UNIT II THREE-DIMENSIONAL CONCEPTS

UNIT II THREE-DIMENSIONAL CONCEPTS 9

Three-Dimensional object representations − Three-Dimensional geometric andmodeling transformations − Three-Dimensional viewing − Hidden surface elimination− Color models − Animation.

THREE DIMENSIONAL DISPLAY METHODS

To obtain display of a three-dimensional scene that has been modeled in world coordinates. we must first set up a coordinate reference for the "camera". This coordinate reference defines the position and orientation for the plane of the carnera film which is the plane we want to us to display a view of the objects in the scene. Object descriptions are then transferred to the camera reference coordinates and projected onto the selected display plane. We can then display the objects in wireframe (outline) form, or we **can** apply lighting surface rendering techniques to shade the visible surfaces.

PARALLEL PROJECTION

In a parallel projection, parallel lines in the world-coordinate scene projected into parallel lines on the two-dimensional display plane.

Perspective Projection

Another method for generating a view of a three-dimensional scene is to project points to the display plane along converging paths. This causes objects farther from the viewing position to be displayed smaller than objects of the same size that are nearer to the viewing position.In a perspective projection, parallel lines in a scene that are not parallel to the display plane are projected into converging lines

DEPTH CUEING

A simple method for indicating depth with wireframe displays is to vary the intensity of objects according to their distance from the viewing position. The viewing position are displayed with the highest intensities, and lines farther away are displayed with decreasing intensities.

Visible Line and Surface Identification

We can also clarify depth relation ships in a wireframe display by identifying visible lines in some way. The simplest method is to highlight the visible lines or to display them in a different color. Another technique, commonly used for engineering drawings, is to display the nonvisible lines as dashed lines. Another approach is to simply remove the nonvisible lines

Surface Rendering

Added realism is attained in displays by setting the surface intensity of objects according to the lighting conditions in the scene and according to assigned surface characteristics. Lighting specifications include the intensity and positions of light sources and the general background illumination required for a scene. Surface properties of objects include degree of transparency and how rough or smooth the surfaces are to be. Procedures can then be applied to generate the correct illumination and shadow regions for the scene.

Exploded and Cutaway View

Exploded and cutaway views of such objects can then be used to show the internal structure and relationship of the object Parts

Three-Dimensional and Stereoscopic View

Three-dimensional views can be obtained by reflecting a raster image from a vibrating flexible mirror. The vibrations of the mirror are synchronized with the display of the scene on the CRT. As the mirror vibrates, the focal length varies so that each point in the scene is projected to a position corresponding to its depth.

Stereoscopic devices present two views of a scene: one for the left eye and the other for the right eye.

THREE DIMENSIONAL OBJECT REPRESENTATIONS

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

Boundary representations (B-reps) describe a three-dimensional object as a set of surfaces that separate the object interior from the environment.Typical examples of boundary representations are polygon facets and spline patches.

- **Space-partitioning representations** are used to describe interior properties, by partitioning the spatial region containing an object into a set of small, nonoverlapping, contiguous solids (usually **cubes).**

POLYGON SURFACES

The most commonly used boundary representation for a three-dimensional graphics object is a set of surface polygons that enclose the object interior. Many graphics systems store all object descriptions as sets of surface polygons. This simplifies and speeds up the surface rendering and display of objects, since all surfaces are described with linear equations. For this reason, polygon descriptions are often referred to as "standard graphics objects."

Polygon Tables

We specify a polygon surface with a set of vertex coordinates and associated attribute parameters. As information for each polygon is input, the data are placed into tables that are to be used in the subsequent' processing, display, and manipulation of the objects in a scene.

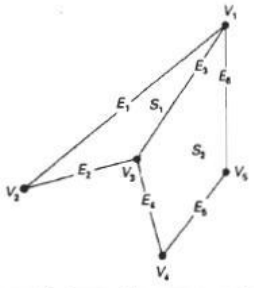
Polygon data tables can be organized into two groups:
geometric tables - attribute tables.

Geometric data tables contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces.

Attribute information for an object includes parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics.

A convenient organization for storing geometric data is to create three lists: **a vertex table, an edge table, and a polygon table.**

Coordinate values for each vertex in the object are stored in the vertex table. The edge table contains pointers back into the vertex table to identify the vertices for each polygon edge. And thepolygon table contains pointers back into the edge table to identify the edges for each polygon.



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

Plane Equations

To produce a display of a three-dimensional object, we must process the input data representation for the object through several procedures.

These processing steps include transformation of the modeling and world-coordinate descriptions to viewing coordinates, then to device coordinates; identification of visible surfaces; and the application of surface-rendering procedures.

The equation for 'I plane surface can be expressed In the
form Ax + By + Cz + D =0

where (r, y, z) is any point on the plane, and the coefficients A, B, C, and D are constants describing the spatial objects of the plane

we select three successive polygon vertices (x1, y1,z1), (x2, y2,z2), (x3, y3,z3),

and solve thc following set of simultaneous linear plane equation5 for the ratios AID, B/D,and ClD

$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1,$ where k =1,2,3…

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

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

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

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

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

As vertex values and other information are entered into the polygon data structure,values tor ***A, B,*** *C.* and D are computed for each polygon and stored with the other polygon data.

Plane equations are used also to identify the position of spatial points relativeto the plane surfaces of an object. For any point (x, y, **z)** not on a plane withparameters ***A, B,*** C, ***D,*** we have Ax + By+ Cz+ D != 0 We **can** identify the point as either inside or outside the plane surface accordingto the sign (negative or positive) of ***Ax*** + ***By*** + Cz + D:

if $Ax + By + Cz + D < 0$, the point (x, y, z) is inside the surface

if $Ax + By + Cz + D > 0$, the point (x, y, z) is outside the surface

CURVED LINES AND SURFACES

Curve and surface equations can be expressed in either a parametric or anonparametric form.

QUADRIC SURFACES

A frequently used class of objects is the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids, and hyperboloids. Quadric surfaces, particularly spheres and ellipsoids, are common elements of graphics scenes.

Sphere

In Cartesian coordinates, a spherical surface with radius r centered on the coordi-nate origin is defined as the set of points (x, y, **z)** that satisfy the equation

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

We can also describe the spherical surface in parametric form, using latitude and longitude angles (Fig. 10-8):

$$x = r\cos\phi\cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$
$$y = r\cos\phi\sin\theta, \qquad -\pi \le \theta \le \pi \qquad (10\text{-}8)$$
$$z = r\sin\phi$$

Torus

A torus is a doughnut-shaped object, as shown in Fig. It can be generated by rotating a circle or other conic about a specified axis. The Cartesian representation for points over the surface of a torus can be written in the form

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

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

$$x = r_x(r + \cos\phi)\cos\theta, \qquad -\pi \le \phi \le \pi$$

$$y = r_y(r + \cos\phi)\sin\theta, \qquad -\pi \le \theta \le \pi$$

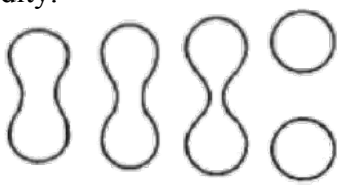
$$z = r_z \sin\phi$$

BLOBBY OBJECTS

Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other opts.

Examples in this class of objects include molecular structures, water droplets and other liquid effects, melting objects, and muscle shapes in the human body. **These** objects can bedescribed as exhibiting "blobbiness" and are often simply referred to as blobby objects, since their shapes show a certain degree of fluidity.

Eg:



Surface function is then defined as

$$f(x, y, z) = \sum_i b_i e^{-a_i r_i^2} - T = 0$$

where $r_i^2 = \sqrt{x_i^2 + y_i^2 + z_i^2}$, parameter T is some specified threshold.

And a& b – to adjust the amount of blobbiness of an object.

SPLINE REPRESENTATIONS

In drafting terminology, a spline is a flexible strip **used** to produce a smoothcurve through a designated **set** of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.

INTERPOLATION AND APPROXIMATION SPLINES

We specify a spline curver by giving a set of coordinate positions, called control points, which indicates the general shape of the curve. These control points are then fitted with piece wise continuous parametric poly nomial functions in one of two ways.

- When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said to interpolate the set of control points.
- On the other hand, when the polynomials are fitted to the general control-point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points

Spline Specifications

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

(1) We can state the set of boundary conditions that are imposed on the spline;

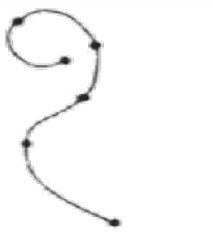


Figure 10-19
A set of six control points interpolated with piecewise continuous polynomial sections.

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

(2) we can state the matrix that characterizes the spline; or

(3) we can state the set of blending functions (or basis functions) that determine how specified geometric constraints on the curve are combined to calculate positions along the curve path **we** have the following parametric cubic polynomial representation for the **x** coordinate along the path of a spline section:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \qquad 0 \le u \le 1$$

Boundary conditions for this curve might be set, for example, on the endpoint coordinates x(0) and x(l) and on the parametric first derivatives at the endpoints x'(0) and x' (1) . These four boundary conditions are sufficient to determine the values of the four coefficients ak, bk, ck, and dk.

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

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

Where U – row matrix of powers of parameter u

C – coefficient column matrix

1. To obtain a polynomial representation for coordinate **x** in terms ***of*** the geometric constraint parameters

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

where ***gk*** are the constraint parameters, such as the control-point coordinates and slope of the curve at the control points, and ***BFk(u)*** are the polynomial blending functions.

BEZIER CURVES AND SURFACES

This spline approximation method was developed by the French engineer Pierre Mzier for use in the design of Renault automobile bodies. **Bezier** splines have a number of properties that make them highly useful and convenient for curve and surface design.

Bezier Curves

In general, a Bezier curve section 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 the BCzier polynomial. As with the interpolation splines, a bezier curve can be specified with boundary conditions, with a characterizing matrix, or with blending functions.

For general Bezier curves, the blending-function specification is the most convenient.

Suppose we are given ***n*** + 1 control-point positions: **pk** = (xk, ***yk,*** zk), with ***k*** varying from **0** to n. These coordinate points can **be** blended to produce the following position vector ***P(u),*** which describes the path of an approximating Bezier polynomial function between **p1,** and p_n.

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k BEZ_{k,n}(u), \qquad 0 \le u \le 1$$

- The Bezier blending functions $\boldsymbol{BEZ_{k,n}(u)}$ are the ***Bemstein polynomials:***

$$BEZ_{k,n}(u) = C(n, k)u^k(1 - u)^{n-k}$$ where the ***C(n, k)*** are the binomial coefficients: $$C(n, k) = \frac{n!}{k!(n - k)!}$$

we can define Bezier blending functions with the recursive calculation

$$BEZ_{k,n}(u) = (1 - u)\,BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u), \qquad n > k \ge 1$$

with $BEZk,k = u^k$, and $BEZ_{0,k} = (1 - u)^k$

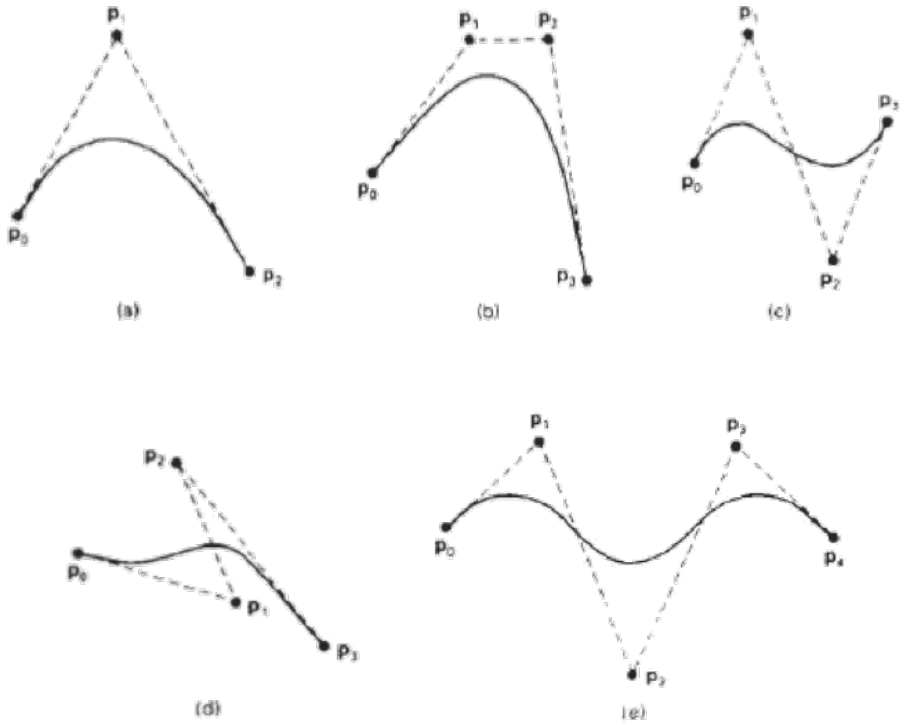
vector equation represents a set **of** three parametric equations for the individual curve coordinates

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

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

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

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 cubiccurve, and so forth. For Example



Properties of Bezier Curves

1. A very useful property of a Bezier curve is that it always passes through the first and last control points. That is, the boundary conditions at the two ends of the curve are

$$\mathbf{P}(0) = \mathbf{p}_0$$

$$\mathbf{P}(1) = \mathbf{p}_n$$

2. Values of the parametric first derivatives of a Bezier curve at the endpoints can be calculated from control point coordinates as

$$\mathbf{P}'(0) = -n\mathbf{p}_0 + n\mathbf{p}_1$$

$$\mathbf{P}'(1) = -n\mathbf{p}_{n-1} + n\mathbf{p}_n$$

Thus, the slope at the beginning of the curve is along the line joining the first two control points, and the slope at the end of the curve is along the line joining the last two endpoints.

3. Similarly, the parametric second derivatives o f a Bezier curve at the endpoints are calculated as

$$\mathbf{P}''(0) = n(n-1)[(\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)]$$

$$\mathbf{P}''(1) = n(n-1)[(\mathbf{p}_{n-2} - \mathbf{p}_{n-1}) - (\mathbf{p}_{n-1} - \mathbf{p}_n)]$$

Another important property of any Bezier curve is that it lies within the convex hull (convex polygon boundary) of the control points. This follows from the properties of Bezier blending functions:

They are all positive and is always one.

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

where the Bezier matrix is

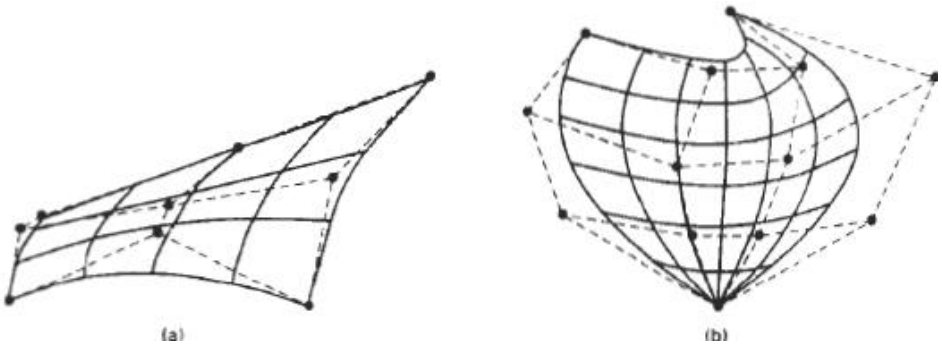
$$\mathbf{M}_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bezier Surtaces

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:

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

with P**j,k** specifying the location of the ***(m*** + **1)** by ***(n*** + **I)** control points.



The control points are connected by dashed lines, and the solid lines show curves of constant ***u*** and constant v. Each curve of constant ***u*** is plotted by varying v over the interval from 0to **1,** with ***u*** fixed at one of the values in this unit interval. Curves of constant v are plotted similarly

B-SPLINE CURVES AND SURFACES

B-splines have two advantages over Bezier splines:

(1) the degree of a B-spline polynomial can be set independently of the number of control points (with certain limitations),

(2) B-splines allow local control over the shape of a spline curve or surface

B-Spline Curves

We can write a general expression for the calculation of coordinate positions along a B-spline curve in a blending-function formulation as

$$P(u) = \sum_{k=0}^{n} p_k B_{k,d}(u), \qquad u_{min} \le u \le u_{max}, \qquad 2 \le d \le n + 1$$

where the **pk** are an input set of **n** + 1 control points.

Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas:

$$B_{k,1}(u) = \begin{cases} 1, & \text{if } u_k \le u < u_{k+1} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u)$$

where each blendjng function is defined over d subintervals of the total range of ***u.***

The selected set of subinterval endpoints u, is referred to as a **knot** vector. B-spline curves have the following **properties**.

- The polynomial curve has degree d - 1 and C^{d-2} continuity over the rangeof ***u.***
  For n + 1 control points, the curve is described with n + 1 blending functions.
- Each blending function $B_{k,d}$ is defined over d subintervals of the total range of u, starting at knot value ***u1.***
- The range of parameter u 1s divided into ***n*** + d subintervals by the **n** + ***d*** +1 values specified in the knot vector.
- With knot values labeled as ***[u1,*** u2, . . . , ***un,],*** the resulting B-spline curve is defined only in the interval from knot value $\mathbf{u_{d-1}}$, up to knot value $\mathbf{u_{n+1}}$.
- Each section of the spline curve (between two successive knot values) is influenced by d control points.
- Any one control point can affect the shape of at most d curve sections.

B-splines are tightly bound to the input positions. For any value of ***u*** in the interval from knot value $\mathbf{u_{d-1}}$ to $\mathbf{u_{n+1}}$ the sum over all basis functions is 1:

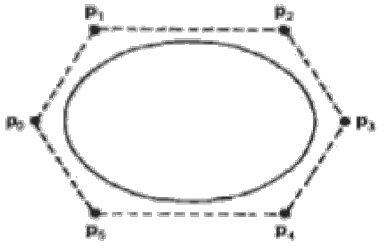
$$\sum_{k=0}^{n} B_{k,d}(u) = 1$$

➢ **Uniform, Periodic B-Splines**

When the spacing between knot values is constant, the resulting curve is called a uniform B-spline.

Cubic, Periodic B-Splines

Since cubic, periodic 8-splines are commonly used in graphics packages, we consider the fornlulation for this class of splines. Periodic splines are particularly useful for generating certain closed curves



A closed, periodic, piecewise, cubic B-spline constructed with cyclic specification of the six control points.

Open Uniform B-Splines

This class of B-splines is a cross between uniform B-splines and nonuniform Bsplines. Sometimes it is treated as a special type of uniform 8-spline, and sometimes it is considered to be in the nonuniform B-splines classification.

For the open uniform B-splines, or simply open B-splines, the knot spacing is uniform except at the ends where knot values are repeated d times.

For any values of parameters ***d*** and n, we can generate an open uniform knot vector with integer values using the calculations

$$u_j = \begin{cases} 0, & \text{for } 0 \le j < d \\ j - d + 1, & \text{for } d \le j \le n \\ n - d + 2, & \text{for } j > n \end{cases}$$

for values of] ranging from 0 to ***n*** + ***d.*** With this assignment, the first d knots are assigned the value 0, and the last ***d*** knots have the value ***n*** - d + *2.*

Non Uniform B-Splines

For this class of splines, we can specify any values and intervals for the knot vector. With nonuniform B-splines, we can choose multiple internal knot values and unequal spacing between the knot values.

B-Spline Surfaces

We can obtain a vector point function over a B-spline surface using the Cartesian product ofB-spline blending functions in the form

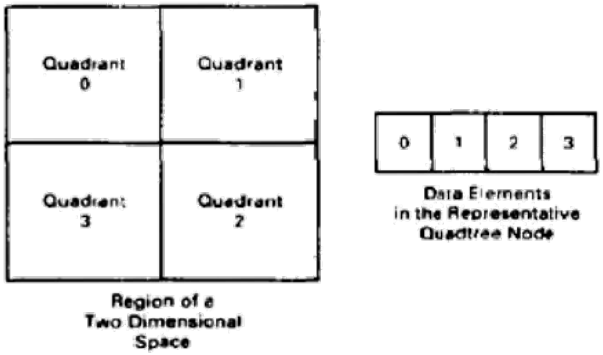
$$\mathbf{P}(u, v) = \sum_{k_1=0}^{n1} \sum_{k_2=0}^{n2} \mathbf{p}_{k_1,k_2} B_{k_1,d_1}(u) B_{k_2,d_2}(v)$$

where the vector values for $\mathbf{p}_{k_1,k_2}$ specify positions of the $(n_1 + 1)$ by $(n_2 + 1)$ control points.

OCTREES

Hierarchical tree structures, called octrees, are used to represent solid objects in some graphics systems.The tree structure is organized so that each node corresponds to a region of three-dimensional space. This representation for solids takes advantage of spatial coherence to reduce storage requirements for three-dimensional objects. It also provides a convenient representation for storing information about object interiors.

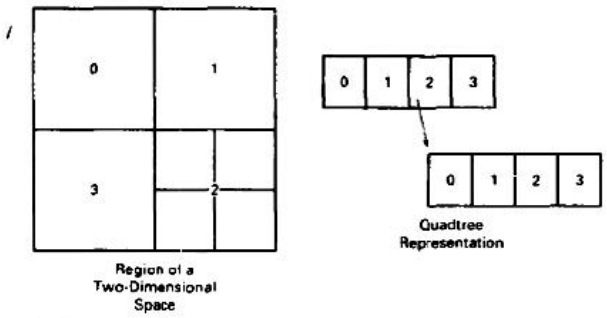
The octree encoding procedure for a three-dimensional space is an extensionof an encoding scheme for two-dimensional space, called quadtree encoding.Quadtrees are generated by successively dividing a two-dimensional region (usually a square) into quadrants. Each node in the quadtree has four data elements,one 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. Suppose all pixels in quadrant are found to be red. The color code for red is then placed in data element 2 of the node. Otherwise, the quadrant is said to be heterogeneous, and that quadrant is itself divided into quadrants

An octree encoding scheme divides regions of three-dimensional space (usually cubes) into octants and stores eight data elements in each node of the tree



- Individual elements of a three-dimensional space are called **volume elements,** or **voxels.**
- When all voxels in an octant are of the same type this type value is stored in the corresponding data element of the node.
- Empty regions of space are represented by voxel type "void."
- Any heterogeneous octant is subdivided into octants, and the corresponding data element in the node points to the next node in the octree.

BSP TREES

This representation scheme is similar to **octree** encoding, except **we** now divide space into two partitions instead of eight at each step.

With a binary space-partitioning **(BSP)** tree, we subdivide a scene into two sections at each step with aplane that can be at any position and orientation.

In an octree encoding, the scene is subdivided at each step with three mutually perpendicular planes aligned with the Cartesian coordinate planes.

2.2 Three-Dimensional Geometric and Modeling Transformations

Methods for geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the *z* coordinate.

We now translate an object by specifying a three-dimensional translation vector, which determines how much the object is tobe moved in each of the three coordinate directions. Similarly, we scale an object with three coordinate scaling factors.

TRANSLATION

In a three-dimensional homogeneous coordinate representation, a point is translated (Fig. **17-1)** from position P = ***(x,*** **y,** ***z)*** to position P' = ***(x',*** **y',** z') with the matrix Operation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = T \cdot P$$

Parameters t_x, t_y, and t_z specifying translation distances for the coordinate directions x, y, and z, are assigned any real values. The matrix representation in Eq. 11-1 is equivalent to the three equations

$$x' = x + t_x, \qquad y' = y + t_y, \qquad z' = z + t_z \tag{11-3}$$

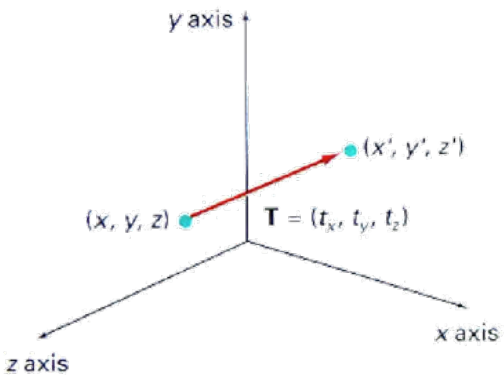


Figure 11-1
Translating a point with translation vector $\mathbf{T} = (t_x, t_y, t_z)$.

An object is translated in three dimensions by transforming each of the defining points of the object.

ROTATION

To generate a rotation transformation for an object, we must designate an axis of rotation (about which the object is to be rotated) and the amount of angular rotation.

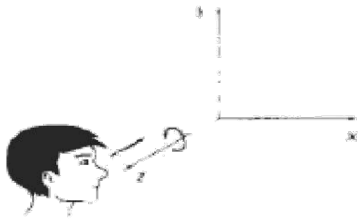
Unlike two-dimensional applications, where all transformations are carried out in the **xy** plane, a three-dimensional rotation can be specified around any line in space.

The easiest rotation axes to handle are those that are parallel to the coordinate axes. Also, we can use combinations of coordinate axis rotations (along with appropriate translations) to specify any general rotation.

By convention, positive rotation angles produce counterclockwise rotations about a coordinate axis, if we are looking along the positive half of the axis toward the coordinate origin.

Coordinate-Axes Rotations

The two-dimensional z-axis rotation equations are easily extended to three dimensions:



$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned} \tag{11-4}$$

Parameter θ specifies the rotation angle. In homogeneous coordinate form, the three-dimensional z-axis rotation equations are expressed as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{11-5}$$

which we can write more compactly as

$$\mathbf{P}' = \mathbf{R}_z(\theta) \cdot \mathbf{P}$$

or

$$\mathbf{P}' = \mathbf{R}_x(\theta) \cdot \mathbf{P} \qquad (11\text{-}10)$$

Rotation of an object around the x axis is demonstrated in Fig. 11.6.

Cyclically permuting coordinates in Eqs. 11-8 give us the transformation equations for a **y-axis rotation**:

$$\begin{aligned} z' &= z\cos\theta - x\sin\theta \\ x' &= z\sin\theta + x\cos\theta \\ y' &= y \end{aligned} \qquad (11\text{-}11)$$

The matrix representation for y-axis rotation is

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (11\text{-}12)$$

or

$$\mathbf{P}' = \mathbf{R}_y(\theta) \cdot \mathbf{P} \qquad (11\text{-}13)$$

General Three-Dimensional Rotations

A rotation matrix for any axis that does not coincide with a coordinate axis can be set up as a composite transformation involving combinations of translations and the coordinate-axes rotations. .

Any coordinate position **P** on the object in this figure is transformed with the sequence shown as

$$\mathbf{P}' = \mathbf{T}^{-1} \cdot \mathbf{R}_x(\theta) \cdot \mathbf{T} \cdot \mathbf{P}$$

where the composite matrix for the transformation is

$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x(\theta) \cdot \mathbf{T}$$

Given the specifications f or the rotation axis and the rotation angle, we can accomplish the required rotation in five s t e p

1 Translate the object so that the rotation axis pass= through the coordinate origin.

2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes.

3. Perform the specified rotation about that coordinate axis.

4. Apply inverse rotations to bring the rotation axis back to original orientation.

5. Apply the inverse translation to bring the rotation axis back to its original position

A rotation axis can be defined with two coordinate positions, as in Fig. 11-10, or with one coordinate point and direction angles (or direction cosines) between the rotation axis and two of the coordinate axes. We will assume that the rotation axis is defined by two points, as illustrated, and that the direction of rotation is to be counterclockwise when looking along the axis from $\mathbf{P}_2$ to $\mathbf{P}_1$. An axis vector is then defined by the two points as

with one coordinate point and axes. We will assume that the axes is to be counter clockwise

$$\begin{aligned} \mathbf{V} &= \mathbf{P}_2 - \mathbf{P}_1 \\ &= (x_2 - x_1, y_2 - y_1, z_2 - z_1) \end{aligned} \qquad (11\text{-}14)$$

A unit vector **u** is then defined along the rotation axis as

$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c) \qquad (11\text{-}15)$$

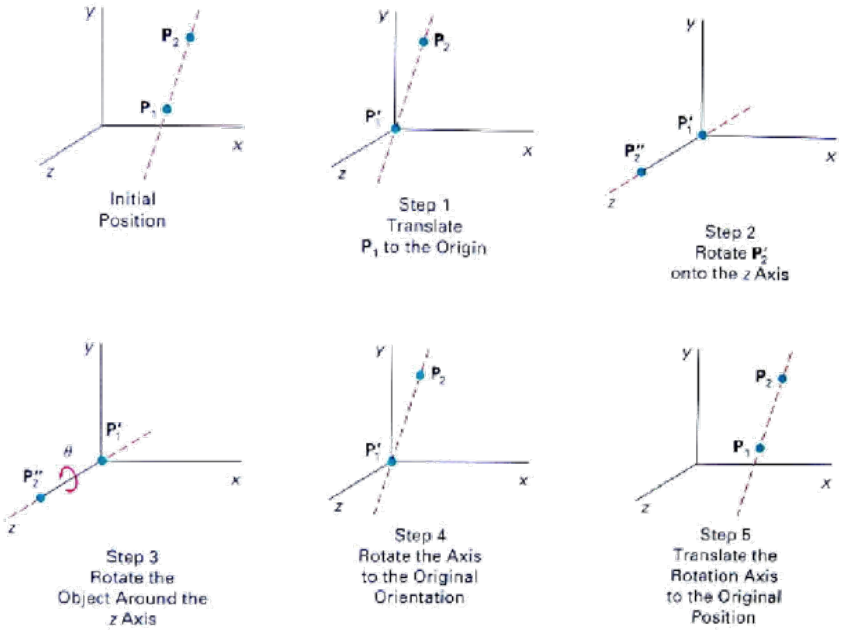


Figure 11-9
Five transformation steps for obtaining a composite matrix for rotation about an arbitrary axis, with the rotation axis projected onto the z axis.

SCALING

The matrix expression tor the scaling transformation of a position P = **(x,** y, *z)* relative to the coordinate origin can be written as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (11\text{-}42)$$

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P} \qquad (11\text{-}43)$$

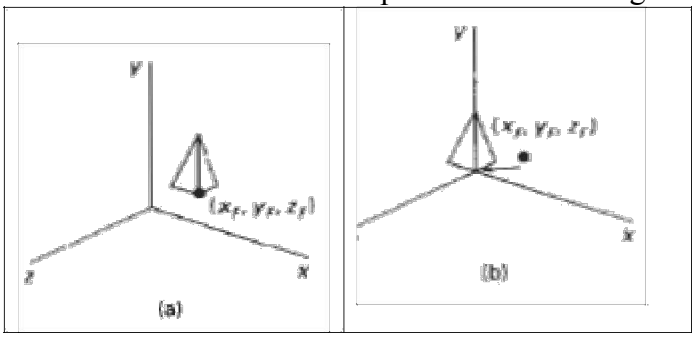
where scaling parameters sx, **sy,** and **sz,** are assigned any positive values. Explicit expressions for the coordinate transformations for scaling relative to the origin are

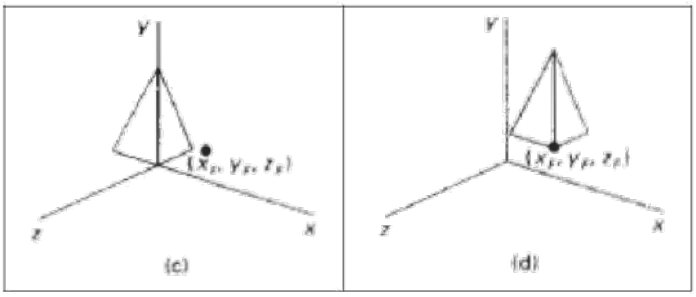
$$x' = x \cdot s_x \qquad y' = y \cdot s_y \qquad z' = z \cdot s_z$$

Scaling an object with transformation changes the size of the object and repositions the object relative to the coordinate origin. Also, if the transformation parameters are not all equal, relative dimensions in the object are changed:

Scaling with respect to a selected fixed position **(x,** y, z*,)* can be represented with the following transformation sequence:

1. Translate the fixed point to the origin.2. Scale the object relative to the coordinate origin using Eq. **11-42.**3. Translate the fixed point back to its original position.





This sequence of transformations is demonstrated in Fig. 11-18. The matrix representation for an arbitrary fixed-point scaling can then be expressed as the concatenation of these translate-scale-translate transformations as

$$\mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S}(s_x, s_y, s_z) \cdot \mathbf{T}(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11\text{-}45)$$

Reflections

A three-dimensional reflection can be performed relative to a selected ***reflection axis*** or with respect to a selected ***reception plane.*** In general, three-dimensional reflection matrices are set up similarly to those for two dimensions.

Reflections relative to a given axis are equivalent to 180 rotations about that axis. Reflections with respect to a plane are equivalent to 160' rotations in four-dimensional space.

When the reflection plane is a coordinate plane (either **xy, *xz, or*** yz), we can think of the transformation as a conversion between Left-handed and right-handed systems.

An example of a reflection that converts coordinate specifications from aright-handed system to a left-handed system (or vice versa)

This transformation changes the sign of the z coordinates, leaving the ***x*** and y-coordinate values unchanged. The matrix representation for this reflection of points relative to the **xy** plane is

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3 THREE-DIMENSIONAL VIEWING

For three-dimensional applications, First of all, we can view an object from any spatial position: from the front, from above, or from the back. ***Or*** we could generate a view of what we would **see** if we were standing in the middle of a group of objects or inside a single object, such as a building. Additionally, three-dimensional descriptions of objects must be projected onto the flat viewing surface of the output device.

VIEWING COORDINATES

Generating a view of an object in three dimensions is similar to photographing the object. We can walk around and take its picture from any angle, at various distances, and with varying camera orientations. Whatever appears in the viewfinder is projected onto the flat film surface. The type and size of the camera lens determines which parts of the scene appear in the final picture.

These ideas are incorporated into three dimensional graphics packages so that views of a scene can be generated, given the spatial position, orientation, and aperture size of the "camera".

Specifying the View Plane

1. We choose a particular view for a scene by first establishing the viewing-coordinate **system,** also called the view reference coordinate **system**. **A** view plane, or projection plane, is then set up perpendicular to the viewing ***z,*** axis.World-coordinate positions in the scene are transformed to viewing coordinates, then viewing coordinates are projected onto the view plane.To establish the viewing-coordinate reference frame, we first pick a world coordinate position called the view reference point.

PROJECTIONS

Once world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the three-dimensional objects onto the triodimensional view plane.

There are two basic projection methods .

In a parallel projection, coordinate positions are transformed to the view plane along parallel lines For a perspective projection object positions are transformed to the view plane a!ong lines that converge to a point called the projection reference point (or center of projection).

The projected view of the object is determined by the interjection ofthe projection lines with the view z. plane.

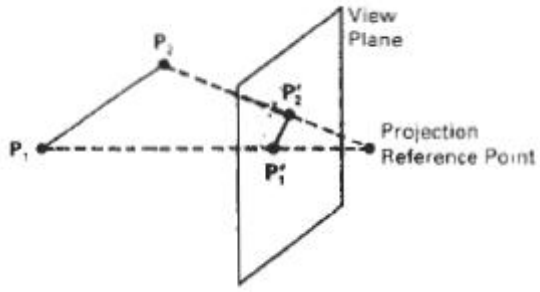


Figure 12-15
Perspective projection of an object to the view plane.

Parallel Projections

We can specify a parallel projection with a projection vector that defines the directionfor the projection lines. When the projection is perpendicular to the view plane, we have an orthographic parallel projection. Otherwise, we have a oblique parallel projection.

Orthographic projections are most often used to produce the front, side, and top views of an object

Perspective projection

- The center of projection is located at a finite point in three space.
- A distant line is displayed smaller than a nearer line of the same length.

- In three-dimensional homogeneous-coordinate representation

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & z_{vp}/d_p & -z_{vp}(z_{prp}/d_p) \\ 0 & 0 & 1/d_p & -z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

When a three-dimensional obpct is projected onto a view plane using per- hree-Dimensional Viewing spective transformation equations, any set of parallel lines in the object that are not parallel to the plane are projected into converging lines.

Parallel Lines that are parallel to the view plane will be projected as parallel lines.

vanishing point:

- The point at which a set of projected parallel lines appears to converge is called a vanishing point.
- Each such set of projected parallel lines will have a separate vanishing point;
- And in general, a scene can have any number of vanishing points, depending on howmany sets of parallel lines there are in the scene.
- The vanishing point for any set of lines that are parallel to one of the principal axes of an object is referred to as a principal vanishing point.
- We control the number of principal vanishing points (one, two, or three) with the orientation of the projection plane, and perspective projections are accordingly classified as one-point, two-point, or three-point projections.

THREE-DIMENSIONAL VIEWING FUNCTIONS

Several procedures are usually provided in a three-dimensional graphics library to enable an application program to set the parameters for viewing transfonnations.

With parameters spenfied in world coordinates, elements of the matrix for transforming worldcoordinate descriptions to the viewing reference frame are calculated using the function ***evaluateViewOrientationMatrix3 (x0, y0,' Z0, xN, yN, zN, xv, yv, zV, error.,viewMatrix)***

1. This function creates the viewMatrix from input coordinates defining the viewingsystem, 2. Parameters xo, yo, and z0 specify the origin (view reference point) of the viewing system. 3. World-coordinate vector defines the normal to the view plane and the direction of the positive, Three-Dimensional Viewing **viewi**ng axis. 4. And world-coordinate vector (xV, yv, zv) gives the elements of the view-up vector. The projection of this vector perpendicular to (xN, yN, zN) estab lishes the direction for the positive y, axis of the viewing system. 6. An integer error code is generated in parameter error if input values are not specified correctlyFor example, an error will be generated if we set (XV, YV, ZV) parallel to (xN, YN, zN)

To specify a second viewing-coordinate system, we can redefine some or allof the coordinate parameters and invoke evaluatevieworientationMatrix3 with a new matrix designation. In this way, we can set up any number of world-to-viewingcoordinate matrix transformations.

2.4 HIDDEN SURFACE ELIMINATION

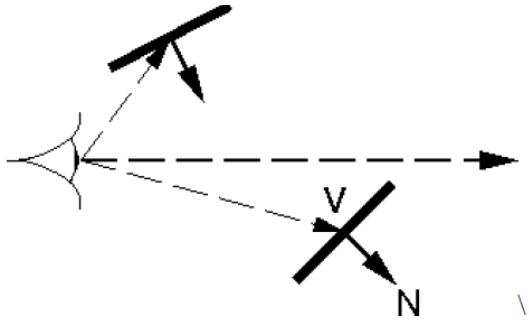
Hidden Surface Removal

When drawing lots of polygons, we want to draw only those ``visible'' to viewer. There are a variety of a orithms with different strong points.

- Issues:
  - Online
  - Device independent
  - Fast
  - Memory requirements
  - Easy to implement in hardware

Backface Culling

A simple way to perform hidden surface is to remove all ``backfacing'' polygons. The observation is that if polygon normal is facing away from the viewer then it is ``backfacing.'' For solid objects, this means



the polygon will not be seen by the viewer.

\

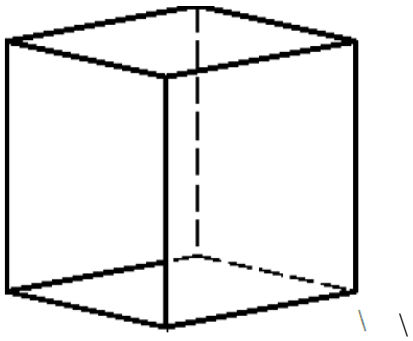
- Thus, if , then cull polygon.
- Note that V is vector from eye to point on polygon

You cannot use the view direction for this.

Backface Culling

Not a complete solution

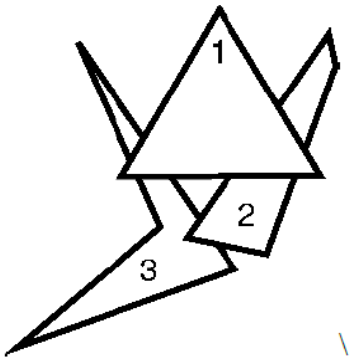
- If objects not convex, need to do more work.
- If polygons two sided (i.e., they do not enclose a volume) then we can't use it.



- A HUGE speed advantage if we can use it since the test is cheap and we expect at least half the polygons will be discarded.
- Usually performed in conjunction with a more complete hidden surface algorithm.
- Easy to integrate into hardware (and usually improves performance by a factor of 2).

Painter's Algorithm

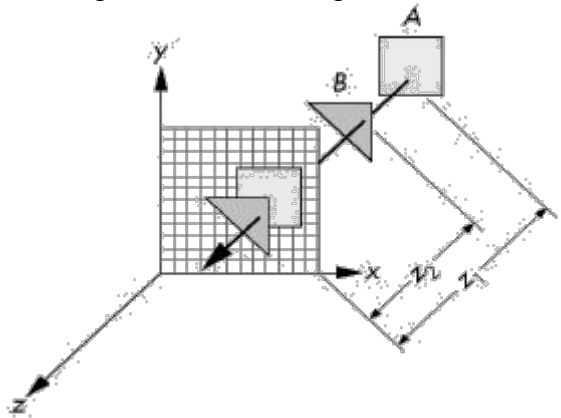
- Idea: Draw polygons as an oil painter might: The farthest one first.
  - Sort polygons on farthest *z*
  - Resolve ambiguities where *z*'s overlap
  - Scan convert from largest *z* to smallest *z*



- Since closest drawn last, it will be on top (and therefore it will be seen).
- Need all polygons at once in order to sort.

z-Buffer Algorithm

•The z or depth buffer–stores the depth of the closest object at each pixel
found so far As we render each polygon, compare the depth ofeach pixel to depth in z buffer
–If less, place the shade of pixel in the color buffer and update z buffer



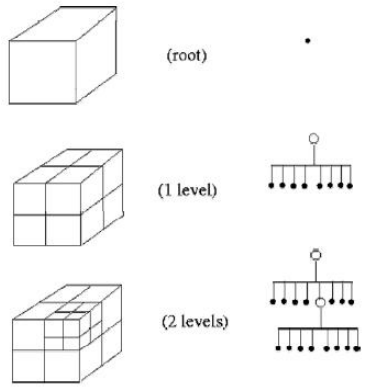
Function setpixel(int i, int j, rgb c, real z)
If z > z-buffer(i, j) **then**
z-buffer(i, j) = z
screen(i,j) = c

Space Partitioning

•Avoid rendering an object when it's unnecessary

–In many real-time applications, we want to eliminateas many objects as possible within the application.

Octree

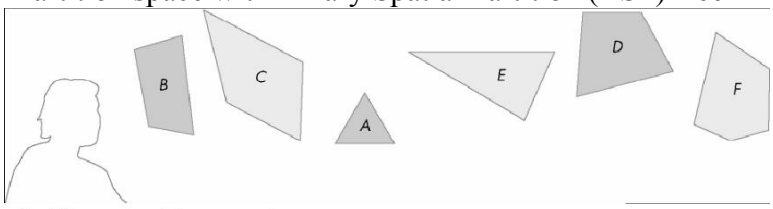


Why do we use BSP trees?

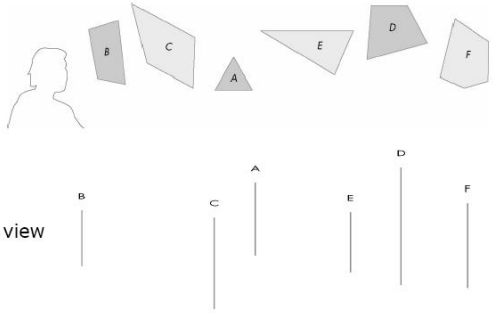
•Hidden surface removal

–A back-to-front painter's algorithm

•Partition space with Binary Spatial Partition (BSP)Tree



A Simple Example



The plane of A separates B and C from D, E and F

Binary Space Partitioning Tree

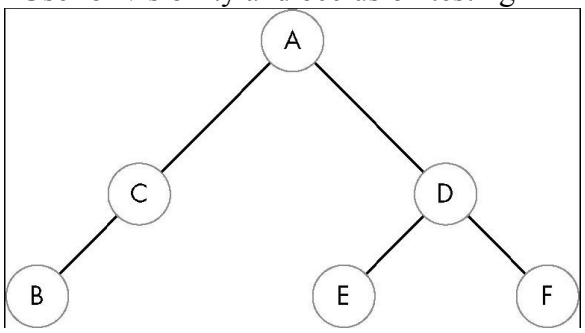
•Can continue recursively

–Plane of C separates B from A

–Plane of D separates E and F

•Can put this information in a BSP tree

–Use for visibility and occlusion testing

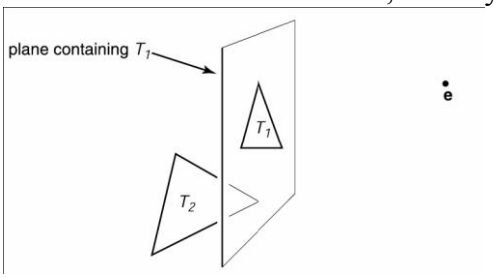


Key Idea of BSP

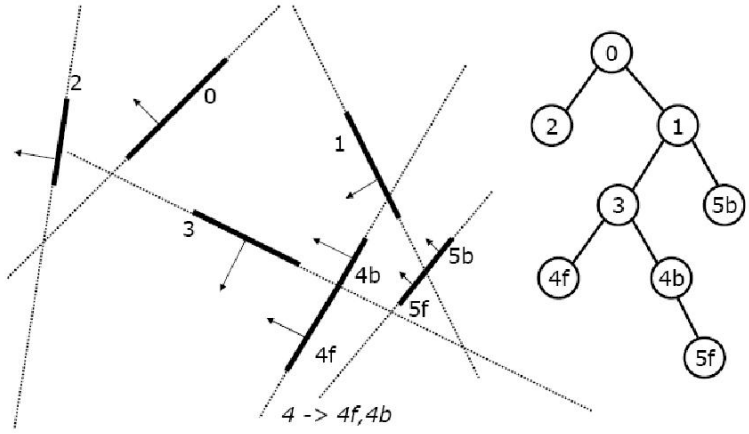
•Assume T2 not cross the plane of T1

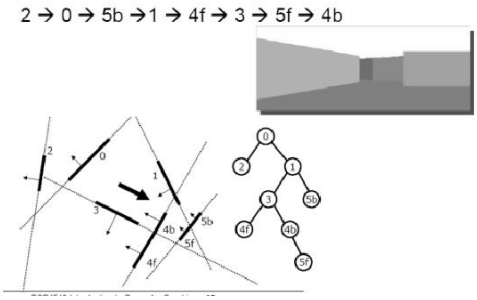
•If e and T2 on the same side, T1 won't block

•If e and T2 on different sides, T2 may block



Creating a BSP tree





2.5 Color models

1. RGB COLOR MODEL

- Based on the tri stimulus theory of vision, our eyes perceive color through the stimulationof three visual pigments in the cones of the retina. These visual pigments have a peak sensitivity at wavelengths of about 630 run (red), 530 nm (green), and 450 nm (blue).
- By comparing intensities in a light source, we perceive the color of the light. This theory of vision is the basis for displaying color output on a video monitor using the three color primaries, red, green, and blue, referred to as the RGB color model.
- We can represent this model with the unit cube defined on R,G, and B axes,

  as shown in Fig. 15-11. The origin represents black, and the vertex with coordinates(1; 1,l) is white. Vertices of the cube on the axes represent the primary colors,and the remaining vertices represent the complementary color for each of the primary colors.
- As with the XYZ color system, the RGB color scheme is an additive model.

  Intensities of the primary colors are added to produce other colors. Each color point within the bounds of the cube can be represented as the triple (R, G, B), where values for R, G, and B are assigned in the range from 0 to 1. Thus, a color C, is expressed in RGB components as

$$C_\lambda = R\mathbf{R} + G\mathbf{G} + B\mathbf{B} \qquad (15\text{-}5)$$

- The magenta vertex is obtained by adding red and blue to produce the triple (1,0,l). and white at (1,l. 1) is thesum of the red, green, and blue vertices. Shadesof gray are represented along the main diagonal of the cube from the origin(black) to the white vertex.
- Each point along this diagonal has an equal contributionfrom each primary color, so that a gray shade halfway between black and white is represented as (0.5, 0.5, 0.5)

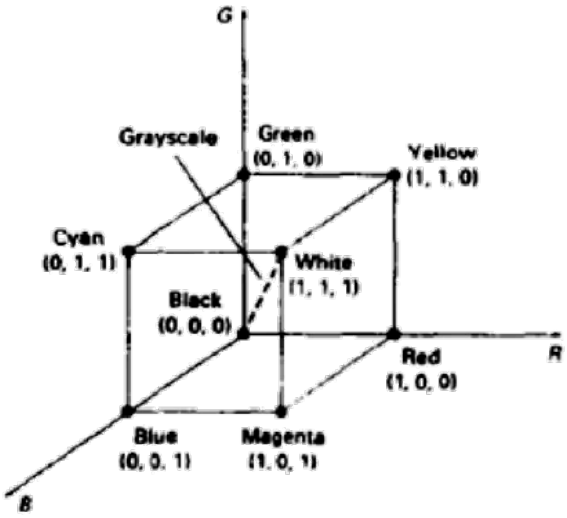


Figure 15-11
The RGB color model, defining colors with an additive process within the unit cube.

2. YIQ COLOR MODEL

. The National Television System Committee (NTSC) color model for forming the composite video signal is the YIQ model, which is based on concepts in the CIE XYZ model.

In the YI Q color model, parameter Y is the same as in the XYZ model. Luminance (brightness) information is contained in the Y parameter, while chromaticity information (hue and purity) is incorporated into the 1 and Q parameters.

A combination of red, green, and blue intensities is chosen for the Y parameter to yield the standard luminosity curve. Since Y contains the luminance information, black-and-white television monitors use only the Y signal.

The largest bandwidth in the NTSC video signal (about 4 MHz) is assigned to the Y information. Parameter I contains orange-cyan hue information that provides the flesh-tone shading, and occupies a bandwidth of approximately 1.5 MHz. Parameter Q carries green-magenta hue information in a bandwidth of about 0.6 MHz.

An RGB signal can be converted to a television signal using an NTSC encoder, which converts RGB values to YIQ values, then modulates and superimposes the I and Q information on the Y signal. The conversion from RGB values to YIQ values is accomplished with the transformation

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (15\text{-}6)$$

This transformation is based on the NTSC standard RGB phosphor, whose chromaticity coordinates were given in the preceding section. The larger proportions of red and green assigned to parameter Y indicate the relative importance of these hues in determining brightness, compared to blue.

An NTSC video signal can be converted to an RGB signaul sing an NTSC decoder, which separates the video signal into the YlQ components,'then converts to RGB values. We convert from YIQ space to RGB space with the inverse matrix transformation from Eq. 15-6:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.108 & 1.705 \end{bmatrix} \cdot \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \qquad (15\text{-}7)$$

3. CMY COLOR MODEL

1 A color model defined with the primary colors cyan, magenta, and yellow (CMY) is useful for describing color output to hard-copy devices.

2. Unlike video monitors, which produce a color pattern by combining light from the screen phosphors, hard-copy devices such as plotters produce a color picture by coating a paper with color pigments. We see the colors by reflected light, a subtractive process.

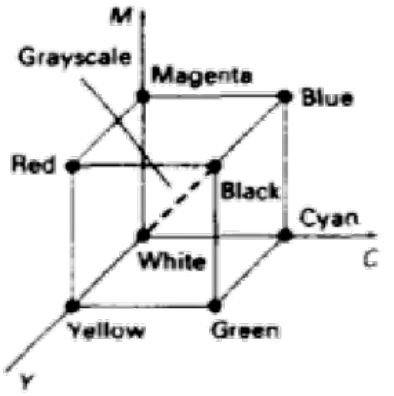


Figure 15-14
The CMY color model, defining colors with a subtractive process inside a unit cube.

In the CMY model, point (1, 1, 1) represents black, because all components of the incident light are subtracted. The origin represents white light.

Equal amounts of each of the primary colors produce grays, along the main diagonal of the cube. A combination of cyan and magenta ink produces blue light, becausethe red and green components of the incident light are absorbed. Other color combinations are obtained by a similar subtractive process

We can express the conversion from an RGB representation to a CMY representation with the matrix transformation

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where the white is represented in the RGB system iis the unit column vector. Similarly,we convert from a CMY color representation to an RGB representationwith the matrix transformation where black is represented In the CMY system as the unit column vector.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

4. HSV COLOR MODEL

lnstead of a set of color primaries, the HSV model uses color descriptions that have a more intuitive appeal to a user. To give a color specification, a user selects a spectral color and the amounts of white and black that are to be added to ohtain differens shades, tints, and tones. Color parameters in this model are hue (H) , saturation(S) and value(V)

The three dimensional representation of the HSV model is derived from the RGB cube. If we imagine viewing the cube along the diagonal from the white vertex to the origin(black), we see an outline of the cube that has the hexagon shape shown in fig. The boundary of the hexagon represents the various hues, and it is used as the top of the HSV hexcone. In the hexcone, saturation is measured along a horizontal; axis and value is along a vertical axis through the center of the hexcone.

Hue is represented as an angle about the vertical axis ranging from 0 degree at red through 360 degrees. Vertices of the hexagon are separated by 60^0 intervals. Yellow is at 60^0, green at 120^0 and cyan opposite e red at H = 180^0 complementary colors are 1800 apart

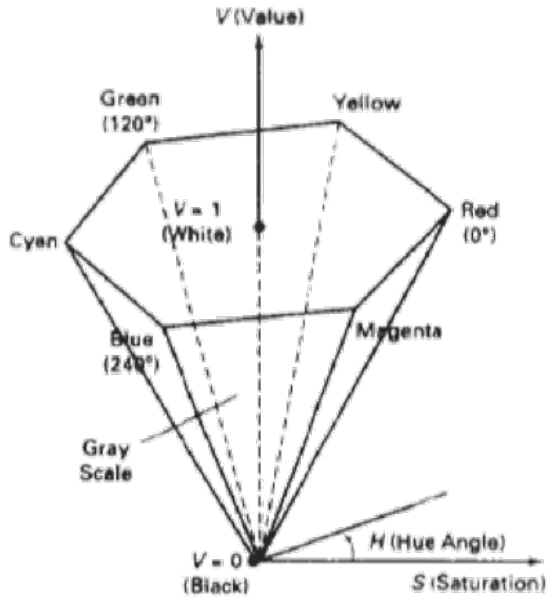


Figure 15-16
The HSV hexcone.

Saturation S varies from 0 to 1. It is represented in this model as the ratio of the purity of a selected hue to its maximum purity at S = 1. A selected hue is said to be one-quarter pure at the value S = 0.25. At S = 0, we have the gray scale.

Value V varies from 0 at the apex of the hexcone to I at the top. The apex represents black. At the top of the hexcone, colors have their maximum intensity. When V = 1 and S = 1, we have the "pure" hues. White is the point at V = 1 andS =o

2.6 ANIMATION

DESIGN OF ANIMATION SEQUENCES

In general, an animation sequence is designed with the tollowing steps:

1. Storyboard layout
2. Object definitions
3. Key-frame specifications
4. Generation of in-between frames

This standard approach for animated cartoons is applied to other animation applications as well, although there are many special application that do not follow this sequence.

The Storyboard layout is an outline of the action. It defines the motion sequence as a

set of basic events that are to take place. Depending on the type of animation to be produced, the storyboard could consist of a set of rough sketches or it could be a list of the basic ideas for the motion.

An object definition is given for each participant in the action. Objects can bedefined interms of basic shapes, such as polygons or splines. In addition, the associatedmovements for each object are speeded along with the shape.

A keyframe is a detailed drawing of the scene at a certain time in the animation sequence.

Within each key frame, each object is positioned according to the time for that frame. Some key frames are chosen at extreme positions in the action; others are spaced so that the time interval between key frames is not to great. More key frames are specified for intricate motions than for simple, slowly varing motions.

In-between frames are the intermediate frames between the key frames.

The nurnber of in-betweens needed is determined by the media to be used to display theanimation. Film requires 24 frames per second, and graphics terminals are refreshedat the rate of 30 to 60 frames per second. Typically, time intervals for themotion are set up so that there arr from three to five in-betweens for each pair ofkey frames.

GENERAL COMPUTER-ANIMATION FUNCTIONS

Some steps in the development of an animation sequence are well-suited to computer solution. These include object manipulations and rendering, camera motions, and the generation of in-betweens. Animation packages, such as Wavefront,

One function available in animation packages is provided to store and manage the object database. Object shapes and associated parameters are stored and updated in the database. Other object functions

include those for motion generation and those for object rendering. Motions can be generated according to specified constraints using two-dimensional or three-dimensional transformations.

RASTER ANIMATIONS

On raster systems, we can generate real-time animation in limited applications using raster operations. Sequences of raster operations can be executed to produce real-time animation of either two-dimensional or three-dimensional objects, as long as we restrict the animation to motions in the projection plane. Then no viewing or visible- surface algorithms need be invoked.The animation is then accomplished by changing the color-table values so that the object is "on"at successively positions along the animation path as the preceding position is set-to the background intensity

COMPUTER-ANIMATION LANGUAGES

Design and control of animation sequences are handled with a set of animationroutines. A general-purpose language, such as C, Lisp, Pascal, or FORTRAN, is often used to program the animation functions, but several specialized animation languages have been developed. Animation functions include

- a graphics editor, a
- key-frame generator,
- an in-between generator,
- and standard graphics routines.

The graphics editor allows us to design and modify object shapes, using spline surfaces, constructive solid-geometry methods, or other representation schemes.

A typical task in an animation specification is scene description. This includes the positioning of objects and light sources, defining the photometric parameters (light-source intensities and surface-illumination properties), and setting the camera parameters (position, orientation, and lens characteristics).

Another standard function is action specification. This involves the layout of motion paths for the objects and camera. And we need the usual graphics routines: viewing and perspective transformations, geometric transformations to generate object movements as a function of accelerations or kinematics path specifications, visible-surface identification, and the surface-rendering operations.

KEY FRAME SYSTEMS

We generate each set of in-betweens from the specification of two (or more) keyframes. Motion paths can be given with a kinematic a s a set of spline curves, or the motions can be physicdly bnscd by specifying the for acting on the objects to be animated.For complex scenes, we can separate the frames into individual components or objects called cels celluloid transparencies)

MORPHING

Transformation of object shapes from one form to another is called morphing,which is a shortened form of metamorphosis. Morphing methods can he applied to any motion or transition involving a change in shape.

MOTION SPECIFICATIONS There are several ways in which the motions of objects can be specified in an animation system.

Direct Motion Specification The most straightforward method for defining a motion sequence is direct specification of the motion parameters. Here, we explicitly give the rotation angles and translation vectors. Then the geometric transformation matrices are applied to transform coordinate positions. Alternatively, we could use an approximating equation to specify certain kinds of motions.

$$y(x) = A\,|\sin(\omega x + \theta_0)|\,e^{-kx}$$

where A is the initial amplitude, w is the angular frequence, 0, is the phase angle,and k is the damping constant. These methods can be used for simple user-programmedanimation sequences.