CHAPTER – 5

3D Graphics System

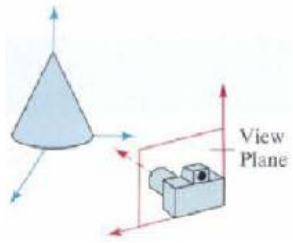
5.1 Three-Dimensional Concepts

- *What are the issue in 3D that makes it more complex than 2D?*

  → When we model and display a three-dimensional scene, there are many more considerations we must take into account besides just including coordinate values as 2D, some of them are:

  * Relatively more co-ordinate points are necessary.
  * Object boundaries can be constructed with various combinations of plane and curved surfaces.
  * Consideration of projection (dimension change with distance) and transparency.
  * Many consideration on visible surface detection and remove the hidden surfaces. Sometime we have to represent the hidden surfaces by dashes on the hardwired structure.
  * Lightning effect and intensity distribution with reference to the light source.

3D display method



* Three-dimensional viewing coordinates must be transformed onto two dimensional device coordinates to view the scene.
* To obtain a 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 camera film, which is the plane we want to use 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 apply lighting and surface-rendering techniques to shade the visible surfaces.

5.2 3D Geometric Transformation

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 to be moved in each of the three coordinate directions.

2D transformations can be represented by 3 x 3 matrices using homogenous coordinates, so 3D transformations can be represented by 4 x 4 matrices, providing we use homogeneous coordinate representations of points in 2 spaces as well. Thus instead of representing a point as (x, y, z), we represent if as (x, y, z, H), where two these quadruples represent the same point it one is a non-zero multiple of the other the quadruple (0, 0, 0, 0) is not allowed. A standard representation of a point (x, y, z, H) with H not zero is given by (x/H, y/H, z/H, 1). Transforming the point to this form is called homogenizing.

A. Translation

A point is translated from position P=(x, y, z) to position P' = (x', y', z') with the matrix operation as:

P' = T.P

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

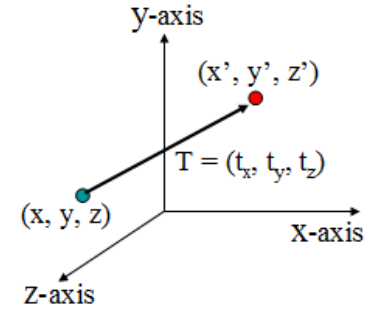


Fig. Translating a point or an object with translating vector T = (t_x, t_y, t_z)

Parameters t_x, t_y, t_z specify translation distances for the coordinate directions x, y and z. This matrix representation is equivalent to three equations: $x' = x + t_x$

$y' = y + t_y$

$z' = z + t_z$

B. Scaling

- Scaling changes size of an object and repositions the object relative to the coordinate origin.
- If transformation parameters are not all equal then figure gets distorted
- So we can preserve the original shape of an object with uniform scaling ($s_x = s_y = s_z$)
- Matrix expression for scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as :

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

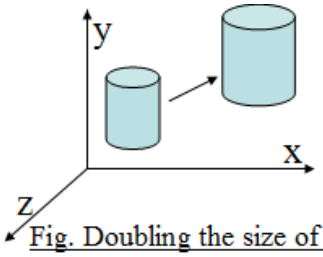


Fig. Doubling the size of an object with transformation

Scaling with respect to a selected fixed point (x_f, y_f, z_f) can be represented with:

- Translate fixed point to the origin
- Scale object relative to the coordinate origin
- Translate fixed point back to its original position

Or $T_{(x, y, z)} \cdot S_{(x, y, z)} \cdot T_{(-x, -y, z)}$

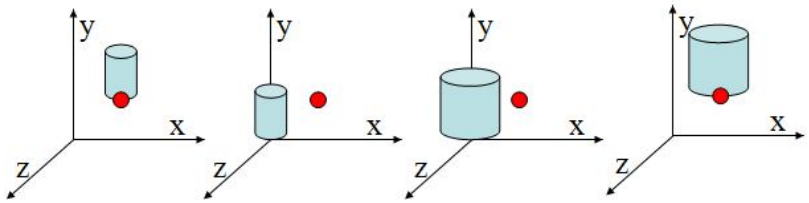


Fig. Scaling an object relative to a selected fixed point without transformation to that point

C. Reflection

A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection 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 degree rotations about that axis. Reflections with respect to a plane are equivalent to 180 degree rotations in four-dimensional space.

- ***The matrix representation for the reflection of a point relative to XY-plane is given by***

$$RFz = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

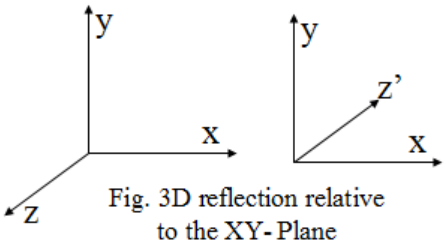


Fig. 3D reflection relative to the XY- Plane

- This transformation changes the sign of the z coordinates, leaving the x and y coordinate values unchanged.

- ***The matrix representation for the reflection of a point relative to XZ-plane is given by***

$$RFy = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

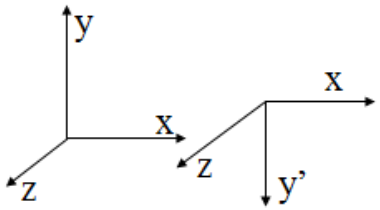


Fig. 3D reflection relative to the ZX- Plane

- ***The matrix representation for the reflection of a point relative to XZ-plane is given by***

$$RFx = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

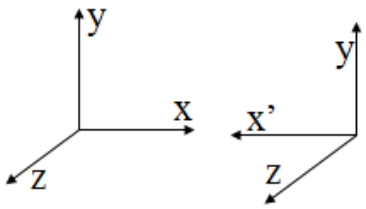


Fig. 3D reflection relative to the YZ- Plane

- *How do you reflect an arbitrary plane characterized by normal vector 'N' in 3D plane?*

D. Shearing

Shearing transformations are used to modify object shapes. E.g. shears relative to the z axis:

$$SHz = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

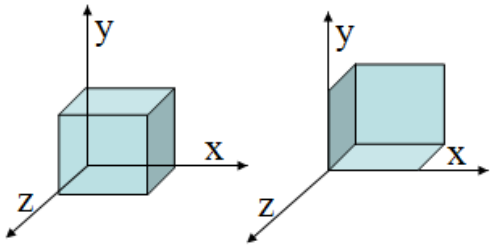


Fig. A unit cube (a) is sheared (b) by transformation matrix with a = b = 1

- The parameter 'a' & 'b' are assuming any real values.
- It alters the x and y coordinate values by an amount that is proportional to the z value while leaving the z coordinate unchanged.

E. Rotation

To generate a rotation transformation for an object in 3D space, we require the following:

- Angle of rotation.
- Pivot point
- Direction of rotation
- Axis of rotation

- Axes that are parallel to the coordinate axes are easy to handle.

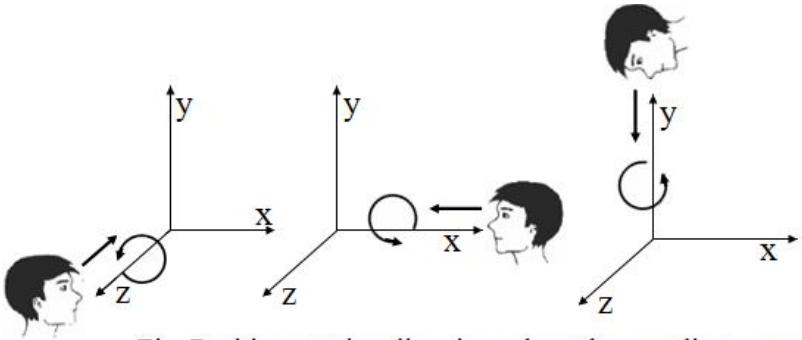
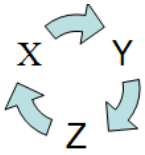


Fig. Positive rotation directions about the coordinate axes are Counterclockwise, when looking toward the origin from a positive coordinate position on each axis.

- Cyclic permutation of the coordinate parameters x, y and z are used to get transformation equations for rotations about the coordinates

x → y → z →



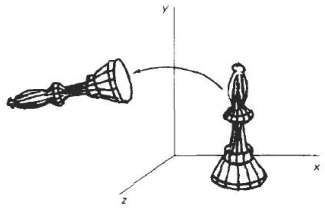
- Taking origin as the centre of rotation, when a point P(x, y, z) is rotated through an angle θ about any one of the axes to get the transformed point P'(x', y', z'), we have the following equation for each.

  - 3D z-axis rotation equations are expressed in homogenous coordinate form as

    $x' = x\cos\theta - y\sin\theta$
    $y' = x\sin\theta + y\cos\theta$
    $z' = z$ .....(i)

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

    P' = Rz (θ) . P

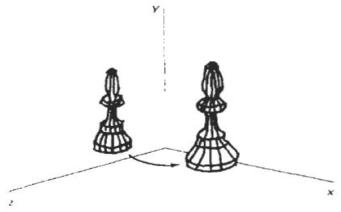
    

  - 3D y-axis rotation equations are expressed in homogenous coordinate form as

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

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

    P' = Ry (θ) . P

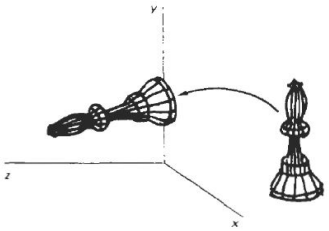
    

  - 3D x-axis rotation equations are expressed in homogenous coordinate form as

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

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

    P' = Rx (θ) . P

Rotation about an axis parallel to one of the coordinate axes:

Steps:

- Translate object so that rotation axis coincides with the parallel coordinate axis.
- Perform specified rotation about that axis
- Translate object back to its original position.

ie. **$P' = [\, T^{-1} \,.\, R_x(\theta) \,.\, T \,] \,.\, P$**

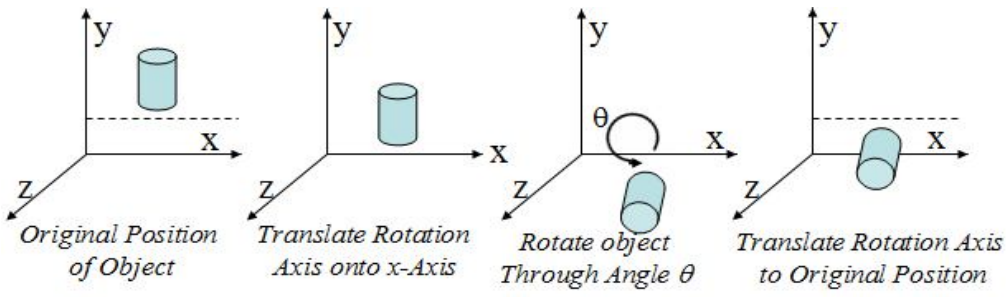


Fig. Sequence of transformations for rotating an object about an axis that is parallel to the *x* axis

Rotation about any arbitrary axis in 3D Space

Here,

Composite matrix = $T'R_y'R_x'R_{FZ}R_yR_xT$

Step-1: Translate the arbitrary axis so that it passes thru origin.

Here, the translation matrix is given by:

$$T = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step-2: Align the arbitrary axis on any major co-ordinate axis (z-axis)

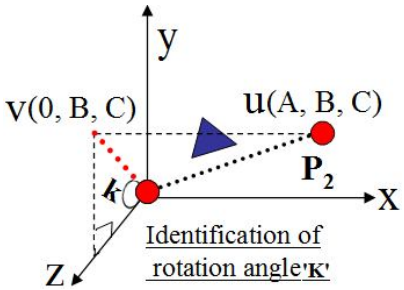
Here, at first to find the angle of rotation, project the arbitrary axis on yz-plane so that the x-coordinate will be eliminated. Here, we can find angle 'k' for rotation about x-axis by projection & 'α' for rotation about y-axis directly. After these two rotations, the arbitrary axis will align with the z-axis.

2.a: Rotation about x-axis by angle 'k' in clockwise direction so that the arbitrary axis lies on xz-plane.

Here,

Sin k = B/V &

Cos k = C/V



Now, the translation matrix is given by:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos k} & \text{-Sin k} & 0 \\ 0 & \text{Sin k} & \text{Cos k} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & -B/V & 0 \\ 0 & B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

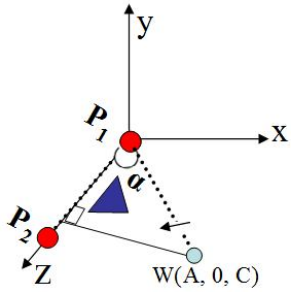
2.b: Rotation about y-axis by angle 'α' in clockwise direction so that the arbitrary axis align with z-axis.

Here,

Sinα= A/W &

Cosα= C/W

Now, the translation matrix is given by:



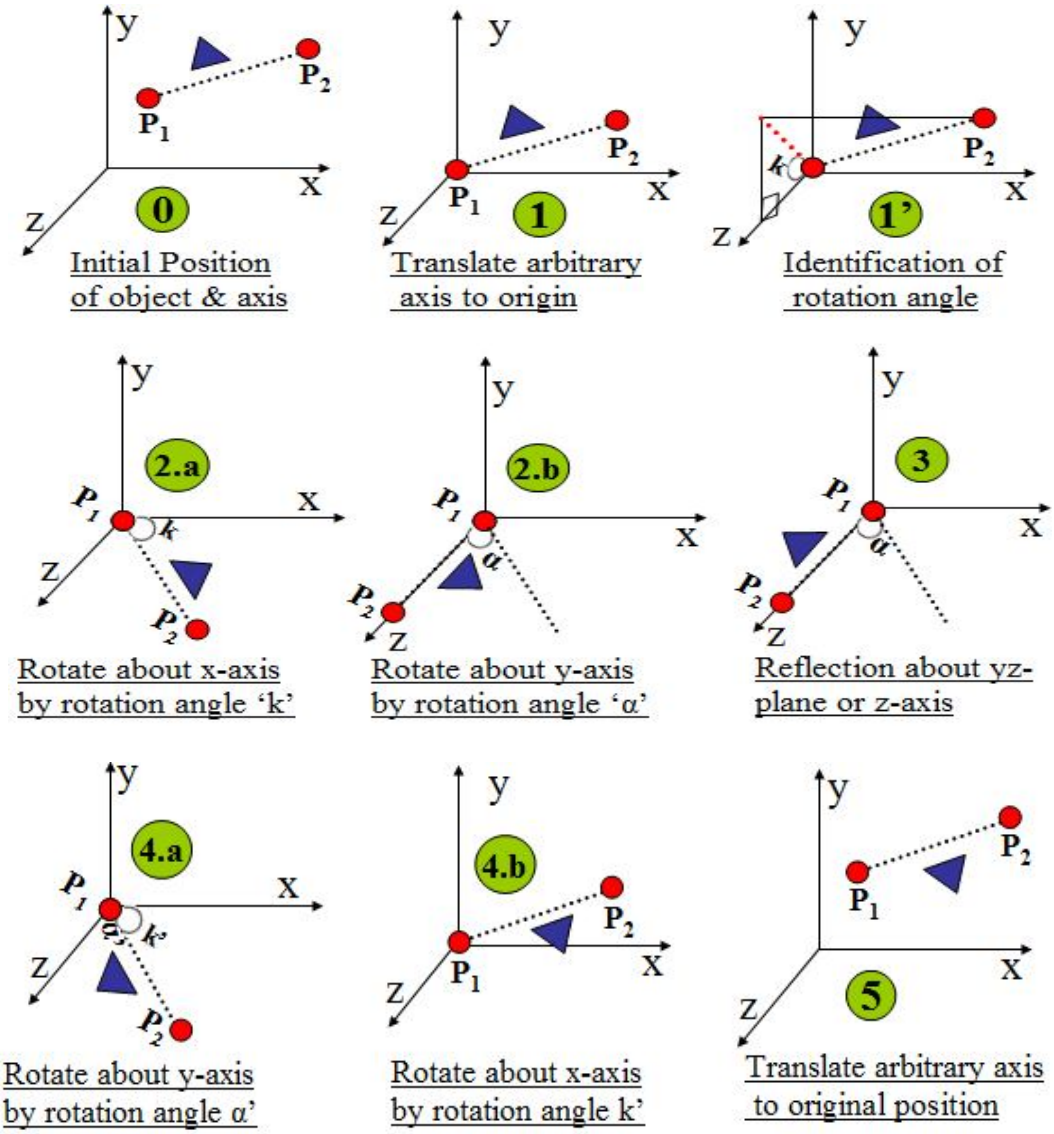
$$R_y = \begin{pmatrix} \text{Cos}\alpha & 0 & \text{-Sin}\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \text{Sin}\alpha & 0 & \text{Cos}\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} C/W & 0 & -A/W & 0 \\ 0 & 1 & 0 & 0 \\ A/W & 0 & C/W & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step-3: Reflect the object about yz-plane or z-axis

$$RFz = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step-4: Perform inverse rotation about y-axis & then x-axis

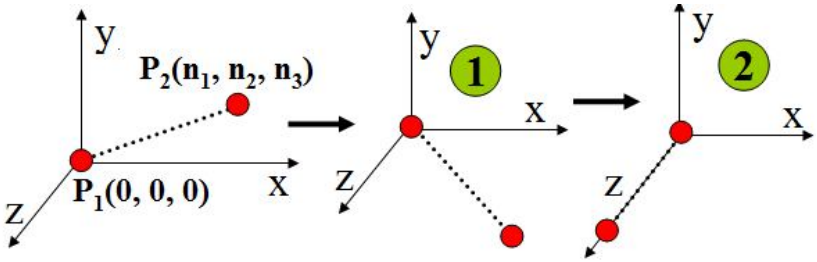
Step-5: Perform inverse translation



 # Find the alignment transformation matrix (AN) that aligns a given vector N = $n_1I + n_2J + n_3K$ to positive z-axis.

Solution

Here,
The given vector is passing through the origin with end points (n_1, n_2, n_3) in 3D space.



Thus, we have to only perform rotation about x-axis & then about y-axis to align the vector on the z-axis.

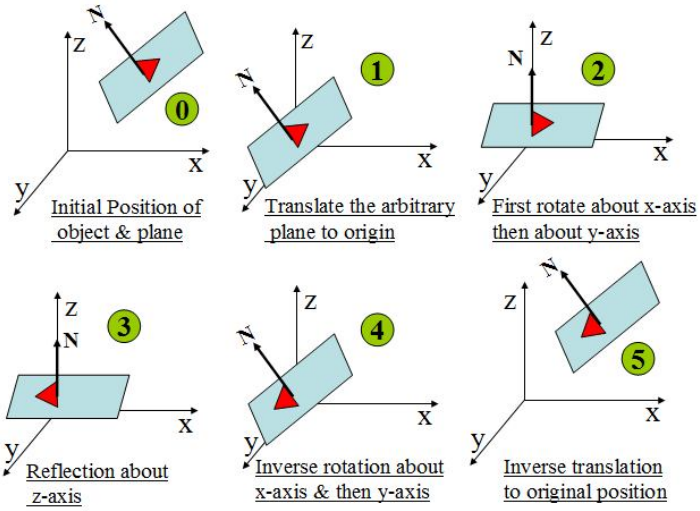
Hence, the composite matrix is given by: Com = $R_y * R_x$

Note: - Here, the angle for rotation about x-axis is identified by the given value (n_1, n_2, n_3).

Rotation about any arbitrary plane in 3D Space

The rotation about any arbitrary plane perform same operation as the rotation about any arbitrary line, the only difference is that we have to characterize the rotation by any normal vector 'N' in that plane.

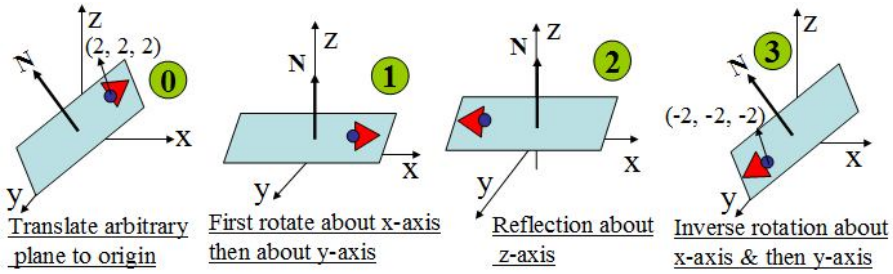
Here,
Composite matrix = $T'R_y'R_x'R_{FZ}R_yR_xT$



 # Find the mirror reflection with respect to a plane passing through point (2, 2, 2) & having normal vector N = I + J + K (Note: if the vector are represented by small letter then we have to put an arrow on their head).

Solution

Here, the composite matrix is given by: Com = $R_y' * R_x' * R_y * R_x$



Note: Here, the angle of rotation about x-axis is 45 degree as the normal passes that angle (according to given vector).

5.2 3D Object Representation

Graphics scenes can contain many different kinds of objects like trees, flowers, clouds, rocks, water etc. these cannot be describe with only one methods but obviously require large precisions such as polygon & quadratic surfaces, spline surfaces, procedural methods, volume rendering, visualization techniques etc. 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. For examples: polygon surfaces 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, non-overlapping, contiguous solids (usually cubes). For example ***octree*** representation.

The visual realism in 3D object can be maintained by maintaining the transparency, projection, lighting & shading effect on each portion. The representation of 3D object include following three stages:

1. Represent the objects with multiple polygons i.e. wired-frame representation.
2. Fill all the polygons either with flat filling or smooth filling.
3. Give the lightning or shading effect and the coloring effect.

5.3 Polygon Surfaces

A set of polygon surfaces are used to represent object boundary that enclose the object interior in 3D graphics. This simplifies and speeds up the surface rendering and display of objects, since all surfaces are described with linear equations of plane: Ax + By + Cz + D = 0. For this reason, polygon descriptions are often referred to as "standard graphics objects." The wireframe representations are common in design & solid modeling application because they can be displayed quickly to of a give a general indication of the surface structure.

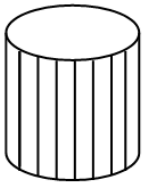


Fig. Wireframe model of a cylinder with back (hidden) lines removed

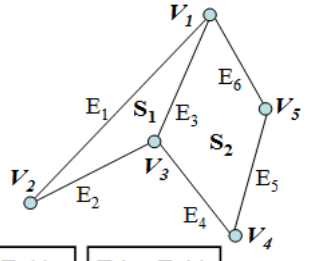
Polygon Tables

To specify a polygon surface, a set of vertex coordinates and associated attribute parameters are placed into tables & that are used in the subsequent processing, display, error checking and manipulation of the objects in a scene. Polygon data tables can be organized into two groups:

a) Geometric tables

Geometric data tables contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces. 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.
* The ***polygon 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 Table
S_1: E_1, E_2, E_2
S_2: E_3, E_4, E_5, E_6

b) Attribute tables

Attribute information for an object includes parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics. The above three table also include the polygon attribute according to their pointer information.

- *What is boundary representation? How are polygon tables useful for modeling 3D surfaces?*

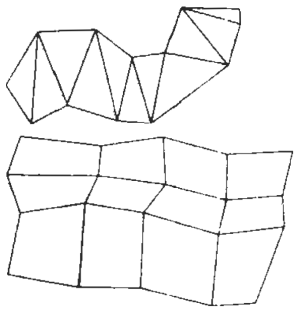
Plane Equations

The equation for a plane surface can be expressed as: Ax + By + Cz + D =0, Where, (x, y, z) is any point on the plane- coefficients ABCD are constants describing the spatial properties of the plane.

* If Ax + By + Cz + D < 0 then the point (x, y, z) is inside the surface
* If Ax + By + Cz + D > 0 then the point (x, y, z) is outside the surface

Polygon Meshes

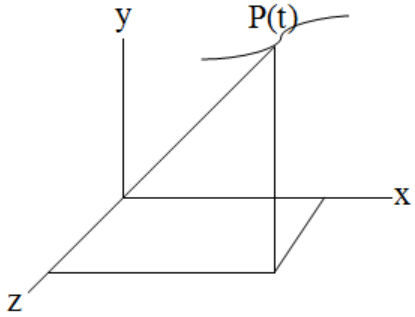
A polygon mesh is a collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons & hence bounded the planer surface. One type of polygon mesh is the ***triangle strip*** that produces n – 2 connected triangles, given the coordinates for n vertices. Another similar functions the ***quadrilateral mesh*** that generates a mesh of (n-1).(m-1) quadrilaterals, given the coordinates for an n by m array of vertices.



5.4 Parametric Cubic Curve

A parametric cubic curve is defined as $P(t) = \sum_{i=0}^{3} a_i t^i$ $\quad 0 <= t <= 1$ ------ (i)

Where, P (*t*) is a point on the curve



Expanding equation (i) yield

$P(t) = a_3t^3 + a_2t^2 + a_1t + a_0$ ---------- (ii)

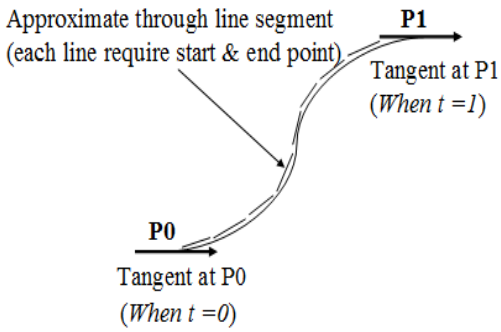
This equation is separated into three components of P (t)

$x(t) = a_{3x}t^3 + a_{2x}t^2 + a_{1x}t + a_{0x}$

$y(t) = a_{3y}t^3 + a_{2y}t^2 + a_{1y}t + a_{0y}$

$z(t) = a_{3z}t^3 + a_{2z}t^2 + a_{1z}t + a_{0z}$ ----------(iii)

- To be able to solve (iii) the twelve unknown coefficients a_{ij} (algebraic coefficients) must be specified
- From the known end point coordinates of each segment, six of the twelve needed equations are obtained.
- The other six are found by using tangent vectors at the two ends of each segment
- The direction of the tangent vectors establishes the slopes (direction cosines) of the curve at the end points
- This procedure for defining a cubic curve using ***end points*** and ***tangent vector*** is one form of **Hermite interpolation**
- Each cubic curve segment is parameterized from 0 to 1 so that known end points correspond to the limit values of the parametric variable *t*, that is P(0) and P(1)
- Substituting *t* = 0 and *t* = 1 the relationship between two end point vectors and the algebraic coefficients are found



P (*0*) = a_0 $\qquad$ **P (*1*) = $a_3 + a_2 + a_1 + a_0$** ---------- (IV)

- To find the tangent vectors equation (ii) must be differentiated with respect to t

P' (*t*) = $3a_3t^2 + 2a_2t + a_1$

- The tangent vectors at the two end points are found by substituting t = 0 and t = 1 in this equation

P' (*0*) = a1 $\qquad$ **P' (*1*) = $3a_3 + 2a_2 + a_1$** ---------- (V)

- The algebraic coefficients 'a_i' in equation (ii) can now be written explicitly in terms of boundary conditions – endpoints and tangent vectors are

a_0= P (*0*) $\qquad$ **a_1= P' (0)**

a_2= -3 P (0) - 3 P (1) -2 P' (0) - P' (1) $\qquad$ **a3 = 2 P (0) - 2 P (1) + P' (0) + P' (1)**

(Note: - The value of a2 & a3 can be determined by solving the equation IV & V)

- Substituting these values of 'a_i' in equation (ii) and rearranging the terms yields

P(t) = $(2t^3 - 3t^2 + 1)$ P(0) + $(-2t^3 + 3t^2)$ P(1) + $(t^3 - 2t^2 + t)$ P'(0)+ $(t^3 - t^2)$ P'(1)

Spline curve & Spline surface

* In computer graphics, continuous curve that are formed with polynomial pieces with certain boundary conditions are called ***spline curve*** or simply ***spline***. A ***spline surface*** can be described with two sets of orthogonal spline curves.
* 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 the position of ***spline curve*** as requirement.

* We can mathematically describe such a curve with a piecewise cubic polynomial function whose first and second derivatives are continuous across the various curve sections.
* Splines are used in graphics applications to design curve and surface shapes, to digitize drawings for computer storage, and to specify animation paths for the objects or image. Typical CAD applications for splines include the design of automobile bodies, aircraft and spacecraft surfaces, and ship hulls.

Control points

We specify a spline curve 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 piecewise continuous parametric polynomial functions in one of two ways.

- ***Interpolation curve***: The polynomial sections are fitted by passing the curve through each control points. Interpolation curves are commonly used to digitize drawings or to specify animation paths.
- ***Approximation curve***: The polynomials are fitted to the general control-point path without necessarily passing through any control point. Approximation curves are primarily used as design tools to structure object surfaces.

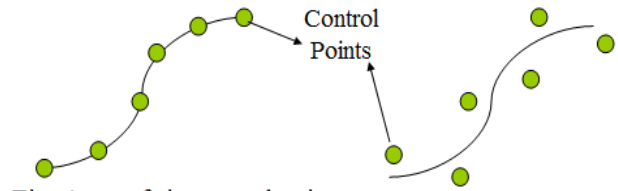


Fig. A set of six control points ***interpolated*** with piecewise continuous polynomial sections

Fig. A set of six control points ***approximated*** with piecewise continuous polynomial sections

A spline curve is defined, modified, and manipulated with operations on the control points. By interactively selecting spatial positions for the control points, a designer can set up an initial curve. After the polynomial fit is displayed for a given set of control points, the designer can then reposition some or all of the control points to restructure the shape of the curve. In addition, the curve can be translated, rotated, or scaled with transformations applied to the control points. CAD packages can also insert extra control points to aid a designer in adjusting the curve shapes.

5.5 Bezier Curve and Surfaces

Bezier splines are highly useful, easy to implement and convenient for curve and surface design so are widely available in various CAD systems, graphics packages, drawing and painting packages.

Bezier curve

In general, a 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 the Bezier polynomial. As with the interpolation splines, a Bezier curve can be specified with boundary conditions, with a characterizing matrix, or with ***blending functions***. The Bezier curve has two important properties:

1. It always passes through the first and last control points.
2. It lies within the convex hull (convex polynomial boundary) of the control points. This follows from the properties of Bezier blending function: they are positive and their sum is always 1, i.e.

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

Suppose we are given n + 1 control-point positions: $\mathbf{p}_k = (x_k, y_k, z_k)$, 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 p_0 and p_n.

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

This, vector equation represents a set of three parametric equations for the individual curve co-ordinates:

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

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

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

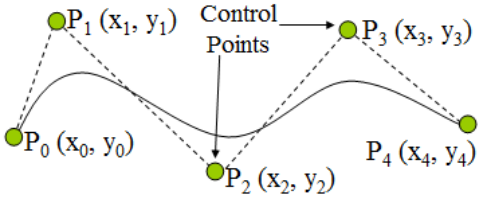
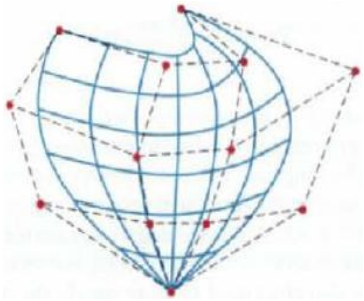


Fig. Bezier curve with 4-control points in xy-plane (z=0)



The Bezier blending functions $BEZ_{k,n}(u)$ are the Bernstein polynomials:

$$BEZ_{k.n}(u) = C(n, k)\, u^k (1-u)^{n-k}$$

Where the ***C (n, k)*** are the binomial coefficients: $C(n, k) = \dfrac{n!}{k!\,(n-k)!}$

Bezier Non-planar 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:

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

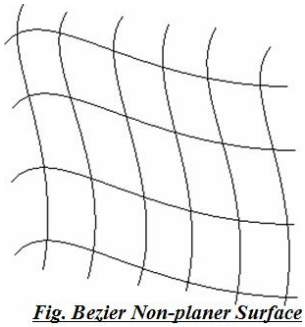
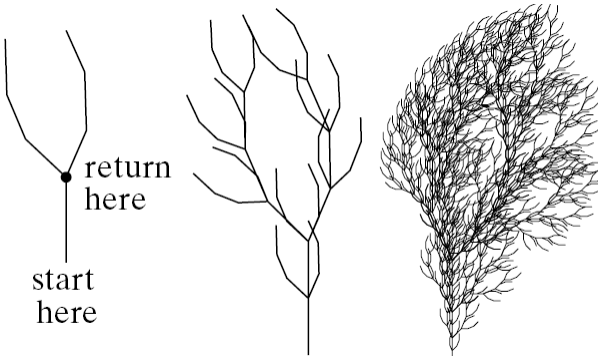


Fig. Bezier Non-planer Surface

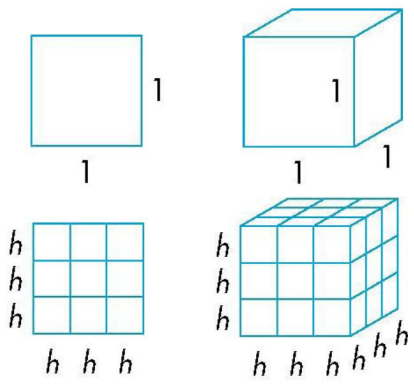
With $\mathbf{p}_{j,k}$ specifying the location of the ***(m*** + **1)** by ***(n*** + **1)** control points

5.6 Fractal Geometry Method

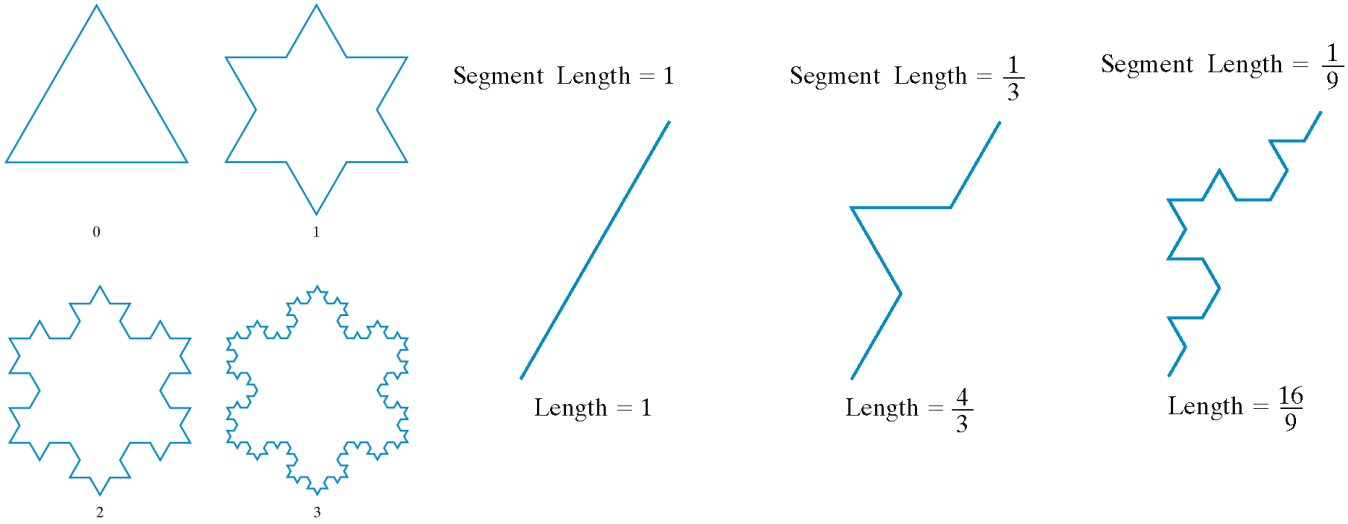
All the previous object representations use Euclidean-geometry methods in which object shapes were described with equations and hence these objects have smooth surfaces and regular shapes. But natural objects, such as mountains and clouds, have irregular or fragmented features, and these are described with fractal-geometry methods. A *fractal* is a mathematical set that typically displays self-similar patterns or recursive operations, which means it is "the same from near as from far".



Fractional object describes and explains the features of natural phenomena with procedures or function in various dimensions that theoretically repeat an infinite number of times but finite number of steps for graphics display. For fractional-Generation procedure, If $P_0 = (x_0, y_0, z_0)$ is a selected initial point, each iteration of a transformation function F generates successive levels of detail with the calculations are: $P_1 = F(P_0)$, $P_2 = F(P_2)$, …



We can describe the amount of variation in the object detail with a number called the ***fractal dimension.*** Unlike the Euclidean dimension, this number is not necessarily an integer. The fractal dimension of an object is sometimes referred to as the ***fractional dimension,*** which is the basis for the name "fractal".



5.7 3D Viewing Transformation

In two-dimensional graphics applications, viewing operations transfer positions from the world-coordinate plane to pixel positions in the plane of the output device. Using the rectangular boundaries for the world-coordinate window and the device viewport, a two-dimensional package maps the world scene to device coordinates and clips the scene against the four boundaries of the viewport. For three-dimensional graphics applications, the situation is a bit more involved, since we now have more choices as to how views are to be generated. 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. And the clipping boundaries now enclose a volume of space, whose shape depends on the type of projection we select.

Viewing Pipeline

The following figure shows general processing steps for modeling and converting a world-coordinate description of a scene to device coordinates. Once the scene has been modeled, world-coordinate positions are converted to viewing coordinates. The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane, which we can think of in analogy with the camera film plane. Next, projection operations are performed to convert the viewing-coordinate description of the scene to coordinate positions on the projection plane, which will then be mapped to the output device. Objects outside the specified viewing limits are clipped from further consideration, and the remaining objects are processed through visible-surface identification and surface-rendering procedures to produce the display within the device viewport.

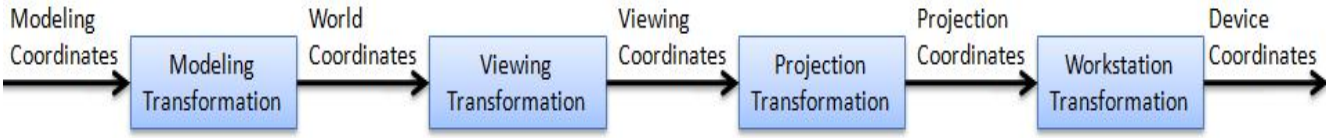


Figure: General three-dimensional transformation pipeline, from modeling coordinates to final device coordinates.

5.8 Projection Methods

Transform points in coordinate system of dimension 'n' into points in a coordinate system of dimension less than 'n'. Projection of 3D object is defined by straight projection rays (projectors) emanating from center of projection, passing thru each point of object and intersecting a projection plane to form projection. These are planar geometric projection as the projection is onto a plane rather than some curved surface and uses straight rather than curved projectors.

- Once the world coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the 3-D objects onto the 2-D view plane
- The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.
- The two types of projections are: Parallel projection and Perspective projection

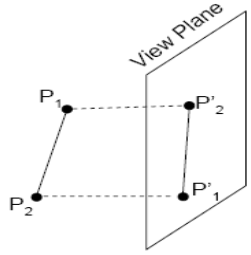


Fig. Parallel Projection of an object to the view plane

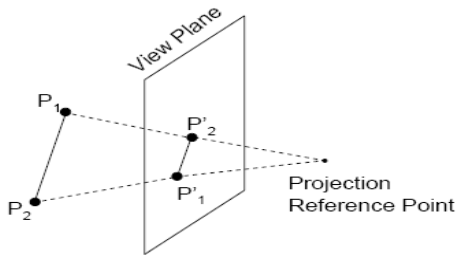


Fig. Perspective Projection of an object to the view plane

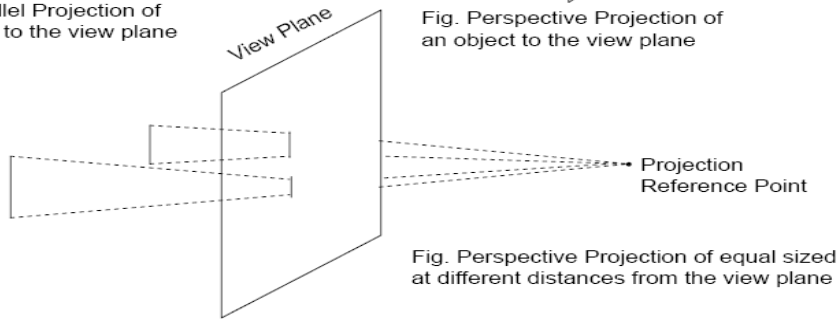
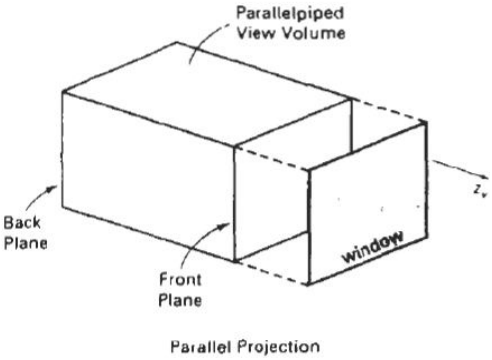


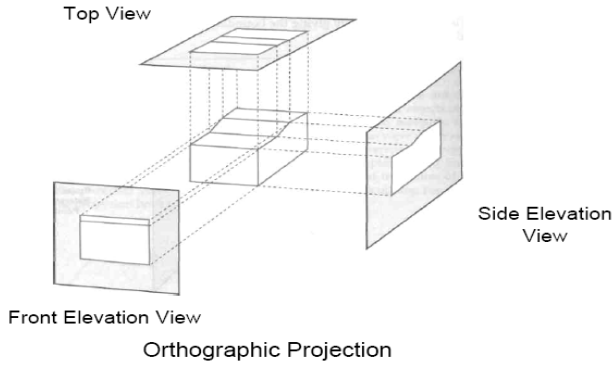
Fig. Perspective Projection of equal sized objects at different distances from the view plane

a) Parallel projection

- The projection is parallel, if the distance is infinite or there is no vanishing point or no converging point of incoming rays from the object.
- The image viewed by the parallel projection is not seemed realistic as they count the exact dimension of the entire surfaces (for realistic view the backend surface must be smaller than that of the front end.



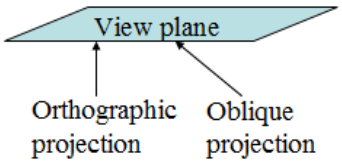
Parallel Projection



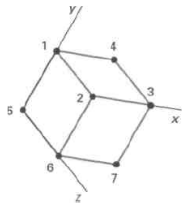
Orthographic Projection

- Coordinate positions are transformed to the view plane along parallel lines.
- Preserves relative proportions of objects so that accurate views of various sides of an object are obtained but doesn't give realistic representation of the 3D object.
- Can be used for exact measurements so parallel lines remain parallel.

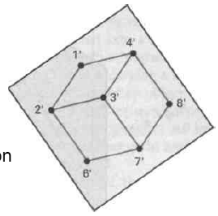
I. Orthographic parallel projection



* When projection is perpendicular to view the plane we have orthographic parallel projection.
* It is used to produce the front, side and top views of an object. Front, side and rear orthographic projections of an object are called ***elevations.***
* Top orthogonal projection is called a ***plane view***.
* This is mainly used in Engineering & architectural drawing.
* Views that display more than one face of an object are called ***axonometric orthographic projections***. Most commonly used axonometric projection is the ***isometric projection***.



Isometric Projection



* *Transformation equations:* If view plane is placed at position z_{vp} along z_v axis then any point (x, y, z) is transformed to projection as: $x_p = x$, $y_p = y$ and z value is preserved for depth information needed (visible surface detection).

II. Oblique parallel projection

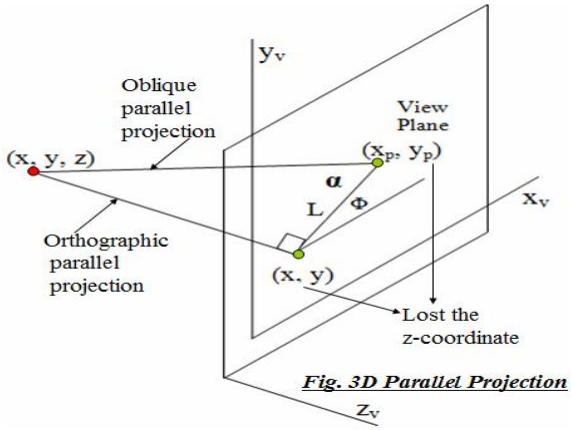


Fig. 3D Parallel Projection

* Obtained by projecting points along parallel lines that are not perpendicular to projection plane.
* Often specified with two angles Φ and α
* Point (x, y, z) is projected to position (x_p, y_p) on the view plane.
* Orthographic projection coordinated on the plane are (x, y).
* Oblique projection line from (x, y, z) to (x_p , y_p) makes an angle with then line on the projection plane that joins (x_p , y_p) and (x , y).
* This line of length L is at an angle Φ with the horizontal direction in the projection plane.
* Expressing projection coordinates in terms of x, y, L and Φ as

$$x_p = x + L\cos\Phi$$
$$y_p = y + L\sin\Phi$$

L depends on the angle α and z coordinate of point to be projected

$$\tan\alpha = z/L$$

Thus,

$$L = z/\tan\alpha = zL_1,$$ where L_1 is the inverse of tanα

So the oblique projection equations are:

$$x_p = x + z(L_1\cos\Phi)$$
$$y_p = y + z(L_1\sin\Phi)$$

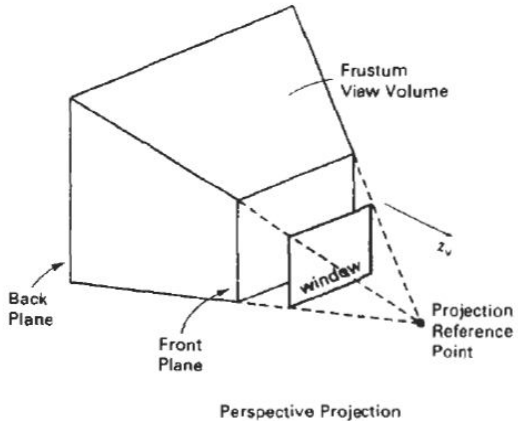
* The transformation matrix for producing any parallel projection onto the x_v y_v-plane can be written as:

$$M_{parallel} = \begin{pmatrix} 1 & 0 & L_1\cos\Phi & 0 \\ 0 & 1 & L_1\sin\Phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

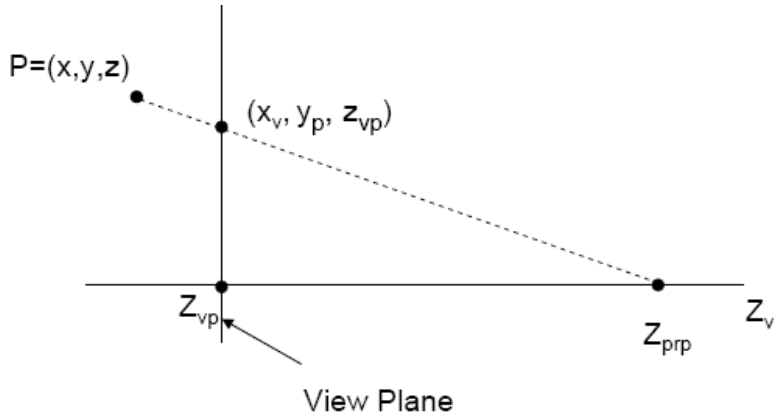
* Orthographic projection is obtained when $L1 = 0$ (occurs at projection angle α of 90 degree)
* Oblique projection is obtained with non-zero values for L_1.

b) Perspective Projection

* The projection is perspective if the distance is finite or has fixed vanishing point or the incoming rays converge at a point. Thus the viewing dimension of object is not exact i.e. seems large or small according as the movement of the view plane from the object.
* Scenes displayed using perspective projections appear more realistic, since this is the way that our eyes and a camera lens form images.
* The perspective projection perform the operation as the scaling (i.e. zoom in & out) but here the object size is only maximize to the size of real present object i.e. only size reduces not enlarge. This operation is possible here only the movement of the view plane towards or far from the object position.



Perspective Projection



View Plane

- To obtain a perspective projection of a 3-D object, we transform points along projection lines that meet at the projection reference point
- Suppose we set the projection reference point at position z_{prp} along the z_v axis, we place the view plane at z_{vp}
- We can write the equations describing coordinate positions along this perspective projection line in a parametric form as

$$x' = x - xu \qquad y' = y - yu \qquad z' = z - (z - z_{prp})\,u$$

- Parameter u takes values 0 to 1, and coordinate position (x', y' ,z') represents any point along the projection line

- When u=0, we are at position P=(x, y, z)
- At the other end of the line, u=1 and we have the projection reference point coordinates (0,0,zprp)
  - On the view plane $z'=z_{vp}$ and we can solve the z' equation for parameter u at this position along the projection line
  - Thus, $z'=z_{vp} = z-(z-z_{prp})\,u$ $\qquad u = \dfrac{z_{vp} - z}{z_{prp} - z}$
  - Substituting this value of u into the equations for x' and y', we obtain the perspective transformation equations:

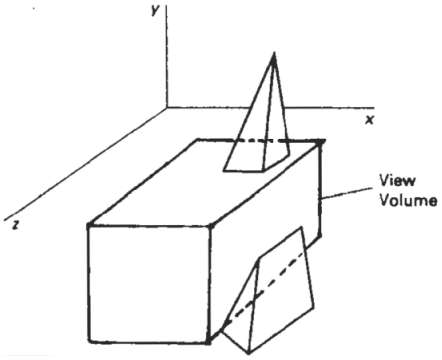
$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = x\left(\frac{d_p}{z_{prp} - z}\right) \qquad \text{(1)}$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = y\left(\frac{d_p}{z_{prp} - z}\right)$$

  - Where $d_p = z_{prp} - z_{vp}$ is the distance of the view plane from the projection reference point.

5.9 Clipping in 3D

Clipping in three dimensions can be accomplished using extensions of two-dimensional clipping methods. Instead of clipping against straight-line window boundaries, we now clip objects against the boundary planes of the view volume. An algorithm for three-dimensional clipping identifies and saves all surface segments within the view volume for display on the output device. All parts of objects that are outside the view volume are discarded. View-volume clipping boundaries are planes whose orientations depend on the type of projection, the projection window, and the position of the projection reference point.



To clip a polygon surface, we can clip the individual polygon edges. To clip a line segment against the view volume, we would need to test the relative position of the line using the view volume's boundary plane equations. An endpoint (x, y, *z)* of a line segment is

- Outside a boundary plane: $Ax + By + Cz + D > 0$, where *A, B,* C, and *D* are the plane parameters for that boundary.
- Inside the boundary if $Ax + By + Cz + D < 0$

Lines with both endpoints outside a boundary plane are discarded, and those with both endpoints inside all boundary planes are saved. The intersection of a line with a boundary is found using the line equations along with the plane equation. Intersection coordinates (x_1, y_1, z_1) are values that are on the line and that satisfy the plane equation $Ax_1 + By_1 + Cz_1 + D = 0$. To clip a polygon surface, we can clip the individual polygon edges.

As in two-dimensional viewing, the projection operations can take place before the view-volume clipping or after clipping. All objects within the view volume map to the interior of the specified projection window. The last step is to transform the window contents to a two-dimensional viewport, which specifies the location of the display on the output device.