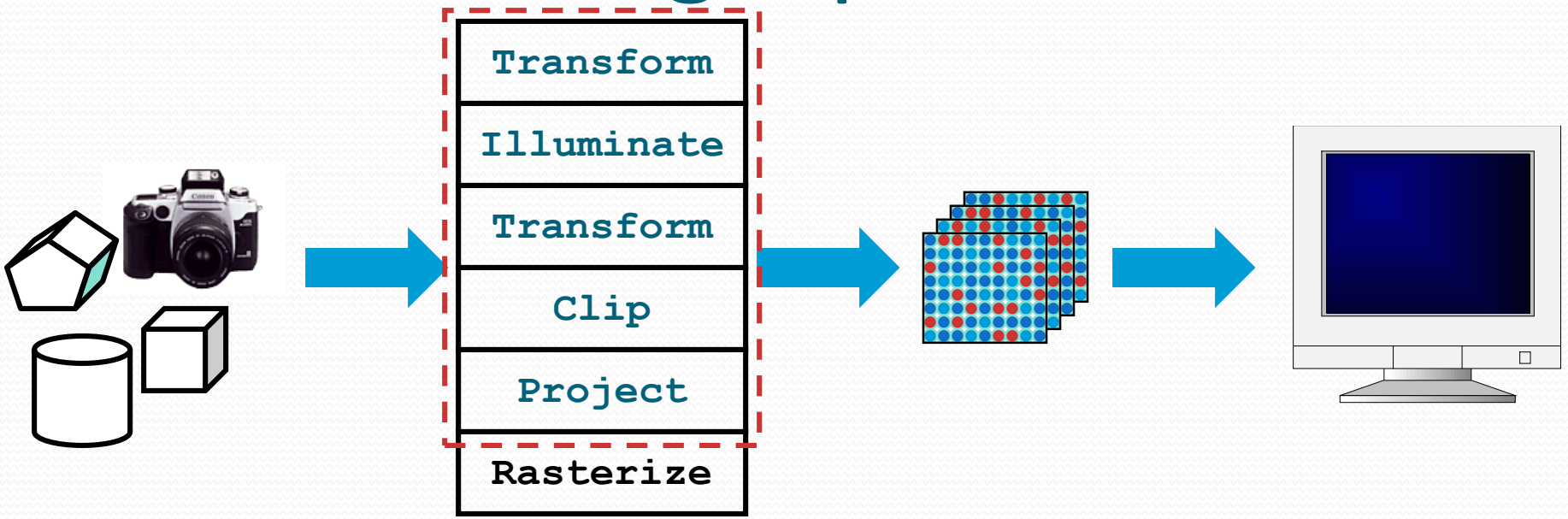
Transformation

The Rendering Pipeline: 3-D



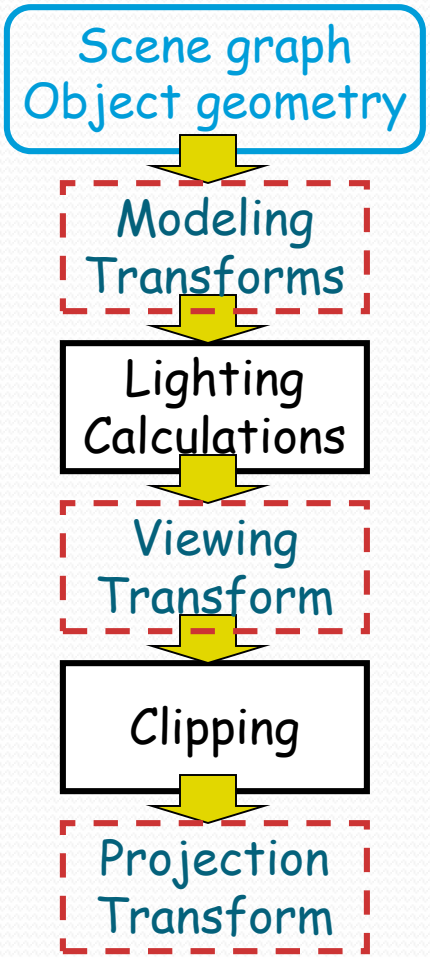
Model & Camera Parameters

Rendering Pipeline

Framebuffer

Display

The Rendering Pipeline: 3-D



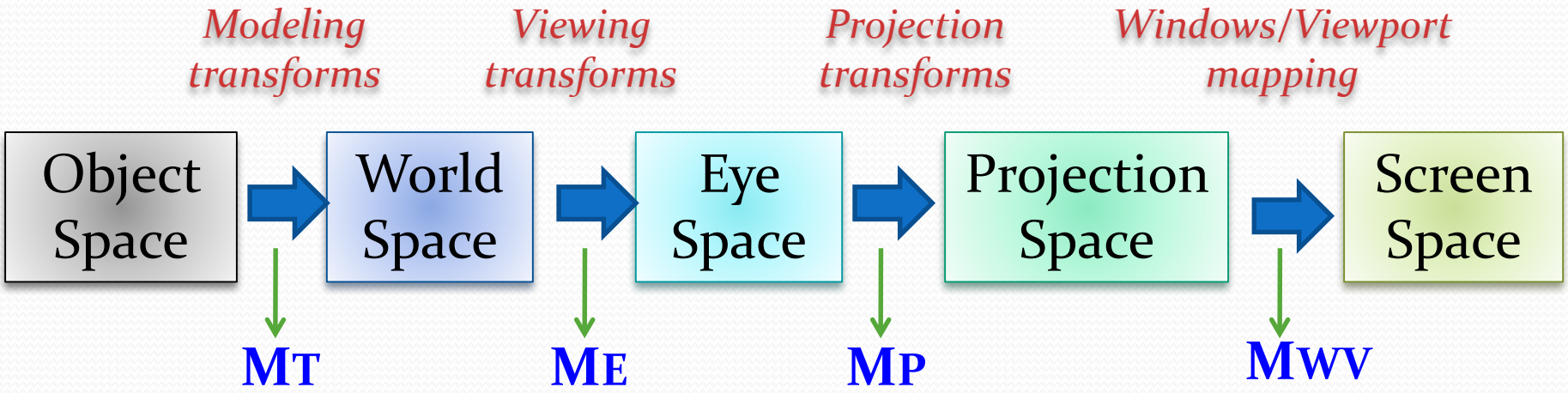
Result:

- All vertices of scene in shared 3-D "world" coordinate system
- Vertices shaded according to lighting model
- Scene vertices in 3-D "view" or "camera" coordinate system
- Exactly those vertices & portions of polygons in view frustum

Rendering: Transformations

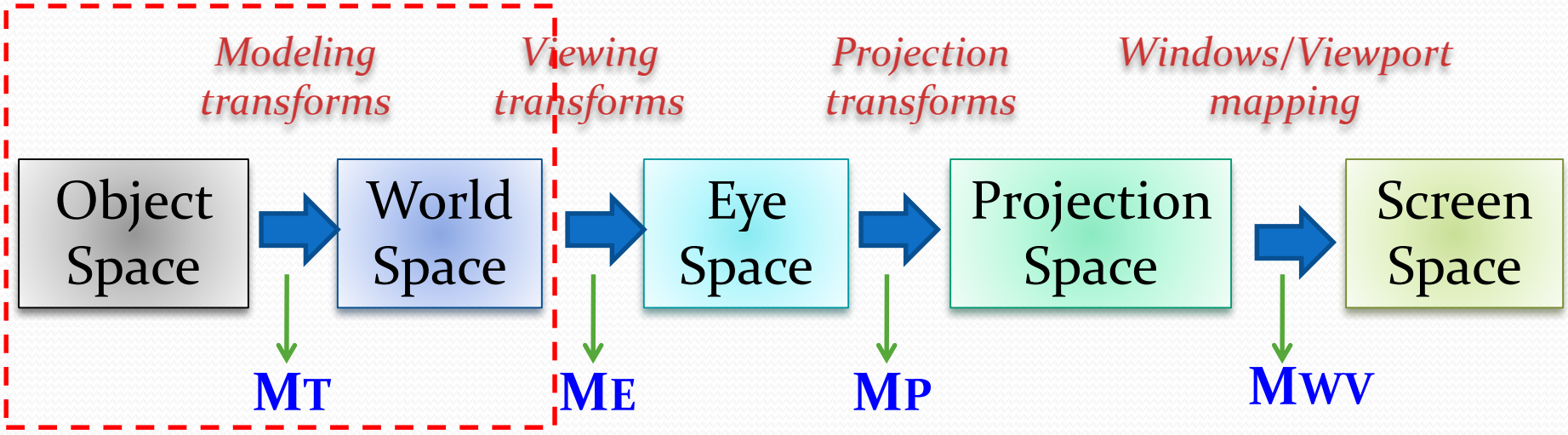
- So far, discussion has been in *screen space*
- But model is stored in *model space*
  (a.k.a. object space or world space)
- Three sets of geometric transformations:
  - *Modeling transforms*
  - *Viewing transforms*
  - *Projection transforms*

3D Rendering Pipeline



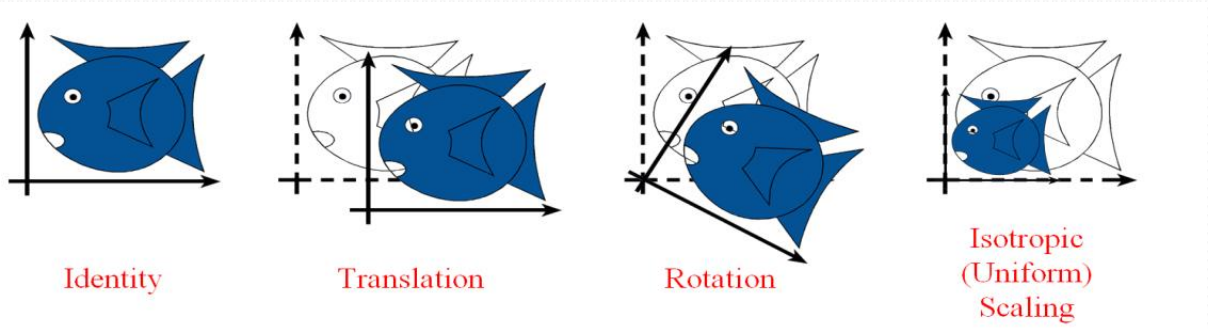
$$M = M_T * M_E * M_P * M_{WV}$$

3D Rendering Pipeline

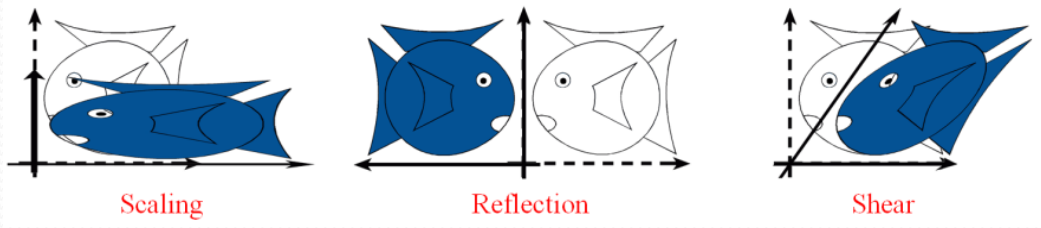


Rendering: Transformation

- **Geometric Transformations**
  - Translation
  - Rotation
  - Scaling



- **Linear** (preserves parallel lines)
  - Non-uniform scales, shears or skews



Rendering: Model Transformation

- 2D Translations

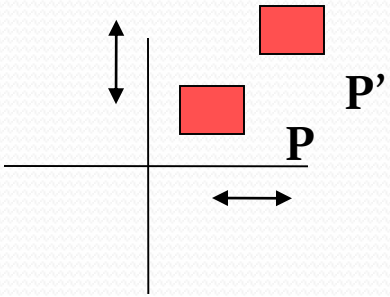
Point P defined as $P(x, y)$,

translate to Point $P'(x', y')$ a distance d_x parallel to x axis, d_y parallel to y axis.

$x' = x + d_x \quad y' = y + d_y$

Define the column vectors

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$



Now

$P' = P + T$

Rendering: Model Transformation

- 2D Scaling

  Point P defined as $P(x, y)$,

  Perform a scale (stretch) to Point $P'(x', y')$ by a factor s_x along the x axis, and s_y along the y axis.

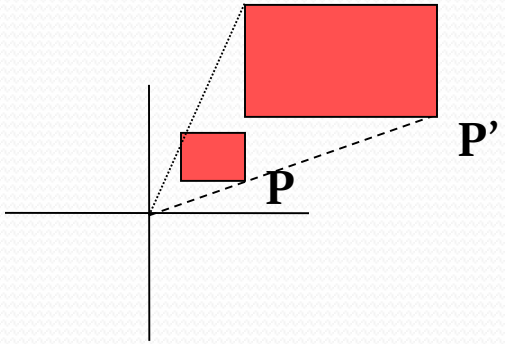
  $x' = s_x.x, \quad y' = s_y.y$

  Define the matrix

  $$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

  Now

  $$P' = S \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$



Rendering: Model Transformation

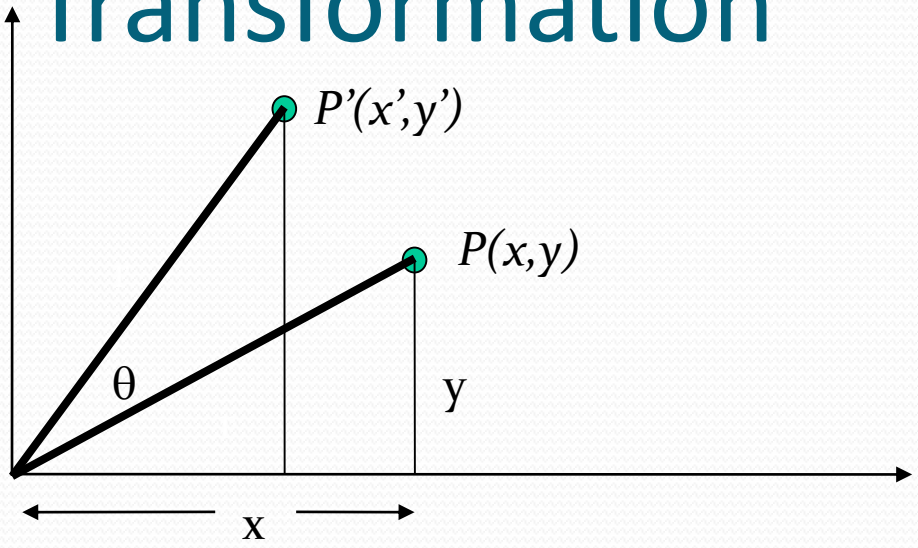
- 2D Rotation

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

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

Rewriting in matrix form gives us :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$



$$\text{Define the matrix } R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad P' = R \cdot P$$

Homogenous Points

- Add 1D, but constrain that to be equal to 1 (x,y,1)
- Homogeneity means that any point in 2-space can be represented by an infinite variety of homogenous 3D points
  - (2 3 1) = (4 6 2) = (3 4.5 1.5)
- Why?
  - 3D allows as to include 2D translation in matrix form

Rendering: Model Transformation

- 2D Transformations as 3x3 Matrices

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

Translate

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

Scale

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

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Rendering: Model Transformation

- Matrix Composition

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}'$ = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ $\mathbf{p}$

Be aware: order of transformations matters

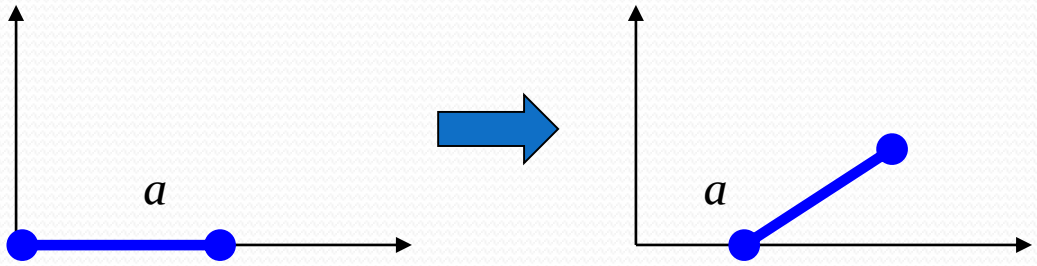
– *Matrix multiplication is not commutative*

$\mathbf{p}' = T * R * S * \mathbf{p}$

"Global" "Local"

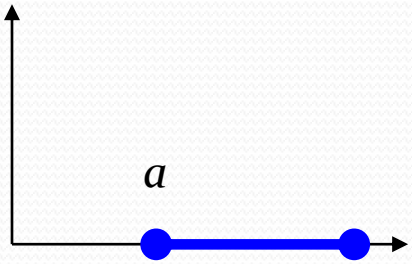
Rendering: Model Transformation

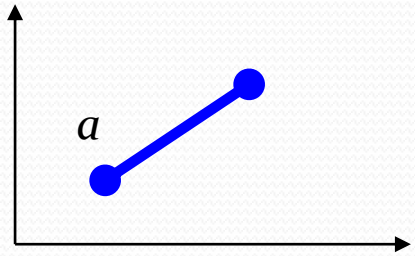
- Matrix Composition
- What if we want to rotate and translate?



Rendering: Model Transformation

- Matrix Composition
- Translate → Rotate

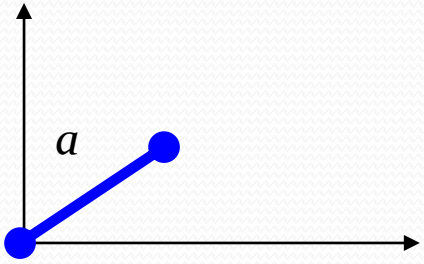


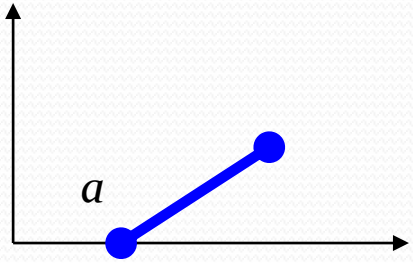


Wrong

R(45)

- Rotate → Translate





Correct

Translations (3D)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$T(t_x, t_y, t_z)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Properties of Translation

$$T(0,0,0)\,\mathbf{v} = \mathbf{v}$$

$$T(s_x, s_y, s_z)\,T(t_x, t_y, t_z)\mathbf{v} = T(s_x + t_x, s_y + t_y, s_z + t_z)\,\mathbf{v}$$

$$T(s_x, s_y, s_z)\,T(t_x, t_y, t_z)\mathbf{v} = T(t_x, t_y, t_z)\,T(s_x, s_y, s_z)\mathbf{v}$$

$$T^{-1}(t_x, t_y, t_z)\,\mathbf{v} = T(-t_x, -t_y, -t_z)\,\mathbf{v}$$

Rotations (3D)

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

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

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

Scaling (3D)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \end{bmatrix}$$

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

Uniform scaling ***iff*** $s_x = s_y = s_z$

Homogenous Points

- Add 1D, but constrain that to be equal to 1 (x,y,z,1)
- Homogeneity means that any point in 3-space can be represented by an infinite variety of homogenous 4D points
  - (2 3 4 1) = (4 6 8 2) = (3 4.5 6 1.5)
- Why?
  - 4D allows as to include 3D translation in matrix form

Rendering: Model Transformation

- Basic 3D Transformations

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Identity

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ \boldsymbol{z'} \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Scale

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ \boldsymbol{z'} \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{t}_x \\ 0 & 1 & 0 & \boldsymbol{t}_y \\ 0 & 0 & 1 & \boldsymbol{t}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Translation

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

Mirror about Y/Z plane

Rendering: Model Transformation

- Basic 3D Transformations

Rotate around Z axis:

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

Rotate around Y axis:

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Rotate around X axis:

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

Homogeneous Coordinates

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

can be represented as

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

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

where $x = \frac{X}{W}, \quad y = \frac{Y}{W}, \quad z = \frac{Z}{W}$

Translation Revisited

homogeneous transformation

$$T(t_x, t_y, t_z)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation & Scaling Revisited homogeneous transformation

$$R_x(\theta)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$S(s_x, s_y, s_z)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Composing Transformations

$$\mathbf{v}' = S\mathbf{v}$$

$$\mathbf{v}'' = R\mathbf{v}' = RS\mathbf{v}$$

$$\mathbf{v}''' = T\mathbf{v}'' = TR\mathbf{v}' = TRS\mathbf{v}$$

$$\mathbf{v}''' = M\mathbf{v}$$

where

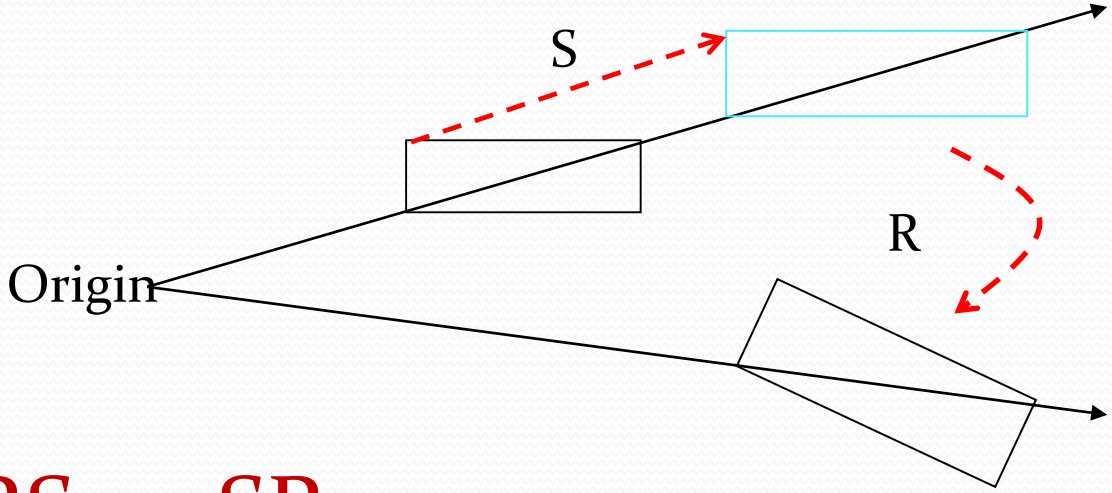
$$M = TRS$$

Scale and Rotation Composed

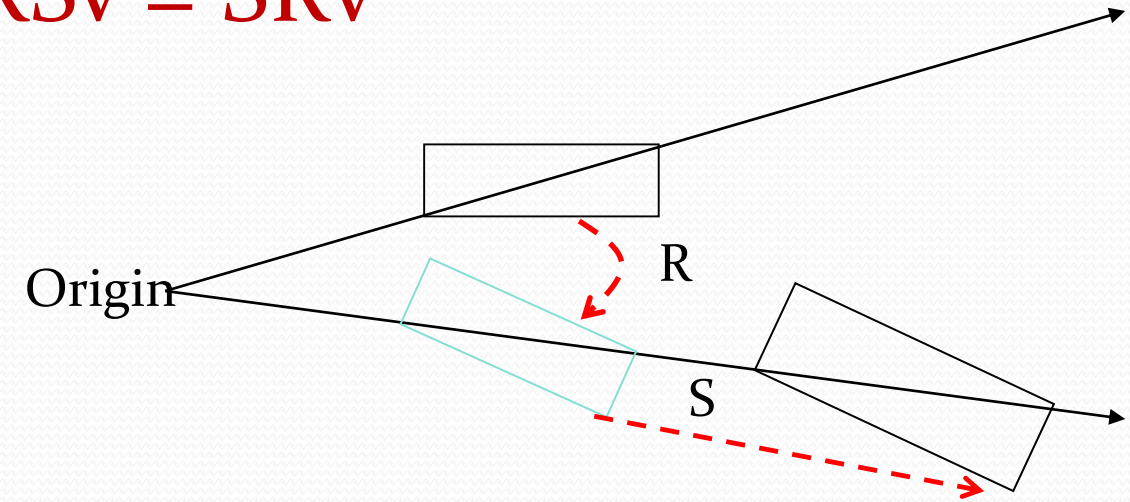
- Xac = Xab Xbc
  - chain matrices to arbitrary length (fully associative)

- Xac = T S R or T R S
  - rotation and scaling commute - neither change origin
  - Translations do not commute with R or S
    - due to change of origin (fixed-point)

Scale and Rotation Composed

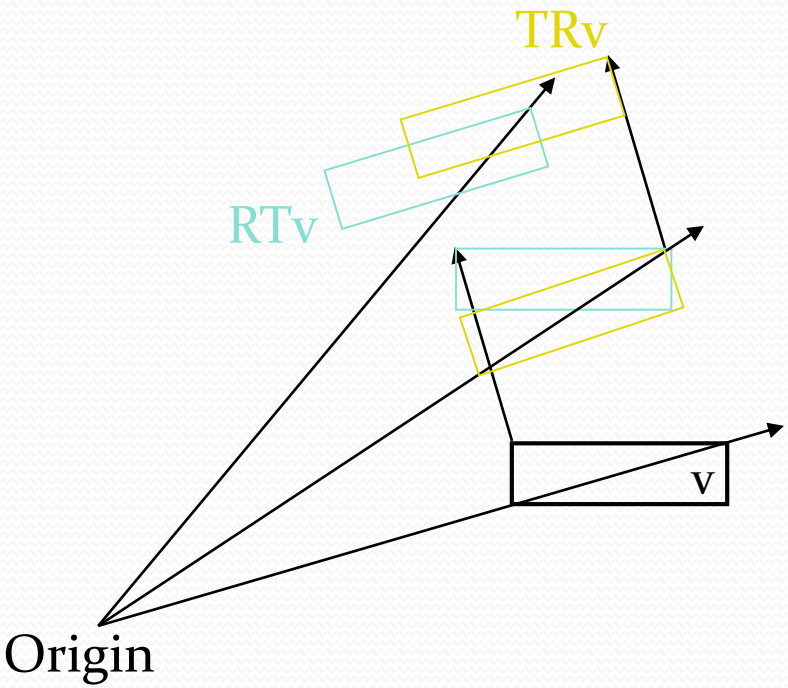


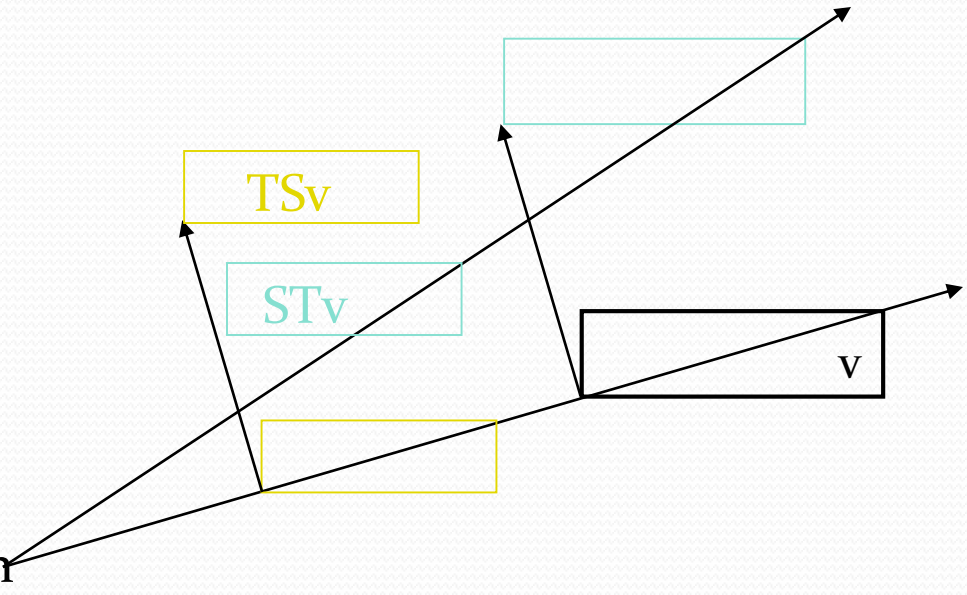
RSv = SRv



Translation

- Translations do not commute
- TR ≠RT TS ≠ST





Rotation and translation composed

- Y-rot and translation = $X_{ab} = T\ R$
  - note translation (in space-a) is applied after rotation

$$\begin{pmatrix} \cos ø & 0 & \sin ø & xt \\ 0 & 1 & 0 & yt \\ -\sin ø & 0 & \cos ø & zt \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - above is different transformation (and matrix) than if we define as: $X_{ab} = R\ T$ where the translation occurs first in space-b

Decomposing Transformations

- Given 4x4 view/projection matrix - we can decompose it into T R S or T S R
  (assume w = 1, and matrix has no shear or non-uniform scaling)
  - T is upper three elements of right column
  - R is 3x3 in UL corner
    - may need to normalize so all ||columns,rows|| = 1
  - R = R' S or S R' - rotation and scaling are mixed in same elements
  - use any row/col and compute scale factor
    $K = 1 / (a^2 + b^2 + c^2)^{1/2}$
  - multiply all elements of 3x3 R : R' = K(R)
  - R' is normalized (unitary) rotation matrix
  - S is a diagonal matrix with elements = 1/K - scale factor extracted from R

Inverse Transformations

- Rotation : $Rca = R^{-1}ac = R^{t}ac$
  - Transpose is inverse for unitary (pure) rotation mat
    - Det = +/- 1
    - Rows and col are orthogonal
    - ||row|| = ||col|| = 1
- Translation: $Tca = T^{-1}ac = -Tac$
  - negate elements of column
- Scale: $Sca = S^{-1}ac = 1/Sac$
  - replace diagonal elements with their reciprocal
- Given $Xac = T\ S\ R$
  - Then we can get the inverse : $X^{-1}ac = R^{-1}\ S^{-1}\ T^{-1}$
  - We know it's the inverse because

    $\mathbf{I} = X^{-1}ac\ Xac = (R^{-1}\ S^{-1}\ T^{-1})\ (T\ S\ R)$

Rendering: Model Transformation

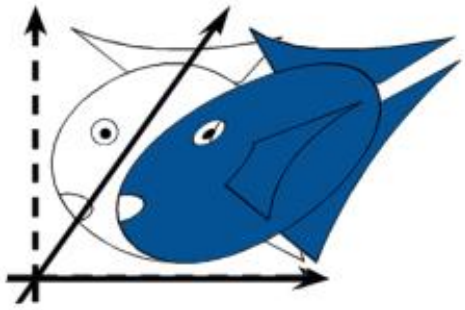
- Affine Transformations
  - Affine transformations are combinations of …
    - Linear transformations, and
    - Translations
  - Properties of affine transformations:
    - Origin does not necessarily map to origin
    - Lines map to lines
    - Parallel lines remain parallel
    - Ratios are preserved
    - Closed under composition

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Rendering: Model Transformation

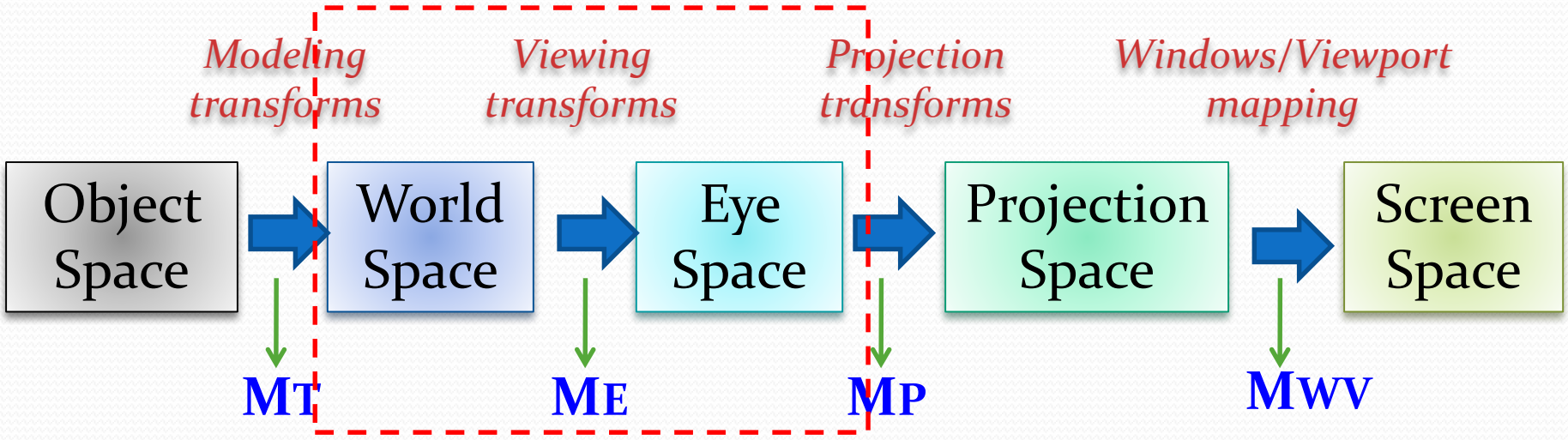
- Rigid Body Transformations
  - Preserve lengths of the sides
  - Preserve interval angles
  - Parallel lines remain parallel

- Affine Transformations
  - Do not preserve lengths of the sides
  - Do not preserve interval angles
  - Parallel lines remain parallel



Shear

3D Rendering Pipeline

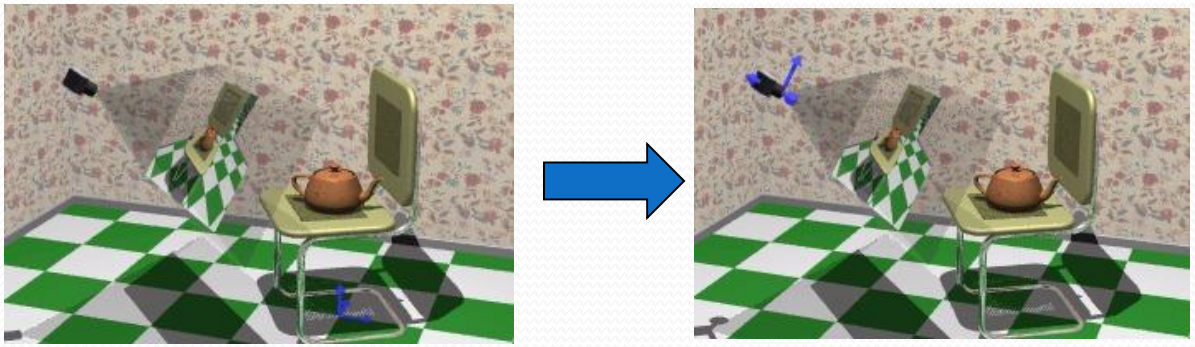


Rendering: Viewing Transform

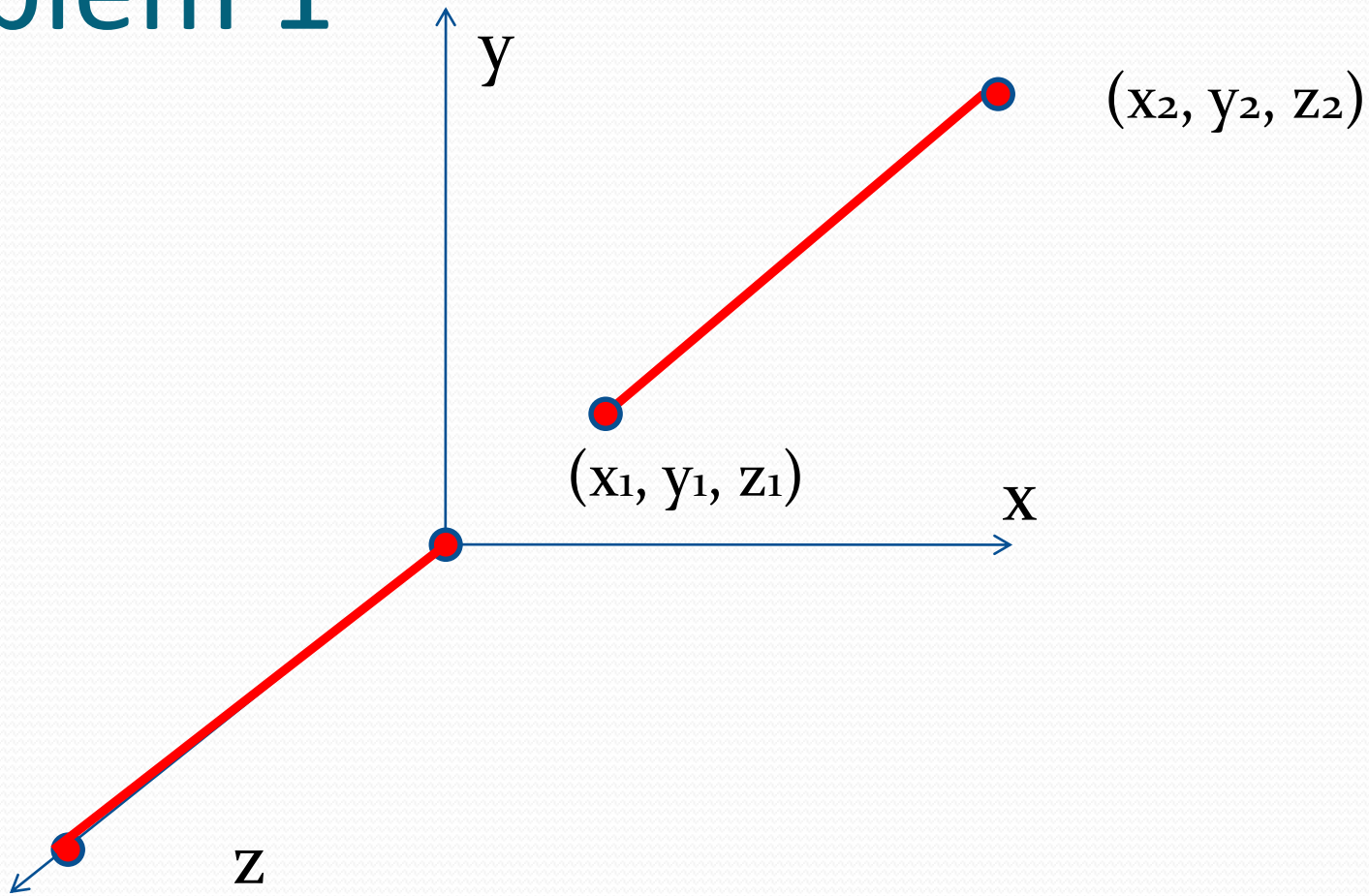
- Rotate & translate the world to lie directly in front of the camera
  - Typically place camera at origin
  - Typically looking down -Z axis

$$\begin{bmatrix} \hat{r} & -\hat{r}\cdot\overline{eye} \\ \hat{u} & -\hat{u}\cdot\overline{eye} \\ -\hat{l} & \hat{l}\cdot\overline{eye} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

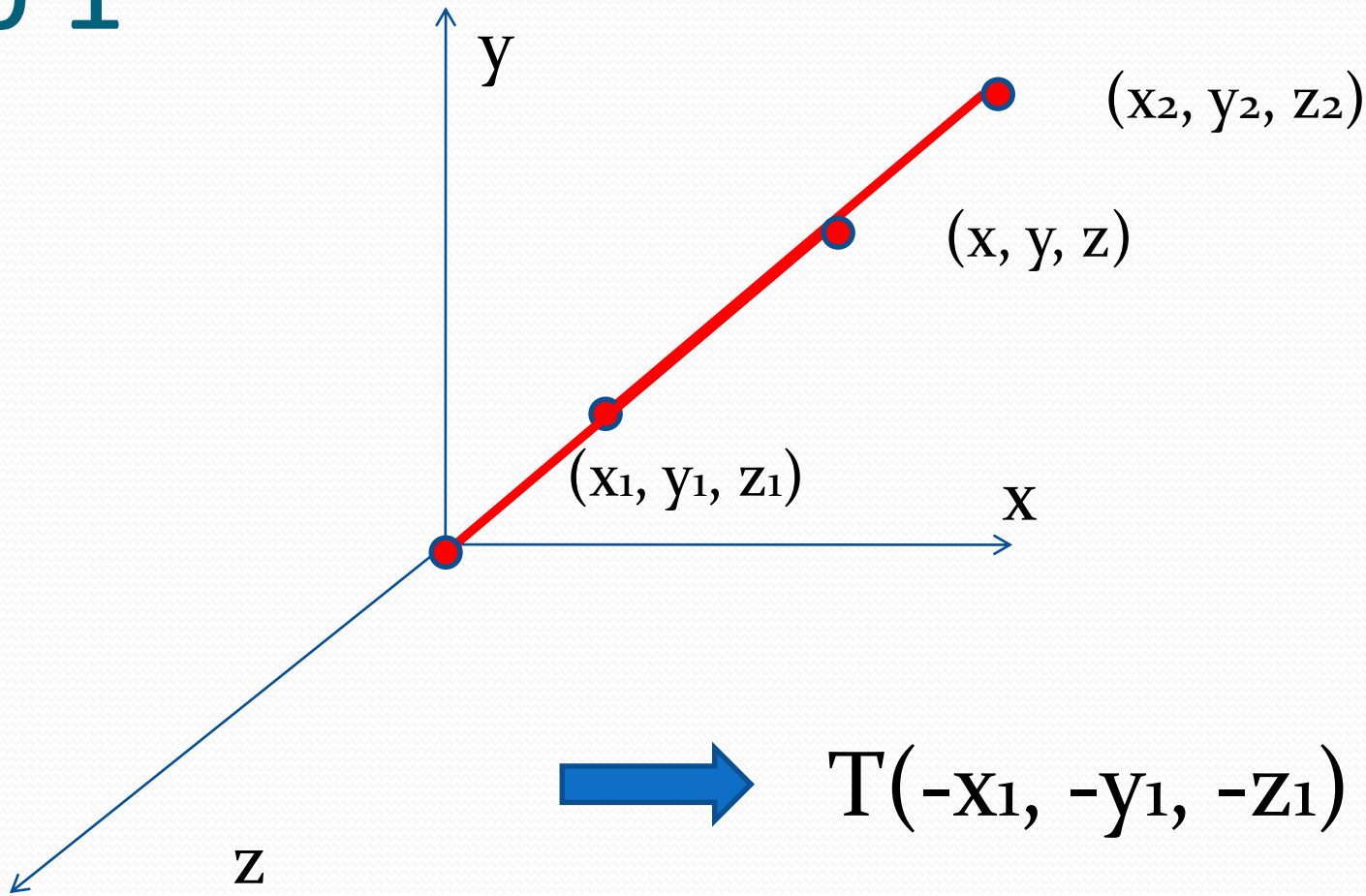
- World coordinates ⮜ view coordinates



Problem 1

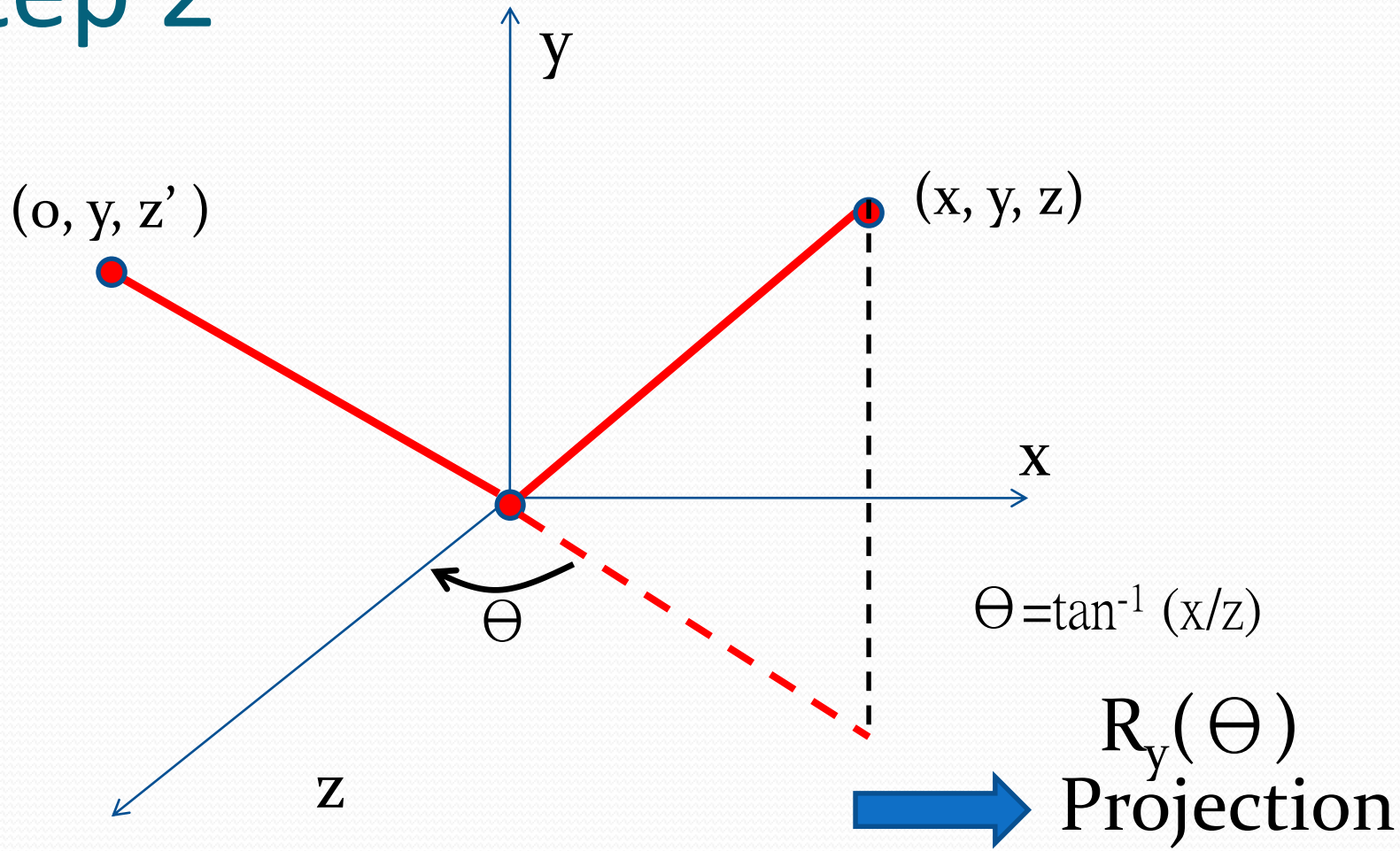


Step 1

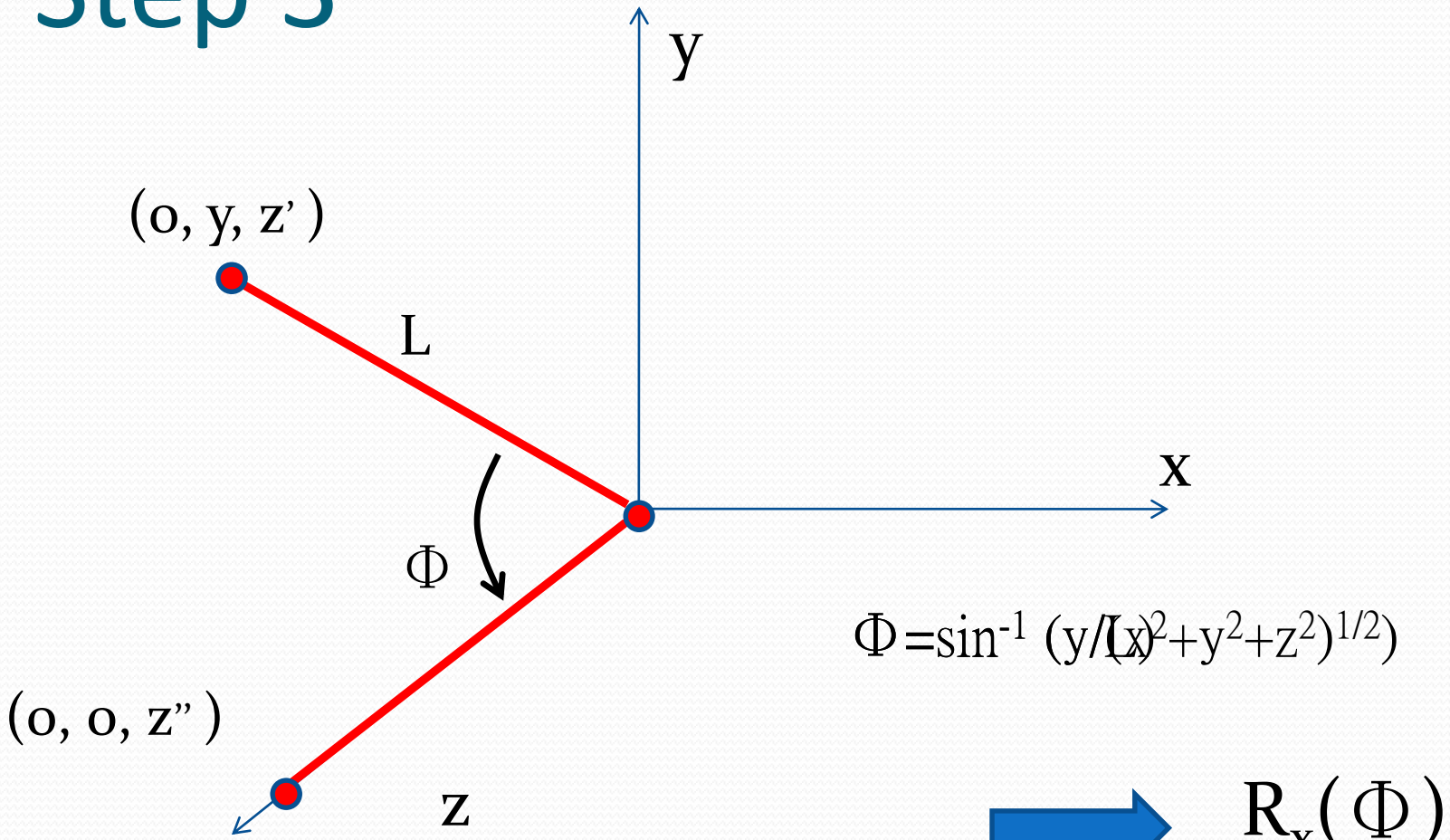


$T(-x_1, -y_1, -z_1)$

Step 2



Step 3



$$\Phi = \sin^{-1}\left(y/(x^2+y^2+z^2)^{1/2}\right)$$

$$R_x(\Phi)$$

Summary

$$M = R_x(\Phi) * R_y(\Theta) * T(-x_1, -y_1, -z_1)$$

Bring one end of the vector to the origin

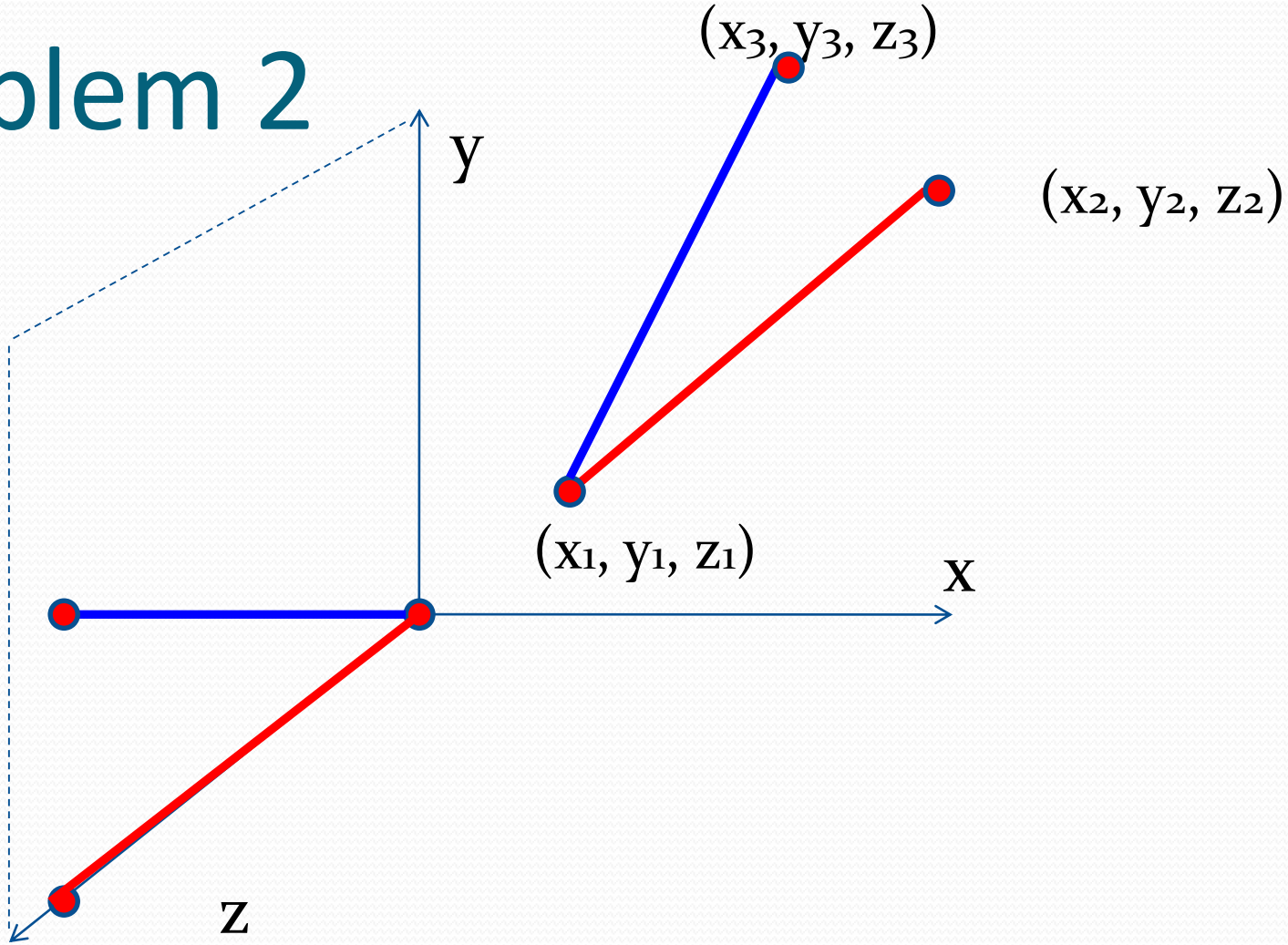
Align vector in y-z plan

Align vector with z axis

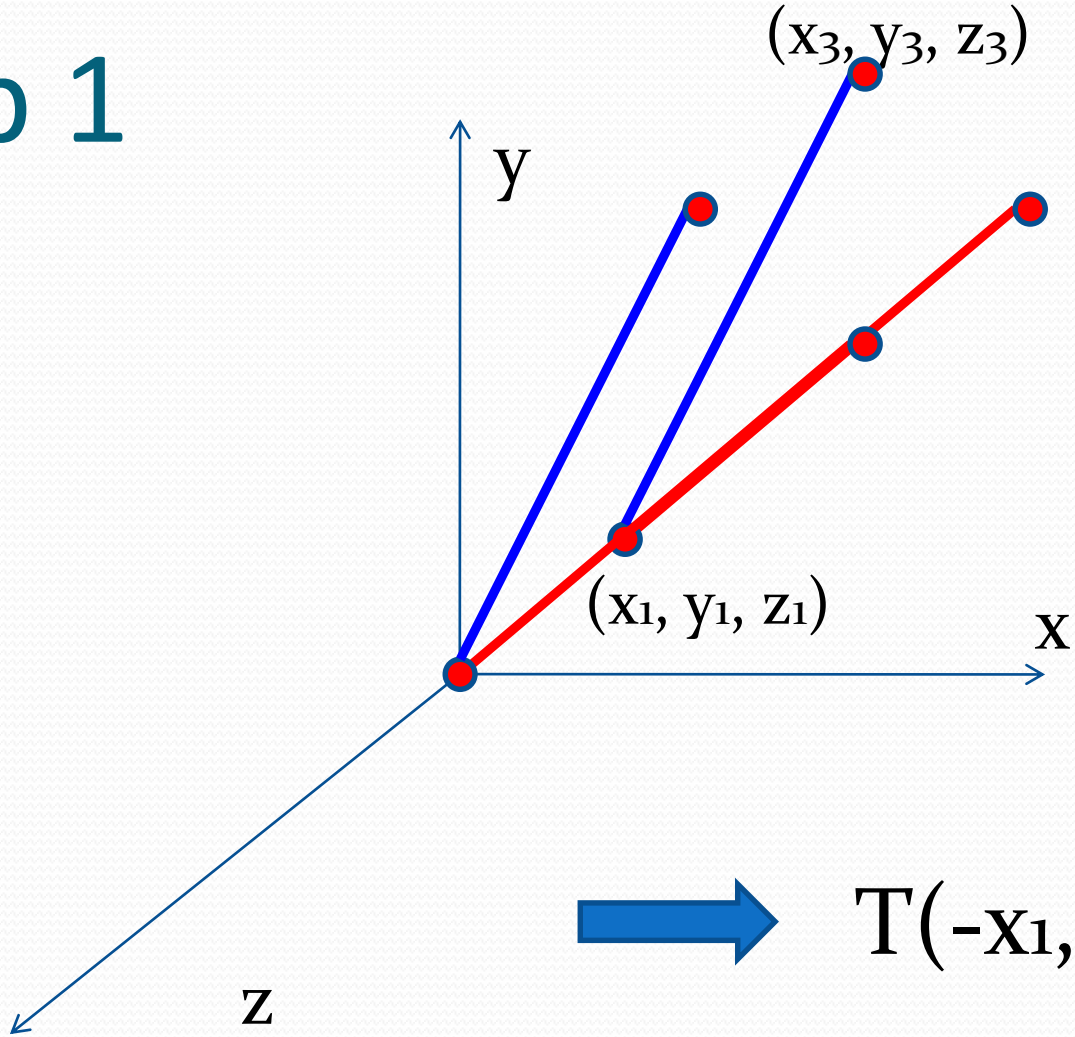
Go back

$$M = T(x_1, y_1, z_1) * R_y(-\Theta) * R_x(-\Phi) * R_x(\Phi) * R_y(\Theta) * T(-x_1, -y_1, -z_1)$$

Problem 2

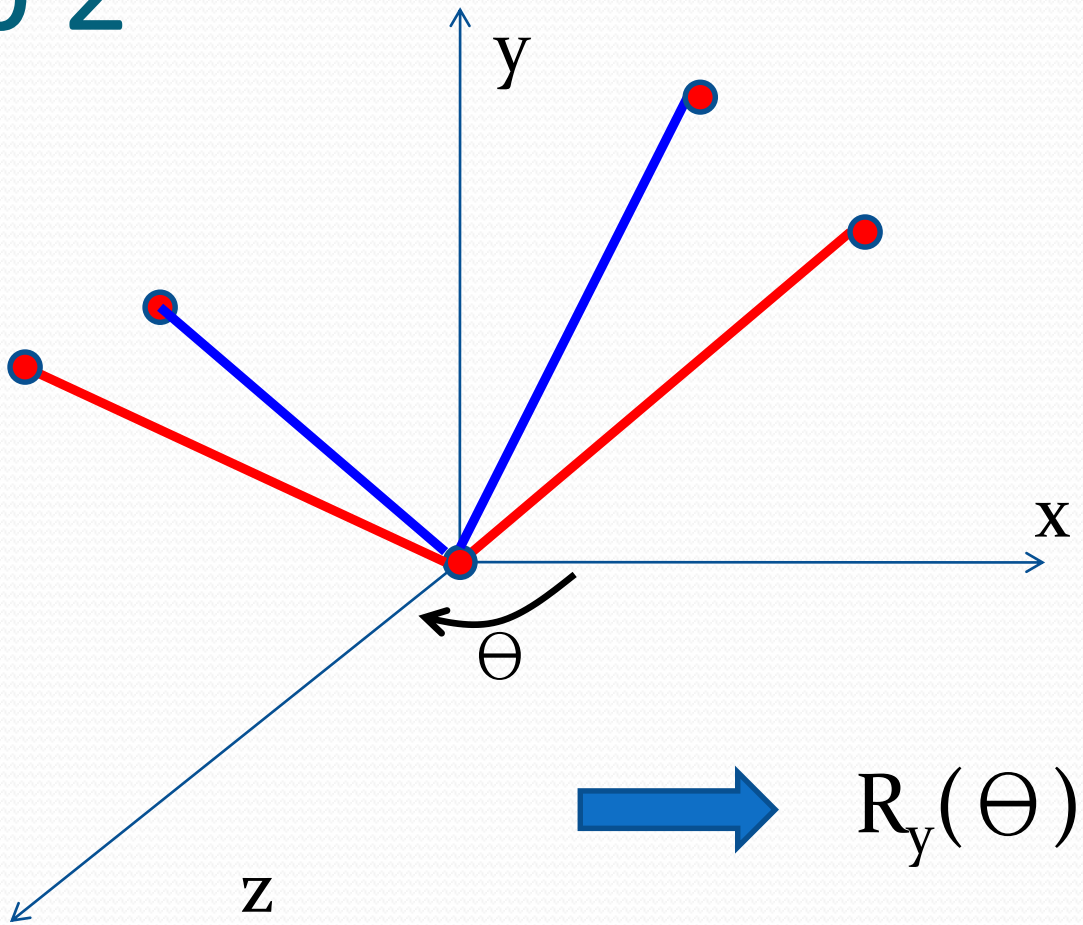


Step 1



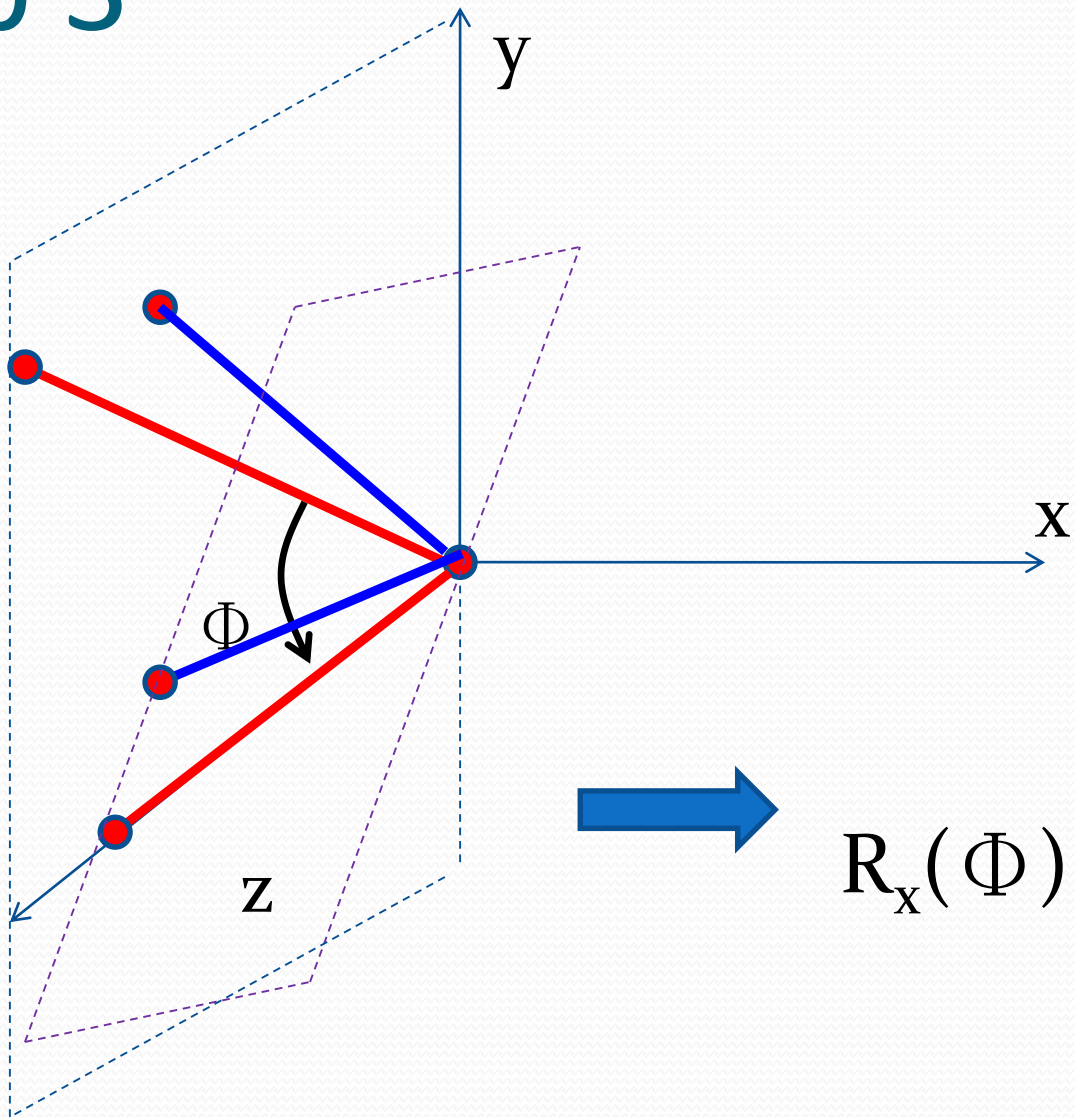
$T(-x_1, -y_1, -z_1)$

Step 2



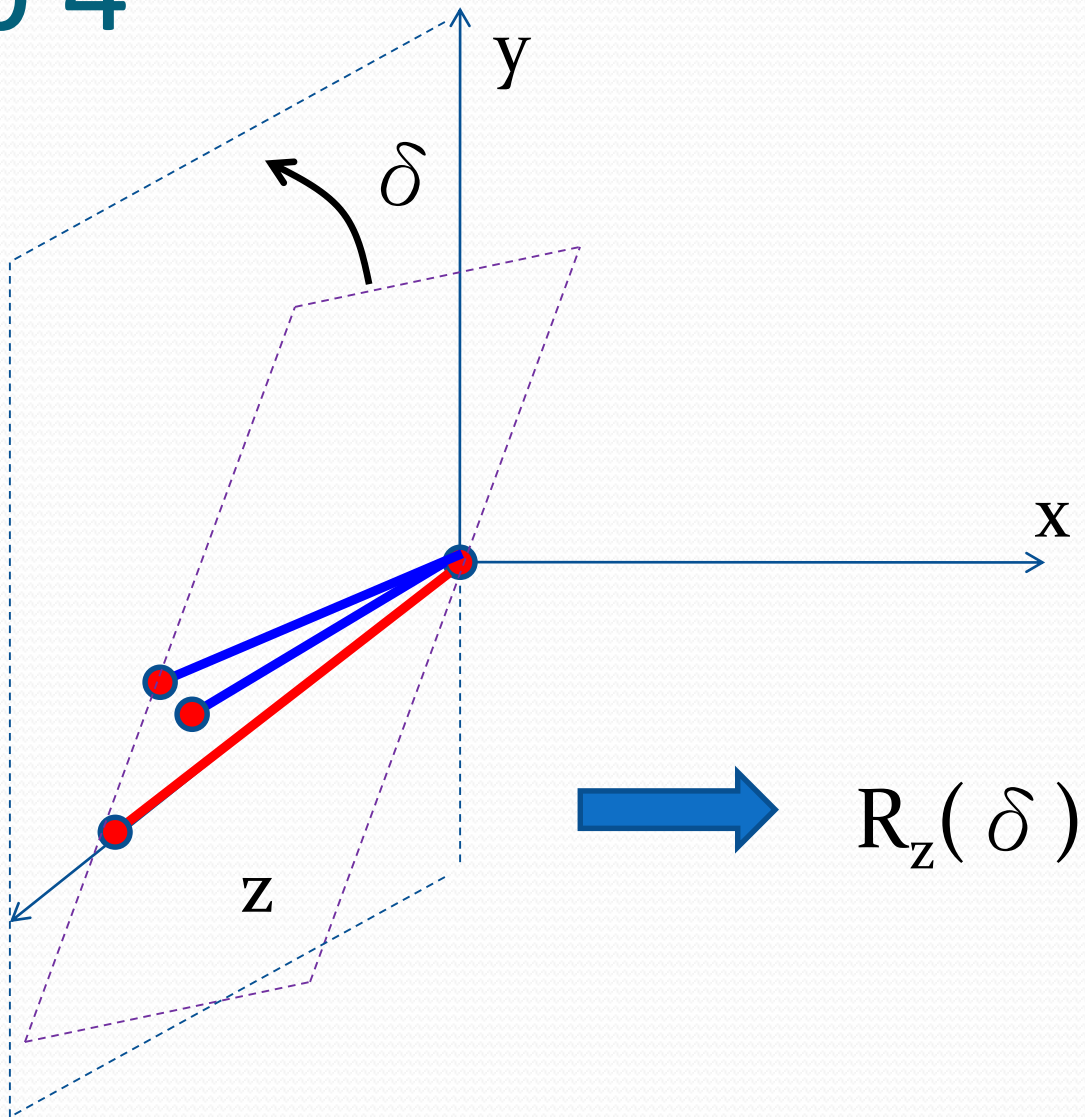
$R_y(\Theta)$

Step 3



$$R_x(\Phi)$$

Step 4



Summary

$$M = R_z(\delta) * R_x(\Phi) * R_y(\Theta) * T(-x_1, -y_1, -z_1)$$

Rotate vector

Align vector with z axis

Align vector in y-z plan

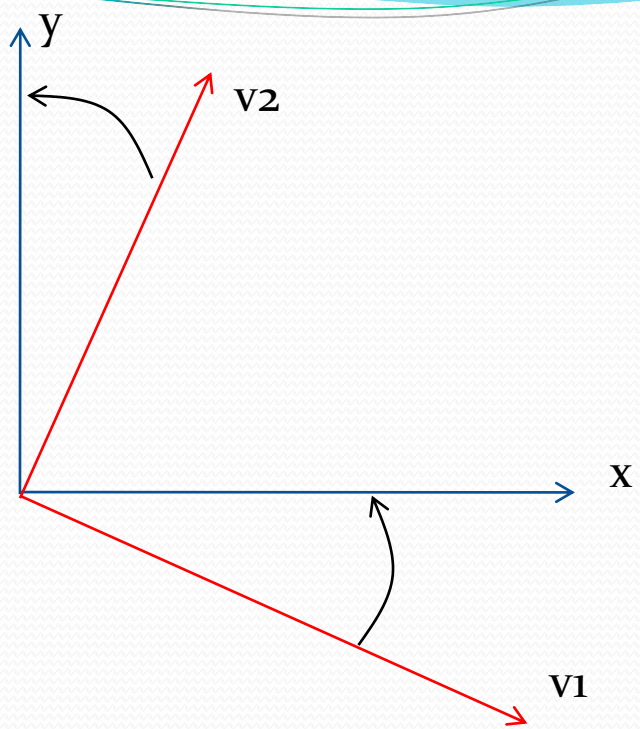
Bring one end of the vector to the origin

Go back

$$M = T(x_1, y_1, z_1)\,R_y(-\Theta)\,R_x(-\Phi)\,R_z(-\delta)\,R_z(\delta)\,R_x(\Phi)\,R_y(\Theta)\,T(-x_1, -y_1, -z_1)$$

Not an easy work !!

Review



Use transform matrix

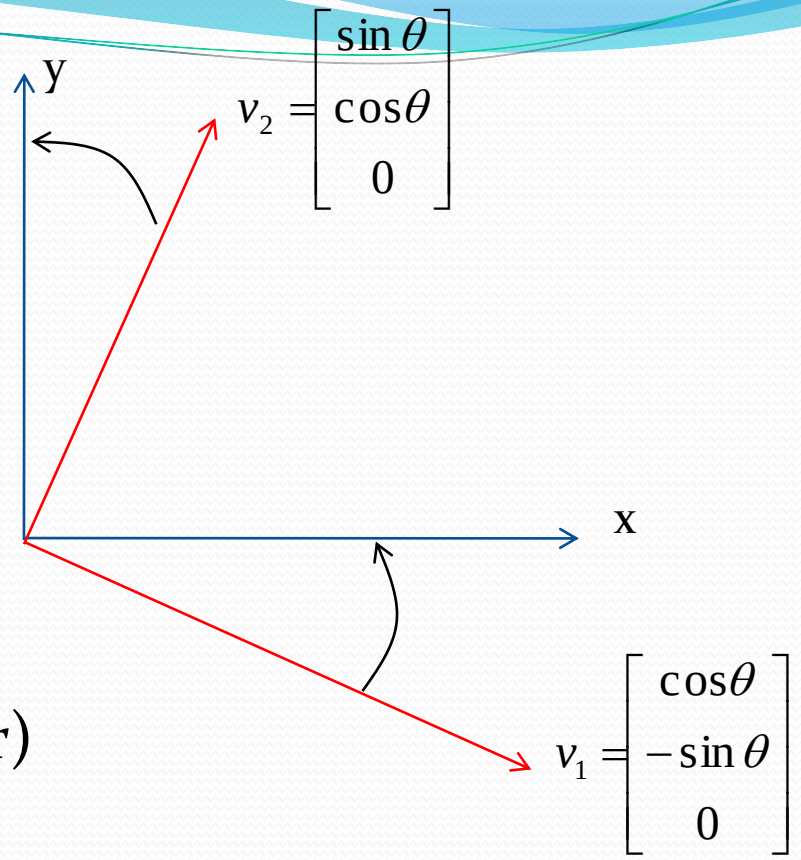
$$\begin{bmatrix} V1x & V1y & 0 \\ V2x & V2y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$|V1| = 1$ (unit vector)

$|V2| = 1$ (unit vector)

$V1 \bullet V2 = 0$ (Perpendicular)

Review



Check:

$|V1| = 1$ (unit vector)

$|V2| = 1$ (unit vector)

$V1 \bullet V2 = 0$ (Perpendicular)

Transform matrix

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

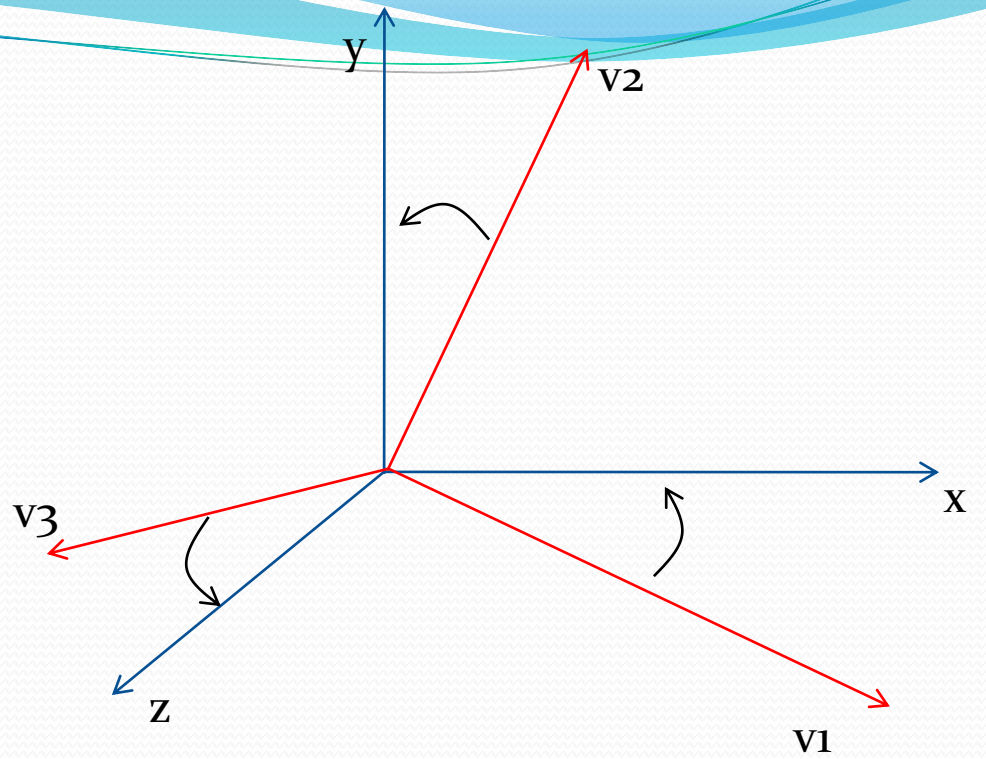
Review

Check:

$$T * v_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta \\ -\sin\theta \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$T * v_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\theta \\ \cos\theta \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Review



General rotation matrix

$$GRM = \begin{bmatrix} v1_x & v1_y & v1_z & 0 \\ v2_x & v2_y & v2_z & 0 \\ v3_x & v3_y & v3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

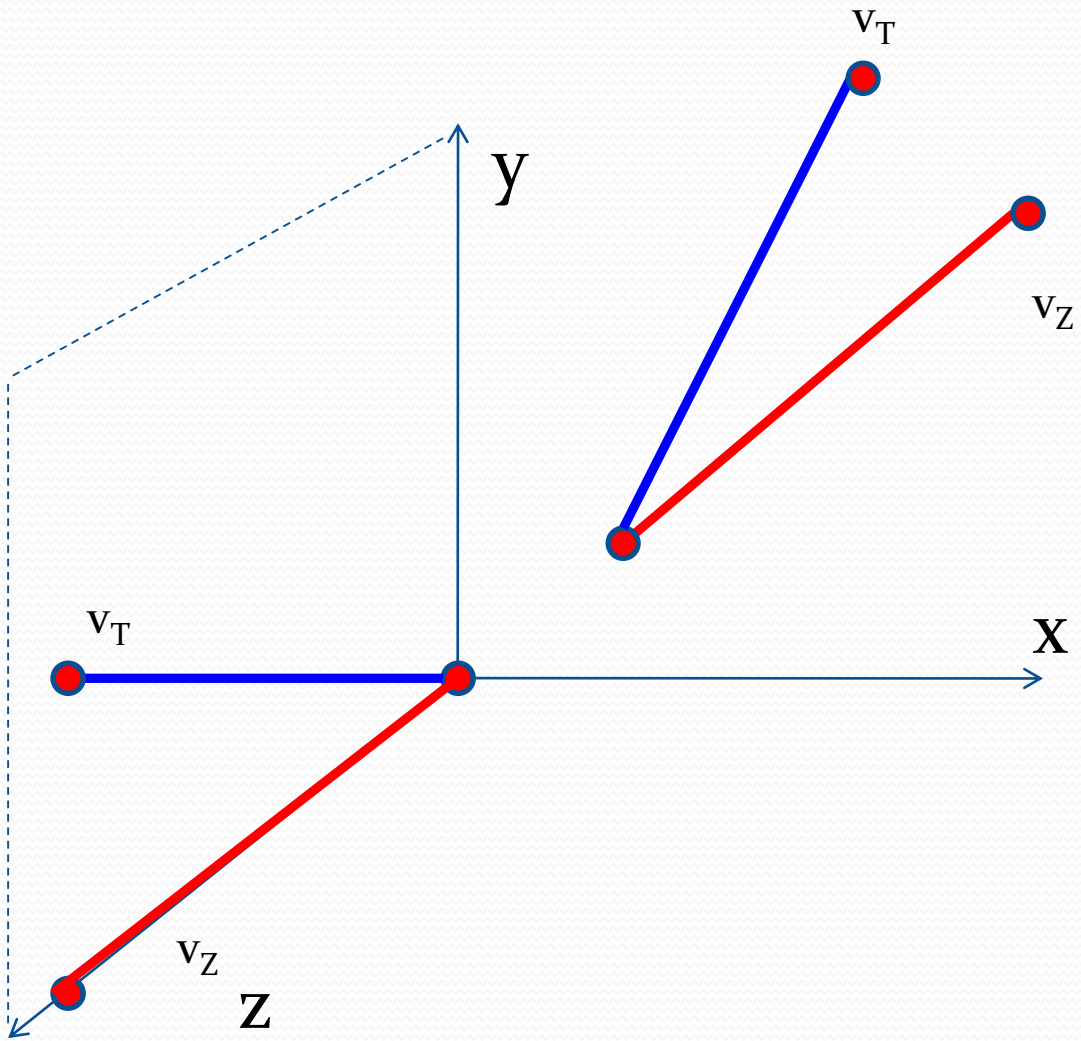
$|V1| = 1,\ |V2| = 1,\ |V3| = 1$

$V1 \bullet V2 = 0$

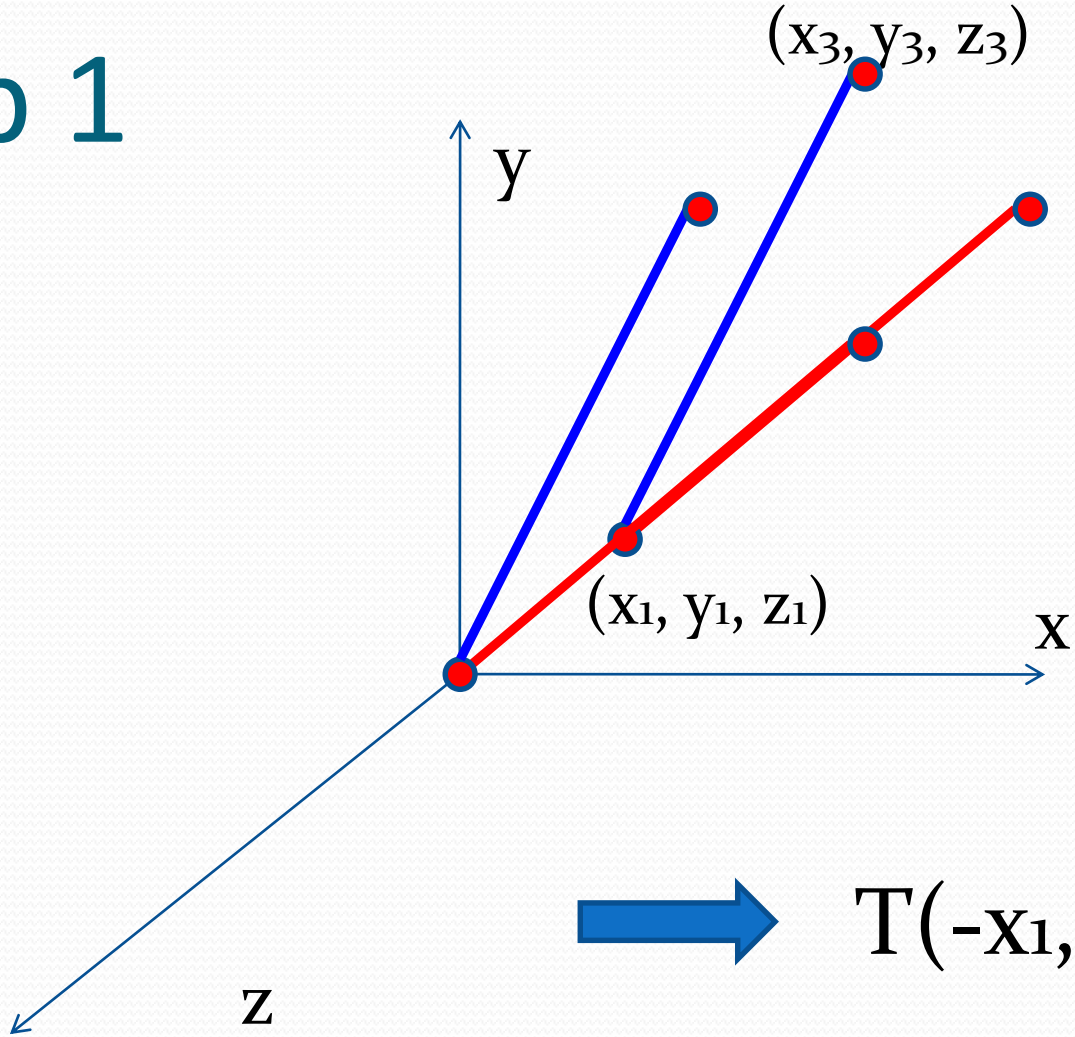
$V2 \bullet V3 = 0$

$V1 \bullet V3 = 0$

Re-visit Problem 2

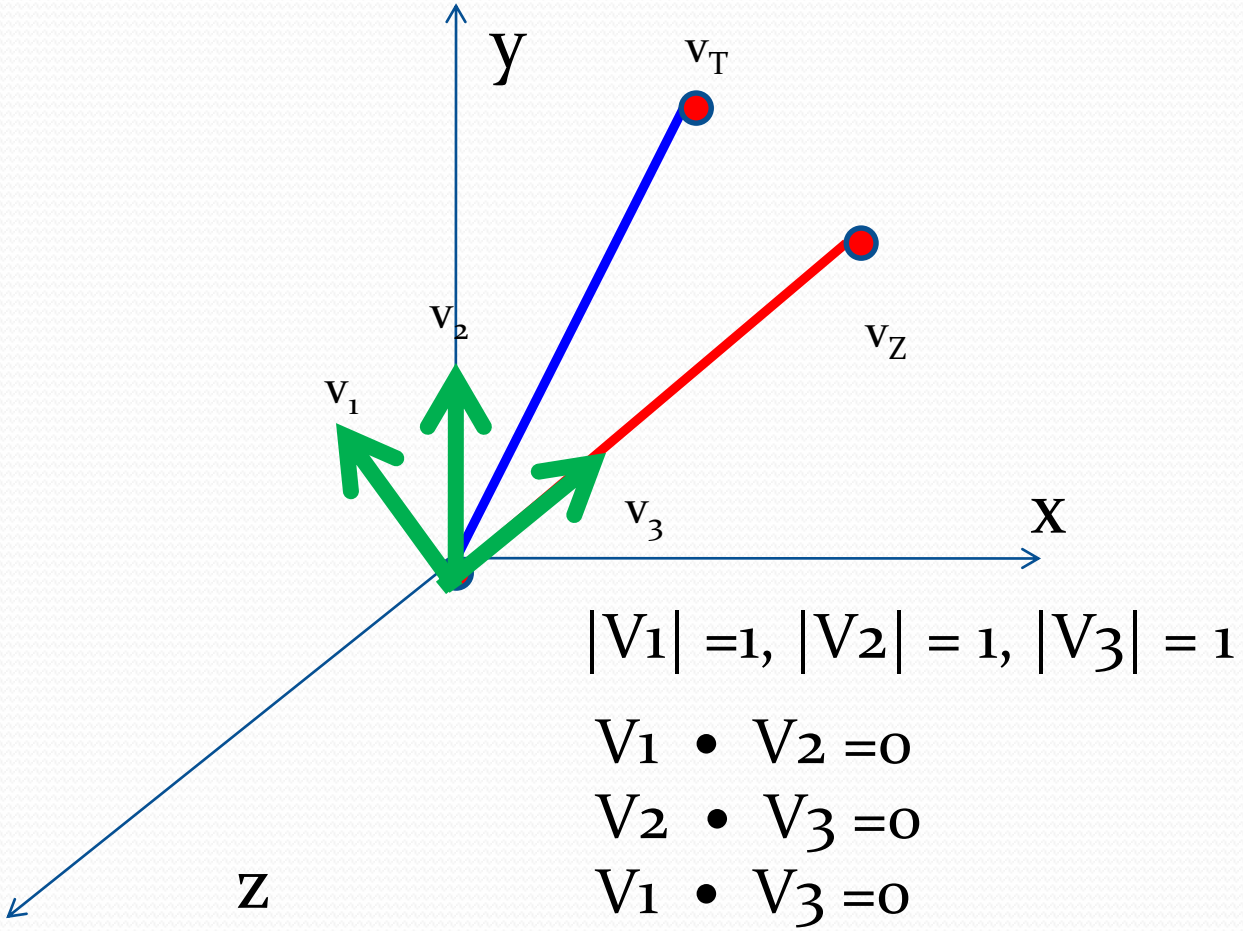


Step 1



$T(-x_1, -y_1, -z_1)$

Generate GRM



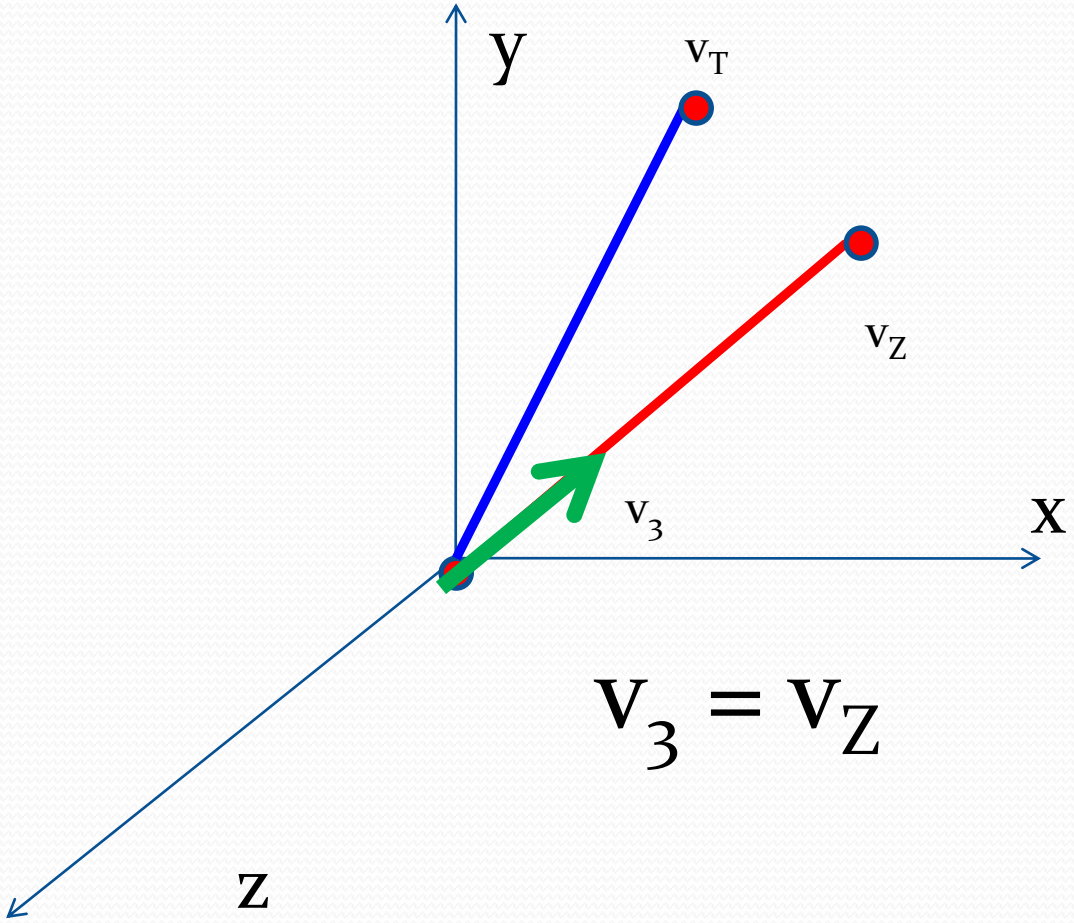
$|V1| = 1, |V2| = 1, |V3| = 1$

$V1 \bullet V2 = 0$

$V2 \bullet V3 = 0$

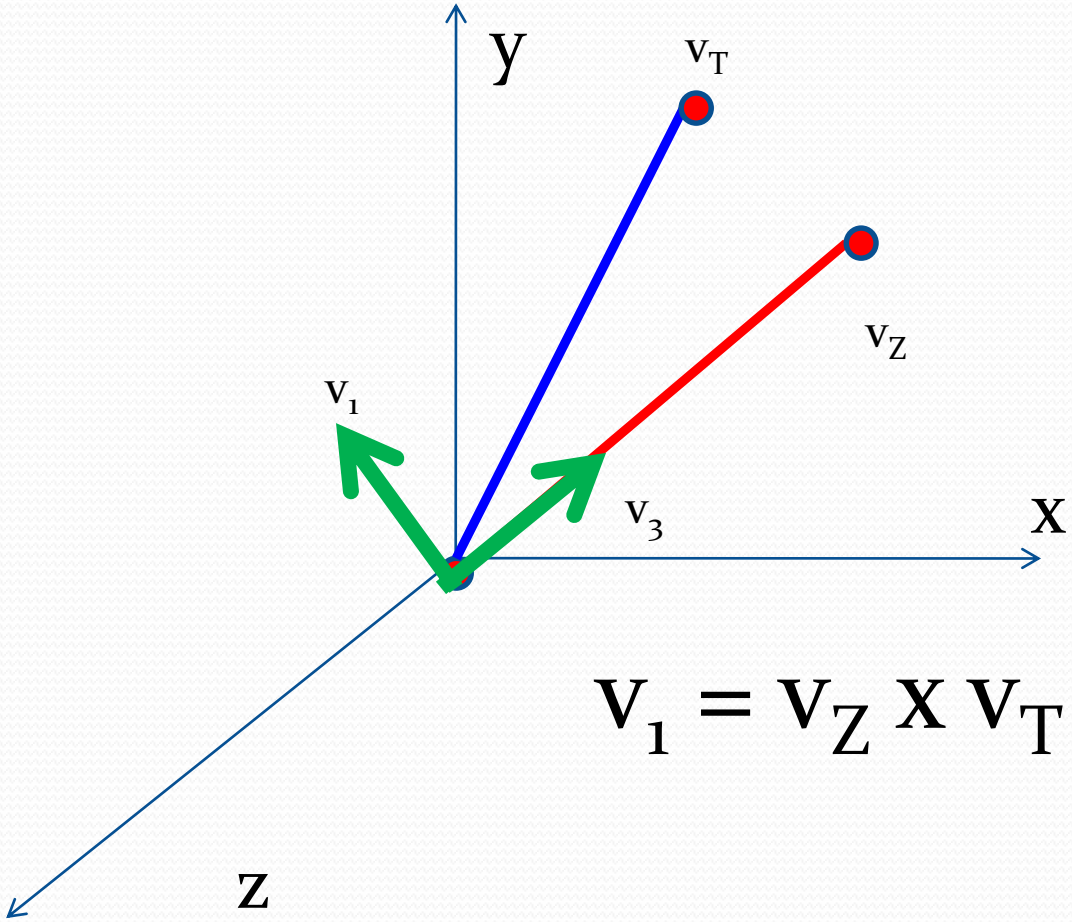
$V1 \bullet V3 = 0$

Step 1



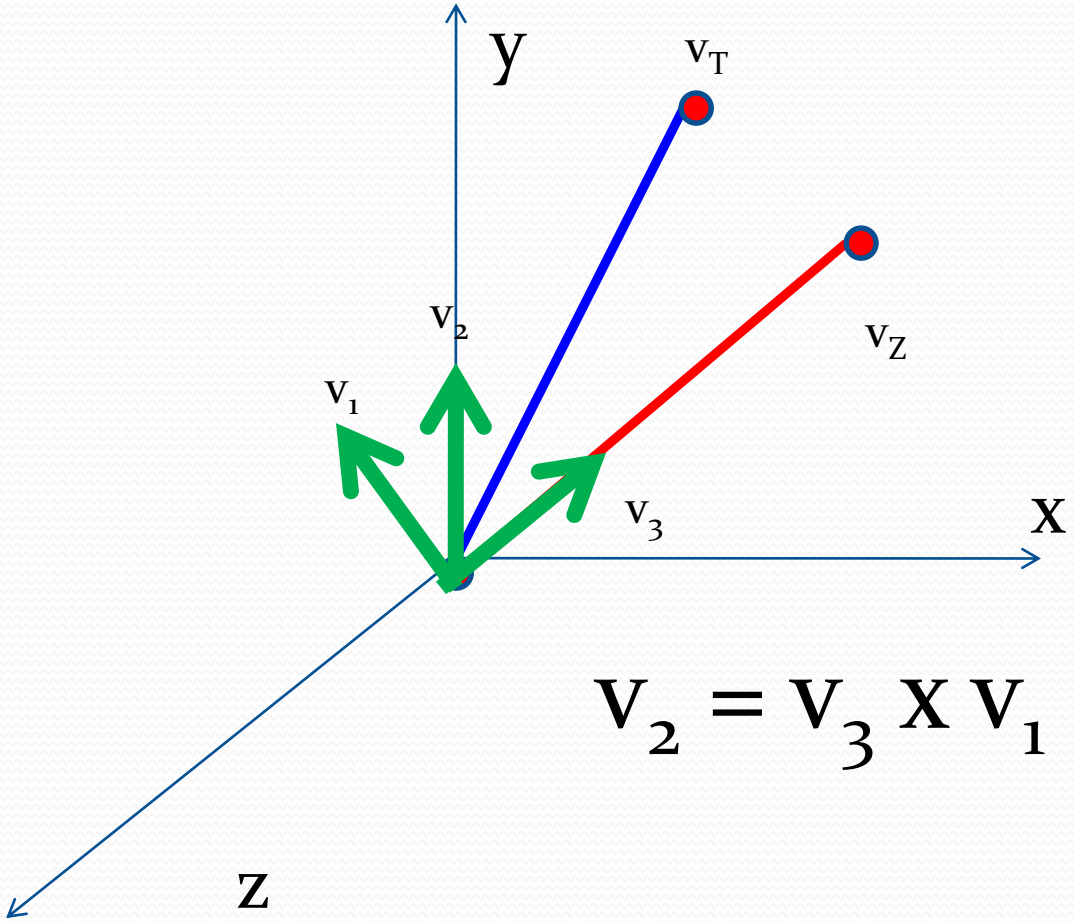
$$v_3 = v_Z$$

Step 2



$$v_1 = v_Z \times v_T$$

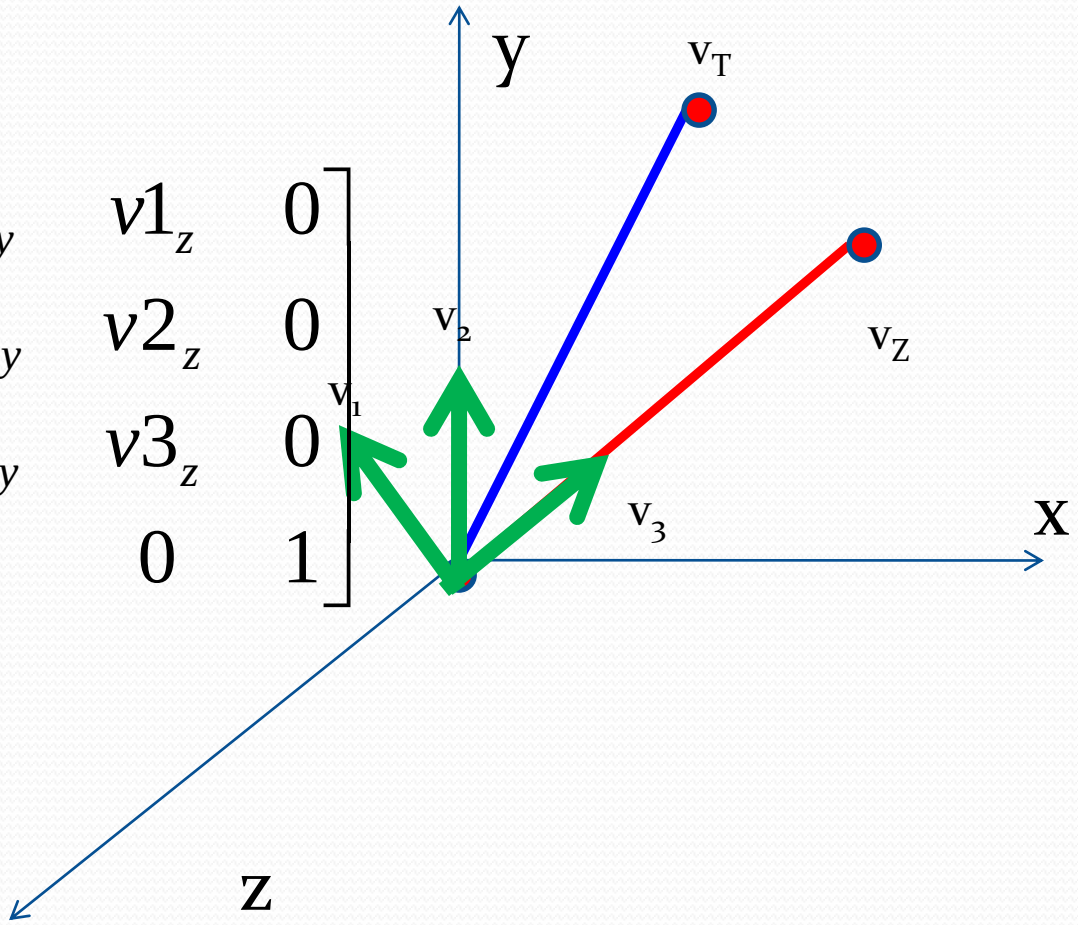
Step 3



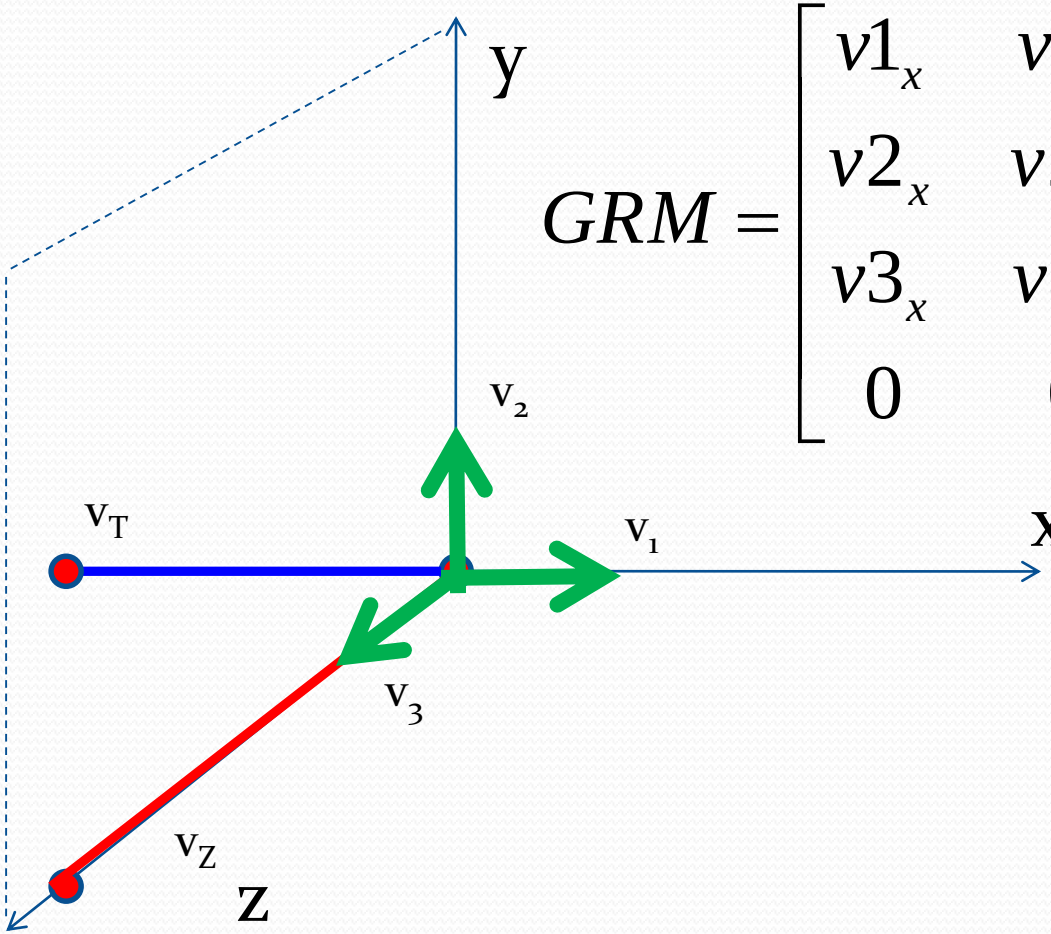
$$v_2 = v_3 \times v_1$$

Before transformation

$$GRM = \begin{bmatrix} v1_x & v1_y & v1_z & 0 \\ v2_x & v2_y & v2_z & 0 \\ v3_x & v3_y & v3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

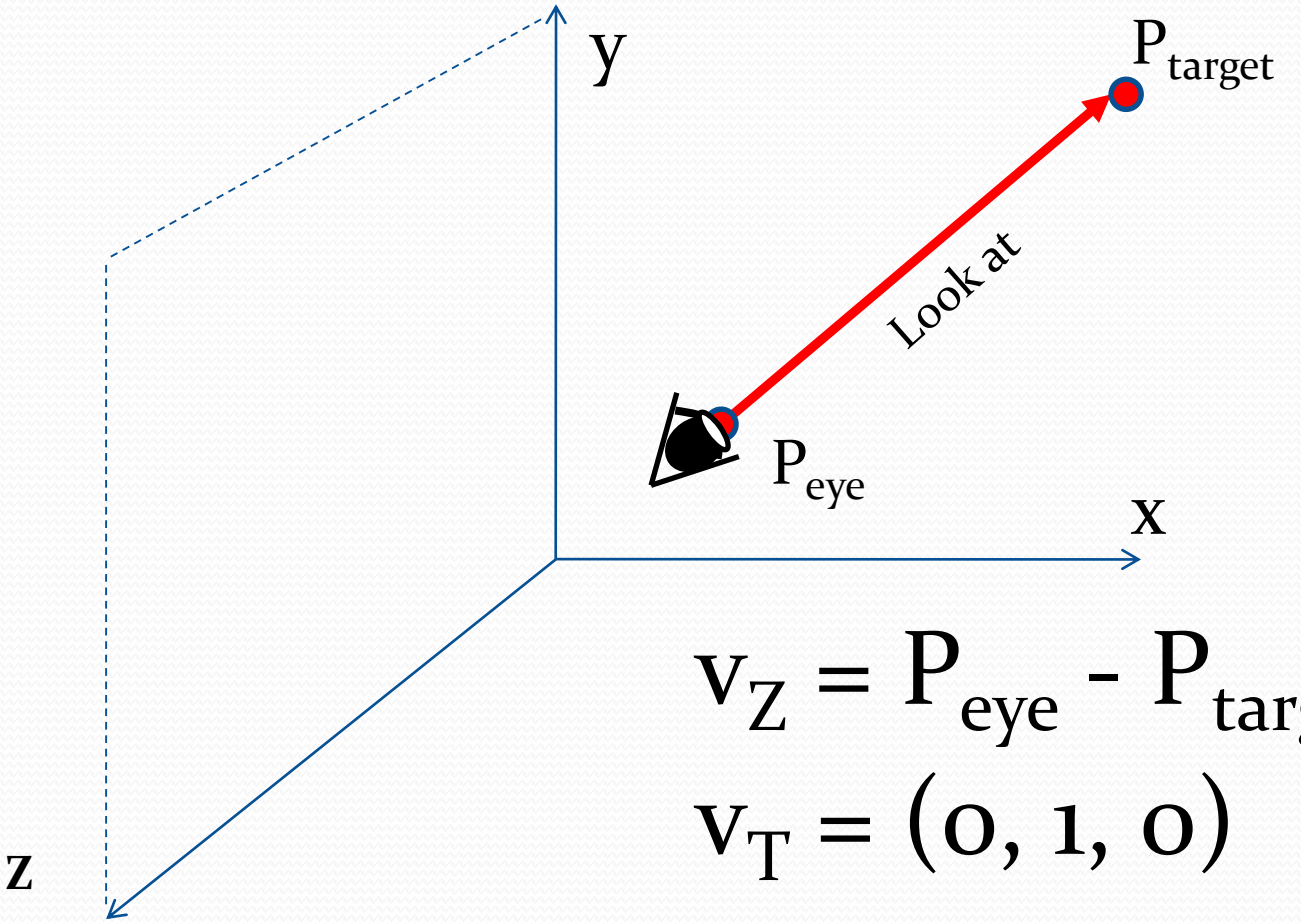


After transformation



$$GRM = \begin{bmatrix} v1_x & v1_y & v1_z & 0 \\ v2_x & v2_y & v2_z & 0 \\ v3_x & v3_y & v3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Simulate Eye



$$v_Z = P_{eye} - P_{target}$$

$$v_T = (0, 1, 0)$$

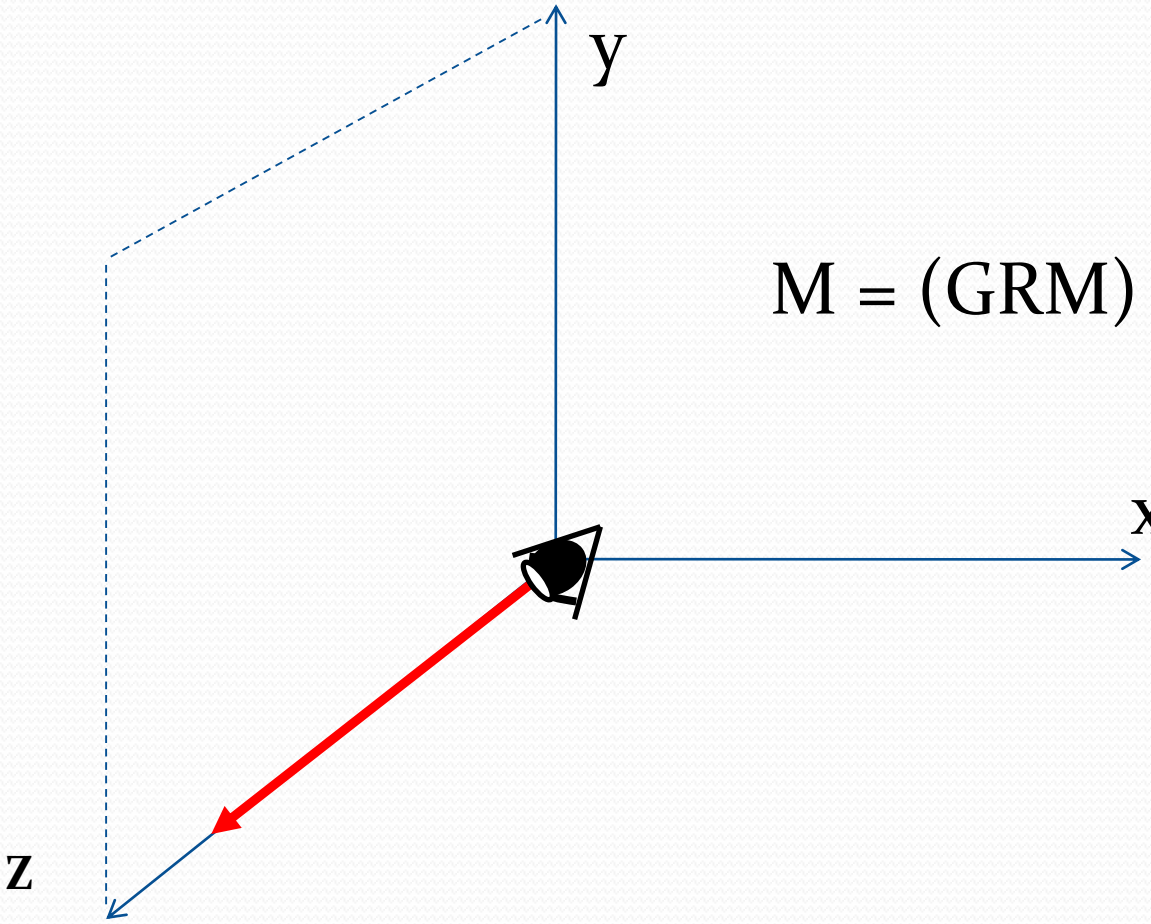
Simulate Eye

- Calculate V1, V2, V3

- Calculate GRM

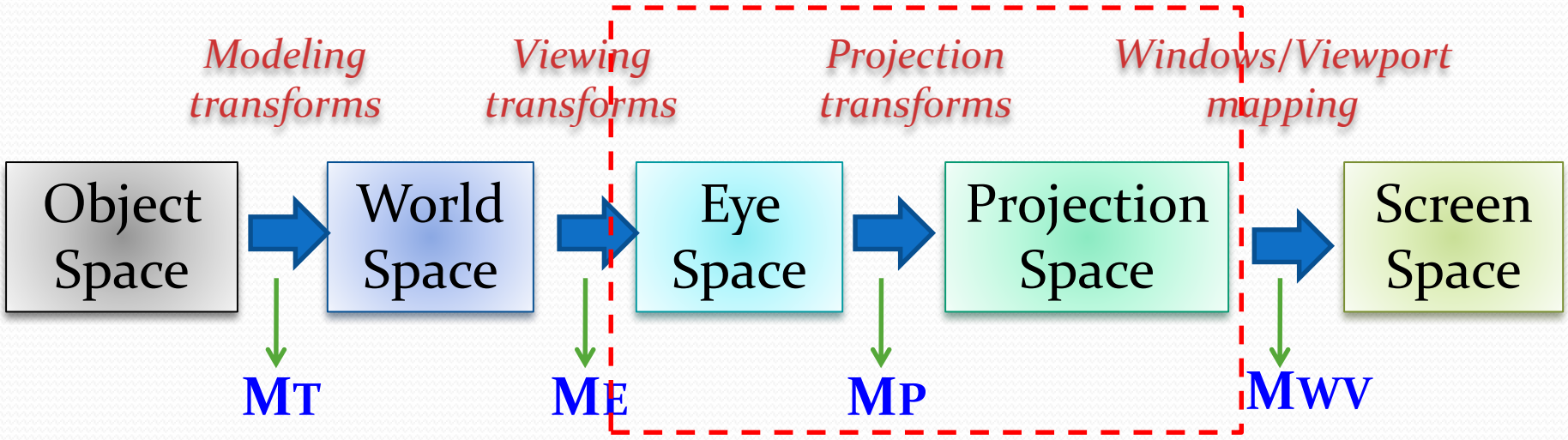
$$GRM = \begin{bmatrix} v1_x & v1_y & v1_z & 0 \\ v2_x & v2_y & v2_z & 0 \\ v3_x & v3_y & v3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Simulate Eye

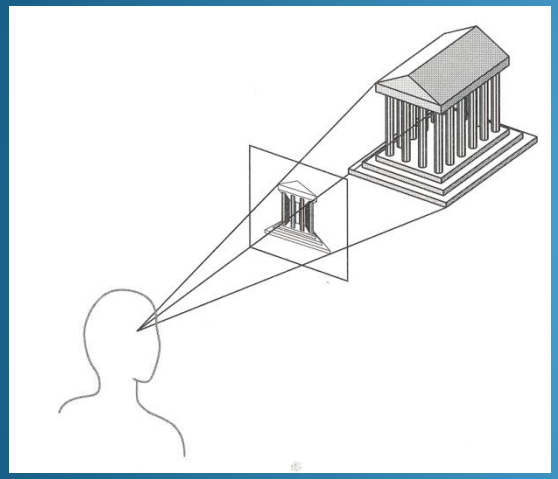


$$M = (GRM)\ T(-P_{eye})$$

3D Rendering Pipeline



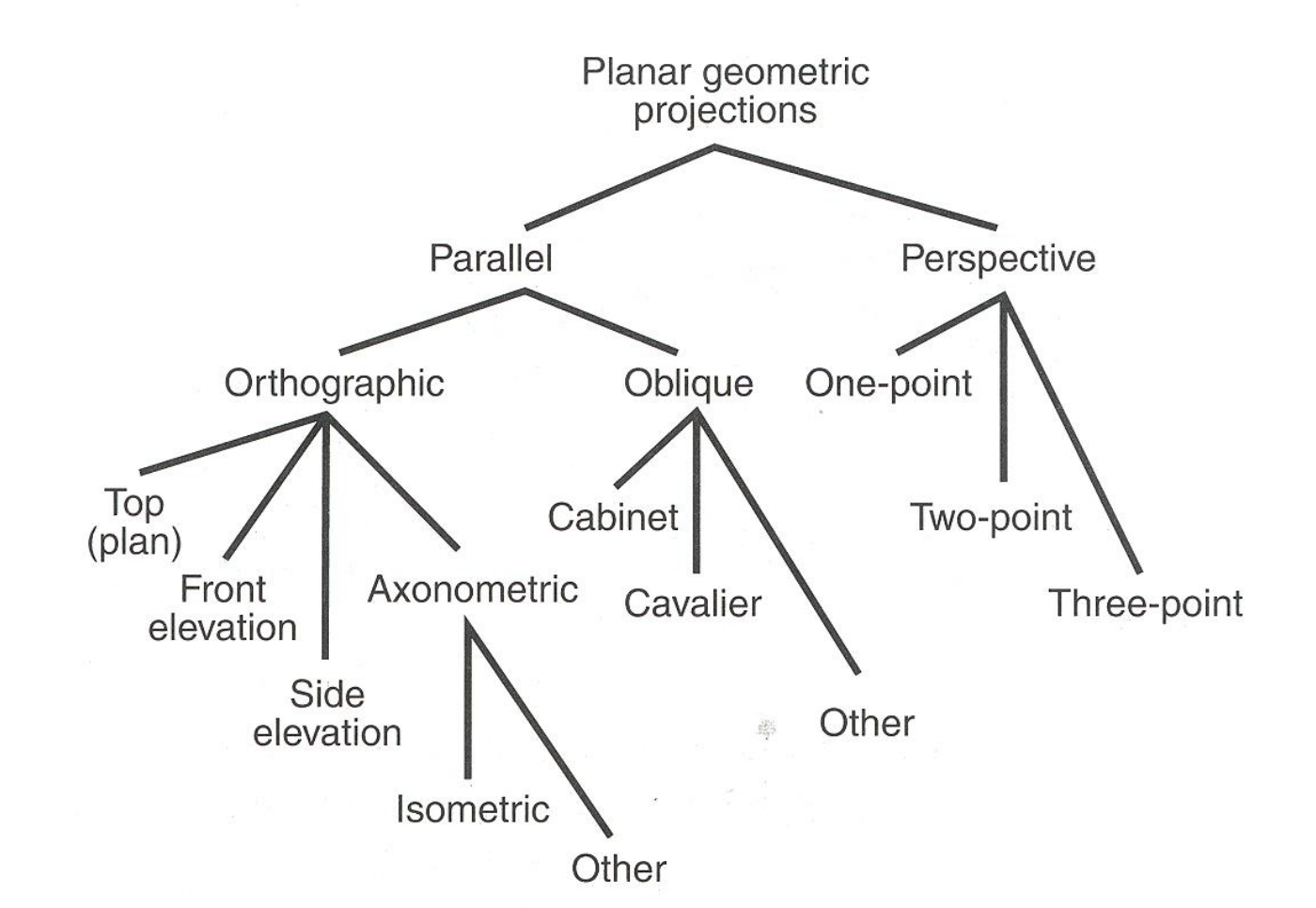
Projections



Angel: Interactive Computer Graphics

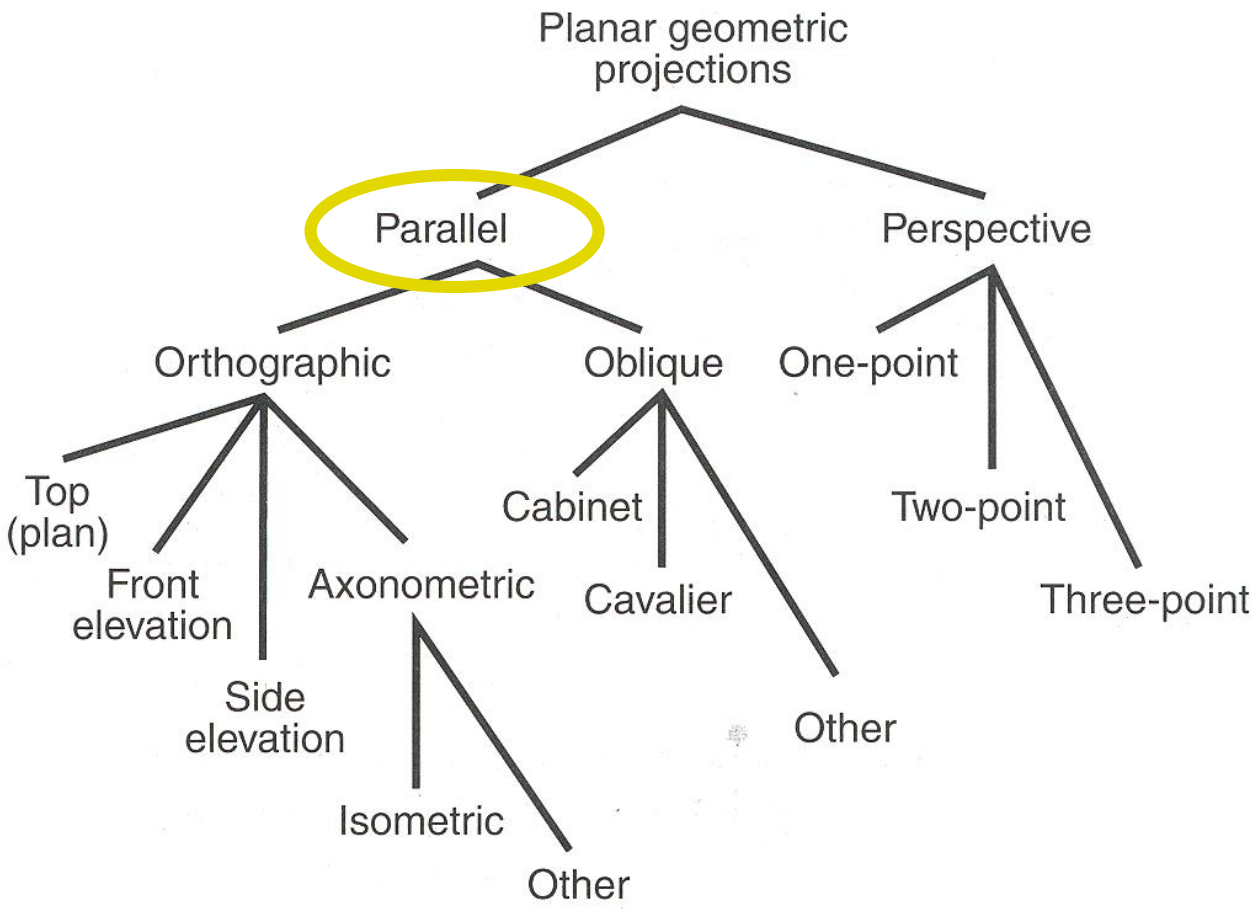
Road in perspective

Taxonomy of Projections



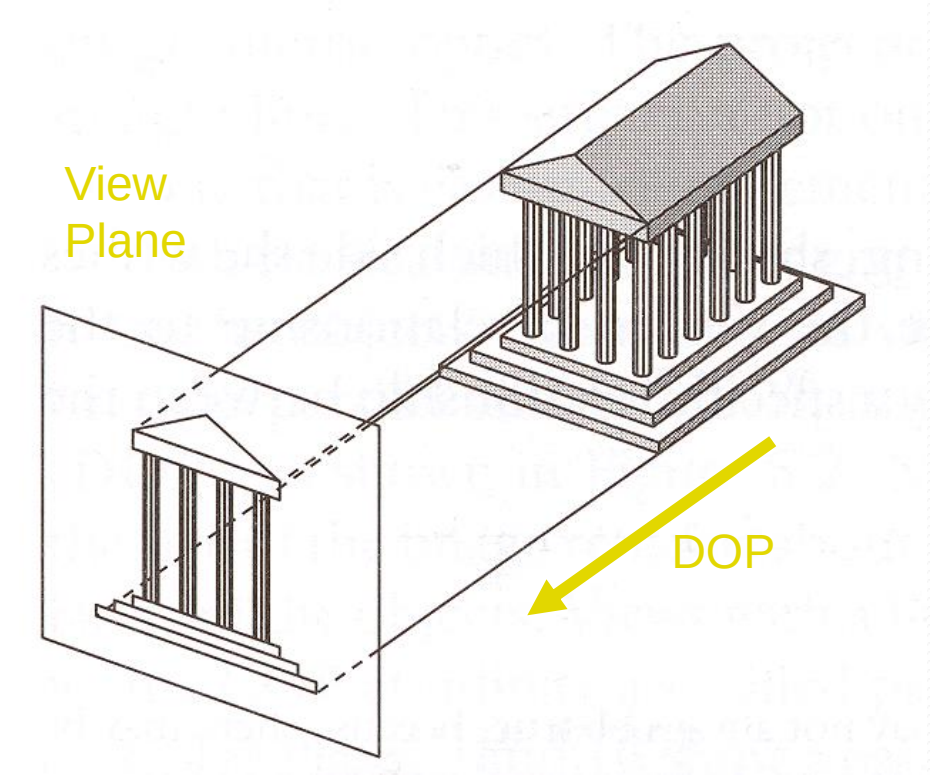
FVFHP Figure 6.10

Taxonomy of Projections



Parallel Projection

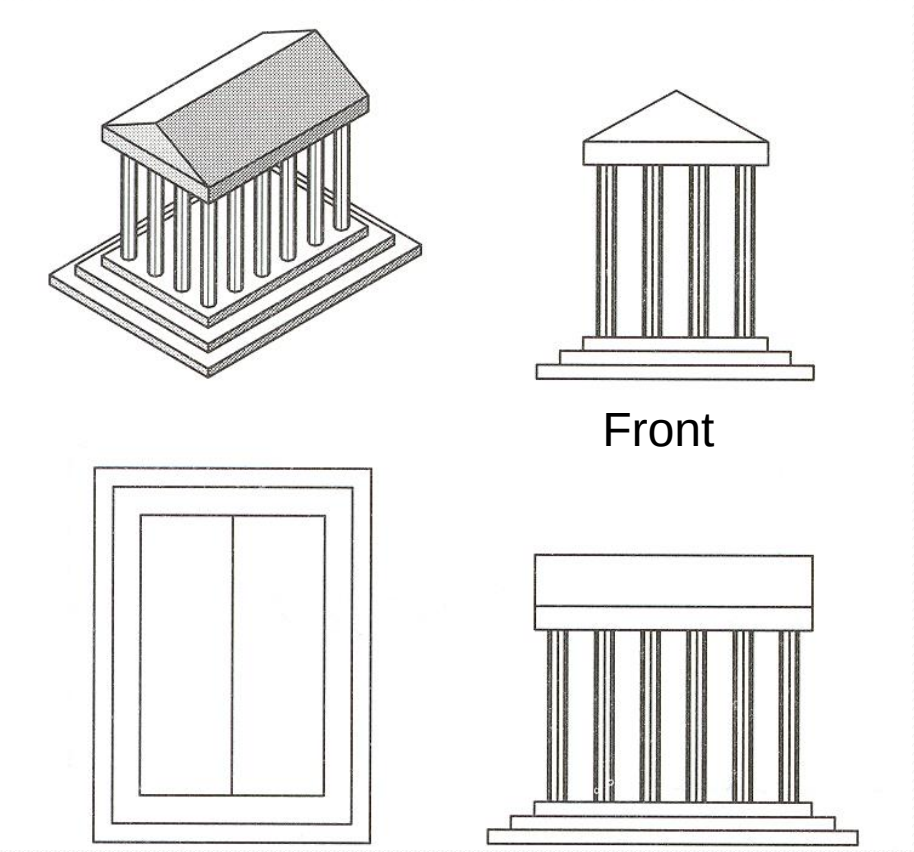
- Center of projection is at infinity
  - Direction of projection (DOP) same for all points



Angel Figure 5.4

Orthographic Projections

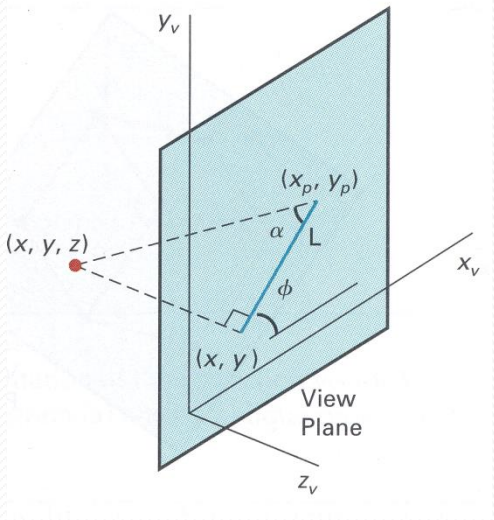
- DOP perpendicular to view plane

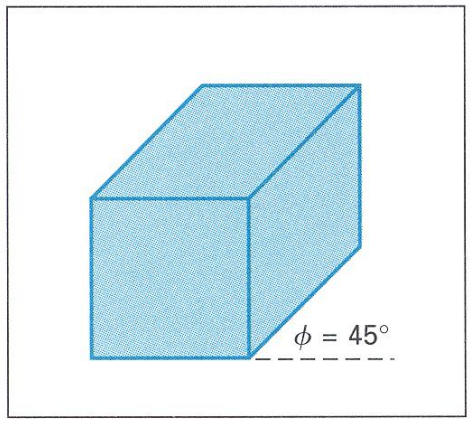


Angel Figure 5.5

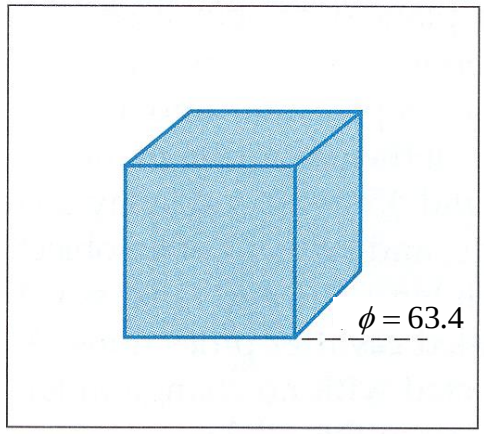
Oblique Projections

- DOP not perpendicular to view plane





Cavalier
(DOP $\alpha = 45^o$)
$\tan(\alpha) = 1$



Cabinet
(DOP $\alpha = 63.4^o$)
$\tan(\alpha) = 2$

H&B

Orthographic Projection

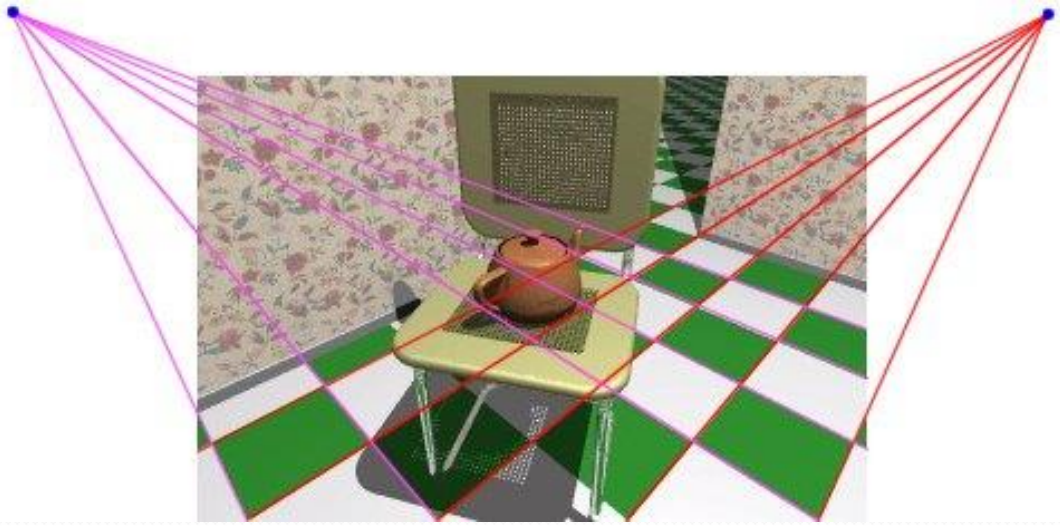
- Simple Orthographic Transformation

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

- Original world units are preserved
  - Pixel units are preferred

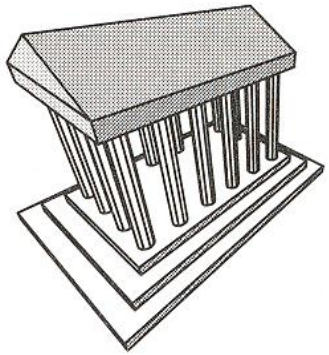
Perspective Transformation

- First discovered by Donatello, Brunelleschi, and DaVinci during Renaissance
- Objects closer to viewer look larger
- Parallel lines appear to converge to single point

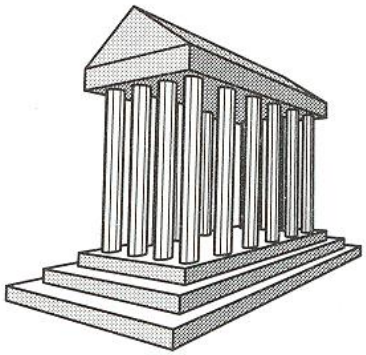


Perspective Projection

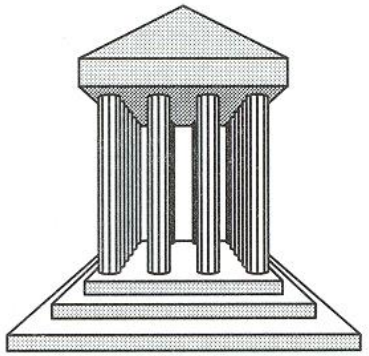
- How many vanishing points?



3-Point Perspective



2-Point Perspective

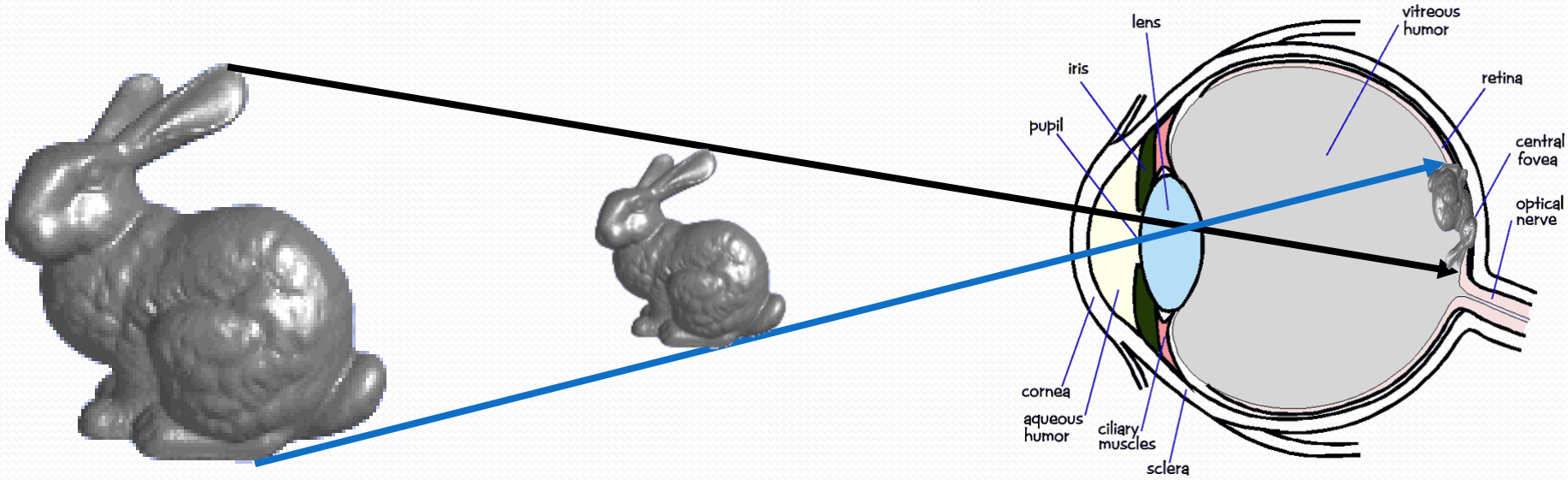


1-Point Perspective

Angel Figure 5.10

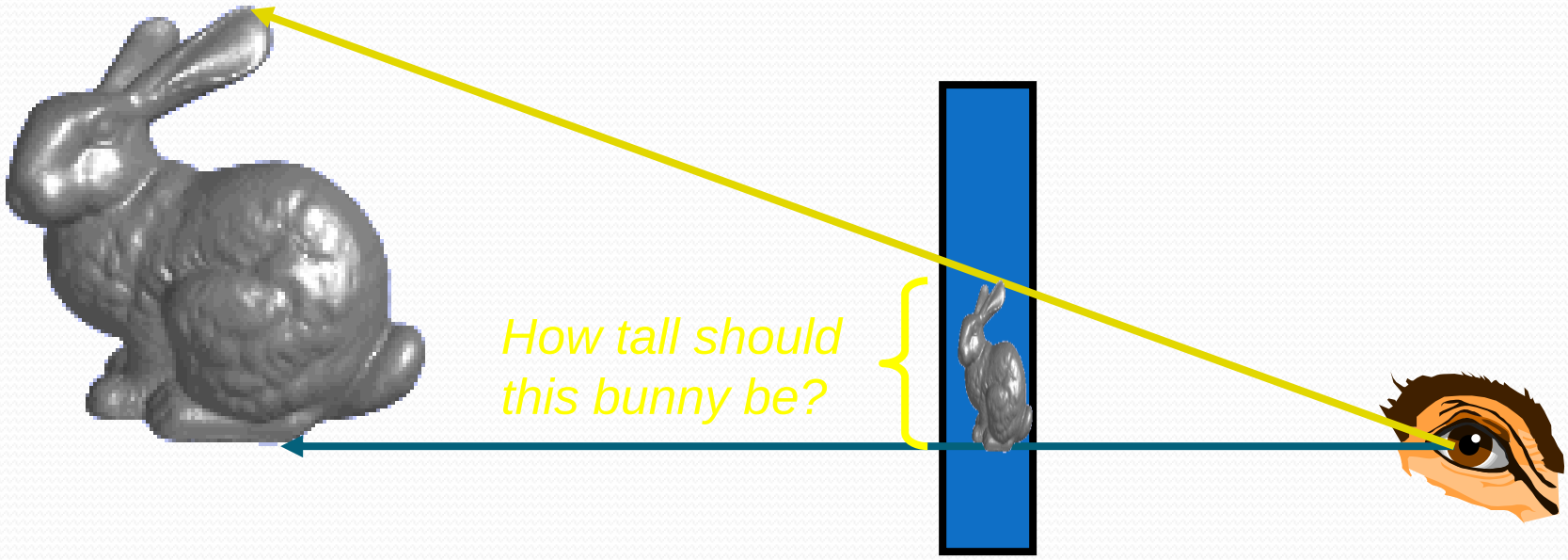
Perspective Projection

- In the real world, objects exhibit
  : distant objects appear smaller
- The basic situation:



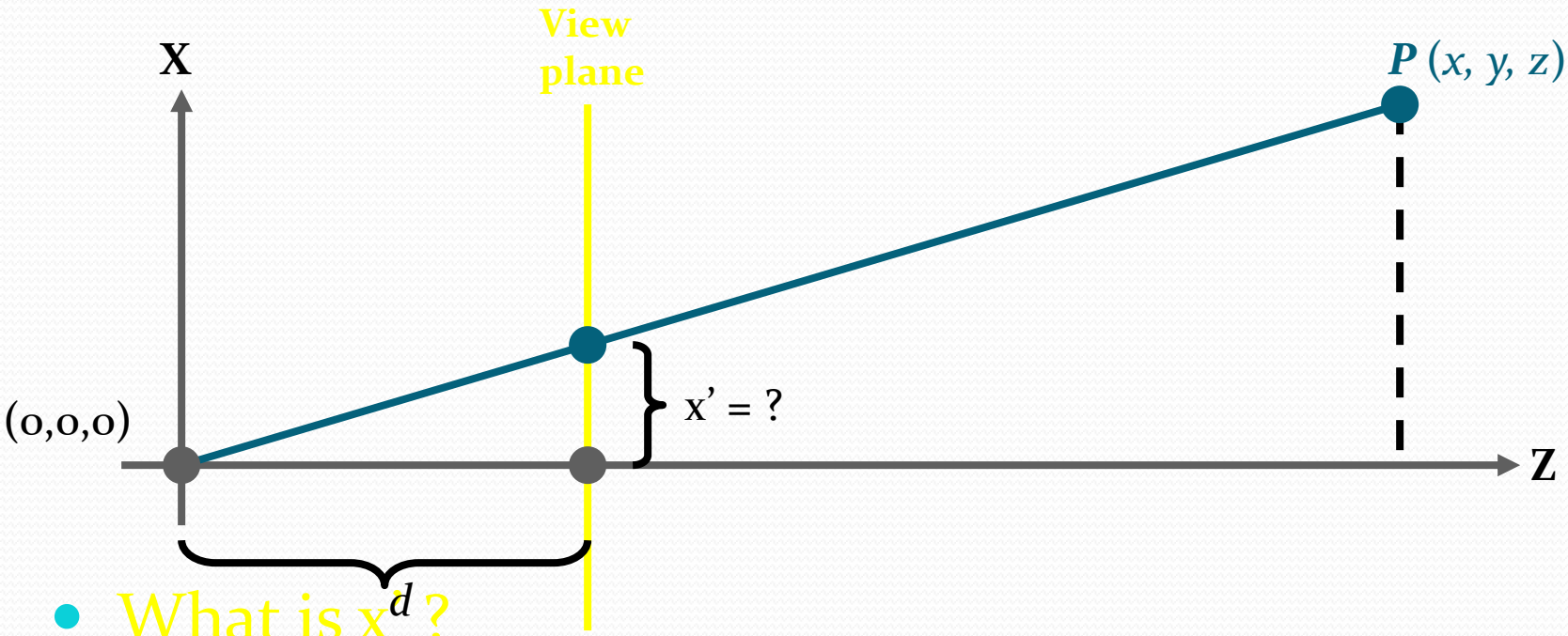
Perspective Projection

- When we do 3-D graphics, we think of the screen as a 2-D window onto the 3-D world:



Perspective Projection

- The geometry of the situation is that of similar triangles. View from above:



- What is x'?

Perspective Projection

- Desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{d \cdot x}{z} = \frac{x}{z/d}, \quad y' = \frac{d \cdot y}{z} = \frac{y}{z/d}, \quad z = d$$

- What could a matrix look like to do this?

A Perspective Projection Matrix

- Answer:

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

A Perspective Projection Matrix

- Example:

$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Or, in 3-D coordinates:

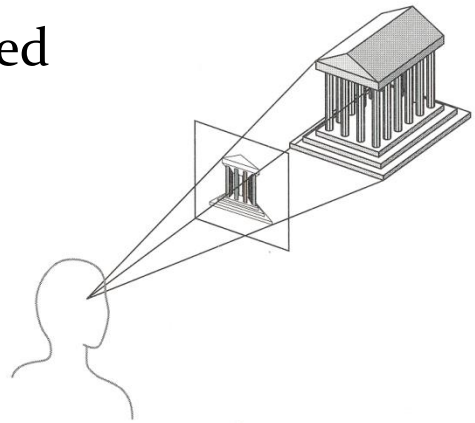
$$\left(\frac{x}{z/d}, \quad \frac{y}{z/d}, \quad d \right)$$

Projection Matrices

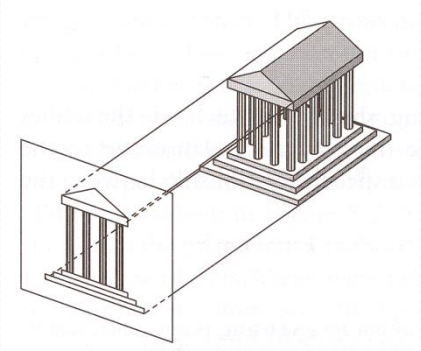
- Now that we can express perspective foreshortening as a matrix, we can compose it onto our other matrices with the usual matrix multiplication
- End result: a single matrix encapsulating modeling, viewing, and projection transforms

Perspective vs. Parallel

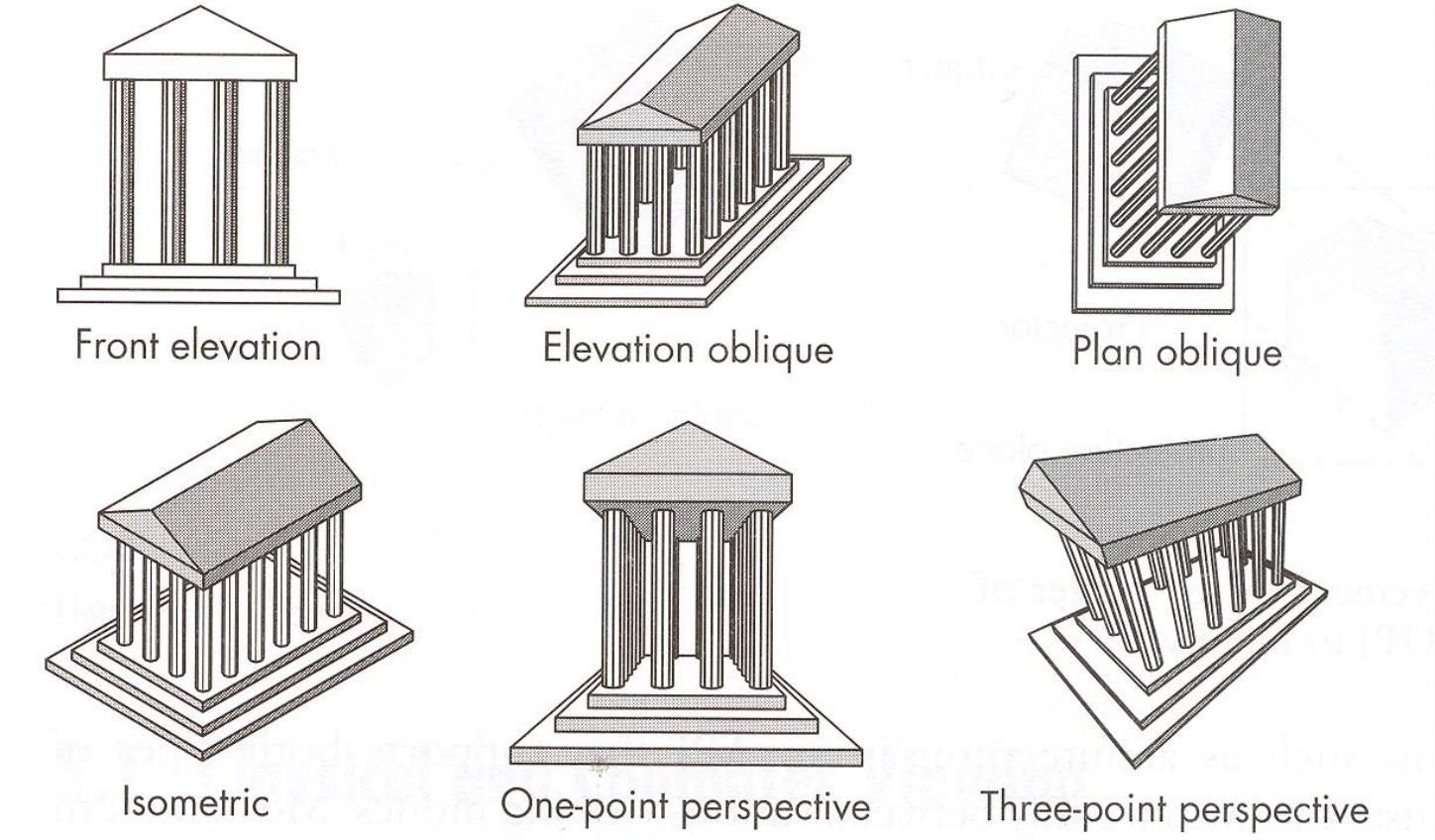
- Perspective projection
  - \+ Size varies inversely with distance - looks realistic
  - – Distance and angles are not (in general) preserved
  - – Parallel lines do not (in general) remain parallel



- Parallel projection
  - \+ Good for exact measurements
  - \+ Parallel lines remain parallel
  - – Angles are not (in general) preserved
  - – Less realistic looking



Classical Projections



Angel Figure 5.3

A 3D Scene

- Notice the presence of the camera, the projection plane, and the world coordinate axes

- Viewing transformations define how to acquire the image on the projection plane

3D Rendering Pipeline