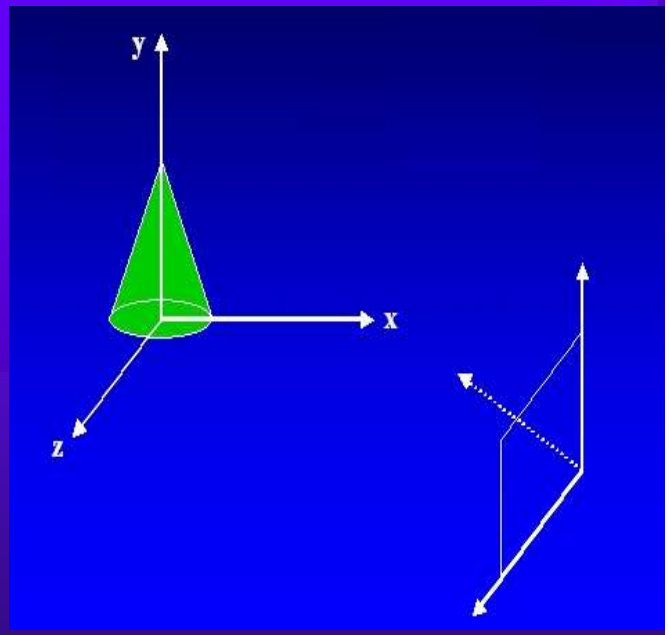
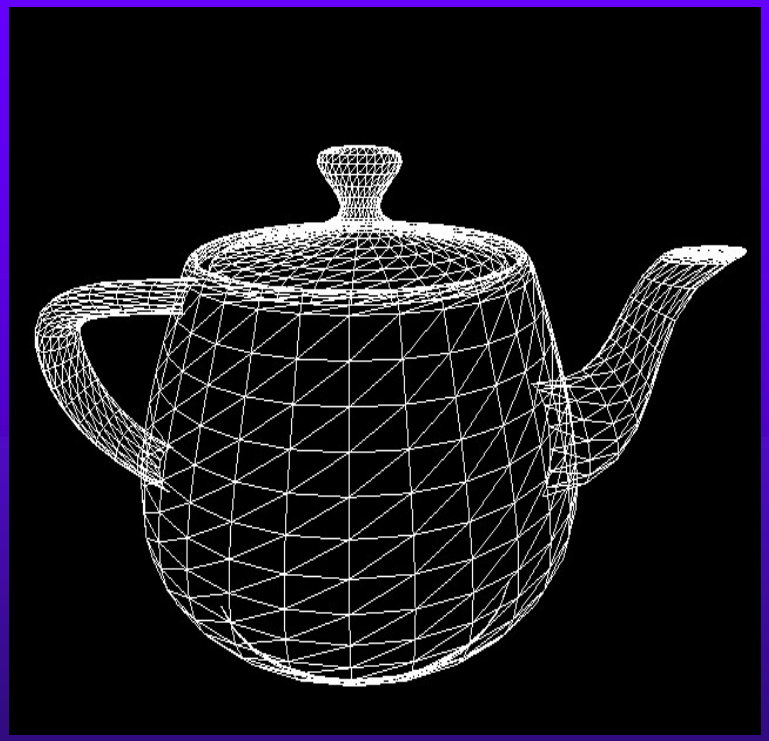
Three-Dimensional Concepts

Coordinate Reference

- To obtain a display of 3d scene that has been modeled in world coordinates ,set up a coordinate reference for the camera.
- This coordinate reference defines the position and orientation for the plane of the camera film.
- Object descriptions are transferred to the camera reference coordinates and projected onto the selected display plane.
- We can display the objects in wireframe form or we can apply lighting and surface rendering techniques to shade the visible surfaces.

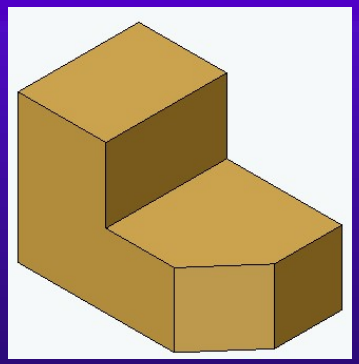


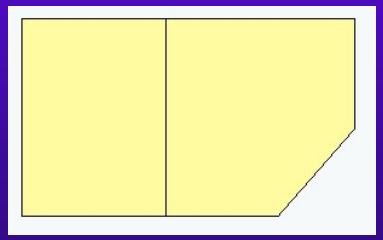
Wireframe

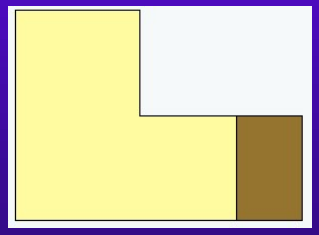


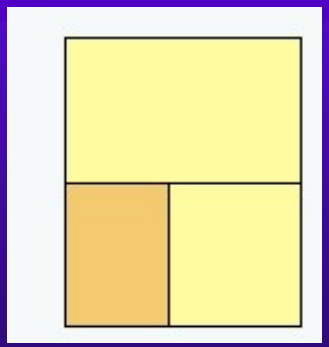
Three-Dimensional Display Methods
Parallel Projection

- Generates view of a solid object.
- Project visible points on the object surface along parallel lines onto the display plane to obtain 2d view.
- Parallel lines are still parallel after projection.



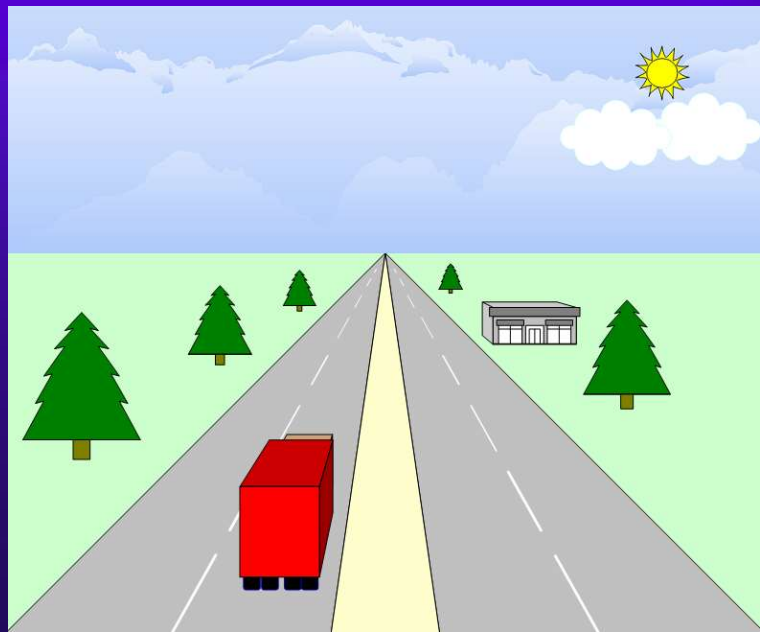






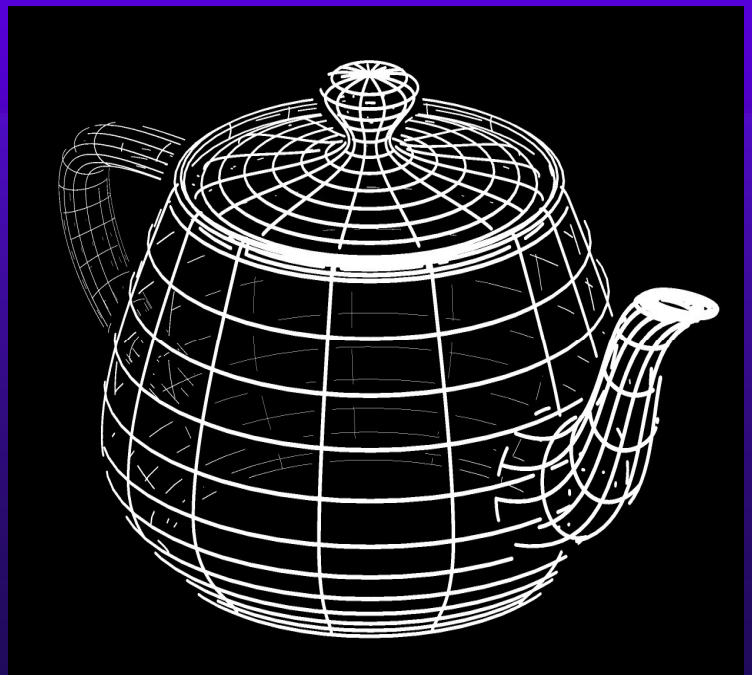
Perspective projection

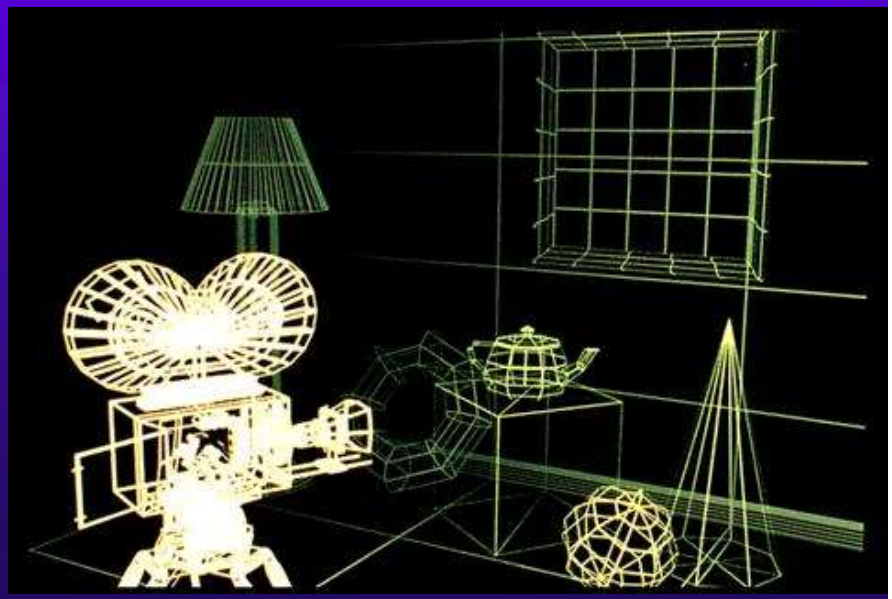
- Project points to the display plane along converging paths.
- This causes objects farther from the viewing position to be displayed smaller than the nearer objects.
- Parallel lines in a scene that are not parallel to the display plane are projected into converging lines.
- This is the way that our eyes and a camera lens form images.



Three-Dimensional Display Methods
Depth cueing

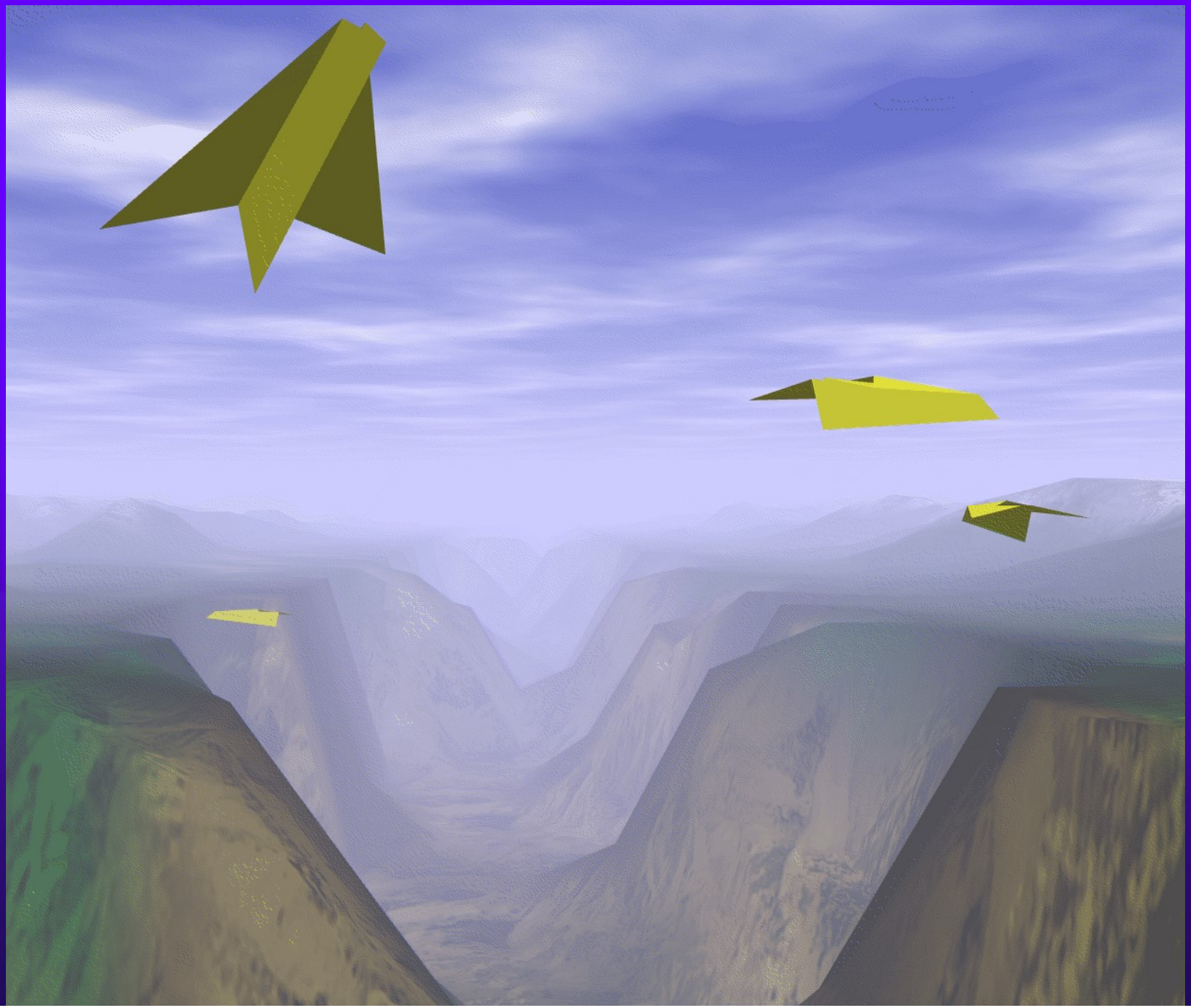
- Identify which is the front and which is the back of displayed objects
- For wireframe displays
  - Vary the intensity of objects according to their distance from viewing position
  - The line closest to the viewing position are displayed with the highest intensities than farther away





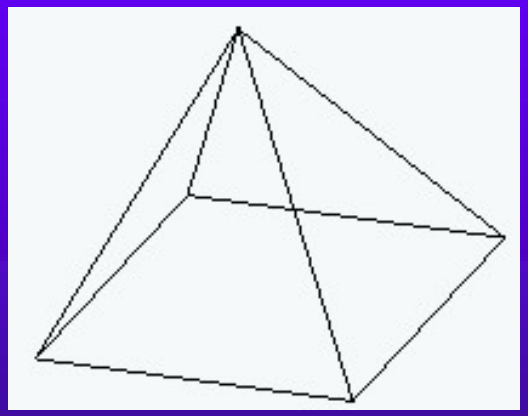
Three-Dimensional Display Methods
Depth cueing

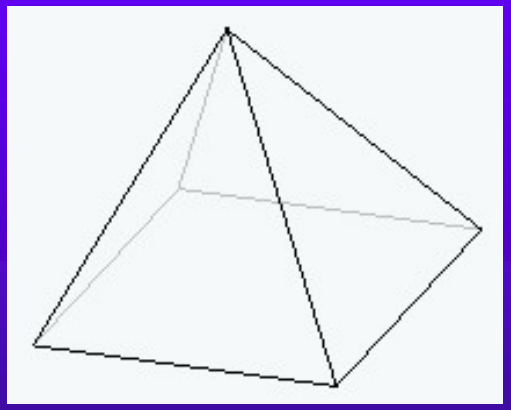
- Another application of depth cueing is modeling the effect of the atmosphere on the perceived intensity of objects.
- Distant objects appear dimmer to us than the nearer objects due to ,light scattering by dust particles, haze and smoke.
- Atmospheric effects can change the perceived color of an object and model.

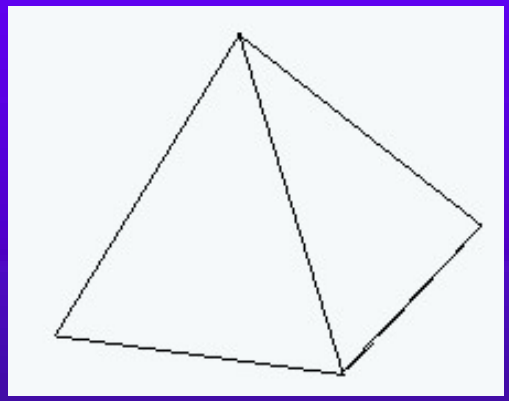


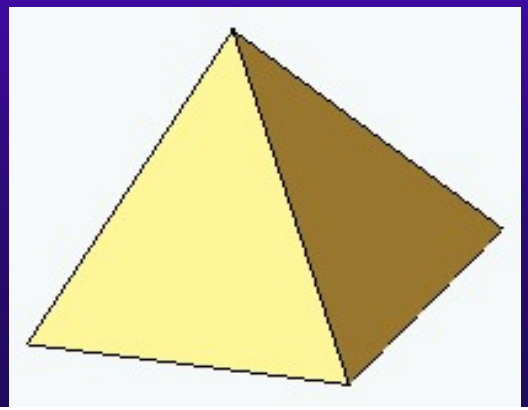
Three-Dimensional Display Methods
Visible line and surface identification

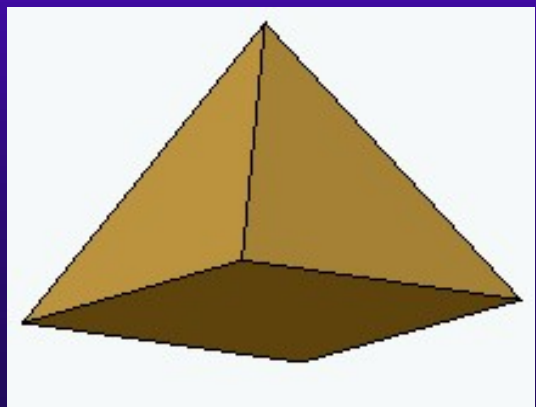
- Highlight the visible lines or display them in different color
- Display nonvisible lines as dashed lines
- Remove the nonvisible lines





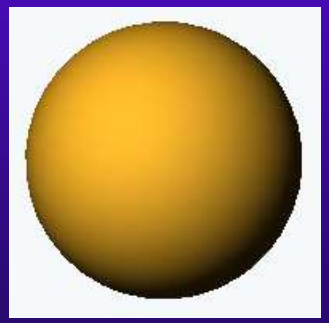


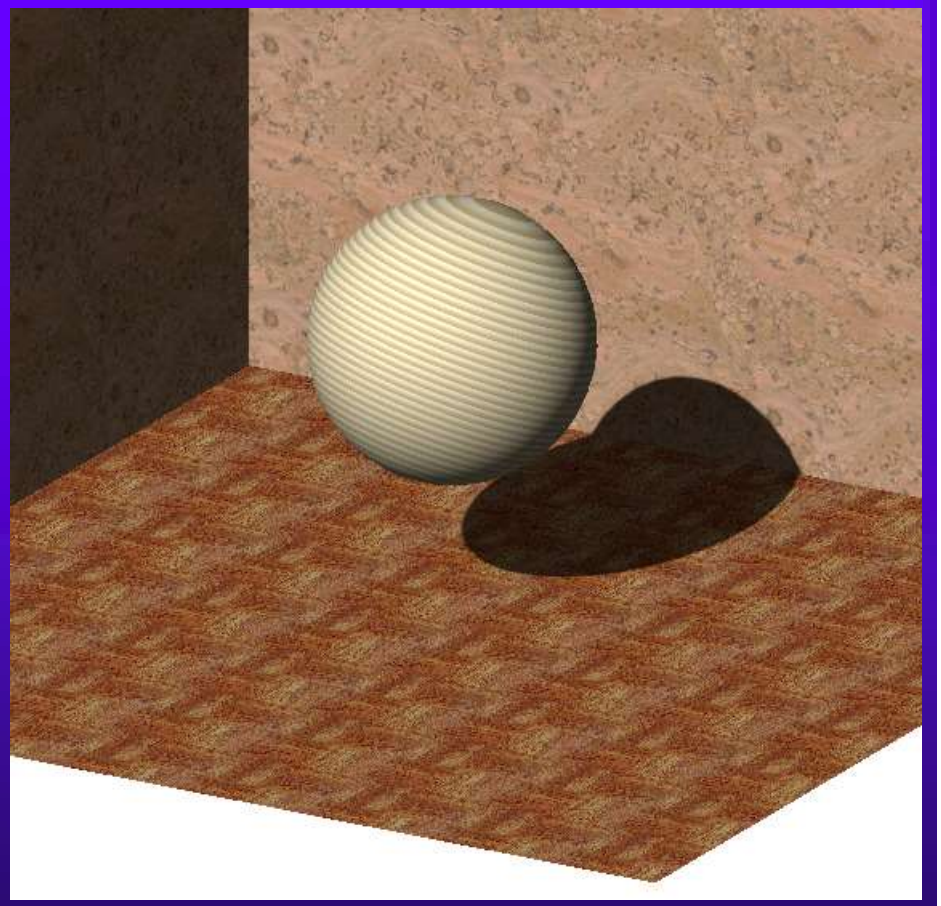




Three-Dimensional Display Methods
Surface Rendering

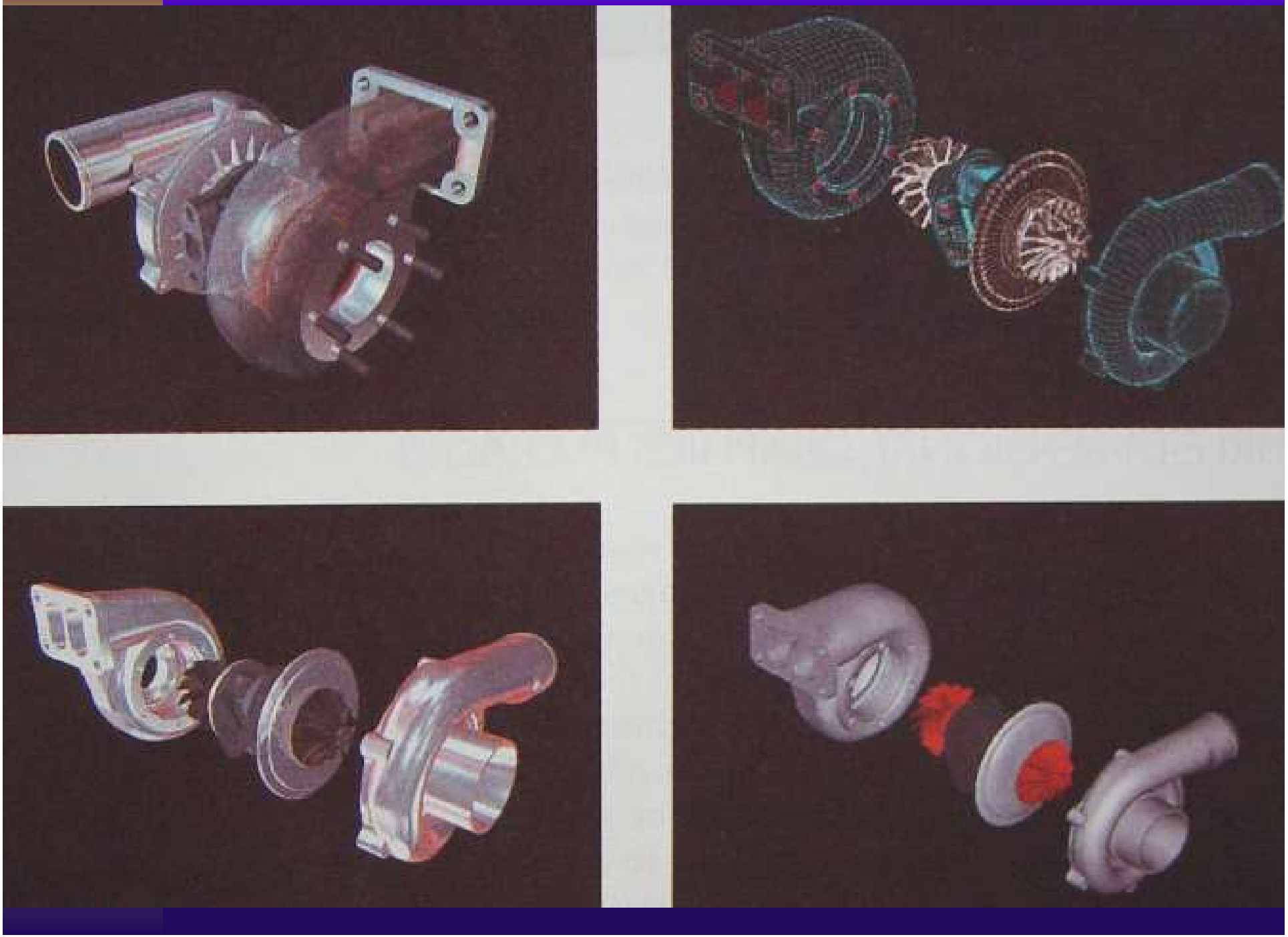
- Set the surface intensity of objects according to
  - Lighting conditions in the scene
  - Assigned surface characteristics





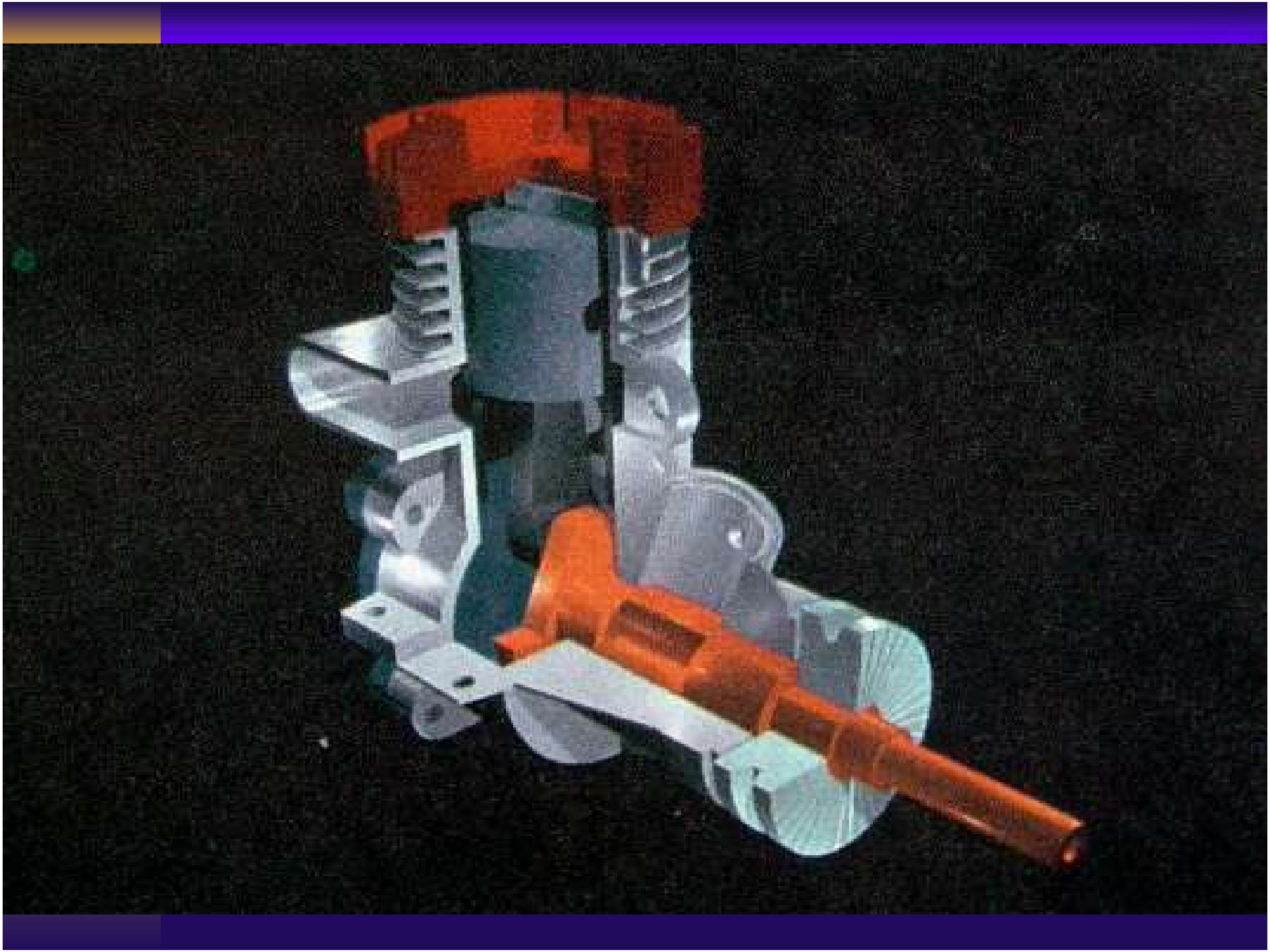
Three-Dimensional Display Methods

- Exploded view
  - Show the internal structure and relationship of the object parts



Three-Dimensional Display Methods

- Cutaway view
  - Remove part of the visible surfaces to show internal structure.



3D Object Representations

Introduction

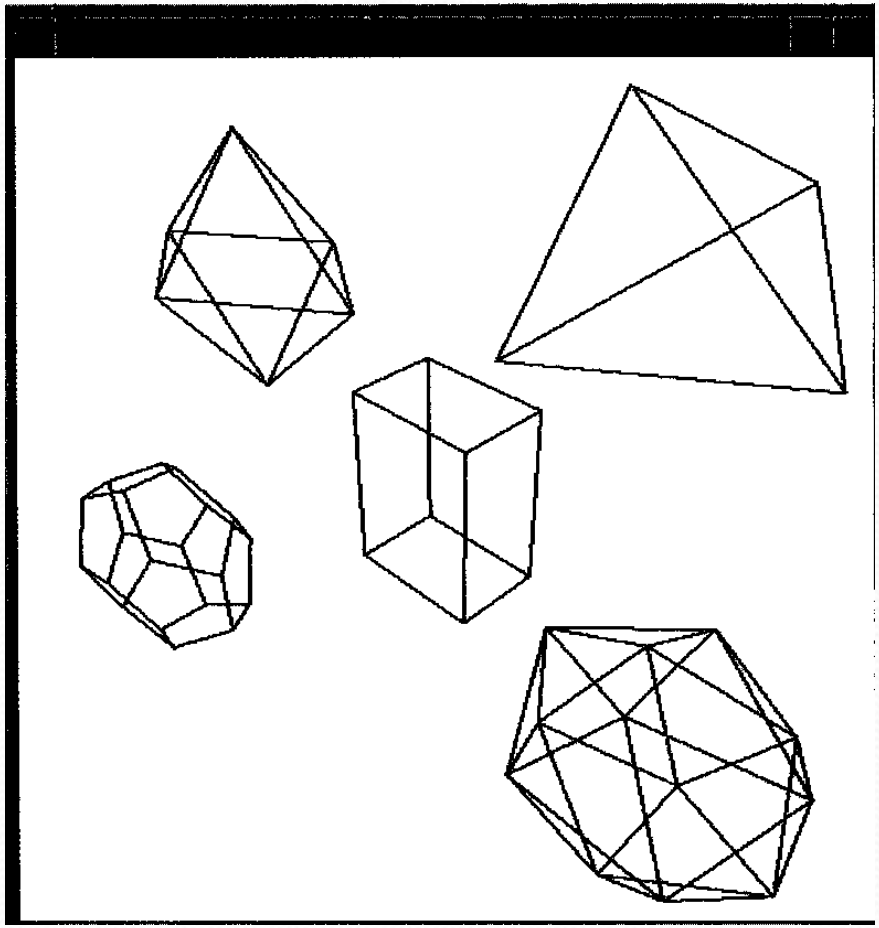
3D Representations

- Basic boundary representations
- Blobby objects
- Various solid modeling principles
- BSP-trees
- Shape Grammars
- L-Grammars (Graftals)
- Particle systems
- Physically based modeling principles

Basic boundary representations

- Polyhedra (a set of surface polygons)
  - triangles, quadrilaterals
- Quadric surfaces (second degree equations)
  - sphere, ellipsoid, torus
- Superquadrics (additional parameters)
  - superellipse (2D), superellipsoid (3D)
- Spline surfaces
  - Bézier, B-spline, rational splines (NURBS)

Polyhedra examples

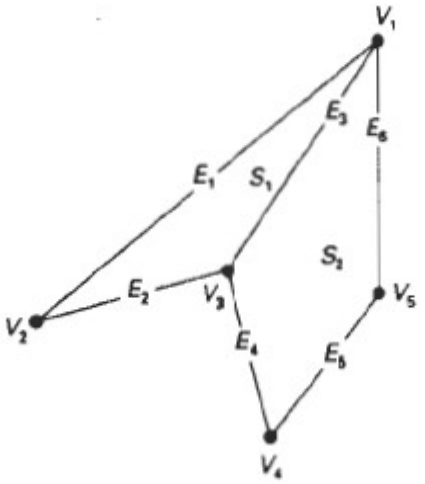


Polygon Surfaces

- Object is represented as a set of surface polygons.
- Simplifies and speeds up surface rendering as surfaces are described as linear equations.
- Polyhedron can be represented
  - By precise surface features
  - Polygon mesh

Polygon Table

- Specify polygon surface as
  - Set of vertices & associated attributes
- Polygon info is stored as data tables
  1. Geometric tables – vertex & orientation
  2. attribute tables – transparency, reflectivity
- Geometric data is stored as 3 lists
  - Vertex table, edge table & 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-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Figure 10-2
Geometric data table representation for two adjacent polygon surfaces, formed with six edges and five vertices.

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

Figure 10-3
Edge table for the surfaces of Fig. 10-2 expanded to include pointers to the polygon table.

Plane Equations

- Equation of a plane surface
  - $Ax + By + Cz + D = 0$
- (x,y,z) – any point on the plane
- Coefficients A,B,C,D – constants
- To find A,B,C,D solve sets of plane eqns.
- (x_1,y_1,z_1) (x_2,y_2,z_2) , (x_3,y_3,z_3)
  - $(A/D)x_k + (B/D)\, y_k + (C/D)z_k = -1$ $k=1,2,3$

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

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

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{10-3}$$

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

$$\begin{aligned} A &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\ B &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\ C &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\ D &= -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1) \end{aligned} \tag{10-4}$$

Orientation of a plane surface

- Can be described with the normal vector, which has Cartesian components(A,B,C)
- Need to distinguish between two sides of the polygon surface (inside and outside)
- Normal vector will be from inside to outside if
  - polygon vertices are specified in counterclockwise direction &
  - Viewing from the outer side of the plane in a right handed coordinate system.

Normal Vector N Calculations using unit cube

- Method 1:
  - Select 3 vertices in counterclockwise direction
  - Compute A=1,B=0,C=0,D=-1 by substituting these vertices in determinant eqns.
- Method 2:
  - Use cross product
    - N = (V2-V1) × (V3-V1)

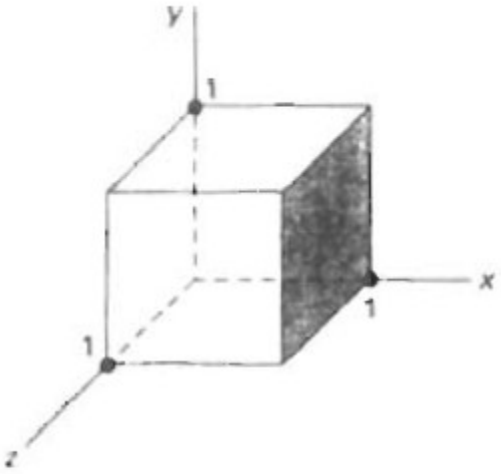


Figure 10-5
The shaded polygon surface of the unit cube has plane equation $x - 1 = 0$ and normal vector $\mathbf{N} = (1, 0, 0)$.

Inequalities of Plane Eqns.

- Plane eqns. are also used to find the position of the spatial points relative to the plane surfaces.

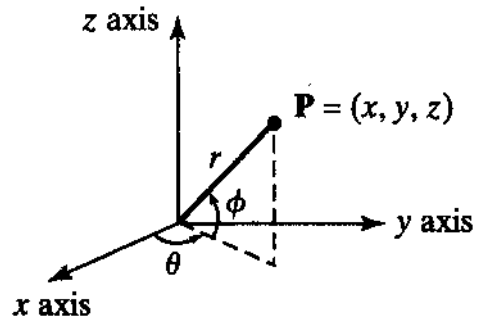
- If (x,y,z) not on surface,
  - $Ax + By + Cz + D \neq 0$

- $Ax + By + Cz + D < 0$ point lies inside the surface
- $Ax + By + Cz + D > 0$ point lies outside the surface

Quadric - Sphere

A point on the sphere surface satisfies:

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

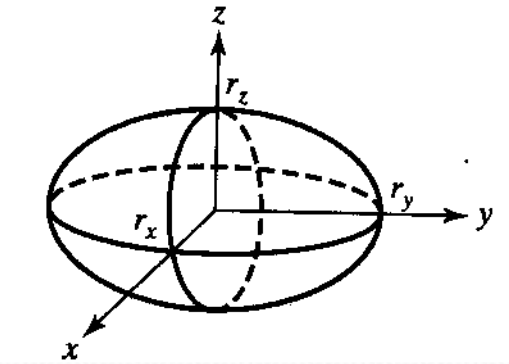


$$x = r\cos\phi\cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r\cos\phi\sin\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r\sin\phi$$

Quadric - ellipsoid



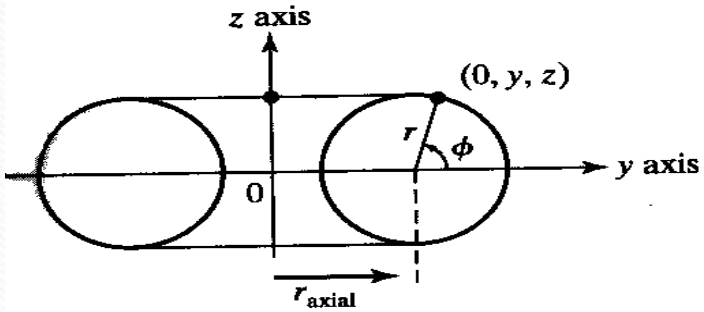
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

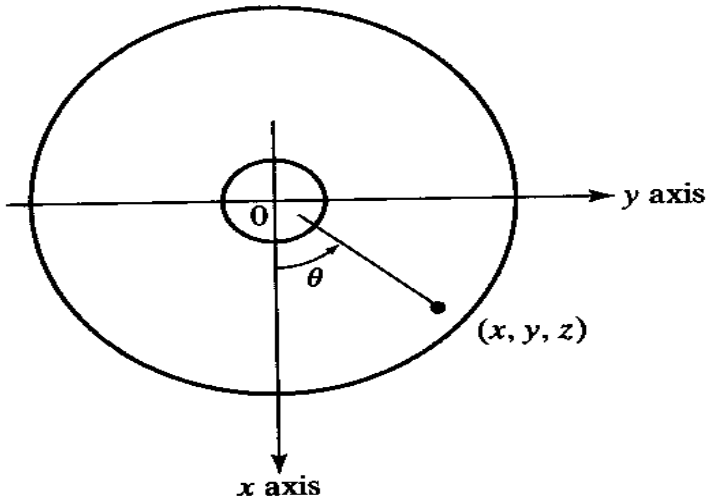
$$x = r_x \cos\phi \cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r_y \cos\phi \sin\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r_z \sin\phi$$

Quadric - torus





Side View

Top View

$$\left[r - \sqrt{\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2}\right]^2 + \left(\frac{z}{r_z}\right)^2 = 1 \qquad (10\text{-}11)$$

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

$$\begin{aligned} x &= r_x(r + \cos\phi)\cos\theta, \qquad -\pi \le \phi \le \pi \\ y &= r_y(r + \cos\phi)\sin\theta, \qquad -\pi \le \theta \le \pi \\ z &= r_z \sin\phi \end{aligned} \qquad (10\text{-}12)$$

Superquadric - superellipse

$$\left(\frac{x}{r_x}\right)^{2/s} + \left(\frac{y}{r_y}\right)^{2/s} = 1$$

$$x = r_x \cos^s \theta, \qquad -\pi \le \theta \le \pi$$

$$y = r_y \sin^s \theta$$

Superquadric - Superellipsoid

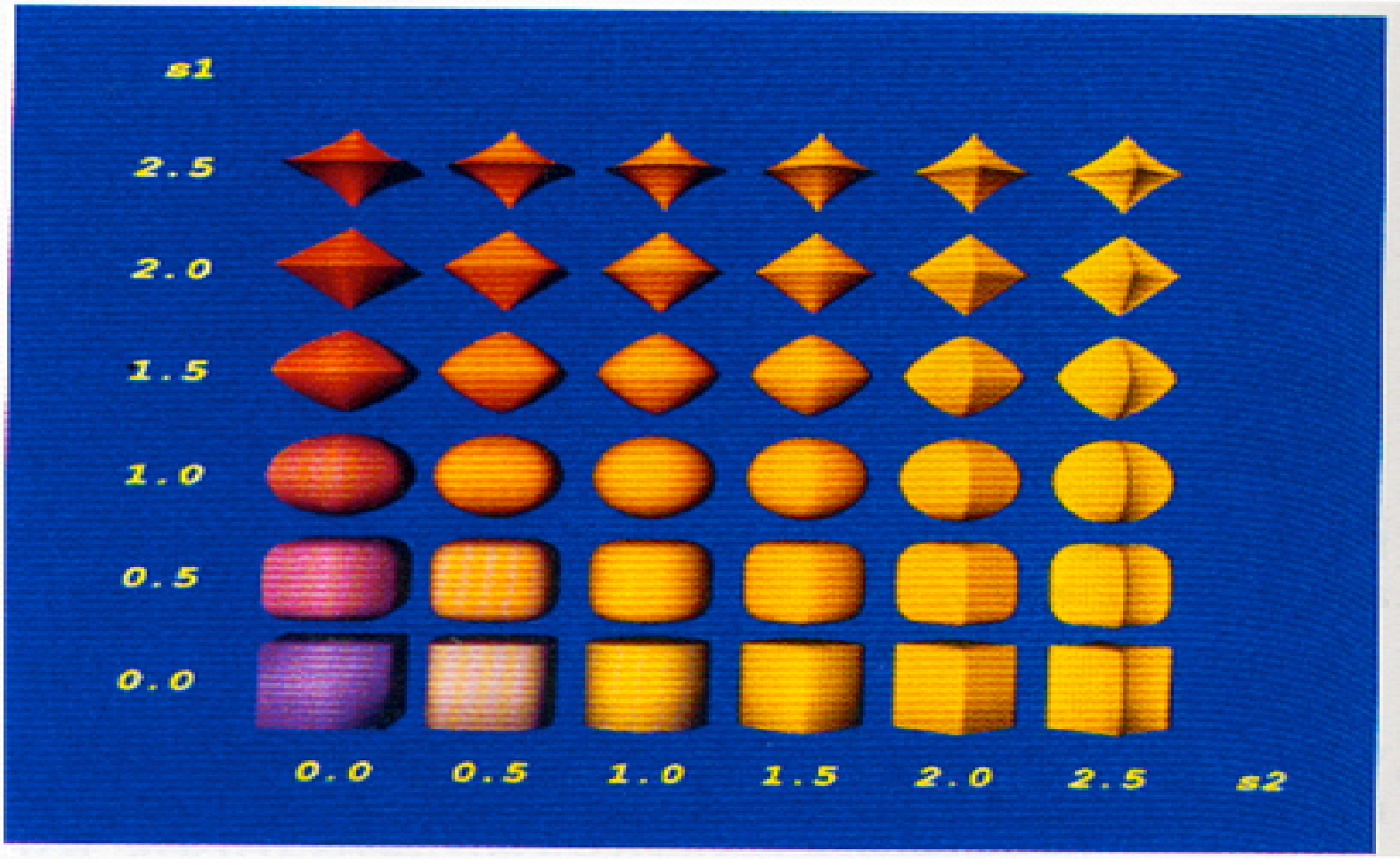
$$\left[\left(\frac{x}{r_x}\right)^{2/s_2} + \left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{z}{r_z}\right)^{2/s_1} = 1$$

$$x = r_x \cos^{s_1}\phi \cos^{s_2}\theta, \qquad -\pi/2 \le \phi \le \pi/2$$

$$y = r_y \cos^{s_1}\phi \sin^{s_2}\theta, \qquad -\pi \le \theta \le \pi$$

$$z = r_z \sin^{s_1}\phi$$

Superellipsoids



Blobby objects

Objects with changing surface shapes, e.g.

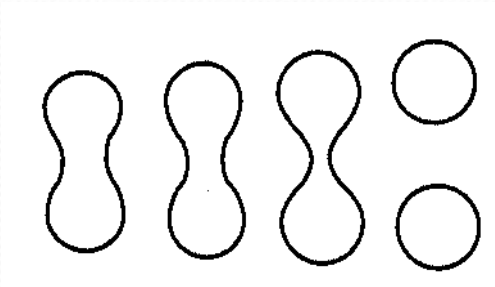
- in certain motions
- in contact with other objects

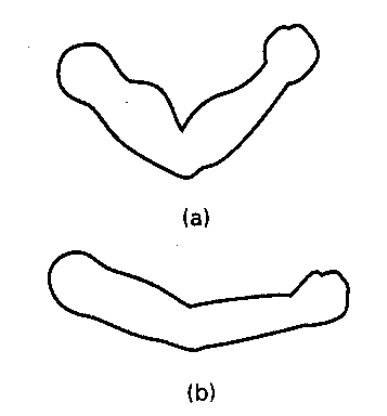
Shape is not fixed, for instance, water droplets and melting objects

Also, various bumps and dents are often used

Usual principle

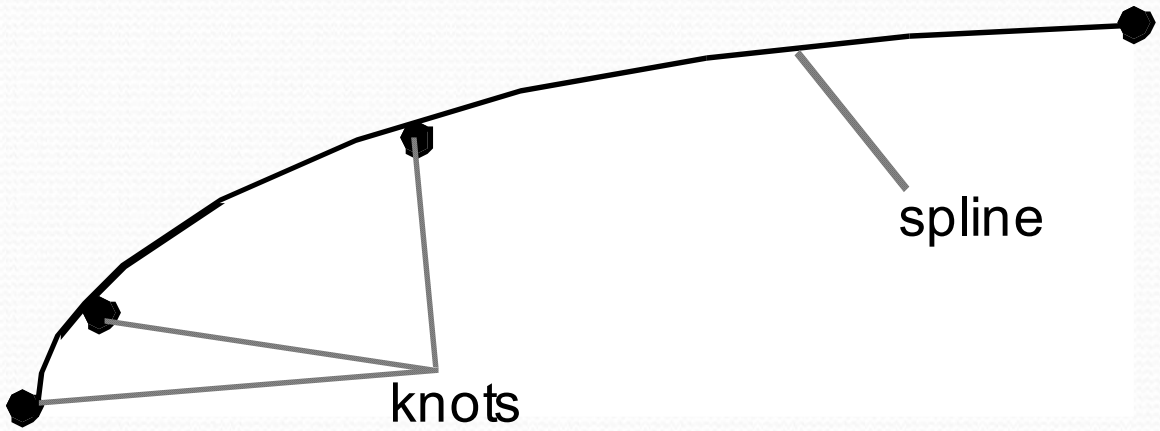
- Fixed volume while shape is changed, e.g. molecules moving apart from each other and human muscles





Splines

- Spline curve – described as a piecewise cubic polynomial function whose 1st and 2nd derivatives are continuous across the various curve sections.
- Used in graphics to design curve and surface shapes, to specify animation path for objects, CAD applications for design of automobile parts.



Interpolation and Approximation Splines

- Control Points – Set of coordinate points that specifies the spline (indicates the general shape of the curve)
- These points are fitted with polynomial functions in one of 2 ways
  - Interpolation- curve passes thro each control point
  - Approximation – does not pass thro any control point.

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

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

Spline Manipulation

- Initial curve is designed and then manipulated using control points.
- Convex Hull – Convex polygon boundary that encloses a set of control points

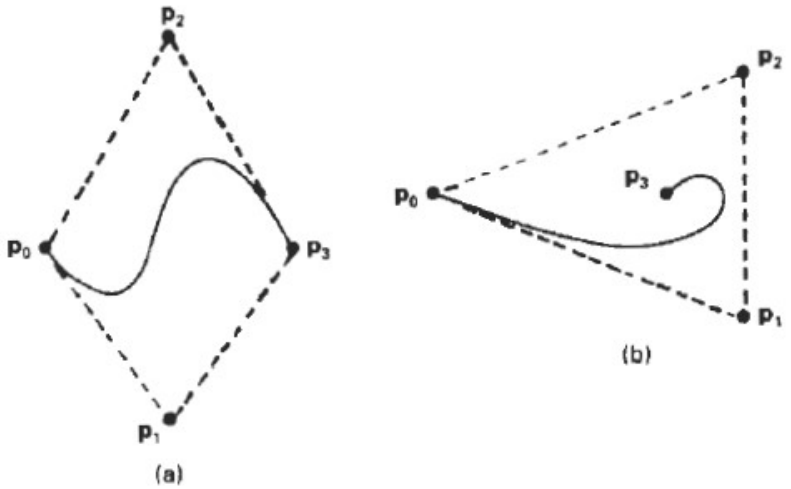


Figure 10-22
Convex-hull shapes (dashed lines) for two sets of control points.

Spline Manipulation

- Control Graph – Polyline connecting a sequence of control points

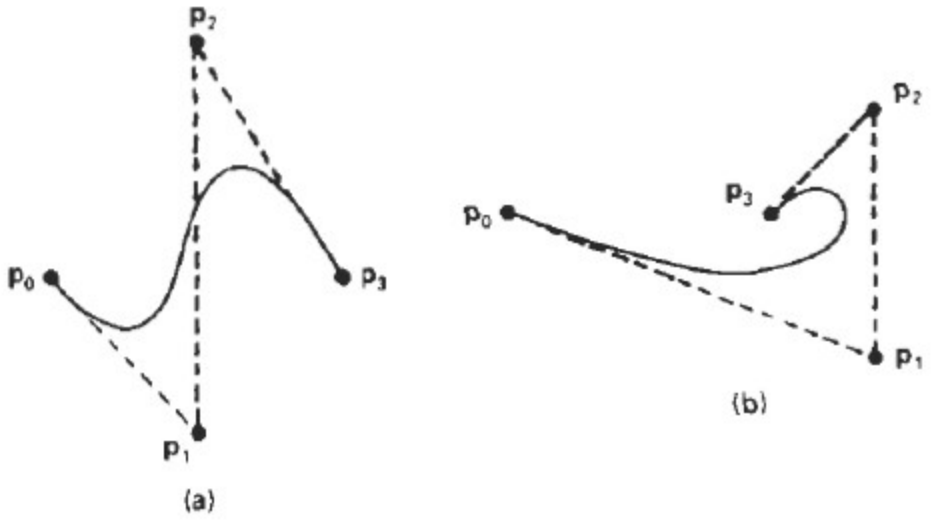


Figure 10-23
Control-graph shapes (dashed lines) for two different sets of control points.

Continuity

- To ensure smooth transition between curve sections
- Each section of the spline is described with a a set of parametric coordinate functions
  - $X = x(u) \quad y = y(u)\ z = z(u)$
- We set continuity by matching the parametric derivates of adjoining curve sections
- Parametric continuity Cx
  - Only P is continuous: C_0
    - Positional continuity – i.e values of x,y,z evaluated at u_2 of first section = values of x,y,z evaluated at u_1 of next section
  - P and first derivative dP/du are continuous: C_1
    - Tangential continuity
  - P + first + second: C_2
    - Curvature continuity

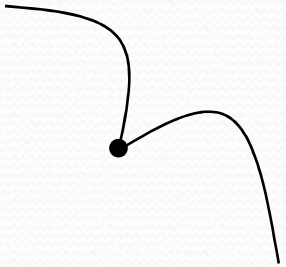
Continuity

- Geometric continuity Gx
  - Parametric derivates should be proportional.
- Zero Order Go
  - Two sections must have the same coordinate position at boundary point
- First Order G1
- Second order G2

Figure 10-25
Three control points fitted with two curve sections joined with (a) parametric continuity and (b) geometric continuity, where the tangent vector of curve C_3 at point $\mathbf{p}_1$ has a greater magnitude than the tangent vector of curve C_1 at $\mathbf{p}_1$.

Order of continuity

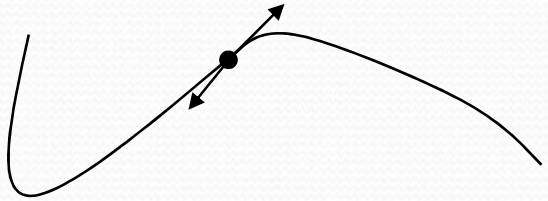
$G^{(0)}$



(a)

$C^{(0)}$ 0^{th} order continuity

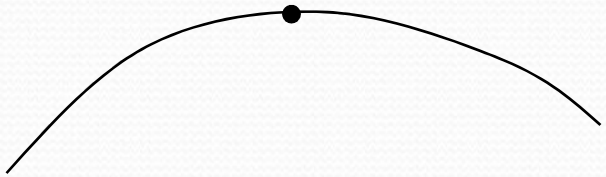
$G^{(1)}$



(b)

$C^{(1)}$ 1^{st} order continuity

$G^{(2)}$



(c)

$C^{(2)}$ 2^{nd} order continuity

Spline Specifications

- 3 Methods for specifying spline representation
  - State set of boundary conditions imposed on the spline
  - State the matrix of the spline
  - State the set of blending functions that determine how specified geometric constraints are combined to calculate positions along the curve path.
  - Suppose, the x coordinate of a spline section has
    - $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 <= u <= 1$
  - Boundary conditions may be set on endpoints x(0) and x(1)
  - Using that determine a_x, b_x, c_x, d_x.

- From the boundary conditions, obtain the matrix

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

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

We can write the boundary conditions in the matrix form

$$\mathbf{C} = \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}}$$

Where M geom is a four element col. Matrix containing the geometric constrints

- So we can say,

$$x(u) = \mathbf{U} \cdot \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}}$$

We can expand this to obtain a polynomial representation for Coordinate x in terms of geometric constraint parameters.

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

Constructive Solid Geometry (CSG)

A new object is constructed by using the following set operations

- *union*
- *intersection*
- *difference*

between any pair of objects

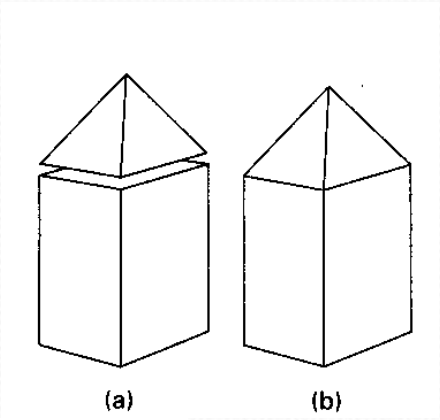
Basic 3D objects

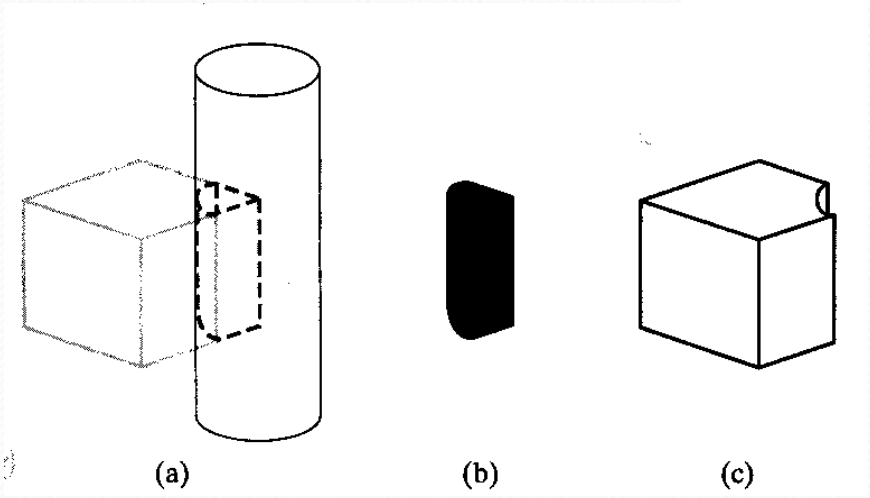
Typical 3D primitives (predefined or modelled):

- block (box, parallelepiped)
- cylinder
- cone
- sphere
- various closed spline surfaces

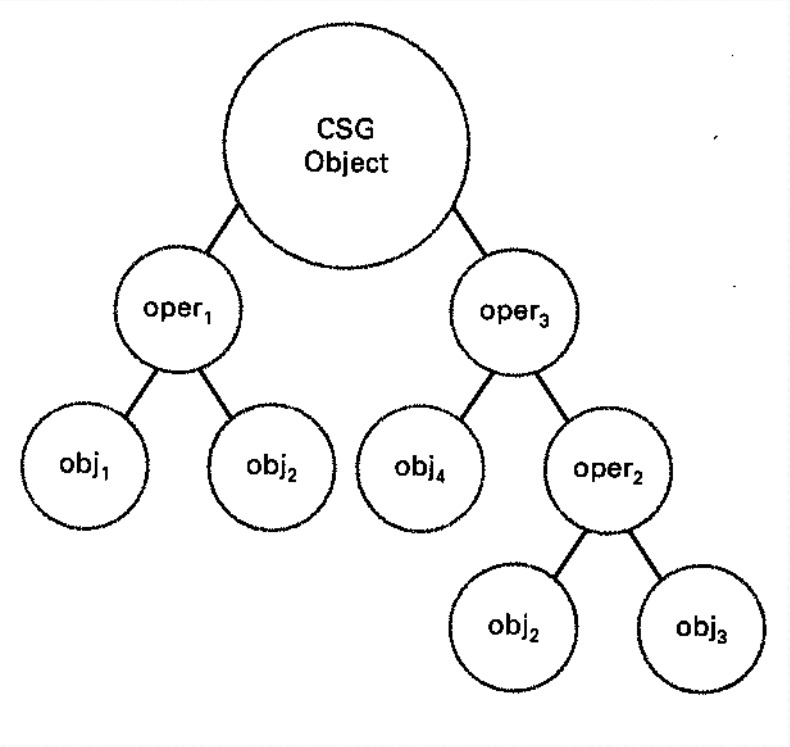
Operation examples

Select any two objects
and a set operation
=> a new object





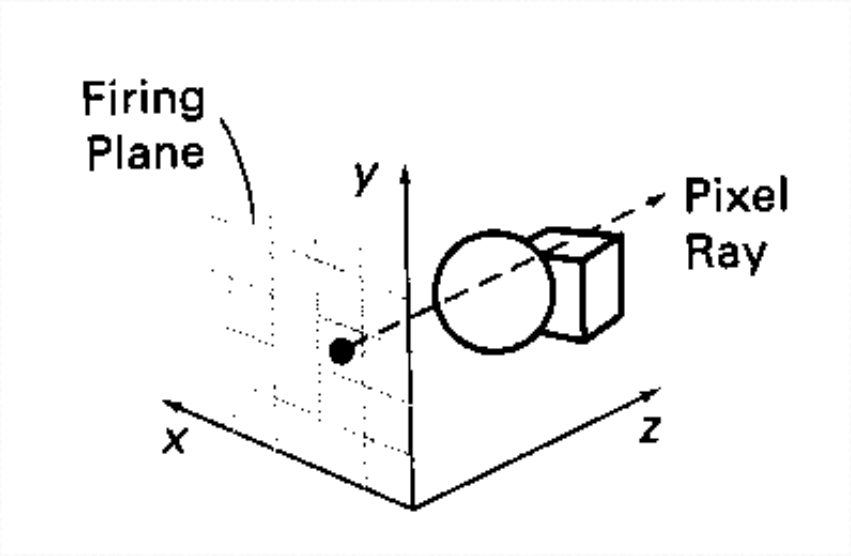
CSG - Binary tree representation



Ray-casting

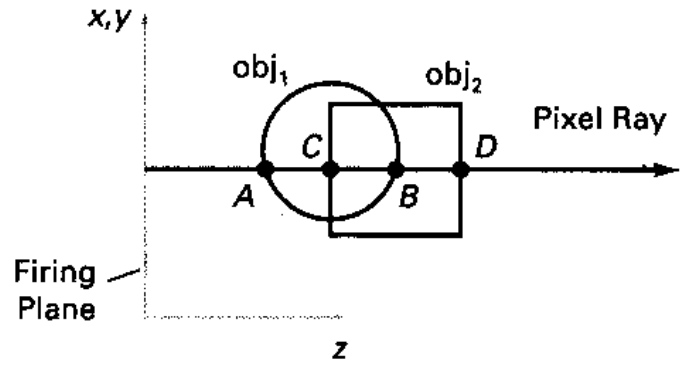
A technique to implement CSG operations:

- an xy-plane (cp the pixel plane) in the display, called *firing plane*, is defined
- parallel rays from the pixel positions are fired in the z-direction



Ray-casting,cont'd

- intersection points with object surfaces are calculated
- these points are sorted on increasing distances from the firing plane
- surface limits are then determined by the current set operation



(a)

Operation	Surface Limits
Union	A, D
Intersection	C, B
Difference ($obj_2 - obj_1$)	B, D

(b)

Octrees

A simple technique to represent also the interior structure of a 3D object

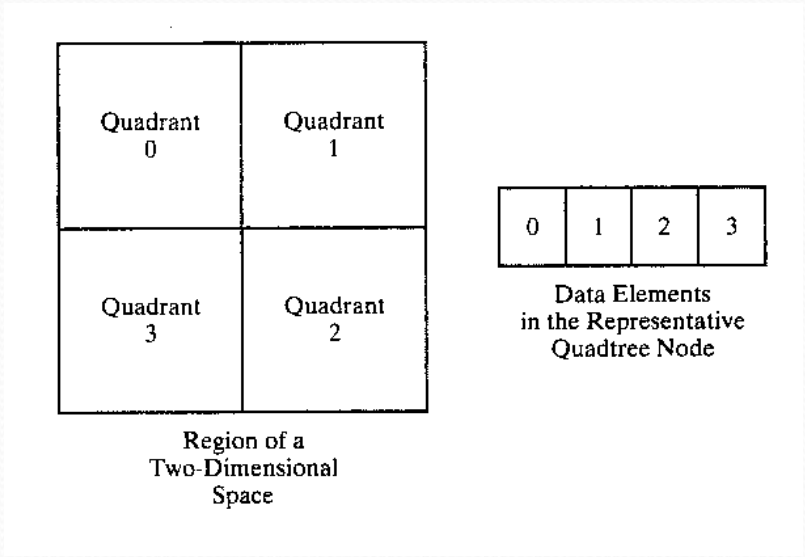
Each node of the octree represents a particular 3D section of an object; coherence will be used to save memory

Octrees - an extension of *quadtrees* from 2D, but quadtrees are easier to illustrate, although still based on the same principle

Quadtrees

Assume that the area to represent is quadratic

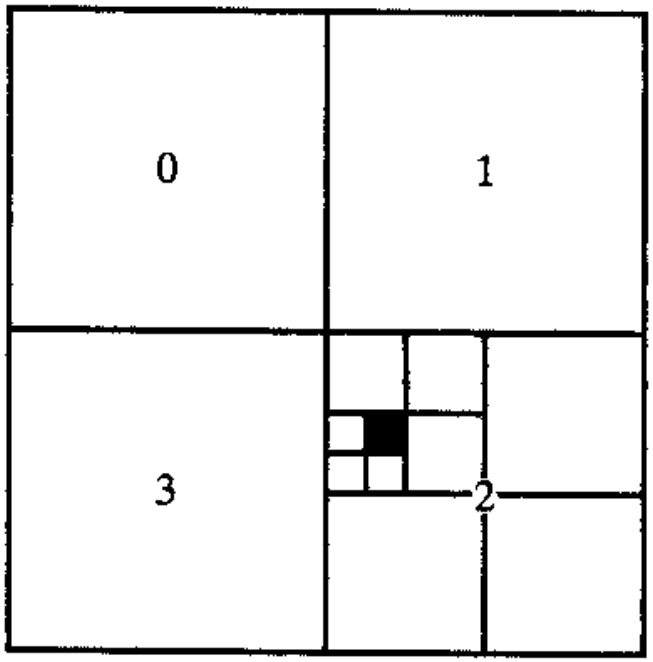
- a quadtree is created by successively dividing the given area into four quadrants
- each node of the tree has four elements; one for each quadrant

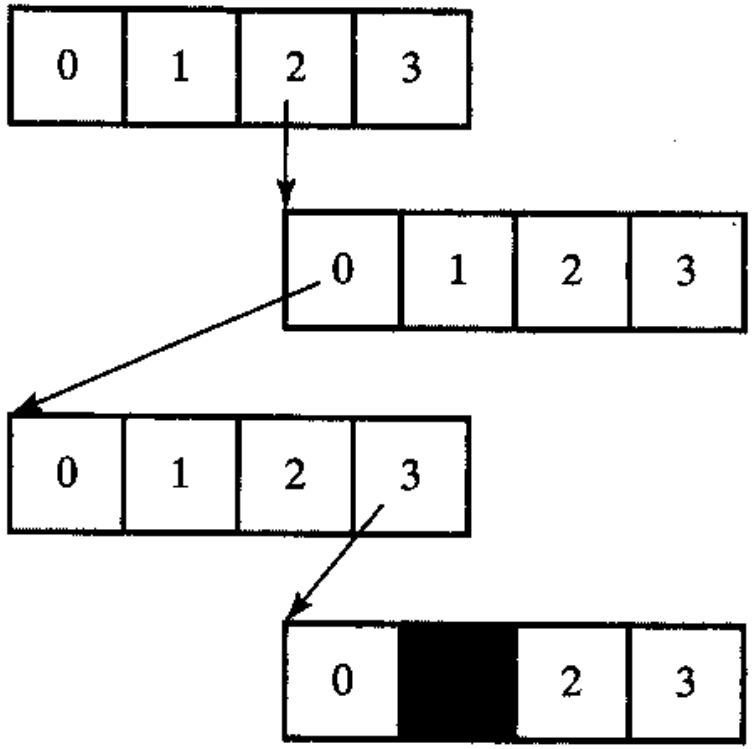


Quadtrees, cont'd

- if a homogeneous quadrant (same intensity on all pixels in it), the intensity is stored in the corresponding data element of the node, and a flag is set to indicate a homogeneous quadrant
- if a heterogeneous quadrant, it is divided into 4 subquadrants, the flag is set to indicate a heterogeneous quadrant, and instead of the intensity, a pointer to the next node in the quadtree is stored

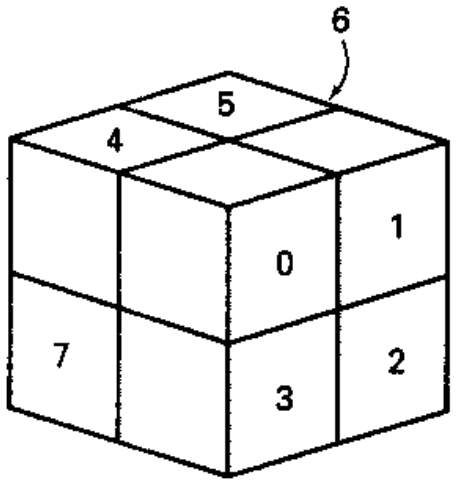
Quadtree - example



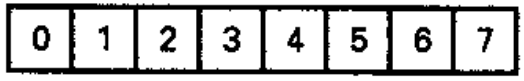


Octree

A 3D space (assumed to be a cube) is divided into eight octants and stores eight data elements in each node of the tree



Region of a Three-Dimensional Space



Data Elements in the Representative Octree Node

Octrees, cont'd

- Individual elements in a 3D space are called volume elements, *voxels*, in which other properties than intensity such as material and density, can be included
- When all voxels in an octant have the same intensity (homogeneous), this intensity is stored in the corresponding data element of the node
- Each heterogeneous octant is divided into 8 new suboctants with the corresponding data element pointing at the next node of the tree

Comments on octrees

Empty regions are represented by ”void”

If the current space is not a cube, it can easily be enclosed by one

To locate the visible parts of an object, first possible front octants are examined; if visible, the hidden octants don’t need to be examined (cp VSD)

BSP trees

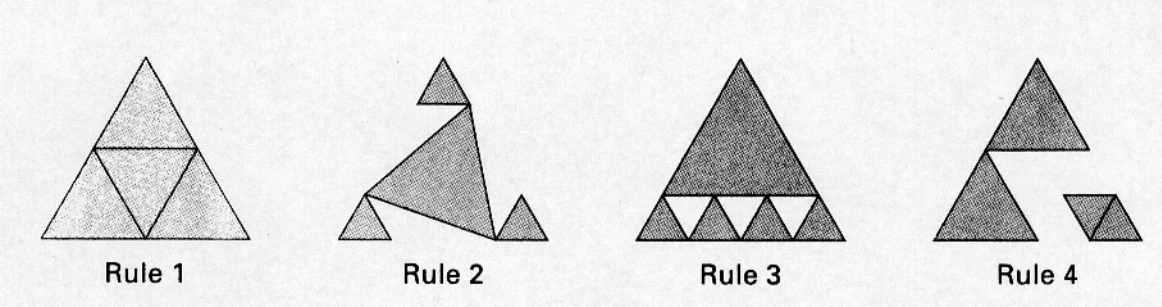
BSP - *Binary Space Partitioning*

Similar to octrees, but successive subdivision into two partitions in each step => a binary tree representation

In each subdivision, the scene is divided into two parts by a plane that can be defined anywhere and with arbitrary orientation

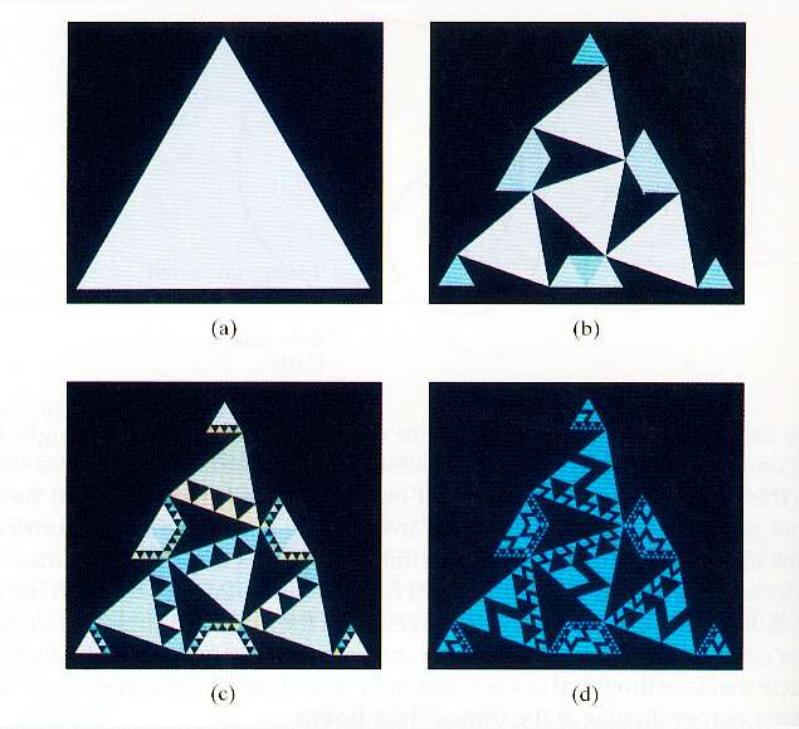
Shape Grammars

- Sets of production rules that can be used on an initial object to add details based on certain rules
- Transformations can be used to
  - change the shape of an object
  - add surface details/textures of an object



Grammars,cont'd

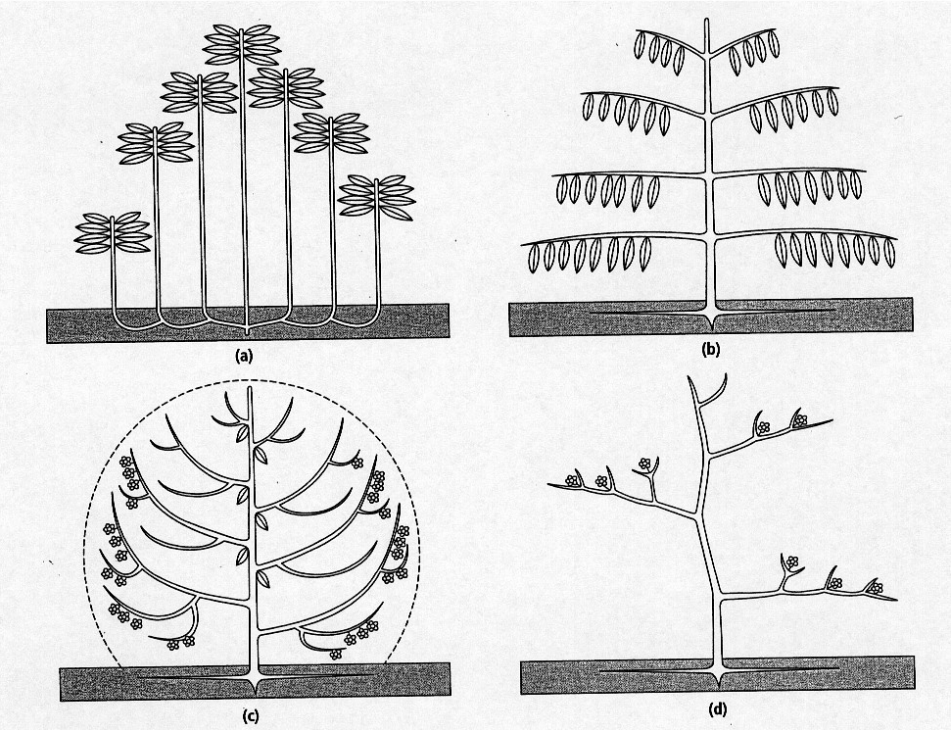
- Use the set of production rules at each step to change a given initial object
- Transformations corresponding t the rules can be generated automatically based on a picture with a special production-rule editor
- Each rule can be described by giv the initial and final shapes

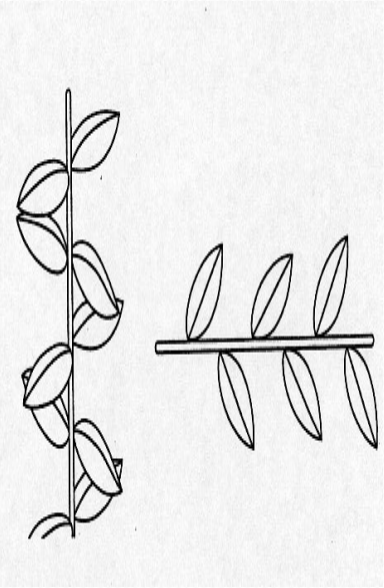


L-Grammars, Graftals

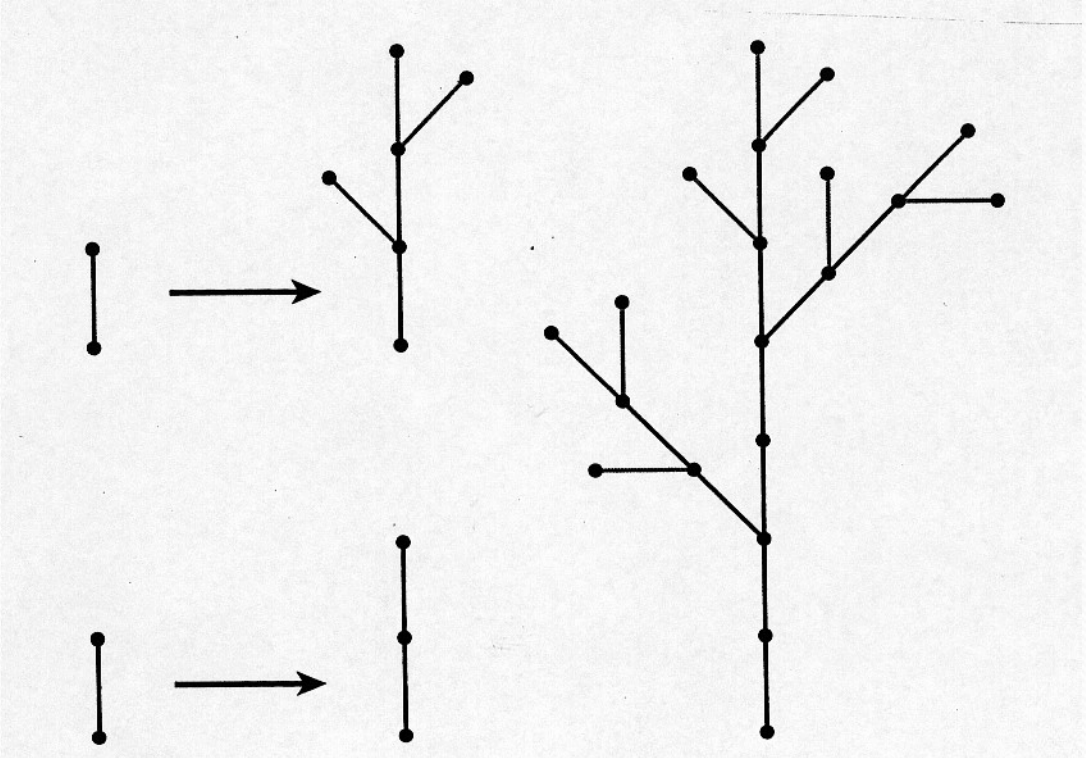
- Used to generate plants, trees and similar objects
- For instance, a tree:
  - a trunk with branches and leaves
  - can be modified with rules for
    - particular connections of the branches
    - leaves on individual branches
    - placing in suitable positions
- There are special plant-generator packages/editors

Graftals examples





Graftal rules,example



Ex with Plant Editor

Physically Based Modeling

Non-rigid objects can be represented by methods that describe the behaviour as a mix of external and internal forces

Typical objects are ropes, cloth material, softballs.

A strict description consider, for example:

* how a stiff object effects a soft material, i.e. textile material

* the interaction between the threads in the textile

Textile on chair

Cloth examples

Models of cotton, wool and polyester using energy-function minimization

In animation

Physically based modeling also improves the realism in animation providing a better description of motion paths.

Instead of spline paths and kinematics, dynamical equations with forces and accelerations give much more realistic motions.

Final example

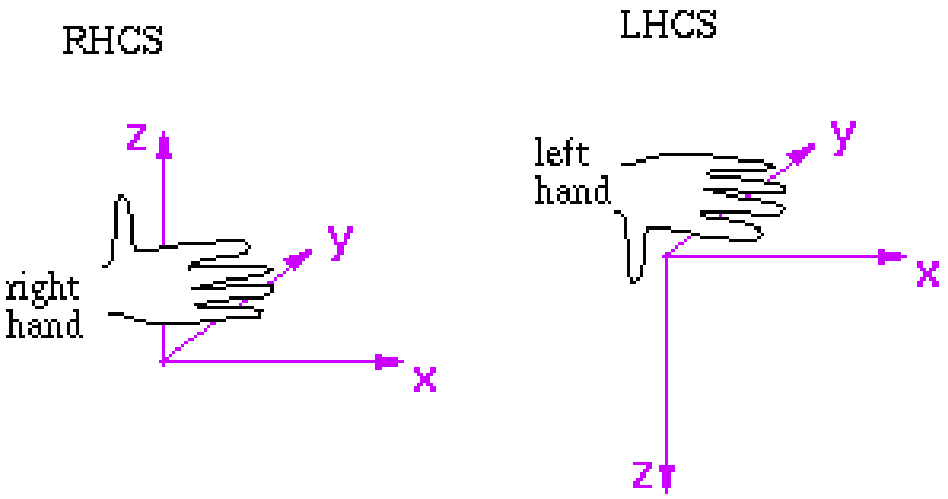
3D TRANSFORMATIONS

3D transformations

- Instead of creating every new object from scratch, we may transform some existing objects
- We also need to transform objects from one space to another, e.g., world space to view space/camera space
- Instead of performing each transformation alone, we accumulate the transformations in a matrix before applying them to objects

3D transformations

- Transformations are measured in a coordinate matrix
- There are two possible ways of specifying the Z-axis, which gives rise to a **left-handed(**suitable for screens) or a **right-handed** system(consisent with math)



3D Point

- We will consider points as column vectors. Thus, a typical point with coordinates (x, y, z) is represented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3D Point Homogenous Coordinate

- A 3D point **P** is represented in homogeneous coordinates by a 4-dim. Vect:

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

2D Transformation

- Scaling
  - X’ = aX
  - Y’ = aY
- Translation
  - X’=x+b
  - Y’=y+b
- Rotation

  x' = r cos (θ + ϕ) =

  r cos θ cos ϕ - r sin θ sin ϕ

  y' = r sin (θ + ϕ) =

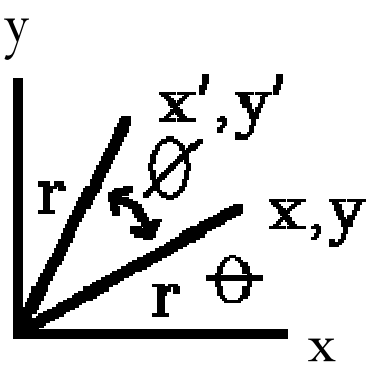
  r sin θ cosϕ + r cos θ sin ϕ

so

x' = x cos ϕ - y sin ϕ

y' = y cos ϕ + x sin ϕ

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \cos\varphi & \sin\varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



$$x = r\cos\theta$$

$$y = r\sin\theta$$

Homogeneous Coordinates

- In 2D, three elements can be used to represent a point, e.g., (x,y,1) to represent (x,y) or (xh,yh,h) h≠0 to represent (x,y)
- (x,y,h) is a homogeneous coordinate representing the point (x/h, y/h).
- The 3D homogenous representation is given by

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

2D Transformation Matrices

- Translation matrix:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Scaling matrix:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

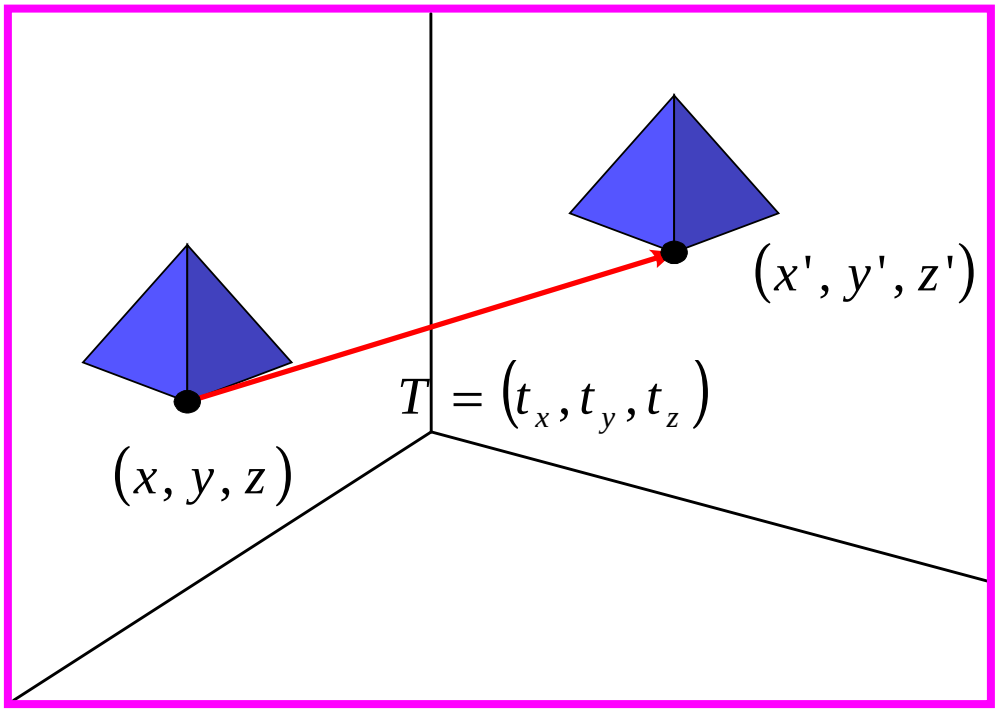
- Rotation matrix:

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

3D Translation

- **P** is **translated** to **P'** by:

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



3D Transformation- Translation

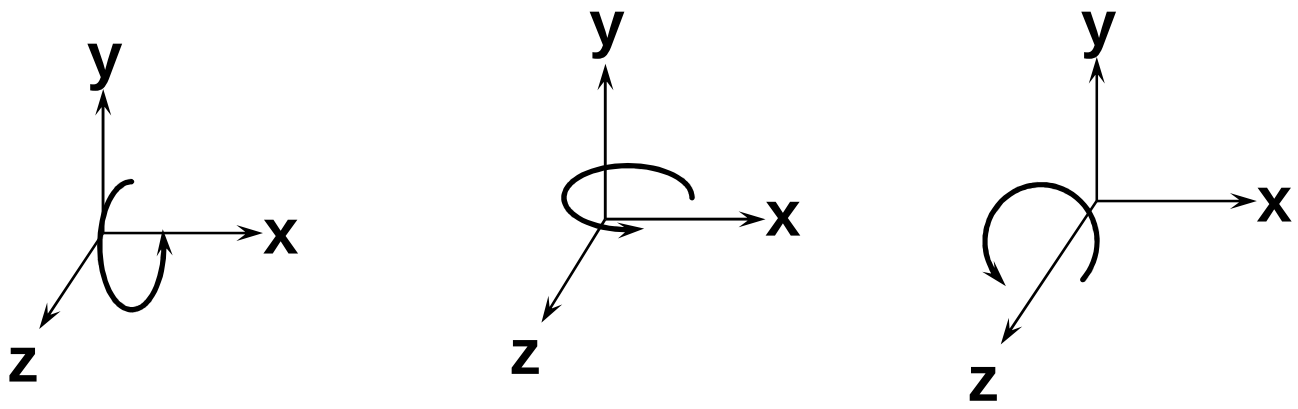
- The matrix representation is equal to three equations:
- $x' = x + t_x \; y' = y + t_y \; z' = z + t_z$

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

- For polygon surfaces, translate each vertex of the surface and redraw the polygon facets in new position

3D Rotation

- In 2D, rotation is about a point
- In 3D, rotation is about a vector, which can be done through rotations about x, y or z axes
- Positive rotations are anti-clockwise, negative rotations are clockwise, when looking down a positive axis towards the origin

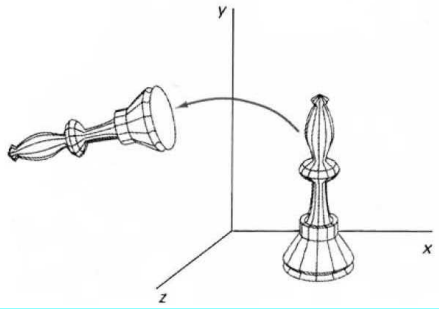


Coordinate Axis Rotations

- **Z-axis rotation:** For z axis same as 2D rotation:

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

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

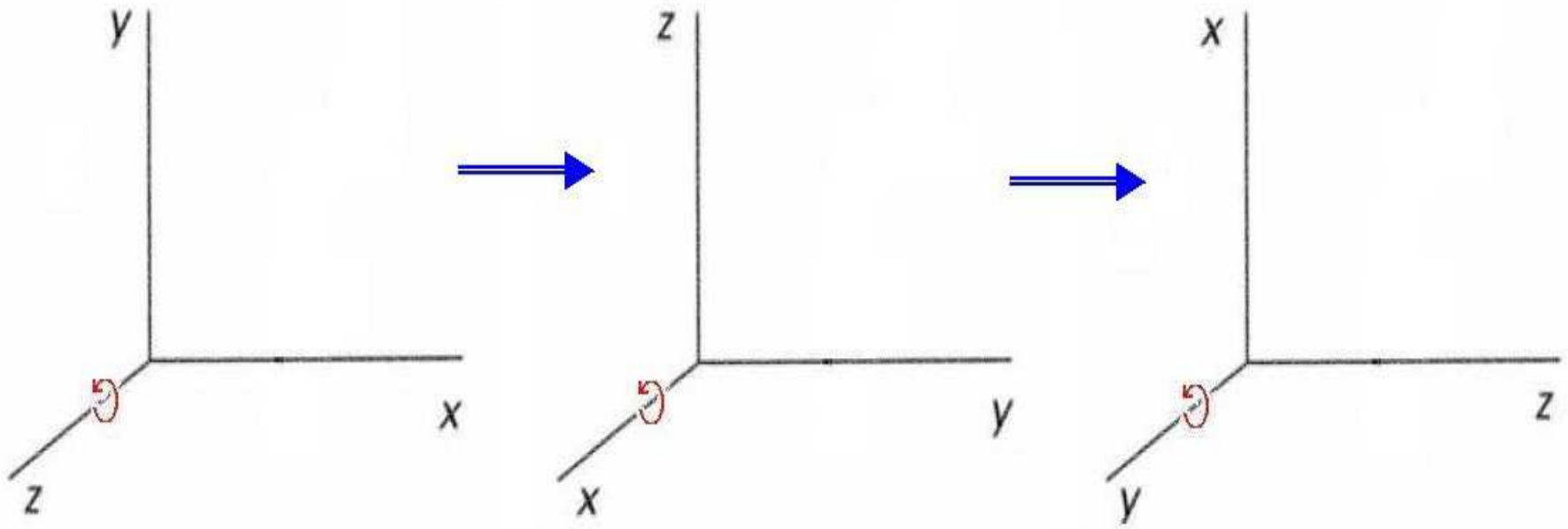


3D Rotation

- The 2D equations are easily extended to 3D z-axis rotation
  - $x' = x \cos \Theta - y \sin \Theta$
  - $y' = x \sin \Theta + y \cos \Theta$
  - $z' = z$
  - $P' = R_z(\Theta).\ P$
- Transformations equations for rotations about other two axes can be obtained with cyclic permutation of the coordinate parameters x ,y and z
- $x \rightarrow y \rightarrow z \rightarrow x$

Coordinate Axis Rotations

- Obtain rotations around other axes through cyclic permutation of coordinate parameters:

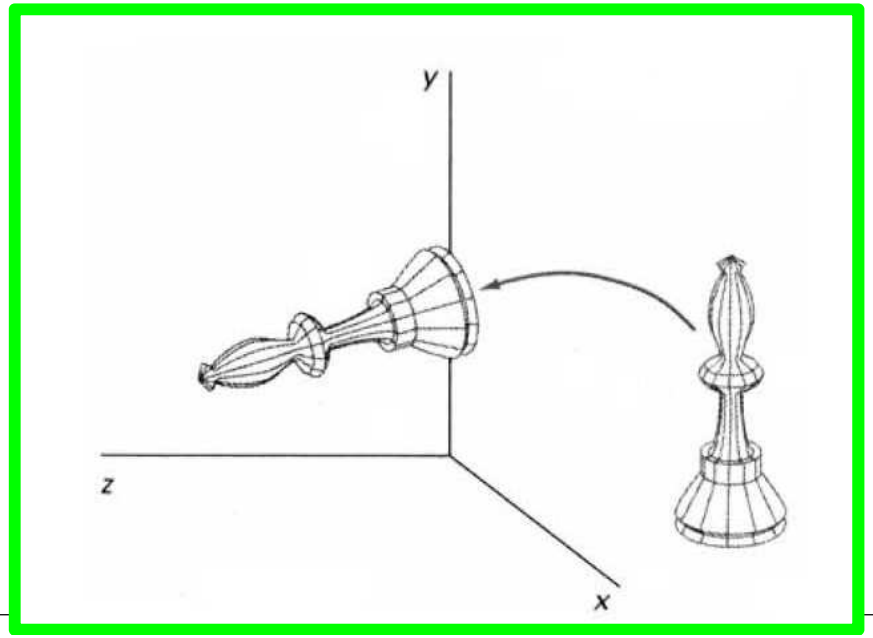


Coordinate Axis Rotations

- **X-axis rotation:**

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

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

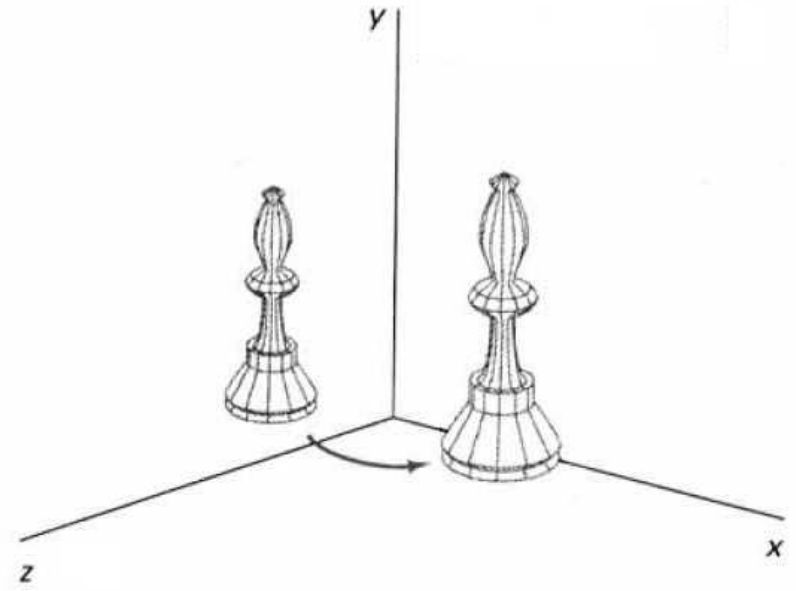


Coordinate Axis Rotations

- Y-axis rotation:

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

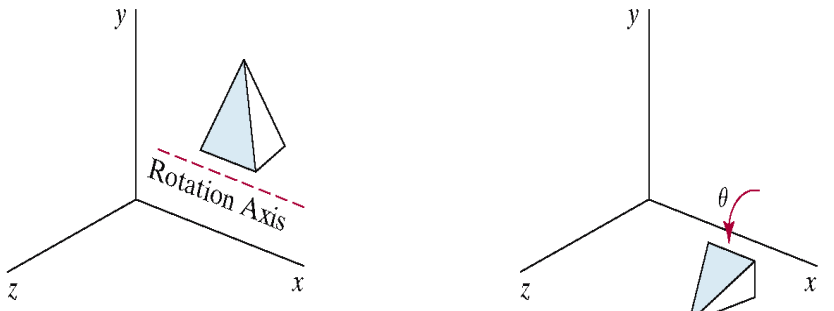
$$\mathbf{P}' = \mathbf{R}_y(\theta) \cdot \mathbf{P}$$



General Three Dimensional Rotations

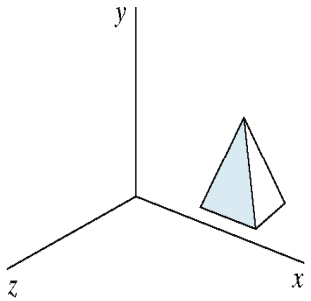
General Three Dimensional Rotations

Rotation axis parallel with coordinate axis (Example x axis):

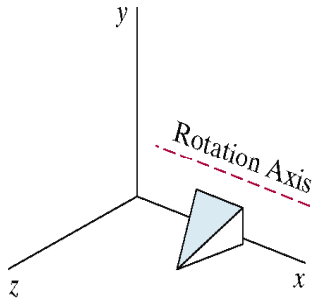


(a)
Original Position of Object

(c)
Rotate Object Through Angle θ



(b)
Translate Rotation Axis onto x Axis



(d)
Translate Rotation
Axis to Original Position

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

General Three Dimensional Rotations

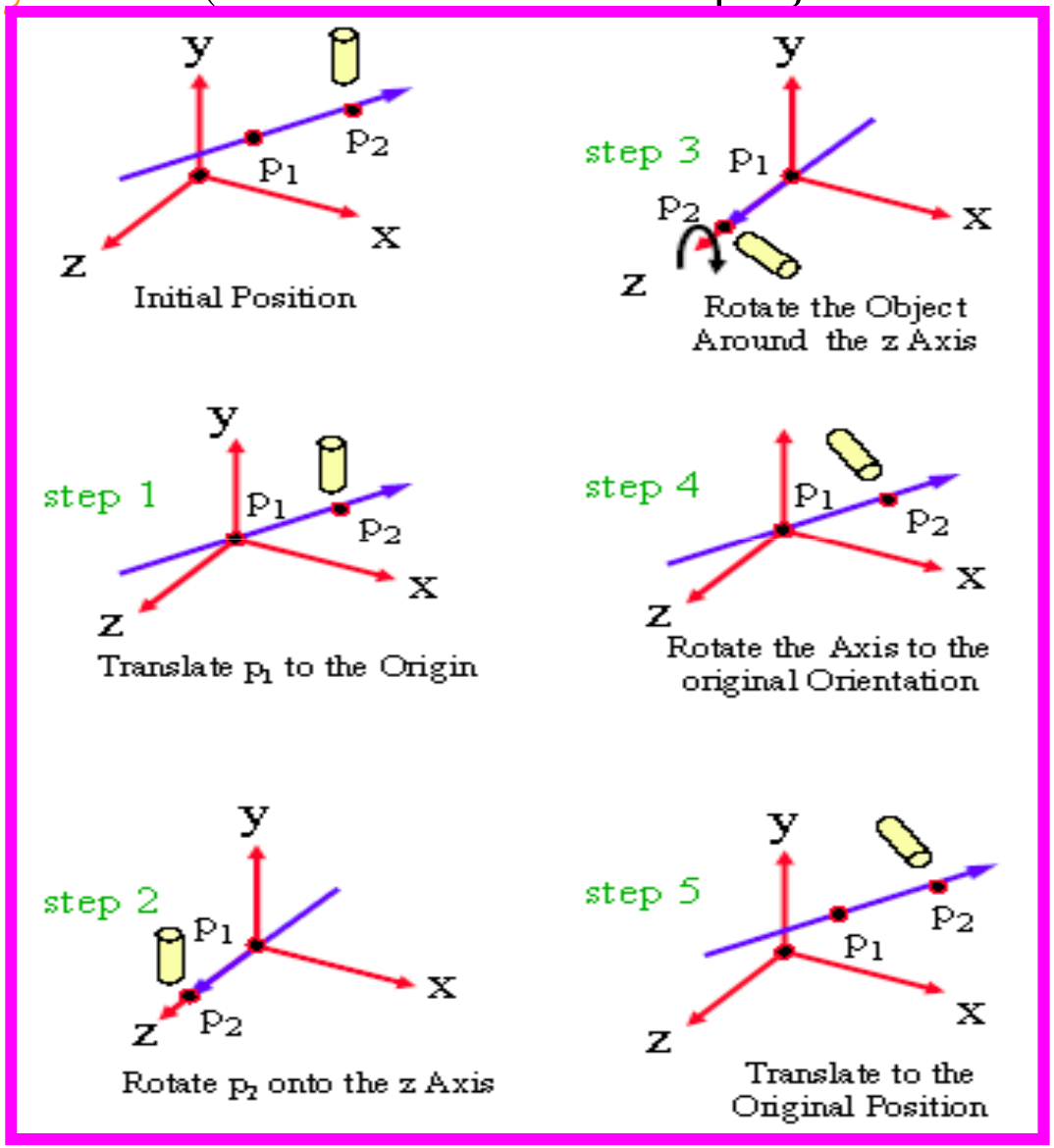
$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}$$

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

1. Translate the object so that the rotation axis passes through the coordinate origin
2. Rotate the object so that the axis rotation coincides with one of the coordinate axes
3. Perform the specified rotation about that coordinate axis
4. Apply inverse rotation axis back to its original orientation
5. Apply the inverse translation to bring the rotation axis back to its original position

General Three Dimensional Rotations

An arbitrary axis (with the rotation axis projected onto the Z axis):



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

Rotations for an arbitrary axis

- Steps:
- 1. Normalize vector *u*
- 2. Compute α
- 3. Compute β
- 4. Create rotation matrix

Rotations for an arbitrary axis

The rotation axis is defined by two points, then the direction of rotation is to be counterclockwise along the axis from p2 to p1

An axis vector defined by the two points

$$\mathbf{V}=\mathbf{P}_2-\mathbf{P}_1=(x_2-x_1,y_2-y_1,z_2-z_1)$$

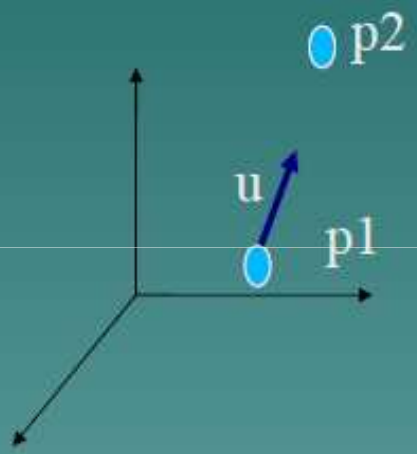
u is the unit vector along **V**:

$$\mathbf{u}=\frac{\mathbf{V}}{|\mathbf{V}|}=(a,b,c)$$

Rotations for an arbitrary axis

- First step: Translate $\mathbf{P}_1$ to origin:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- Next step: Align $\mathbf{u}$ with the z axis

we need two rotations:

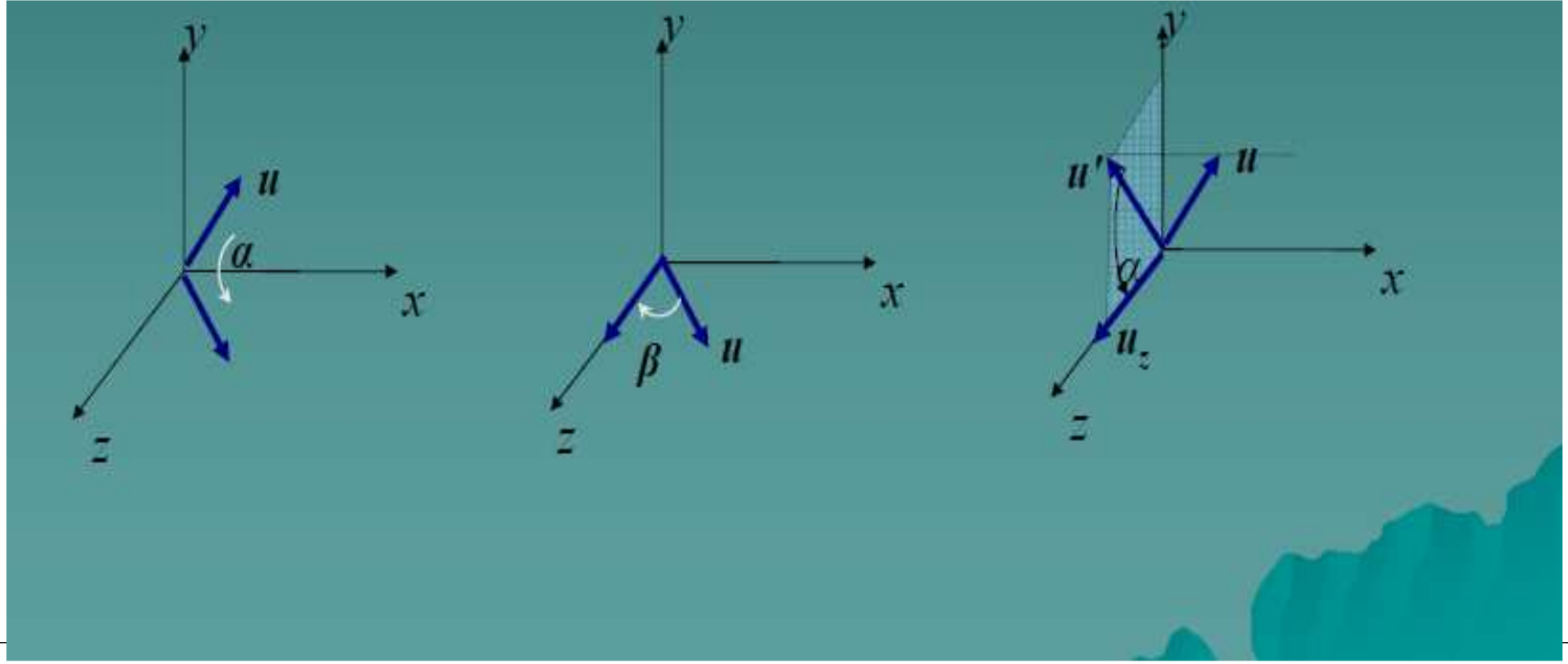
- rotate around x axis to get $\mathbf{u}$ onto the xz plane,
- rotate around y axis to get $\mathbf{u}$ aligned with z axis.

Rotations for an arbitrary axis

Align **u** with the z axis

1) rotate around x axis to get **u** into the xz plane,

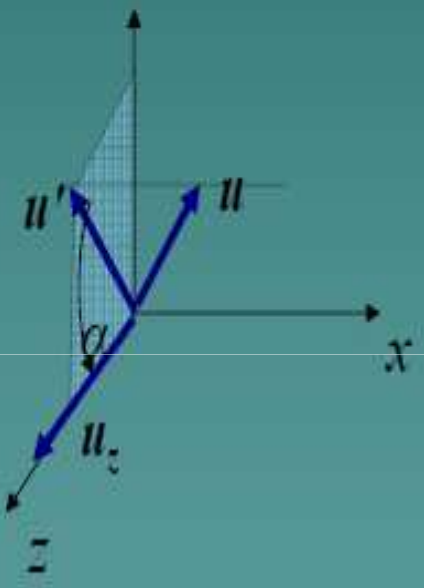
2) rotate around y axis to get **u** aligned with the z axis



Dot product and cross product

- Since rotation calculations involves sine and cosine functions standard vector operations are used to obtain the elements of rotation matrices
- The dot product allows to determine cosine terms:
  - $V \cdot U = |V|\,|U| \cos \Theta$
- The cross product determine sine term
- |v|*|u|*sin(a),

Rotations for an arbitrary axis



We need cosine and sine of α for rotation

$$\cos\alpha = \frac{\mathbf{u}' \cdot \mathbf{u}_z}{|\mathbf{u}'||\mathbf{u}_z|} = \frac{c}{d} \qquad d = \sqrt{b^2 + c^2}$$

$$\mathbf{u}' \times \mathbf{u}_z = \mathbf{u}_x |\mathbf{u}'||\mathbf{u}_z| \sin\alpha = \mathbf{u}_x b$$

$$\mathbf{u}' = (0, b, c)$$

$$b = d \sin\alpha$$

Projection of **u** on *yz* plane

$$\cos\alpha = \frac{c}{d} \qquad \sin\alpha = \frac{b}{d}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations for an arbitrary axis

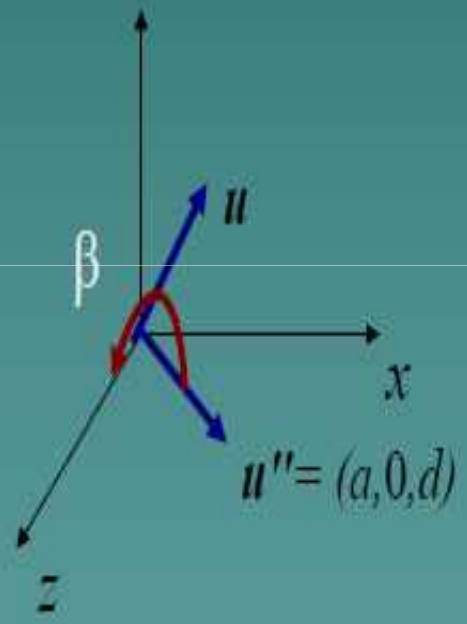
2) rotate around y axis to get $\mathbf{u}$ aligned with the z axis

$$\cos\beta = \frac{u'' \cdot u_z}{|u''| \cdot |u_z|} = d$$

$$\mathbf{u}'' \times \mathbf{u}_z = \mathbf{u}_y |\mathbf{u}''| |\mathbf{u}_z| \sin\beta = \mathbf{u}_y \cdot (-a)$$

$$\cos\beta = d \qquad \sin\beta = -a$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rotations for an arbitrary axis

- Aligned the rotation axis with positive z axis.
- Specified rotation angle given by

$$Rz(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\mathbf{R}(\theta) = \mathbf{T}(x_1, y_1, z_1) \cdot \mathbf{R}_x(-\alpha) \cdot \mathbf{R}_y(-\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}(-x_1, -y_1, -z_1)$$

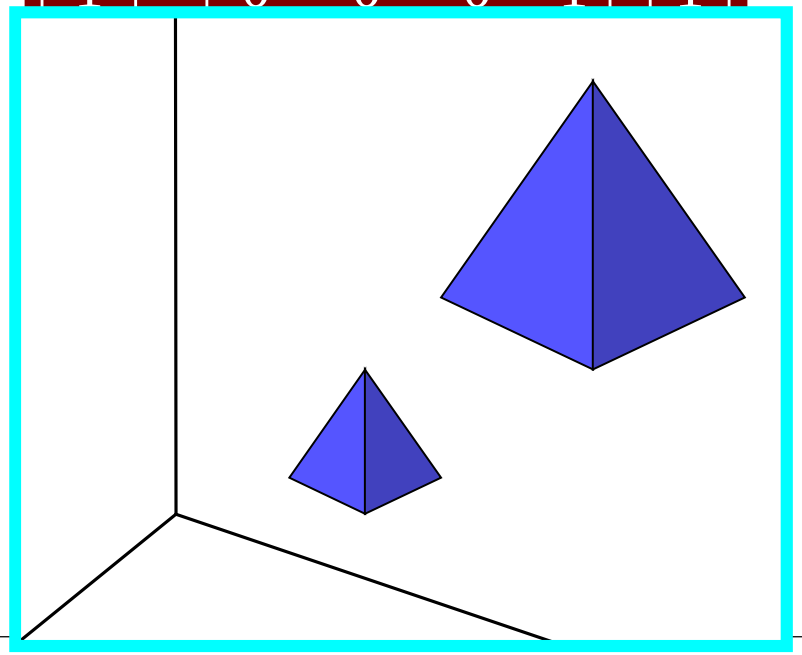
3D Scaling

3D Scaling

- **About origin:** Changes the size of the object and repositions the object relative to the coordinate origin.

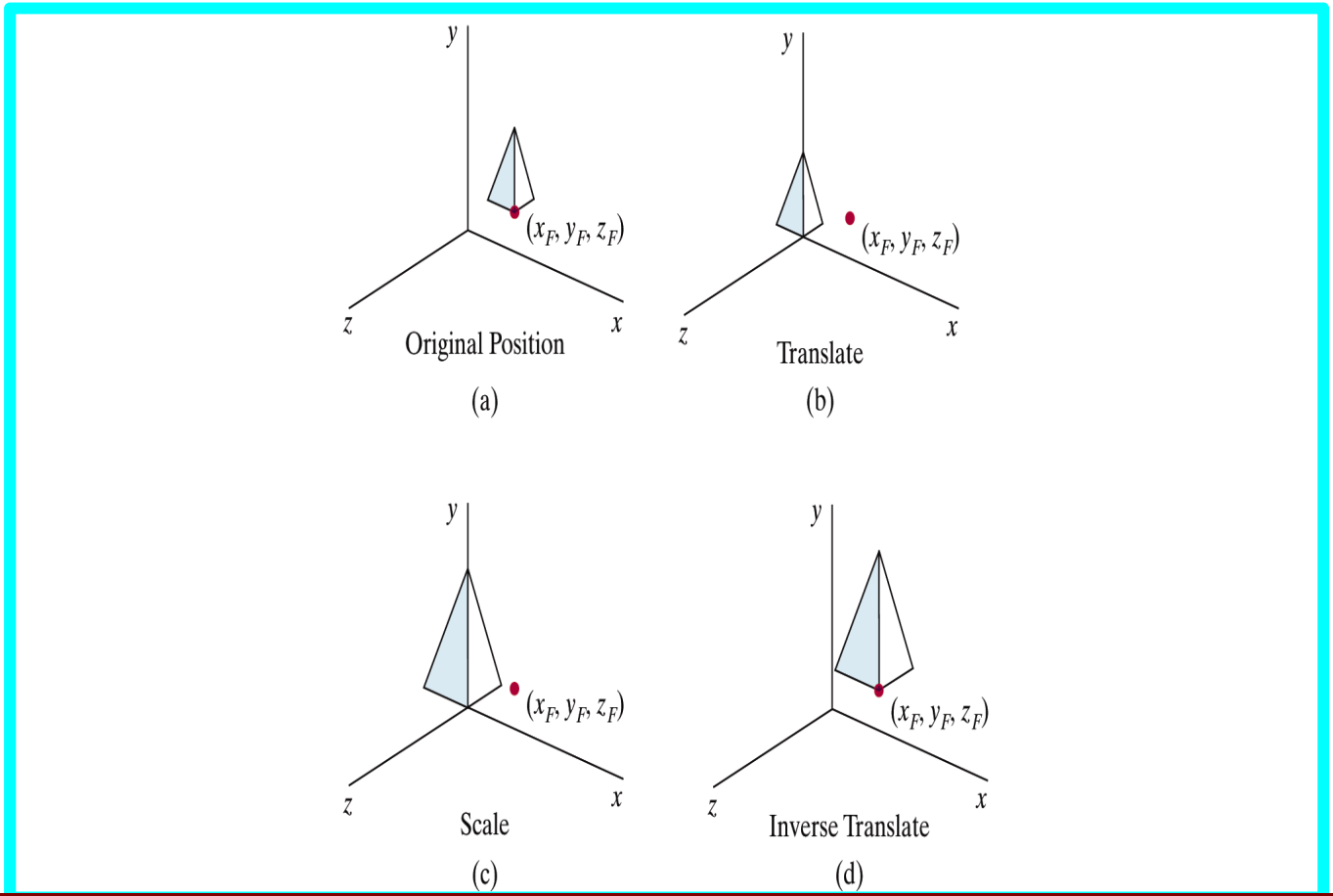
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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



3D Scaling

- About any fixed point:



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

Composite
3D Transformations

Composite 3D Transformations

- **Same way as in two dimensions:**
  - Multiply matrices
  - Rightmost term in matrix product is the first transformation to be applied

3D Reflections

3D Reflections

- **About an axis:** equivalent to 180°rotation about that axis

About a plane: 3D Reflections

- A reflection through the **xy** plane:

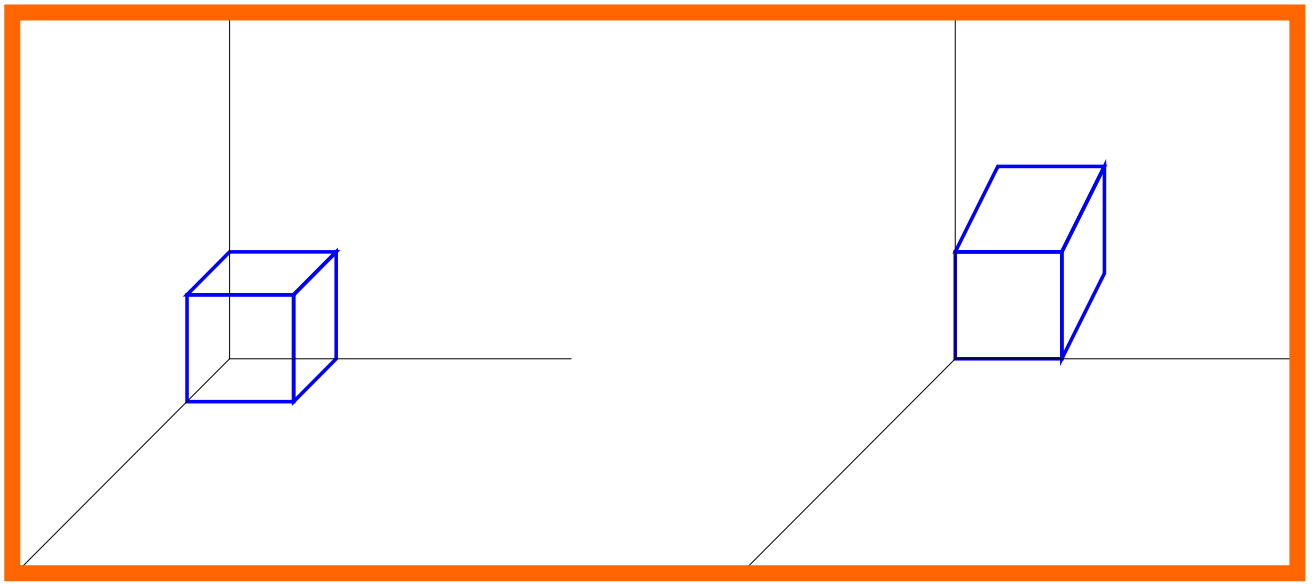
$$\begin{bmatrix} x \\ y \\ -z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- A reflections through the **xz** and the **yz** planes are defined similarly.

3D Shearing

3D Shearing

- **Modify object shapes**
- **Useful for perspective projections:**
  - E.g. draw a cube (3D) on a screen (2D)
  - Alter the values for **x** and **y** by an amount proportional to the distance from z_{ref}



3D Shearing

$$M_{zshear} = \begin{bmatrix} 1 & 0 & sh_{zx} & -sh_{zx} \cdot z_{ref} \\ 0 & 1 & sh_{zy} & -sh_{zy} \cdot z_{ref} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

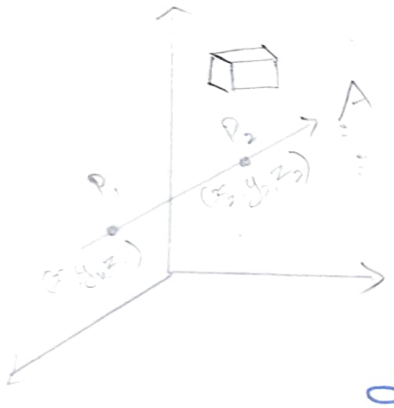
Part - C

(a) Given a cube in 3D space

⇒ Axis : $P_1(x_1, y_1, z_1)$ $P_2(x_2, y_2, z_2)$
→ Not parallel to any of the axis

⇒ Rotate cube about θ about axis P_1P_2

⇒ Rotation along arbitrary axis P_1P_2

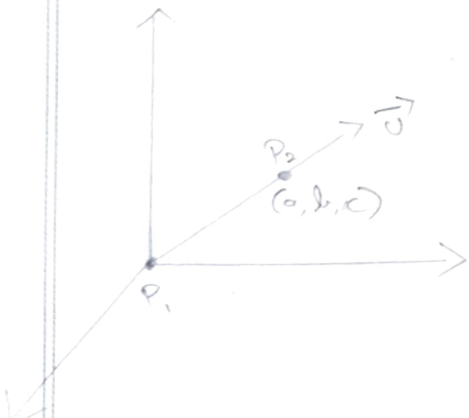


Axis vector, $V = P_2 - P_1$
$= (x_2 - x_1, y_2 - y_1, z_2 - z_1)$

Normal (unit) vector of axis $= U = \frac{V}{|V|}$

Now we are going to do composite transformation

Step 1 : Move P_1 to origin



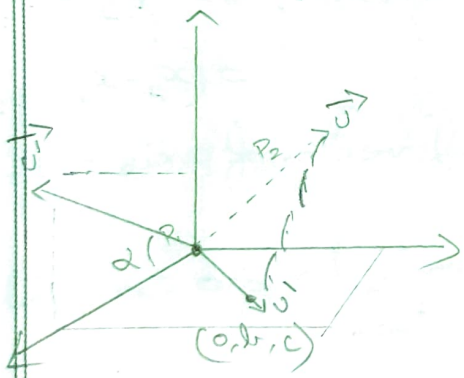
$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2: Move the axis vector along z axis

Here we need 2 transformation

a) Rotate anticlockwise in x axis by α to bring $\vec{v}$ to xz plane

b) Rotate clockwise in y axis by β to bring $\vec{v}$ to z axis



a) Rotate α along x axis

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

α - angle between the projection in yz plane & z axis

$\therefore \alpha \Rightarrow$ angle between $\vec{v}$ & v_2

$$U \cdot V = |U||V| \cos\alpha$$

$$\cos\alpha = \frac{U \cdot V}{|U||V|}$$

$$= \frac{(0, b, c) \cdot (0, 0, c)}{|U||V|}$$

$$U \times V = |U||V| \sin\alpha$$

$$\sin\alpha = \frac{U \times V}{|U||V|}$$

$$= \frac{(0, b, c) \times (0, 0, c)}{|U||V|}$$

$\cos\alpha = \frac{c}{d}$ $\quad \sin\alpha = \frac{b}{d}$

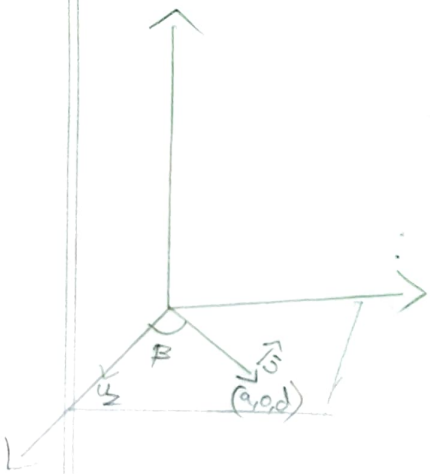
$d = \sqrt{b^2 + c^2}$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Rotate β along y-axis $\quad \theta = -\beta$

$\Rightarrow$ clockwise

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$\cos\beta = d$ $\quad \sin\beta = a$

$$R_y(\beta) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Now rotate the cube along $\vec{v_z}$ (z axis)

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

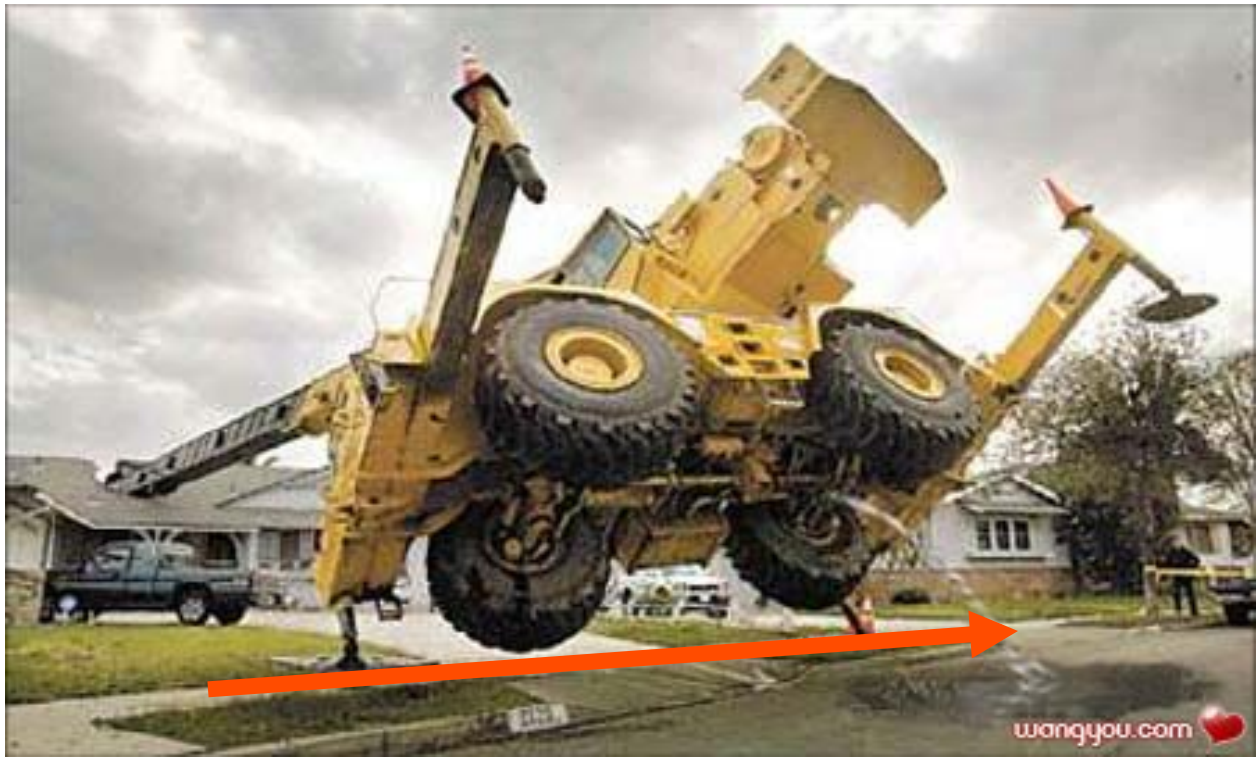
Step 4: Now move back the coordinate to original positio

(ie) Rotate along y in β in anticlockwise
Rotate along x in α in clockwise
~~Final Trans~~ ~~Rotat~~ Translate P_1 back

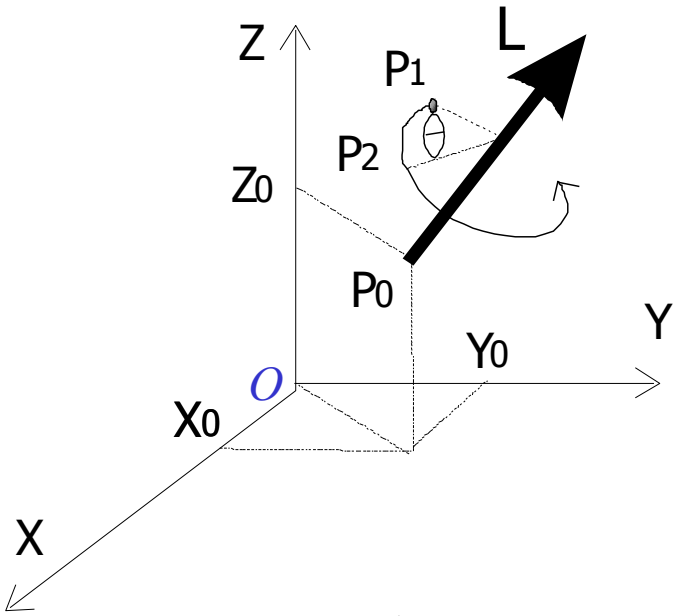
Final Transformation matrix :

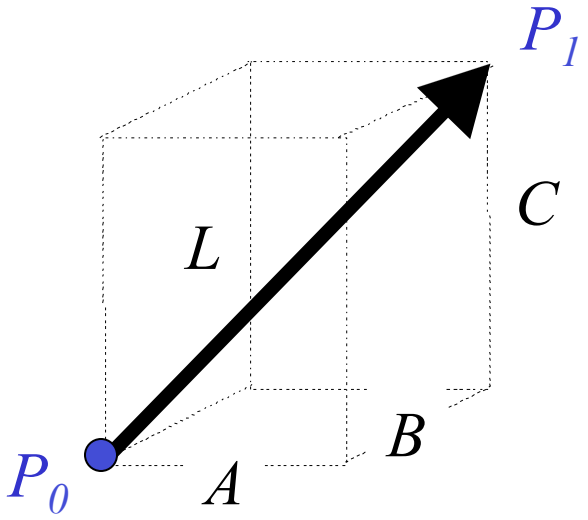
$$T = T(x_1, y_1, z_1)\, R_x(-\alpha)\, R_y(-\beta)\, R_z(\theta)\, R_y(\beta)$$
$$R_x(\alpha)\, T(-x_1, -y_1, -z_1)$$

Rotation about an Arbitrary Axis (Line)



Rotation about an Arbitrary Axis (Line)





$$x = Au + x_0$$

$$y = Bu + y_0 \qquad 0 <= u <= 1$$

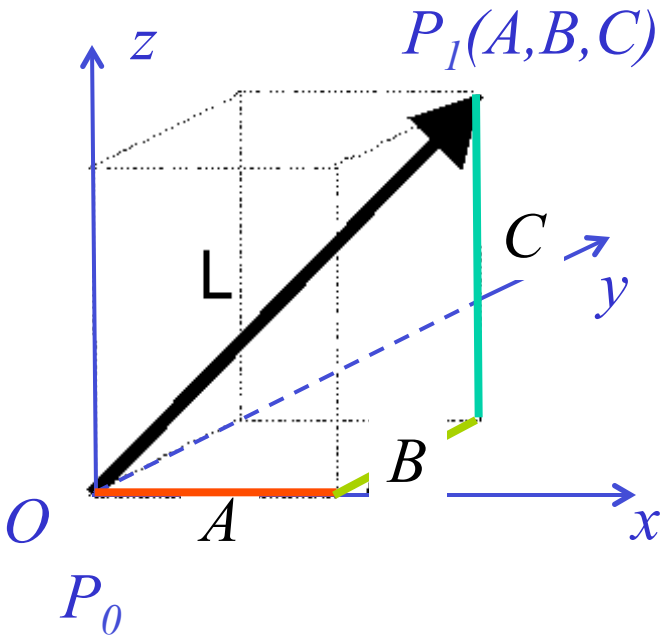
$$z = Cu + z_0$$

$$L = \sqrt{A^2 + B^2 + C^2}\,u$$

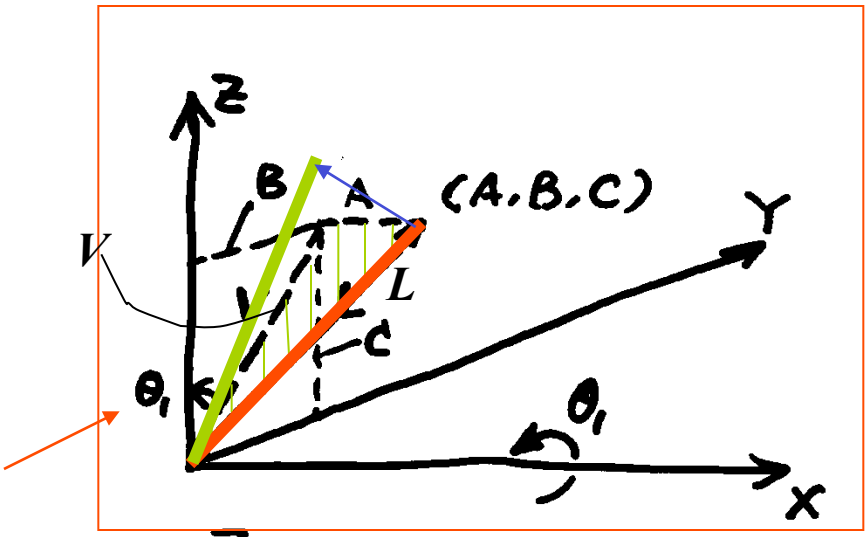
Step 1: Translate Point P_0 to Origin O

$$P_0 = [x_o \; y_o \; z_o]^T$$

$$[D] = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Step 2: Rotate Vector about X Axis to get into the x - z plane

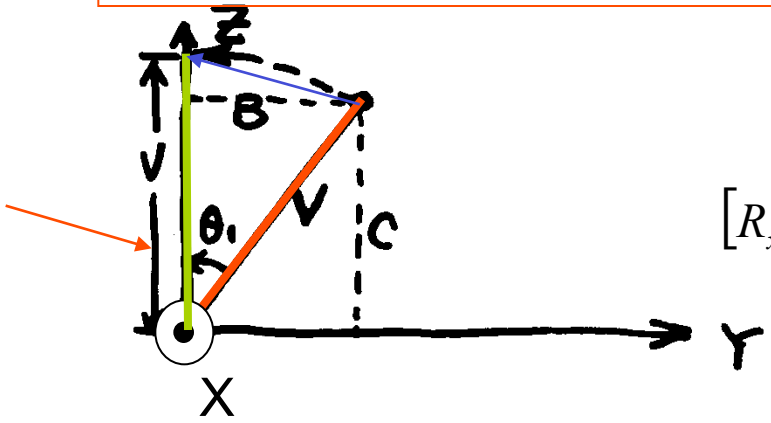


$$L = \sqrt{A^2 + B^2 + C^2}$$

$$V = \sqrt{B^2 + C^2}$$

$$\sin\theta_1 = \frac{B}{V}$$

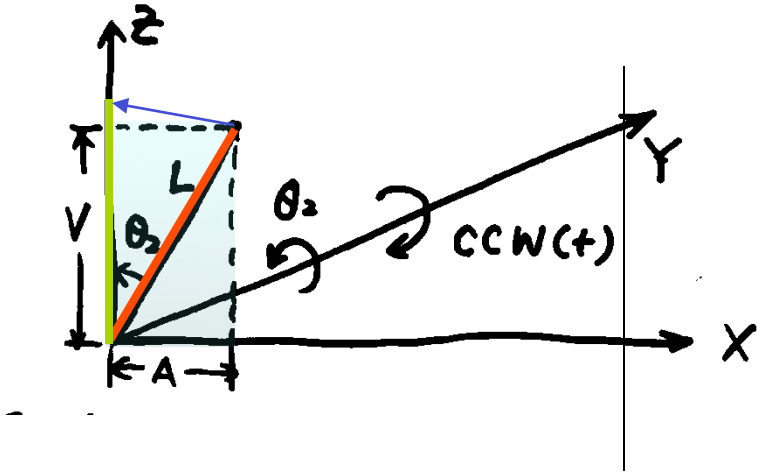
$$\cos\theta_1 = \frac{C}{V}$$



$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{C}{V} & -\frac{B}{V} & 0 \\ 0 & \frac{B}{V} & \frac{C}{V} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 3: Rotate about the Y axis to get it in the *Z* direction

Rotate a negative angle (CW)!

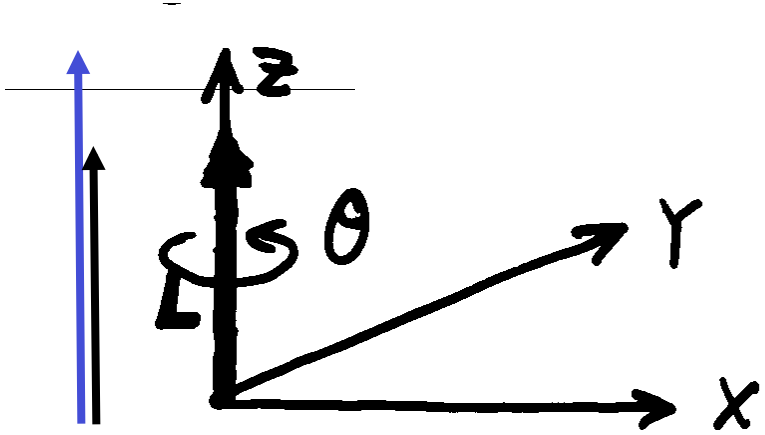


$$\sin\theta_2 = -\frac{A}{L}$$

$$\cos\theta_2 = \frac{V}{L}$$

$$[R_y] = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{V}{L} & 0 & -\frac{A}{L} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{A}{L} & 0 & \frac{V}{L} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 4: Rotate angle θ about axis $\vec{L}$



$$[R_z] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 5: Reverse the rotation about the Y axis

$$[R_y]^{-1} = \begin{bmatrix} \frac{V}{L} & 0 & \frac{A}{L} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{A}{L} & 0 & \frac{V}{L} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[R_y] = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{V}{L} & 0 & -\frac{A}{L} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{A}{L} & 0 & \frac{V}{L} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse of Rotation:

Replace θ by $-\theta$

$\sin\theta$ by $-\sin\theta$

$\cos\theta$ remains $\cos\theta$ (why?)

Step 6: Reverse rotation about the X axis

$$[R_x]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{C}{V} & \frac{B}{V} & 0 \\ 0 & -\frac{B}{V} & \frac{C}{V} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{C}{V} & -\frac{B}{V} & 0 \\ 0 & \frac{B}{V} & \frac{C}{V} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 7: Reverse translation

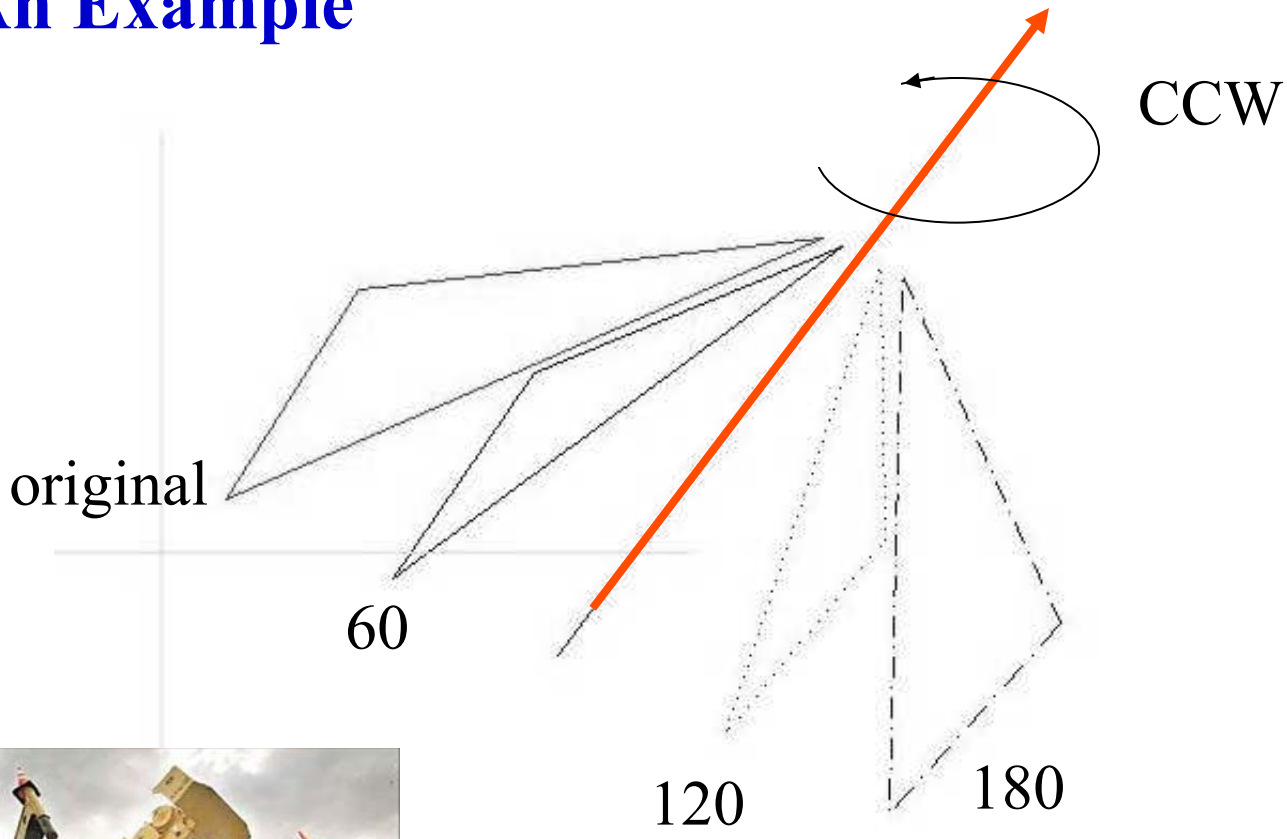
$$[D]^{-1} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Overall Transformation

$$[T] = [D]^{-1}[R_x]^{-1}[R_y]^{-1}[R_z^{\theta}][R_y][R_x][D]$$

$$P_2 = [T]P_1$$

An Example



An Example

Given the point matrix (four points) on the right; and a line, *NM*, with point *N* at (6, -2, 0) and point *M* at (12, 8, 0).

$$[\boldsymbol{P}_1]= \begin{array}{c} \begin{matrix} \boldsymbol{P}_1 & \boldsymbol{P}_2 & \boldsymbol{P}_3 & \boldsymbol{P}_4 \end{matrix} \\ \begin{pmatrix} 3 & 10 & 1 & 3 \\ 5 & 6 & 1 & 5 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Rotate the these four points 60 degrees around line *NM* (alone the *N* to *M* direction) *N: u=0; M: u=1*

$P_o = N$

$P_1 = M$

$$\begin{aligned} A &= 12 - 6 = 6 \\ B &= 8 - (-2) = 10 \\ C &= 0 - 0 = 0 \end{aligned}$$

1. Calculate the constants (the Line/Axis of Rotation)

$x = 6 + 6u$

$y = -2 + 10u$

$z = 0$

Thus

$A = 6,\ B = 10,\ C = 0$

$L = \sqrt{A^2 + B^2 + C^2} = 11.6619$

$V = \sqrt{B^2 + C^2} = 10$

2. Translate *N* to the origin

$$[D] = \begin{pmatrix} 1 & 0 & 0 & -6 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Rotate about the *X* axis

$$[R]_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & -B/V & 0 \\ 0 & B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Rotate about the *Y* axis

$$[R]_y = \begin{pmatrix} V/L & 0 & -A/L & 0 \\ 0 & 1 & 0 & 0 \\ A/L & 0 & V/L & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. Rotate 60 degree (positive)

$$[R]_z = \begin{pmatrix} \cos(60) & -\sin(60) & 0 & 0 \\ \sin(60) & \cos(60) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6. Reverse $[R]_y$

$$[R]_y^{-1} = \begin{pmatrix} V/L & 0 & A/L & 0 \\ 0 & 1 & 0 & 0 \\ -A/L & 0 & V/L & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7. Reverse $[R]_x$

$$[R]_x^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & B/V & 0 \\ 0 & -B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

8. Reverse the Translation

$$[D]^{-1} = \begin{pmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

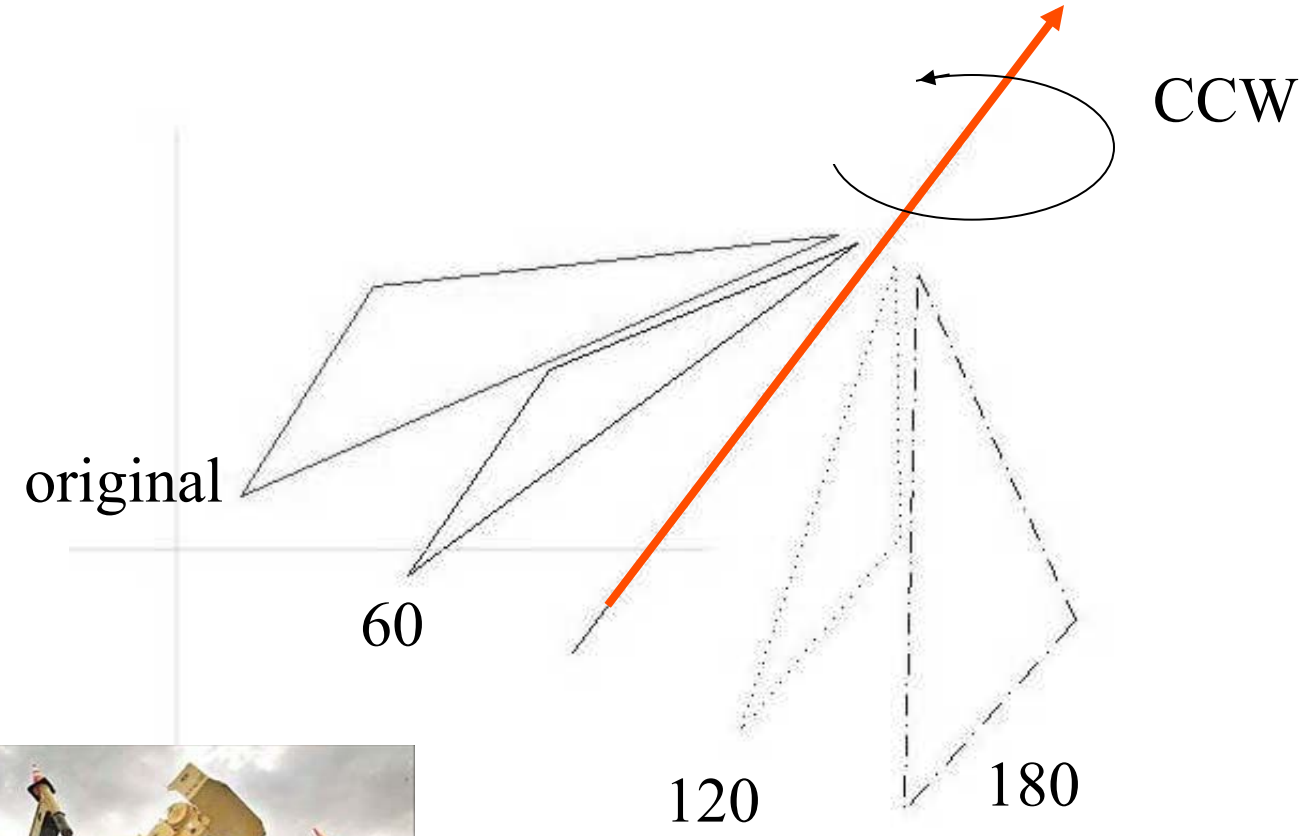
9. Calculate the total transformation

$$[T] = [D]^{-1}[R_x]^{-1}[R_y]^{-1}[R_z^{60}][R_y][R_x][D]$$

$$P_2 = [T]P_1$$

$$[\mathrm{P}]_2 = \begin{pmatrix} 5.6471 & 10.2941 & 3.5000 & 5.6471 \\ 3.4118 & 5.8235 & -0.5000 & 3.4118 \\ 5.3468 & 0.5941 & 5.0498 & 5.3468 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 \end{pmatrix}$$

P_1 P_2 P_3 P_4

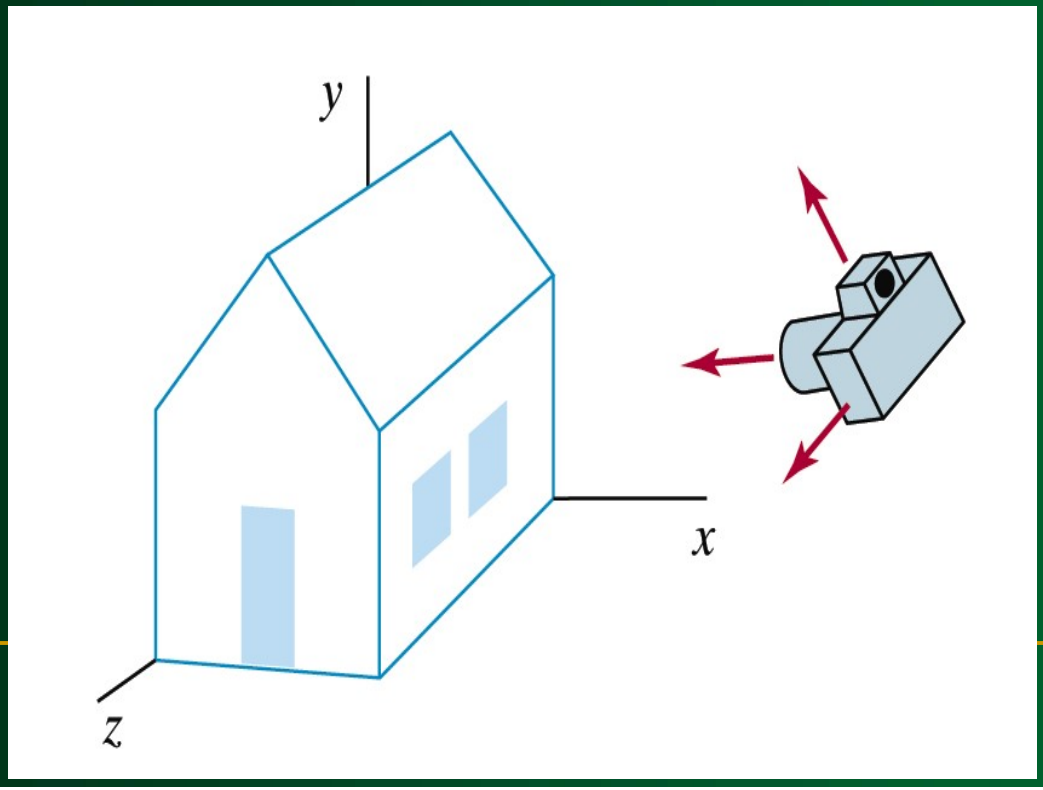

CCW
original
60
120
180


wangyou.com

Three Dimensional Viewing

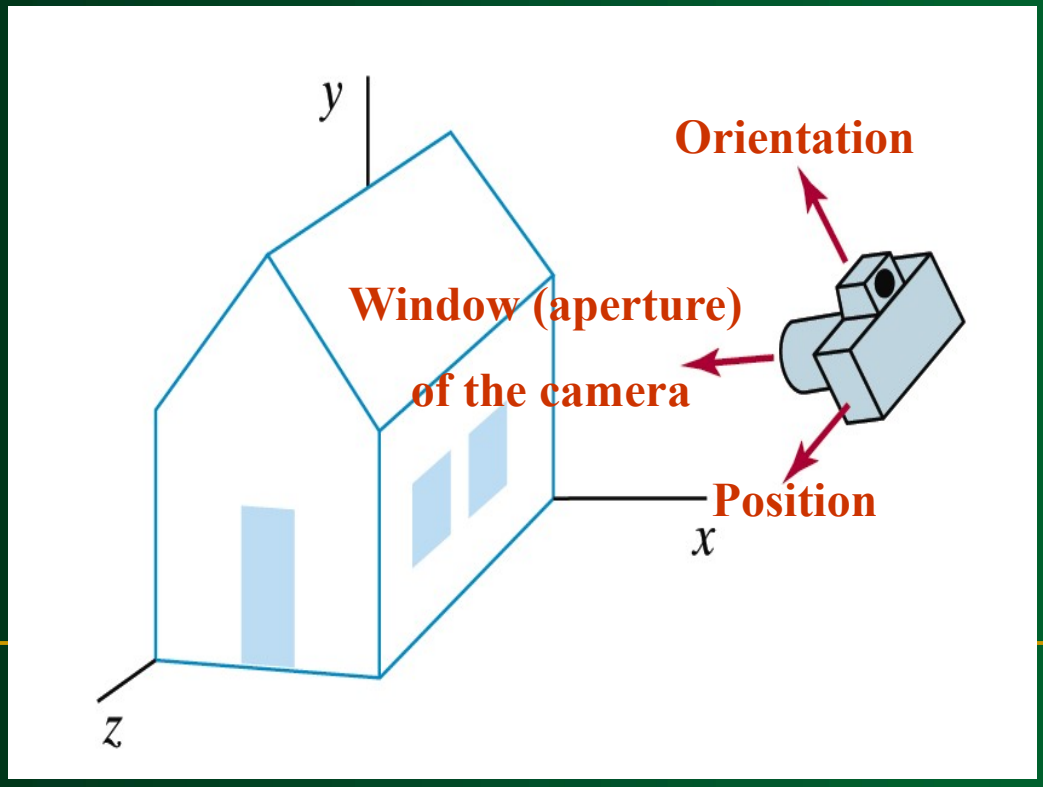
3D Viewing

- The steps for computer generation of a view of a three dimensional scene are somewhat analogous to the processes involved in taking a photograph.



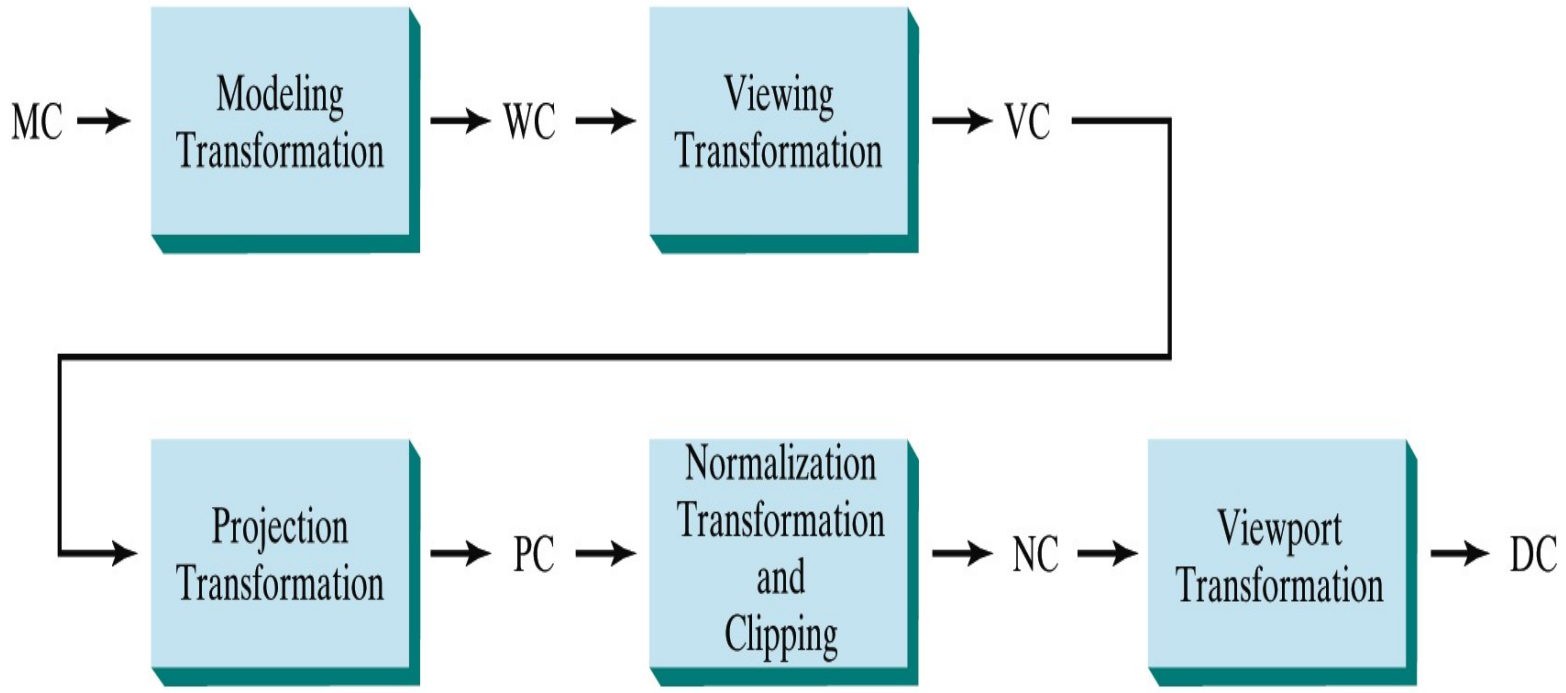
Camera Analogy

1. Viewing position
2. Camera orientation
3. Size of clipping window



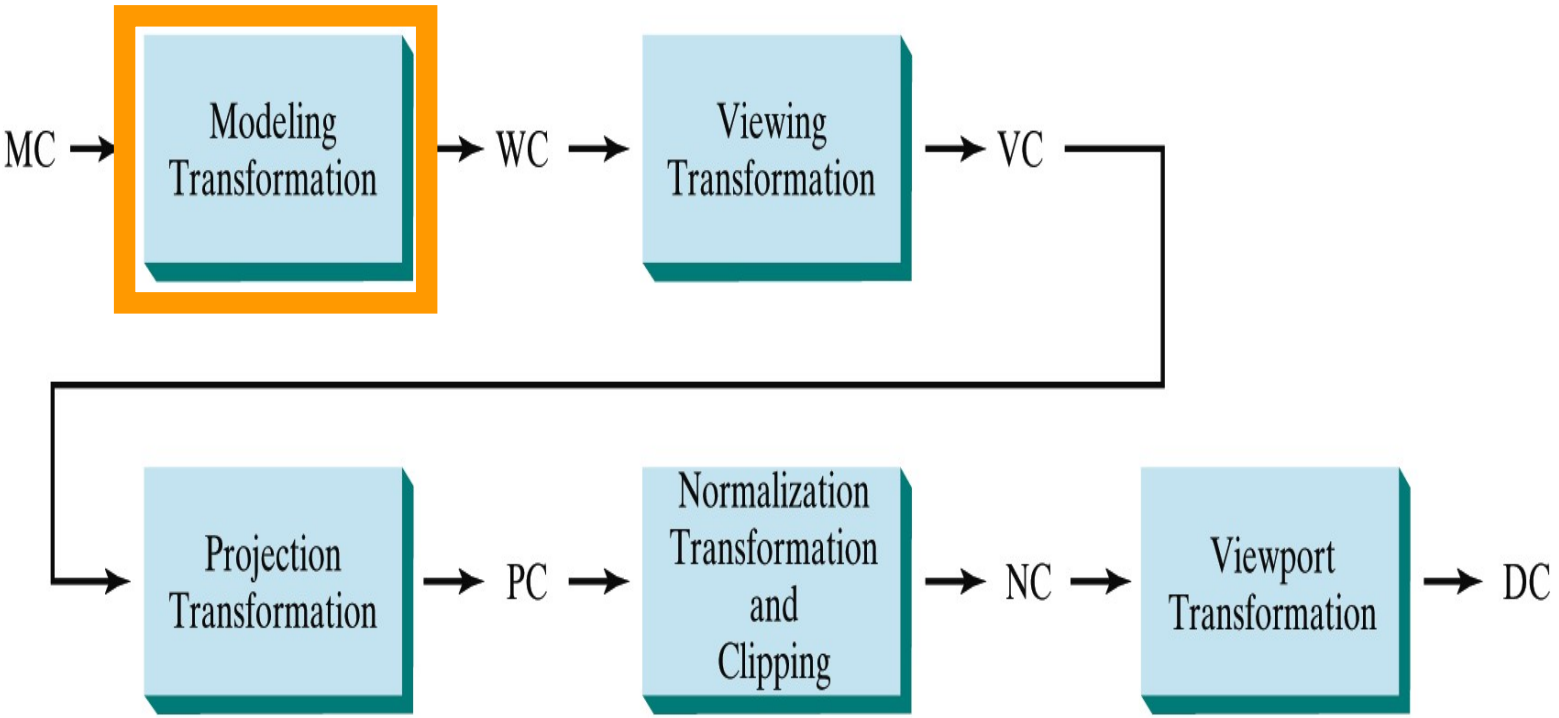
Viewing Pipeline

- The general processing steps for modeling and converting a world coordinate description of a scene to device coordinates:



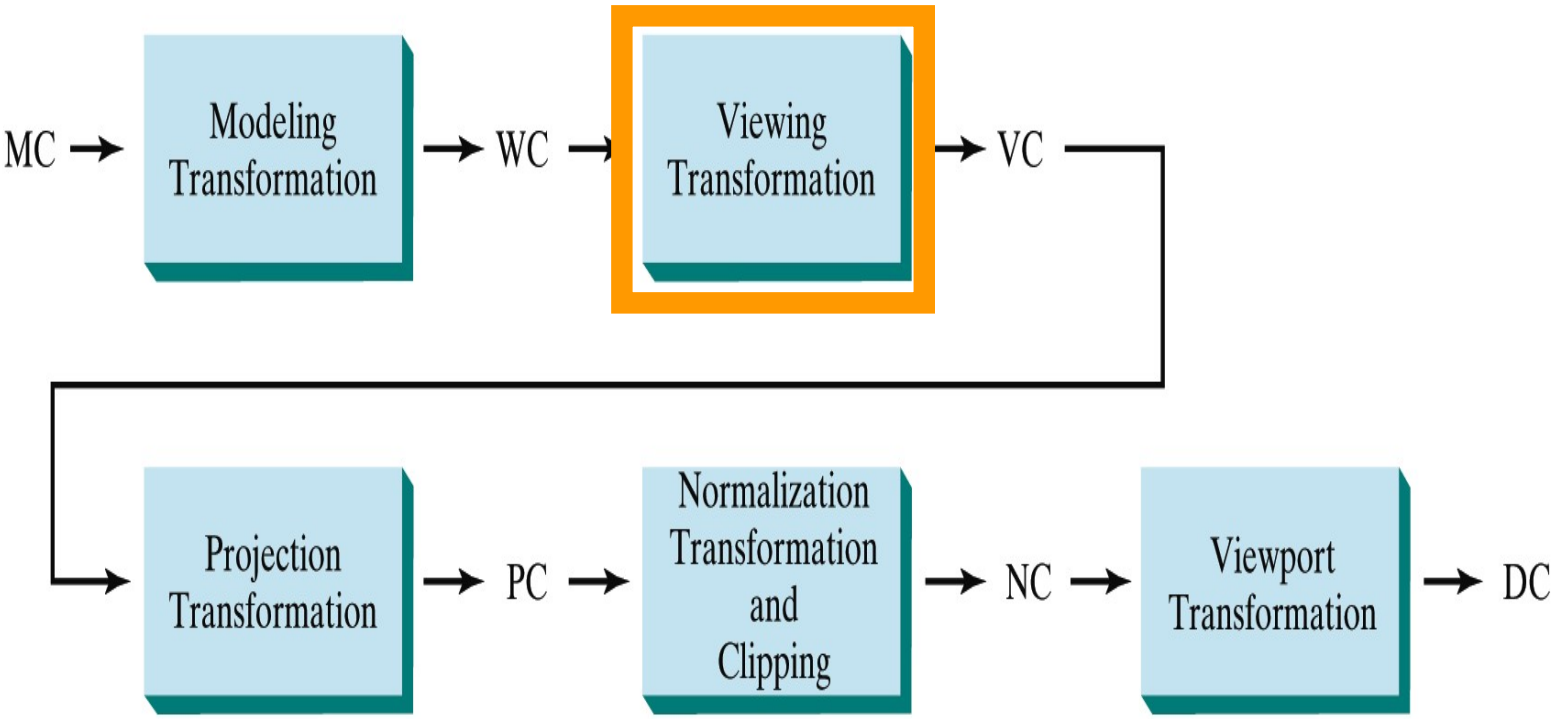
Viewing Pipeline

1. Construct the shape of individual objects in a scene within modeling coordinate, and place the objects into appropriate positions within the scene (world coordinate).



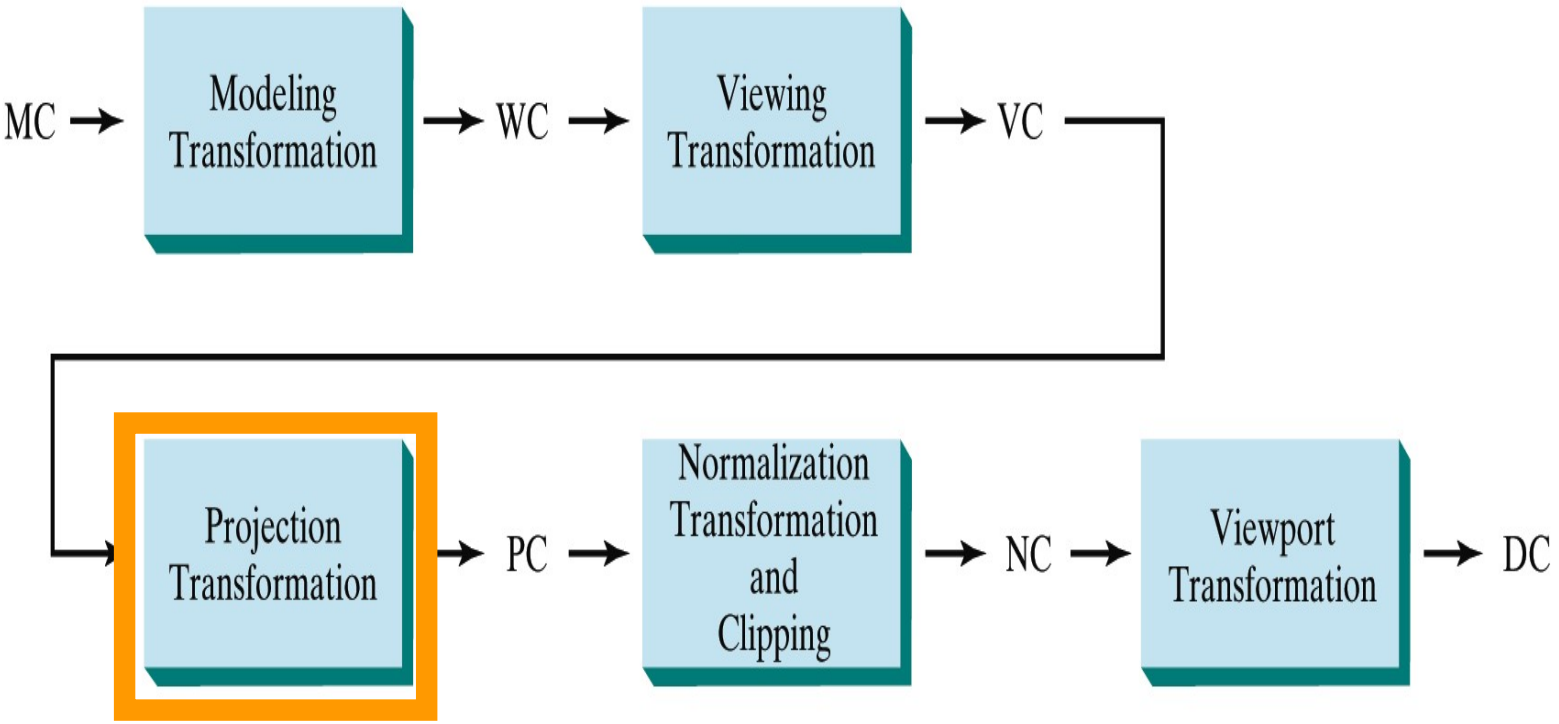
Viewing Pipeline

2. World coordinate positions are converted to viewing coordinates.



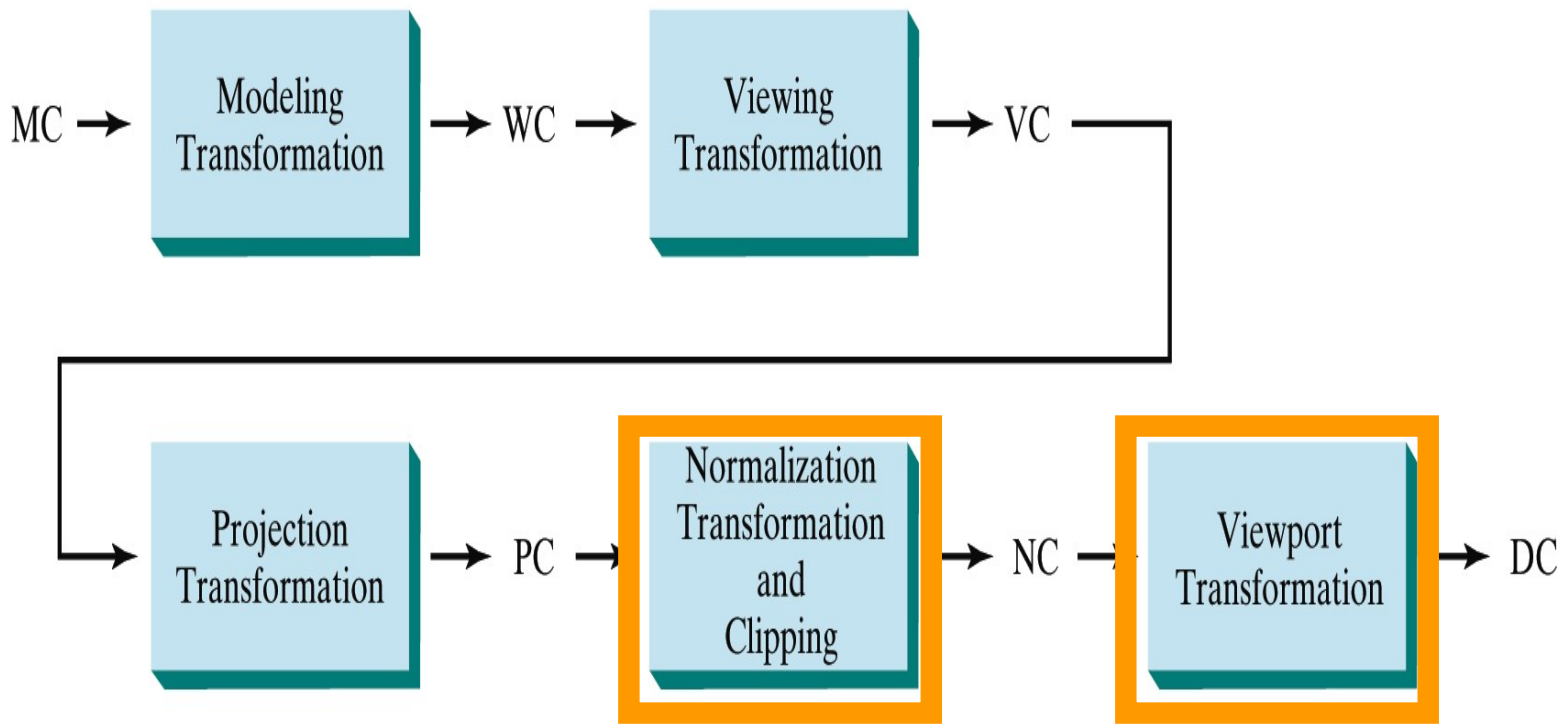
Viewing Pipeline

3. Convert the viewing coordinate description of the scene to coordinate positions on the projection plane.

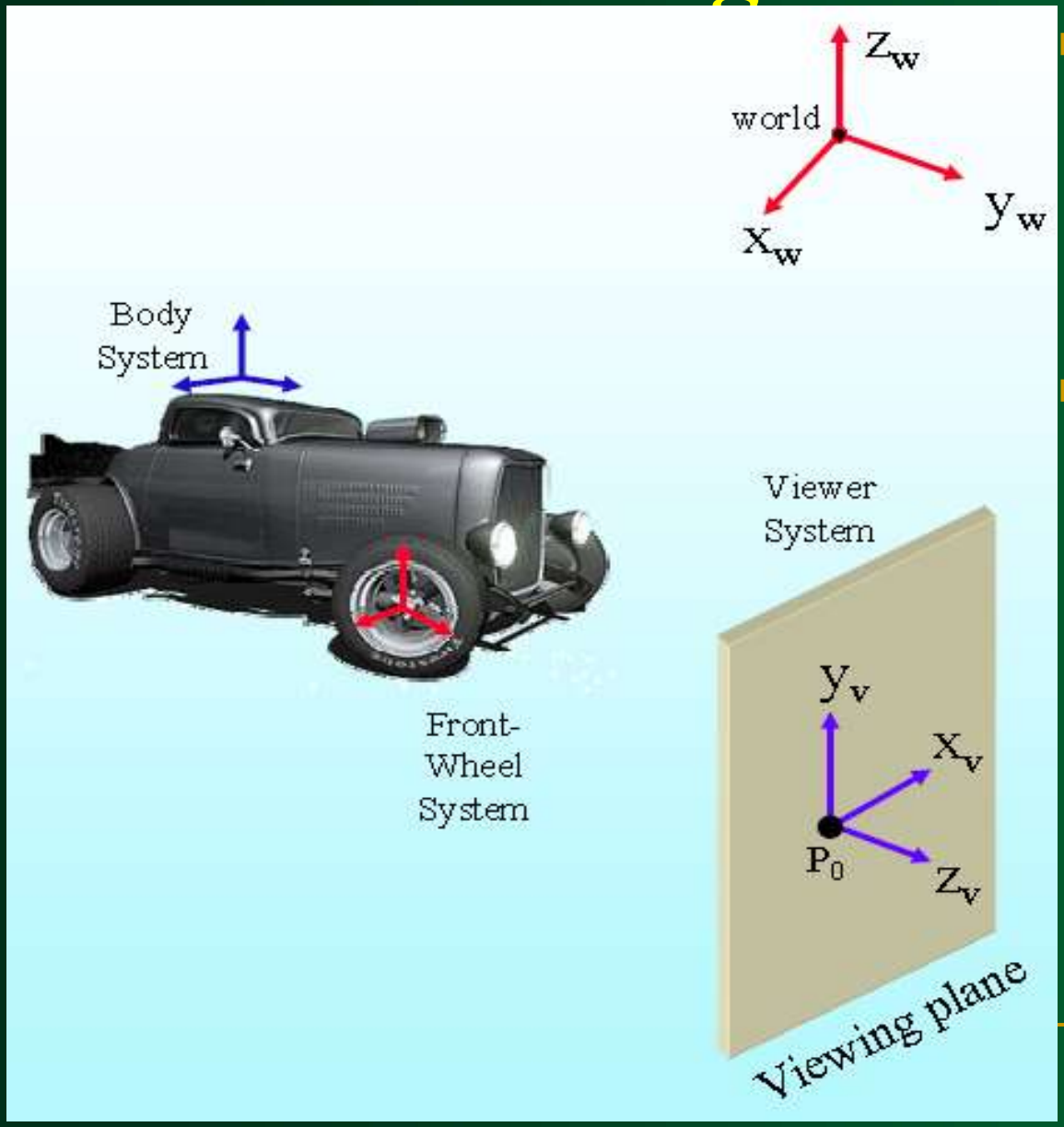


Viewing Pipeline

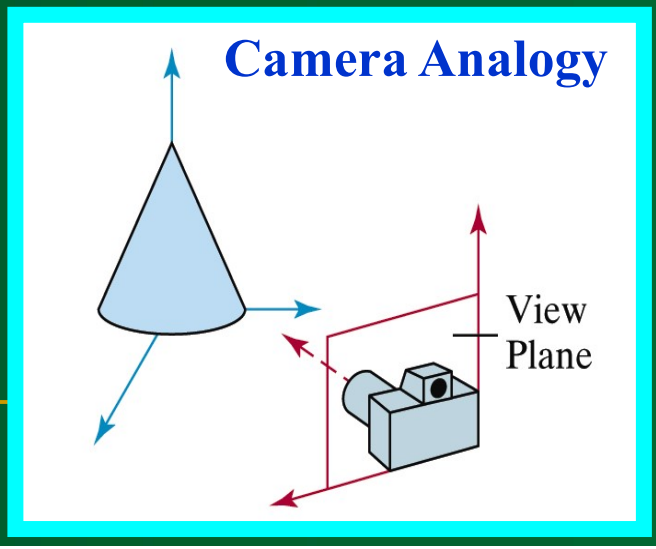
4. Positions on the projection plane, will then mapped to the Normalized coordinate and output device.



Viewing Coordinates

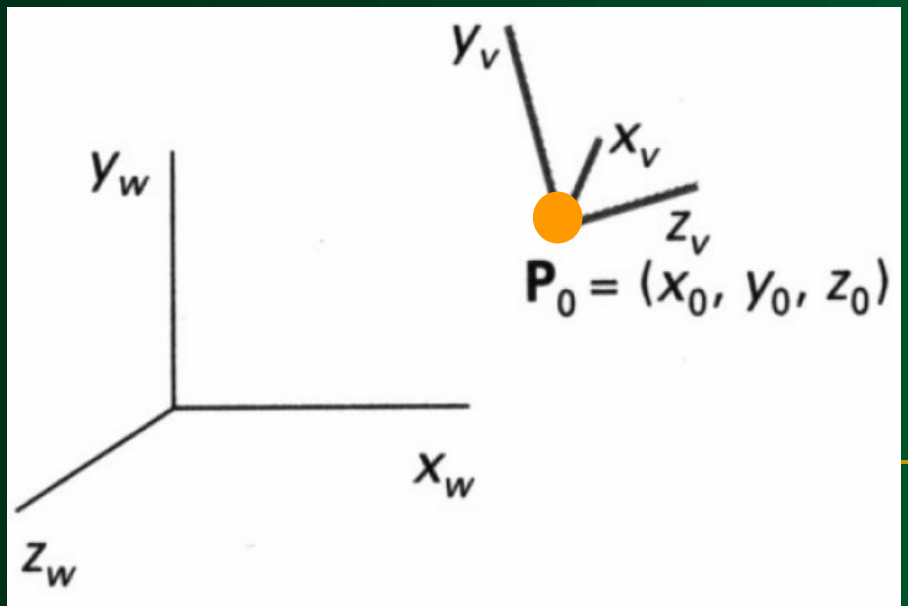


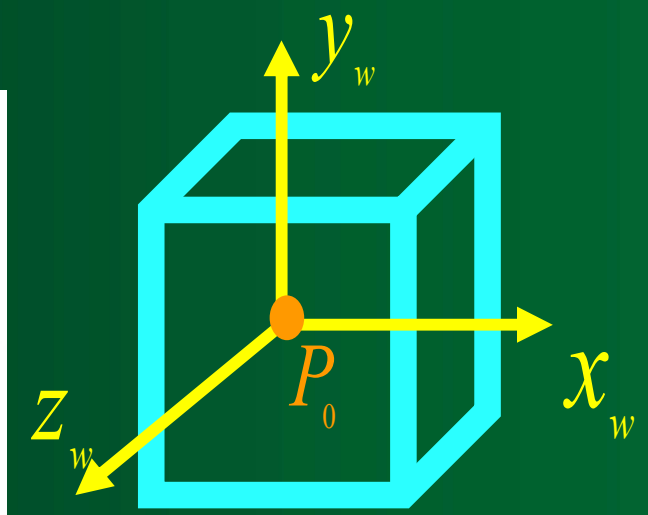
- Viewing coordinates system describes 3D objects with respect to a viewer.
- A Viewing (Projector) plane is set up perpendicular to z_v and aligned with (x_v,y_v).

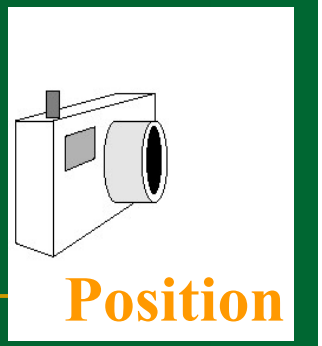


Specifying the Viewing Coordinate System (View Reference Point)

- We first pick a world coordinate position called **view reference point** (origin of our viewing coordinate system).
- P_0 is a point where a camera is located.
- The view reference point is often chosen to be close to or on the surface of some object, or at the center of a group of objects.

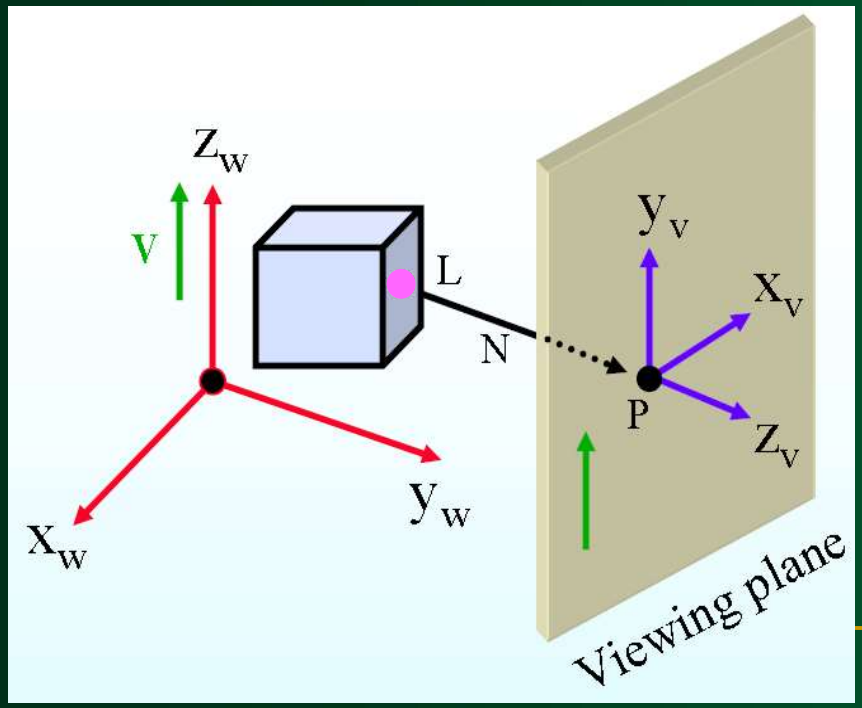


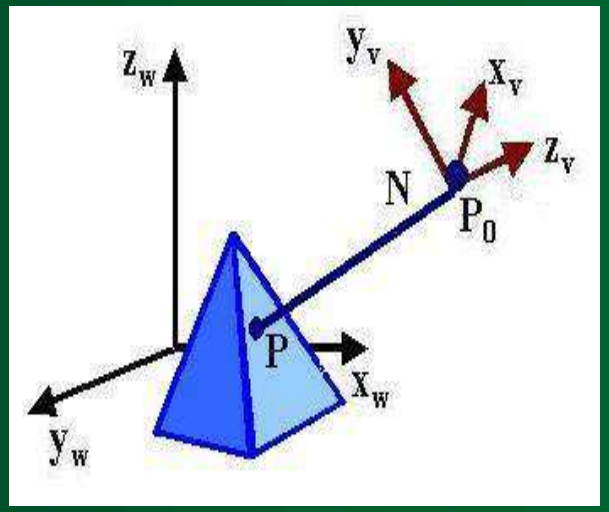


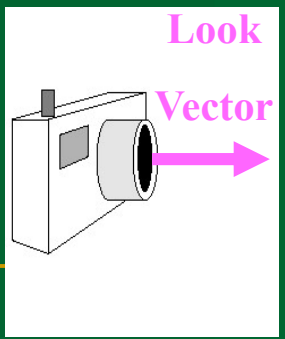


Specifying the Viewing Coordinate System (Z_v Axis)

- Next, we select the positive direction for the viewing z_v axis, by specifying the **view plane normal vector**, **N**.
- The direction of **N**, is from the **look at point** (L) to the view reference point.

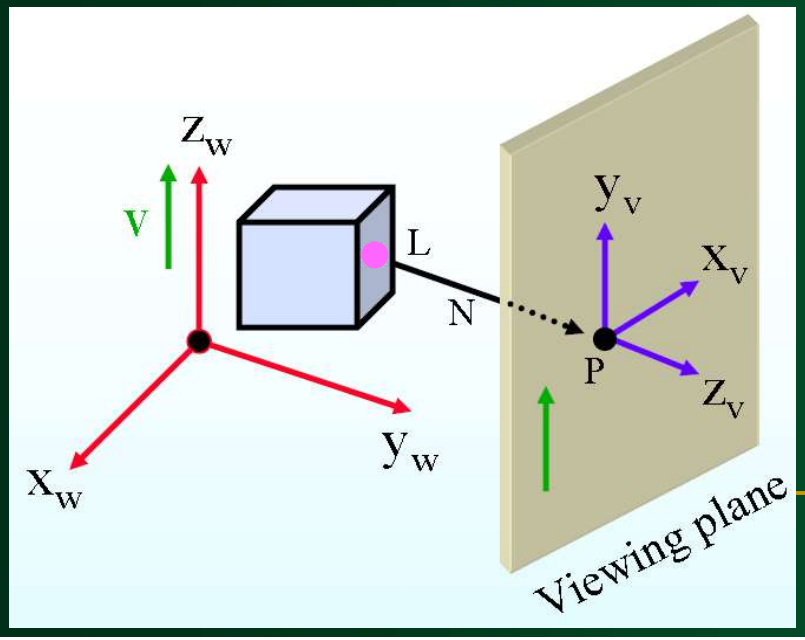


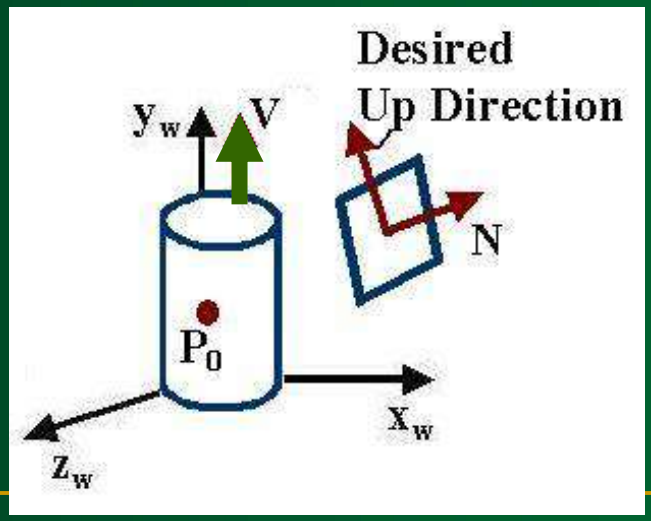


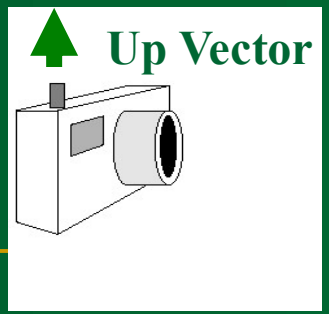


Specifying the Viewing Coordinate System (y_v Axis)

- Finally, we choose the ***up direction*** for the view by specifying a vector ***V***, called the ***view up vector***.
- This vector is used to establish the positive direction for the y_v axis.
- ***V*** is projected into a plane that is perpendicular to the normal vector.

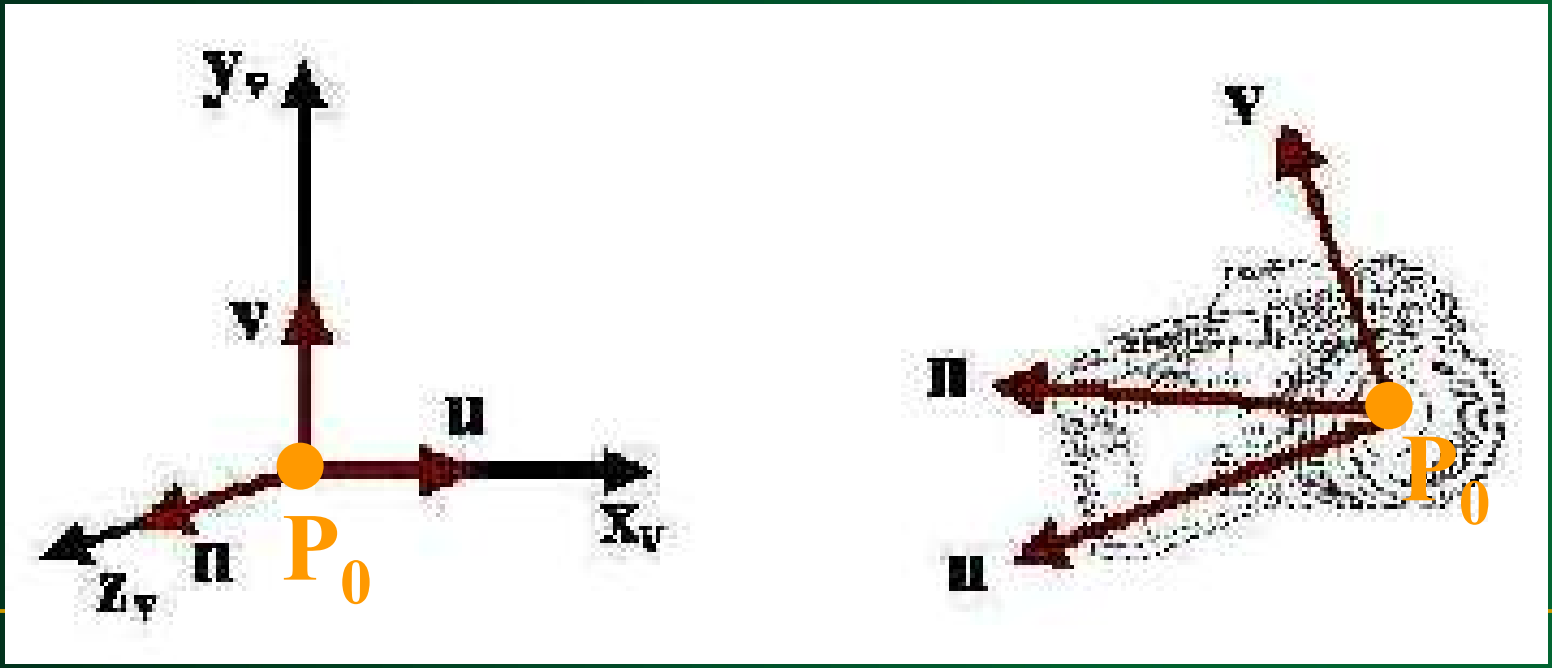






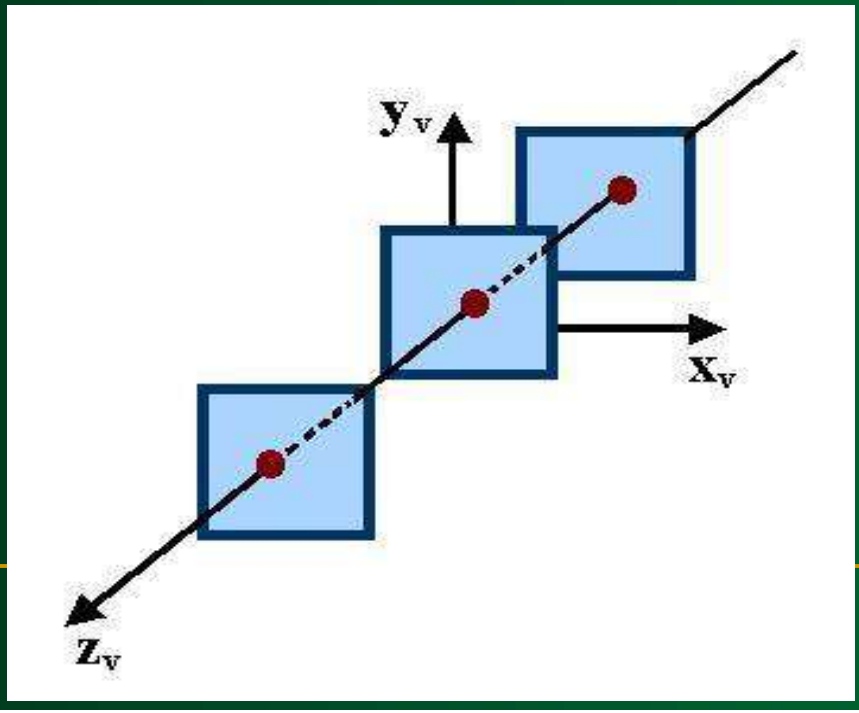
Specifying the Viewing Coordinate System (x_v Axis)

- Using vectors **N** and **V**, the graphics package computer can compute a third vector **U**, perpendicular to both **N** and **V**, to define the direction for the x_v axis.



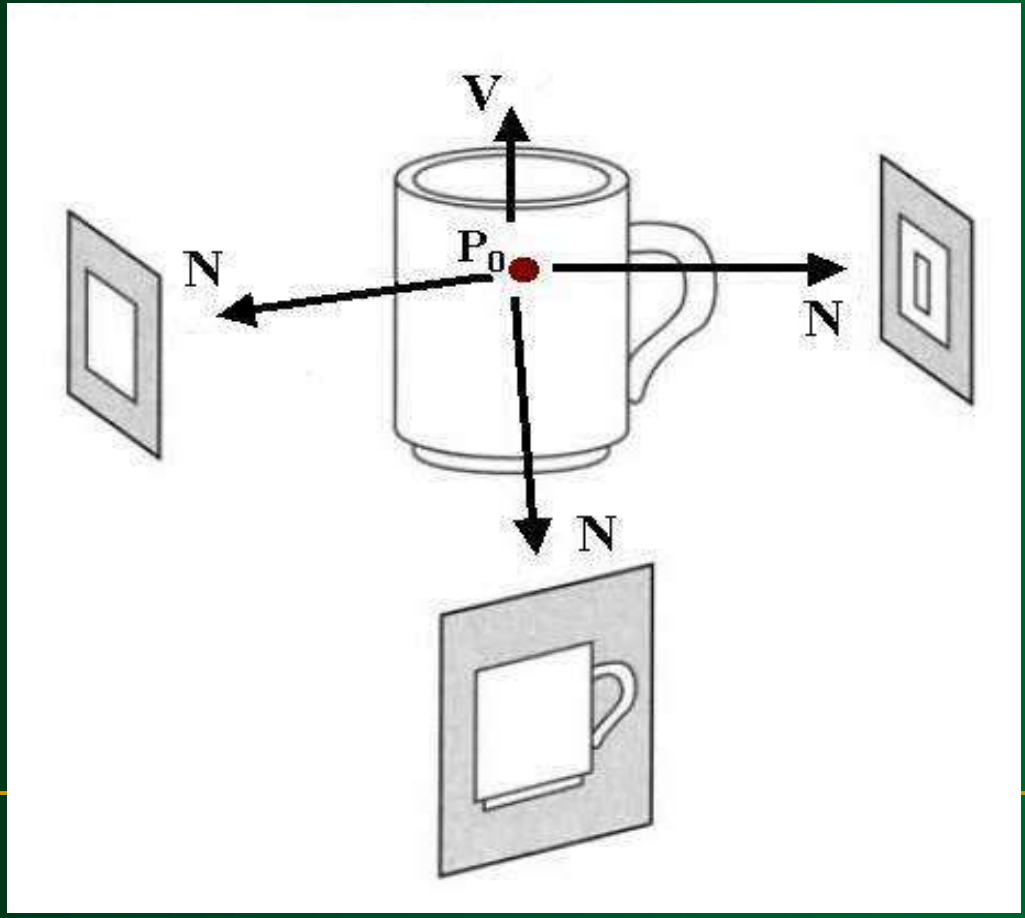
The View Plane

- Graphics package allow users to choose the position of the view plane along the z_v axis by specifying the **view plane distance** from the viewing origin.
- The view plane is always parallel to the $x_v y_v$ plane.



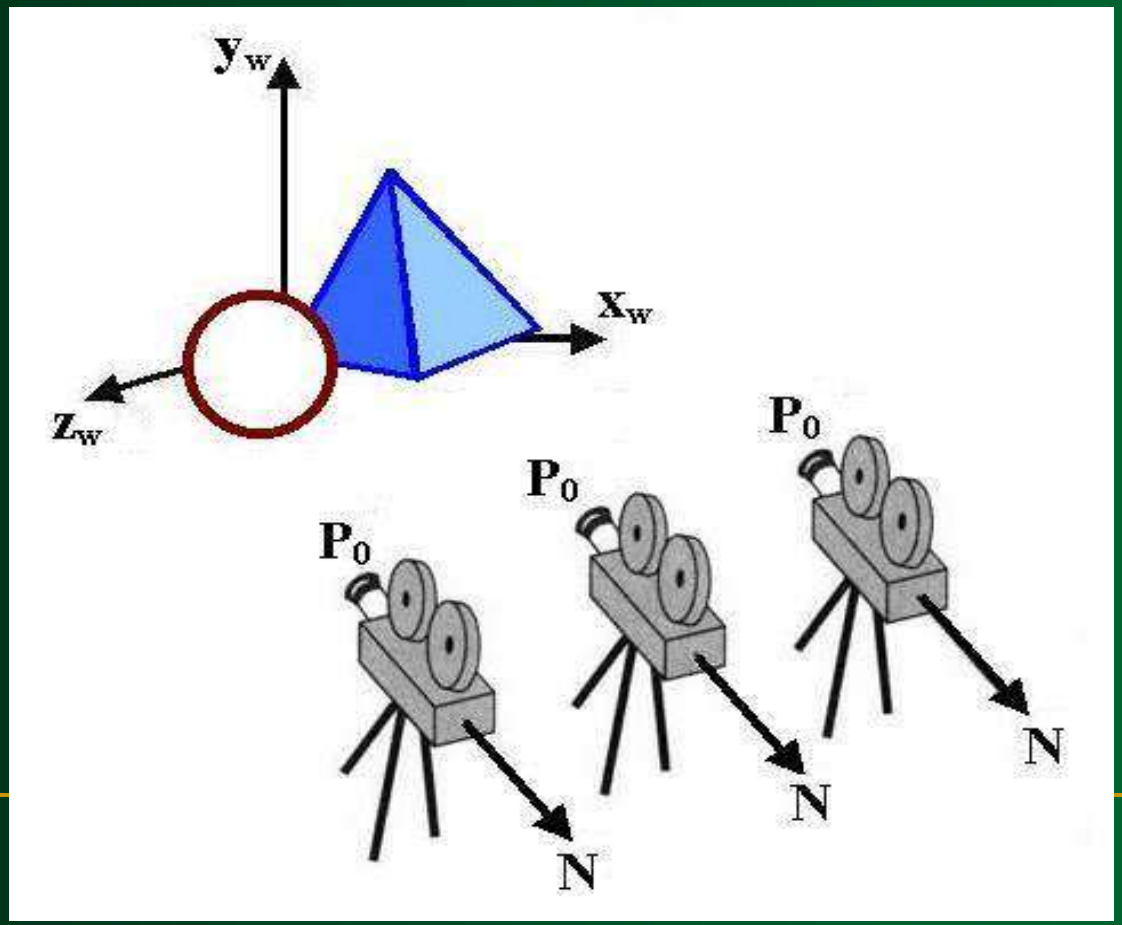
Obtain a Series of View

- To obtain a series of view of a scene, we can keep the view reference point fixed and **change** the direction of **N**.



Simulate Camera Motion

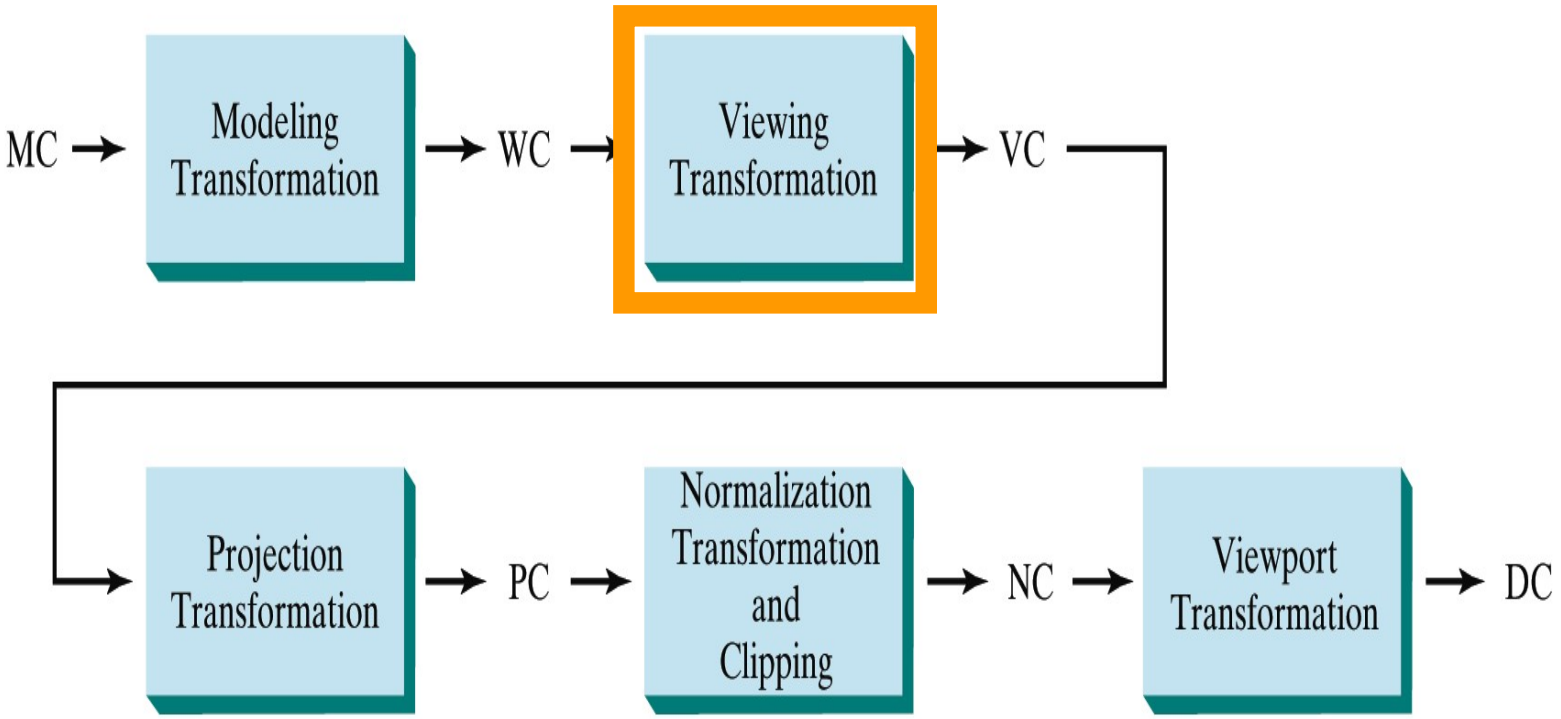
- To simulate camera motion through a scene, we can keep **N fixed** and **move** the view reference **point** around.



Transformation from World to Viewing Coordinates

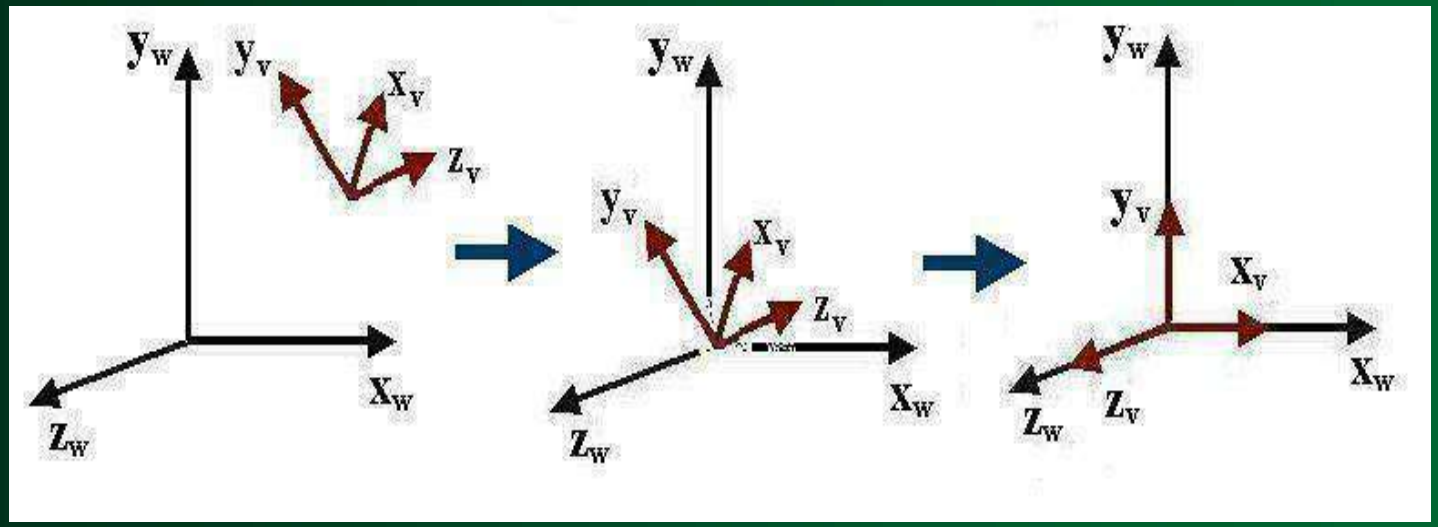
Viewing Pipeline

- Before object description can be projected to the view plane, they must be transferred to viewing coordinates.
- World coordinate positions are converted to viewing coordinates.



Transformation from World to Viewing Coordinates

- Transformation sequence from world to viewing coordinates:
- Translate the view reference point to the origin of the world-coordinate system.
- Apply rotations to align the x_v, y_v and z_v axes with the world x_w, y_w and z_w axes.

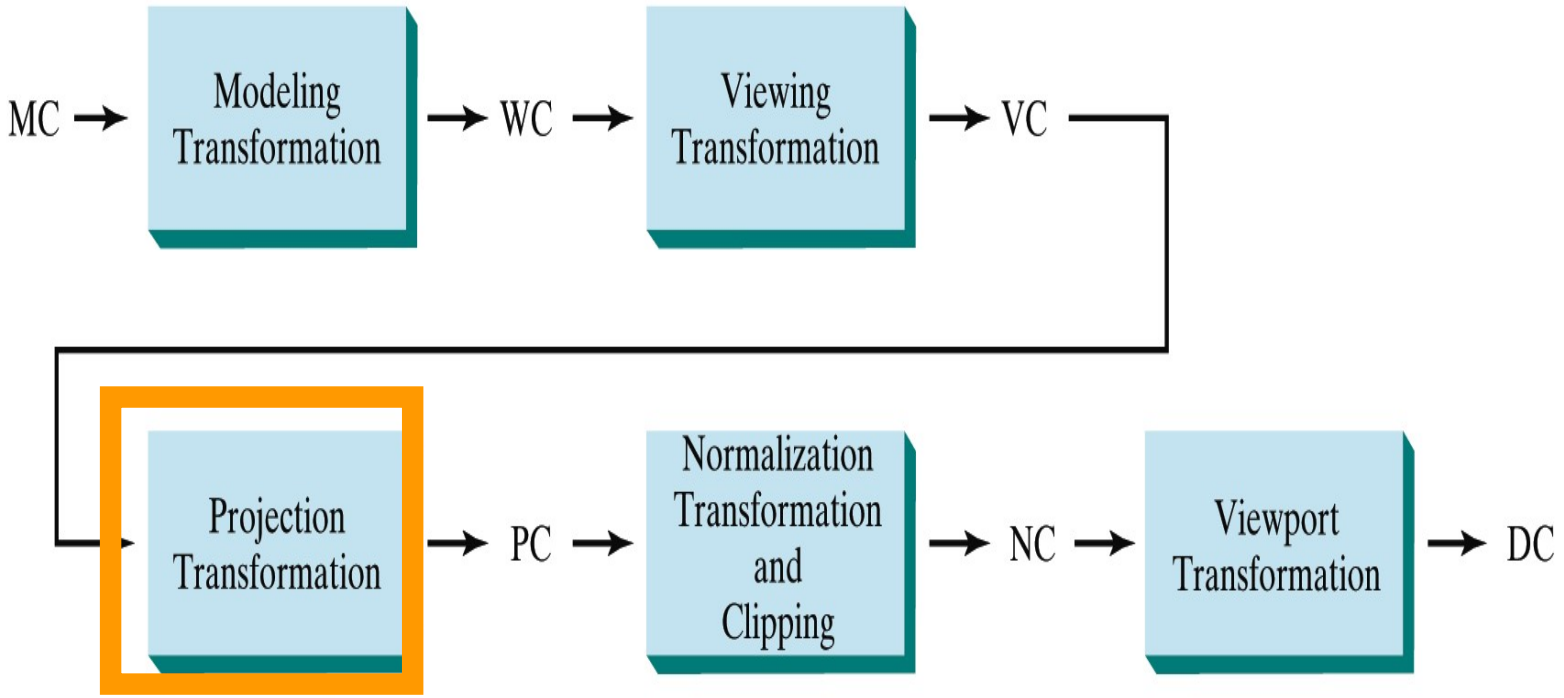


$$\mathbf{M}_{WC,VC} = \mathbf{R}_z \cdot R_y \cdot R_x . \mathbf{T}$$

Projection

Viewing Pipeline

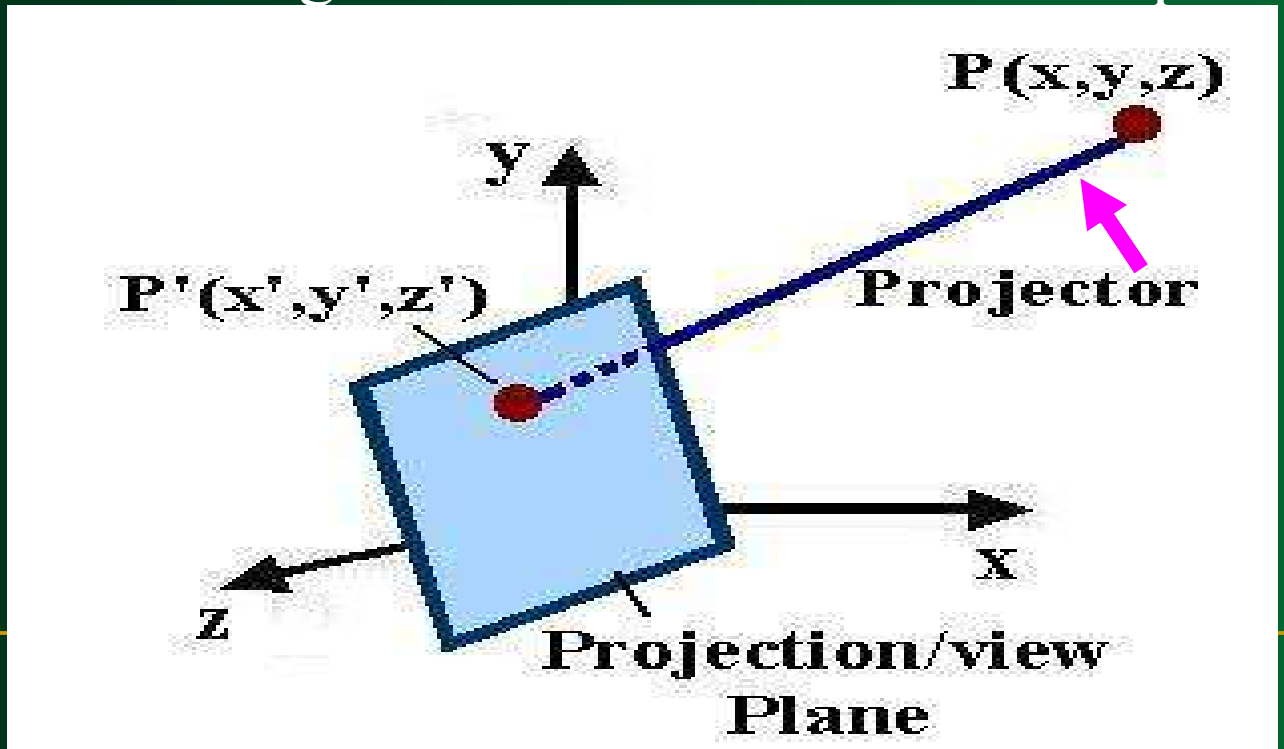
- Convert the viewing coordinate description of the scene to coordinate positions on the projection plane.
- Viewing 3D objects on a 2D display requires a mapping from 3D to 2D.



Projection

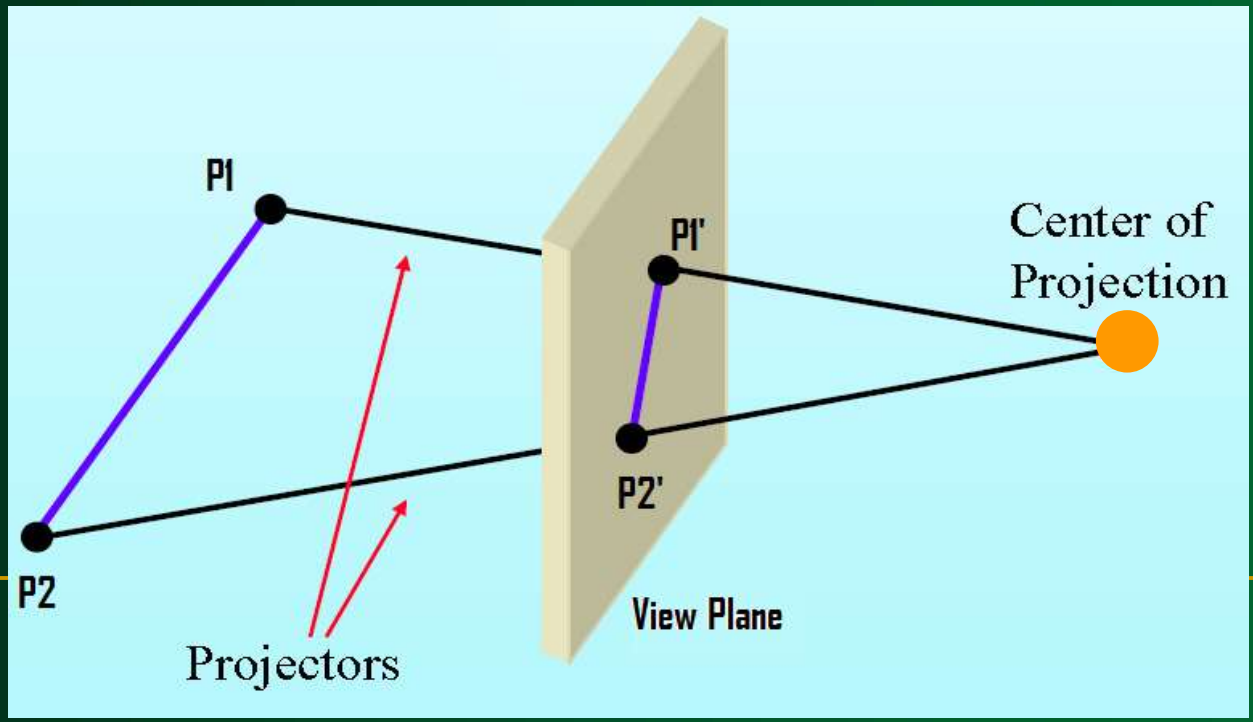
- **Projection** can be defined as a mapping of point P(x,y,z) onto its image $P'(x', y', z')$ in the projection plane.

- The mapping is determined by a ***projector*** that passes through P and intersects the view plane (P').

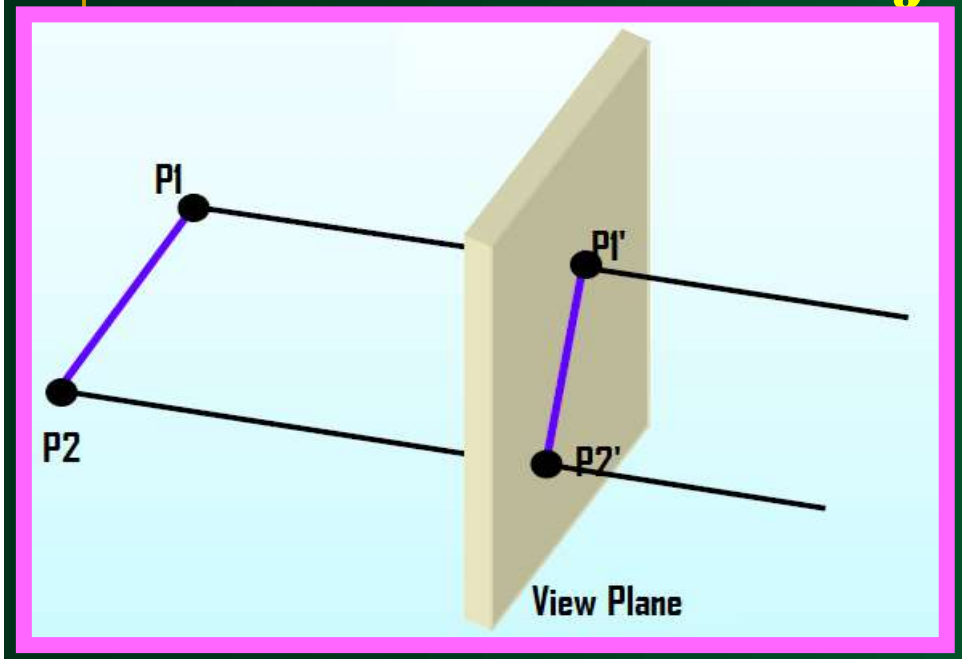


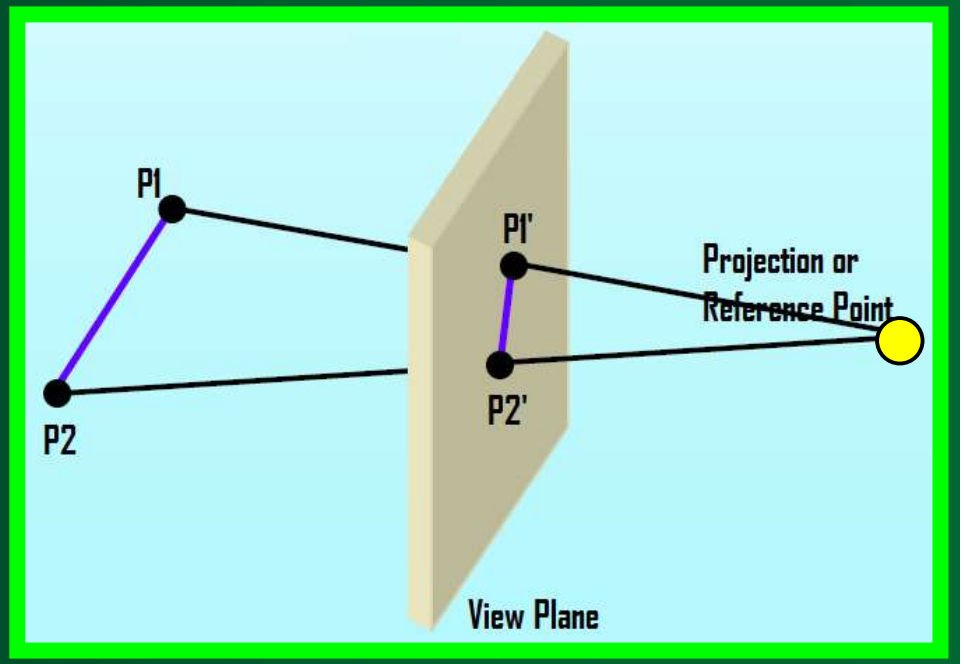
Projection

- Projectors are lines from **center (reference) of projection** through each point in the object.
- The projected view of the object is determined by calculating the intersection of projection lines with the view plane.



Projection



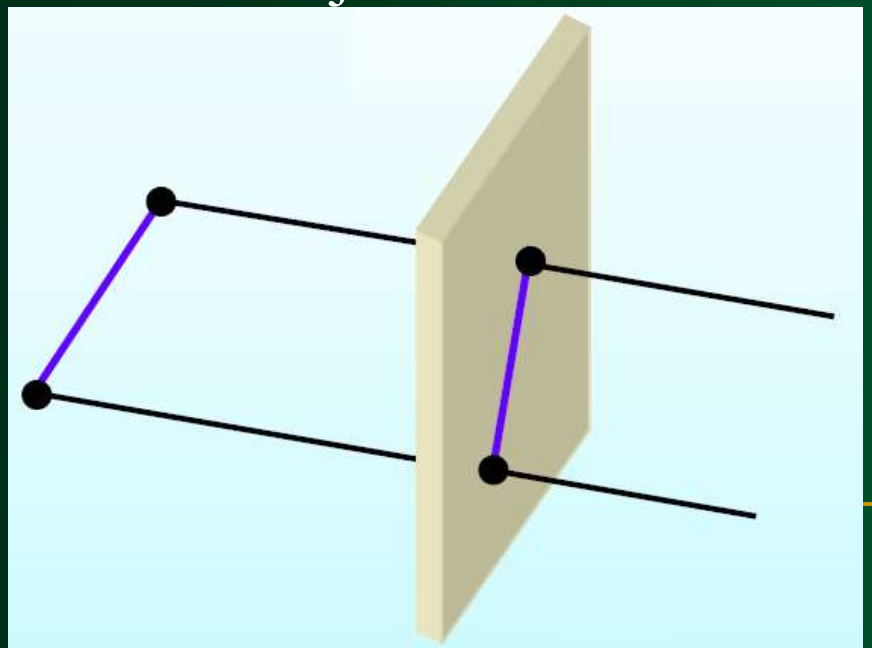


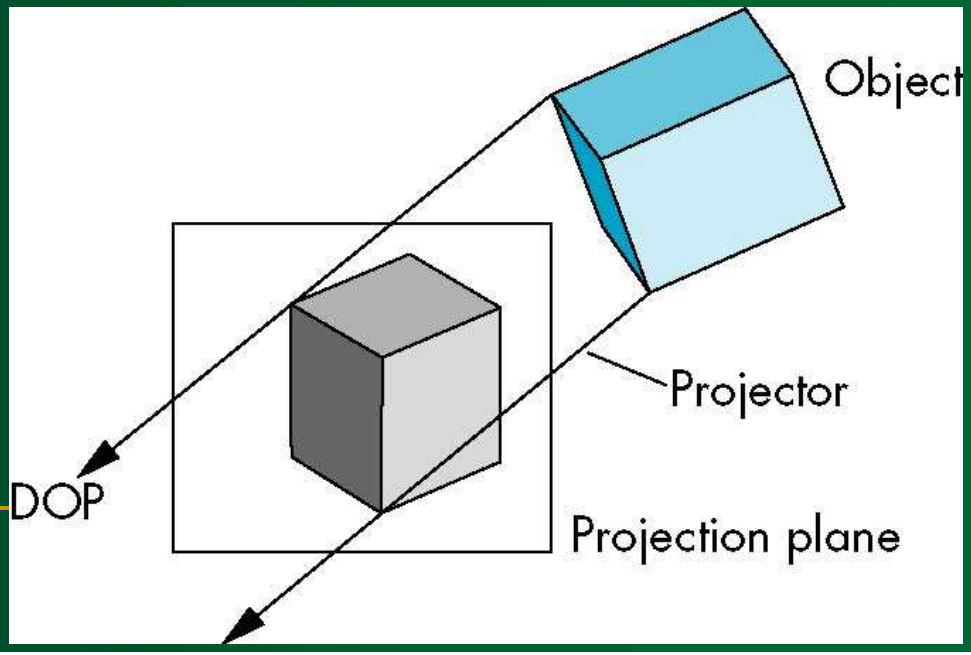
Parallel Projection :
Coordinate positions are transformed to the view plane along **parallel lines**.

Perspective Projection:
Object positions are transformed to the view plane along lines that converge to the **projection reference (center) point**.

Parallel Projection

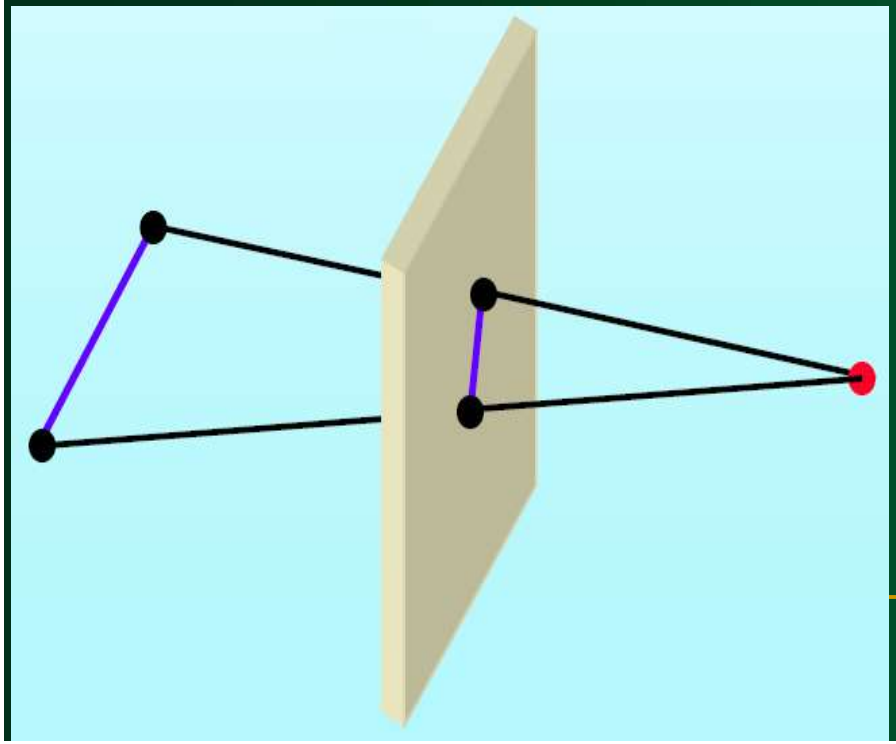
- Coordinate position are transformed to the view plane along parallel lines.
- Center of projection at infinity results with a parallel projection.
- A parallel projection preserves relative proportion of objects, but does not give us a realistic representation of the appearance of object.

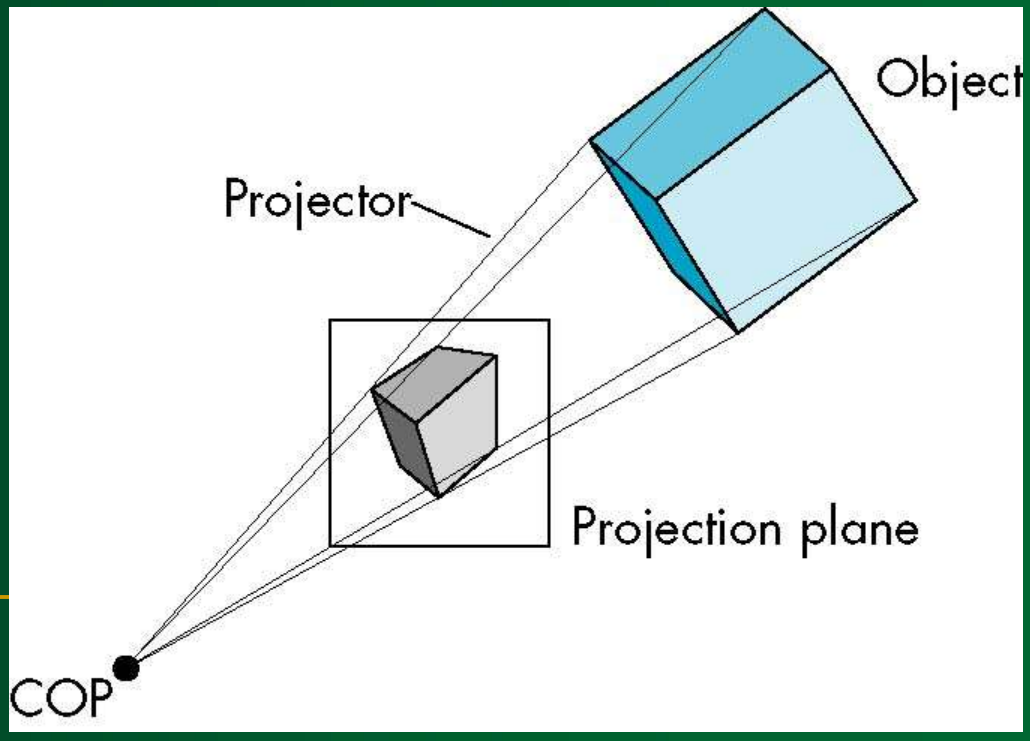




Perspective Projection

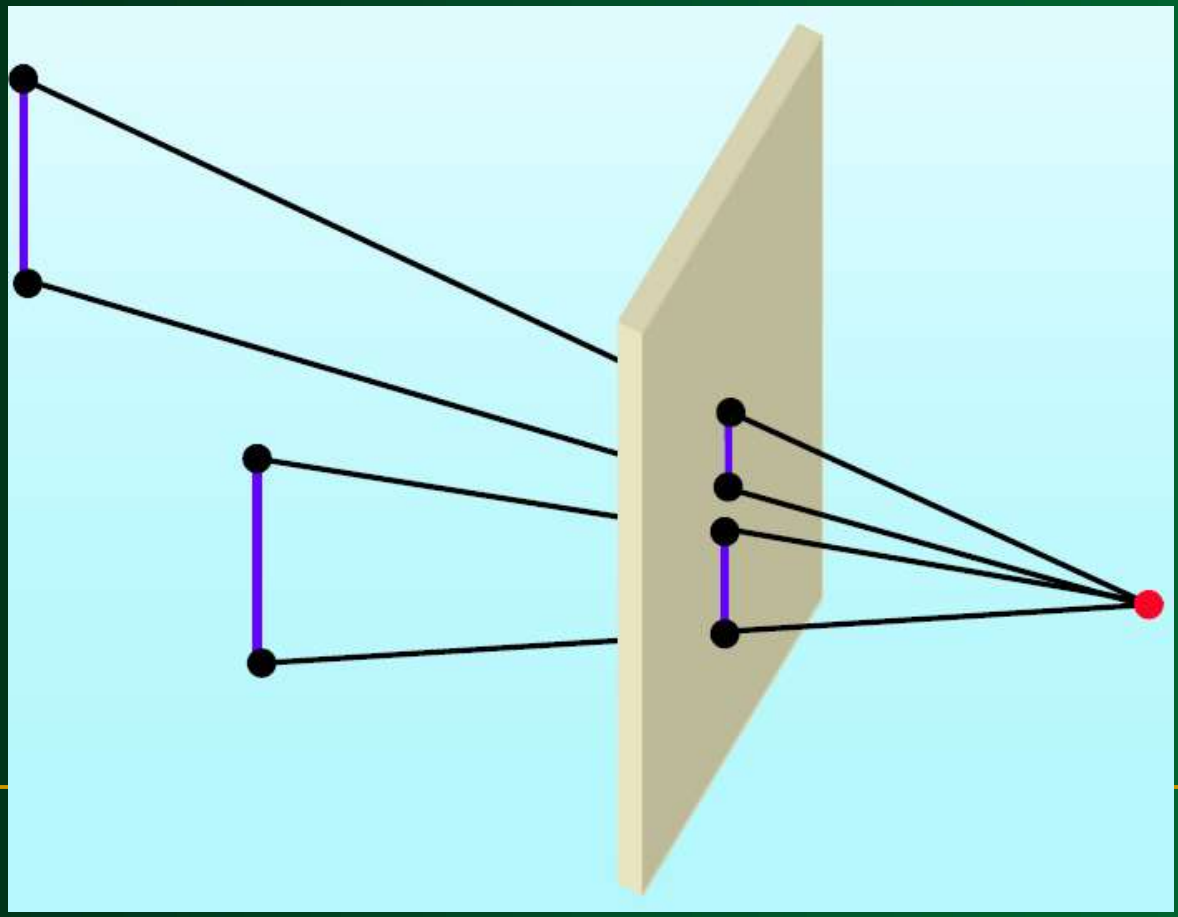
- Object positions are transformed to the view plane along lines that converge to the **projection reference (center) point.**
- Produces realistic views but does not preserve relative proportion of objects.



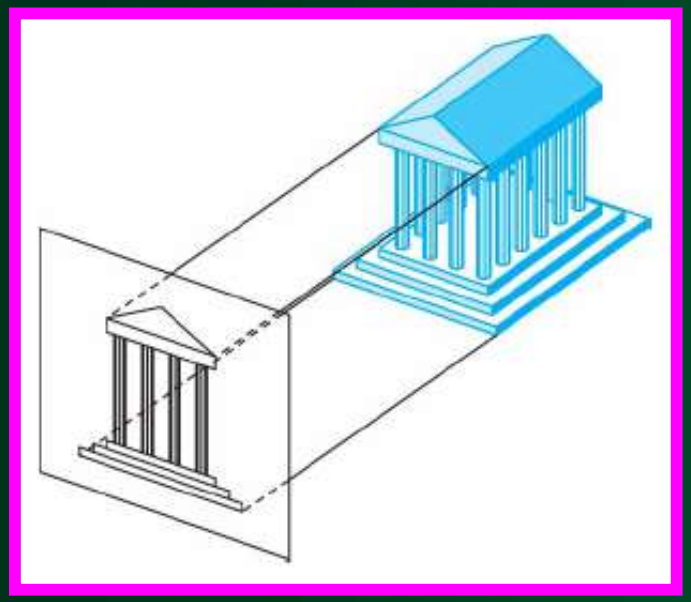


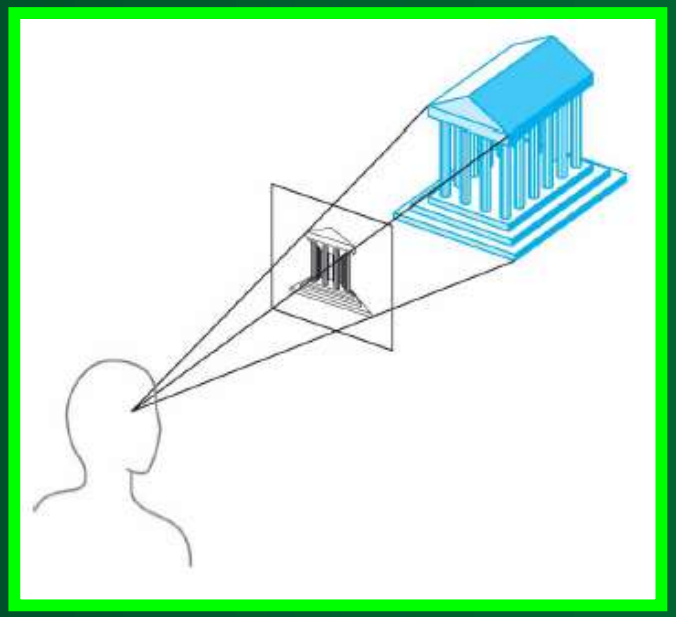
Perspective Projection

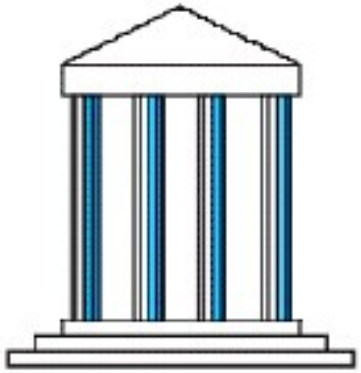
- Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.



Parallel and Perspective Projection







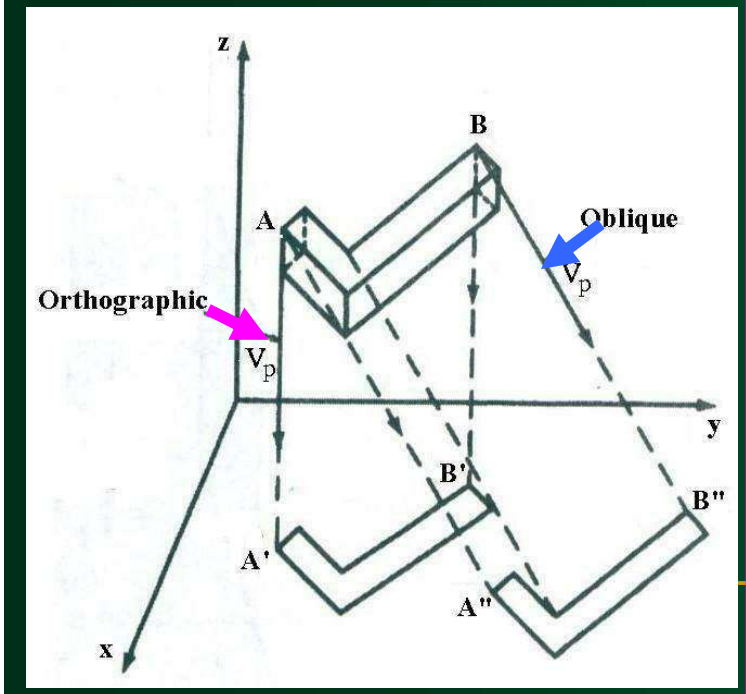
parallel
(orthographic)

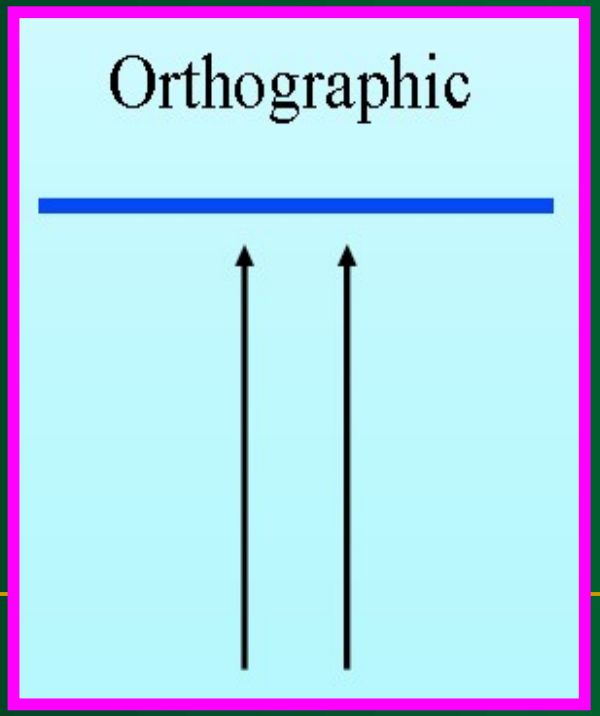
perspective

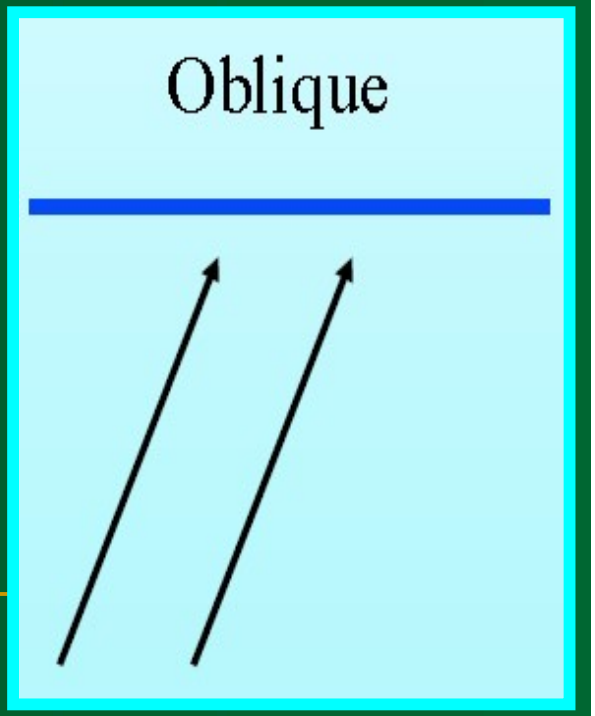
Parallel Projection

Parallel Projection

- **Projection vector**: Defines the direction for the projection lines (projectors).
- ***Orthographic Projection***: Projectors (projection vectors) are **perpendicular** to the projection plane.
- ***Oblique Projection***: Projectors (projection vectors) are ***not*** perpendicular to the projection plane.



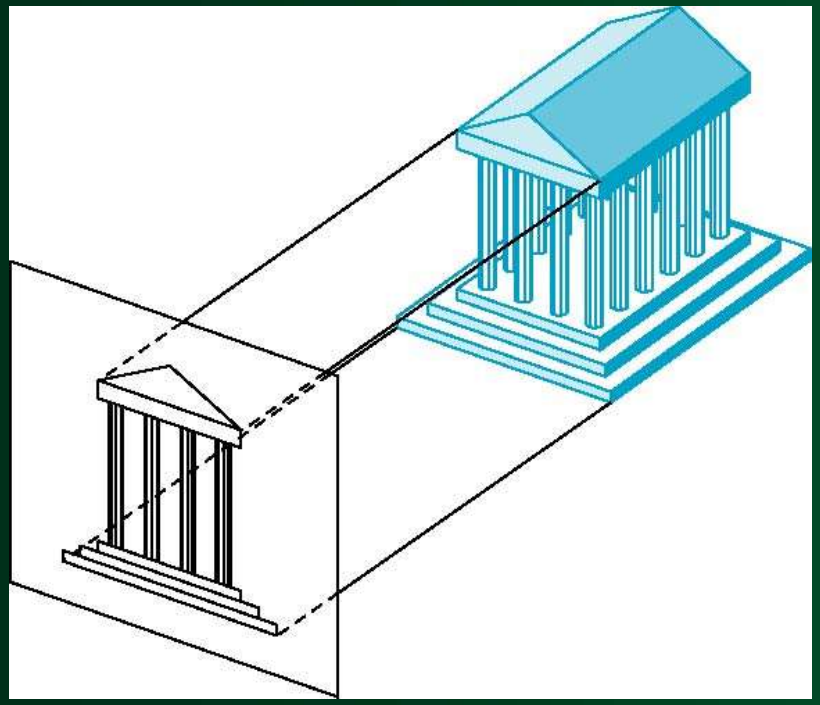


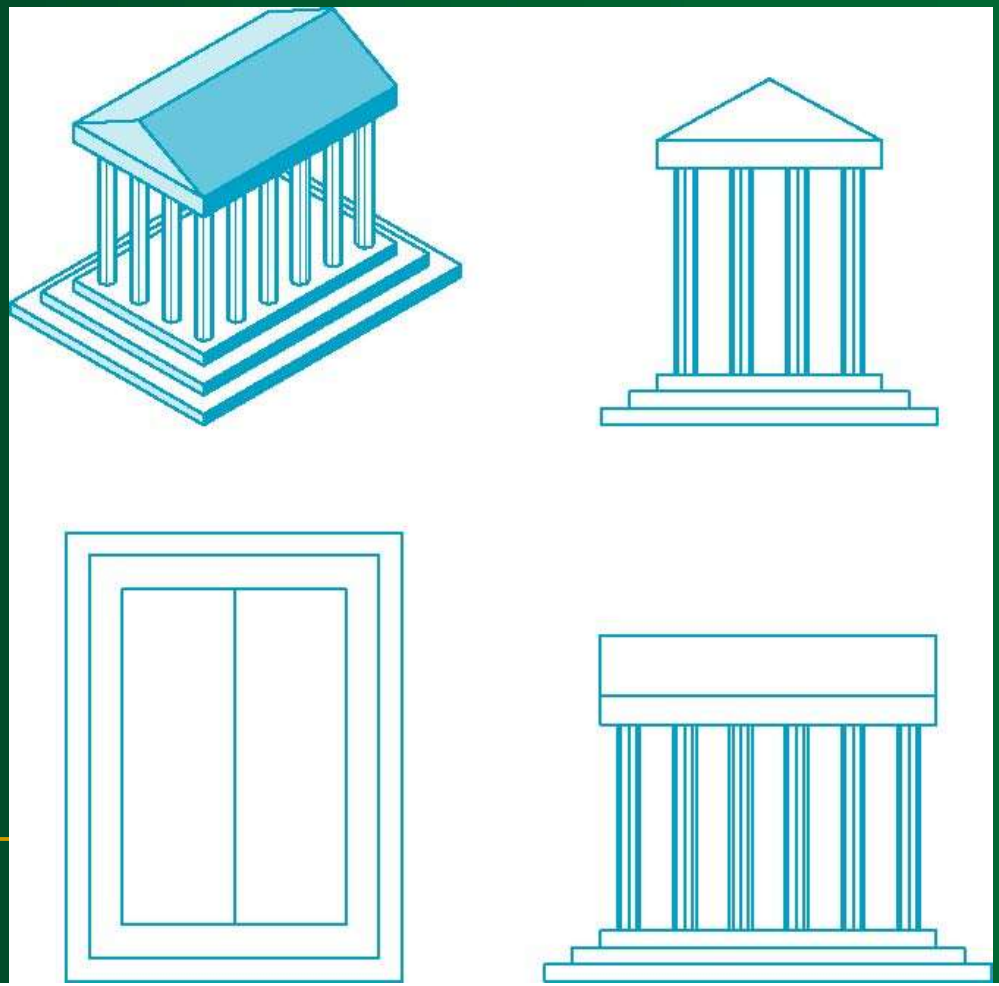


Orthographic Parallel Projection

Orthographic Parallel Projection

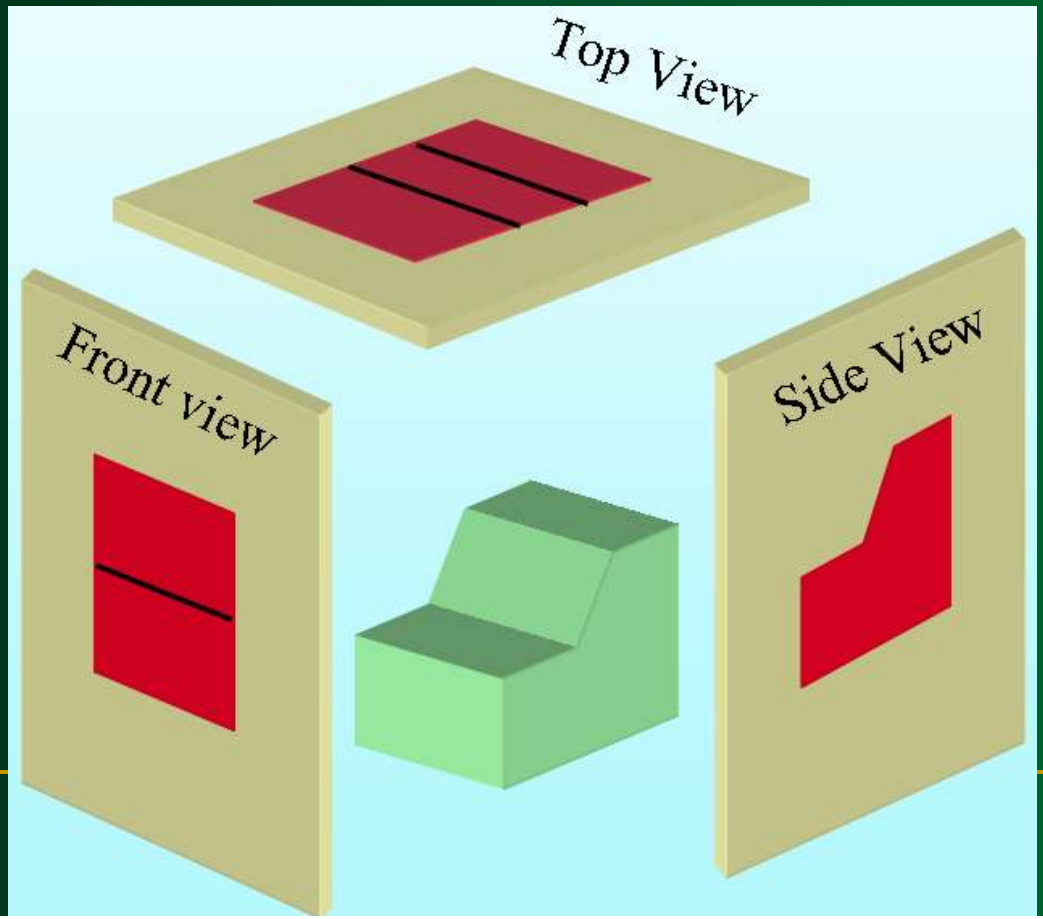
- Orthographic projection used to produce the **front**, **side**, and **top** views of an object.
- Engineering and architectural drawings employ orthographic projections.



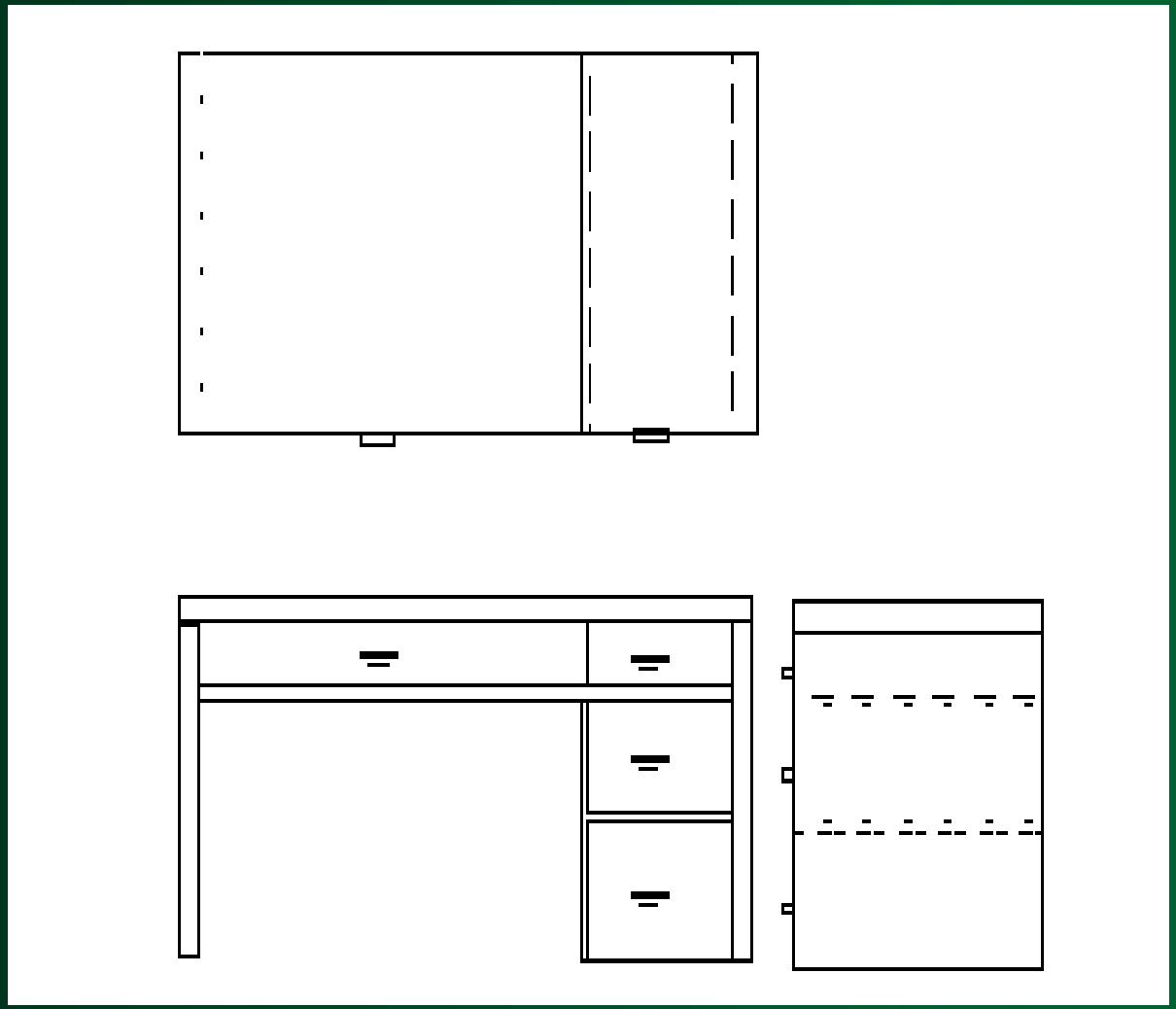


Orthographic Parallel Projection

- ***Front***, ***side***, and ***rear*** orthographic projections of an object are called ***elevations***.
- ***Top*** orthographic projection is called a ***plan*** view.



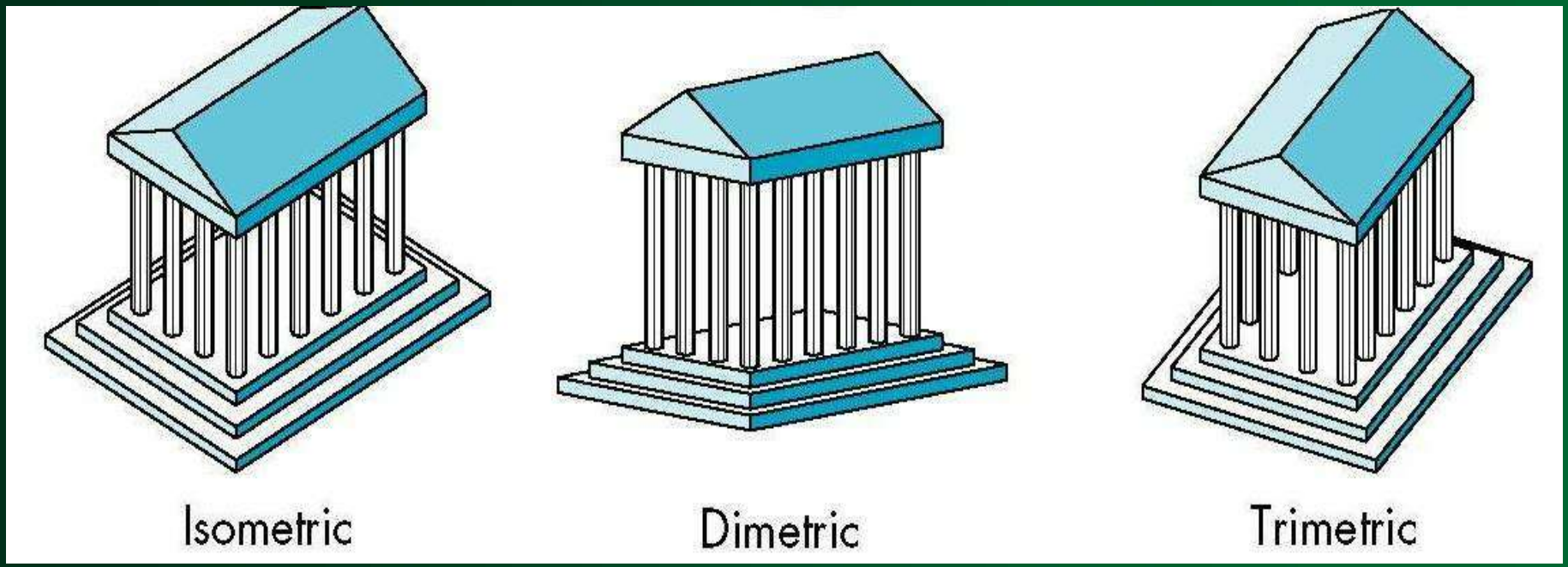
Orthographic Parallel Projection



Multi View Orthographic

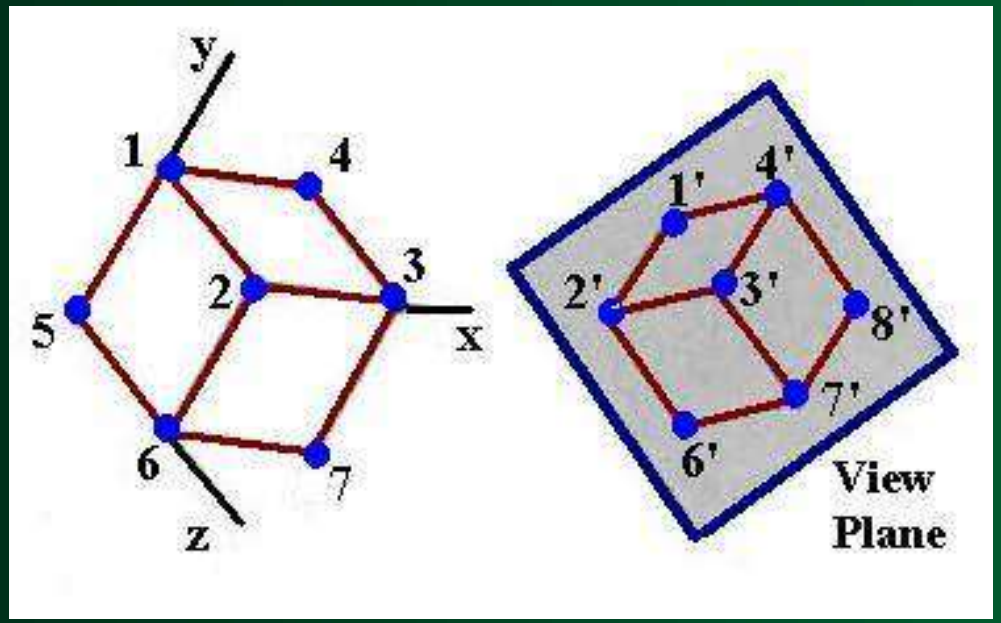
Orthographic Parallel Projection

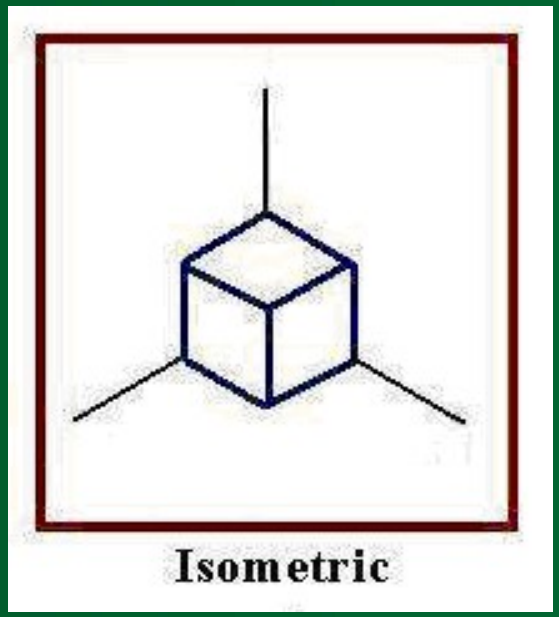
- ***Axonometric orthographic*** projections display more than one face of an object.



Orthographic Parallel Projection

- ***Isometric Projection***: Projection plane intersects each coordinate axis in which the object is defined (principal axes) at the same distant from the origin.
- Projection vector makes equal angles with all of the **three principal axes**.

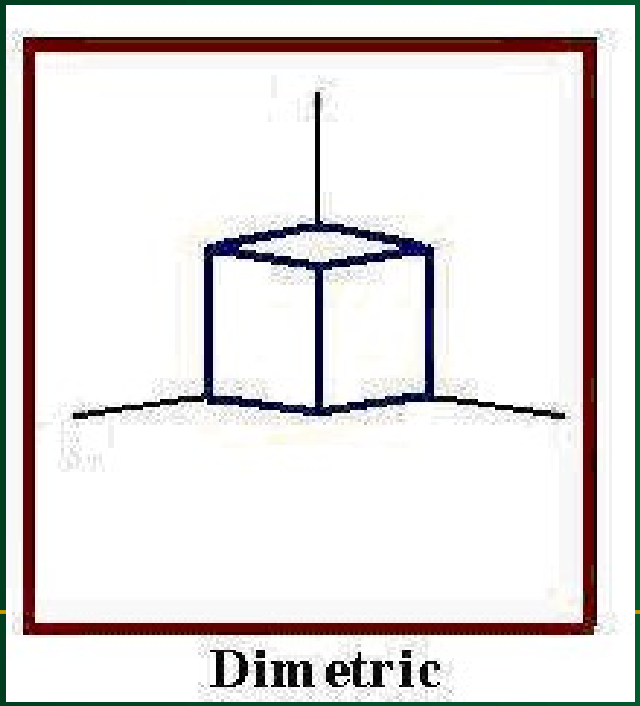




Isometric projection is obtained by **aligning** the **projection vector** with the **cube diagonal**.

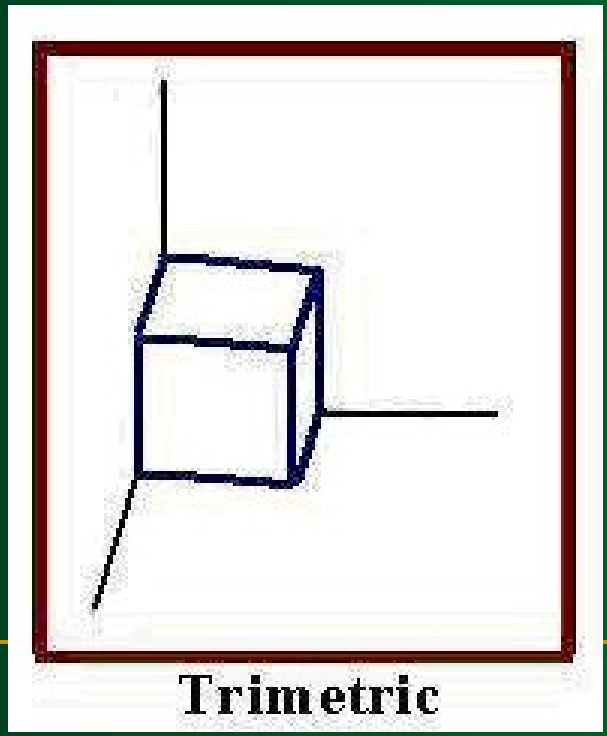
Orthographic Parallel Projection

- ***Dimetric Projection***: Projection vector makes **equal angles** with exactly **two** of the principal axes.

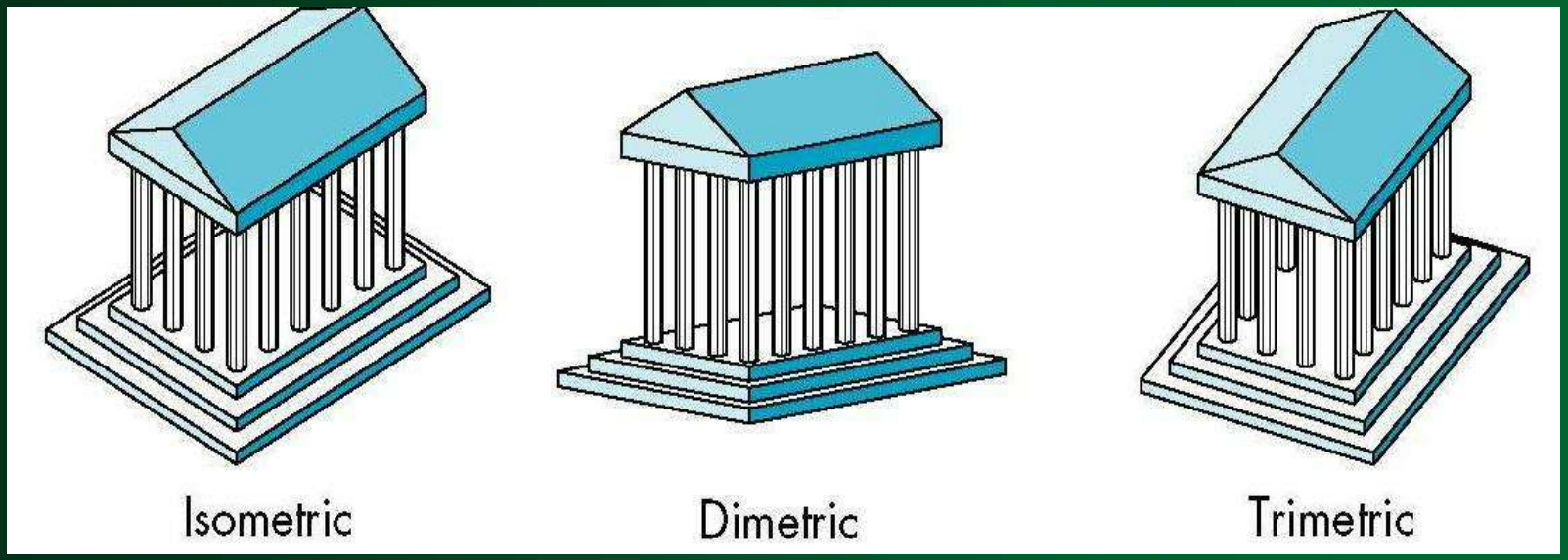


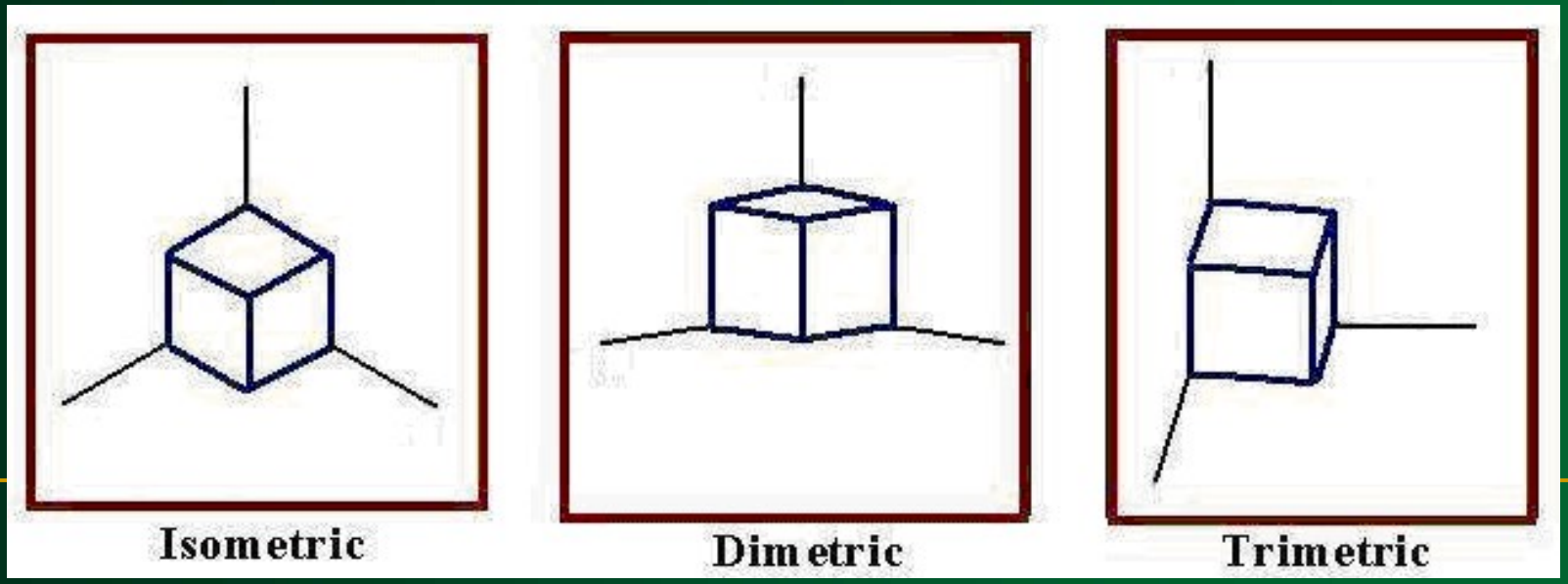
Orthographic Parallel Projection

- ***Trimetric Projection***: Projection vector makes **unequal angles** with the three principal axes.



Orthographic Parallel Projection

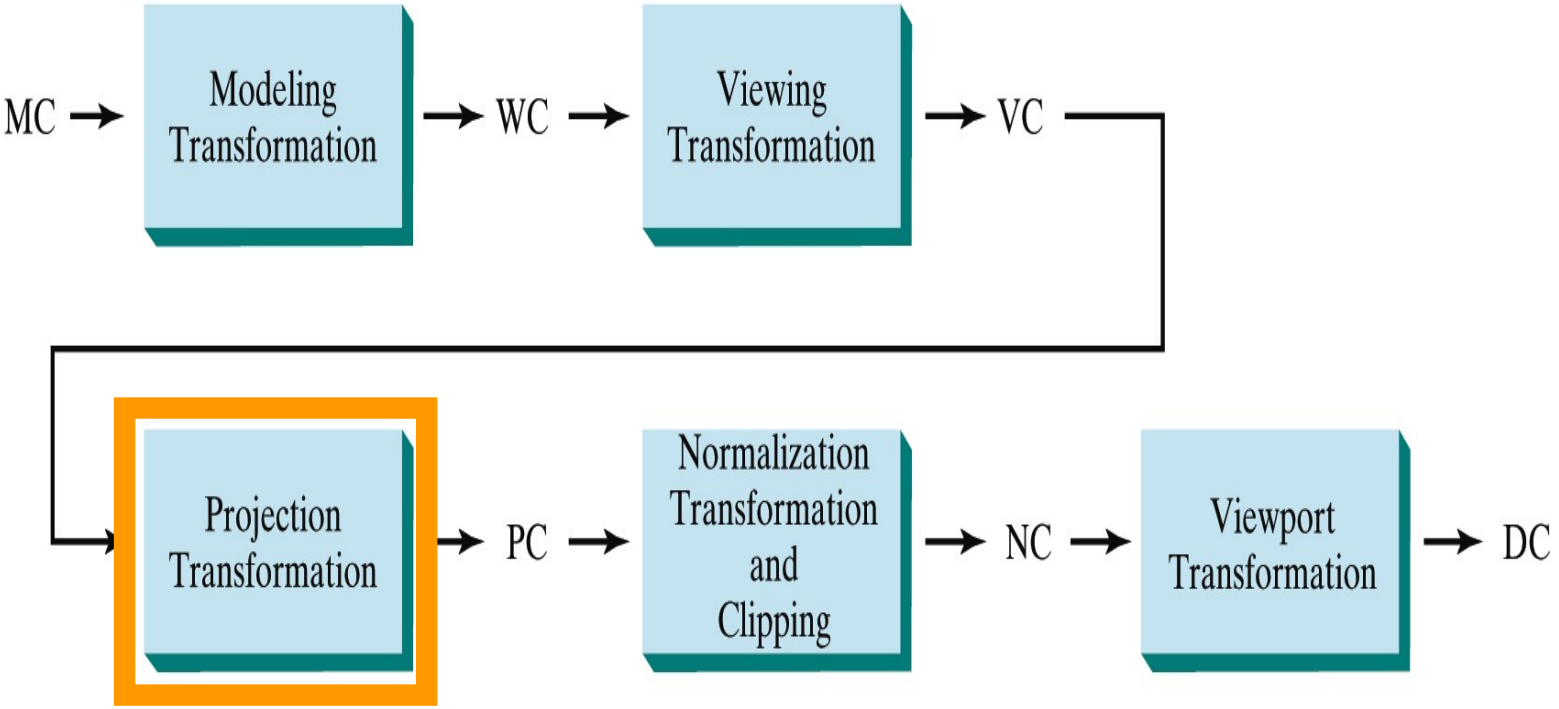




Orthographic Parallel Projection Transformation

Orthographic Parallel Projection Transformation

- Convert the **viewing coordinate** description of the scene to coordinate positions on the **Orthographic parallel projection plane**.



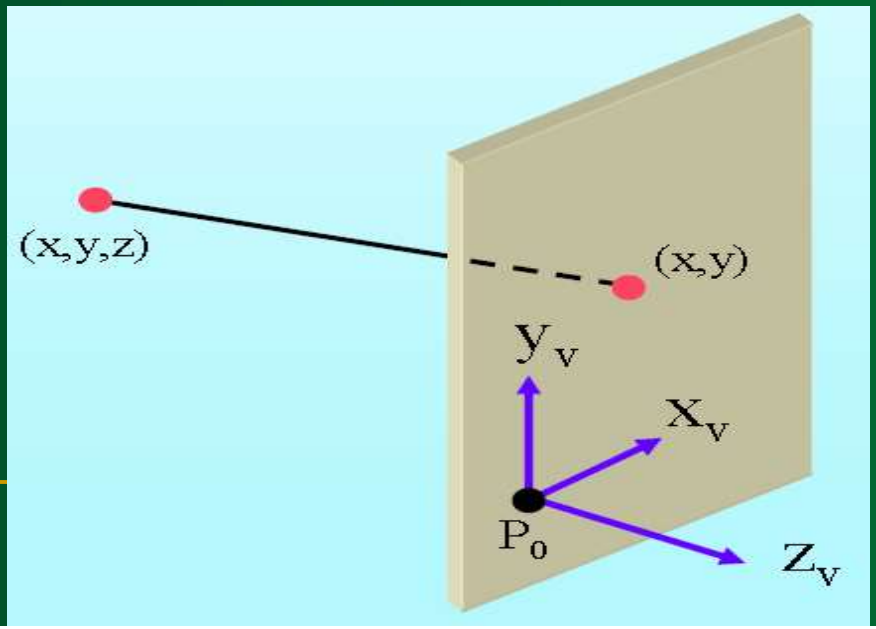
Orthographic Parallel Projection Transformation

- Since the view plane is placed at position z_{vp} along the z_v axis. Then any point (x,y,z) in viewing coordinates is transformed to projection coordinates as:

$$x_p = x, \quad y_p = y$$

The original z-coordinate value is preserved for the depth information.

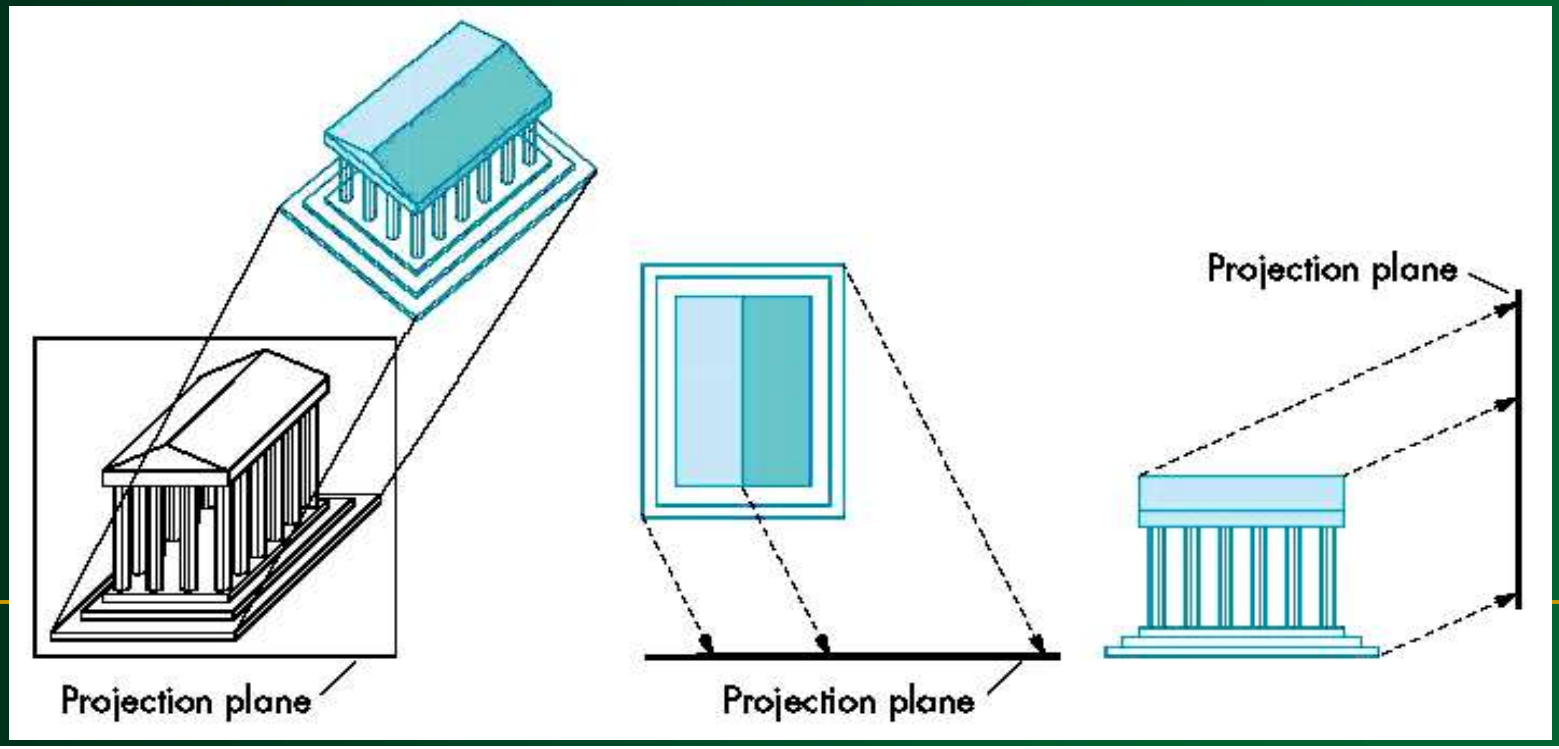
$$\mathbf{M}_{Orthographic\ \ Parallel} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Oblique
Parallel Projection

Oblique Parallel Projection

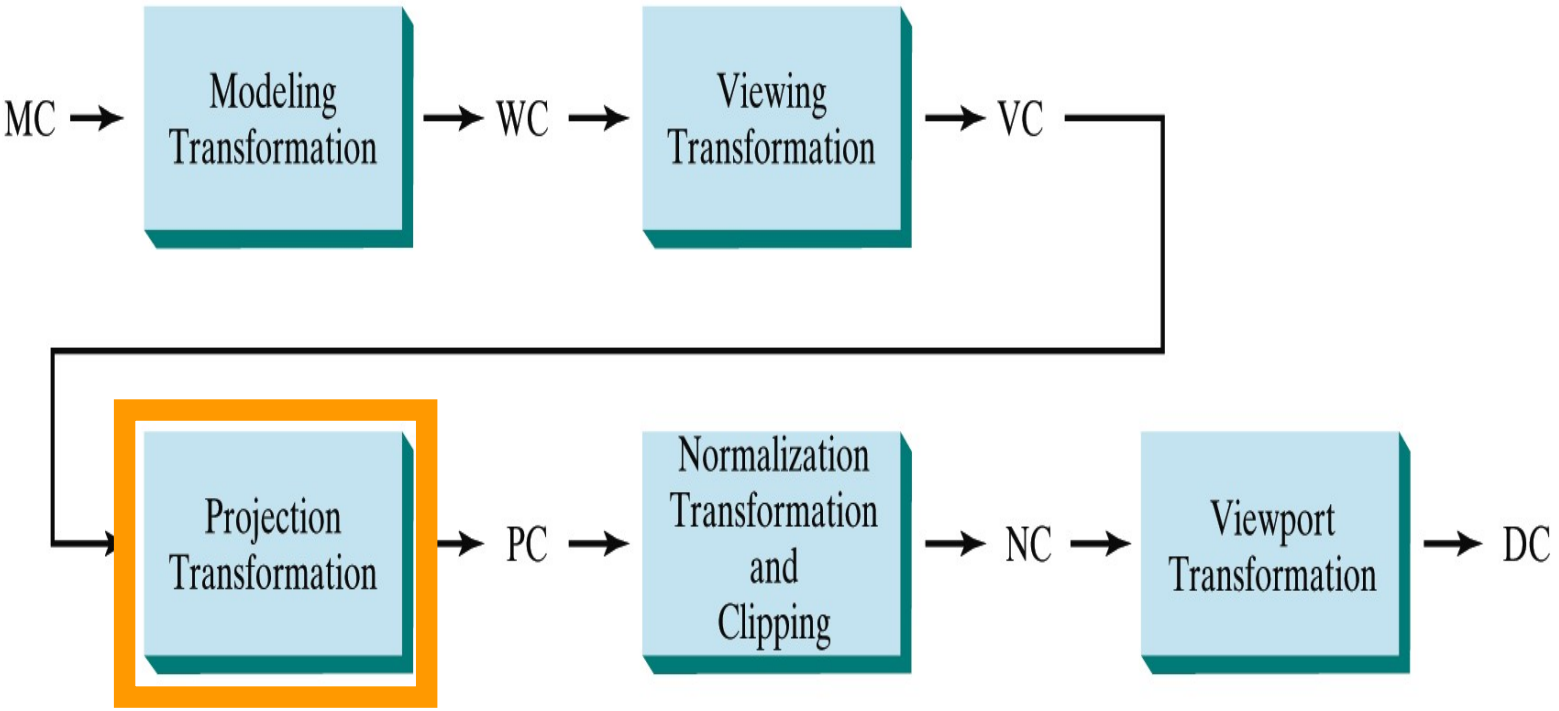
- Projections are **not** perpendicular to the viewing plane.
- Angles and lengths are preserved for faces parallel to the plane of projection.
- Preserves 3D nature of an object.



Oblique Parallel Projection Transformation

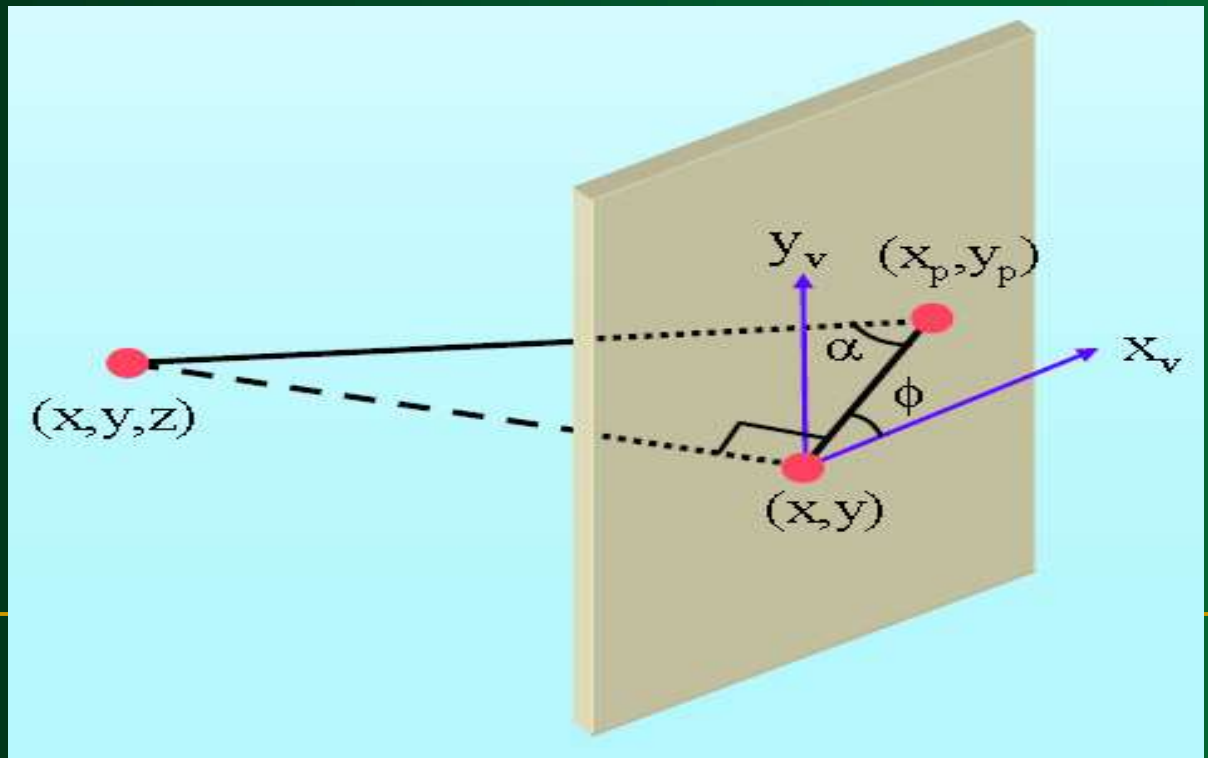
Oblique Parallel Projection Transformation

- Convert the **viewing coordinate** description of the scene to coordinate positions on the **Oblique parallel projection plane**.



Oblique Parallel Projection

- Point (x,y,z) is projected to position (x_p,y_p) on the view plane.
- Projector (oblique) from (x,y,z) to (x_p,y_p) makes an angle α with the line (of length L) on the projection plane that joins (x_p,y_p) and (x,y).
- Line L is at an angle φ with the horizontal direction in the projection plane.



Oblique Parallel Projection

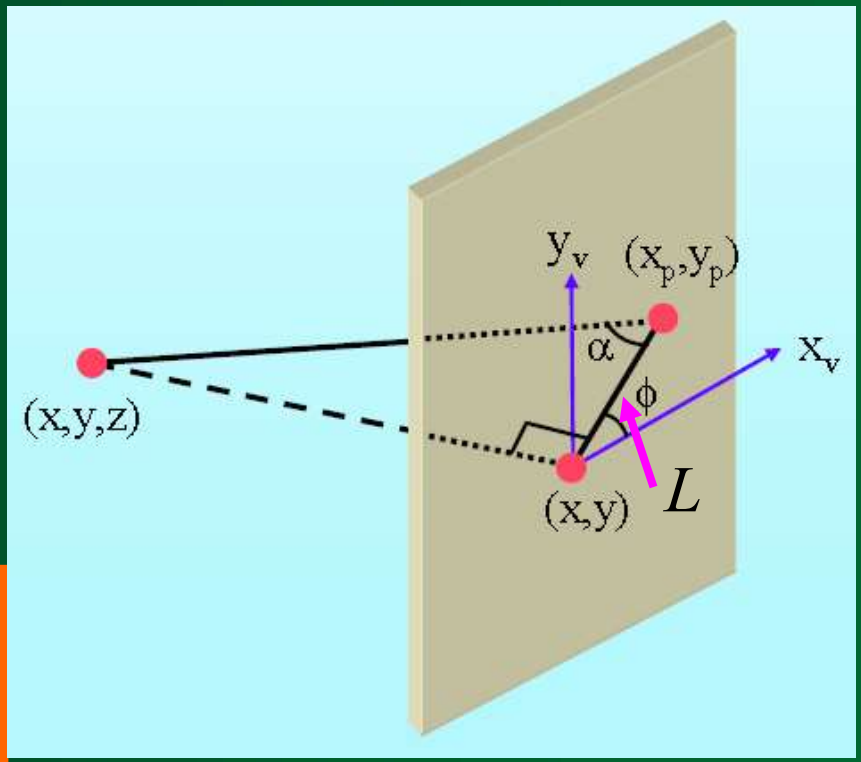
$$x_p = x + L\cos\varphi$$

$$y_p = y + L\sin\varphi$$

$$\tan\alpha = \frac{z}{L} \qquad L = \frac{z}{\tan\alpha} = zL_1$$

$$x_p = x + z(L_1\cos\varphi)$$

$$y_p = y + z(L_1\sin\varphi)$$



$$\mathbf{M}_{Parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\varphi & 0 \\ 0 & 1 & L_1\sin\varphi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

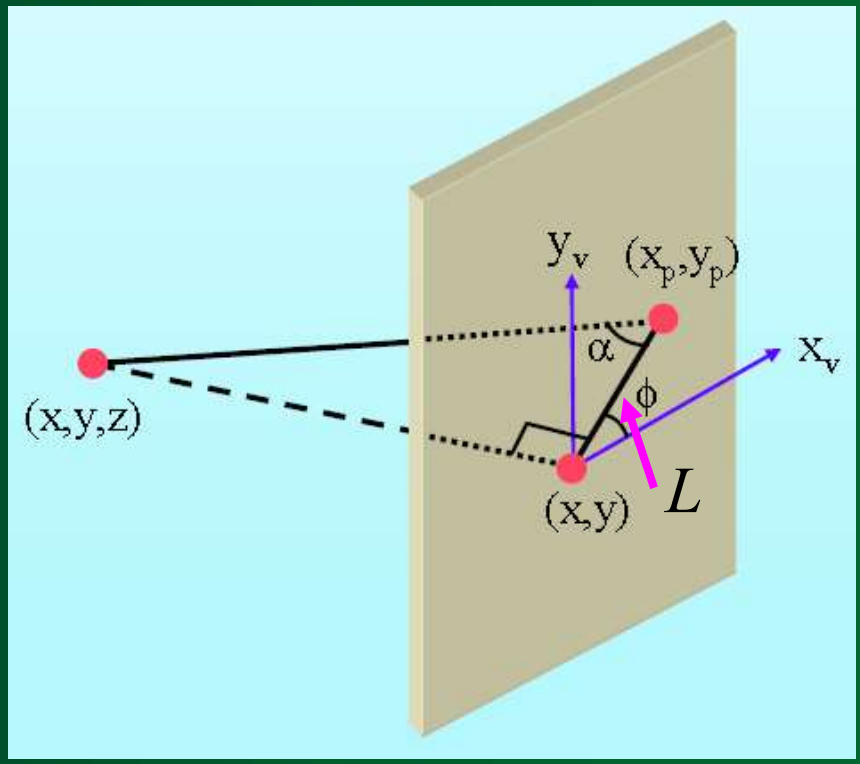
Oblique Parallel Projection

Orthographic Projection:

$$L_1 = 0$$

$$\alpha = 90^\circ$$

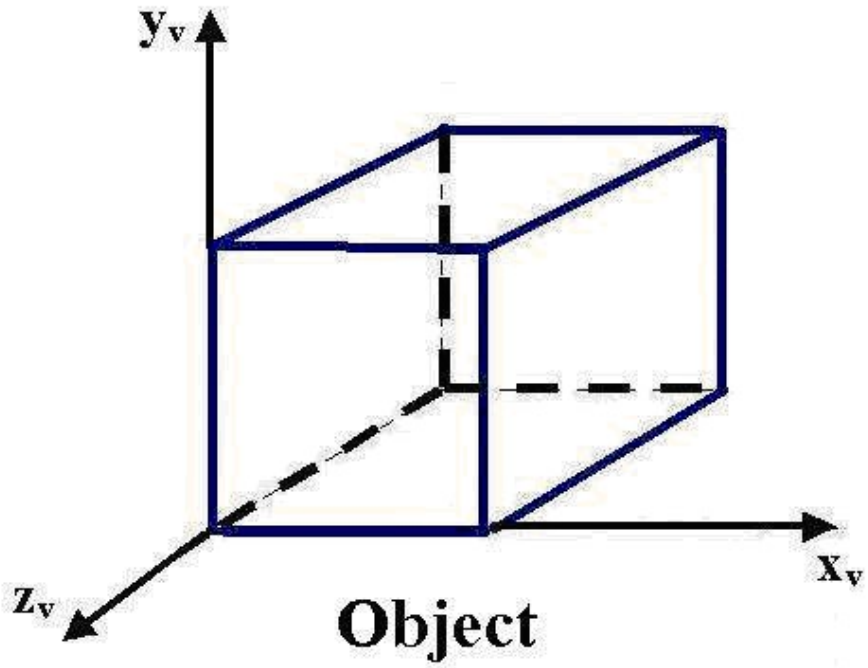
$$x_p = x, \quad y_p = y$$



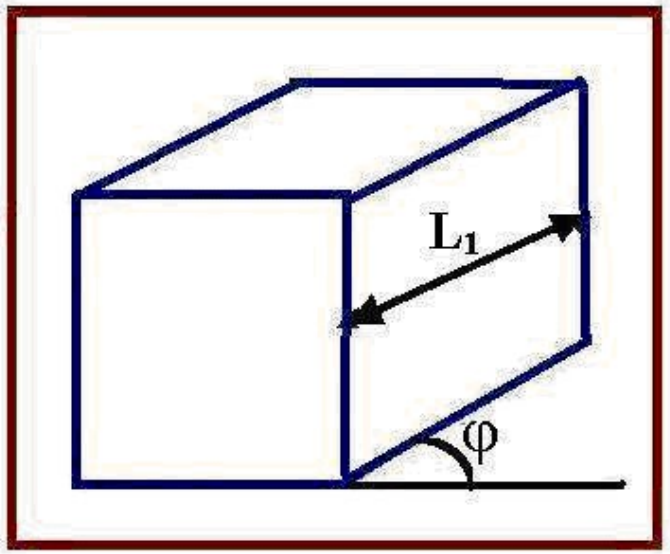
$$\mathbf{M}_{Orthographic\ \ Parallel} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oblique Parallel Projection

- **Angles**, **distances**, and **parallel lines** in the plane are projected accurately.



Object



Projection on the Viewing Plane

Cavalier Projection

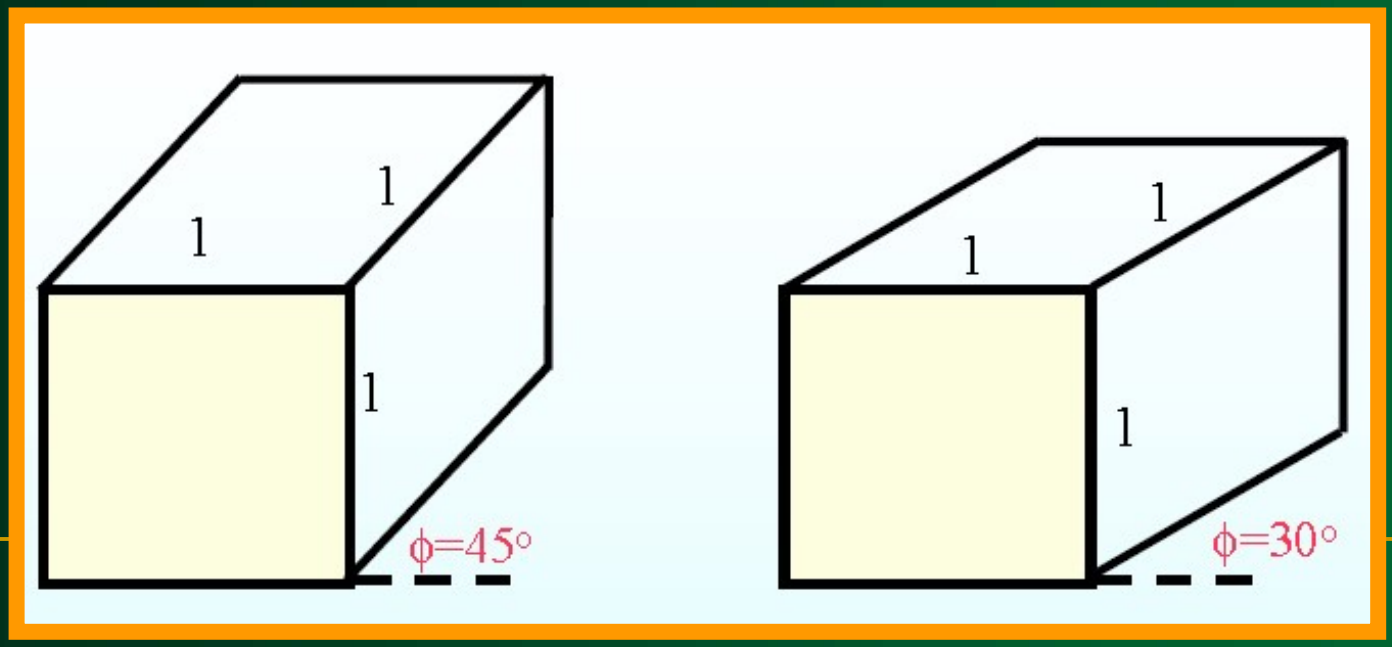
Cavalier Projection:

$\phi = 30^\circ \quad and \quad 45^\circ$

$$\tan\alpha = 1$$

$$\alpha = 45^\circ$$

- Preserves lengths of lines perpendicular to the viewing plane.
- 3D nature can be captured but shape seems distorted.
- Can display a combination of front, and side, and top views.



Cabinet Projection

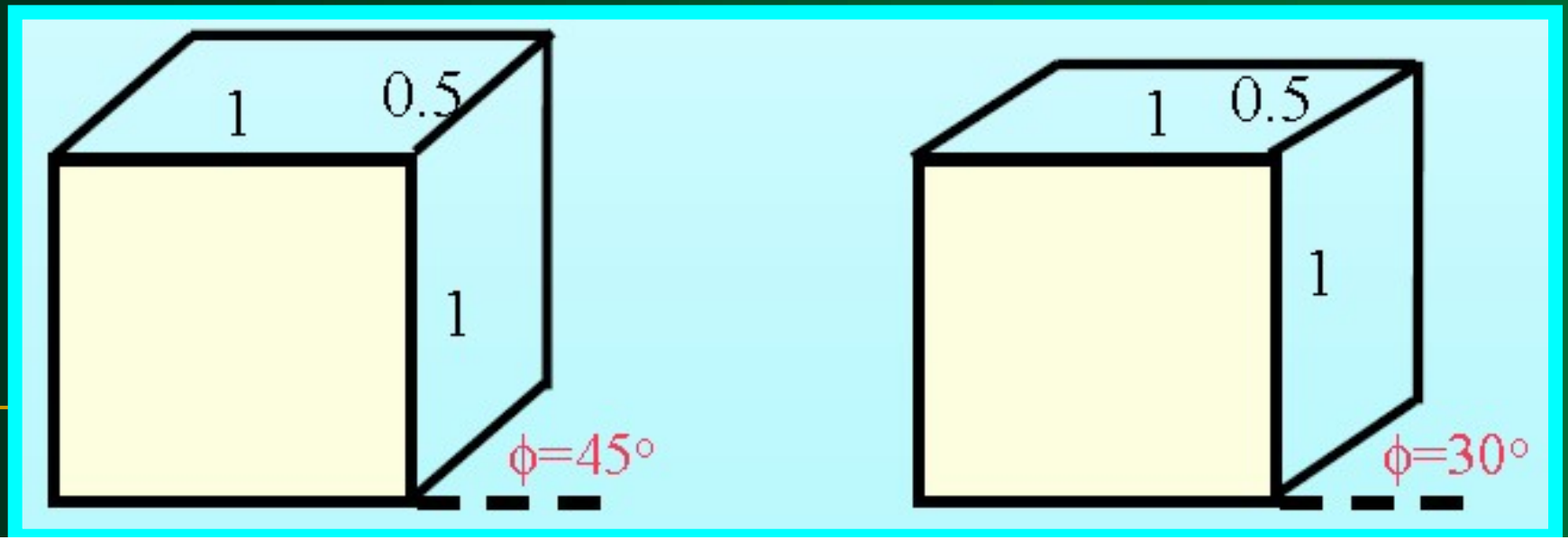
Cabinet Projection:

$\phi = 30^\circ$ *and* 45°

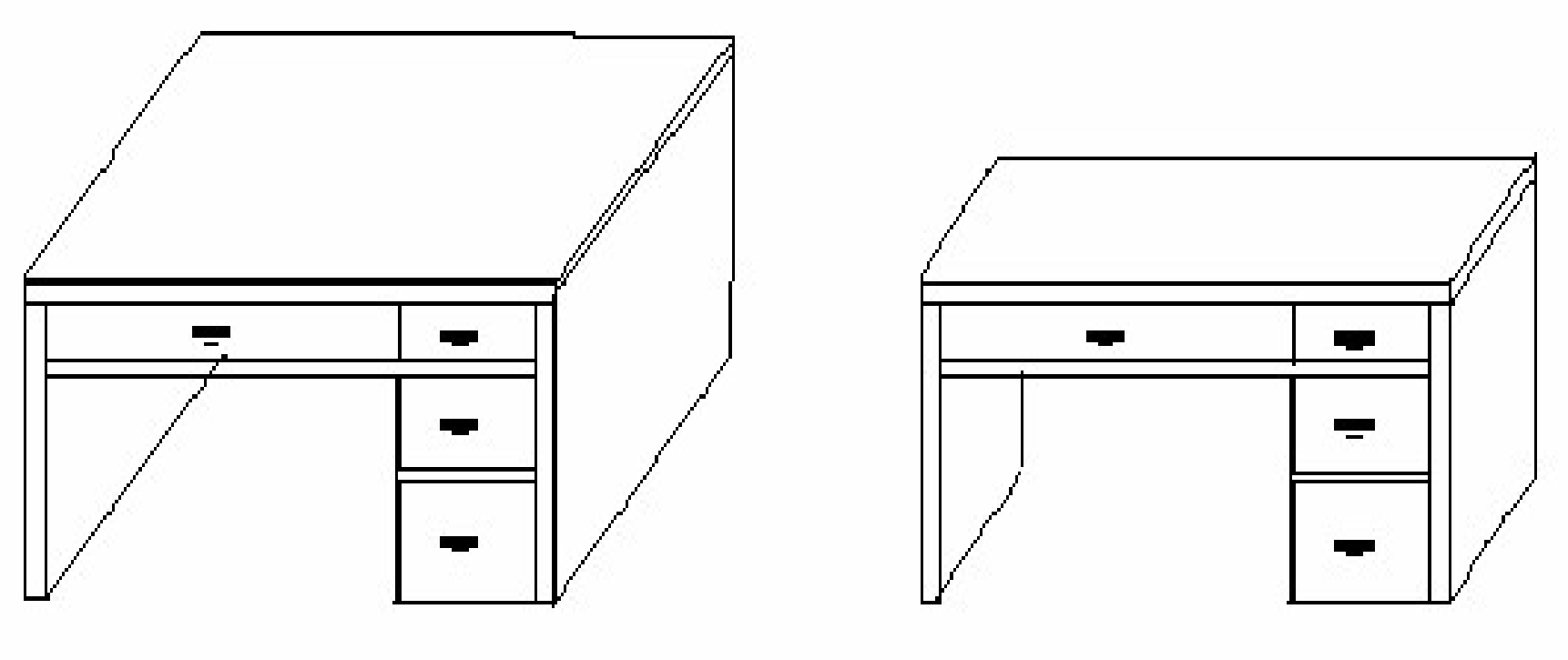
$\tan\alpha = 2$

$\alpha \approx 63.4^\circ$

- Lines perpendicular to the viewing plane are projected at ½ of their **length**.
- A more realistic view than the cavalier projection.
- Can display a combination of front, and side, and top views.



Cavalier & Cabinet Projection

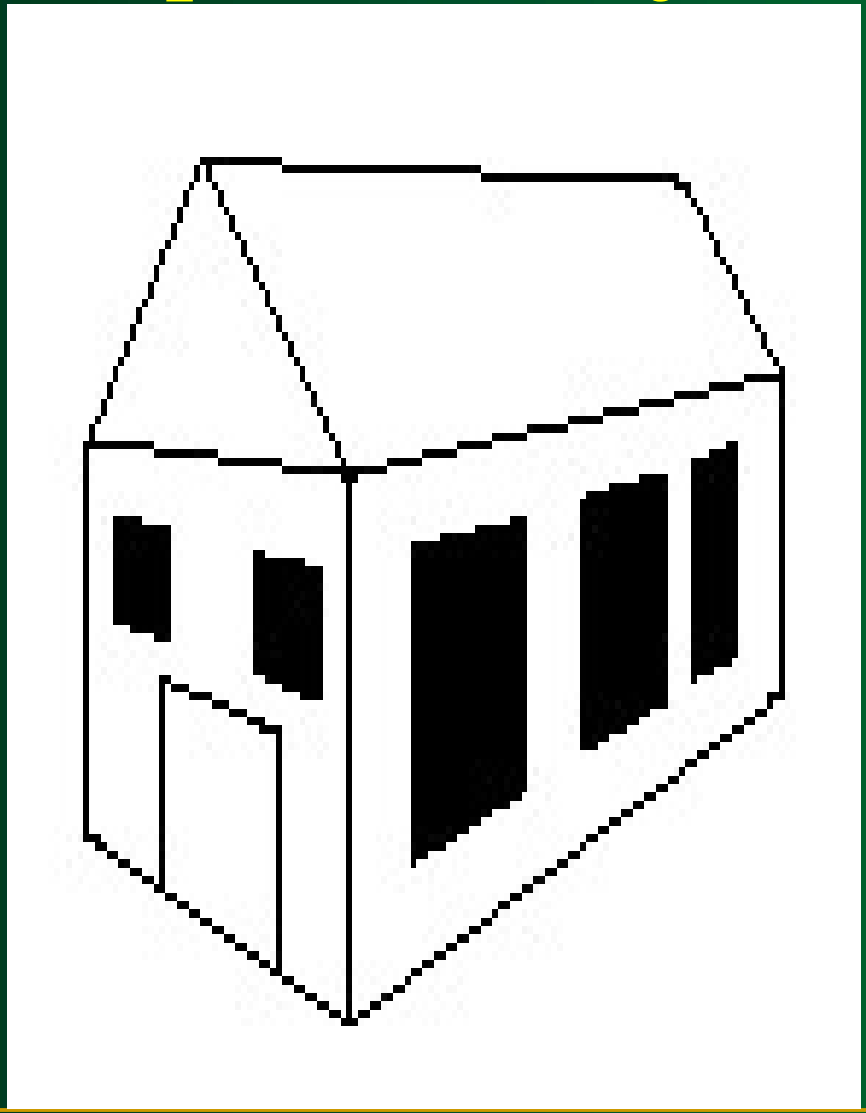


Cavalier

Cabinet

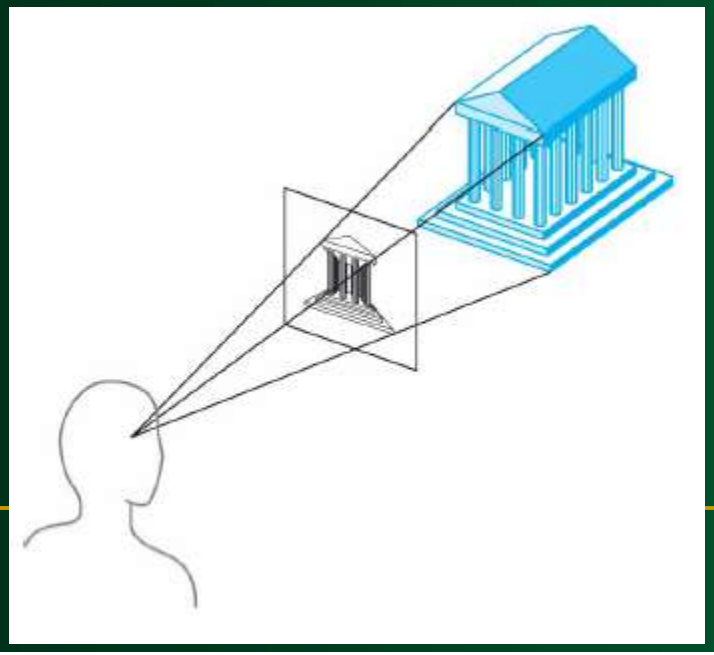
Perspective Projection

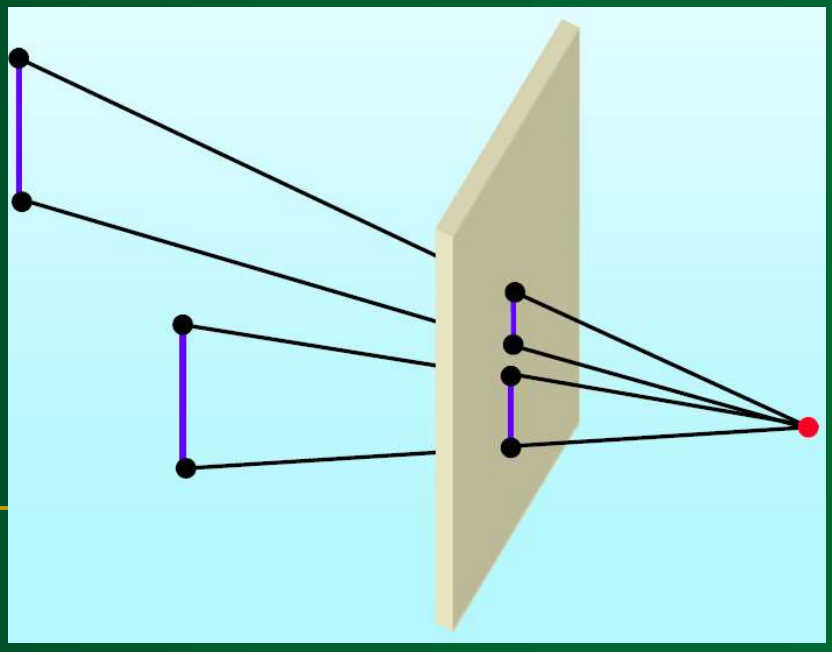
Perspective Projection



Perspective Projection

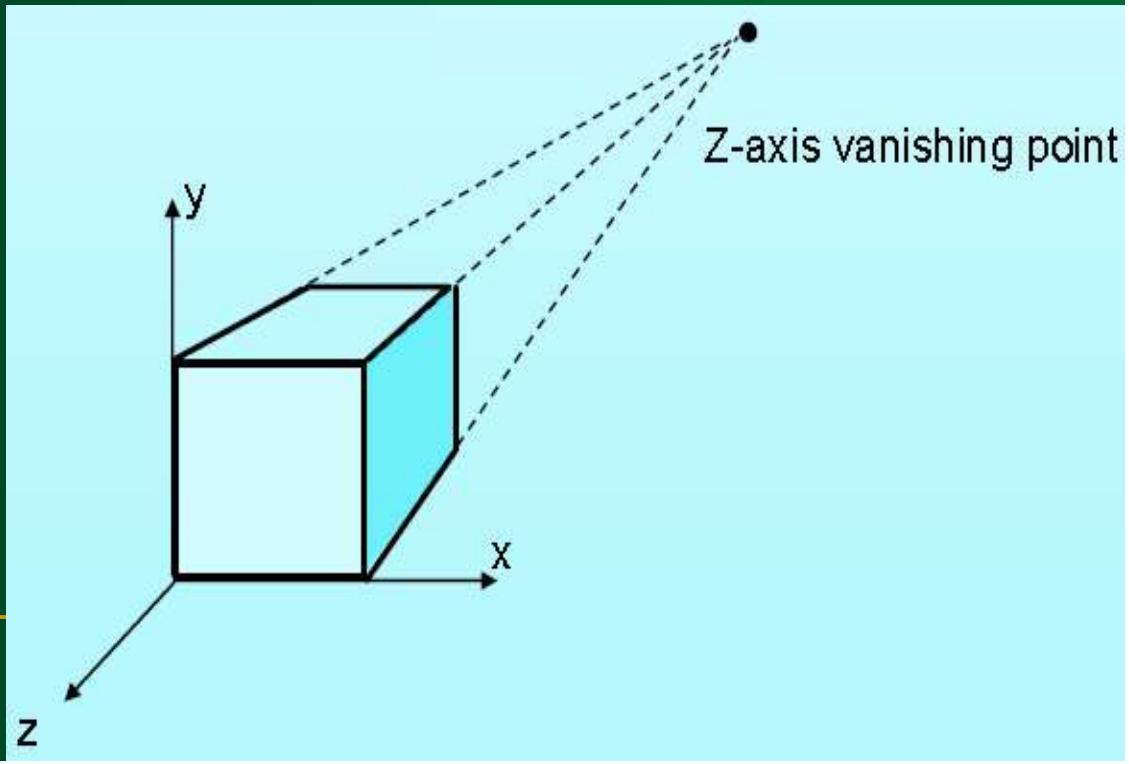
- In a perspective projection, the center of projection is at a finite distance from the viewing plane.
- Produces realistic views but does not preserve relative proportion of objects
- The size of a projection object is inversely proportional to its distance from the viewing plane.





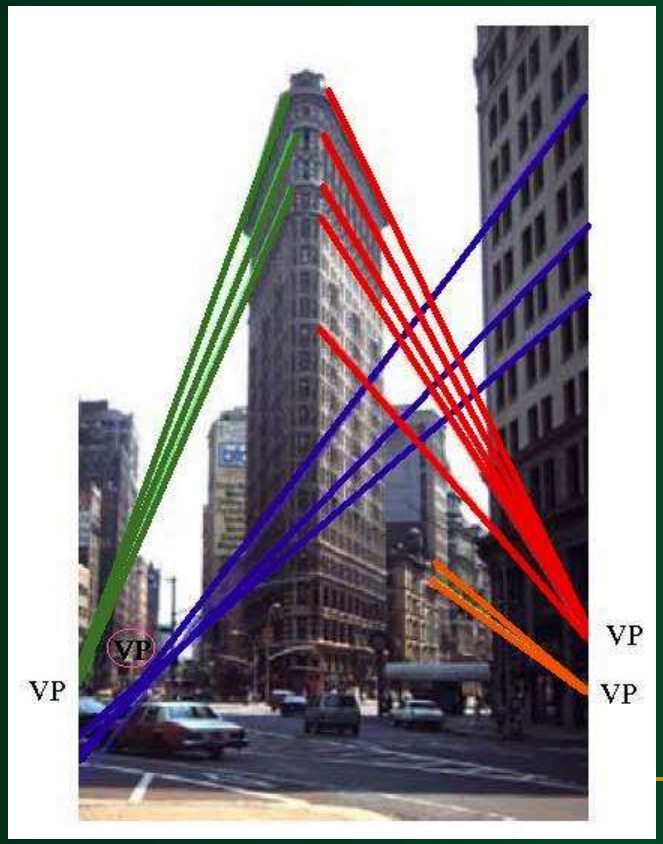
Perspective Projection

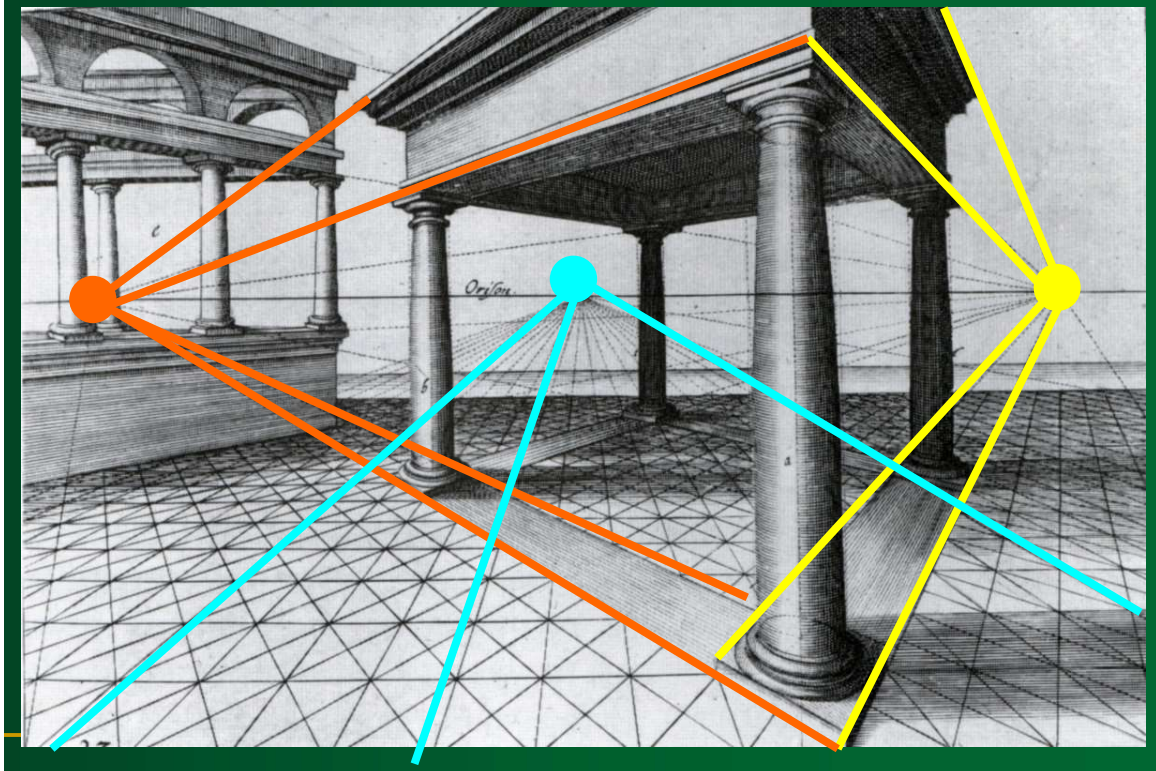
- Parallel lines that are not parallel to the viewing plane, **converge** to a ***vanishing point***.
- A vanishing point is the projection of a point at infinity.



Vanishing Points

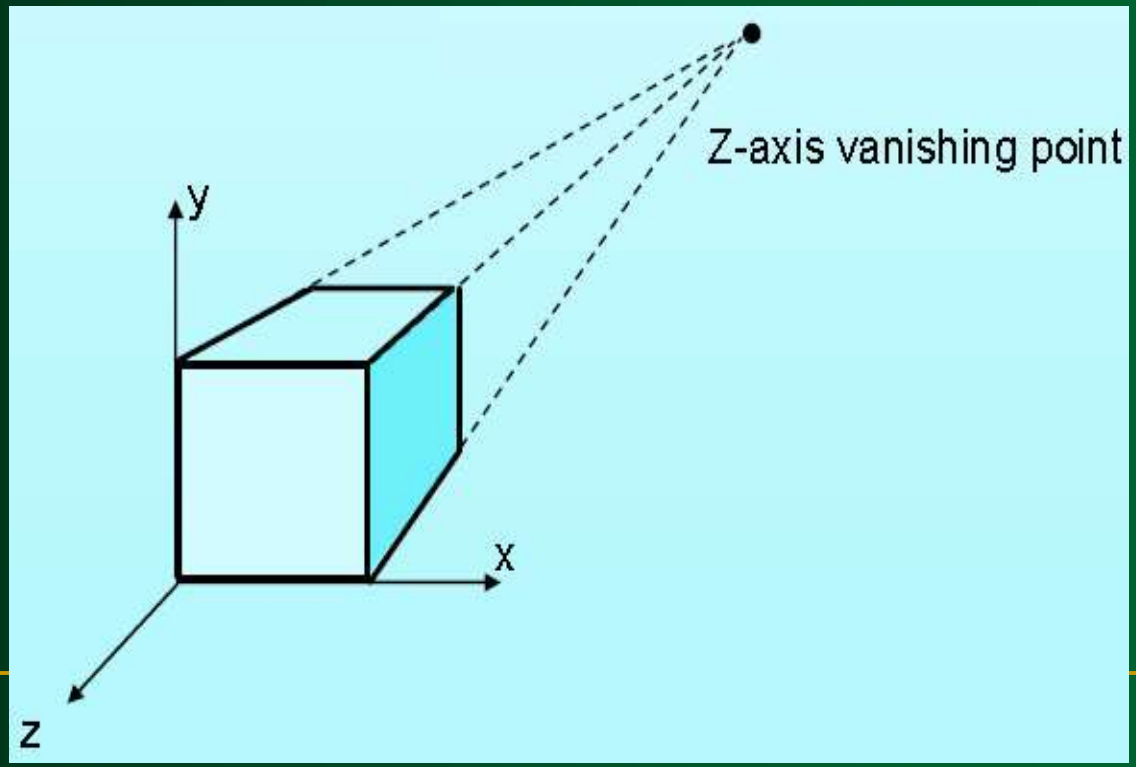
- Each set of projected parallel lines will have a separate vanishing points.





Perspective Projection

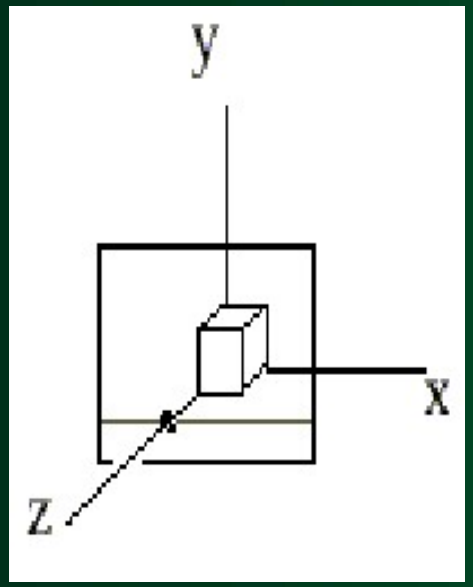
- The vanishing point for any set of lines that are parallel to one of the principal axes of an object is referred to as a **principal vanishing point**.
- We control the number of principal vanishing points (one, two, or three) with the orientation of the projection plane.

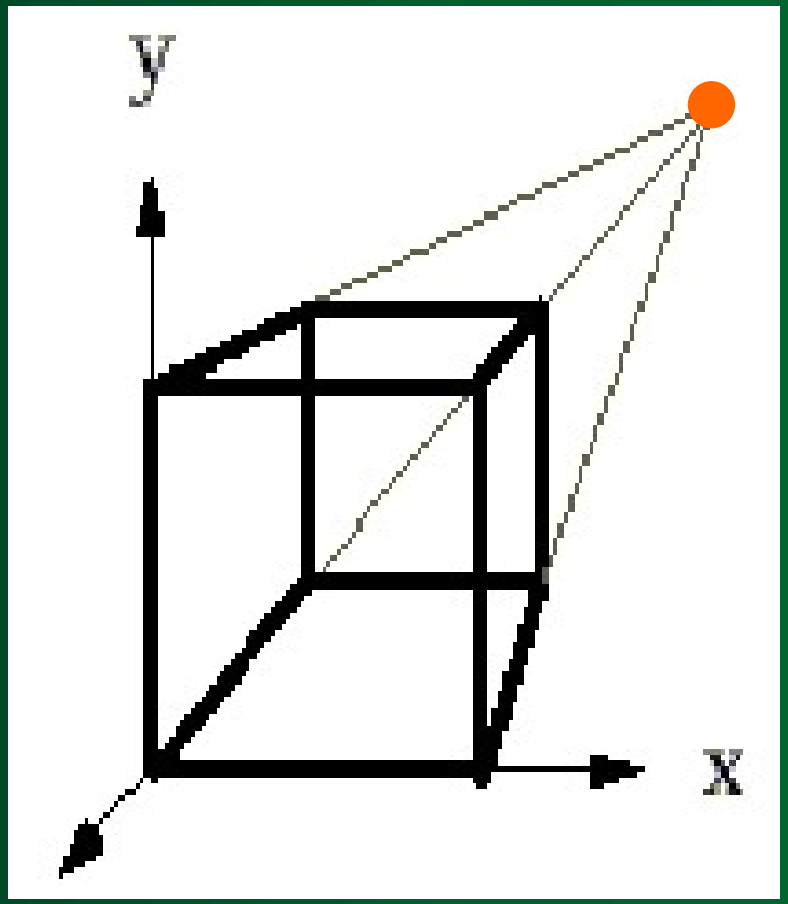


Perspective Projection

- The number of principal vanishing points in a projection is determined by the number of principal axes **intersecting** the view plane.

Perspective Projection

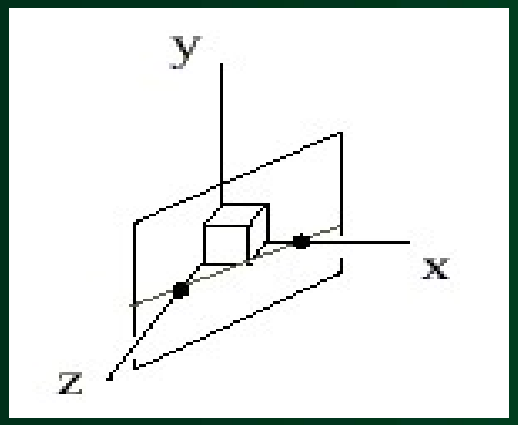


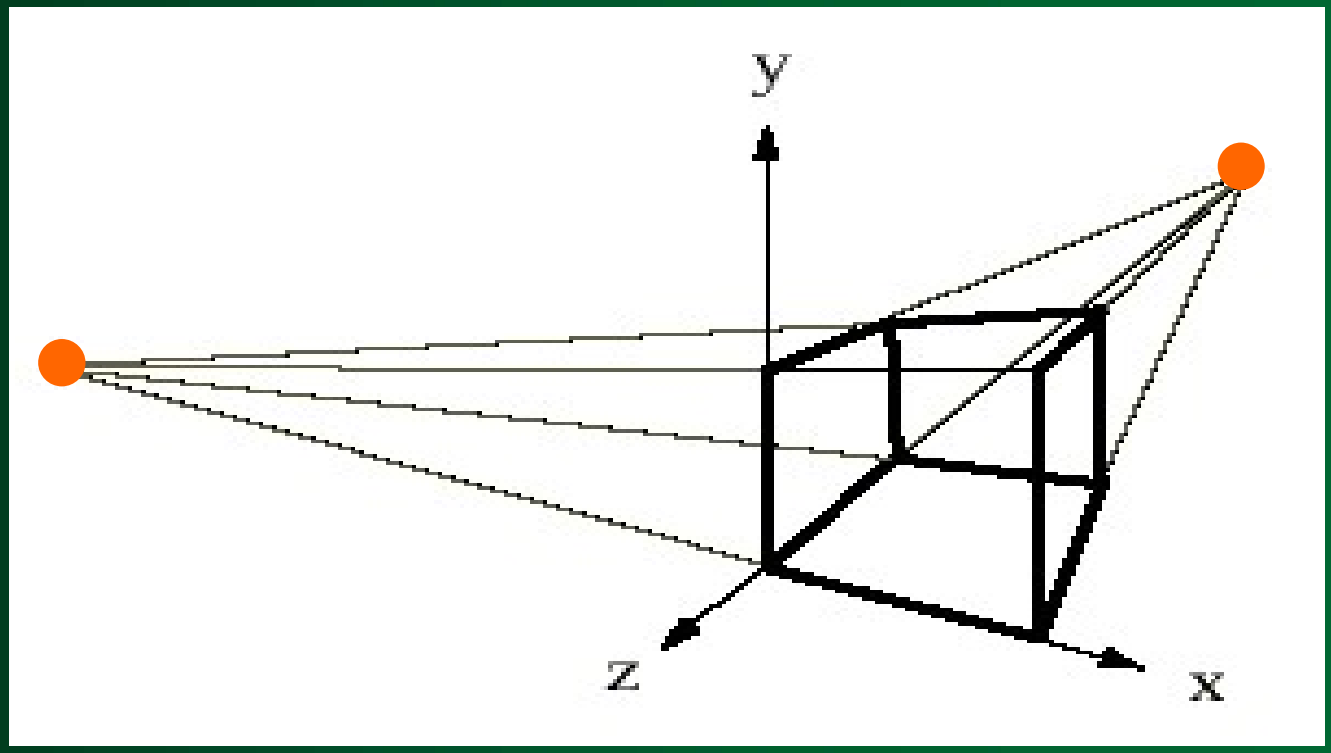


One Point Perspective

(z-axis vanishing point)

Perspective Projection

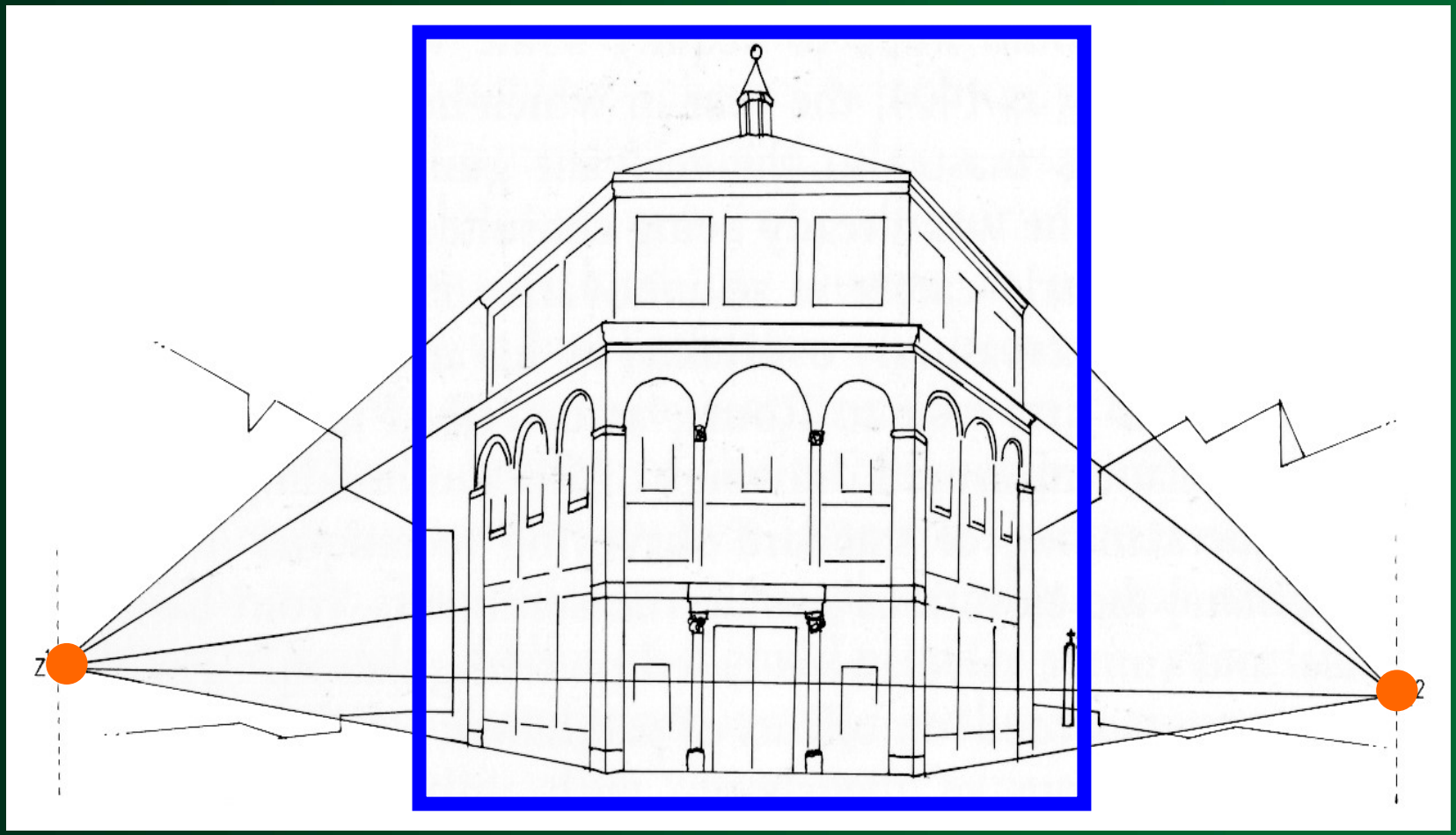




Two Point Perspective

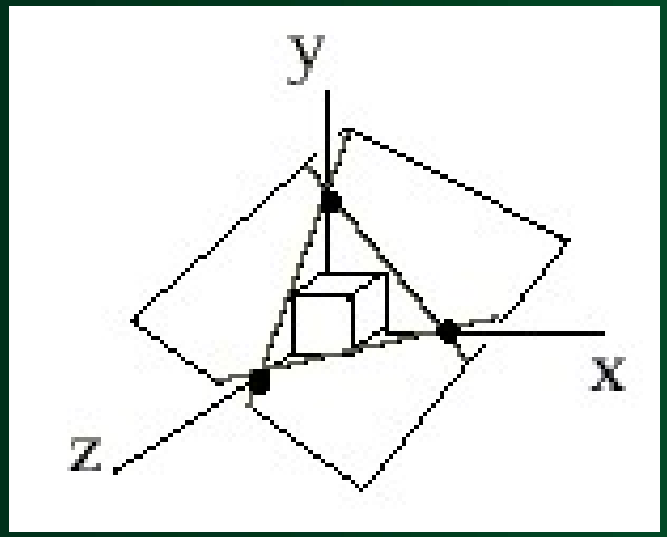
(z, and x-axis vanishing points)

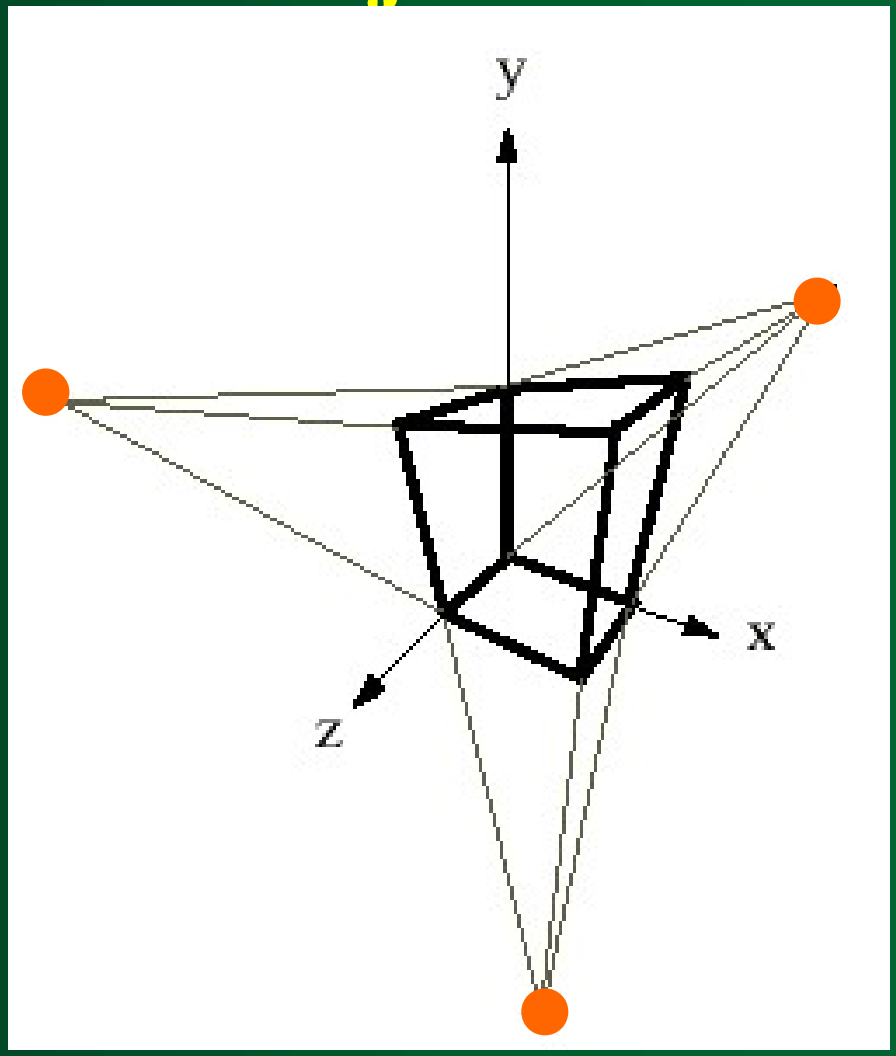
Perspective Projection



Two Point Perspective

Perspective Projection

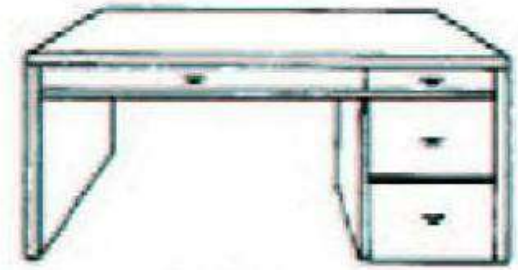




Three Point Perspective

(z, x, and y-axis vanishing points)

Perspective Projection



One-Point Perspective Projection

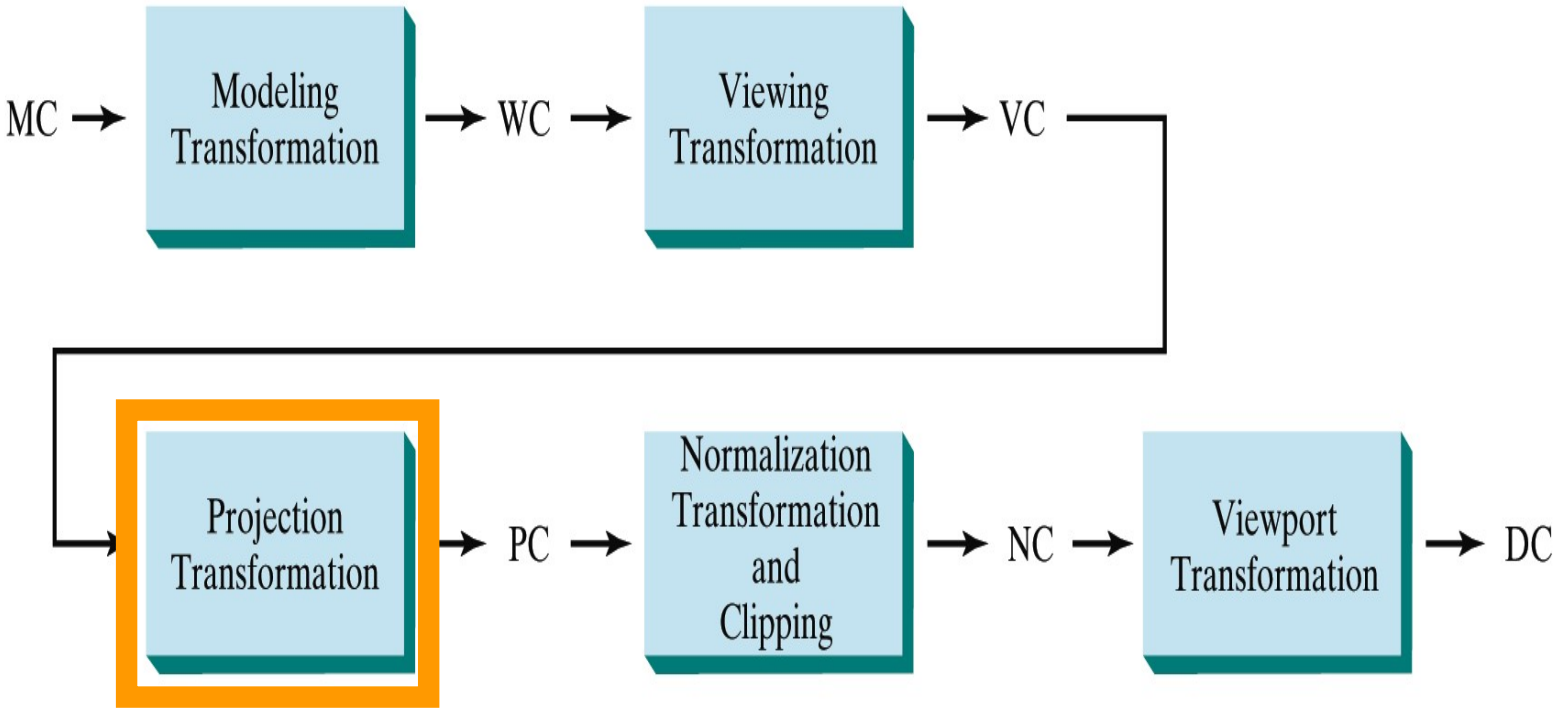
Two-Point Perspective Projection

Tree-Point Perspective Projection

Perspective Projection Transformation

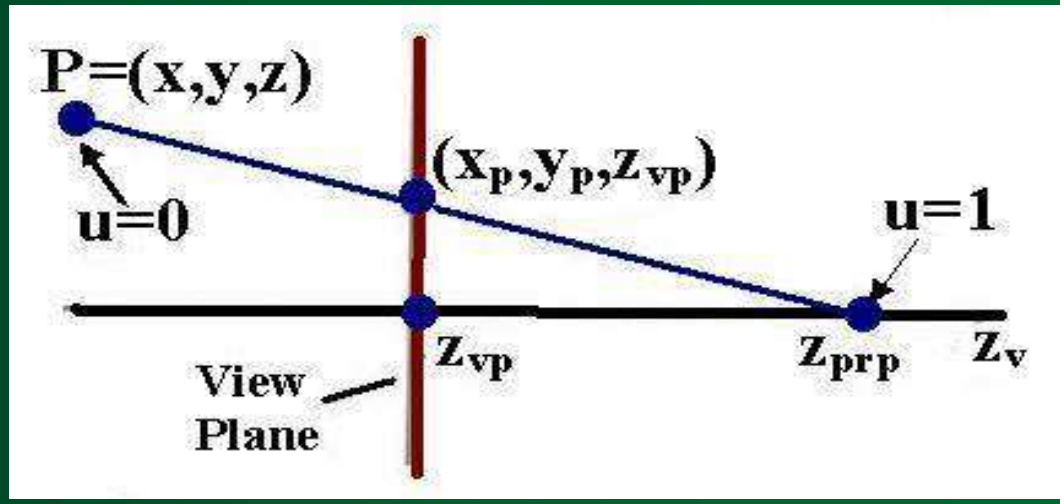
Perspective Projection Transformation

- Convert the **viewing coordinate** description of the scene to coordinate positions on the **perspective projection plane**.



Perspective Projection Transformation

- To obtain a perspective projection of a three dimensional object, transform points along the projection lines that meet a projection reference point.
- Set the projection reference point at position z_{prp} along the z_v axis.
- Set the view plane at z_{vp}.

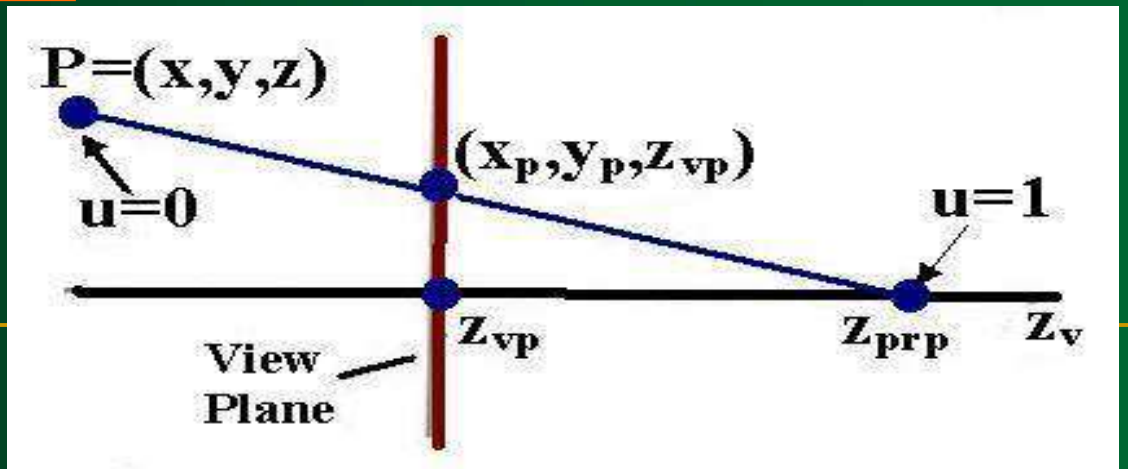


Perspective Projection Transformation

- Suppose the projection reference point at position z_{prp} along the z_v axis, and the view plane at z_{vp}.
- Equations of the perespective projection line in parametric form as

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

$$0 \le u \le 1$$



Perspective Projection Transformation

At u=0, x'=x ; y=y'; z=z' at p

At u=1 , x'=0; y'=0; z'=Zprp at Zprp

On the view plane: $z' = z_{vp}$

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

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

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

dp is the distance of the view plane from the projection reference point

Perspective Projection Transformation

The perspective projection transformation of three-dimensional can be represented using homogeneous coordinates: $x_p = x_h / h, \quad y_p = y_h / h$

The homogeneous factor is:

$$h = \frac{z_{prp} - z}{d_p}$$

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

Perspective Projection Transformation

Special Cases:If view plane is uv plane

$$z_{vp} = 0$$

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

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

$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) = x\left(\frac{1}{1 - z/z_{prp}}\right)$$

$$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) = y\left(\frac{1}{1 - z/z_{prp}}\right)$$

Perspective Projection Transformation

Special Cases: The projection reference point is at the viewing coordinate origin:

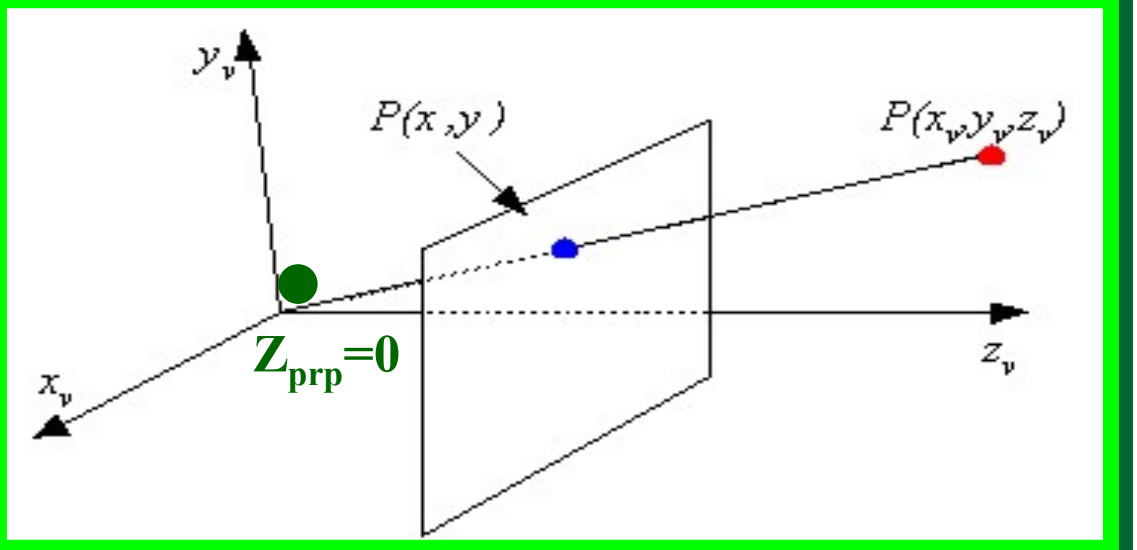
$$z_{prp} = 0$$

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

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

$$x_p = x\left(\frac{z_{vp}}{z}\right) = x\left(\frac{1}{z/z_{vp}}\right)$$

$$y_p = y\left(\frac{z_{vp}}{z}\right) = y\left(\frac{1}{z/z_{vp}}\right)$$



Exercise

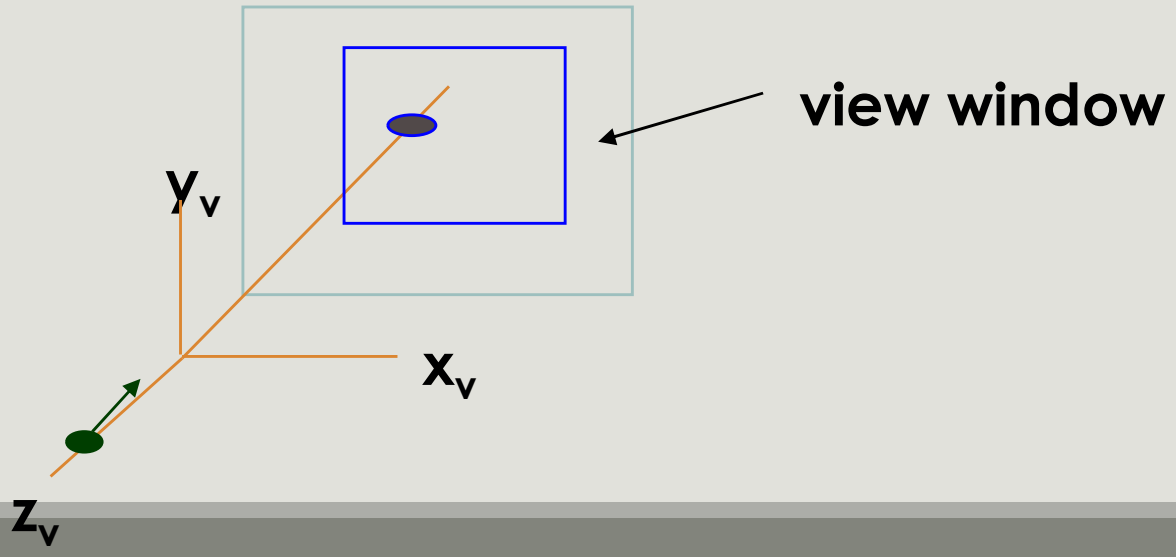
- Consider a 3D coordinate system where Y-axis is vertical and Z-axis is pointing towards the viewer. A line A(10,-10,10) B(10,-10,0) is viewed from the point P(0,0,20). Find where points A and B would be projected on the XY screen?

Summary

View Volumes and General Projection Transformations

View Volumes - View Window

- Type of lens in a camera is one factor which determines how much of the view is captured
  - wide angle lens captures more than regular lens
- Analogy in computer graphics is the view window, a rectangle in the view plane



View Volumes

- Edges of the view window are parallel to the $x_v y_v$ axes and window boundary positions are specified in the viewing coordinates.
- View volume can be set up using the window boundaries.
- Objects within the view volume will appear on an output device all others are clipped from the display.
- The size of the view volume depends on the size of the window while the shape depends on the type of projection to be used.
- For parallel projection, the view volume forms an infinite parallelepiped and for perspective view volume is a pyramid.

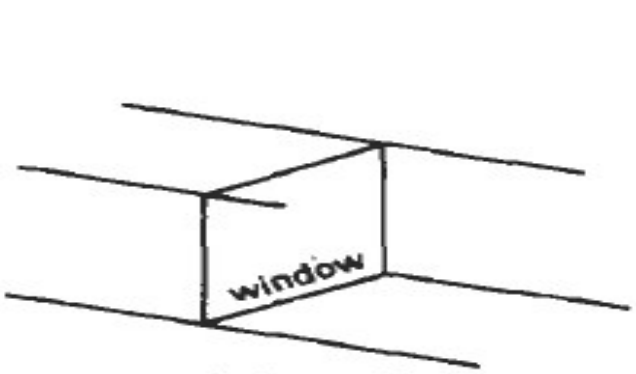
View Volume - Front and Back Planes

- We will also typically want to limit the view in the z_V direction
- We define two planes, each parallel to the view plane, to achieve this
  - front plane (or near plane)
  - back plane (or far plane)

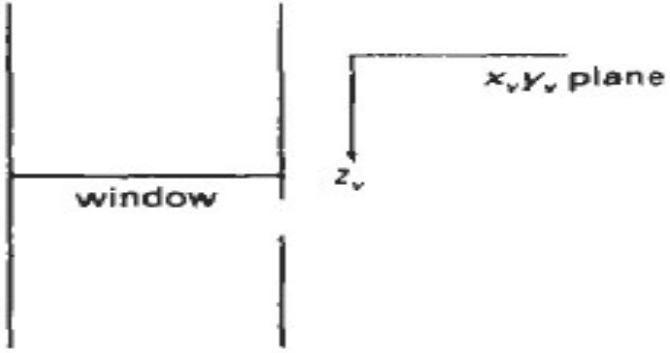
View Volume

- Front and back clipping planes allow us to eliminate parts of the scene from the viewing operations based on the depth.
- Both the planes must be on the same side of the projection reference point
- Back plane must be farther from the projection point than the front plane.
- Including the front and back planes a view volume is bounded by six planes.
- Orthographic parallel projection-->rectangular parallelepiped
- Oblique parallel projection--> oblique parallelepiped
- Perspective projection→ truncate the infinite pyramidal view volume to form a frustum

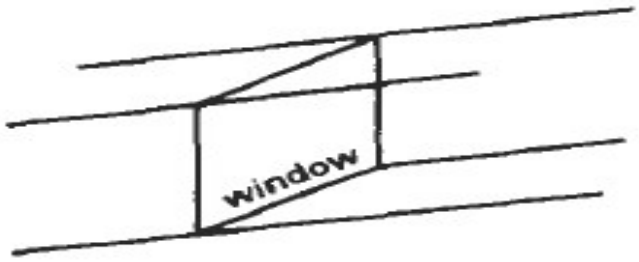
Parallel Projection View Volume



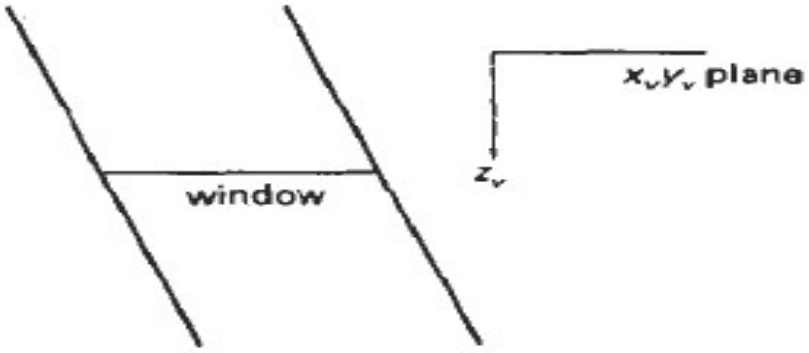
Orthographic-Projection View Volume (a)



Orthographic-Projection View Volume (b)



Oblique-Projection View Volume (c)



Oblique-Projection View Volume (d)

Figure 12-28
View volume for a parallel projection. In (a) and (b), the side and top views of the view volume for an orthographic projection are shown; and in (c) and (d), the side and top views of an oblique view volume are shown.

Perspective Projection View Volume

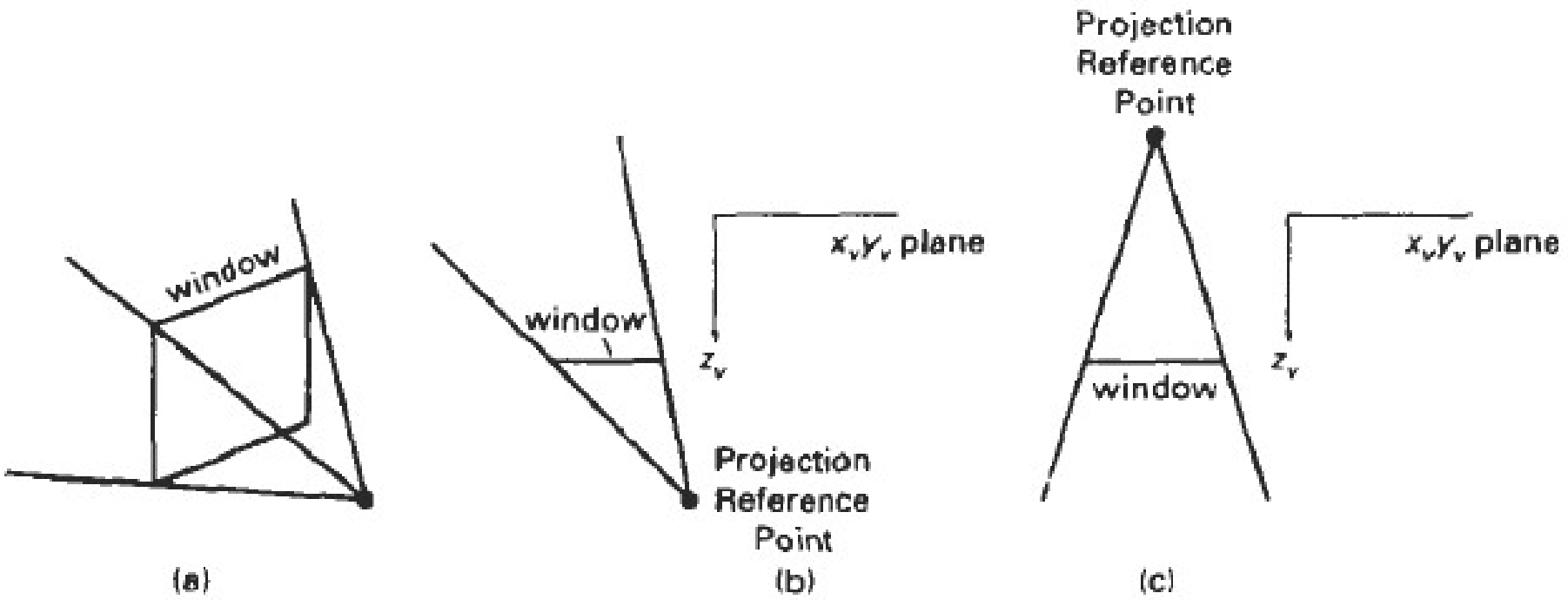
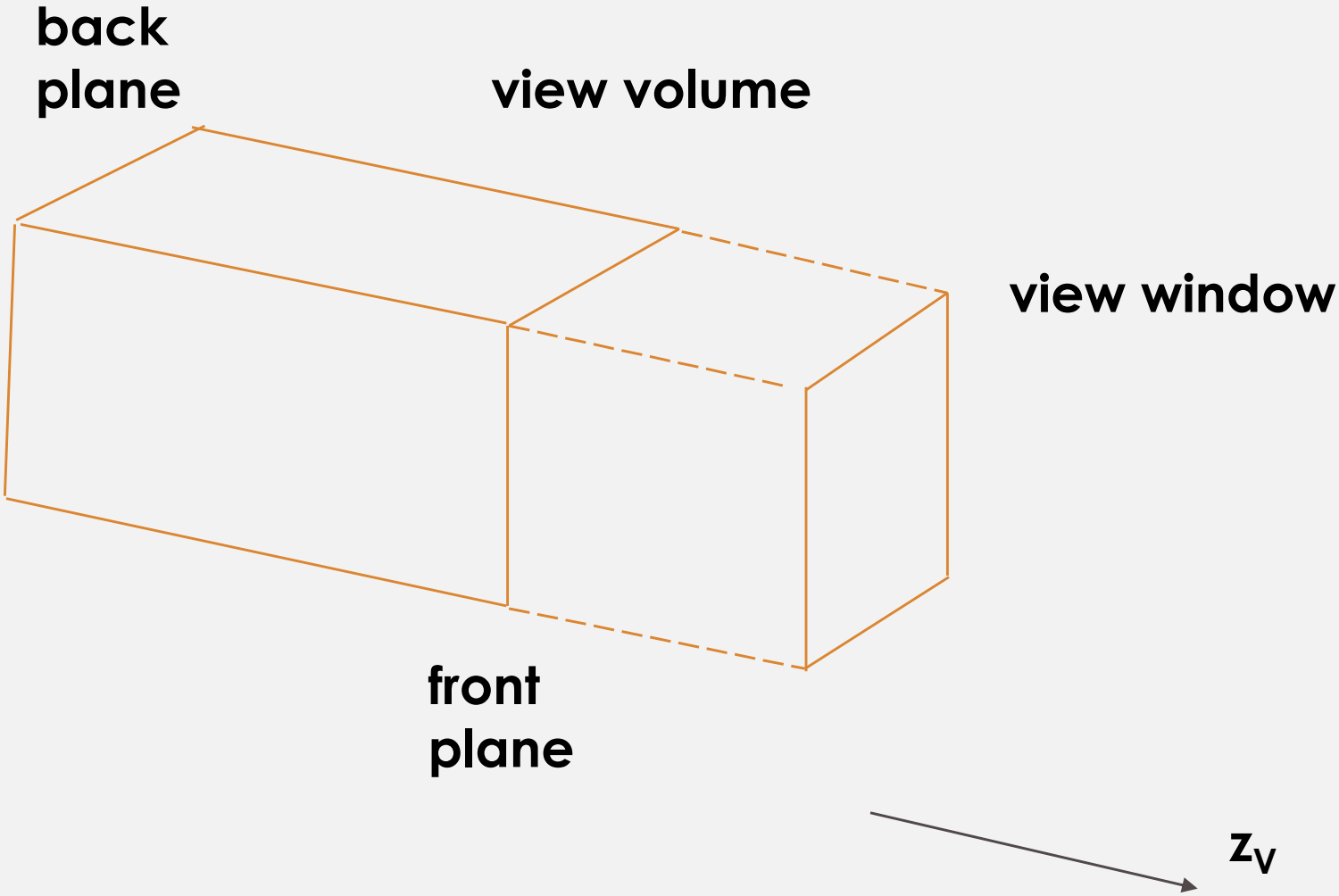


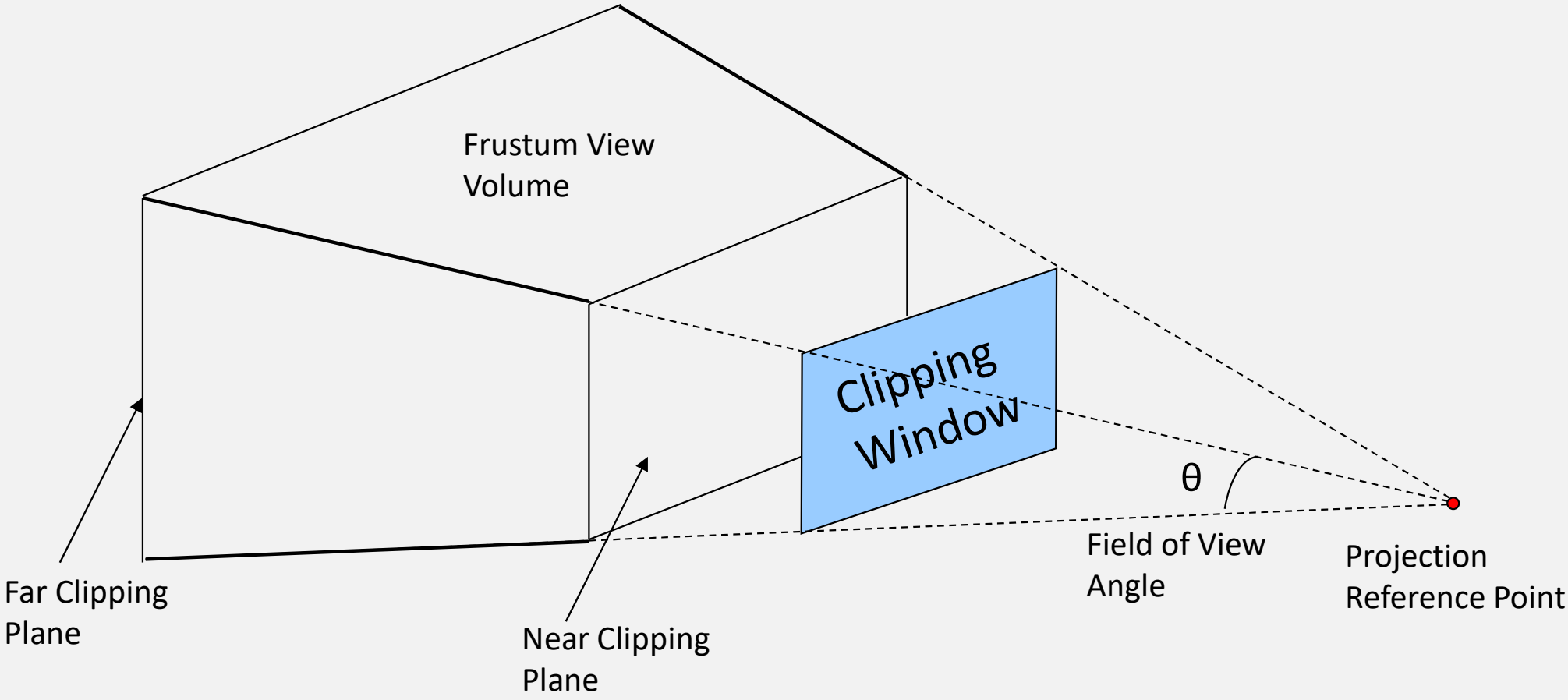
Figure 12-29
Examples of a perspective-projection view volume for various positions of the projection reference point.

View Volume - Parallel Projection

View Volume bounded by six planes



Perspective Projection View Volume



Effects of Moving the PRP w.r.t the view plane

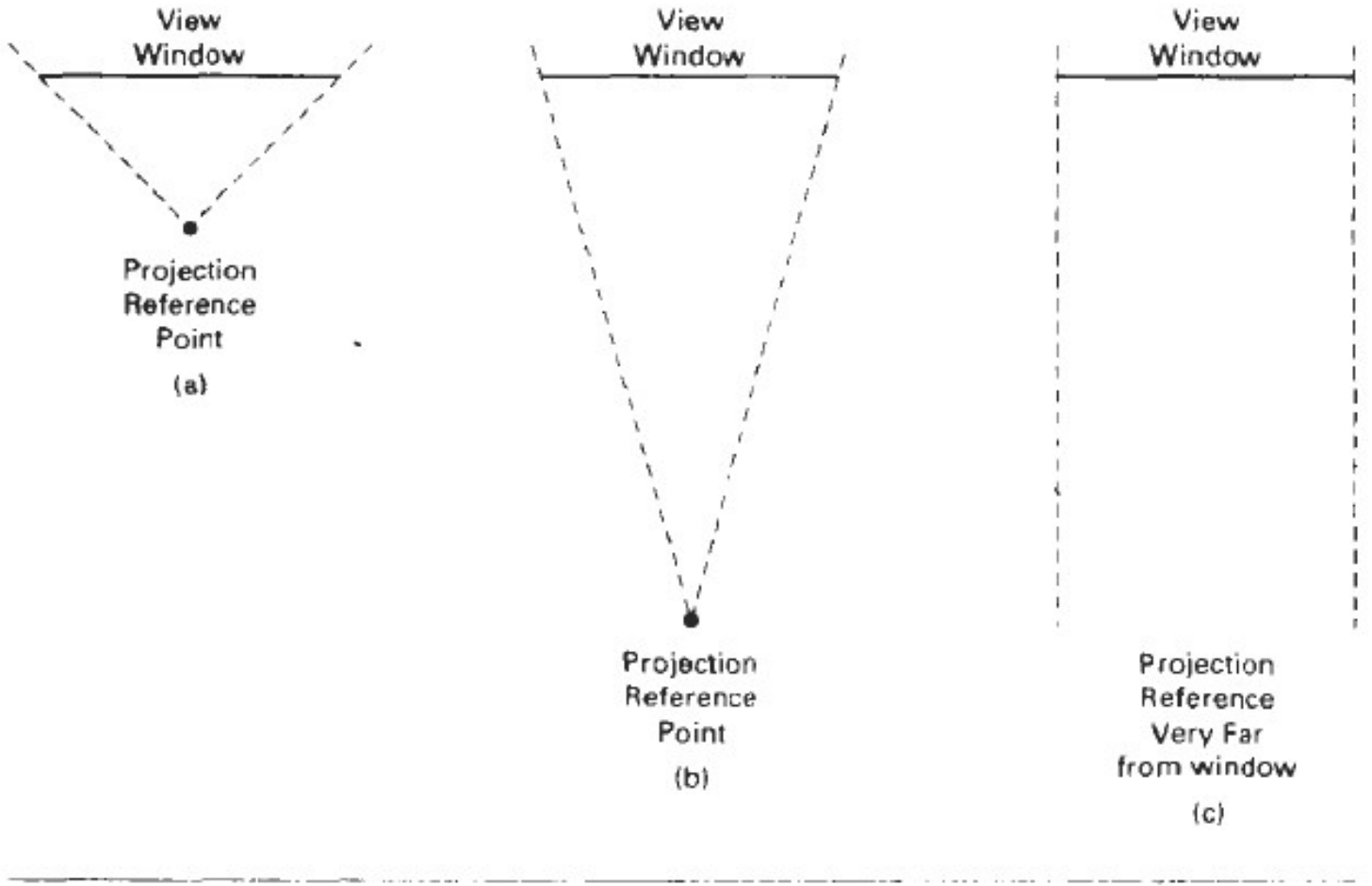


Figure 12-33
Changing perspective effects by moving the projection reference point away from the view plane

Projected Object size w.r.t the View Plane Position

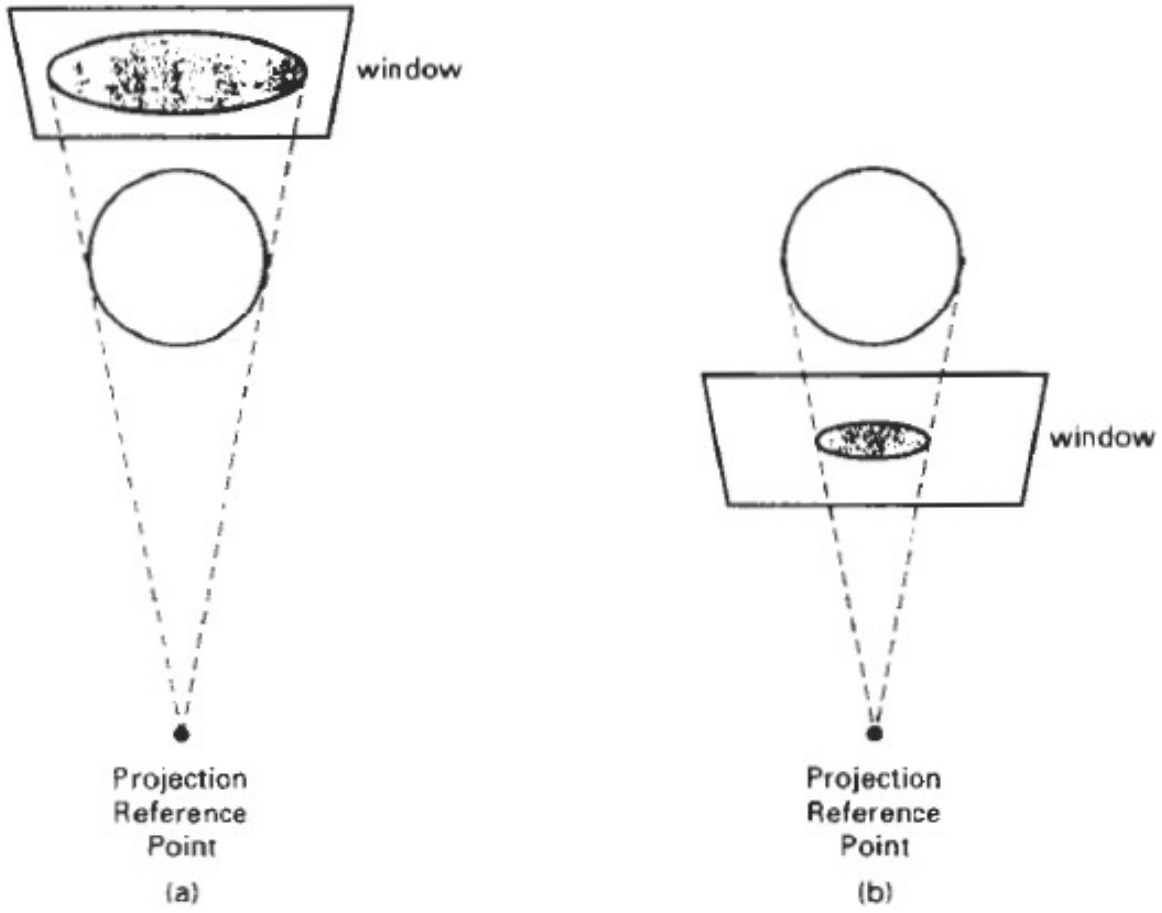


Figure 12-34
Projected object size depends on whether the view plane is positioned in front of the object or behind it, relative to the position of the projection reference point.

General Parallel Projection Transformation

General Parallel Projection Transformation

- The direction of the parallel projection is specified with a projection vector from the projection reference point to the center of the view window.
- Oblique parallel projection transformation is done by shearing to a regular parallelepiped.

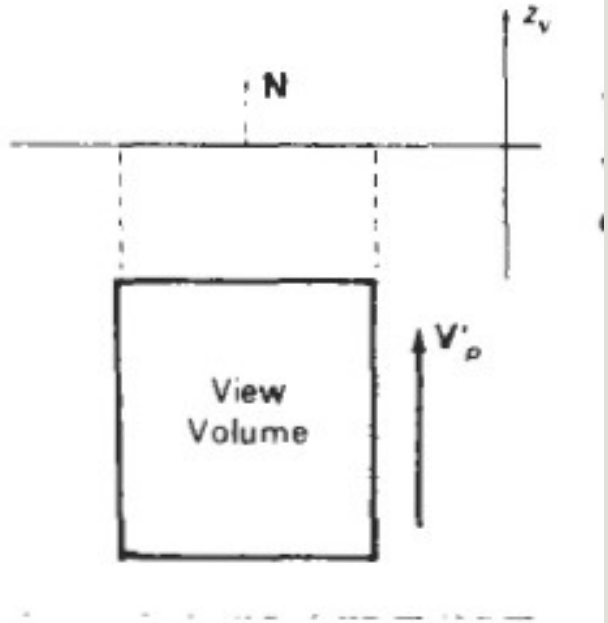


Figure 12-37
Regular parallelepiped view volume obtained by shearing the view volume in Fig. 12-36.

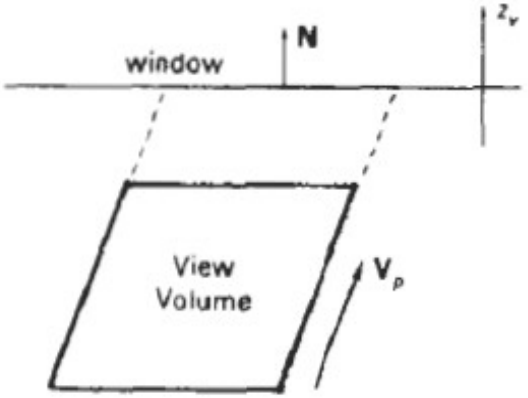


Figure 12-36
Oblique projection vector and associated view volume.

General Parallel Projection Transformation

The elements of the projection vector in viewing coordinates are

$$Vp = (p_x, p_y, p_z)$$

Shear the projection vector V_p with normal vector N

$$Vp' = M_{parallel} \cdot Vp$$

$$M_{parallel} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Parallel Projection Transformation

Explicit transformation equations in terms of shear parameters a and b are

$$0 = p_x + a p_z$$
$$0 = p_y + b p_z$$

$$a = -\frac{p_x}{p_z}, \qquad b = -\frac{p_y}{p_z}$$

General parallel projection matrix in terms of elements of projection vector is

$$\mathbf{M}_{\text{parallel}} = \begin{bmatrix} 1 & 0 & -p_x/p_z & 0 \\ 0 & 1 & -p_y/p_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Perspective Projection Transformation

General Perspective Projection Transformation

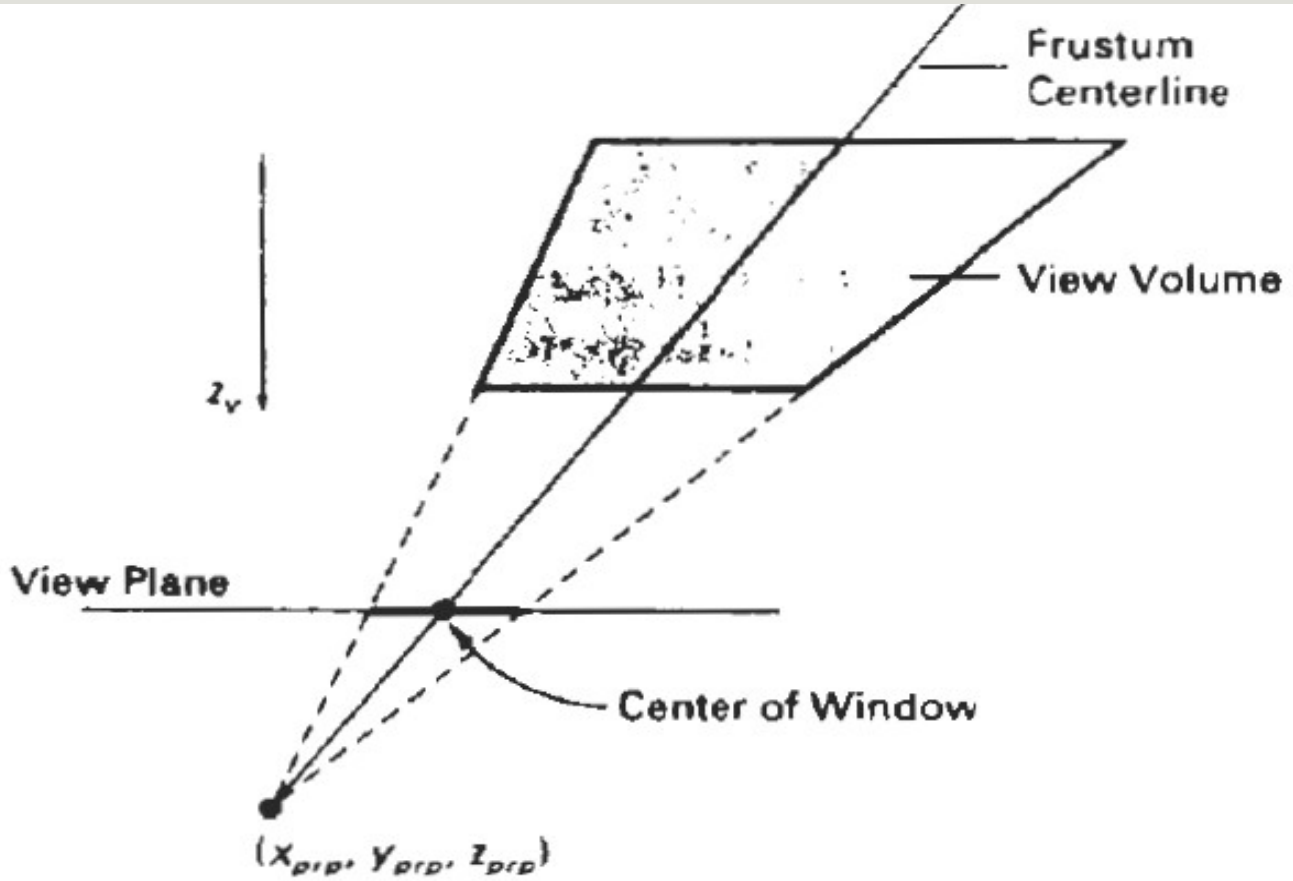


Figure 12-39
General shape for the perspective view volume with a projection reference point that is not on the z_v axis

General Perspective Projection Transformation

Done with the following 2 operations

1. Shear the view volume so that the centerline of the frustum is perpendicular to the view plane

2. Scale the view volume with a scaling factor that depends on 1/z

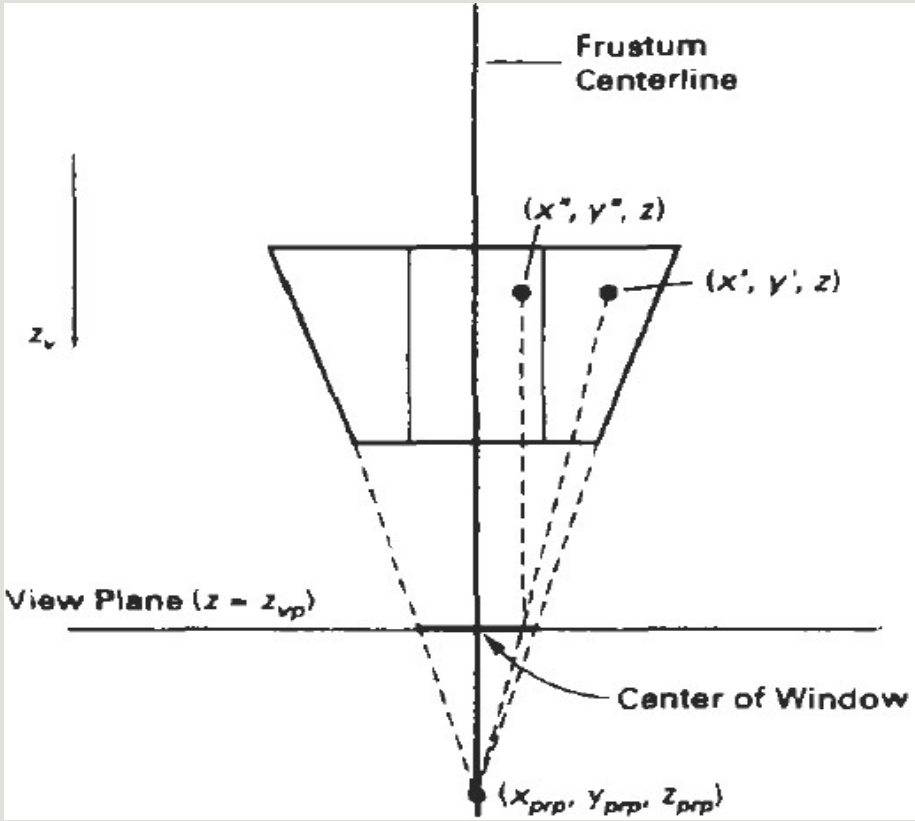


Figure 12-40
Shearing a general perspective view volume to center it on the projection window.

General Perspective Projection Transformation

The transformation matrix is

$$\mathbf{M}_{\text{shear}} = \begin{bmatrix} 1 & 0 & a & -a\,z_{prp} \\ 0 & 1 & b & -b\,z_{prp} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Points within the view volume are transformed by this operation as

$$x' = x + a(z - z_{prp})$$
$$y' = y + b(z - z_{prp})$$
$$z' = z$$

When projection reference point is on z axis

$$x_{prp} = y_{prp} = 0$$

General Perspective Projection Transformation

After shearing we apply scaling transformation to produce a regular parallelepiped.

$$x'' = x'\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + x_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

$$y'' = y'\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + y_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

General Perspective Projection Transformation

and the homogeneous matrix representation is

$$\mathbf{M}_{scale} = \begin{bmatrix} 1 & 0 & \frac{-x_{prp}}{z_{prp} - z_{vp}} & \frac{x_{prp} z_{vp}}{z_{prp} - z_{vp}} \\ 0 & 1 & \frac{-y_{prp}}{z_{prp} - z_{vp}} & \frac{y_{prp} z_{vp}}{z_{prp} - z_{vp}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{z_{prp} - z_{vp}} & \frac{z_{prp}}{z_{prp} - z_{vp}} \end{bmatrix}$$

The general perspective projection transformation can be expressed as

$$M_{perspective} = M_{scale} \cdot M_{shear}$$