

UNIT 2

3D Concepts

Three Dimensional Concepts

- When we model and display a 3D scene, there are many more considerations we must take into account besides just including coordinate values for the third dimension.
- Object boundaries can be constructed with various combinations of plane and curved surfaces.
- There are some graphics packages that are use for 3D object representation.
- Viewing transformations in 3D are much more complicated because there may be so many parameters to select when specifying how 3D scene is to be mapped to display device.

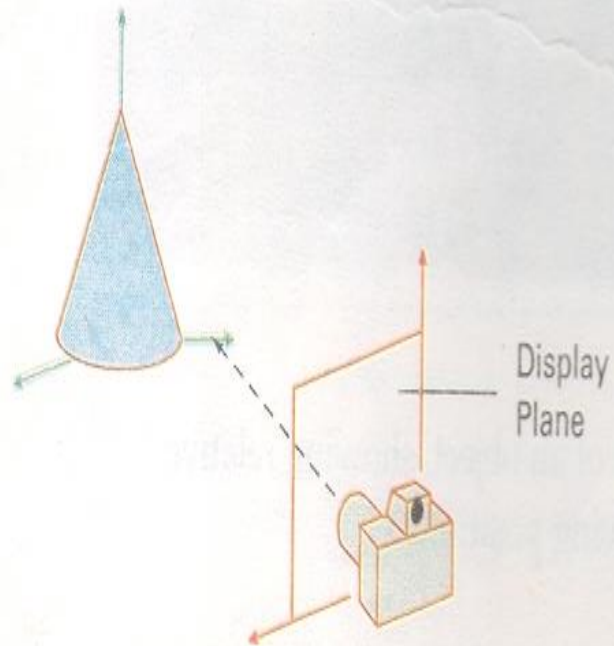



Figure 9-1
Coordinate reference for obtaining
a particular view of a
three-dimensional scene.

- 
- To obtain a display of a 3D scene that has been modeled in world coordinates, we must first set up a coordinate reference for camera.
 - This coordinate reference defines the position and orientation for the plane of the camera film, which is the plane we want to use to display a view of the object 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 form.

Parallel Projection

- One method of generating a view of solid objects is to project points on the object surface **along parallel lines** on to the display plane.
- By selecting different viewing positions we can project visible points on the object onto the display plane to obtain different two dimensional views of the object.
- In a parallel projection parallel lines in the world-coordinate scene project into parallel lines on the two-dimensional display plane.
- This technique is used in engineering and architectural drawings to display an object with a set of views that maintain relative proportions of the object.

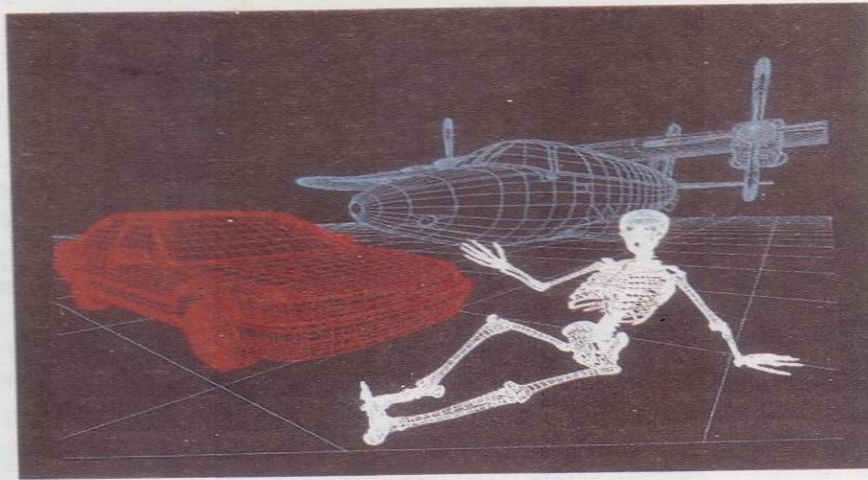


Figure 9-2

Wireframe display of three objects, with back lines removed, from a commercial database of object shapes. Each object in the database is defined as a grid of coordinate points, which can then be viewed in wireframe form or in a surface-rendered form. (Courtesy of Viewpoint DataLabs.)



Figure 9-3

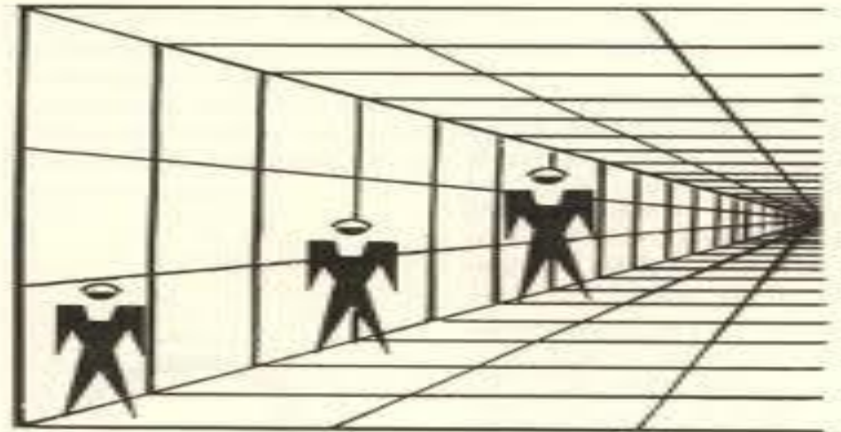
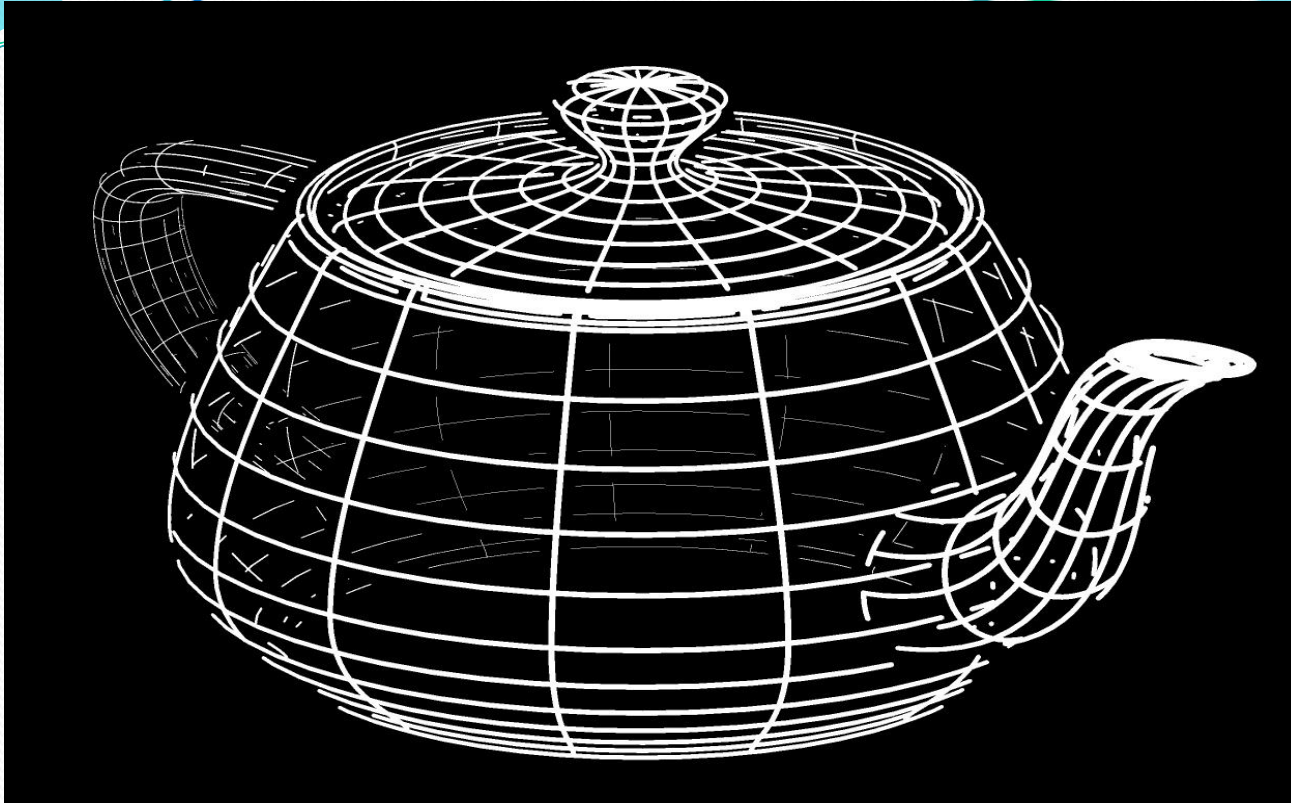
Three parallel-projection views of an object, showing relative proportions from different viewing positions.

Perspective Projection

- Another method of 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 the objects of the same size than are nearer to the viewing position.
- In perspective projection, parallel lines in a scene that are not parallel to the display plane are projected into converging lines.
- Scenes displayed using perspective projection appear more realistic, since this the way that our eyes and camera lens form images.

Depth Cueing

- With a few exceptions, depth information is more important, so that we can easily identify for a particular viewing direction, which is the front and which is the back of the displayed objects.
- There are several ways to include depth info. in two dimensional representation of the solid objects.
- A simple method for indicating depth with wireframe displays is to vary intensity of objects according to their distance from the viewing position. The lines closest to the viewing position are displayed with the highest intensities and lines farther away are displayed with the decreasing intensities.
- Depth cueing is applied by choosing maximum and minimum intensity (color values) and a range of distances over which the intensities are to vary.



Visible Line and Surface Identification

- 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 non-visible lines as dashed lines.
- Another approach is to simply remove the non-visible lines.
- When objects are to be displayed with color or shaded surfaces, we apply surface-rendering procedures to the visible surfaces so that the hidden surfaces are obscured.

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.

Three Dimensional Object Representations

- Representation schemes for solid objects are often divided into two broad categories, although not all representations fall neatly into any one of two categories:
 1. Boundary representations (B-reps).
 2. Space-partitioning representations

Boundary representations (B-reps) :

- This describes a three-dimensional object as a set of surfaces that separate the object interior from the environment.

Space-partitioning representations

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

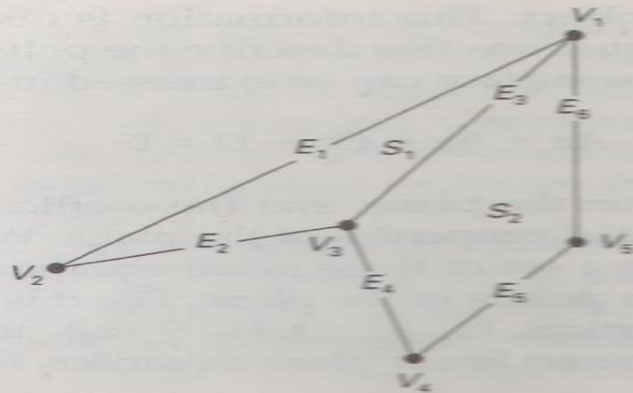
Polygon Surfaces

- It is example of boundary representation method.
- It simplify and speed up the surface rendering and display of objects, since all the surface are described with linear equations.

➤ Polygon Tables

- we specify the polygon surface with a set of vertex coordinate and associated attribute parameters.
- Polygon data table can be organized in two way:
 1. Geometric table
 2. Attribute table

- **Geometric data tables** contains vertex coordinate and parameter to identify the spatial orientation of the polygon surface.
- Attribute data table includes texture characteristics and its surface reflectivity.
- A convenient organization for storing geometric data is to create three list:
 1. **A vertex table**
 2. **An edge table**
 3. **A polygon table**
- Coordinate value for each vertex of an object is stored in **vertex table**.
- **Edge table** contains pointers back into vertex table to identify the vertices of each polygon edge.
- **Polygon table** contains pointer back into 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

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

Curved Lines and Surfaces

- Displays of three-dimensional curved lines and surfaces can be generated from an input set of mathematical functions defining the objects or from a set of user specified data points.
- When functions are specified, a package can project the defining equations for a curve to the display plane and plot pixel positions along the path of the projected function.

Quadric Surfaces

- Frequently used classes of objects are the **quadric surfaces**, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, paraboloids, and hyperboloids.
- Quadric surfaces, particularly spheres and ellipsoids, are common elements of graphics scenes, and they are often available in graphics packages as primitives from which more complex objects can be constructed.

Blobby Objects

- Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other objects.
- Examples in this class of objects include molecular structures, water droplets, muscle shapes in the human body etc.
- These objects can be described as exhibiting “blobbiness” and are often simply referred to as blobby objects, since their shapes show a certain degree of fluidity.

Spline Representations

- In drafting terminology, a Spline is a flexible strip used to produce a smooth curve 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.
- The term spline curve originally referred to a curve drawn in this manner.
- In computer graphics, the term spline curve now refers to any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces.
- A spline surface can be described with two sets of orthogonal spline curves.

3-D Transformations

TRANSLATION

- In a three-dimensional homogeneous coordinate representation, a point is translated from position $P = (x, y, z)$ to position $P' = (x', y', z')$ with the matrix operation

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

OR

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

Translation

- 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

$$x' = x + t_x,$$

$$y' = y + t_y,$$

$$z' = z + t_z$$

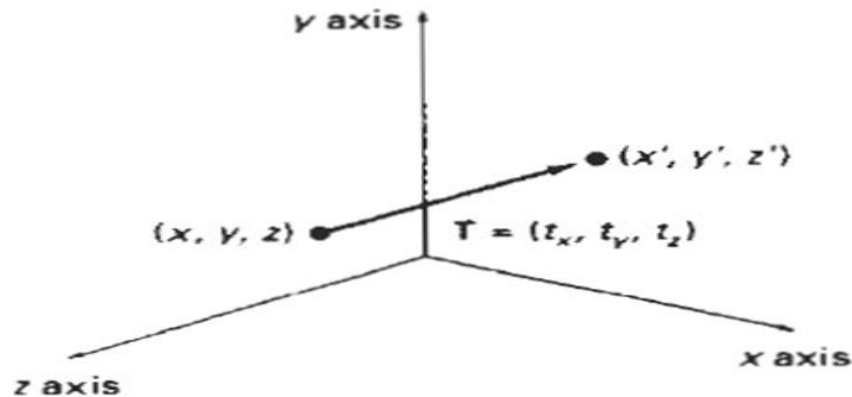
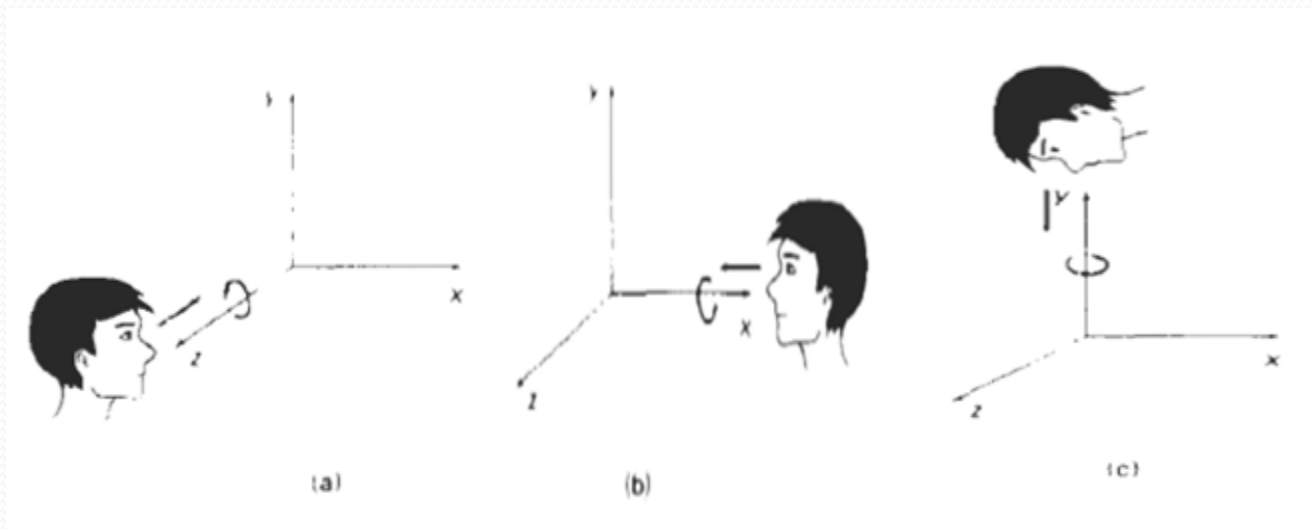


Figure 11-1

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

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**.



Rotation

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

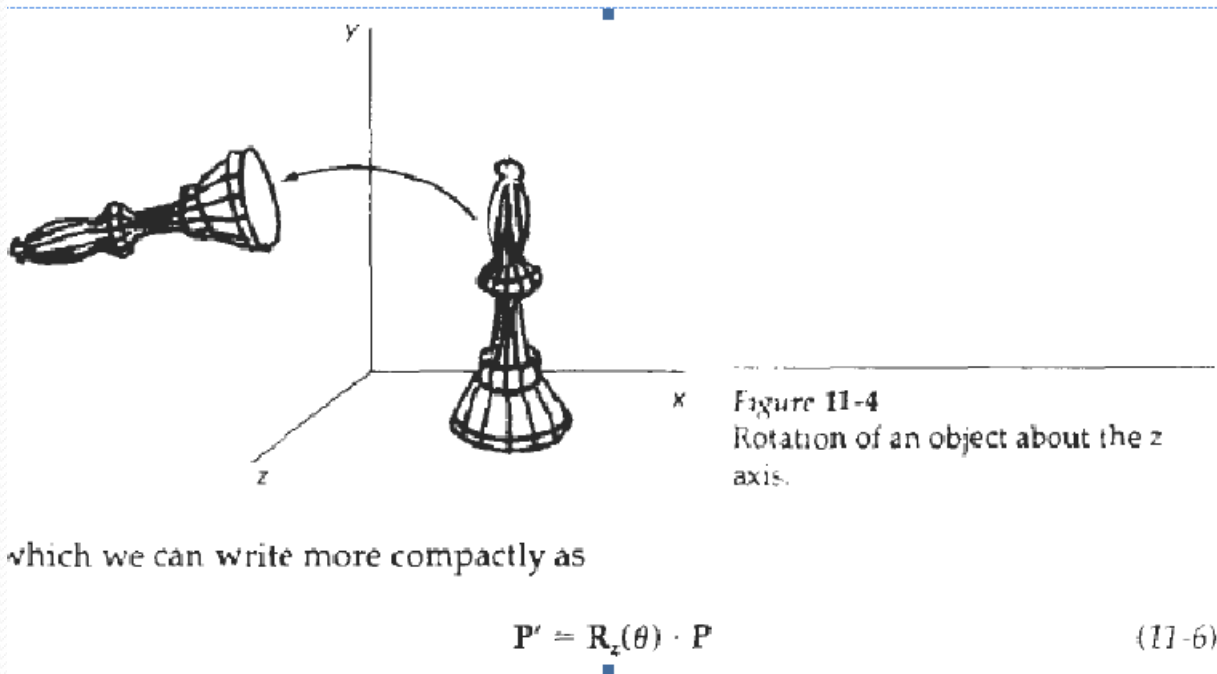
$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

$$z' = z$$

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

Rotation



Rotation

- The equations for an **x-axis rotation**:

$$y' = y \cos \theta - z \sin \theta$$

$$z' = y \sin \theta + z \cos \theta$$

$$x' = x$$

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

Rotation

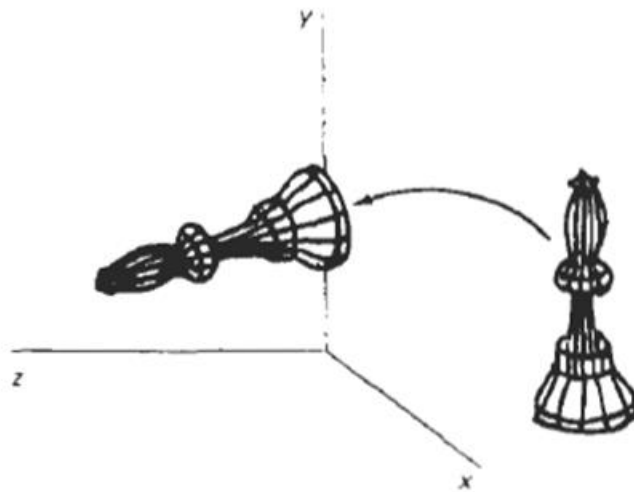


Figure 11-6
Rotation of an object about the
x axis.

or

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

(11-10)

Rotation

- the transformation equations for a **y-axis rotation**:

$$z' = z \cos \theta - x \sin \theta$$

$$x' = z \sin \theta + x \cos \theta$$

$$y' = y$$

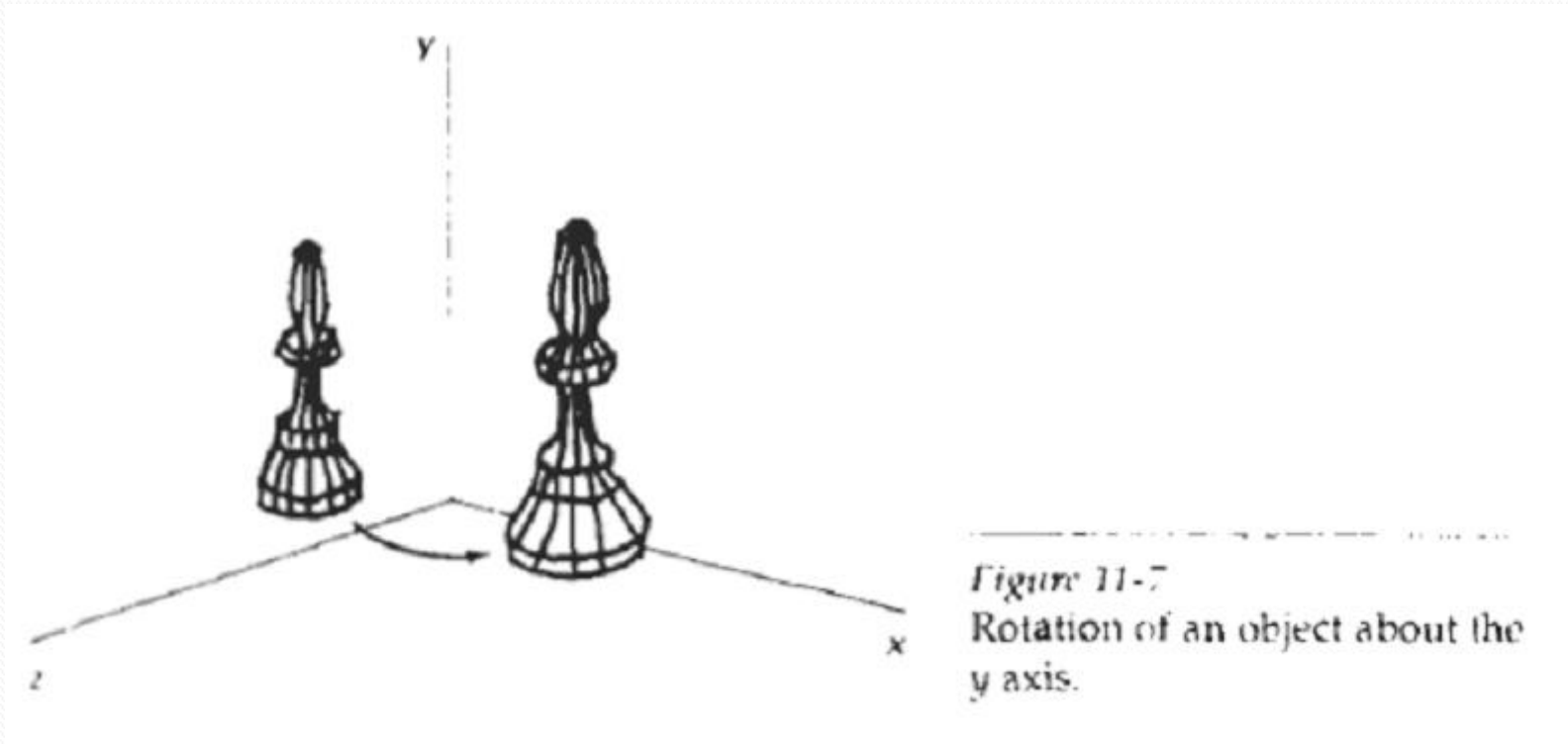
The matrix representation for y-axis rotation is

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

or

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

Rotation



Scaling

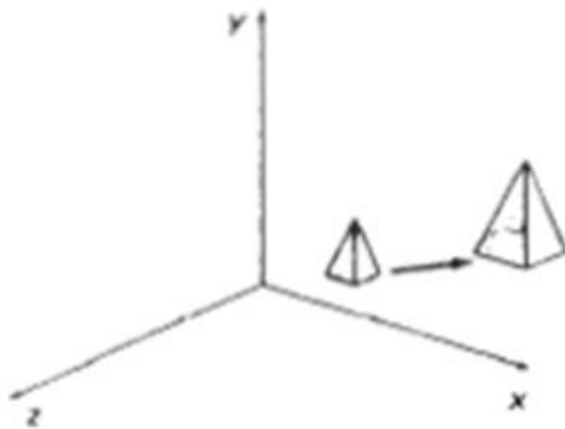
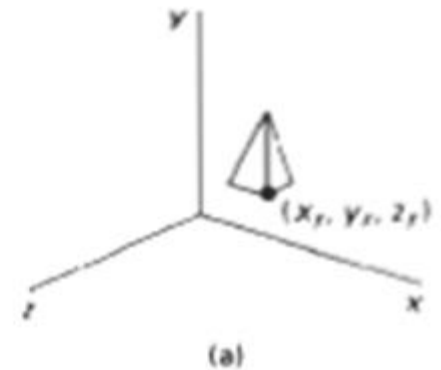


Figure 11-17

Doubling the size of an object with transformation 11-42 also moves the object farther from the origin.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(11-42)



OR

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

(11.43)

Scaling

- Explicit expressions for the coordinate transformations for scaling relative to the origin are

$$x' = x \cdot s_x, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z$$

Scaling with respect to a selected fixed position (**Xf**, **Yf**, **Zf**,) 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.

Scaling

The matrix representation for an arbitrary fixed-point scaling can then be expressed as the concatenation of these translate-scale-translate transformations as

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

Visible Surface Detection

- A major consideration in the generation of realistic graphics displays is identifying those parts of a scene that are visible from a chosen viewing position.
- Numerous algorithms have been devised for efficient identification of visible objects for different types of applications.
- Some methods require more memory, some involve more processing time, and some apply only to special types of objects.

Classification of Visible Surface Detection Algorithms

- An object-space method compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.
- In an image-space algorithm, visibility is decided point by point at each pixel position on the projection plane. Most visible-surface algorithms use image-space methods,
- Line display algorithms, on the other hand, generally use object-space methods to identify visible lines in wire frame displays, but many image-space visible-surface algorithms can be adapted easily to visible-line detection.

BACK-FACE DETECTION

- 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 the coefficients A, B, C, D are constants describing the spatial properties of the plane.


- A fast and simple object-space method for identifying the **back faces** of a polyhedron is based on the "**inside-outside**" tests.
- A point (x, y, z) is "**inside**" a polygon surface with plane parameters A, B, C , and D if

$$Ax + By + Cz + D < 0$$

- When an inside point is along the line of sight to the surface, the polygon must be a back face.

DEPTH-BUFFER/Z-BUFFER METHOD

- A commonly used image-space approach to detecting visible surfaces is the depth-buffer method, which compares surface depths at each pixel position on the projection plane.
- This procedure is also referred to as the z-buffer method, since object depth is usually measured from the view plane along the z axis of a viewing system.
- Each surface of a scene is processed separately, one point at a time across the surface.
- For each pixel position (x, y) on the view plane, object depths can be compared by comparing z values.

- 
- Following figure shows three surfaces at varying distances along the orthographic projection line from position (x, y) in a view plane taken as the $X_v Y_v$ plane.
 - Surface S_1 , is closest at this position, so its surface intensity value at (x, y) is saved.

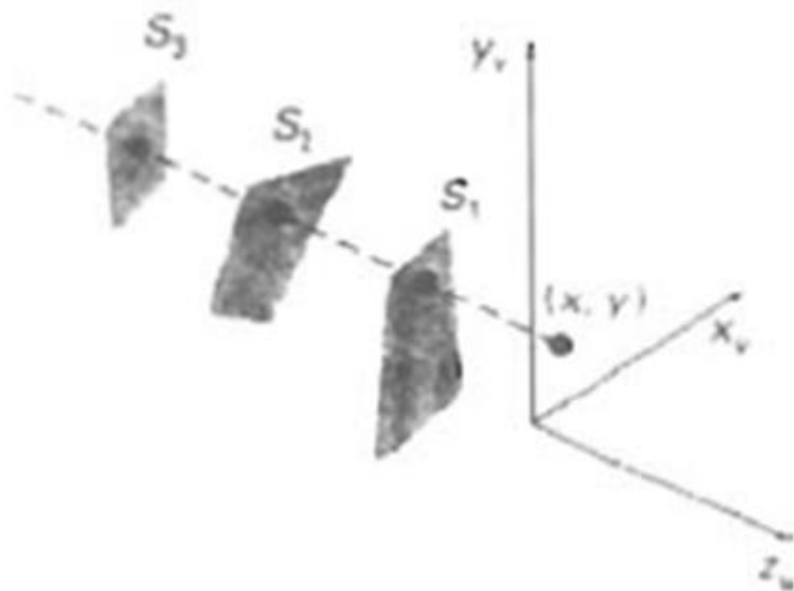




Figure 13-4

At view-plane position (x, y) , surface S_1 has the smallest depth from the view plane and so is visible at that position.

- 
- As implied by the name of this method, two buffer areas are required:
 1. A depth buffer is used to store depth values for each (x,y) position as surfaces are processed, and
 2. The refresh buffer stores the intensity values for each position
 - Initially, all positions in the depth buffer are set to o(minimum depth).
 - The refresh buffer is initialized to the background intensity.

- 
- Each surface listed in the polygon tables is then processed, one scan line at a time, calculating the depth (z value) at each (x,y) pixel position.
 - The calculated depth is compared to the value previously stored in the depth buffer is at that position.

- If the calculated depth is greater than the value stored in the depth buffer, the new depth value is stored, and the surface intensity at that position is determined and placed in the same x,y location in the refresh buffer.

Depth values for a surface position (x, y) are calculated from the plane equation for each surface:

$$z = \frac{-Ax - By - D}{C} \quad (13.4)$$

Steps of Z-Buffer Algorithm

Step-1 –

Set the buffer values –
Depthbuffer $x,y = 0$
Framebuffer $x,y = \text{background color}$

Step-2 – Process each polygon One at a time

For each projected x,y pixel position of a polygon, calculate depth z .

If $Z > \text{depthbuffer } x,y$

Compute surface color,
set depthbuffer $x,y = z$,
framebuffer $x,y = \text{surface color } x,y$

3D Transformation Pipeline

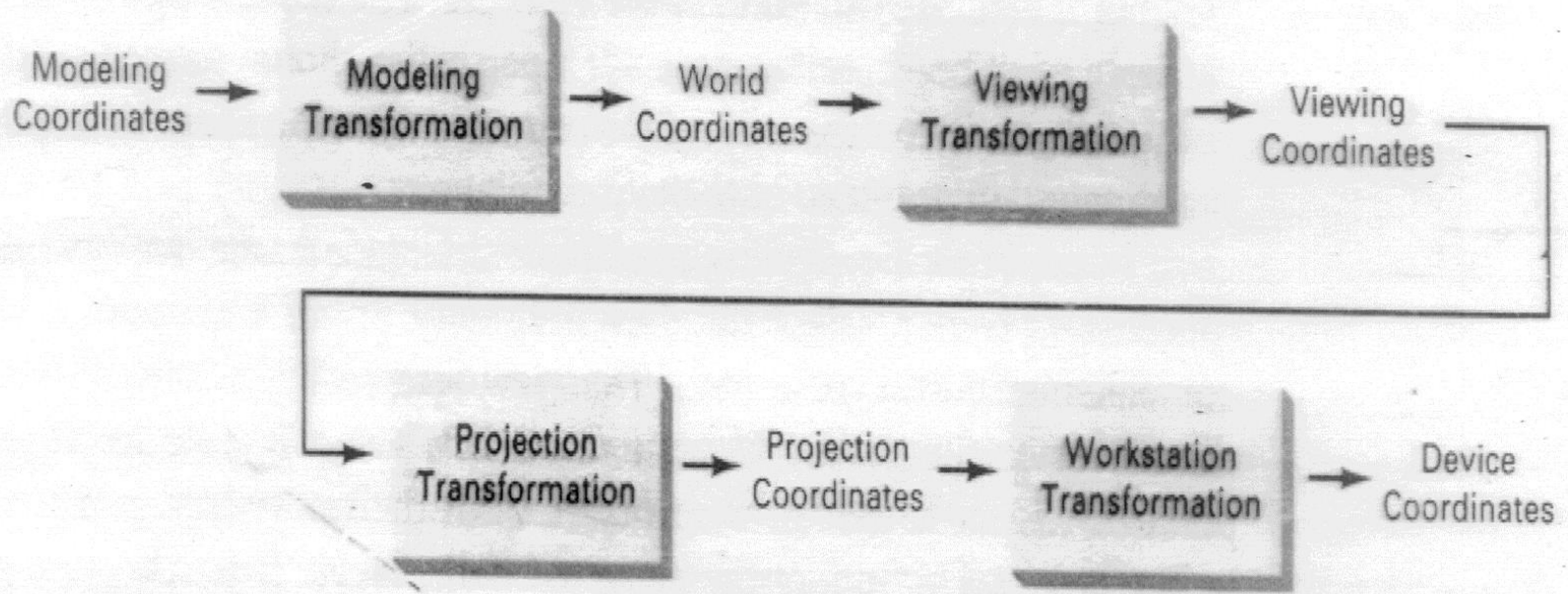


Figure 12-2

General three-dimensional transformation pipeline, from modeling coordinates to final device coordinates.