Lecture 4-5
Transformations, Projections, and Viewing

Matrices for 3D Transformations

- 3D transformations → 4 by 4 matrices, using homogeneous coordinates
  - A point (x,y,z) → points (Wx, Wy, Wz, W), or (x/W, y/W, z/W, 1)
  - W=0 → corresponds to the point at infinity
- 3D Translation and Scaling are extended from 2D case straightforwardly.
- As for Rotation:

The positive direction of 3D Rotations in right-handed system:

positive rotations = when looking from a positive axis toward the origin, a 90 degree counterclockwise rotation will transform one positive axis into the other, according to the following table:

Rotation axis = x →	positive rotation is from y to z
axis = y →	from z to x
axis = z →	from x to y

Matrices for 3D Transformations

- 3D Rotation:
  - Previous 2D rotation $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ can be treated as a 3D rotation about z axis
  - i.e. $$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$
  - Rotation matrices about x and y axes: $$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1. All these basic transformation matrices have inverse, therefore any affine transformation composed by them does too.
2. Any number of rotation, scaling, and translation matrices can be multiplied together. The result always has the form: $$M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming lines and planes

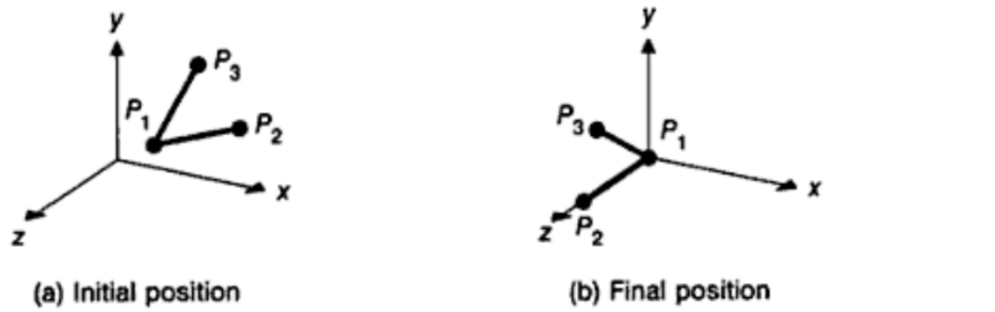
Each transformation matrix applies on vectors, or individual points.

- Transforming lines by transforming the endpoints
- Transforming triangles by transforming the three vertices.
- Transforming planes (represented using $N=[A\ B\ C\ D]^T$, and defined through $\{P|N \cdot P=0\}$) by transforming its normal.

Composition of 3D Transformations

Example: A common situation when we setup a local 3D coordinate system.

Given the directed line segments P_1P_2 and P_1P_3 in (a), find the transformation to transform it to their ending positions in (b): P_1 at the origin, P_1P_2 on positive z-axis, and P_1P_3 in the positive y axis half of the (y,z) plane



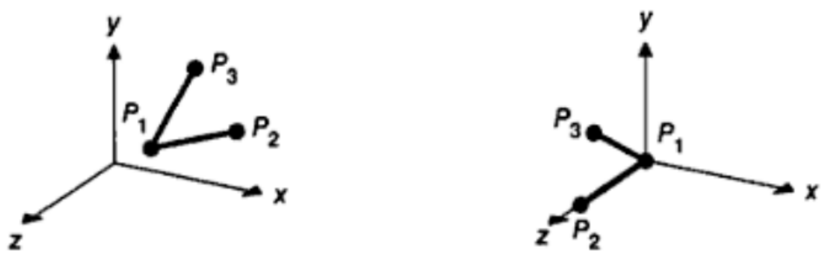
Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

Show two ways to solve it:

(1) Apply a sequential primitive transformations: translation + rotations…

(2) Using the properties of orthogonal matrices

Method 1



(a) Initial position (b) Final position

Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

Break it into simpler sub-problems, we can get it in four steps:

1. Translate P_1 to the origin
2. Rotate about the y-axis → P_1P_2 lies in the (y,z) plane
3. Rotate about the x-axis → P_1P_2 lies on the z-axis
4. Rotate about the z-axis → P_1P_3 lies in the (y,z) plane

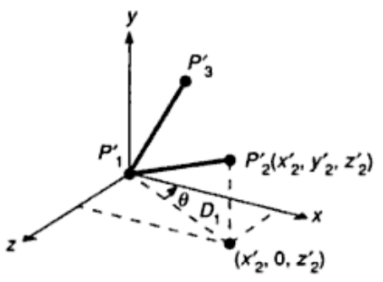
Step 1. Translate.

$$T_1(-x_1,-y_1,-z_1) = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

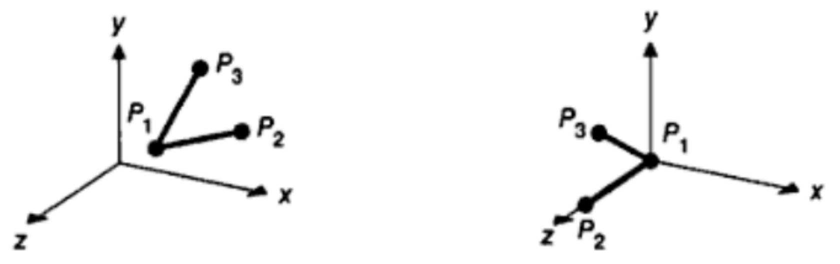
$$P'_1 = T(-x_1, -y_1, -z_1) \cdot P_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$P'_2 = T(-x_1, -y_1, -z_1) \cdot P_2 = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \\ 1 \end{bmatrix},$$

$$P'_3 = T(-x_1, -y_1, -z_1) \cdot P_3 = \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \\ 1 \end{bmatrix},$$



Method 1



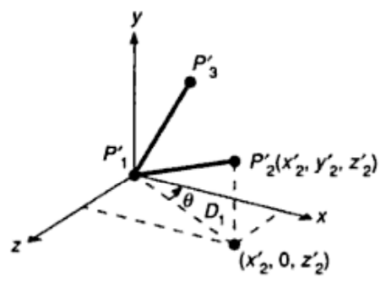
(a) Initial position (b) Final position

Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

Break it into simpler sub-problems, we can get it in four steps:

1. Translate P_1 to the origin
2. Rotate about the y-axis → P_1P_2 lies in the (y,z) plane
3. Rotate about the x-axis → P_1P_2 lies on the z-axis
4. Rotate about the z-axis → P_1P_3 lies in the (y,z) plane

Step 2. Need to rotate about y-axis by $-(90-\theta)=(\theta-90)$



$$R_2 = R_y(\theta - 90^\circ)$$

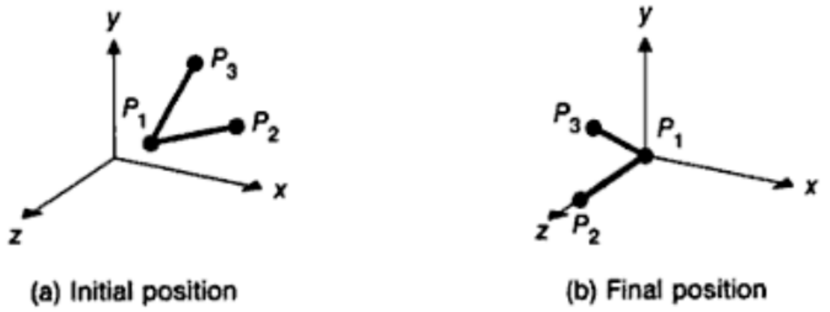
$$\cos(\theta - 90) = \sin\theta = \frac{z_2'}{D_1} = \frac{z_2 - z_1}{D_1},$$

$$\sin(\theta - 90) = -\cos\theta = -\frac{x_2'}{D_1} = -\frac{x_2 - x_1}{D_1},$$

where

$$D_1 = \sqrt{(z_2')^2 + (x_2')^2} = \sqrt{(z_2 - z_1)^2 + (x_2 - x_1)^2}.$$

Method 1

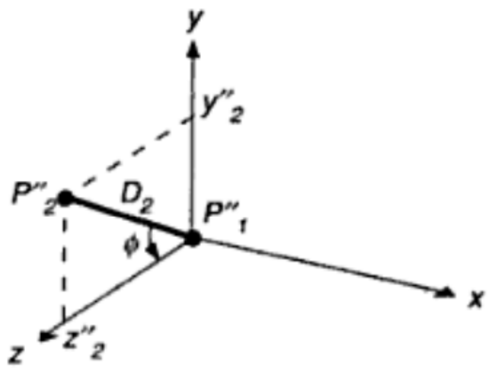


Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

Break it into simpler sub-problems, we can get it in four steps:

1. Translate P_1 to the origin
2. Rotate about the y-axis → P_1P_2 lies in the (y,z) plane
3. Rotate about the x-axis → P_1P_2 lies on the z-axis
4. Rotate about the z-axis → P_1P_3 lies in the (y,z) plane

Step 3. Need to rotate about x-axis by Φ



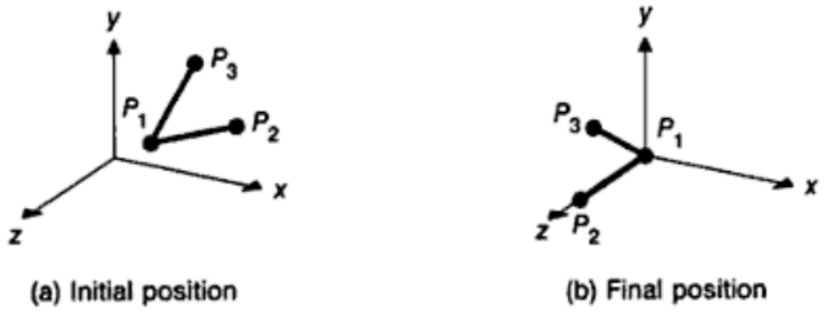
$$R_3 = R_x(\phi)$$

$$\cos\phi = \frac{z_2''}{D_2}, \ \sin\phi = \frac{y_2''}{D_2},$$

where

$$D_2 = |P_1''P_2''| = |P_1P_2| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Method 1

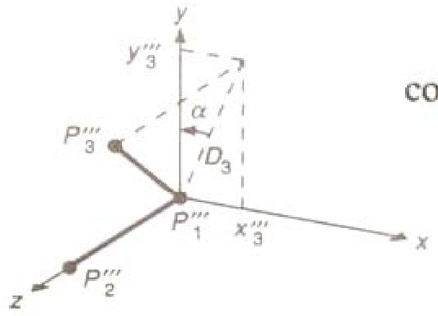


Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

Break it into simpler sub-problems, we can get it in four steps:

1. Translate P_1 to the origin
2. Rotate about the y-axis → P_1P_2 lies in the (y,z) plane
3. Rotate about the x-axis → P_1P_2 lies on the z-axis
4. Rotate about the z-axis → P_1P_3 lies in the (y,z) plane

Step 4. Need to rotate about z-axis by α



$$R_4 = R_z(\alpha)$$

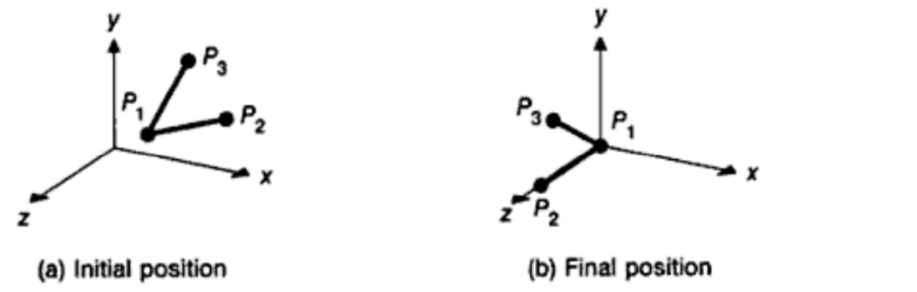
$$\cos\alpha = y_3'''/D_3, \quad \sin\alpha = x_3'''/D_3, \quad D_3 = \sqrt{x_3'''^2 + y_3'''^2}.$$

So the final composited transformation is:

$$R_z(\alpha) \cdot R_x(\phi) \cdot R_y(\theta - 90) \cdot T(-x_1, -y_1, -z_1)$$

Method 2

A generally useful quicker way to get the rotation matrix.



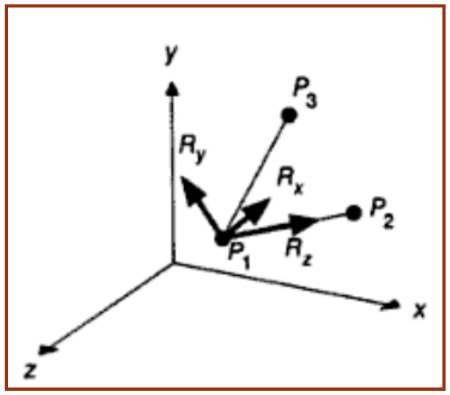
Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

To obtain the rotation matrix R using the properties of orthogonal matrices

1) Each row is a unit vector
2) Each row is perpendicular to the other
3) R's transpose is its inverse matrix
4) The ith row (except the last row) as a vector, will be rotated by $R(\theta)$ to lie on the positive axis $\mathbf{e}_i$

$$\begin{bmatrix} r_{1_x} & r_{2_x} & r_{3_x} & 0 \\ r_{1_y} & r_{2_y} & r_{3_y} & 0 \\ r_{1_z} & r_{2_z} & r_{3_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If R rotates a normalized vector $\mathbf{v}$ to $\mathbf{e}_i$, then its ith row is $\mathbf{v}^T$



1. we want to rotate P_1P_2 to "+z"-axis, so the 3rd row should be the normalized P_1P_2:

$$R_z = [r_{1_z} \quad r_{2_z} \quad r_{3_z}]^T = \frac{P_1P_2}{|P_1P_2|}.$$

2. R_x unit vector is perpendicular to the plane of P_1, P_2 and P_3, and will rotate into the "+x"-axis, so the 1st row:

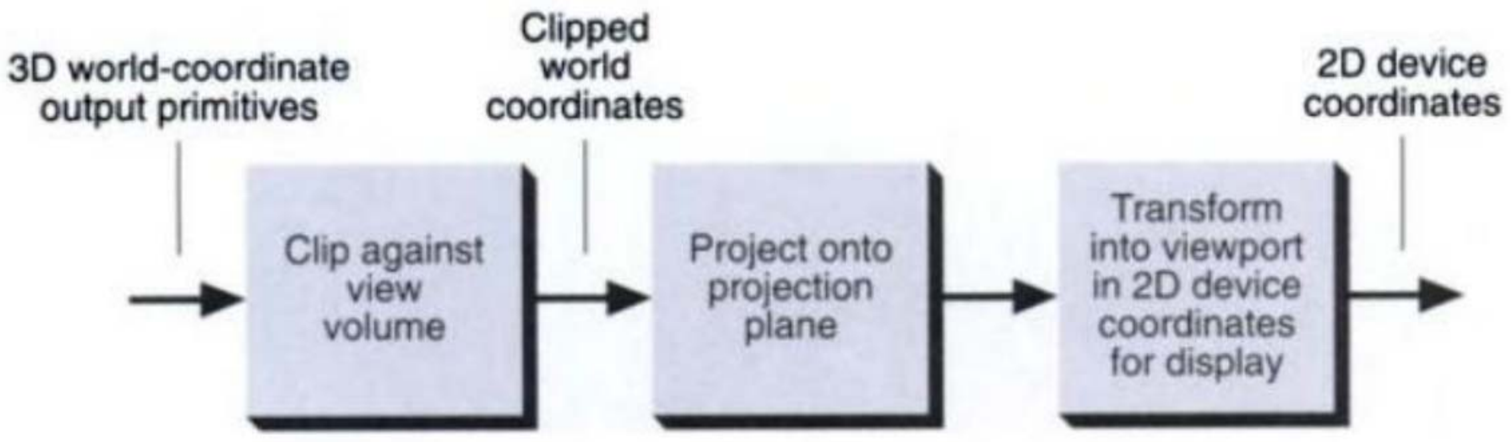
$$R_x = [r_{1_x} \quad r_{2_x} \quad r_{3_x}]^T = \frac{P_1P_3 \times P_1P_2}{|P_1P_3 \times P_1P_2|}$$

3. Finally:

$$R_y = [r_{1_y} \quad r_{2_y} \quad r_{3_y}]^T = R_z \times R_x$$

Viewing in 3D

2D shapes → window clip, translate, scale, translate → 2D viewport
3D shapes → projection → 2D viewport
(view volume clip → projection → 2D transform)


3D world-coordinate output primitives
Clipped world coordinates
2D device coordinates
Clip against view volume
Project onto projection plane
Transform into viewport in 2D device coordinates for display

Conceptual model of the 3D viewing process.

Projections

Generally:

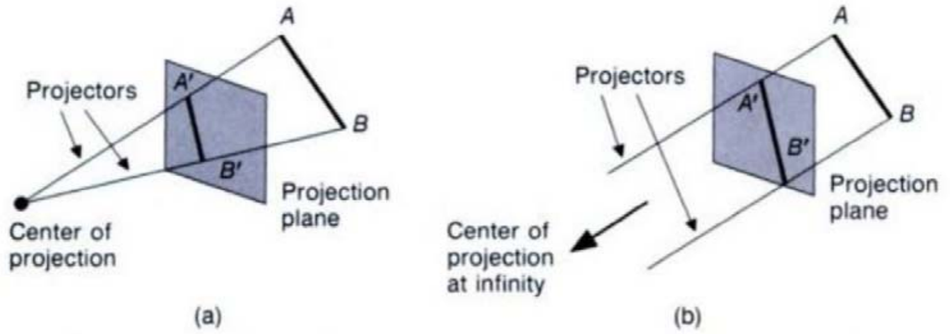
- ❑Projections transform points in a n-D coordinate system into points in a m-D coordinate system (m<n)

- ❑Computer Graphics has long been used for studying n-D objects by projecting them into lower dimensional (especially, 2D) space.

  Noll, M. , "A Computer Technique for Displaying N-dimensional hyperobjects", CACM, 10(8), Aug. 1967, 469-473.

Here:

- ❑We focus on projections from 3D to 2D.



(a) Line *AB* and its perspective projection *A'B'*. (b) Line *AB* and its parallel projection *A'B'*. Projectors *AA'* and *BB'* are parallel.

- ❑Projection :
  - ❑straight projection rays (projectors) emanating from a center of projection
  - ❑Passing through each point of the object
  - ❑Intersecting a projection plane to form the projection image

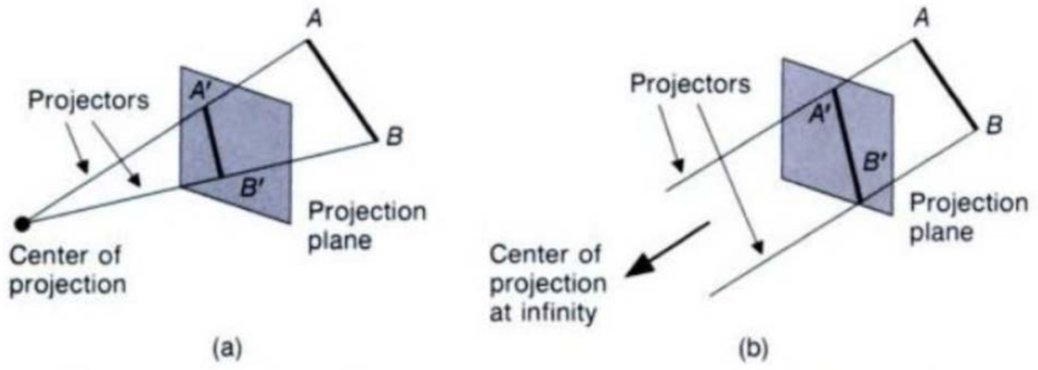
Classification of Projections

General Classification:

- We deal with planar geometric projections
- Non-planar projection: the projection plane is a curved surface (e.g. many cartographic projections)
- Non-geometric projection: the projection rays are curved (e.g. the Omnimax film)

Planar Geometric Projections:

- Parallel projection: projection center is infinitely far away
    - so that all projectors are parallel
    - We only need to specify direction of projection
- Perspective projection: projection center is finite distance away
    - Need to specify projection center



(a) Line *AB* and its perspective projection *A′B′*. (b) Line *AB* and its parallel projection *A′B′*. Projectors *AA′* and *BB′* are parallel.

Classification of Projections

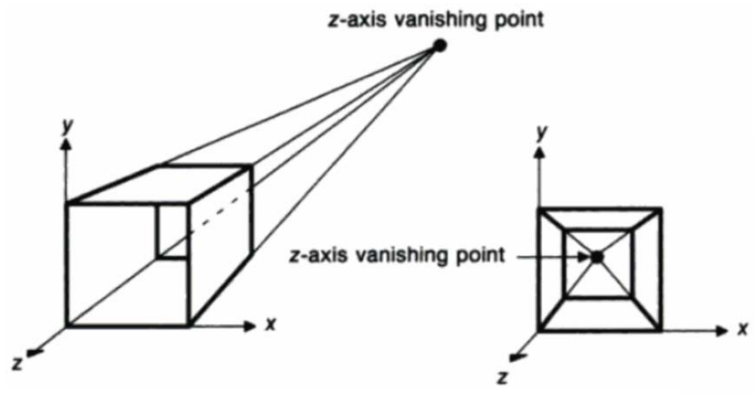
<u>General Classification:</u>
<u>Planar Geometric Projections:</u>

- ❑Perspective projection:
    - ❑Visually : perspective foreshortening (the object size varies inversely with the distance from the projection center), similar to human visual system
    - ❑Measurement:
        - ❑not good for recording exact shape,
        - ❑angles are preserved only on those faces of the object parallel to the projection plane,
        - ❑parallel lines generally are not projected to be parallel
- ❑Parallel projection:
    - ❑Visually : less realistic
    - ❑Measurement:
        - ❑good for exact measurement,
        - ❑parallel lines remain parallel,
        - ❑angles only preserved on faces that are parallel to the projection plane

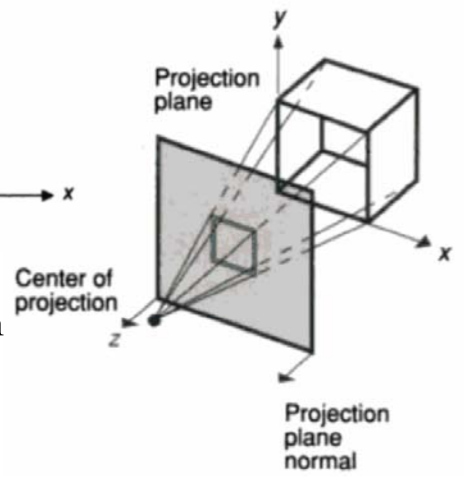
For more detailed discussions, check: Carlbom, I. and J. Paciorek, “Planar Geometric Projections and Viewing Transformations”, Computing Surveys, 10(4), Dec. 1978, pp. 465-502.

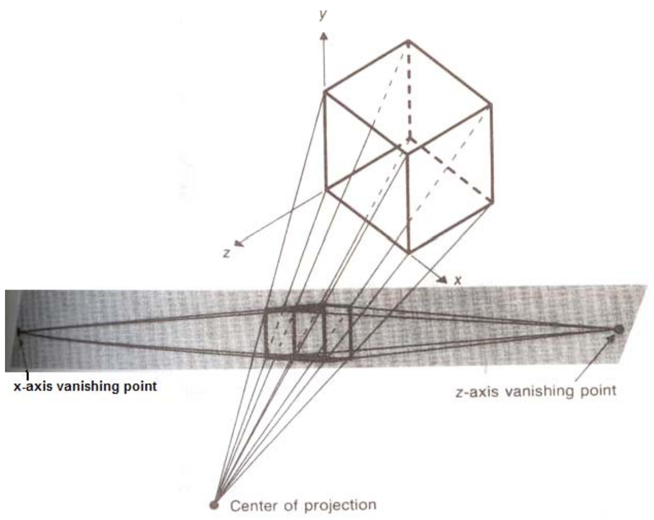
Perspective Projections

- Any set of parallel lines, that are not parallel to the projection plane, converge to a vanishing point.
- Axis vanishing point: if the set of lines that converges is parallel to one of the three principal axes
- There are at most three such points
  - e.g. if the projection plane cuts only the z-axis, then only the z axis has a principal vanishing point (lines parallel to either x or y axes have no vanishing point)
- Perspective projections are categorized by # of principal vanishing pts (i.e. by the # of axes the projection plane cuts)



One point perspective projection





Two point perspective projection

Parallel Projections

❑Categorized into two types, depending on the relation between (1) the direction of projection and (2) the normal to the projection plane:

❑**Orthographic parallel projections:** (1) and (2) have the same direction

❑Oblique parallel projections: (1) and (2) have different directions

Orthographic Parallel projections:

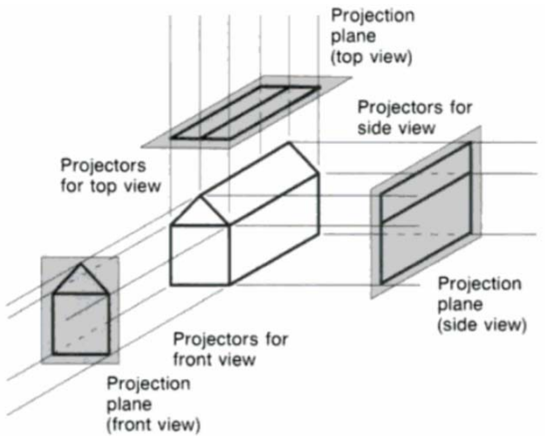
❑Front,top,side-elevation projections:

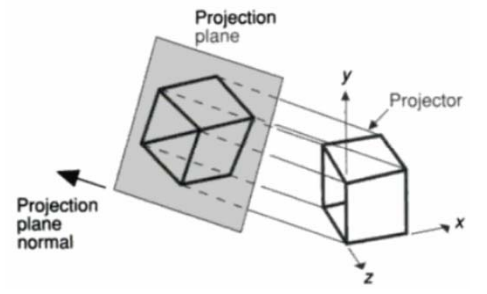
- oOften used in engineering drawings
- oDistances and angles can be measured
- oBut…

❑Axonometric orthographic projections:

- oProjections not normal to a principal axis
- oParallelism of lines preserved, angles not preserved
- oDistances can be measured along each principal axis (with different scales)
- oA commonly used type: isometric projection

→ projction direction makes equal angles with each axis (e.g. direction (1,1,1), every axis pair looks like 120 degree)


Projection plane (top view)
Projectors for side view
Projectors for top view
Projection plane (side view)
Projectors for front view
Projection plane (front view)

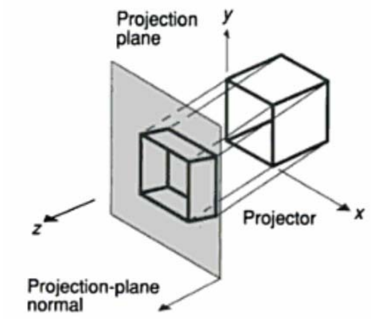


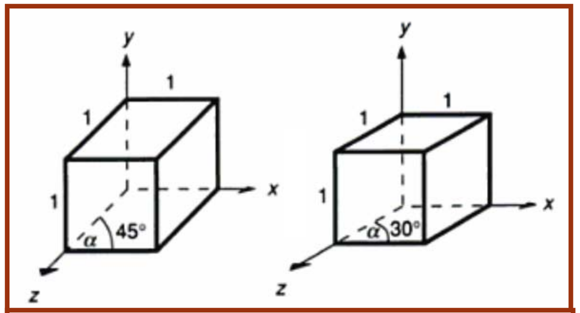
Parallel Projections (cont.)

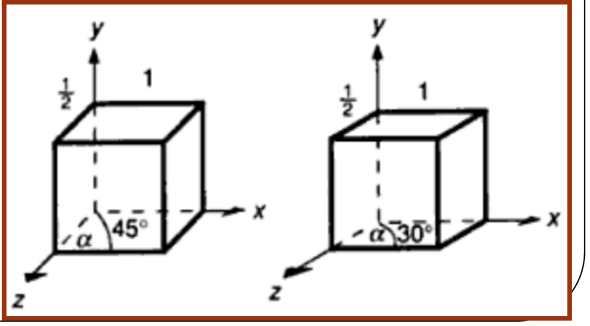
- Categorized into two types, depending on the relation between (1) the direction of projection and (2) the normal to the projection plane:
  - <u>Orthographic parallel projections:</u> (1) and (2) have the same direction
  - <u>**Oblique parallel projections:**</u> (1) and (2) have different directions

<u>**Oblique Parallel projections:**</u>

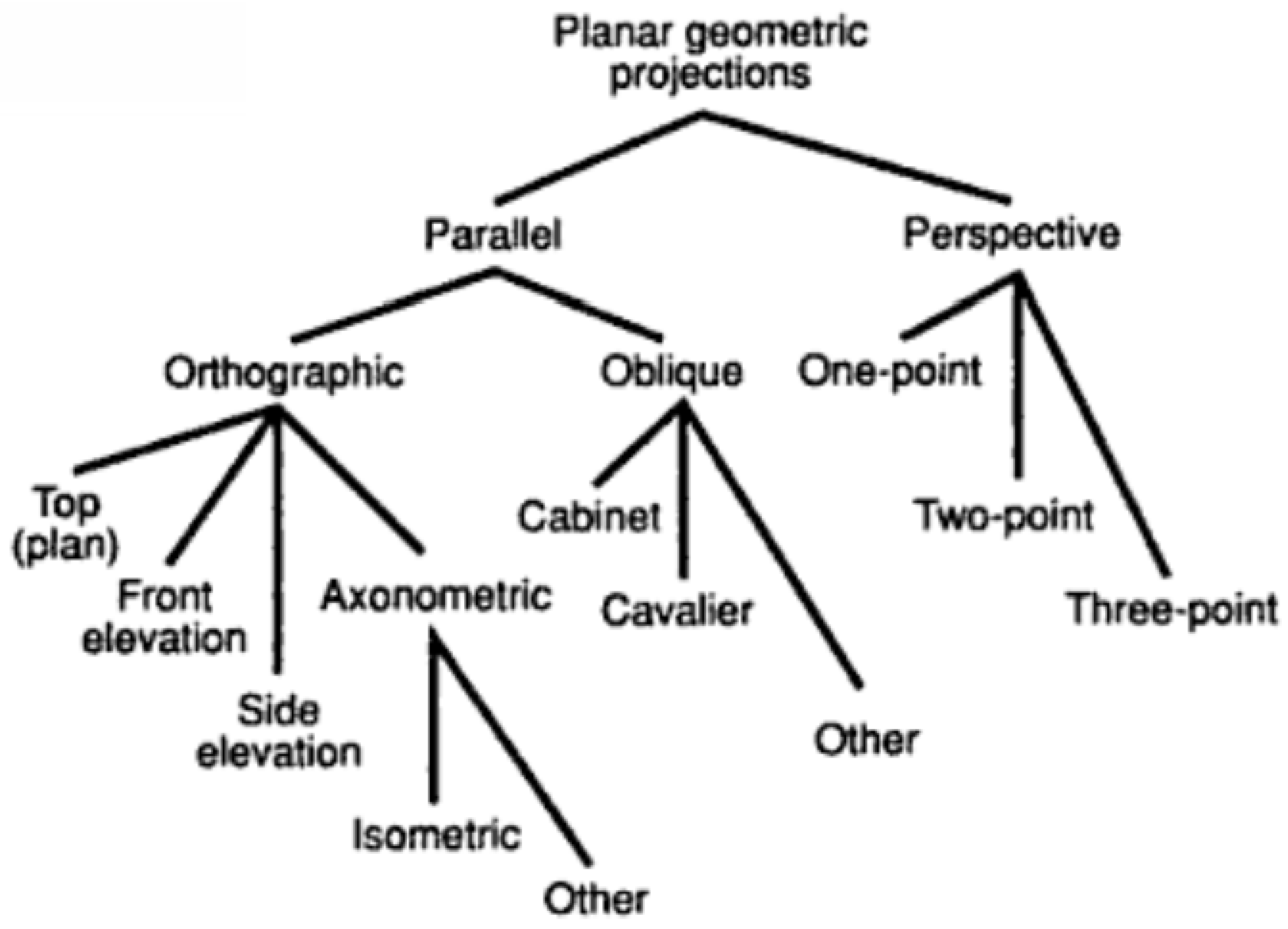
- Cavalier projections:
  - Projection direction makes a 45° angle with the projection plane
  - Result: line perpendicular to the projection plane preserves length, no foreshortening
  - 2 examples, right middle 2 figs.
    direction: $(\sqrt{2}/2, \sqrt{2}/2, -1)$ and $(\sqrt{3}/2, 1/2, -1)$
    projection plane: z=0 plane
- Cabinet projections:
  - Projection direction makes a 63.4° angle with the projection plane
  - Result: line perpendicular to the projection plane has half length
  - Visually more realistic, right bottom 2 figs
    direction: $(\sqrt{2}/4, \sqrt{2}/4, -1)$ and $(\sqrt{3}/4, 1/4, -1)$
    projection plane: the z=0 plane







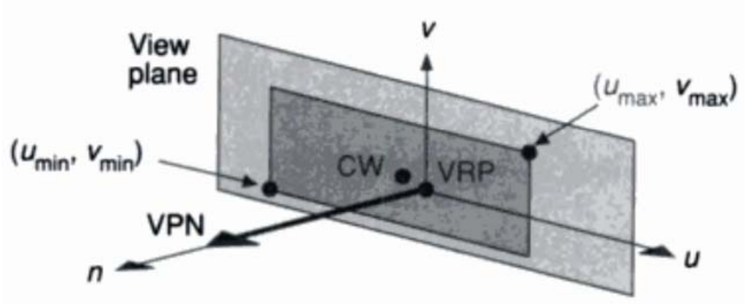
Planar Geometric Projections



Specifying an 3D View

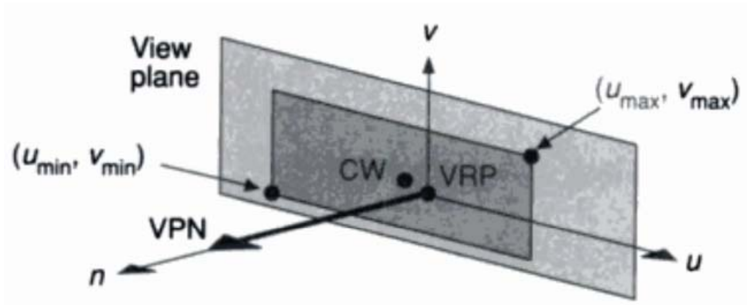
3D viewing = view volume clip → projection → 2D transformation

1. Projection plane (also called view plane)
   - ❑Defined by a point on the plane (view reference point, VRP) + a normal to the plane (view-plane normal, VPN)
   - ❑3D viewing-reference coordinate (VRC) system:
     - ❑Origin → VRP
     - ❑One axis of VRC → VPN (now called the n axis)
     - ❑Another axis → view up vector (VUP), v-axis on the view plane
     - ❑The last axis →u-axis to form the right-handed system
2. A window on the view plane
   - ❑Contents projected outside of the window is not shown
   - ❑Defined by a min and max window coordinates along two axes



Specifying an 3D View

3. The center and direction of projection (DOP)
   - ❑Defined by a projection reference point (PRP) + an indicator of the projection type
     - ❑If the projection type is "perspective", then PRP is the center of projection (COP)
     - ❑If the projection type is "parallel", then the DOP is from the PRP to center of the window on view plane (CW)
       (CW might not be VRP, depending on max/min of (u,v))



View Volume

The View volume bounds that portion of the world that is to be clipped out and projected onto the view plane.

- ❑ For perspective projections:
  - ❑ view volume → semi-infinite pyramid with apex at the PRP and edges passing through the corners of the window
  - ❑ Positions behind the center of projection are not projected
- ❑ For parallel projections:
  - ❑ View volume → infinite parallelepiped with sides parallel to the direction of projection (direction from the PRP to the center of the window)
- ❑ Limiting the view volume (for both projection types)
  - ❑ To eliminate extraneous objects
  - ❑ To eliminate unnecessary computation
    - ❑ e.g. displaying distant objects in perspective projections
  - ➢ setup a front clipping plane and back clipping plane, specified by the signed quantities front distance (F) and back distance (B)

3D View Port

Now we have the view volume (decided by PRP, VRC system, window range u and v, F and B)

To map all contents onto the display surface:

- Map the view volume to the viewport rectangular window with a z-depth
  - front (back) clipping plane → z_{max} (z_{min})
  - u_{min} (u_{max})side of the view volume → x_{min} (x_{max}) plane
  - v_{min} (v_{max})side of the view volume → y_{min} (y_{max}) plane
- To display images for all points after their projections, simply discard the z-component
- For visible surface detection: the hidden surface removal simply uses the z-component to determine which primitives are closer to viewer and should be visible

Matrix for Planar Geometric Projections

Start from the simplest cases:

- ❑ A perspective projection whose projection plane is normal to z-axis at z=d
- ❑ A parallel projection whose projection plane is the z=0 plane
- → Each projection can be defined as a 4 by 4 matrix (coherent with other transformation matrices)

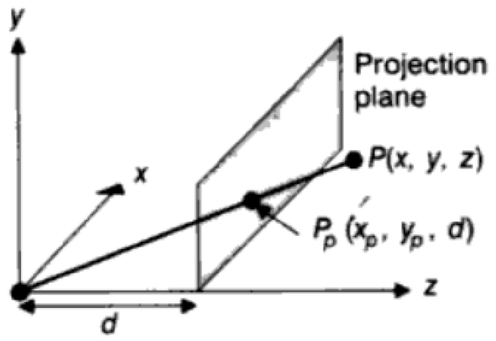
…Later more general cases will be transformed to these cases.

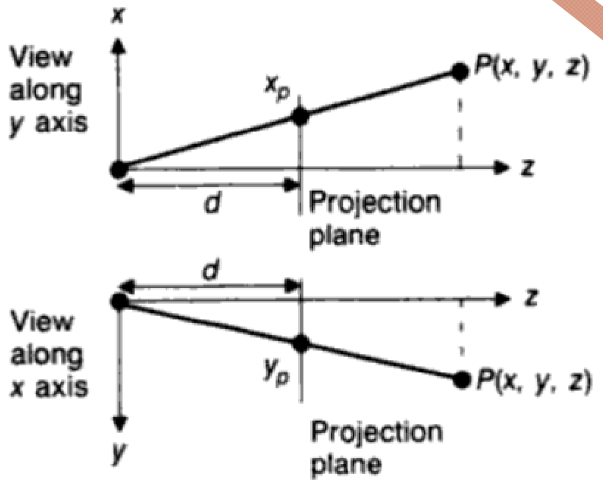
Case 1. Perspective Projection

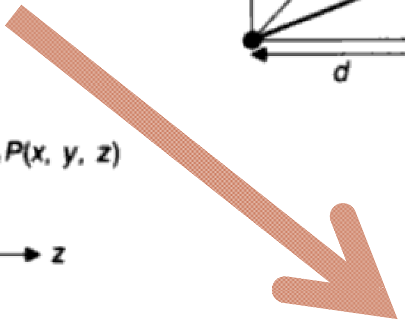
Case 1.1: projection plane at z=d, projection center at the origin, a point P(x,y,z) to be projected onto it as $P_p(x_p, y_p, z_p=d)$

$$\frac{x_p}{d} = \frac{x}{z}; \qquad \frac{y_p}{d} = \frac{y}{z}.$$

$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}, \qquad y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}.$$



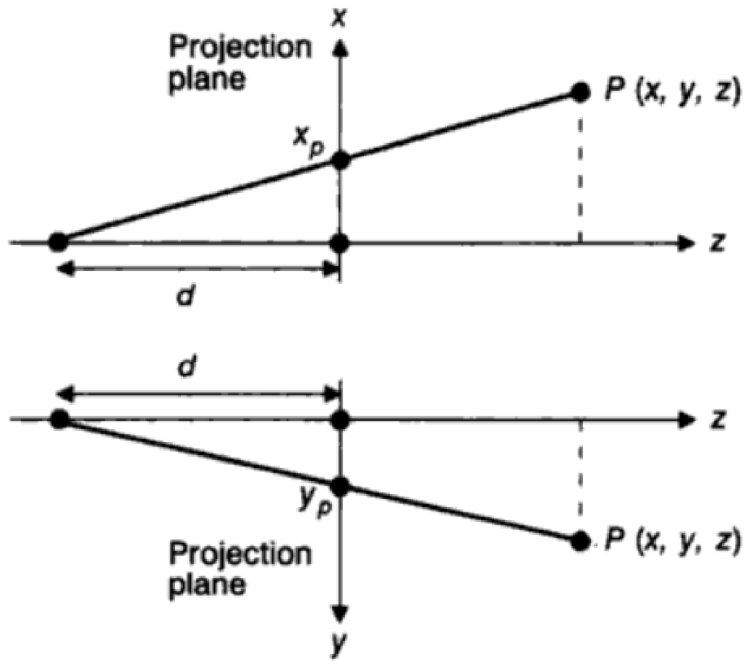




$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Case 1. Perspective Projection

Case 1.2: projection plane at z=0, the center of projection at z=-d, a point P(x,y,z) to be projected onto it as $P_p(x_p, y_p, z_p=0)$



$$\frac{x_p}{d} = \frac{x}{z+d}, \qquad \frac{y_p}{d} = \frac{y}{z+d}.$$

$$x_p = \frac{d \cdot x}{z+d} = \frac{x}{(z/d)+1},$$

$$y_p = \frac{d \cdot y}{z+d} = \frac{y}{(z/d)+1}.$$

$$\Downarrow$$

$$M'_{\text{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}.$$

Case 2. Parallel Projection

Case 2.1: orthographic projection onto projection plane at z=0

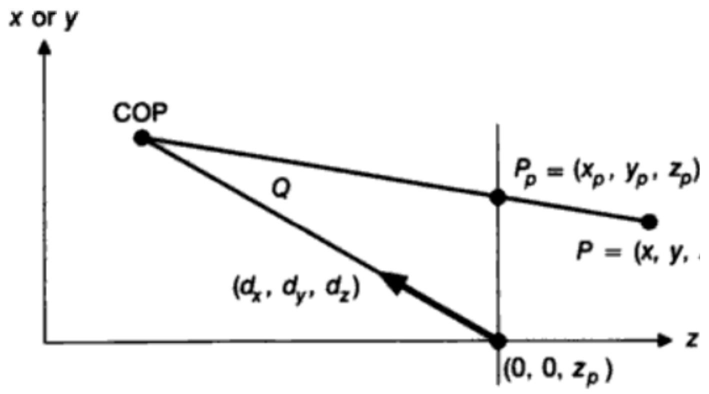
$$x_p = x, \quad y_p = y, \quad z_p = 0.$$

$$M_{\text{ort}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Case 3. A More General Projection

Case 3.1: a perspective projection $P(x,y,z) \rightarrow P_p(x_p, y_p, z_p)$

- ❑ projection plane is perpendicular to z axis
- ❑ the center of projection (COP) has the distance Q from the point $(0,0,z_p)$
- ❑ the direction from $(0,0,z_p)$ to COP is the normalized (d_x, d_y, d_z)



$$x_p = \frac{x - z\frac{d_x}{d_z} + z_p\frac{d_x}{d_z}}{\frac{z_p - z}{Q\,d_z} + 1},$$

$$y_p = \frac{y - z\frac{d_y}{d_z} + z_p\frac{d_y}{d_z}}{\frac{z_p - z}{Q\,d_z} + 1}.$$

$$z_p = z_p\frac{\frac{z_p - z}{Q\,d_z} + 1}{\frac{z_p - z}{Q\,d_z} + 1} = \frac{-z\frac{z_p}{Q\,d_z} + \frac{z_p^2 + z_p Q\,d_z}{Q\,d_z}}{\frac{z_p - z}{Q\,d_z} + 1}.$$

$$M_{general} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p\frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p\frac{d_y}{d_z} \\ 0 & 0 & -\frac{z_p}{Qd_z} & \frac{z_p^2}{Qd_z} + z_p \\ 0 & 0 & -\frac{1}{Qd_z} & \frac{z_p}{Qd_z} + 1 \end{bmatrix}$$

$P_p = (1-t)COP + tP, \quad 0 <= t <= 1$

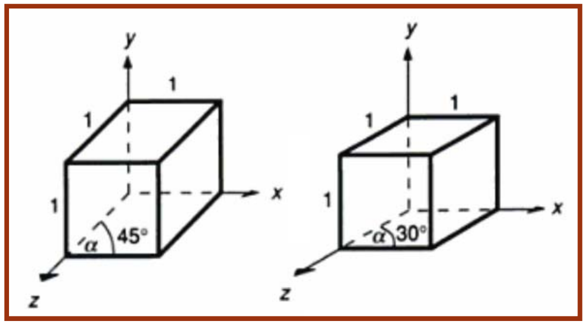
$COP = (0,0,z_p) + Q(d_x, d_y, d_z)$

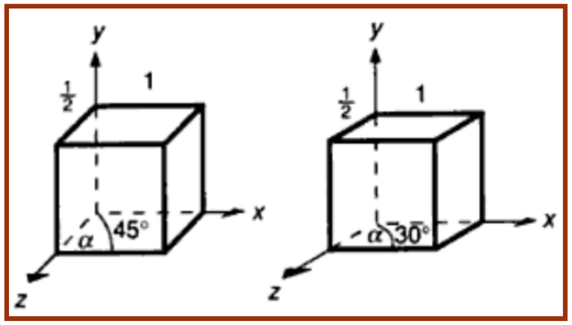
$$t = \frac{z_p - (z_p + Q\,d_z)}{z - (z_p + Q\,d_z)}.$$

Case 3. A More General Projection

Case 3.2: a parallel projection but not orthographic

- Cavalier and Cabinet projection onto the (x,y) plane, with α angle shown below.





$$M_{general} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p\frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p\frac{d_y}{d_z} \\ 0 & 0 & -\frac{z_p}{Qd_z} & \frac{z_p^2}{Qd_z}+z_p \\ 0 & 0 & -\frac{1}{Qd_z} & \frac{z_p}{Qd_z}+1 \end{bmatrix}$$

	Z_p	Q	d_x	d_y	d_z
Cavalier (left)	0	Infinity	cosα	sinα	-1
Cabinet (right)	0	Infinity	(cosα)/2	(sinα)/2	-1

Case 3. A More General Projection

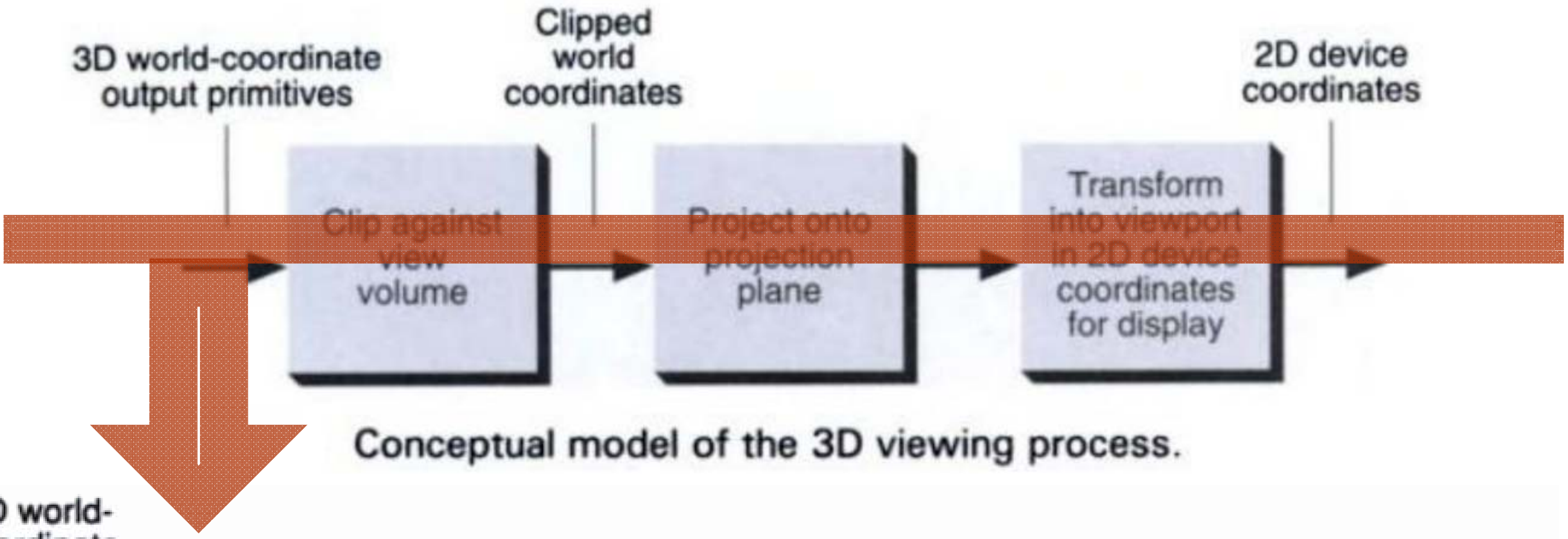
Therefore, this is a general representation (for projection plane z=0, … etc):

The following table includes previously derived projection matrixes.

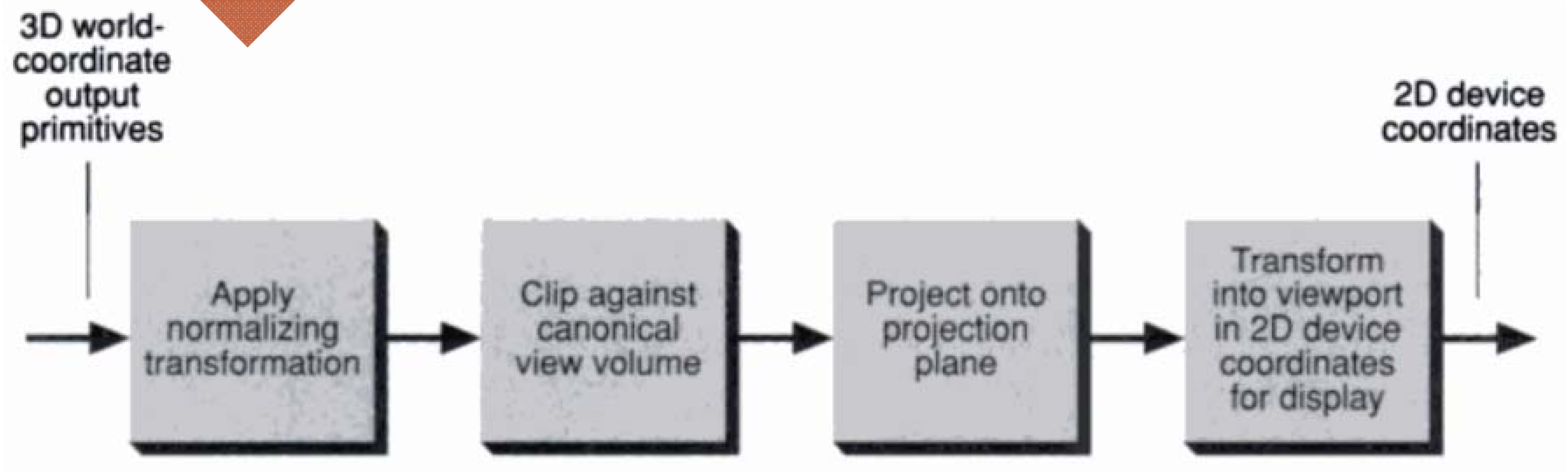
$$M_{general} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p \frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p \frac{d_y}{d_z} \\ 0 & 0 & -\frac{z_p}{Qd_z} & \frac{z_p^2}{Qd_z} + z_p \\ 0 & 0 & -\frac{1}{Qd_z} & \frac{z_p}{Qd_z} + 1 \end{bmatrix}$$

	Z_p	Q	d_x	d_y	d_z
$M_{cavalier}$	0	Infinity	cosα	sinα	-1
$M_{cabinet}$	0	Infinity	(cosα)/2	(sosα)/2	-1
M_{ort}	0	Infinity	0	0	-1
M_{per}	d	d	0	0	-1
M'_{per}	0	d	0	0	-1

3D Viewing Process


3D world-coordinate output primitives
Clipped world coordinates
2D device coordinates
Clip against view volume
Project onto projection plane
Transform into viewport in 2D device coordinates for display

Conceptual model of the 3D viewing process.


3D world-coordinate output primitives
2D device coordinates
Apply normalizing transformation
Clip against canonical view volume
Project onto projection plane
Transform into viewport in 2D device coordinates for display

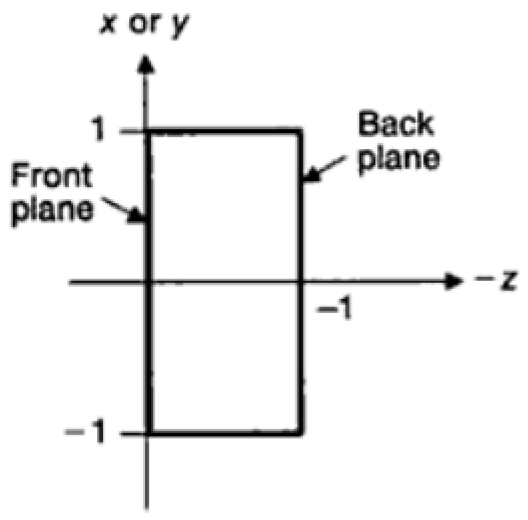
Implementing Planar Geometric Projections

For Computational efficiency!

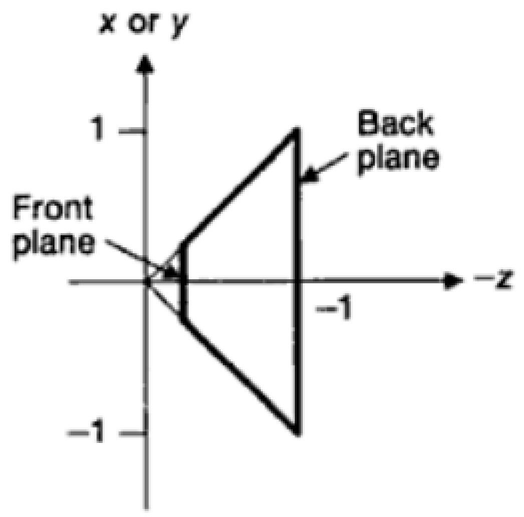
Certain view volumes are easier to clip against (canonical view volumes):

- ❑ For parallel-projection view volume:
  (x,y,z) in {([-1,1],[-1,1],[-1,0])}
- ❑ For perspective-projection view volume:
  (x,y,z) in {[-z, z],[-z,z], [-z_{min}, -1]}

Normalizing transformations N_{par} and N_{per}
transform an arbitrary view volume into the canonical view volumes



(a) Parallel



(b) Perspective

Normalizing Transformation Matrix N_{par}

Derive N_{par} for the most general case → the oblique parallel projection

General Pipeline:

1. Translate the VRP to the origin
2. Rotate VRC : the (u,v,n)-axis (VPN) → (x,y,z)-axis
3. Shear : the direction of projection (DOP) → z-axis
4. Translate and scale into the parallel-projection canonical view volume

View-orientation matrix ← step 1, 2
View-mapping matrix ← step 3,4

Example (see the figure in the next page)

1. Translation T(-VRP)
2. Rotation R(vectors u,v,n → x,y,z)
3. DOP = CW – PRP = $[\frac{u_{\max}+u_{\min}}{2} \quad \frac{v_{\max}+v_{\min}}{2} \quad 0 \quad 1]^T - [prp_u \quad prp_v \quad prp_n \quad 1]^T$

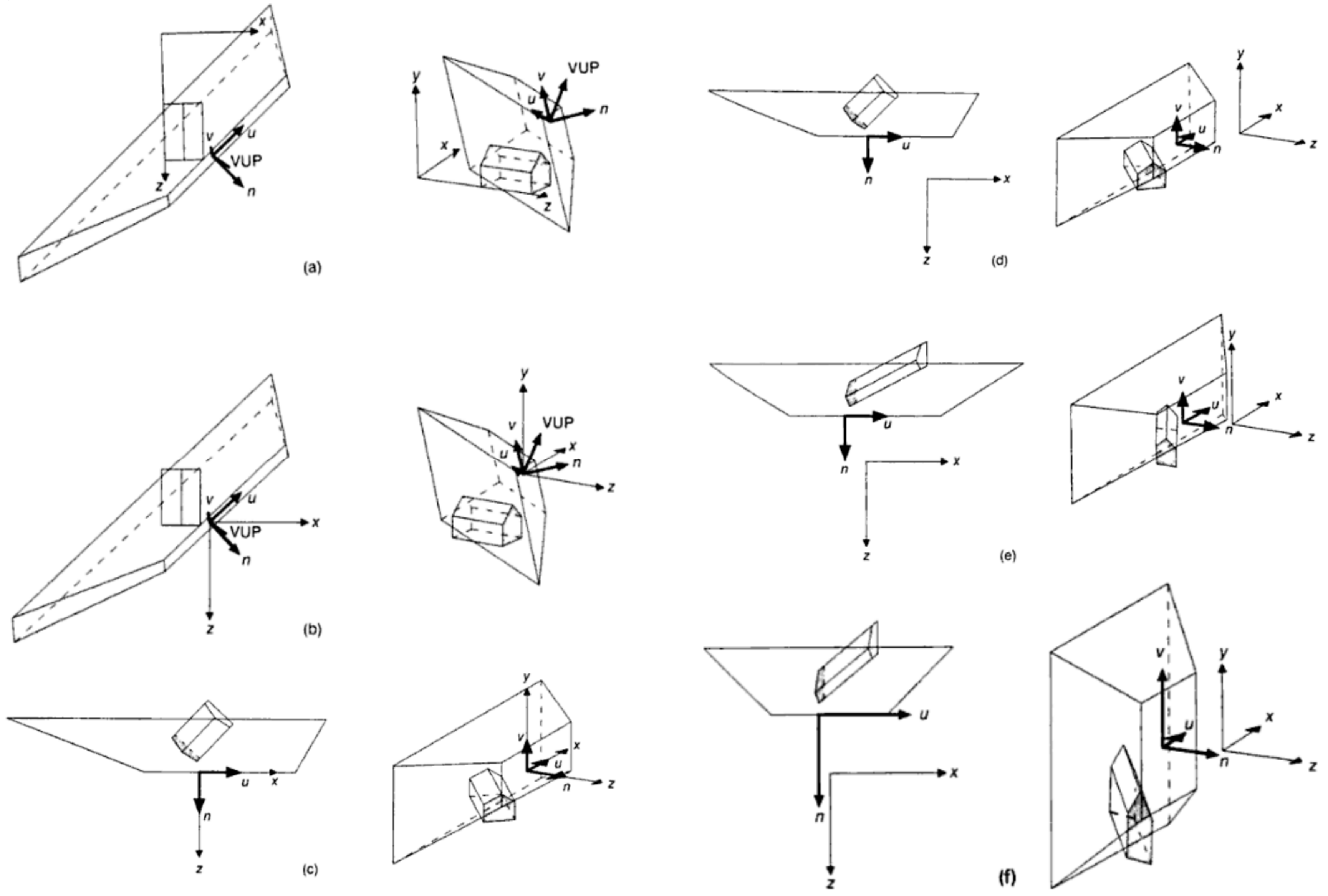
   The shear : $Sh_{xy}(shx_{par}, shy_{par})$ $\longrightarrow$ $SH_{xy}(shx_{par}, shy_{par}) = \begin{bmatrix} 1 & 0 & shx_{par} & 0 \\ 0 & 1 & shy_{par} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

4. Translation: $T_{par} = T(-\frac{u_{\max}+u_{\min}}{2}, -\frac{v_{\max}+v_{\min}}{2}, -F)$

   and Scaling: $S_{par} = S(\frac{2}{u_{\max}-u_{\min}}, \frac{2}{v_{\max}-v_{\min}}, \frac{1}{F-B})$

$$shx_{par} = -\frac{dop_x}{dop_z}, shy_{par} = -\frac{dop_y}{dop_z}$$

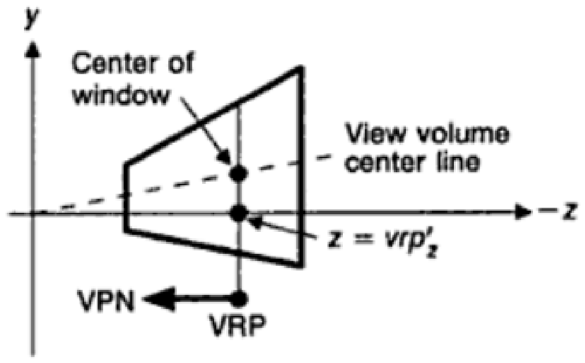
Normalizing Transformation Matrix N_{par}



Normalizing Transformation Matrix N_{per}

General Pipeline:

1. Translate the VRP to the origin
2. Rotate VRC : the (u,v,n)-axis (VPN) → (x,y,z)-axis
3. Translate : COP (given by PRP) → origin
4. Shear : the center line of the view volume → z axis
5. Scale into the canonical view volume

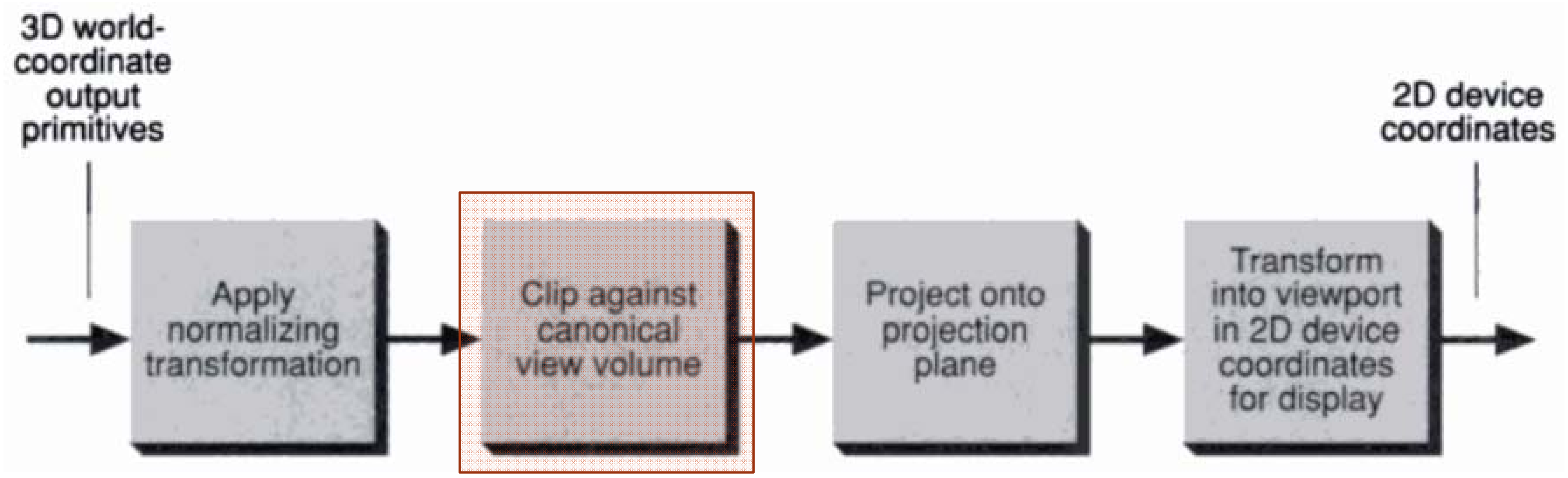


Example

1. Same as N_{par}
2. Same as N_{par}
3. T(-PRP)
4. Center line = CW – PRP= $\begin{bmatrix} \frac{u_{\max}+u_{\min}}{2} & \frac{v_{\max}+v_{\min}}{2} & 0 & 1 \end{bmatrix}^T - \begin{bmatrix} prp_u & prp_v & prp_n & 1 \end{bmatrix}^T$

   → Shear (same as N_{par})
5. Scaling: $S_{per} = S\left(\frac{2vrp'_z}{(u_{\max}-u_{\min})(vrp'_z+B)}, \frac{2vrp'_z}{(v_{\max}-v_{\min})(vrp'_z+B)}, \frac{-1}{vrp'_z-B}\right)$

Clipping Against a Canonical View Volume

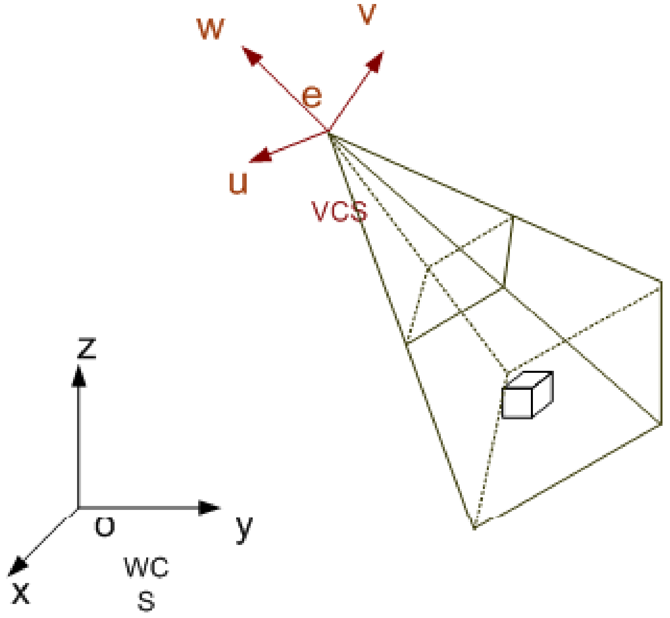


3D world-coordinate output primitives
Apply normalizing transformation
Clip against canonical view volume
Project onto projection plane
Transform into viewport in 2D device coordinates for display
2D device coordinates

- Geometric Clipping Algorithm
- Clipping in Homogeneous Coordinates

Projection Transformations in OpenGL

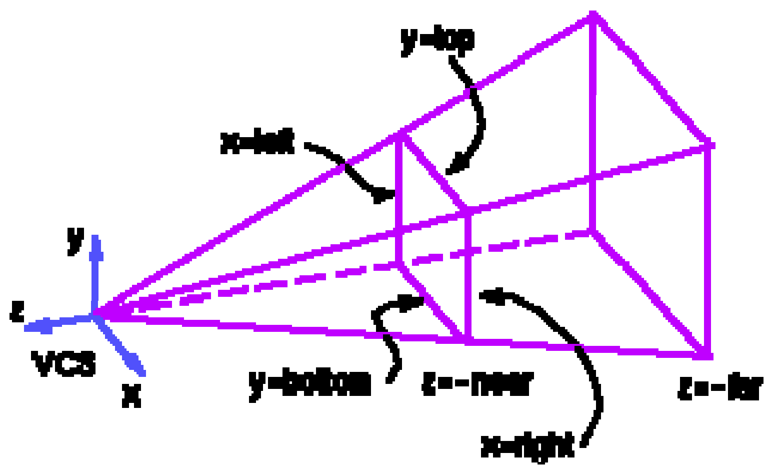
- Perspective Projection



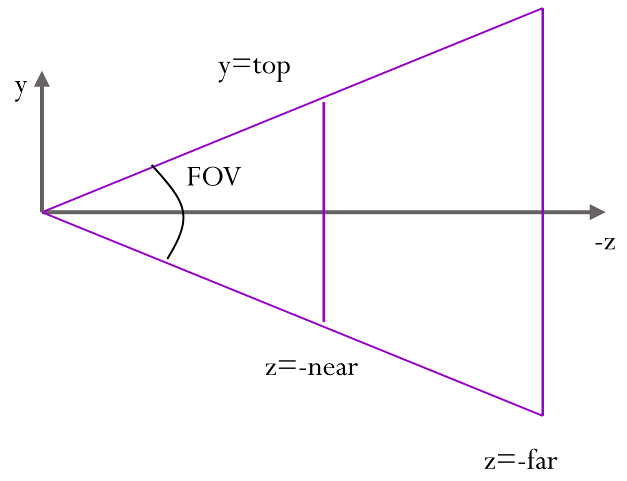
Viewing Coordinate System (*VCS*)

Projection Transformations in OpenGL (cont.)

- **Perspective Projection**
  - void glFrustum(double left, double right, double bottom, double top, double near, double far);

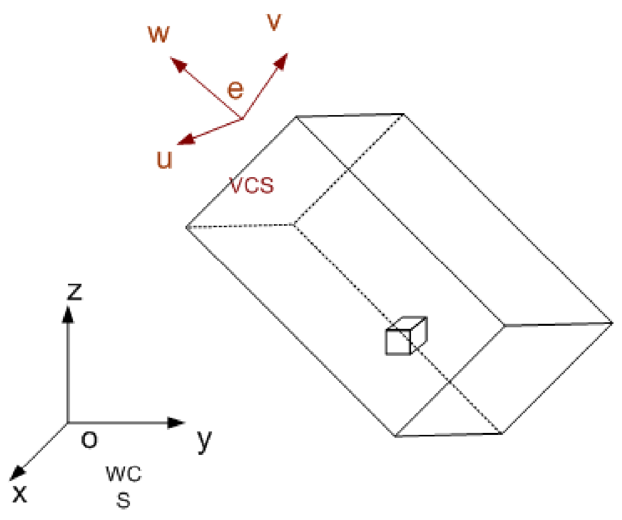


  - void gluPerspective(double fovy, double aspect, double near, double far);



Projection Transformations in OpenGL (cont.)

- **Orthographic Projection**
  - void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar);
  - void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

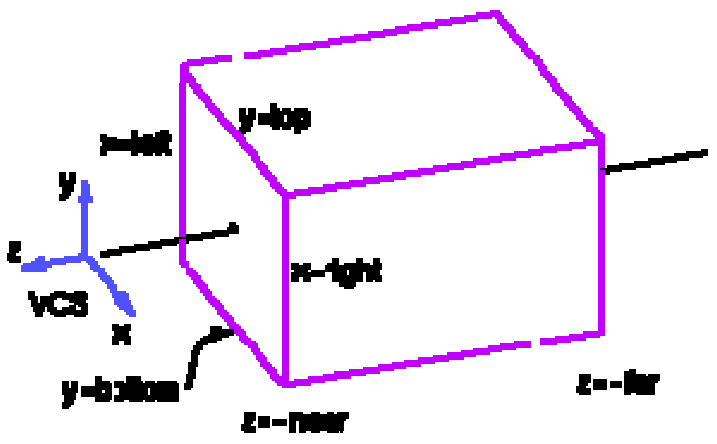


- **Routines:**
  - First, put:

    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();
  - Then, put one of:

    glFrustrum/gluPerspective or glOrtho


y=top
x=left
y
z
VCS
x
x=right
y=bottom
z=-near
z=-far