Computer Graphics (L11)

EG678EX

3-D Viewing

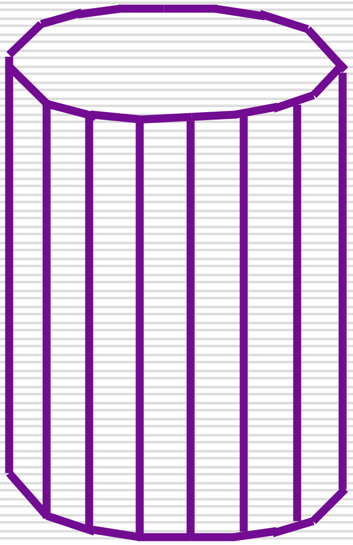
Some Background Concepts

3-D Object Representation

- Many kinds of objects in graphics scene. E.g. flowers, trees, clouds, rocks, bricks, marble, steel etc
- All characteristics can't be described with single way of representation of objects
- Two broad categories to represent **Euclidean** (geometric) solid objects:
  - Boundary Representation (B-reps)→ object is assumed as set of surfaces that separate object from surroundings. E.g. surface polygons
  - Space-Partitioning Representations→ represented by set of non overlapping contiguous solids. E.g. Octree representation (with cubes)
- **Natural objects** (Non-Euclidean i.e. non-geometrical shape) → Represented by **Fractal Geometry**.

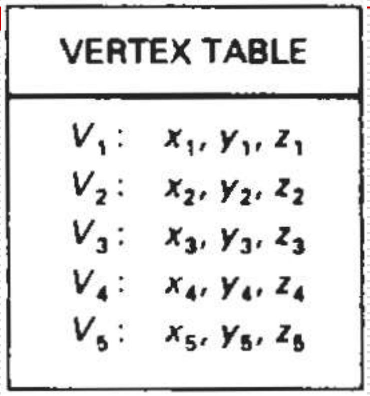
3-D Object Representation

- Polygon Surfaces
  - Approximate representation 3-D objects by set of surface polygons
    - Advantage:- speeds up rendering; since all surfaces are represented with linear equations
  - More accurate representation for polyhedron
  - Curved surface approximation can be improved with increase in number of polygons
- Polygon Tables:- To specify polygon surfaces
  - Geometric Tables
    - Consist of parameters that specify polygon vertices and spatial orientation of polygons
  - Attribute Tables
    - Consist of parameters that specify transparency, surface texture, color etc.

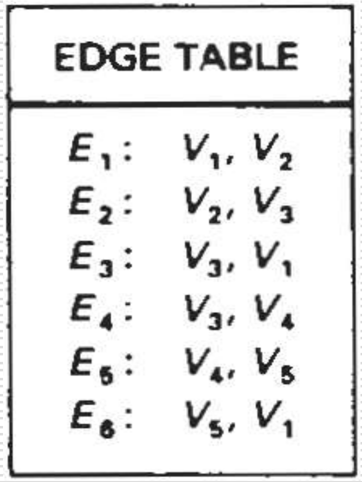

Polygon Surface Approximation of Cylinder

3-D Object Representation

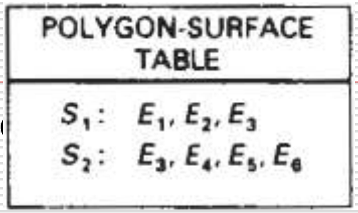
- Geometric Tables
  - Specified by three lists (each successive tables consists of pointers back to previous table
    - Vertex Table
    - Edge Table
    - Polygon Surface Table
  - Only Edge Tables or (Vertex table + Polygon Table) is sufficient ?
    - Yes but redundancy (repeated) in drawing
  - Alternate representation:- we add extra information for quick identification of redundancy. E.g. as in figure (Alternate Representation), pointers to polygon table are added to edge table so as to notice common edges quickly.



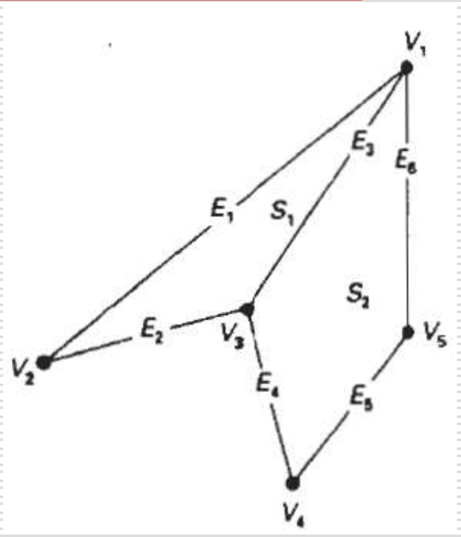
VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

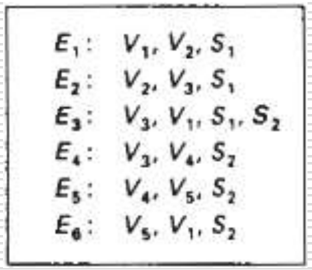


EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1



POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6





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

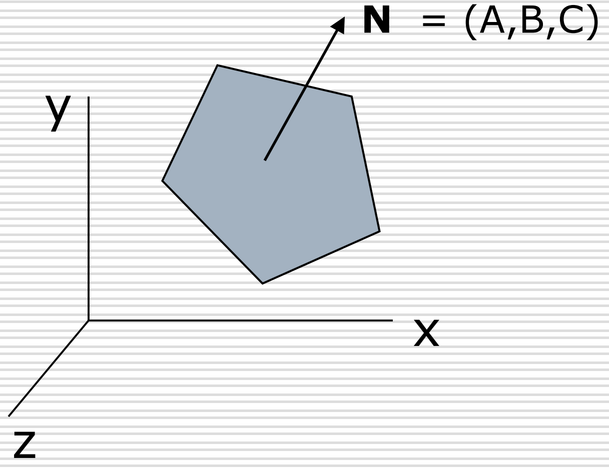
Alternate Representation

3-D Object Representation

- Spatial Orientation of Surface
  - Orientation of surface normal
  - How to find surface normal if surface vertices (x_1,y_1,z_1), (x_2,y_2,z_2) and (x_3,y_3,z_3) are given ?
    - Plane equation: $Ax + By + Cz + D = 0$
    - Put the vertex coordinates to get

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

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



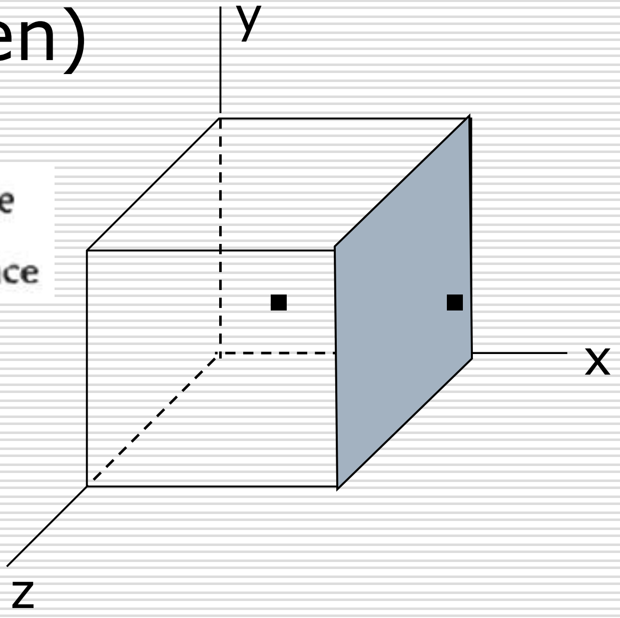
    - One of the surface normals is: **N** = (A,B,C)
      (Note: Remember plane equation of form $lx + my + nz = p$
      i.e. **N** = (l,m,n))

3-D Object Representation

- Two sides of a surface
  - For right handed cartesian system (i.e if vertices are taken in C-Clockwise order to evaluate A,B,C and D then)

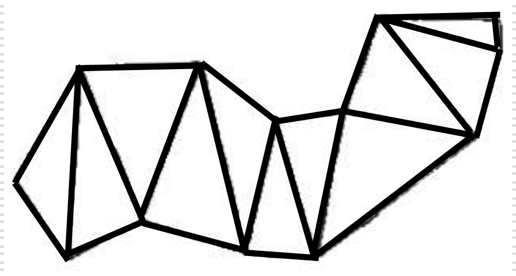
if $Ax + By + Cz + D < 0$, the point (x, y, z) is inside the surface

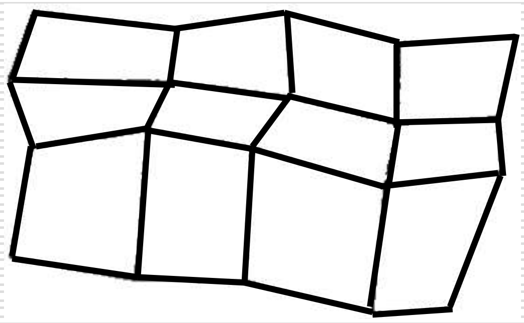
if $Ax + By + Cz + D > 0$, the point (x, y, z) is outside the surface



3-D Object Representation

- Polygon Meshes:- object represented as:
  - Triangle Strip
    - Given n vertices co-ordinates, n-2 trangles are tiled to produce strip
  - Quadrilateral Mesh
    - Given n by m array of vertices co-ordinates, (n-1) by (m-1) quadrilaterals produce mesh

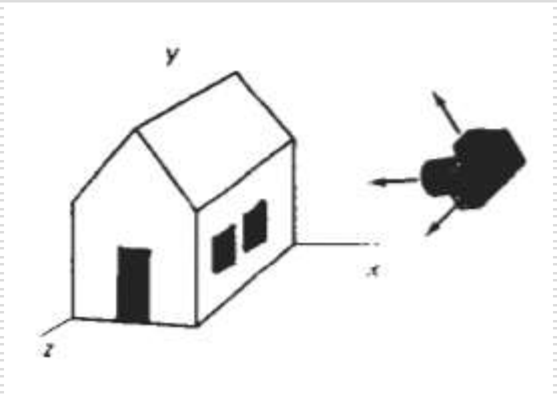


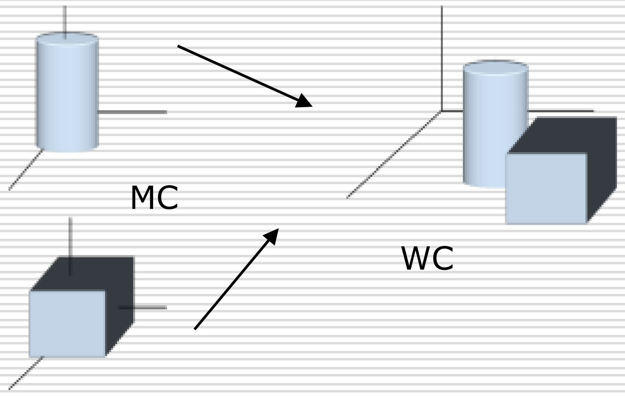


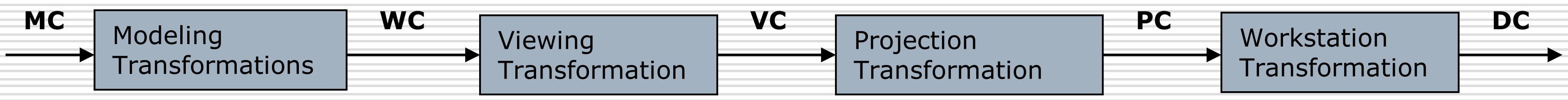
3-D Viewing

3-D Viewing Pipeline

- Analogous to 2-D scene generation
- 3-D scene generation process analogous to taking photographs with camera
  - More flexible method than camera analogy in graphics to generate 3-D scene



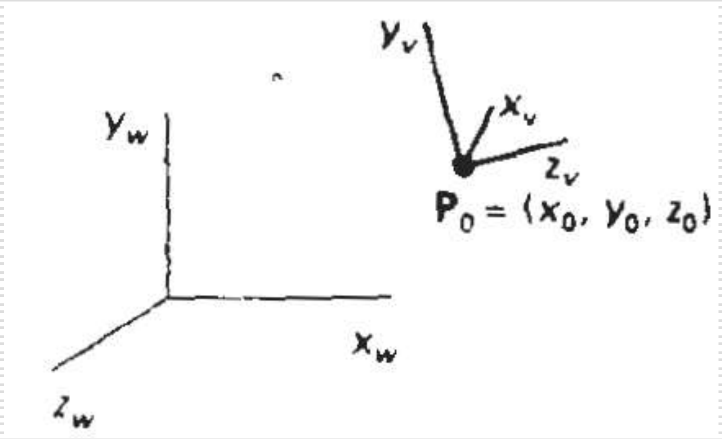




Viewing Coordinates

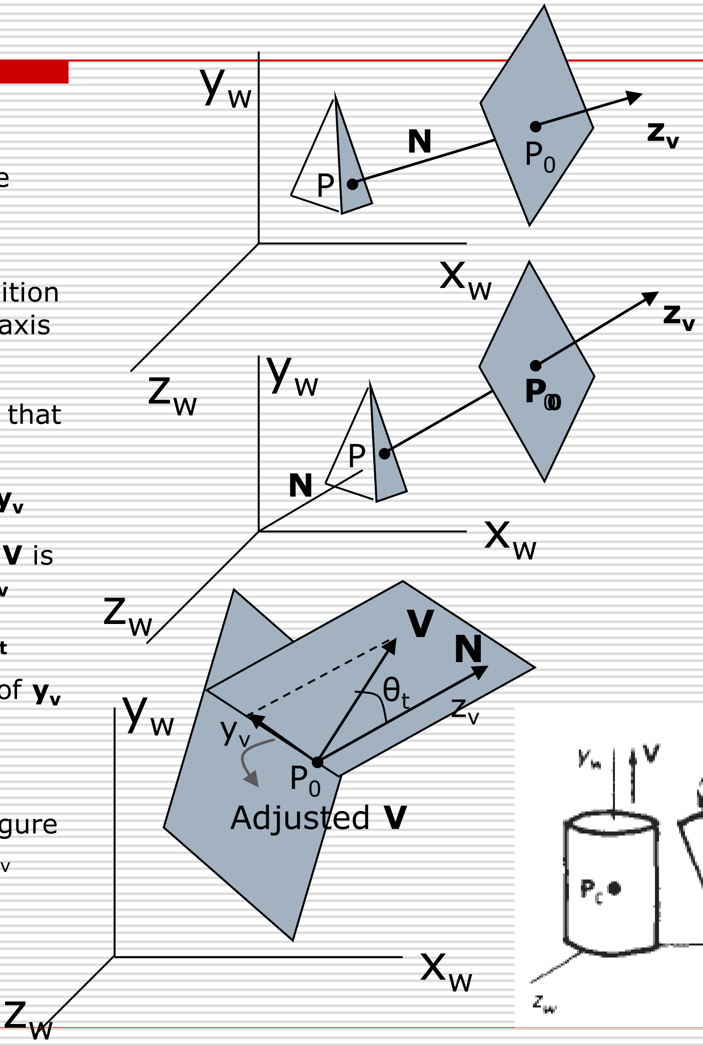
- Specifying Viewing Plane similar to camera orientation
- Steps:
  - Setup viewing Coordinate Reference → how?
  - Setup a view plane (projection plane) perpendicular to a viewing z_v axis
    - Note: plane could be assumed to be similar to camera film
  - Transform world co-ordinate scene to viewing coordinate scene

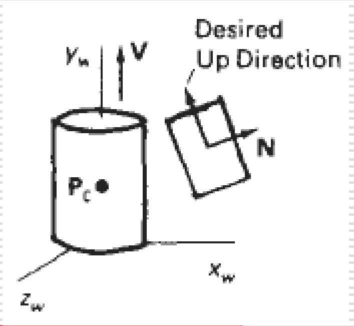
Viewing co-ordinates are projected unto the view plane



Setting Viewing Coordinates Reference

- Pick a point in world coordinate as **view reference point** (origin of viewing coordinate system)
  - view reference point often chosen closed to or on the surface of some object or at center of a group of objects.
  - If it is near to object→ we aim camera to that object
  - If it is some distance away from scene→ camera position
- Specify view plane orientation with Normal vector **N** i.e. z_v axis direction. **How?**
  - Chose a point in world coordinate
  - Vector from world origin to that point or vector from that point to view reference point is direction of Normal Vector **N**
- Specify *view-up vector* **V** to establish positive direction for $\mathbf{y_v}$ axis
  - Specifying **V** that is perpendicular to **N** is difficult so **V** is specified in any convenient direction and establish $\mathbf{y_v}$ direction by projecting **N** to View plane
  - In some packages **V** is specified as the twist angle $\boldsymbol{\theta_t}$ about the $\mathbf{z_v}$ axis and adjusted by projecting to view plane. The projection specifies the desired direction of $\mathbf{y_v}$ axis.
  - In some packages view reference point P_0 is at the center of the object, **V** is specified by world vector (0,1,0) and this vector is projected to the plane perpendicular to **N** to establish y_v axis as shown in figure
- Establish third vector **U** perpendicular to both the vectors y_v and z_v to specify vector x_v

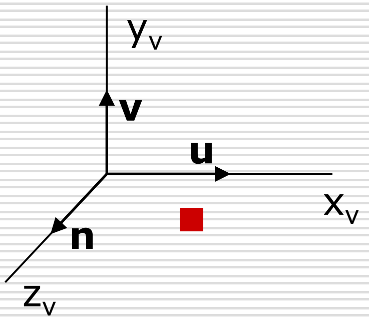


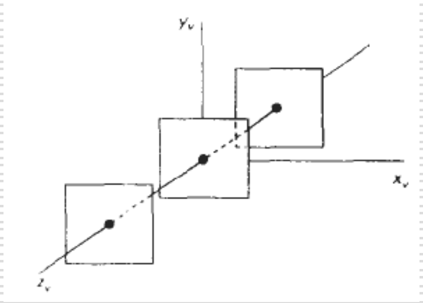


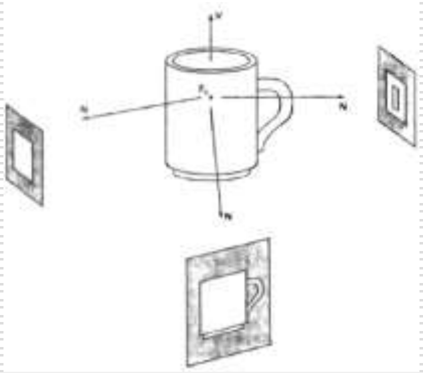
Setting Viewing Coordinates Reference (contd…)

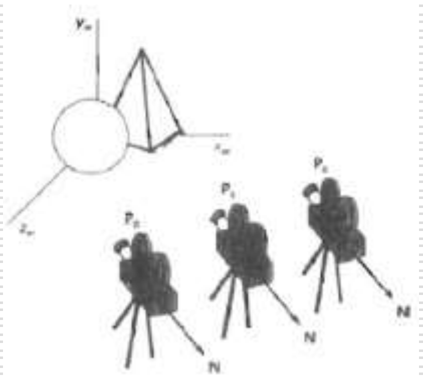
- Normalized viewing reference axis
  - Specified by unit vectors u, v and n (uvn system) along viewing axes
- View plane position
  - Specified by view-plane-distance from viewing origin
  - View plane is always parallel to $x_v y_v$ plane

  - Series of view is obtained by changing direction of **N** and keeping view reference point fixed. (How to obtain view along the line of **v**?)
  - To simulate camera motion through a scene keep the direction of N fixed and change the view reference point



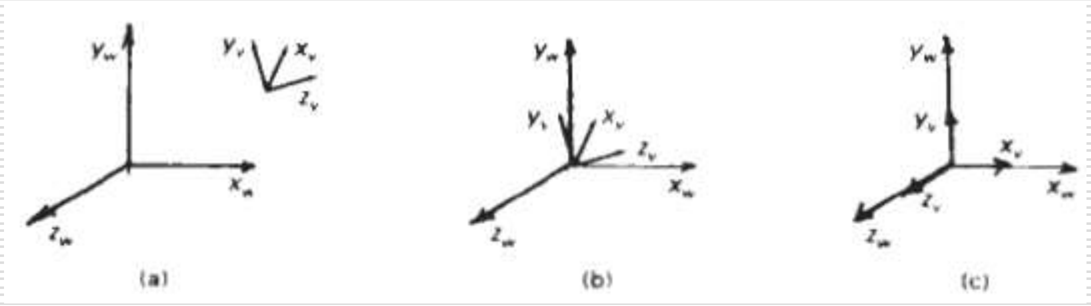






Transformation From World To Viewing Coordinates

1. Translate the view reference point to world origin
2. Rotate the view reference axis x_v, y_v, z_v to align with world axes x_w, y_w, z_w respectively



$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = (n_1, n_2, n_3)$$

$$\mathbf{u} = \frac{\mathbf{V} \times \mathbf{N}}{|\mathbf{V} \times \mathbf{N}|} = (u_1, u_2, u_3)$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_1, v_2, v_3)$$

$$\mathbf{R} = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T}$$

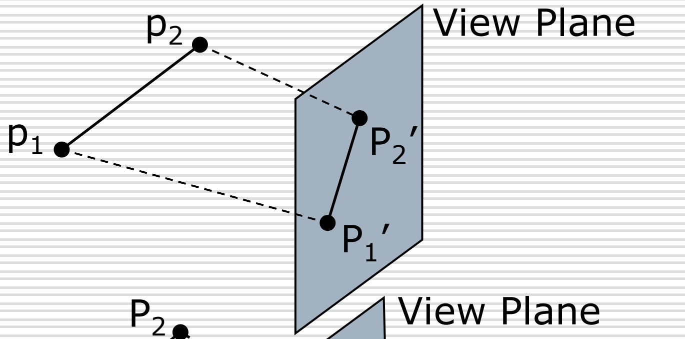
$$R. \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

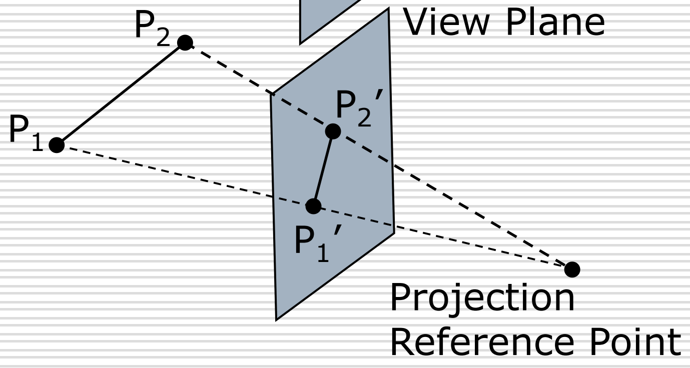
Thus the rotation matrix aligns vector **u** to x-axis
Similarly it allings vectors **v** and **n** to y and z axis of world

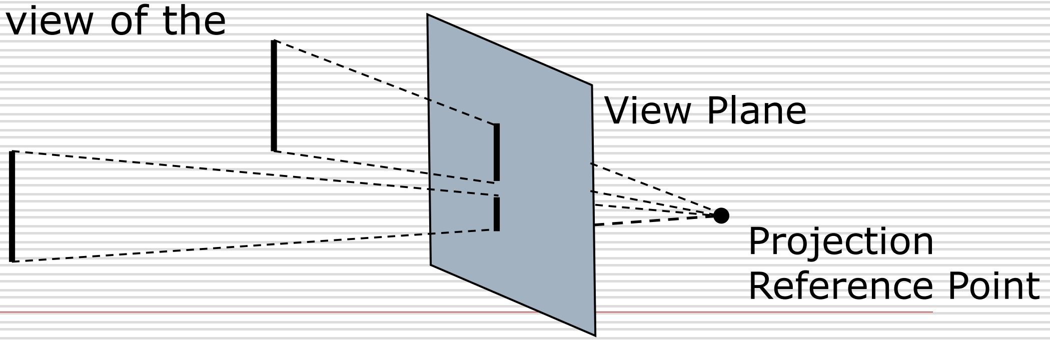
Thus the rotation matrix is the desired matrix to align view reference frame to world coordinate frame

Projections

- Converts 3-D viewing co-ordinates to 2-D projection co-ordinates
  - Two types of projection
    1. Parallel Projection
       - Coordinate positions are transformed to view plane along parallel lines (projection lines)
       - **Orthographic** and **Oblique** Projection
    2. Perspective Projection
       - Coordinate positions are transformed to view plane along lines (projection lines) that converges to a point called projection reference point (center of projection)
       - Equal sized object appears in different size according as distance from view plane
  - Intersection of projection lines and view plane gives the projected view of the object

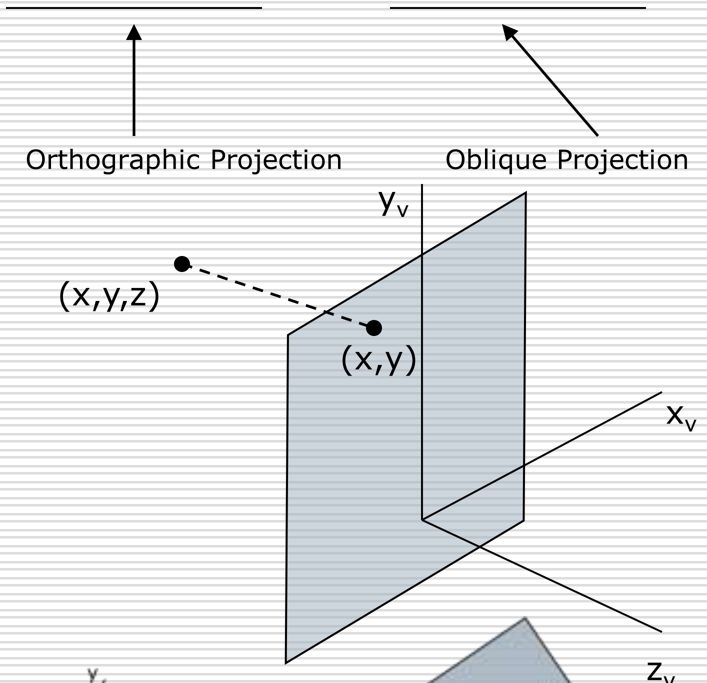


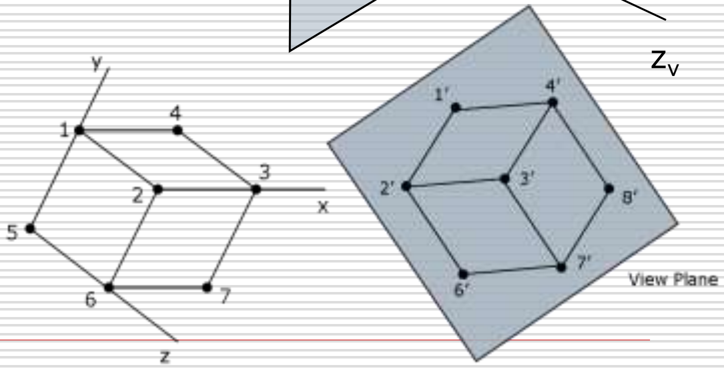




Parallel Projection

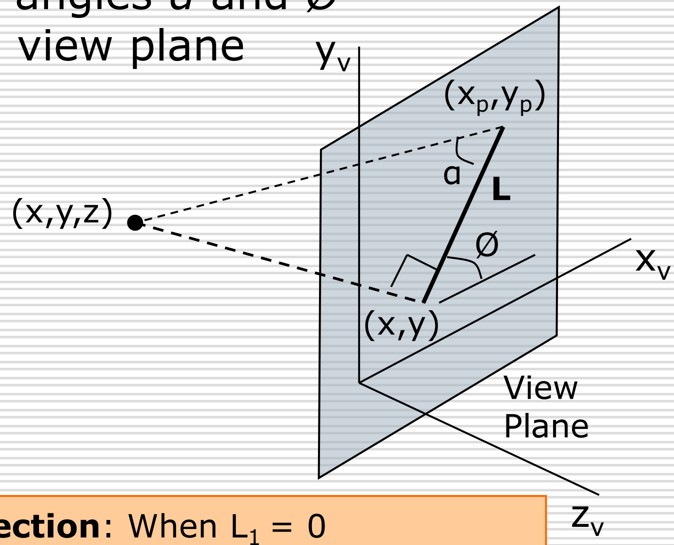
- **Orthographic projection**:
  - Projection lines are perpendicular to view plane
  - Used to produce Front, Side and Top view of an object
  - Axonometric Orthographic Projection:
    - To display more than one face
    - **Isometric Projection** is commonly used orthographic projection
      - Generated by aligning the projection plane so that it intersect each coordinate axis in which object is defined at same distance from origin
- **Oblique projection**: Projection lines are not perpendicular to view plane





Oblique Projection

- Oblique projection vectors are specified by two angles *α* and *Ø*
- (x,y) are also the orthographic co-ordinates on view plane



Projection Matrix Calculation

$x_p = x + L.\cos Ø$
$y_p = y + L.\sin Ø$

Coordinates (x,y,z) are viewing coordinates

$\tan\alpha = z/L$
$L = z/\tan\alpha = z.L_1$ where, $L_1 = 1/\tan\alpha$

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

- Final projection matrix will be

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic projection: When $L_1 = 0$
$x_p = x$
$y_p = y$

Cavalier projection: when $\tan\alpha = 1$ *(i.e $\alpha = 45^0$)*
$Ø = 30^0$ or 45^0
Cabinet Projection: when $\tan\alpha = 2$ *(i.e $\alpha = 63.4^0$)*
$Ø = 30^0$ or 45^0

Perspective Projection

- Projection vectors meet at projection reference point z_{prp} along the z_v axis

  $x' = x - xu$

  $y' = y - yu$

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

- On view plane

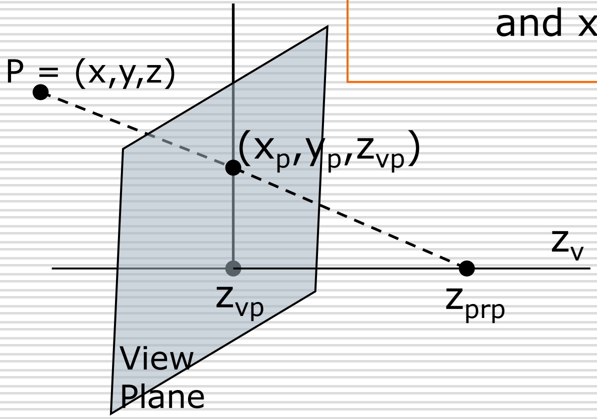
  $z' = z_{vp}$; therefore

  $u = (z_{vp} - z)/(z_{prp} - z)$

- thus projection transformation equations are

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

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



Projection Equation in homogeneous coordinates

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

Where, $h = (z_{prp} - z)/d_p$

and $x_p = x_h/h$, $y_p = y_h/h$

Special cases

uv plane as view plane (i.e $z_{vp} = 0$)

Viewing coordinate origin as projection reference point (i.e $z_{prp} = 0$)

Perspective projection

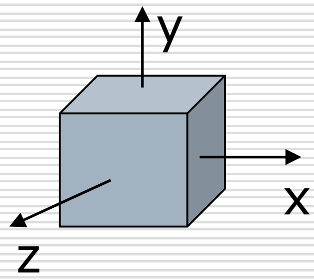
Vanishing Point: a set of parallel lines that are not parallel to view plane are projected as converging lines that appear to converge at a point called vanishing point

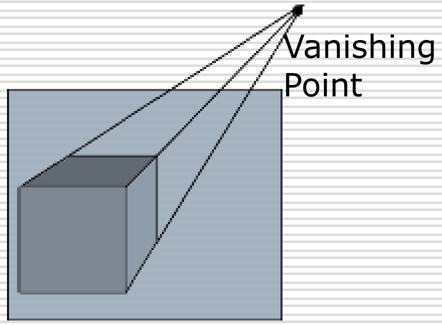
•a set of parallel lines that are parallel to view plane are projected as parallel lines

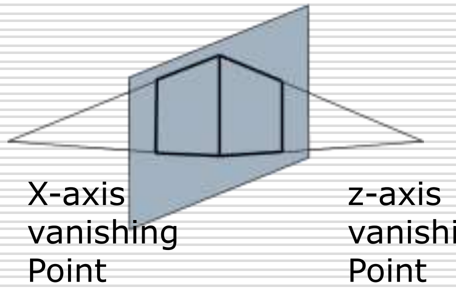
•More than one set of parallel lines form more than one vanishing points in the scene

Principal Vanishing point: Vanishing point for a set of parallel lines parallel to one of the principal axis of object

•We can control the number of principal vanishing point to one, two or three with the orientation of projection plane and classify as **one, two or three point perspective projection**




Vanishing
Point


X-axis
vanishing
Point
z-axis
vanishi
Point