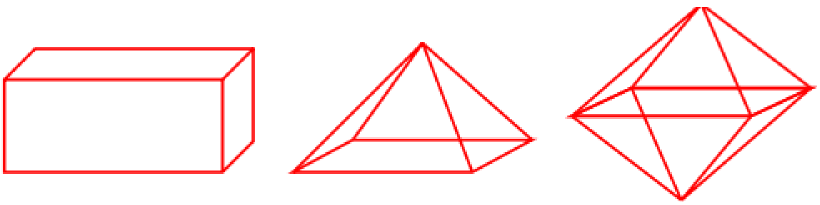
- A closed 2D surface defines a 3D object.
- At each point on the boundary there is an "in" and an "out" side.
- Boundary representations can be defined in two ways:
    - Primitive based: A collection of primitives forming the boundary (polygons, for example)
    - Free form based (splines, parametric surfaces, implicit forms)
- A polyhedron is a solid bounded by a set of polygons.
- A polyhedron is constructed from:
    - Vertices V
    - Edges E
    - Faces F
- Each edge must connect two vertices and be shared by exactly two faces.
- At least three edges must meet at each verted.





- A simple polyhedron is one that can be deformed into a sphere (contains no holes).
- A simple polyhedron must satisfies Euler's formula:

$$V - E + F = 2$$



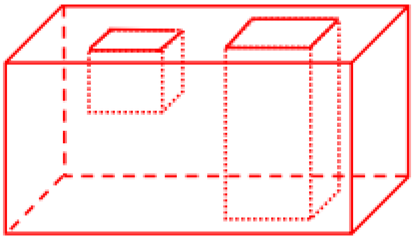
V = 8 E = 12 F = 6	V = 5 E = 8 F = 5	V = 6 E = 12 F = 8

- Euler's formula can be generalized to a polyhedron with holes and multiple components.

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

Where, H is the number of holes in the faces.
C is the number of separate components.

G is the number of pass-through holes (genus if C=1)
V, E and F are respectively vertices, edges and faces.



$$V - E + F - H = 2\,(C - G)$$
$$24 - 36 + 15 - 3 = 2\,(1 - 1)$$

Exercise

1. Explain with algorithm of generating curves.
2. Set up a procedure for establishing polygon tables for any input set of data points defining an object.
3. Define the term: Polygon Messes
4. Why polygon description is consider as standard graphics objects? Explain the importance of polygon table.
5. Model the Bezier curve. Explain the importance of Bezier curve in graphical modelling.
6. Define polygon. What are the different types of polygons? Explain with example.
7. Differentiate between periodic B-spline curves and non-periodic B-spline curves.
8. Explain in detail about polygon table. How can you apply in the case of virtual reality?
9. What is polygon mesh? Explain the application of polygon mesh with example.
10. Explain in detail about polygon table. How can you apply in the case of computer animation?
11. How curves be generated? Explain it with any suitable algorithm.