FACULTY NAME: AKANKSHA YADAV

COMPUTER GRAPHICS NOTES

MCA/2ND YEAR

UNIT 3:Three Dimensional Graphics

The three-dimensional transformations are extensions of two-dimensional transformation. In 2D two coordinates are used, i.e., x and y whereas in 3D three co-ordinates x, y, and z are used.

For three dimensional images and objects, three-dimensional transformations are needed. These are translations, scaling, and rotation. These are also called as basic transformations are represented using matrix. More complex transformations are handled using matrix in 3D.

The 2D can show two-dimensional objects. Like the Bar chart, pie chart, graphs. But some more natural objects can be represented using 3D. Using 3D, we can see different shapes of the object in different sections.

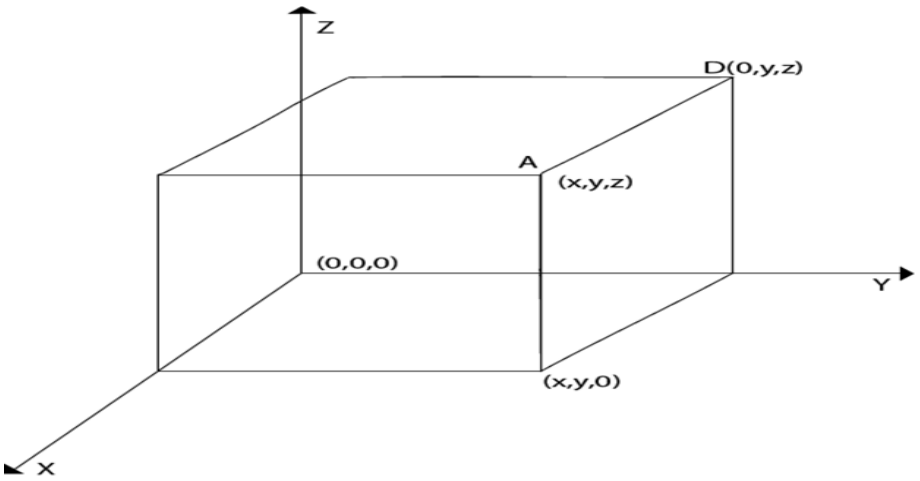
In 3D when a translation is done we need three factors for rotation also, it is a component of three rotations. Each can be performed along any three Cartesian axis. In 3D also we can represent a sequence of transformations as a single matrix.

Computer Graphics uses CAD. CAD allows manipulation of machine components which are 3 Dimensional. It also provides automobile bodies, aircraft parts study. All these activities require realism. For realism 3D is required. In the production of a realistic 3D scene from 2D is tough. It require three dimension, i.e., depth.

3D Geometry

Three dimension system has three axis x, y, z. The orientation of a 3D coordinate system is of two types. Right-handed system and left-handed system.

In the right -handed system thumb of right- hand points to positive z-direction and left- hand system thumb point to negative two directions. Following figure show right-hand orientation of the cube.



Using right-handed system co-ordinates of corners A, B, C, D of the cube

Point A x, y, z

Point B x, y, 0

Point C 0, y, 0

Point D 0, y, z

Producing realism in 3D: The three-dimensional objects are made using computer graphics. The technique used for two Dimensional displays of three Dimensional objects is called projection. Several types of projection are available, i.e.,

Parallel Projection

Perspective Projection

Orthographic Projection

1. Parallel Projection: In this projection point on the screen is identified within a point in the three-dimensional object by a line perpendicular to the display screen. The architect Drawing, i.e., plan, front view, side view, elevation are nothing but lines of parallel projections.

2. Perspective Projection: This projection has a property that it provides idea about depth. Farther the object from the viewer, smaller it will appear. All lines in perspective projection converge at a center point called as the center of projection.

3. Orthographic Projection: It is simplest kind of projection. In this, we take a top, bottom, side view of the object by extracting parallel lines from the object.

Three Dimensional Models

The techniques for generating different images of a solid object depend upon the type of object. Two viewing techniques are available for viewing three-dimensional objects.

Geometry: It is concerned with measurements. Measurement is the location of a point concerning origin or dimension of an object.

Topological Information: It is used for the structure of a solid object. It is mainly concerned with the formation of polygons with the help of points of objects or the creation of the object with polygons.

Three Dimensional Transformations

The geometric transformations play a vital role in generating images of three Dimensional objects with the help of these transformations. The location of objects relative to others can be easily expressed. Sometimes viewpoint changes rapidly, or sometimes objects move in relation to each other. For this number of transformation can be carried out repeatedly.

Translation

It is the movement of an object from one position to another position. Translation is done using translation vectors. There are three vectors in 3D instead of two. These vectors are in x, y, and z directions. Translation in the x-direction is represented using Tx. The translation is y-direction is represented using Ty. The translation in the z-direction is represented using Tz.

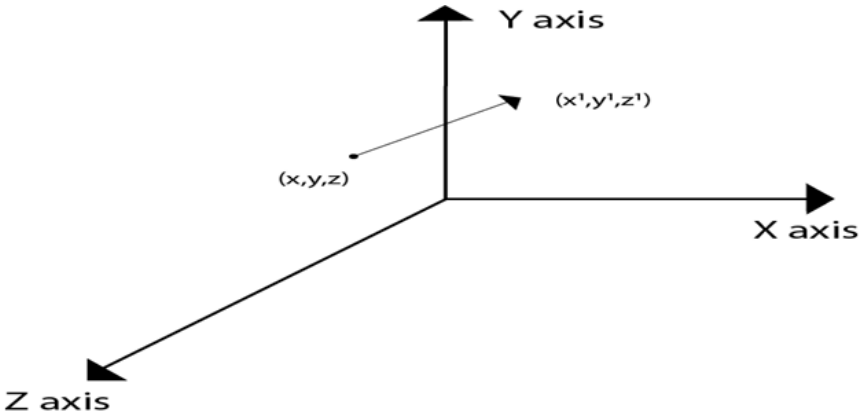
If P is a point having co-ordinates in three directions (x, y, z) is translated, then after translation its coordinates will be (x1 y1 z1) after translation. Tx Ty Tz are translation vectors in x, y, and z directions respectively

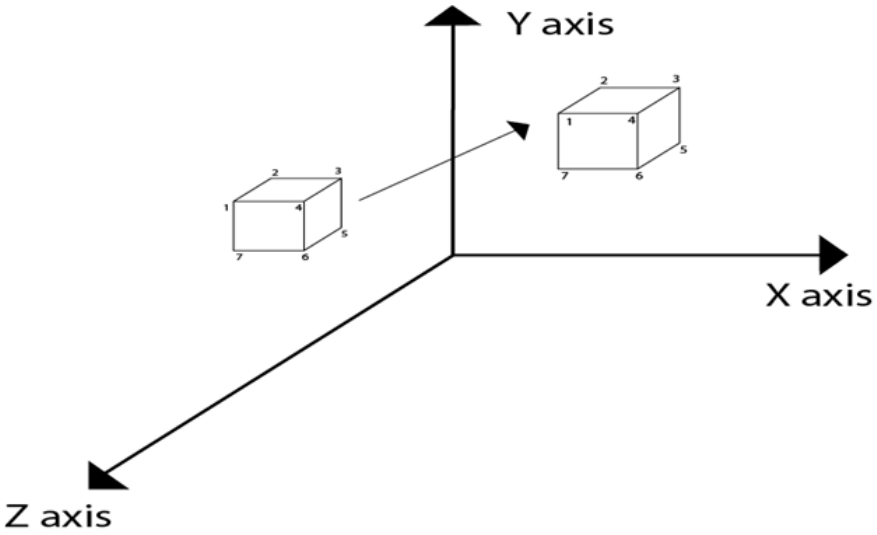
x1=x+ Tx

y1=y+Ty

z1=z+ Tz

Three-dimensional transformations are performed by transforming each vertex of the object. If an object has five corners, then the translation will be accomplished by translating all five points to new locations. Following figure 1 shows the translation of point figure 2 shows the translation of the cube.





Matrix representation of point translation

Point shown in fig is (x, y, z). It become (x1,y1,z1) after translation. Tx Ty Tz are translation vector.

$$\begin{pmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Example: A point has coordinates in the x, y, z direction i.e., (5, 6, 7). The translation is done in the x-direction by 3 coordinate and y direction. Three coordinates and in the z- direction by two coordinates. Shift the object. Find coordinates of the new position.

Solution: Co-ordinate of the point are (5, 6, 7)

Translation vector in x direction = 3

Translation vector in y direction = 3

Translation vector in z direction = 2

Translation matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix}$$

Multiply co-ordinates of point with translation matrix

$$(x^1 y^1 z^1) = (5,6,7,1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 3 & 2 & 1 \end{pmatrix}$$

= [5+0+0+30+6+0+30+0+7+20+0+0+1] = [8991]

x becomes x^1=8
y becomes y^1=9
z becomes z^1=9

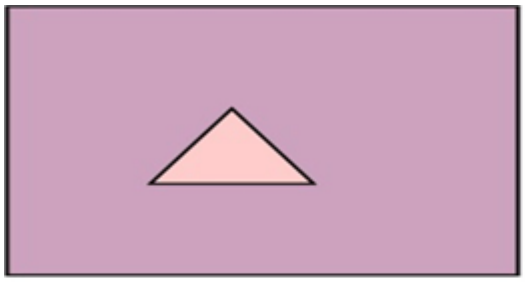
Scaling

Scaling is used to change the size of an object. The size can be increased or decreased. The scaling three factors are required Sx Sy and Sz.

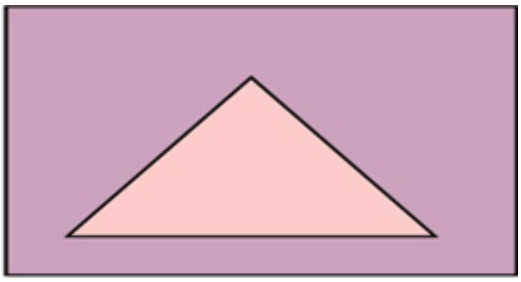
Sx=Scaling factor in x- direction

Sy=Scaling factor in y-direction

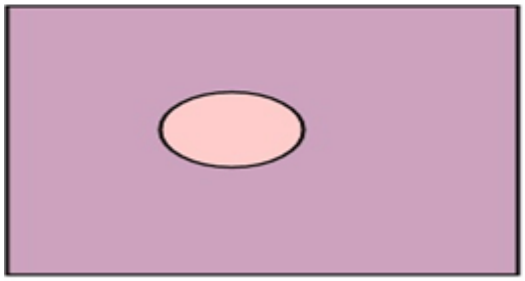
Sz=Scaling factor in z-direction



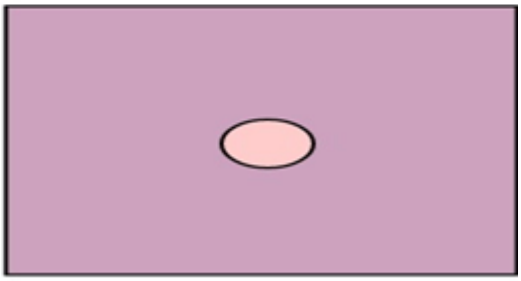
Original
(a)



Enlarged
(b)



Original
(a)



Reduced
(b)

Matrix for Scaling

$$\begin{Bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

Scaling of the object relative to a fixed point

Following are steps performed when scaling of objects with fixed point (a, b, c). It can be represented as below:

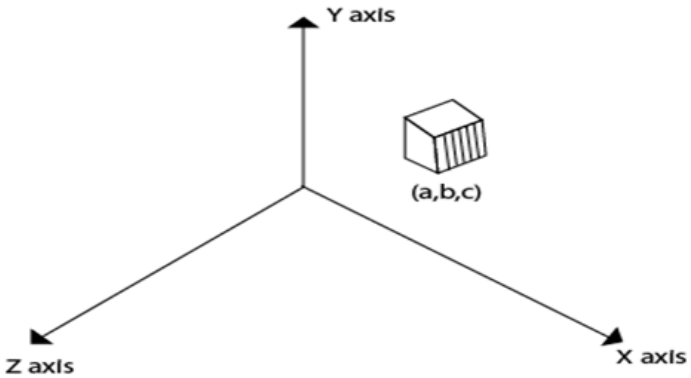
1.Translate fixed point to the origin

2.Scale the object relative to the origin

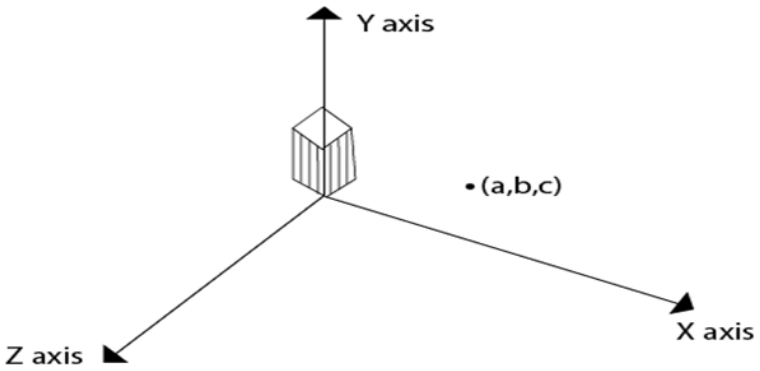
3.Translate object back to its original position.

Note: If all scaling factors Sx=Sy=Sz.Then scaling is called as uniform. If scaling is done with different scaling vectors, it is called a differential scaling.

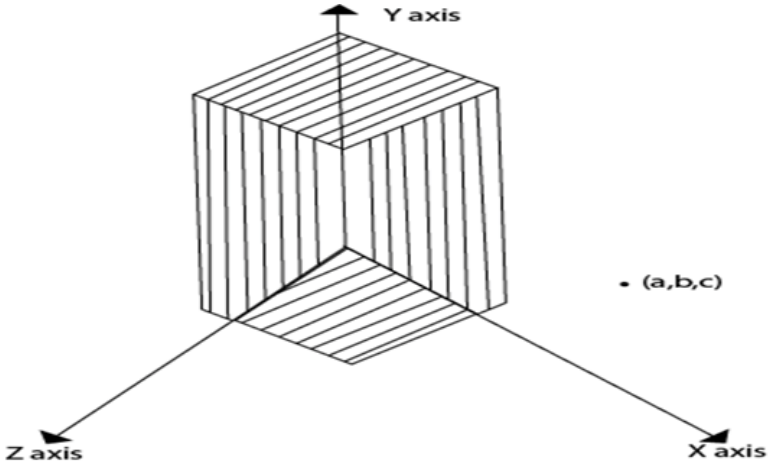
In figure (a) point (a, b, c) is shown, and object whose scaling is to done also shown in steps in fig (b), fig (c) and fig (d*).*



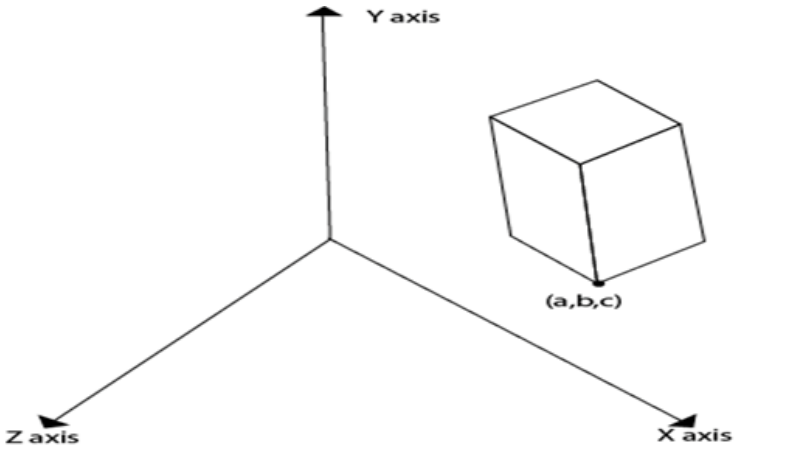
Object & point for scaling
(a)



Object transfer at origin
(b)



Scaling of object
(c)

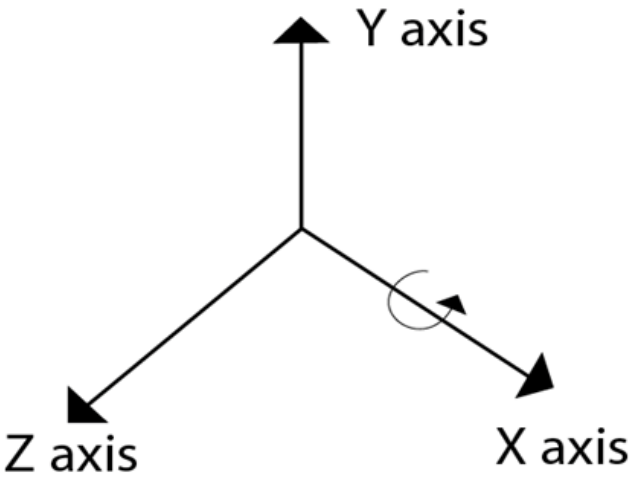


Object shifted to original position after scaling
(d)

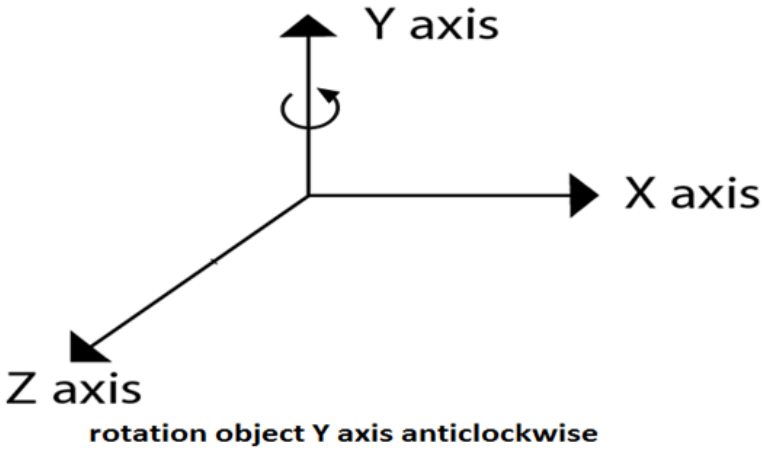
Rotation

It is moving of an object about an angle. Movement can be anticlockwise or clockwise. 3D rotation is complex as compared to the 2D rotation. For 2D we describe the angle of rotation, but for a 3D angle of rotation and axis of rotation are required. The axis can be either x or y or z.

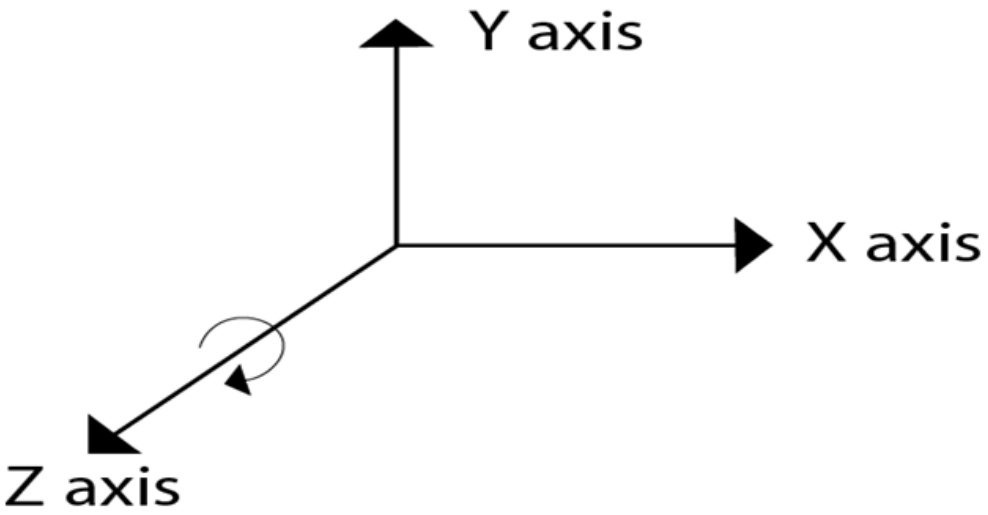
Following figures shows rotation about x, y, z- axis



rotation object X axis anticlockwise



rotation object Y axis anticlockwise



rotation about Z axis clockwise

Rotation about Arbitrary Axis

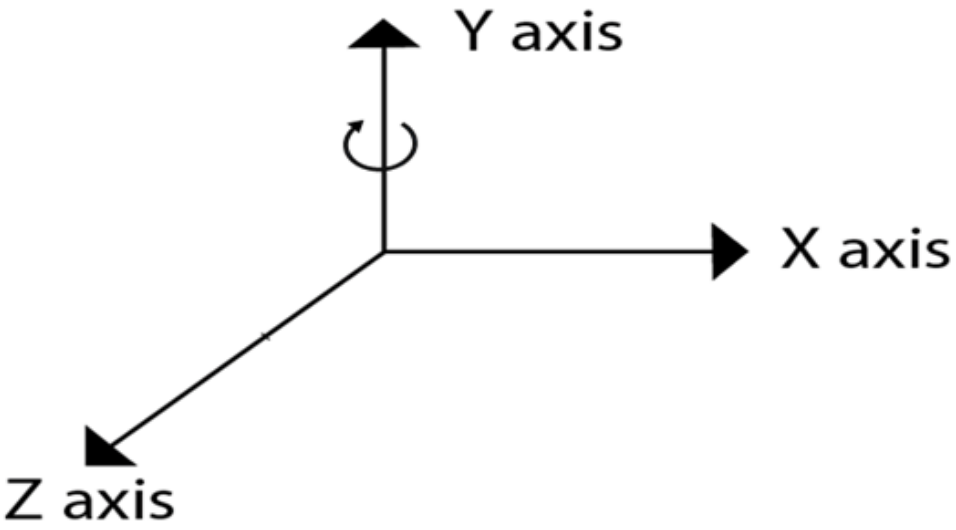
When the object is rotated about an axis that is not parallel to any one of co-ordinate axis, i.e., x, y, z. Then additional transformations are required. First of all, alignment is needed, and then the object is being back to the original position. Following steps are required

Translate the object to the origin

Rotate object so that axis of object coincide with any of coordinate axis.

Perform rotation about co-ordinate axis with whom coinciding is done.

Apply inverse rotation to bring rotation back to the original position.



rotation object Y axis clockwise

Matrix for representing three-dimensional rotations about the Z axis

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matrix for representing three-dimensional rotations about the X axis

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

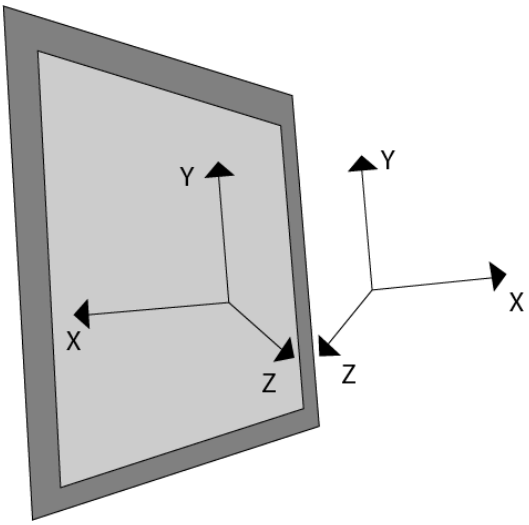
Matrix for representing three-dimensional rotations about the Y axis

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Reflection

It is also called a mirror image of an object. For this reflection axis and reflection of plane is selected. Three-dimensional reflections are similar to two dimensions. Reflection is 180° about the given axis. For reflection, plane is selected (xy,xz or yz). Following matrices show reflection respect to all these three planes.

Reflection relative to XY plane



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

Reflection relative to YZ plane

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

Reflection relative to ZX plane

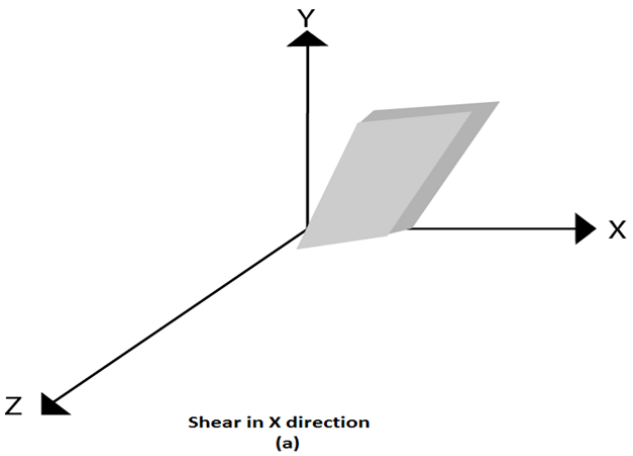
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Shearing

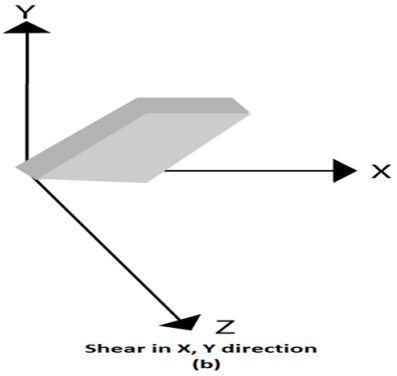
It is change in the shape of the object. It is also called as deformation. Change can be in the x -direction or y -direction or both directions in case of 2D. If shear occurs in both directions, the object will be distorted. But in 3D shear can occur in three directions.

Matrix for shear

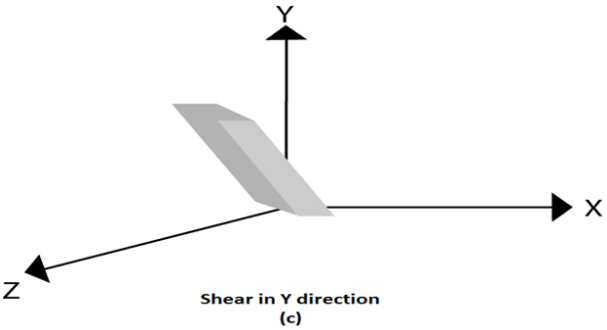
$$\begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Shear in X direction
(a)



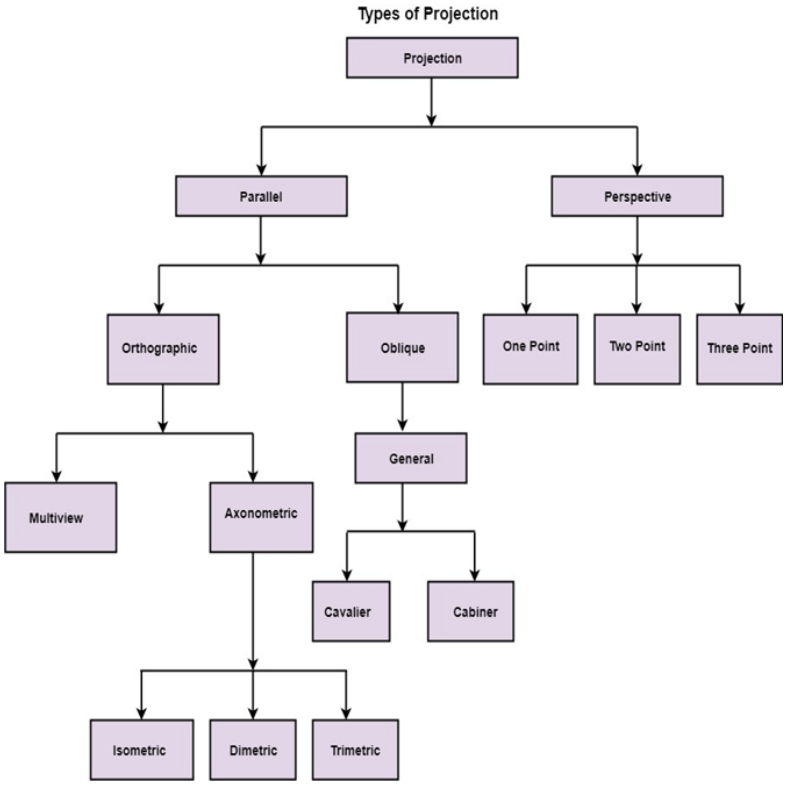
Shear in X, Y direction
(b)



Shear in Y direction
(c)

Projection

It is the process of converting a 3D object into a 2D object. It is also defined as mapping or transformation of the object in projection plane or view plane. The view plane is displayed surface.



Perspective Projection

In perspective projection farther away object from the viewer, small it appears. This property of projection gives an idea about depth. The artist use perspective projection from drawing three-dimensional scenes.

Two main characteristics of perspective are vanishing points and perspective foreshortening. Due to foreshortening object and lengths appear smaller from the center of projection. More we increase the distance from the center of projection, smaller will be the object appear.

Vanishing Point

It is the point where all lines will appear to meet. There can be one point, two point, and three point perspectives.

One Point: There is only one vanishing point as shown in fig (a)

Two Points: There are two vanishing points. One is the x-direction and other in the y -direction as shown in fig (b)

Three Points: There are three vanishing points. One is x second in y and third in two directions.

In Perspective projection lines of projection do not remain parallel. The lines converge at a single point called a center of projection. The projected image on the screen is obtained by points of intersection of converging lines with the plane of the screen. The image on the screen is seen as of viewer's eye were located at the centre of projection, lines of projection would correspond to path travel by light beam originating from object.

Important terms related to perspective

View plane: It is an area of world coordinate system which is projected into viewing plane.

Center of Projection: It is the location of the eye on which projected light rays converge.

Projectors: It is also called a projection vector. These are rays start from the object scene and are used to create an image of the object on viewing or view plane.

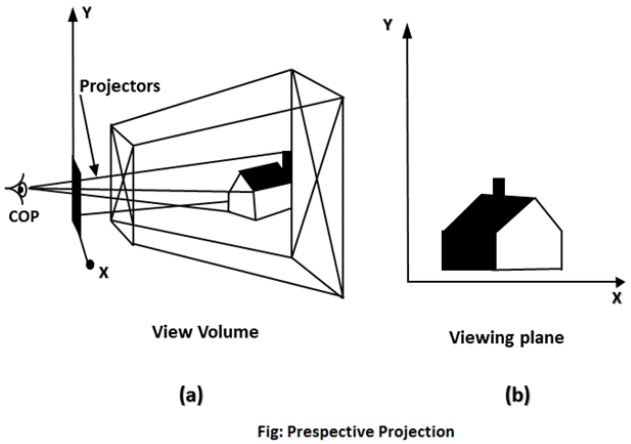


Fig: Prespective Projection

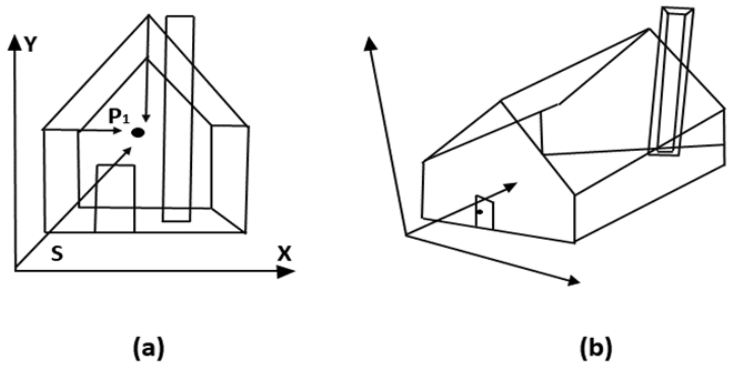
Anomalies in Perspective Projection

It introduces several anomalies due to these object shape and appearance gets affected.

Perspective foreshortening: The size of the object will be small of its distance from the center of projection increases.

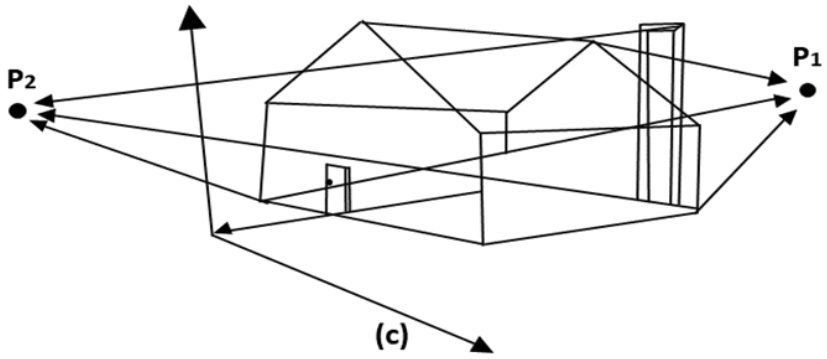
Vanishing Point: All lines appear to meet at some point in the view plane.

Distortion of Lines: A range lies in front of the viewer to back of viewer is appearing to six rollers.



(a)

(b)

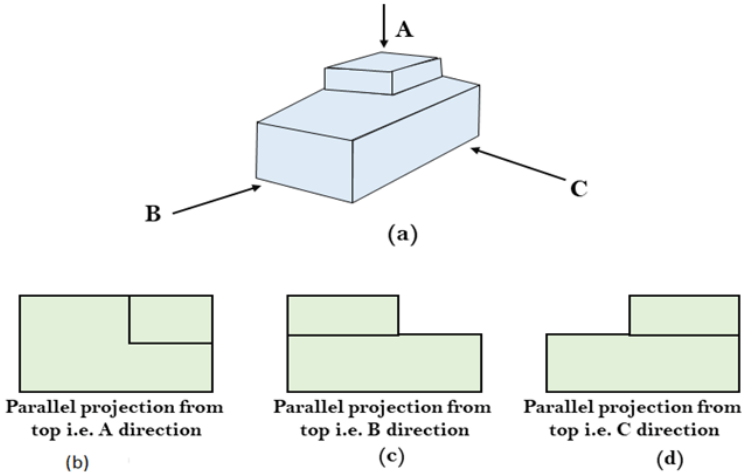


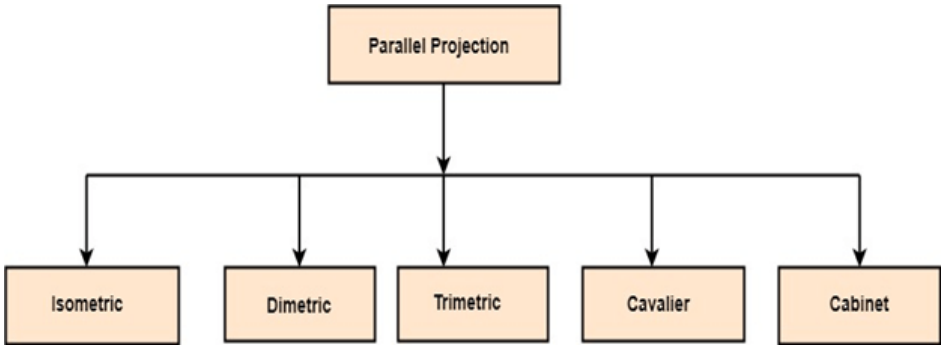
(c)

Parallel Projection

Parallel Projection use to display picture in its true shape and size. When projectors are perpendicular to view plane then is called orthographic projection. The parallel projection is formed by extending parallel lines from each vertex on the object until they intersect the plane of the screen. The point of intersection is the projection of vertex.

Parallel projections are used by architects and engineers for creating working drawing of the object, for complete representations require two or more views of an object using different planes.





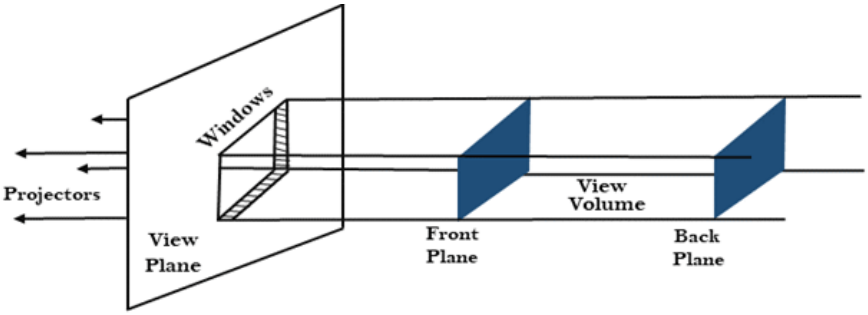
Isometric Projection: All projectors make equal angles generally angle is of 30°.

Dimetric: In these two projectors have equal angles. With respect to two principle axis.

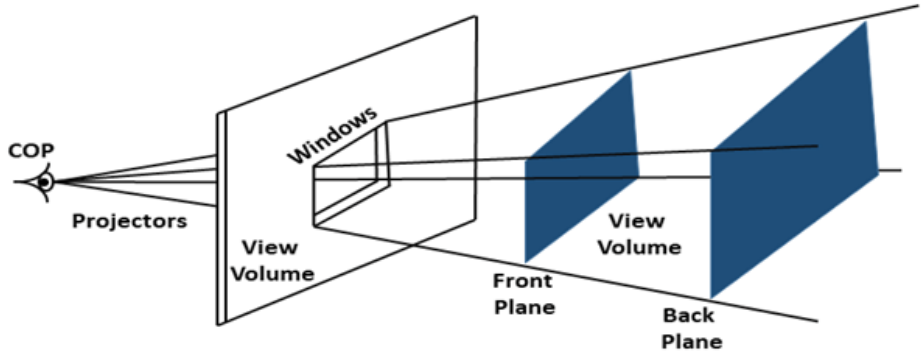
Trimetric: The direction of projection makes unequal angle with their principle axis.

Cavalier: All lines perpendicular to the projection plane are projected with no change in length.

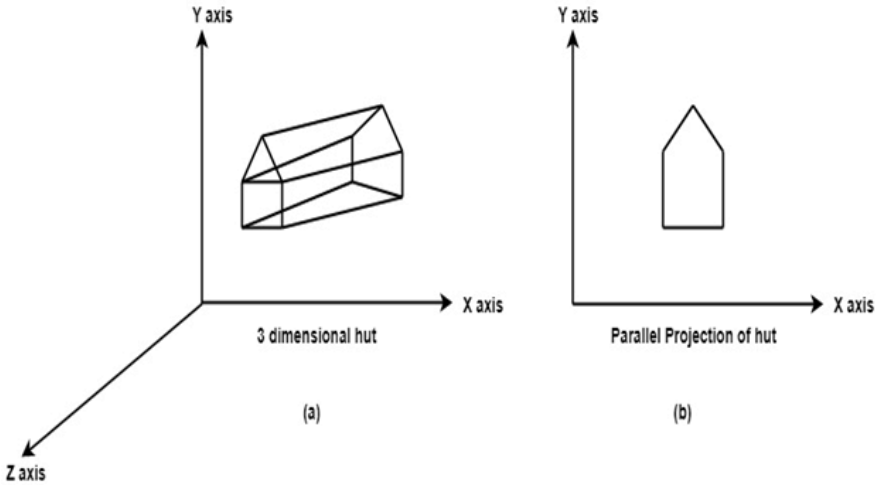
Cabinet: All lines perpendicular to the projection plane are projected to one half of their length. These give a realistic appearance of object.

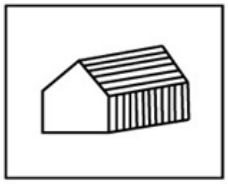


(a) Viewing Volume in orthographic projection

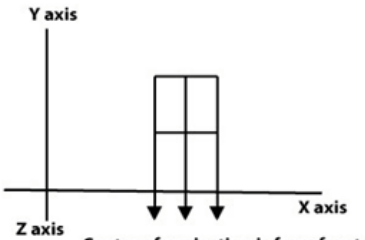


(b) Viewing volume in perspective projection

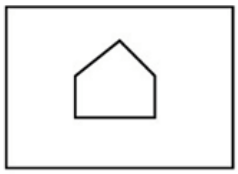




(a) Object original

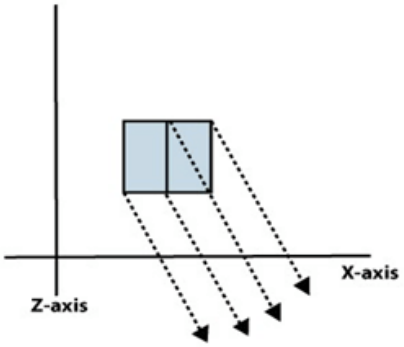


(b)

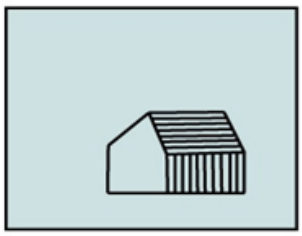


projected object
(c)

Fig (a) shows original object. Fig (b) shows object when projection is taken. Fig (c) gives projected object.



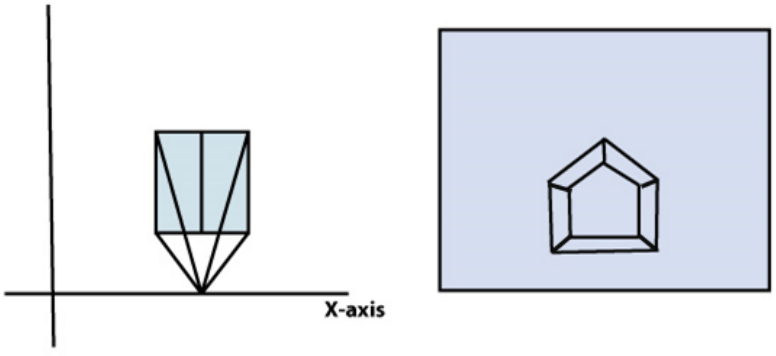
(d)



(e)

Fig (d) changes the direction of projection

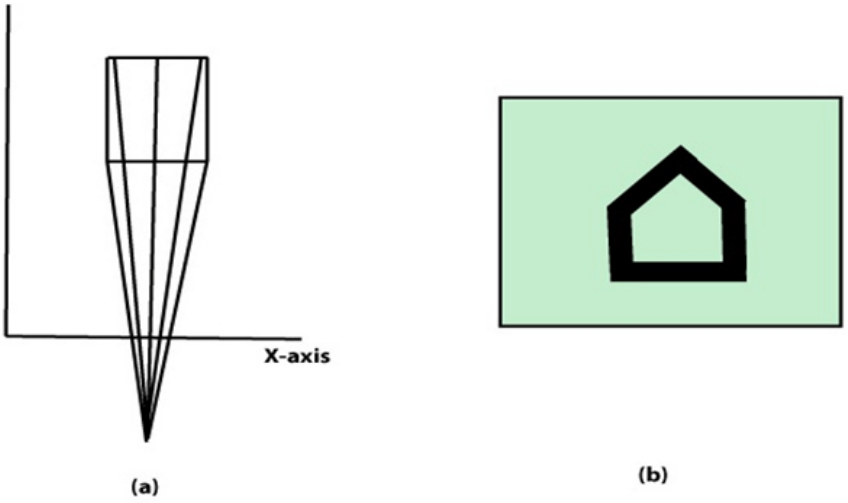
Fig (e) shows object after changing direction of projection.



Center of projection meet at a chosen point

(a)

(b)



(a)

(b)

3D Clipping

3D clipping is required when displaying 3D objects that can have negative z values when transformed to world space. Consider again our 40x40x40 cube. If we assume the cube vertices are defined in world space instead of object space, the cube will be drawn centered at the world space origin. This means the front half of the cube will extend through z=0 and into negative z space. This will cause major problems when we project the world space points to screen space when displaying the cube.

3D clipping solves this problem by clipping a polygon's world space points against a specified z plane, called the near clipping plane. Any z coordinates that extend into

this plane are set to the near clipping value, and their x and y coordinates are adjusted accordingly. Besides the near clipping plane, Fastgraph's 3D clipping also supports a far clipping plane. Objects that are completely behind the far clipping plane are not drawn, as we assume these are too far away for the viewer to see. By default, Fastgraph uses a near clipping value of 1 and a far clipping value of 1000, but we can redefine these Dsetzclip(). Its two parameters are floating point z values that respectively specify the near and far clipping planes. They must be greater than zero (recall that z values are positive in a left-handed 3D coordinate system), or if z-buffering is used, greater than or equal to 1. The far clipping plane must be greater than the near clipping plane.

UNIT 4:CURVES AND SURFACES

Quadric Surfaces

Quadric surfaces are defined by quadratic equations in two dimensional space. Spheres and cones are examples of quadrics. The quadric surfaces of RenderMan are surfaces of revolution in which a finite curve in two dimensions is swept in three dimensional space about one axis to create a surface. A circle centered at the origin forms a sphere. If the circle is not centered at the origin, the circle sweeps out a torus. A line segment with one end lying on the axis of rotation forms a cone. A line segment parallel to the axis of rotation forms a cylinder. The generalization of a line segment creates a hyperboloid by rotating an arbitrary line segment in three dimensional space about the Z axis. The axis of rotation is always the z axis. Each quadric routine has a sweep parameter, specifying the angular extent to which the quadric is swept about z axis. Sweeping a quadric by less than 360 degrees leaves an open surface.

Quadrics

Many common shapes can be modeled with quadrics. Although it is possible to convert quadrics to patches, they are defined as primitives because special-purpose rendering programs render them directly and because their surface parameters are not necessarily preserved if they are converted to patches. Quadric primitives are particularly useful in solid and molecular modeling applications.

All the following quadrics are rotationally symmetric about the z axis. In all the quadrics u and v are assumed to run from 0 to 1. These primitives all define a bounded region on a quadric surface. It is not possible to define infinite quadrics. Note that each quadric is defined relative to the origin of the object coordinate system. To position them at another point or with their symmetry axis in another direction requires the use a modeling transformation. The geometric normal to the surface points ``outward" from the z-axis, if the current orientation matches the orientation of the current transformation and "inward" if they don't match. The sense of a quadric can be reversed by giving negative parameters. For example, giving a negative thetamax parameter in any of the following definitions will turn the quadric inside-out.

Each quadric has a parameterlist. This is a list of token-array pairs where each token is one of the standard geometric primitive variables or a variable which has been defined with RiDeclare. Position variables should not be given with quadrics. All angular arguments to these functions are given in degrees. The trigonometric functions used in their definitions are assumed to also accept angles in degrees.

RiSphere(radius, zmin, zmax, thetamax, parameterlist)

RtFloat radius;

RtFloat zmin, zmax;

RtFloat thetamax;

Requests a sphere defined by the following equations:

$$\phi_{min} = \begin{cases} asin\left(\frac{zmin}{radius}\right) & \text{if } zmin > -radius \\ -90.0 & \text{if } zmin \le -radius \end{cases}$$

$$\phi_{max} = \begin{cases} asin\left(\frac{zmax}{radius}\right) & \text{if } zmax < radius \\ 90.0 & \text{if } zmax \ge radius \end{cases}$$

$$\phi = \phi_{min} + v \cdot (\phi_{max} - \phi_{min})$$
$$\theta = u \cdot thetamax$$
$$x = radius \cdot cos(\theta) \cdot cos(\phi)$$
$$y = radius \cdot sin(\theta) \cdot cos(\phi)$$
$$z = radius \cdot sin(\phi)$$

Note that if zmin > -radius or zmax < radius, the bottom or top of the sphere is open, and that if thetamax is not equal to 360 degrees, the sides are also open.

RIB BINDING

Sphere radius zmin zmax thetamax parameterlist

Sphere [radius zmin zmax thetamax] parameterlist

EXAMPLE

RiSphere(0.5, 0.0, 0.5, 360.0, RI_NULL);

RiCone(height, radius, thetamax, parameterlist)

RtFloat height;

RtFloat radius;

RtFloat thetamax;

Requests a cone defined by the following equations:

$\theta = u \cdot \text{thetamax}$
$x = \text{radius} \cdot (1-v) \cdot \cos(\theta)$
$y = \text{radius} \cdot (1-v) \cdot \sin(\theta)$
$z = v \cdot \text{height}$

Note that the bottom of the cone is open, and if thetamax is not equal to 360 degrees, the sides are open.

RIB BINDING

Cone height radius thetamax parameterlist

Cone [height radius thetamax] parameterlist

EXAMPLE

```
RtColor four_colors[4];

RiCone(0.5, 0.5, 270.0, "Cs", (RtPointer)four_colors, RI_NULL);
```

RiCylinder(radius, zmin, zmax, thetamax, parameterlist)

RtFloat radius;

RtFloat zmin, zmax;

RtFloat thetamax;

Requests a cylinder defined by the following equations:

$$\theta = u \cdot \text{thetamax}$$
$$x = \text{radius} \cdot \cos(\theta)$$
$$y = \text{radius} \cdot \sin(\theta)$$
$$z = \text{zmin} + v \cdot (\text{zmax} - \text{zmin})$$

Note that the cylinder is open at the top and bottom, and if thetamax is not equal to 360 degrees, the sides also are open.

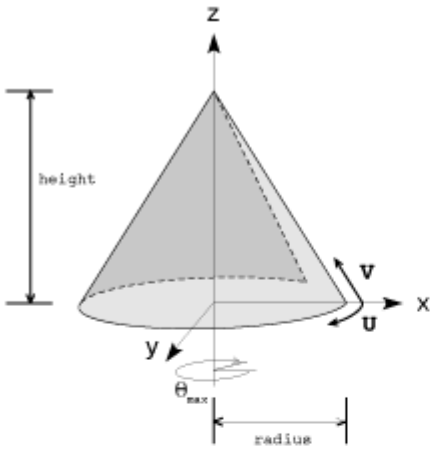
RIB BINDING

Cylinder radius zmin zmax thetamax parameterlist

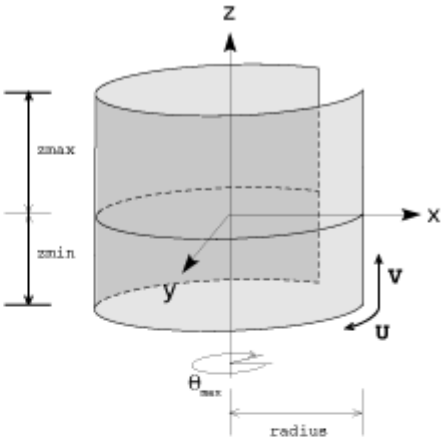
Cylinder [radius zmin zmax thetamax] parameterlist

EXAMPLE

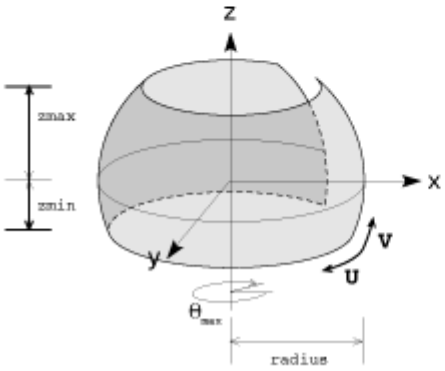
Cylinder .5 .2 1 360



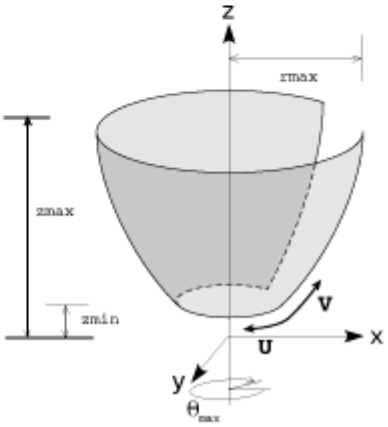
RiCone



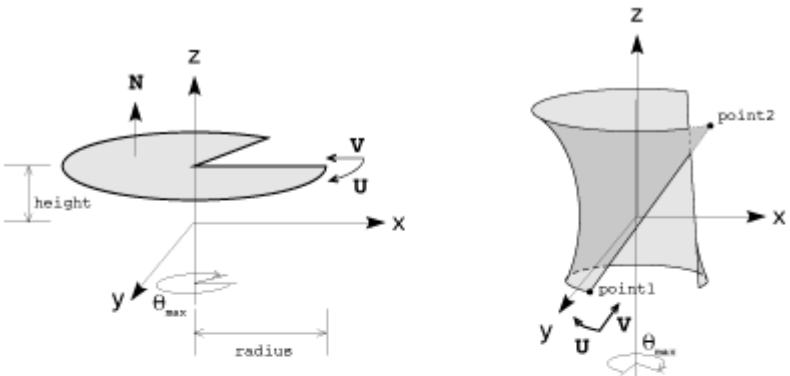
RiCylinder



RiSphere

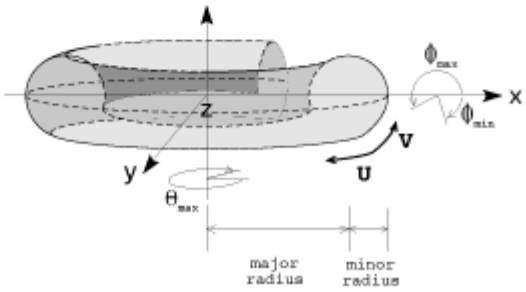


RiParaboloid



RiDisk

RiHyperboloid



RiTorus

Quadric surface primitives

An ellipsoid is a surface that may be obtained from a sphere by deforming it by means of directional scalings, or more generally, of an affine transformation.

An ellipsoid is a quadric surface; that is, a surface that may be defined as the zero set of a polynomial of degree two in three variables. Among quadric surfaces, an ellipsoid is characterized by either of the two following properties. Every planar cross section is either an ellipse, or is empty, or is reduced to a single point (this explains the name, meaning "ellipse like"). It is bounded, which means that it may be enclosed in a sufficiently large sphere.

An ellipsoid has three pairwise perpendicular axes of symmetry which intersect at a center of symmetry, called the center of the ellipsoid. The line segments that are delimited on the axes of symmetry by the ellipsoid are called the principal axes, or simply axes of the ellipsoid. If the three axes have different lengths, the ellipsoid is said to be tri-axial or rarely scalene, and the axes are uniquely defined.

If two of the axes have the same length, then the ellipsoid is an ellipsoid of revolution, also called a spheroid. In this case, the ellipsoid is invariant under a rotation around

the third axis, and there are thus infinitely many ways of choosing the two perpendicular axes of the same length. If the third axis is shorter, the ellipsoid is an oblate spheroid; if it is longer, it is a prolate spheroid. If the three axes have the same length, the ellipsoid is a sphere,

.Standard equation:

Using a Cartesian coordinate system in which the origin is the center of the ellipsoid and the coordinate axes are axes of the ellipsoid, the implicit equation of the ellipsoid has the standard form

{\displaystyle {x^{2} \over a^{2}}+{y^{2} \over b^{2}}+{z^{2} \over c^{2}}=1,}{x^{2} \over a^{2}}+{y^{2} \over b^{2}}+{z^{2} \over c^{2}}=1,

where a, b, c are positive real numbers.

The points (a, 0, 0), (0, b, 0) and (0, 0, c) lie on the surface. The line segments from the origin to these points are called the principal semi-axes of the ellipsoid, because a, b, c are half the length of the principal axes. They correspond to the semi-major axis and semi-minor axis of an ellipse.

If {\displaystyle a=b>c,}{\displaystyle a=b>c,} one has an oblate spheroid; if {\displaystyle a=b<c,}{\displaystyle a=b<c,} one has a prolate spheroid; if {\displaystyle a=b=c,}{\displaystyle a=b=c,} one has a sphere.

Hidden line and surfaces

When we view a picture containing non-transparent objects and surfaces, then we cannot see those objects from view which are behind from objects closer to eye. We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called Hidden-surface problem.

There are two approaches for removing hidden surface problems − Object-Space method and Image-space method. The Object-space method is implemented in physical coordinate system and image-space method is implemented in screen coordinate system.

When we want to display a 3D object on a 2D screen, we need to identify those parts of a screen that are visible from a chosen viewing position.

Depth Buffer Z−Buffer Method

This method is developed by Cutmull. It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest visible surface.

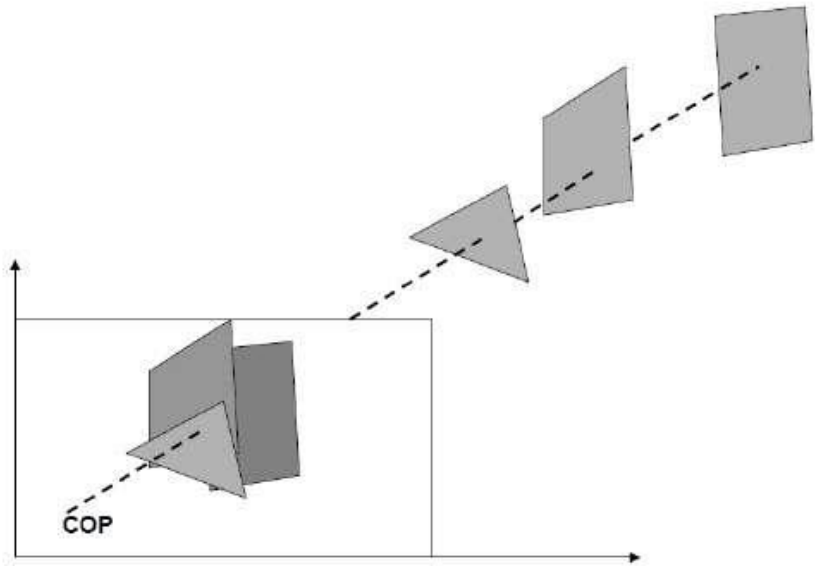
In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest smallestz surface determines the color to be displayed in the frame buffer.

It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named frame buffer and depth buffer, are used.

Depth buffer is used to store depth values for x,y position, as surfaces are processed $0 \leq depth \leq 1$.

The frame buffer is used to store the intensity value of color value at each position x,y.

The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.



Algorithm

Step-1 − Set the buffer values −

Depth buffer x,y = 0

Frame buffer x,y = background color

Step-2 − Process each polygon One at a time

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

If Z > depth buffer x,y

Compute surface color,

set depth buffer x,y = z,

framebuffer x,y = surfacecolor x,y

Advantages:

It is easy to implement.

It reduces the speed problem if implemented in hardware.

It processes one object at a time.

Disadvantages:

It requires large memory.

It is time consuming process.

Back-Face Detection

A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" tests. A point x,y,z is "inside" a polygon surface with plane parameters A, B, C, and D if When an inside point is along the line of sight to the surface, the polygon must be a back face weareinsidethatfaceandcannotseethefrontofitfromourviewingposition.

We can simplify this test by considering the normal vector N to a polygon surface, which has Cartesian components A,B,C.In general, if V is a vector in the viewing direction from the eye or"camera" position, then this polygon is a back face if

V.N > 0

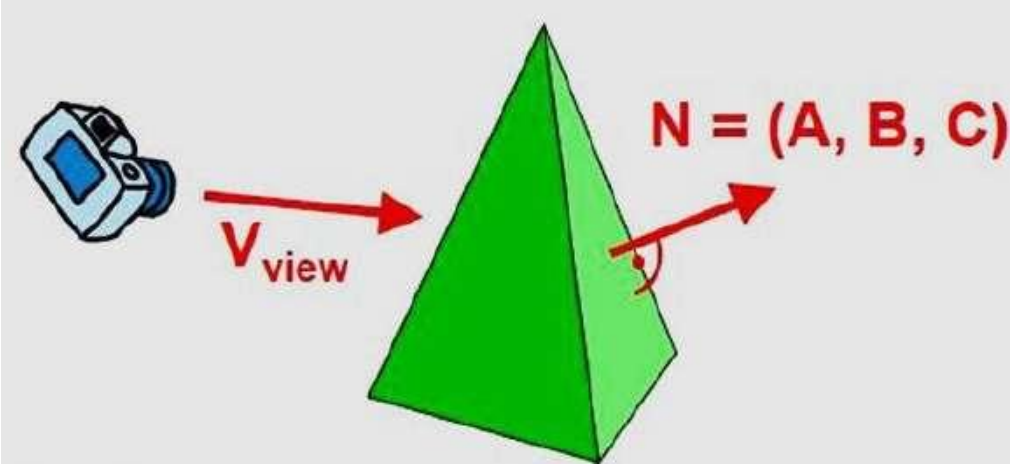
Furthermore, if object descriptions are converted to projection coordinates and your viewing direction is parallel to the viewing z-axis, then −

V = (0, 0, Vz) and V.N = VZC

So that we only need to consider the sign of C the component of the normal vector N.

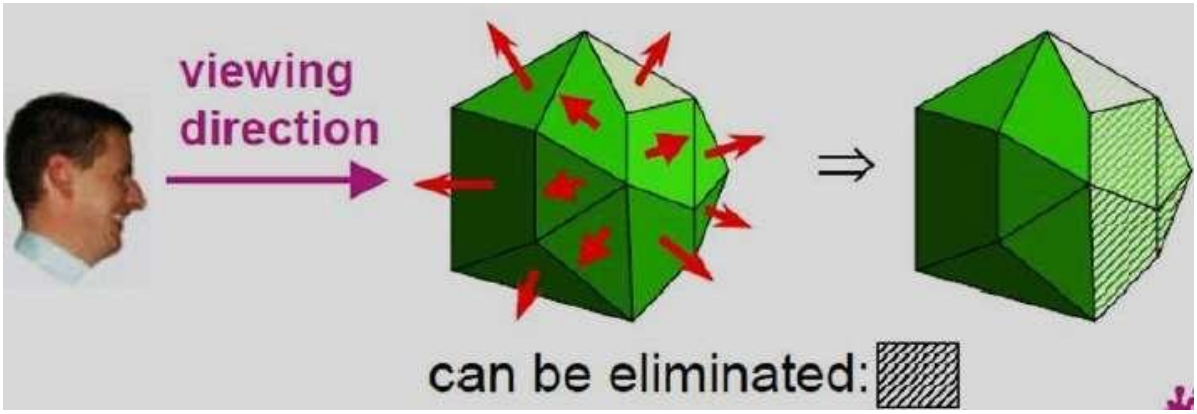
In a right-handed viewing system with viewing direction along the negative ZV axis, the polygon is a back face if C < 0. Also, we cannot see any face whose normal has z component C = 0, since your viewing direction is towards that polygon. Thus, in general, we can label any polygon as a back face if its normal vector has a z component value −

C <= 0



Similar methods can be used in packages that employ a left-handed viewing system. In these packages, plane parameters A, B, C and D can be calculated from polygon vertex coordinates specified in a clockwise direction unlike the counterclockwise direction used in a right−handed system.

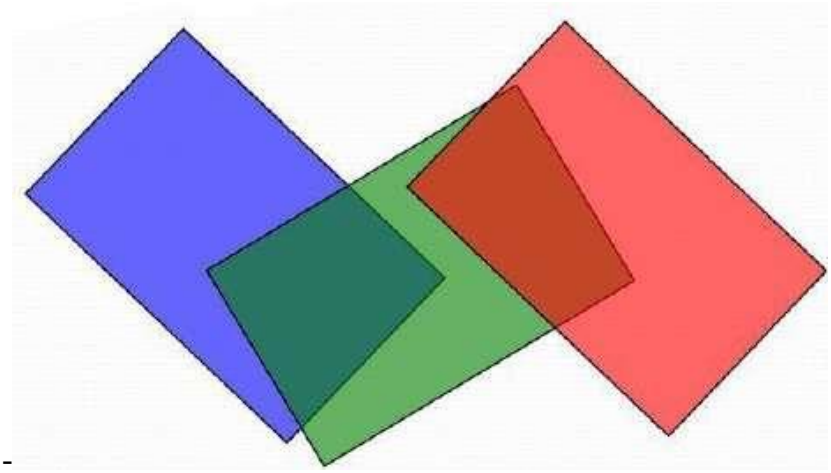
Also, back faces have normal vectors that point away from the viewing position and are identified by C >= 0 when the viewing direction is along the positive Zv axis. By examining parameter C for the different planes defining an object, we can immediately identify all the back faces.



A-Buffer Method

The A-buffer method is an extension of the depth-buffer method. The A-buffer method is a visibility detection method developed at Lucas film Studios for the rendering system Renders Everything You Ever Saw REYES.

The A-buffer expands on the depth buffer method to allow transparencies. The key data structure in the buffer is the accumulation buffer.

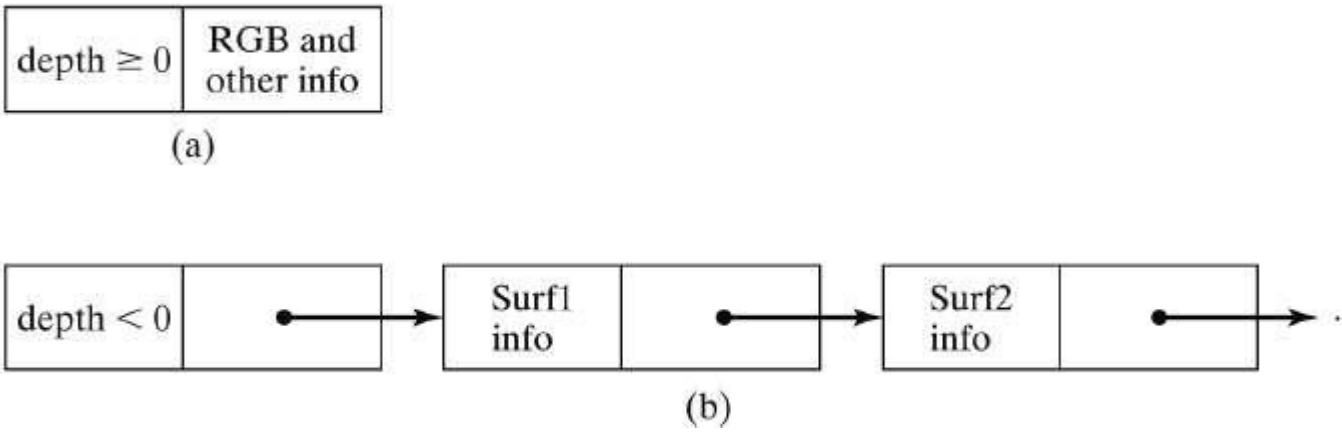


-

Each position in the A-buffer has two fields −

Depth field − It stores a positive or negative real number

Intensity field − It stores surface-intensity information or a pointer value



If depth >= 0, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

If depth < 0, it indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data. The surface buffer in the A-buffer includes −

-RGB intensity components

-Opacity Parameter

-Depth

-Percent of area coverage

-Surface identifier

The algorithm proceeds just like the depth buffer algorithm. The depth and opacity values are used to determine the final color of a pixel.

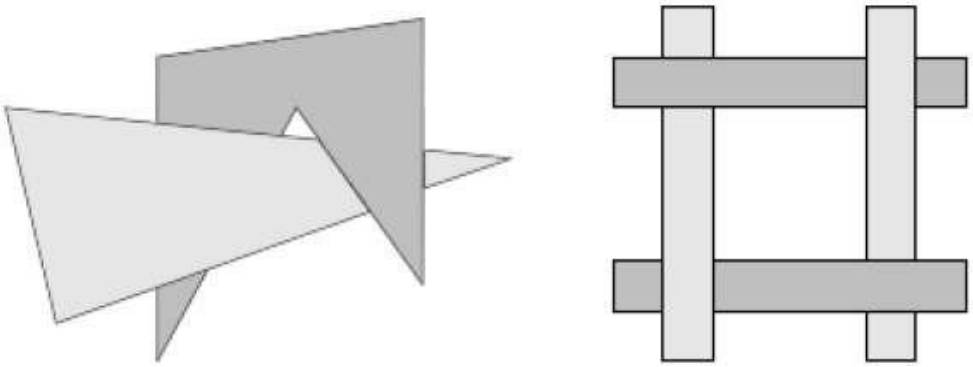
Depth Sorting Method

Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions −

First, the surfaces are sorted in order of decreasing depth.

Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the painter's algorithm. The following figure shows the effect of depth sorting −



The algorithm begins by sorting by depth. For example, the initial “depth” estimate of a polygon may be taken to be the closest z value of any vertex of the polygon.

Let us take the polygon P at the end of the list. Consider all polygons Q whose z-extents overlap P’s. Before drawing P, we make the following tests. If any of the following tests is positive, then we can assume P can be drawn before Q.

Do the x-extents not overlap?

Do the y-extents not overlap?

Is P entirely on the opposite side of Q’s plane from the viewpoint?

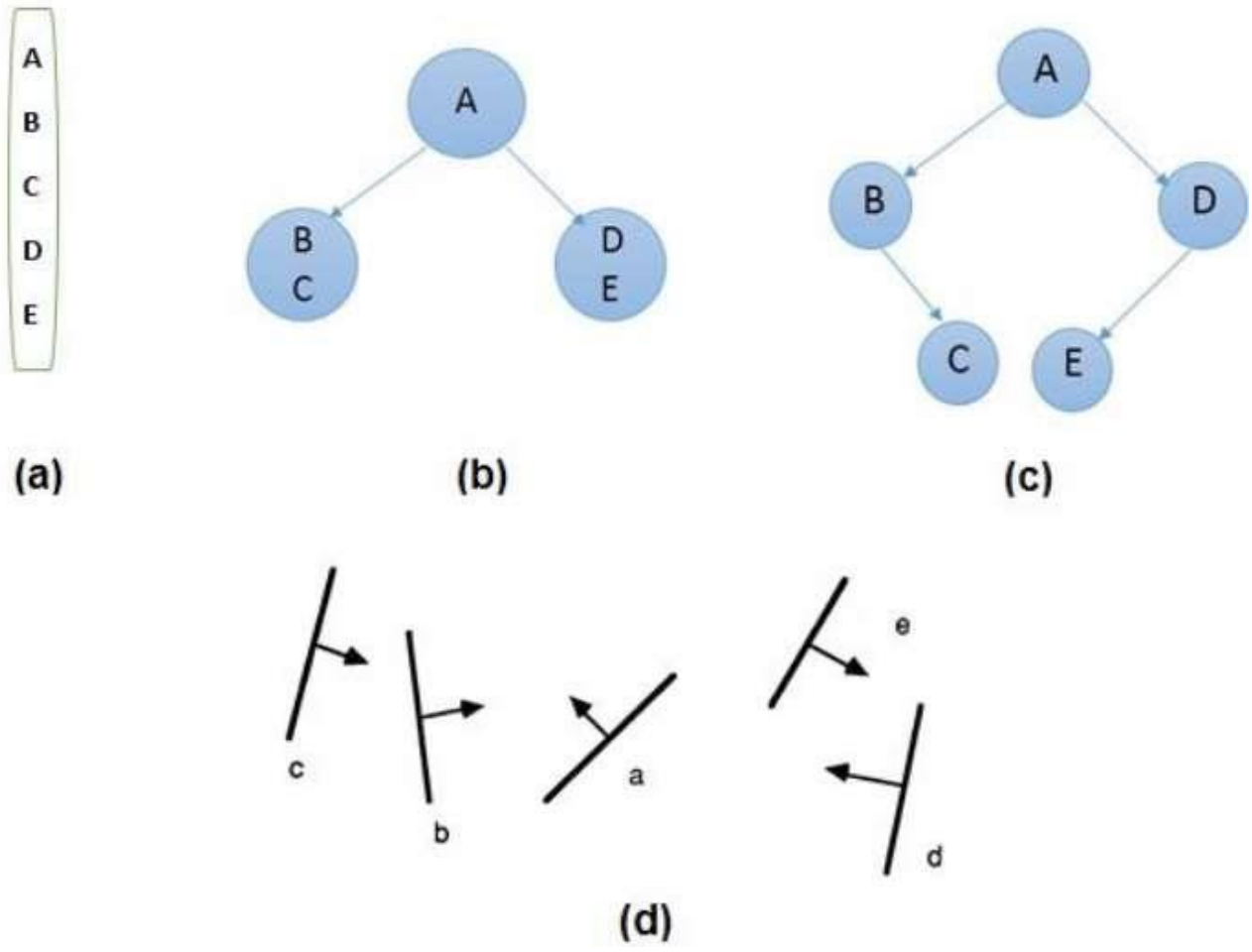
Is Q entirely on the same side of P’s plane as the viewpoint?

Do the projections of the polygons not overlap?

If all the tests fail, then we split either P or Q using the plane of the other. The new cut polygons are inserting into the depth order and the process continues. Theoretically, this partitioning could generate O(n2) individual polygons, but in practice, the number of polygons is much smaller.

Binary Space Partition BSP Trees

Binary space partitioning is used to calculate visibility. To build the BSP trees, one should start with polygons and label all the edges. Dealing with only one edge at a time, extend each edge so that it splits the plane in two. Place the first edge in the tree as root. Add subsequent edges based on whether they are inside or outside. Edges that span the extension of an edge that is already in the tree are split into two and both are added to the tree.



From the above figure, first take A as a root.

Make a list of all nodes in figure a.

Put all the nodes that are in front of root A to the left side of node A and put all those nodes that are behind the root A to the right side as shown in figure b.

Process all the front nodes first and then the nodes at the back.

As shown in figure c, we will first process the node B. As there is nothing in front of the node B, we have put NIL. However, we have node C at back of node B, so node C will go to the right side of node B.

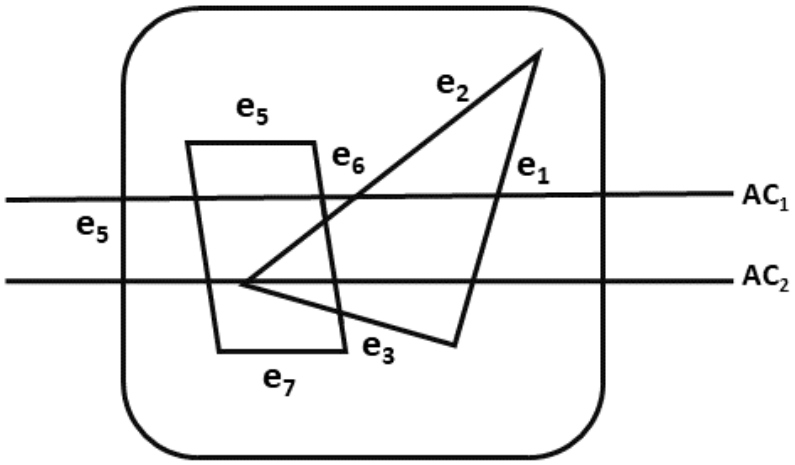
Repeat the same process for the node D.

Scan Line Algorithm

It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active

edge list. So accurate bookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. Active Edge List (AEL) contain edges a given scan line intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.

Following figures shown edges and active edge list. The active edge list for scan line AC1contain e1,e2,e5,e6 edges. The active edge list for scan line AC2contain e5,e6,e1.



Scan line can deal with multiple surfaces. As each scan line is processed, this line will intersect many surfaces. The intersecting line will determine which surface is visible. Depth calculation for each surface is done. The surface rear to view plane is defined. When the visibility of a surface is determined, then intensity value is entered into refresh buffer.

lgorithm

Step1: Start algorithm

Step2: Initialize the desired data structure

1.Create a polygon table having color, edge pointers, coefficients

2.Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.

3.Create Active edge list. This will be sorted in increasing order of x.

4.Create a flag F. It will have two values either on or off.

Step3: Perform the following steps for all scan lines

1.Enter values in Active edge list (AEL) in sorted order using y as value

2.Scan until the flag, i.e. F is on using a background color

3.When one polygon flag is on, and this is for surface S1enter color intensity as I1into refresh buffer

4.When two or image surface flag are on, sort the surfaces according to depth and use intensity value Sn for the nth surface. This surface will have least z depth value

5.Use the concept of coherence for remaining planes.

Step4: Stop Algorithm