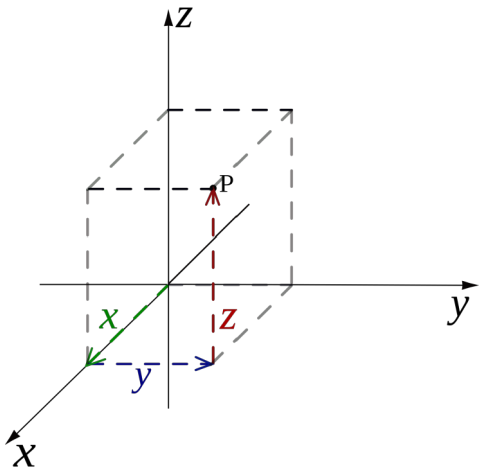
Chapter 5

3D Graphics System

Three-dimensional Space

Three-dimensional space is a geometric 3-parameters model of the physical universe in which all known matter exists. These three dimensions can be labeled by a combination of length, breadth, and depth. Any three directions can be chosen, provided that they do not all lie in the same plane.



3 Dimensional Object

An object that has height, width and depth, like any object in the real world is a 3 dimensional object.

Types of objects: Trees, terrains, clouds, rocks, glass, hair, furniture, human body, flowers, rubber etc

3D Graphics

3D computer graphics or three-dimensional computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images.

2D is "flat", using the horizontal and vertical (X and Y) dimensions, the image has only two dimensions. 3D adds the depth (Z) dimension. This third dimension allows for rotation and visualization from multiple perspectives. It is essentially the difference between a photo and a sculpture.

We can perform different transformation by specifying the three dimensional transformation vector, however the 3d Transformation is more complex than 2D transformation.

Q. 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.

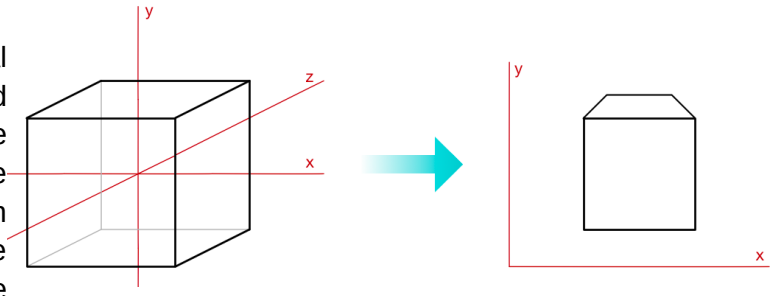
Pseudo-3D and true 3D

Pseudo-3D is a term used to describe either 2D graphical projections and similar techniques used to cause images or scenes to simulate the appearance of being three-dimensional (3D) when in fact they are not.

By contrast, games using 3D computer graphics without such restrictions are said to use true 3D.

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.

3D Geometric Transformation

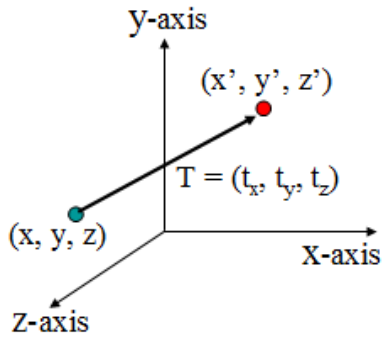
Methods for geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the z coordinate. We now translate an object by specifying a three-dimensional translation vector, which determines how much the object is to be moved in each of the three coordinate directions.

2D transformations can be represented by 3 x 3 matrices using homogeneous 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.

1. **Translation**

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix}$$



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:

$$x' = x + t_x$$

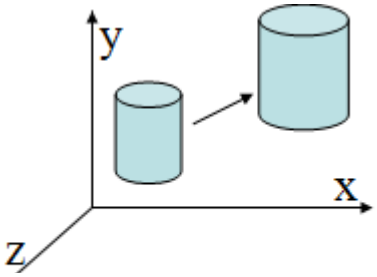
$$y' = y + t_y$$

$$z' = z + t_z$$

2. **Scaling**

Matrix expression for scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as :

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



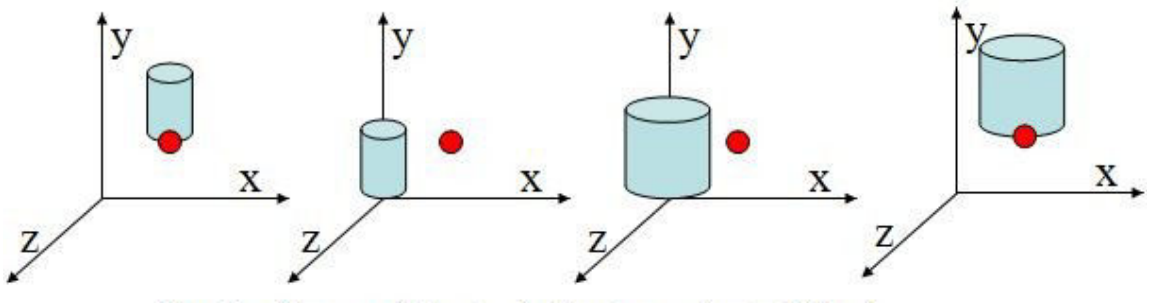
Scaling with respect to a selected fixed point (x_f, y_f, z_f)

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

i.e $\mathbf{T}_{(x, y, z)} \cdot \mathbf{S}_{(x, y, z)} \cdot \mathbf{T}_{(-x, -y, z)}$

$$= \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

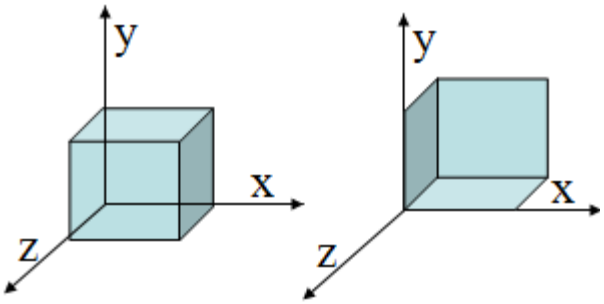


3. **Shearing**

Shearing transformations are used to modify object shapes.

a) **Z axis Shearing**

This transformation alters x and y co-ordinate values by amount that is proportional to the z value while leaving z co-ordinate unchanged.



$x' = x + S_{hx}.Z$

$z' = y + S_{hx}.Z$

$z' = z$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & S_{hx} & 0 \\ 0 & 1 & S_{hy} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Parameters S_{hx} and S_{hy} can be assigned any real values

b) **X axis Shearing**

This transformation alters Y and Z coordinate values by an amount that is proportional to the X value while leaving the X value unchanged i.e.

$x' = x$

$y' = y + S_{hy}x$

$z' = z + s_{hz}X$

c) **Y axis shearing**

This transformation alters Y and Z coordinate values by an amount that is proportional to the X value while leaving the X value unchanged i.e.

$x' = x + S_{hx}.y$

$y' = y$

$z' = z + S_{hy}X$

4. **Reflection**

In 3D-reflection the reflection takes place about a plane. The matrix in case of pure reflections, along basic planes, viz. *X-Y plane, Y-Z plane and Z-X plane* are given below:

a) **X-Y plane**

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

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

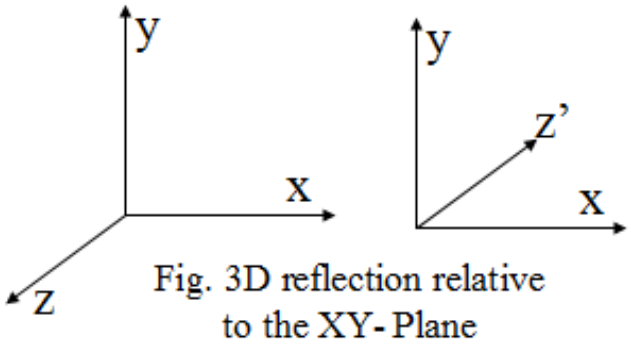


Fig. 3D reflection relative to the XY- Plane

b) **Y-Z plane**

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

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

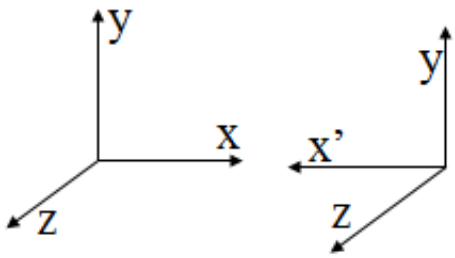


Fig. 3D reflection relative to the YZ- Plane

c) **Z-X plane**

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

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

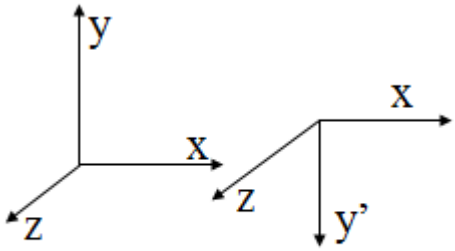


Fig. 3D reflection relative to the ZX- Plane

d) **Reflection about any axis parallel to one of the coordinate axes**

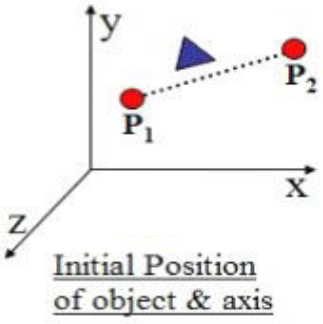
Steps:

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

$$P' = [\, T^{-1} \,.\, R \,.\, T \,] \,.\, P$$

e) **Reflection about an arbitrary axis**

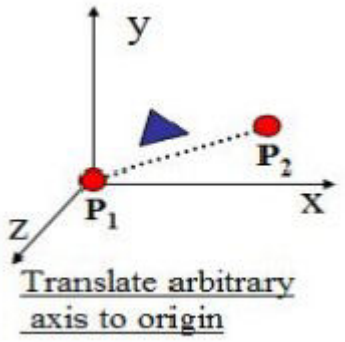
A reflection matrix for any axis that does not coincide with a coordinate axis can be set up as a composite transformation involving combinations of translation, the coordinate-axes rotation and reflection.



Initial Position of object & axis

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

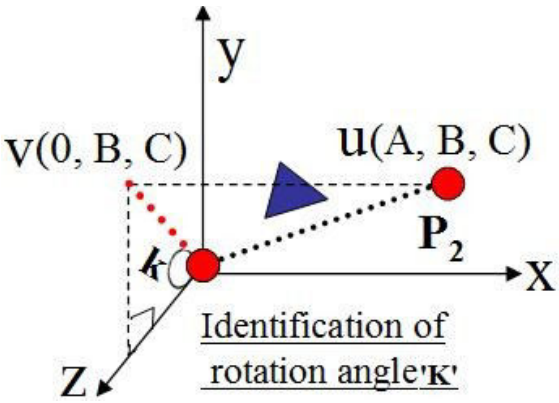
$$T = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

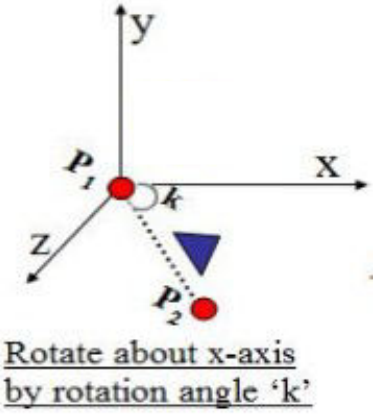


Step-2: Rotate the object so that the axis of rotation coincides with one of the coordinate axes. Usually the z axis is preferred

To coincide the axis of rotation to z axis we have to first perform rotation about x axis to bring it into X-Z plane and then perform rotation about y axis to coincide it with z axis.

a) **Rotation about x-axis by angle 'k' in clockwise direction so that the arbitrary axis lies on X-Z-plane**





Here,

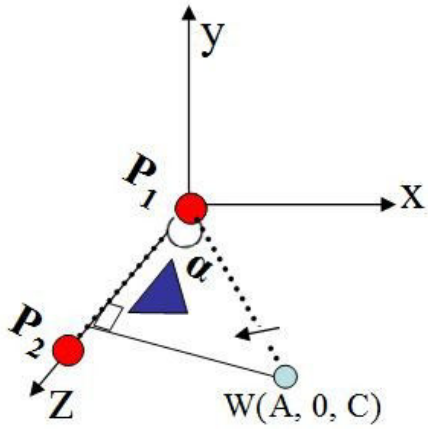
Sin k = B / V Where, $V=\sqrt{(B^2 + C^2)}$

Cos k = C / V

Now, the translation matrix is given by:

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

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



Here,

Sin α= A / W &

Cos α= C / W

Now, the translation matrix is given by:

$$R_y = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} C/W & 0 & -A/W & 0 \\ 0 & 1 & 0 & 0 \\ A/W & 0 & C/W & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

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

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

Step-5: Perform inverse translation

Hence,

Composite matrix = $\mathbf{T' R_x 'R_y ' RF_z R_y R_x T}$

f) Reflection about any arbitrary plane in 3D Space

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

Step 1: Translate the reflection plane to the origin of the coordinate system

Step 2: Perform appropriate rotations to make the normal vector of the reflection plane at the origin until it coincides with the z-axis.

Step 3: After that reflect the object through the z= 0 coordinate plane.

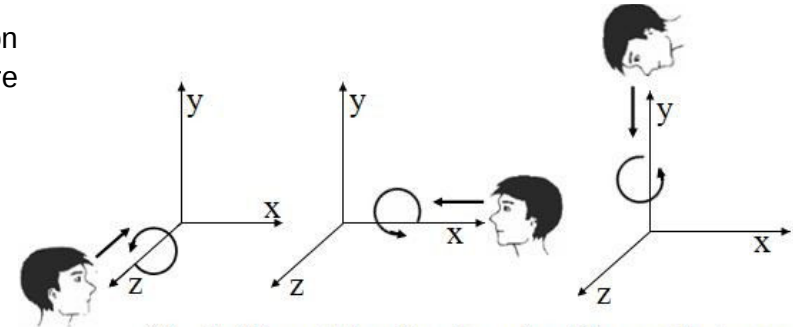
Step 4: Perform the inverse of the rotation transformation

Step 5: Perform the inverse of the translation

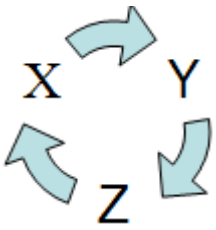
5. **Rotation**

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



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



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

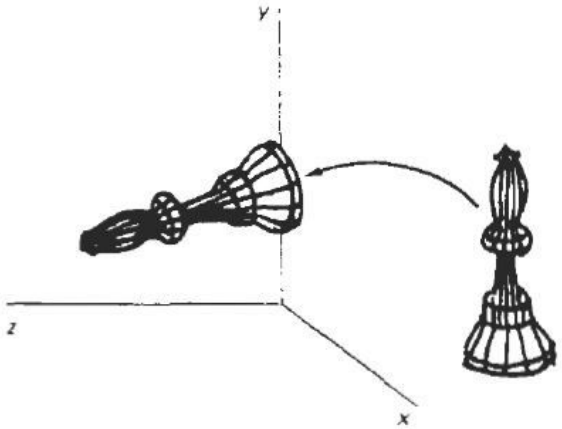
a) **3D x-axis rotation ($P' = R_{x(\theta)} \cdot P$)**

$x' = x$

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

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

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



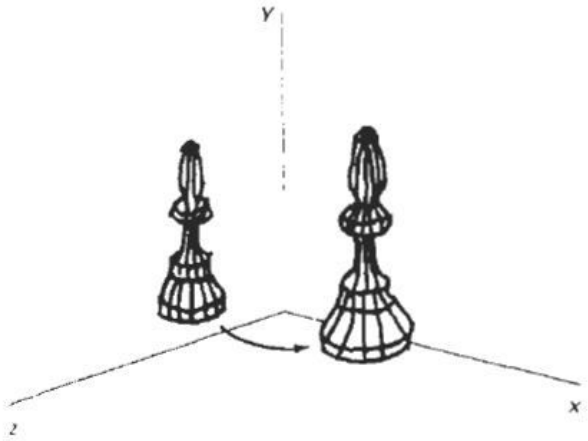
b) **3D y-axis rotation ($P' = R_{y(\theta)} \cdot P$)**

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

$y' = y$

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

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



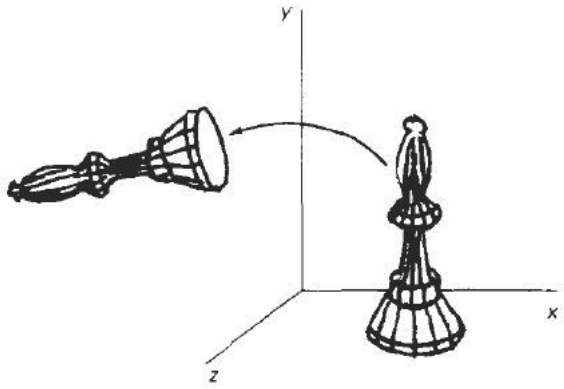
c) **3D z-axis rotation ($P' = R_{z(\theta)} . P$)**

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

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

$z' = z$

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

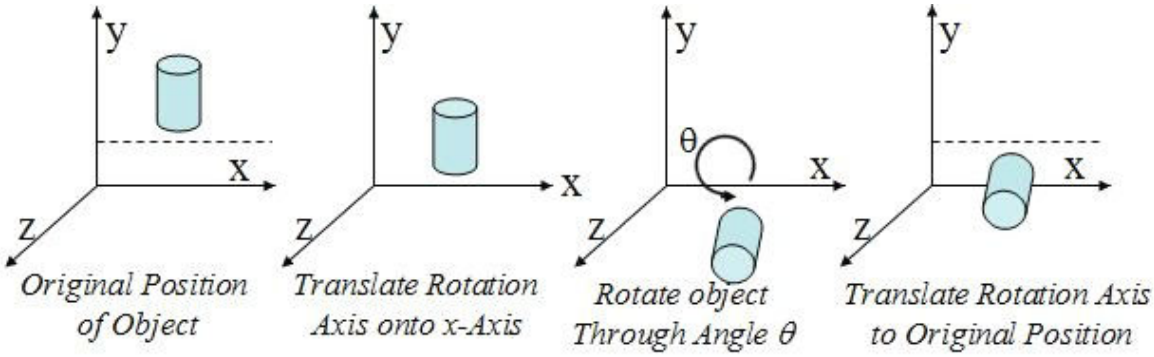


d) **Rotation about an axis parallel to one of the coordinate axes**

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

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



e) **Rotation about an arbitrary axis**

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 translation and the coordinate-axes rotations.

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

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

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

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

Step-5: Perform inverse translation

Hence,

Composite matrix = $\mathbf{T' R_x 'R_y ' R_z R_y R_x T}$

f) **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.

Step 1: Translate the rotation plane to the origin of the coordinate system

Step 2: Perform appropriate rotations to make the normal vector of the rotation plane at the origin until it coincides with the z-axis.

Step 3: After that rotate the object through the z= 0 coordinate plane.

Step 4: Perform the inverse of the rotation transformation

Step 5: Perform the inverse of the translation

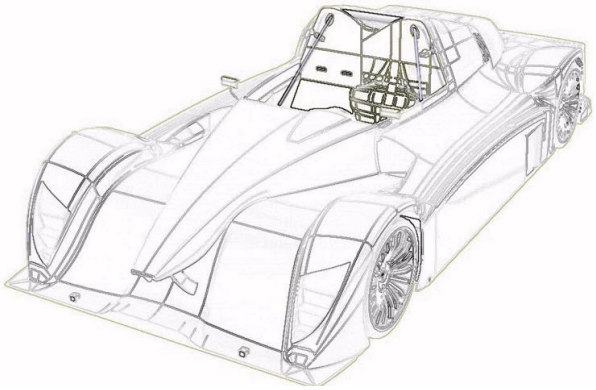
3D Representation

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)**: Often abbreviated as **B-rep** or **BREP,** boundary representation is a method for representing shapes (a set of surfaces that separate the object interior from the environment) using the limits.

  A solid is represented as a collection of connected surface elements, the boundary between solid and non-solid.

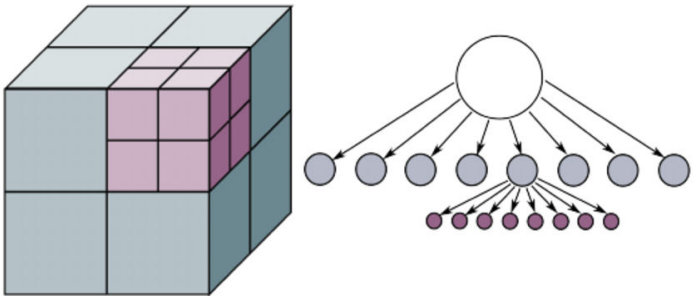
  

  For examples: polygon surfaces and spline patches.

  Boundary representation of models are composed of two parts: topology and geometry (surfaces, curves and points). The main topological items are: *faces*, *edges* and *vertices*. A face is a bounded portion of a surface; an edge is a bounded piece of a curve and a vertex lies at a point.

  Other elements are the *shell* (a set of connected faces), the *loop* (a circuit of edges bounding a face) and *loop-edge links* (also known as *winged edge links* or *half-edges*) which are used to create the edge circuits. The edges are like the edges of a table, bounding a surface portion.

- **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).

  Space-partitioning systems are often hierarchical, meaning that a space (or a region of space) is divided into several regions, and then the same space-partitioning system is recursively applied to each of the regions thus created. The regions can be organized into a tree, called a space-partitioning tree.

  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.

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 wire frame 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.

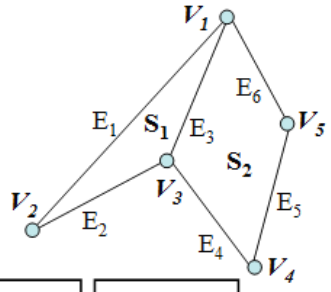
A. Polygon Tables

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.**



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

- 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.

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.

B. Plane Equations

Plane equation method is the method for representing polygon surface for 3D object. The information about spatial orientation of object is described by its individual surface which is obtained by vertex coordinate values & equation of each plane gives a description of each surface. The equation for plane surface can be expressed in the form of:

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

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

Let (x_1,y_1,z_1), (x_2,y_2,z_2) , (x_3,y_3,z_3) are three such points on points on the plane. Then

$$Ax_1 + By_1 + Cz_1 + D = 0 \ldots\ldots\ldots\ldots (1)$$

$$Ax_2 + By_2 + Cz_2 + D = 0 \ldots\ldots\ldots\ldots (2)$$

$$Ax_3 + By_3 + Cz_3 + D = 0 \ldots\ldots\ldots\ldots (3)$$

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

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

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

For any point (x, y, z)

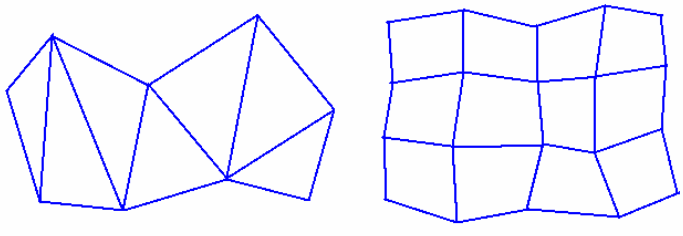
if $Ax + By + Cz + D \neq 0$ then (x, y, z) is not on the plane

if $Ax + By + Cz + D < 0$, the point (x, y, z) is inside plane i.e. invisible side

if $Ax + By + Cz + D > 0$ the point lies outside the surface.

C. 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.

Cubic Spline and Bezier Curve

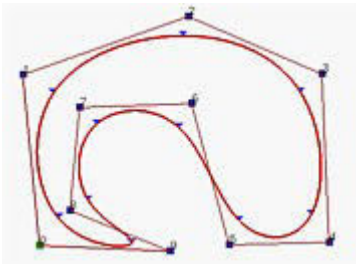
Splines are the smooth curves passing through a set of given points. By varying the number and position of the lead weights, the shape of the spline is changed so that it passes through some designated set of points.

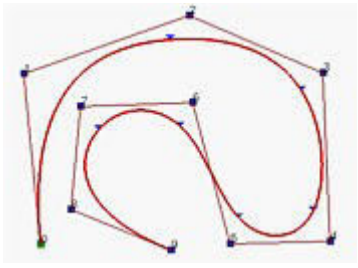
In computer graphics, continuous curve that are formed with polynomial section with certain boundary conditions are called spline curve.

Splines are of two types

a. Closed Splines: the generated curve will touch the first and last legs of the control

b. Open Splines : the generated curve will not touch the first and last legs of the control





A cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set *m* of control points.

Splines are used:

- To design curve and surface shapes in graphics applications,
- To digitize drawings for computer storage
- 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.

A. 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.

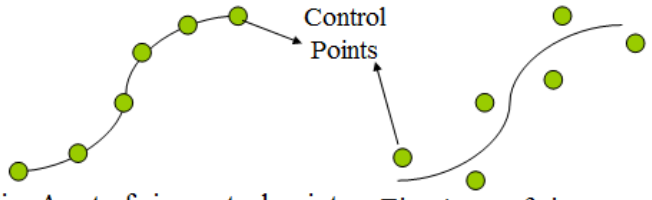


Fig. A set of six control points ***interpolated*** with piecewise continuous polynomial sections

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

B. Convex hulls

The convex polygon boundary that encloses the set of control points is called convex hull. Convex hulls provide the measure for deviation of a curve or surface from a region bounding the control points.

Some splines are bounded by convex hull, thus ensuring that polynomials smoothly follow control points

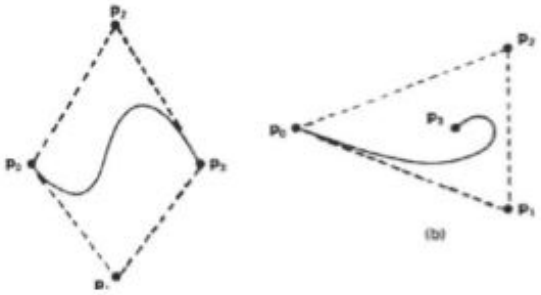
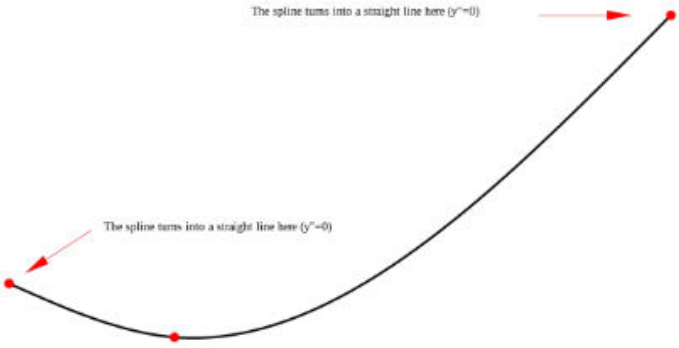


Fig. Convex hull shapes (dashed lines) for two sets of control points.

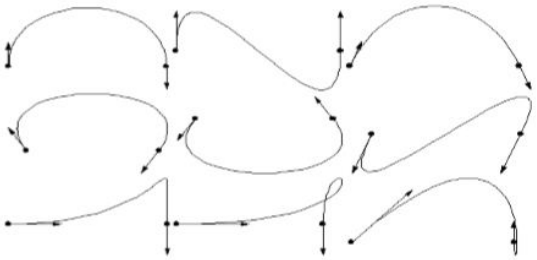
C. Natural Cubic Splines

One of first spline curves to be developed for graphics applications is a natural cubic spline. This interpolation curve is the mathematical representation of a original drafting spline. We formulate natural cubic spline by requiring that 2 adjacent curve sections have same first & 2nd parametric derivatives at their common boundary. Thus, natural cubic splines have C-2 continuity. Although natural cubic splines are the mathematical model for drafting spline, they have major disadvantage. If a position of any one control point is altered, then the entire curve is affected. Thus, natural cubic splines allow for no "local control", so that we cannot restructure part of curve without specifying an entirely new set of control points.



D. Hermite Curves

Hermite form is defined by 2 endpoints & 2 tangent vectors at these endpoints. Hermite spline is an interpolating piece-wise cubic polynomial with specified tangent at each control point. Unlike natural cubic splines, Hermite splines can be adjusted locally because each curve section is only dependent on endpoint constraints.

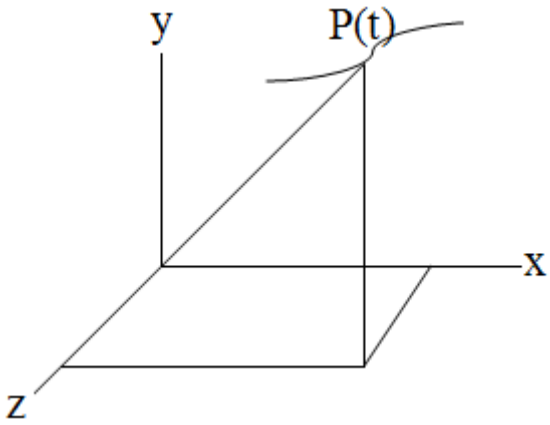


E. Parametric Cubic Curve

A parametric cubic curve is defined as

$$P(t) = \sum_{i=0}^{3} a_i t^i \quad 0 <= t <= 1 \text{ ------ (i)}$$

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



Expanding equation (i) yield

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

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

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

$$y(t) = a_{3y}t3 + a_{2y}t^2 + a_{1y}t + a_{0y}$$

$$z(t) = a_{3z}t3 + a_{2z}t^2 + a_{1z}t + a_{0z} \text{ ----------(iii)}$$

F. Bezier Curve

A Bezier curve is a mathematically defined curve used in two-dimensional graphic applications. The curve is defined by four points: the initial position and the terminating position (which are called "anchors") and two separate middle points (which are called "handles"). The shape of a Bezier curve can be altered by moving the handles. The mathematical method for drawing curves was created by Pierre Bézier in the late 1960's for the manufacturing of automobiles**.**

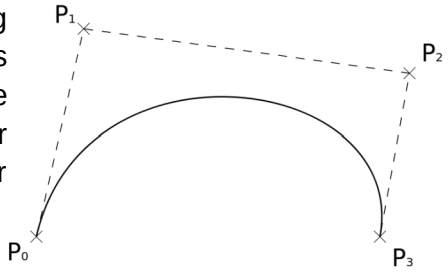


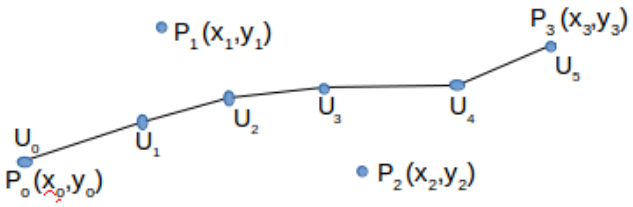
Fig: Bezier curve with four control points

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:

a) It always passes through the first and last control points.

b) It lies within the convex hull (convex polynomial boundary) of the control points.



Let, P be the control points and N_{seg} is number of segments in a curve segment.

i.e $u = \frac{i}{N_{seg}}$; where $0 \le u \le 1$ and $i = 0$ to N_{seg}

we have

$$x(u) = \sum_{j=0}^{n} x_j BeZ_{(j,n)}(u) \quad \text{where n= no. of control points.}$$

$$= x_0BeZ_{0,3}(u) + x_1BeZ1_{,3}(u) + x_2BeZ_{2,3}(u) + x_3BeZ_{3,3}(u)$$

$$y(u) = \sum_{j=0}^{n} y_j BeZ_{(j,n)}(u)$$

$= y_0BeZ_{0,3}(u) + y_1BeZ1_{,3}(u) + y_2BeZ_{2,3}(u) + y_3BeZ_{3,3}(u)$

The blending function $BeZ_{j,n}(u)$ is defined as a Bernstein polynomial

$$BeZ_{j,n}(u) = \frac{n!}{j!(n-j)!} u^j (1-u)^{n-j}$$

Also we have

$$Q(u) = P_0 BeZ_{0,3}(u) + P_1 BeZ_{1,3}(u) + P_2BeZ_{2,3}(u) + P_3 BeZ_{3,3}(u)$$

Calculating

$$BeZ_{0,3}(u) = \frac{3!}{0!(3-0)!} u^0 (1-u)^{3-0} = (1-u)^3$$

$$BeZ_{1,3}(u) = \frac{3!}{1!(3-1)!} u^1 (1-u)^{3-1} = 3u(1-u)^2$$

$$BeZ_{2,3}(u) = \frac{3!}{2!(3-2)!} u^2 (1-u)^{3-2} = 3u^2(1-u)$$

$$BeZ_{3,3}(u) = \frac{3!}{3!(3-3)!} u^3 (1-u)^{3-3} = u^3$$

Substituting the values, we get

$$Q(u) = P_0 (1-u)^3 + P_1 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3$$

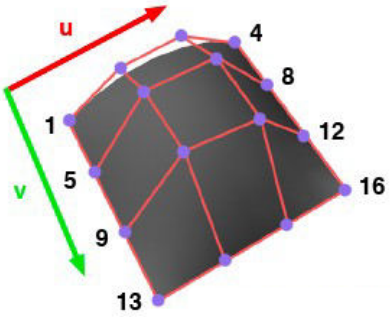
Q. Calculate the co-ordinates of Beizer curve described by 4 control points $P_0(0,0)$, $P_1(1,2)$, $P_2(3,3)$, $P_3(4,0)$ and approximate with 5 lines segments.

G. B-Spline

While drawing the bezier curve, each section of the curve is connected to the next section at a sample point but the slopes of these two sections need not match at this point. This means that there can be corners at the sample points and that we may not have a complete smooth curve. To solve this problems, we force the curve to pass through the sample point but rather pull it to the neighborhood of the sample point. A set of blending functions which take this approach are called **B-Splines**

H. Bezier Non-planar Surfaces

Like Bezier curve, Bézier surface is defined by a set of control points. Similar to interpolation in many respects, a key difference is that the surface does not pass through the central control points; rather, it is "stretched" toward them as though each were an attractive force.



A two-dimensional Bézier surface can be defined as a parametric surface where the position of a point p as a function of the parametric coordinates *u*, *v* is given by:

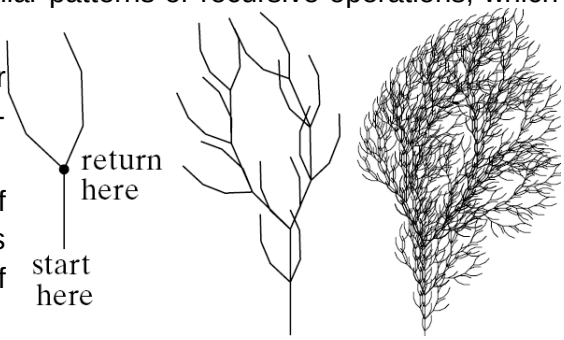
$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, \mathbf{k}_{i,j}$$

evaluated over the unit square, where

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

Fractal Geometry Method

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".



Natural objects, such as mountains and clouds, have irregular or fragmented features, and these are described with fractal-geometry methods.

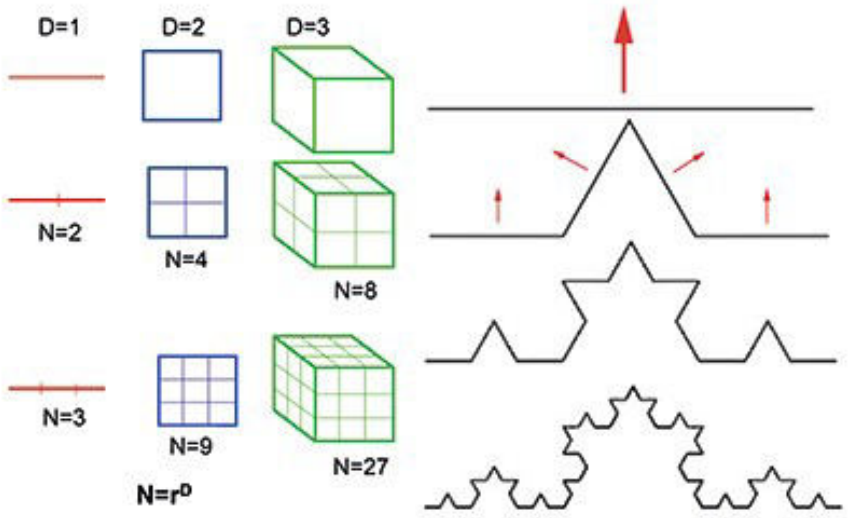
Fractional object describes and explains the features of natural phenomena with procedures or function in various dimensions that theoretically repeat an infinite number of times but finite number of steps for graphics display.

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:

$$P_1 = F(P_0),$$

$$P_2 = F(P_2), \ldots$$

We can describe the amount of variation in the object detail with a number called the fractal dimension.



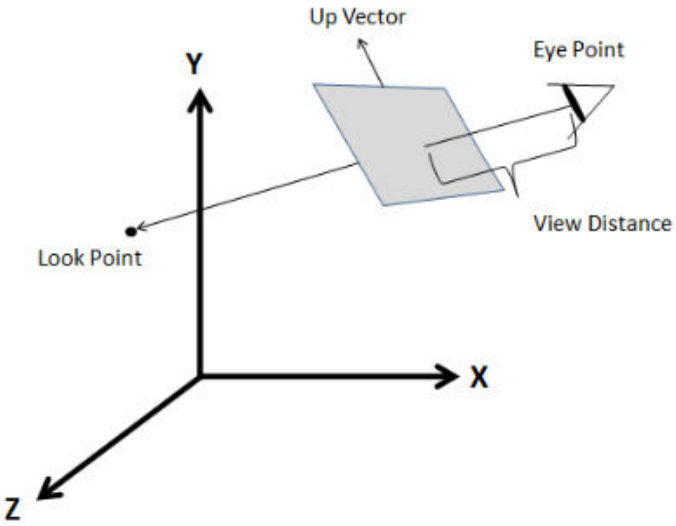
3D Viewing Transformation

The basic idea of the 3D viewing transformation is similar to the 2D viewing transformation. That is, a viewing window is defined in world space that specifies how the viewer is viewing the scene. A corresponding view port is defined in screen space, and a mapping is defined to transform points from world space to screen space based on these specifications. The view port portion of the transformation is the same as the 2D case.

Specification of the window, however, requires additional information and results in a more complex mapping to be defined. Defining a viewing window in world space coordinates is exactly like it sounds; sufficient information needs to be provided to define a rectangular window at some location and orientation. The usual viewing parameters that are specified are:

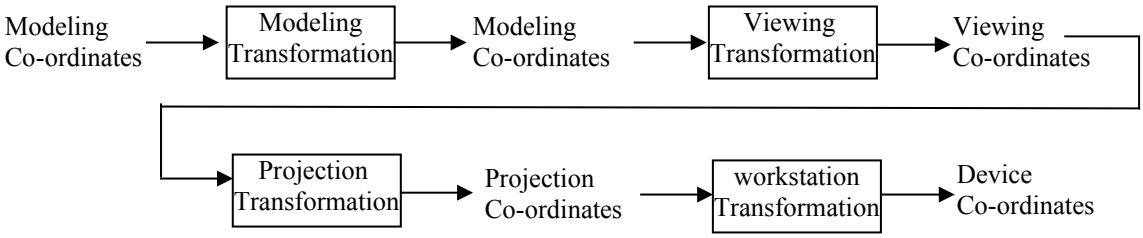
Eye Point	The position of the viewer in world space
Look Point	The point that the eye is looking at
View Distance	The distance that the window is from the eye
Window Size	The height and width of the window in world space coordinates Up Vector which direction represents “up” to the viewer, this

	parameter is sometimes specified as an angle of rotation about the viewing axis



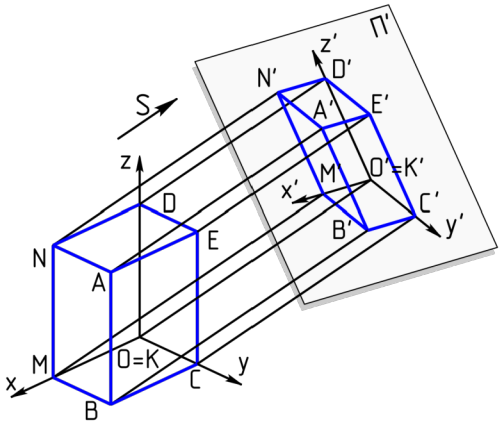
3D Viewing Pipeline

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 view port.



Projection

3D projection is any method of mapping three-dimensional points to a two-dimensional plane. In general, a projection transforms a N-dimension points to N-1 dimensions. Ideally, an an object is projected by projecting each of its endpoints. But an object has infinite number of points. So we cannot project all those points. What we do is that we project only the corner points of an object on a 2D plane and we will join these projected points by a straight line in a 2D plane.



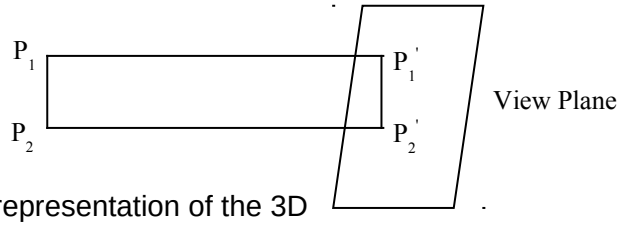
Basic points in projection:

- **Center of projection**: Point from where projection is taken.
- **Projection plane**: the plane on which the projection of the object is formed.
- **Projectors**: Lines emerging from the center of projection and hitting the projection plane
- **Parallel projection**: In this projection, the co-ordinate positions are transformed to the view plane along parallel lines.
- **Perspective Projection**: In this projection, the co-ordinate positions are transformed to the view plane along lines that converges to a point called as center of projection.

The two types of projections are:

a) Parallel projection

- In this projection, the co-ordinate positions are transformed to the view plane along parallel lines.
- 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.



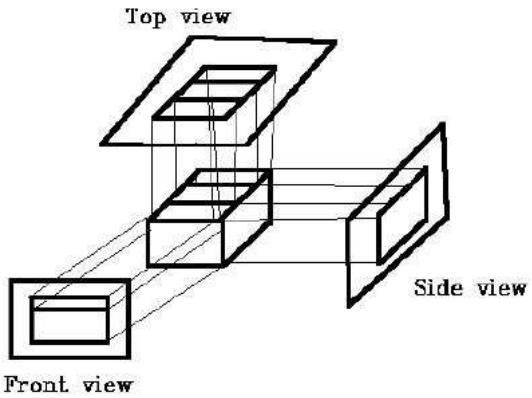
- Can be used for exact measurements so parallel lines remain parallel
- There are two different types of parallel projections

a. Orthographic parallel projection

In orthographic projection, the center of projection lies at infinity and the projections are perpendicular to the view plane. So, a true size and shape of a single face of object is obtained.

In orthographic projection the direction of projection is normal to the projection of the plane. There are three types of orthographic projections −

- Front Projection
- Top Projection
- Side Projection



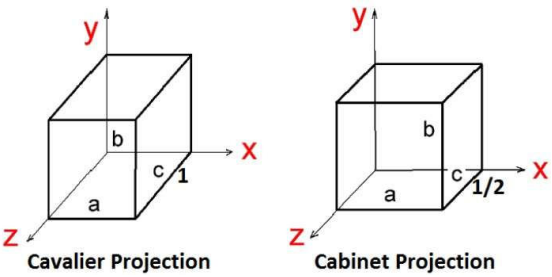
b. Oblique parallel Projection

A projection in which the angle between the projectors and the plane of plane of projection is not equal to 90^0 is called oblique projection.

An oblique projection is formed by parallel projections from a center of projection at infinity that intersects the plane of projection at an oblique angle.

There are two types of oblique projections − **Cavalier** and **Cabinet**.

The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.



The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length.

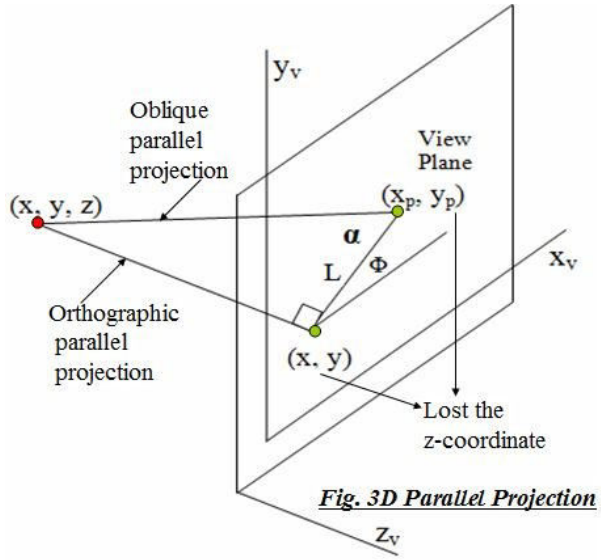


Fig. 3D Parallel Projection

- 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 the 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,

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

$$= z L_1, \text{ where } L_1 \text{ is the inverse of } \tan\alpha$$

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:

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

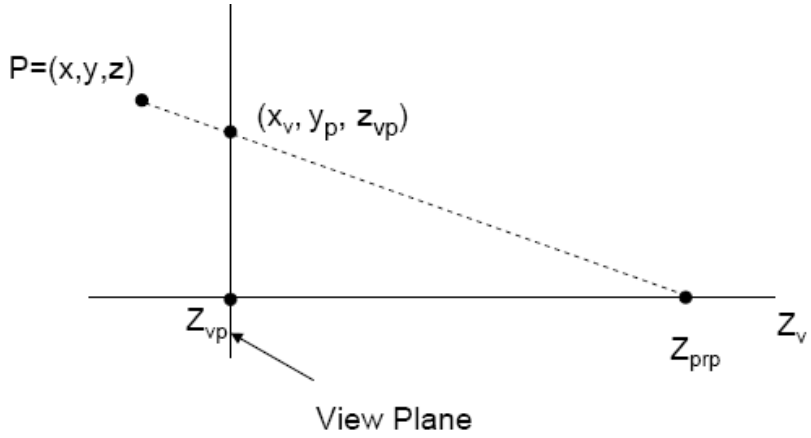
Orthographic projection is obtained when $L_1 = 0$ (occurs at projection angle α of 90 degree) Oblique projection is obtained with non-zero values for L_1.

b) Perspective Projection

In this projection, the co-ordinate positions are transformed to the view plane along lines that converges to a point called as center of 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.

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.



- 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

$$x' = x - x_u$$

$$y' = y - y_u$$

$$z' = z - (z - z_{prp})\, u$$

- 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 $(0,0,z_{prp})$

- On the view plane $z'=z_{vp}$ and we can solve the z' equation for parameter u at this position along the projection line

$$\text{Thus, } z' = z_{vp} = z - (z - z_{prp})\, u$$

$$\mathbf{u} = \frac{\mathbf{z_{vp}} - \mathbf{z}}{\mathbf{z_{prp}} - \mathbf{z}}$$

- Substituting this value of u into the equations for x' and y', we obtain the perspective transformation equations:

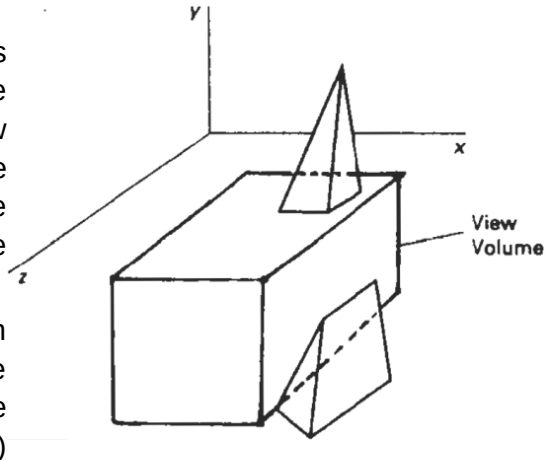
$$xp = x\left(\frac{Z_p}{z}\right)$$

$$yp = y\left(\frac{Z_p}{z}\right)$$

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

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. *(The view volume defines the three-dimensional volume in space that, once projected, is to fit within the view port. There are two parts to the view volume: the view plane rectangle and the near and far clipping planes)*



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 view port, which specifies the location of the display on the output device.

Z-clipping

Z-clipping, or depth clipping, refers to techniques that selectively render certain scene objects based on their depth relative to the screen. Most graphics toolkits allow the programmer to specify a "near" and "far" clip depth, and only portions of objects between those two planes are displayed.

Importance of clipping in video games

- Maximize the game's frame rate and visual quality
- Speed up the rendering of the current scene
- Economizing the use of renderer time and memory within the hardware's capability