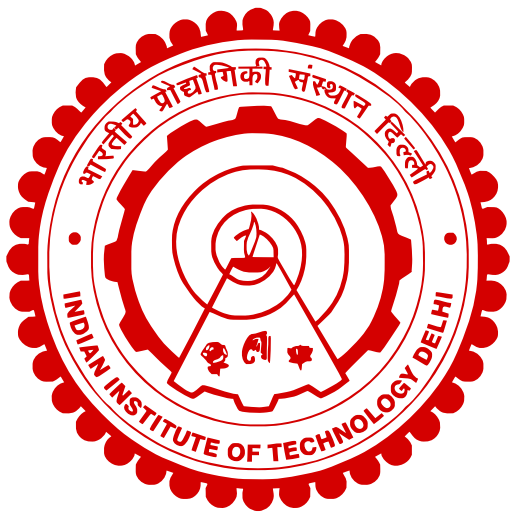


Computer Graphics

Transformations

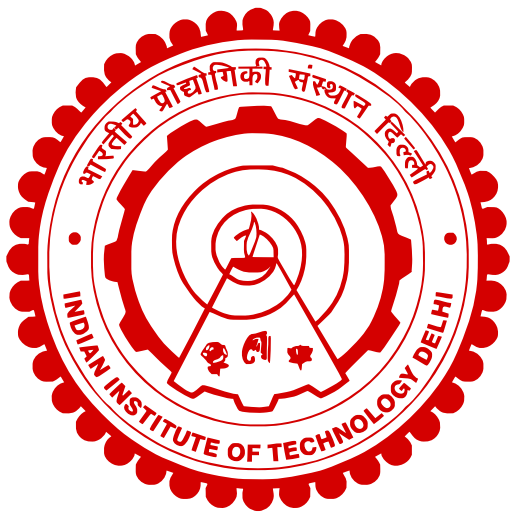
Subodh Kumar, Dept of Computer Sc. & Engg., IIT Delhi



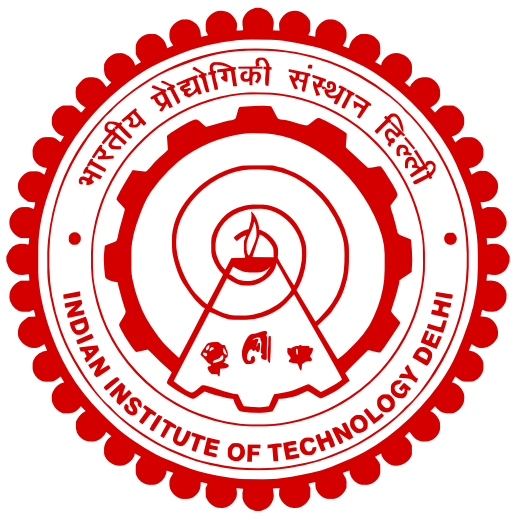
Geometry and Transformations

- Geometry = shape: an (infinite) set of points (satisfying some property)
 - ➔ Simplexes can be tiled to form the geometry
 - Point, Two points (line segment), Three points (triangle), Four point (tetrahedron)
- We differentiate between points and directions (also called vectors)
 - ➔ Vectors can be added, subtracted and scaled.
 - Dot and Cross products are defined on vectors
 - ➔ Point – Point is OK, Point + Vector is OK, Point * scalar is OK
 - ➔ Technically, both form vector spaces
- In some contexts, we may interpret points as 'position vectors'

Coordinate Systems

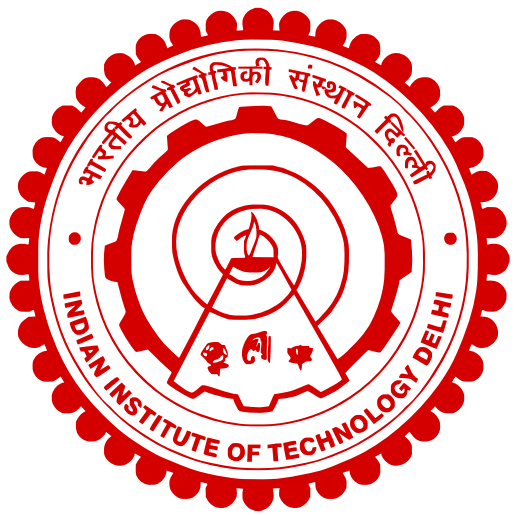


- (Cartesian) Coordinate geometry requires a choice of basis
 - ➔ Model coordinates: where models are defined
 - ➔ World coordinates: where camera is defined
 - ➔ Viewing coordinates: camera's coordinate system
 - ➔ Clip coordinates: homogeneous coordinates
 - ➔ Image coordinates: 2D, normalized image plane
 - ➔ Pixel coordinates: 2D pixels (integer)
- Orthonormal basis, unit length (for scale) and mutually perpendicular



Transformations

- Linear, Rigid, and affine transformations of points and vectors is related to
 - ➔ matrix multiplication
 - Reposition the object with a given coordinate system
- =
- Find the coordinates of a point in a new coordinate system (change of basis)
 - Transform all points of an object
 - Attributes as well
 - Compose transformations



Simple Transformation

- Rigid body (e.g., Translate, Rotate)
 - ➔ No length changes
- Linear (e.g., Rotate, Scale, Reflect, Shear)
 - ➔ $aT(P) + bT(Q) = T(aP+bQ)$
 - nxn Matrix-multiplication
- Affine (e.g., Translation, Rotation, Scale)
 - ➔ Linear + constant addition: $aT(P) + bT(Q) = T(aP+bQ) + \underline{K}$
 - ➔ Preserves parallel lines
 - ➔ Linear in projective space $n+1 \times n+1$ matrix multiplication
- Projective (not linear, projection)
 - ➔ Preserves lines

$V' = (1-t) V_1 + t V_2$

$V = uV_0 + vV_1 + wV_2$

$T(V) = uT(V_0) + vT(V_1) + wT(V_2)$

Convention:

Points, Vectors are column matrices

Transform p by $M \equiv M * p$

⚠ OpenGL uses column-major Matrices

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



Matrix Multiplication

- Affine transformation =

$$T(x,y,z) = \mathbf{a}_1x + \mathbf{b}_1y + \mathbf{c}_1z + \mathbf{d}_1$$

$$\mathbf{a}_2x + \mathbf{b}_2y + \mathbf{c}_2z + \mathbf{d}_2$$

$$\mathbf{a}_3x + \mathbf{b}_3y + \mathbf{c}_3z + \mathbf{d}_3$$

- Compose:

➔ Matrix multiplication is associative

➔ M_1 followed by M_2 , then M_3 : $M_3(M_2(M_1 V^T)) = [M_3 M_2 M_1] V^T = [M] V^T$

➔ Remember: Matrix Multiplication is not commutative (in general)

$$\begin{bmatrix} \bullet \\ \bullet \\ \circ \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

vertex Matrix Homogeneous Coordinate



Matrix Multiplication

- Affine transformation =

$$T(x,y,z) = \mathbf{a}_1x + \mathbf{b}_1y + \mathbf{c}_1z + \mathbf{d}_1$$

$$\mathbf{a}_2x + \mathbf{b}_2y + \mathbf{c}_2z + \mathbf{d}_2$$

$$\mathbf{a}_3x + \mathbf{b}_3y + \mathbf{c}_3z + \mathbf{d}_3$$

$$\begin{bmatrix} \bullet \\ \bullet \\ \circ \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

vertex Matrix Homogeneous Coordinate

- Compose:

- ➔ Matrix multiplication is associative (Or is it M_3 followed by M_2 etc.?)
- ➔ M_1 followed by M_2 , then M_3 : $M_3(M_2(M_1 V^T)) = [M_3 M_2 M_1] V^T = [M] V^T$
- ➔ Remember: Matrix Multiplication is not commutative (in general)



Affine Transformation

Linear Transformation using Homogeneous Coordinates

Transform:

$$\begin{bmatrix} c_{11} & c_{21} & c_{31} & c_{41} \\ c_{12} & c_{22} & c_{32} & c_{42} \\ c_{13} & c_{23} & c_{33} & c_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$\text{transformed } x = c_{11}x_1 + c_{21}y_1 + c_{31}z_1 + c_{41}$$

Transform:

$$\begin{bmatrix} c_{11} & c_{21} & c_{31} & c_{41} \\ c_{12} & c_{22} & c_{32} & c_{42} \\ c_{13} & c_{23} & c_{33} & c_{43} \end{bmatrix} \begin{bmatrix} (1-t)x_1 + tx_2 \\ (1-t)y_1 + ty_2 \\ (1-t)z_1 + tz_2 \\ (1-t)1 + t1 \end{bmatrix}$$

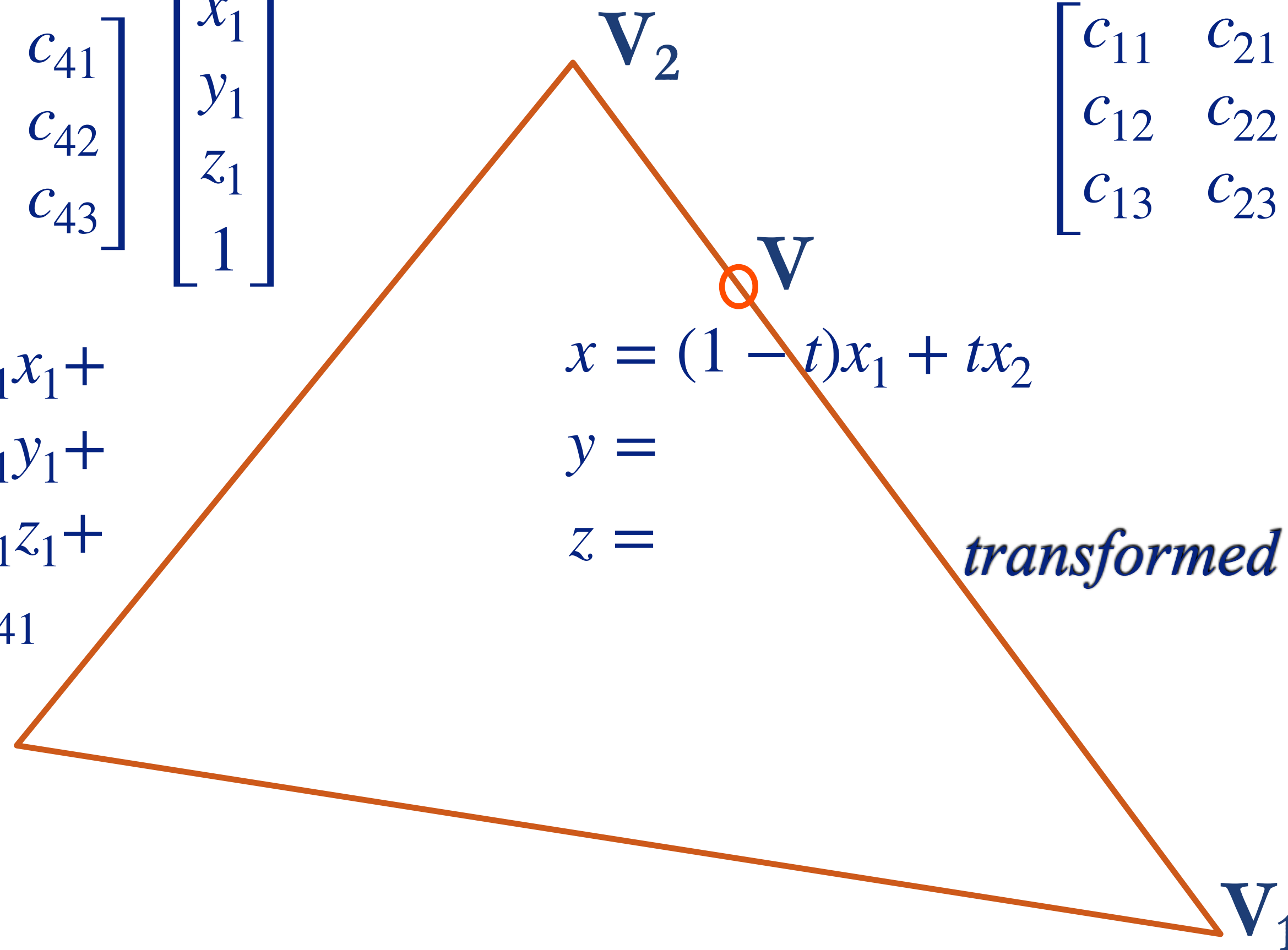
$$x = (1-t)x_1 + tx_2$$

$$y =$$

$$z =$$

transformed $x =$

$$(1-t)c_{11}x_1 + tc_{11}x_2 + (1-t)c_{21}y_1 + tc_{21}y_2 + (1-t)c_{31}z_1 + tc_{31}z_2 + c_{41}$$



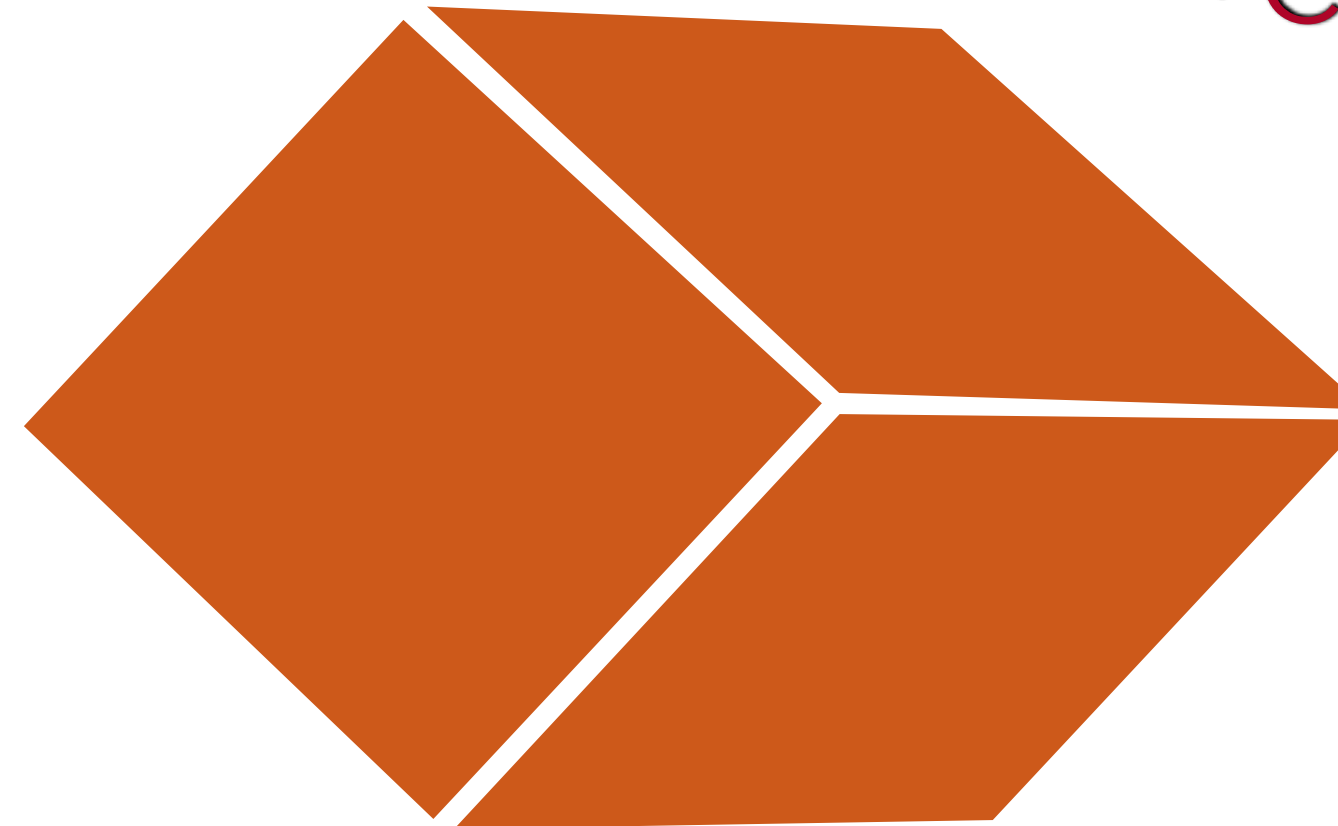
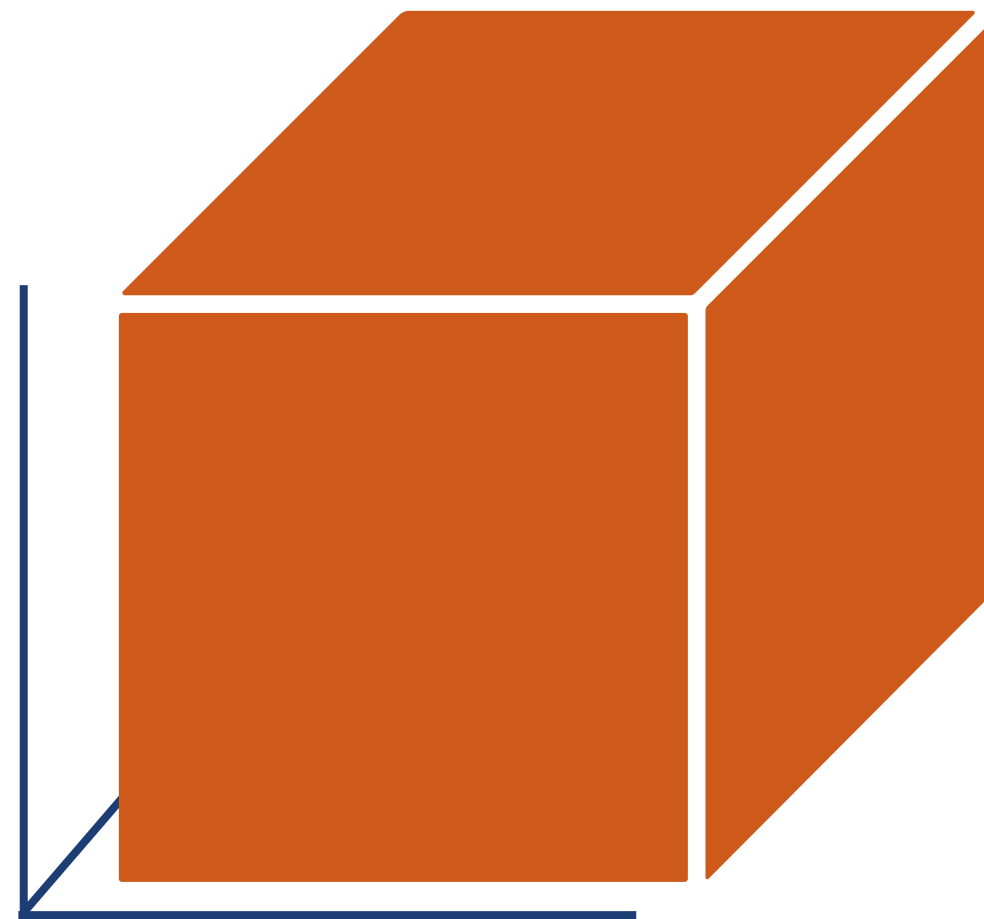


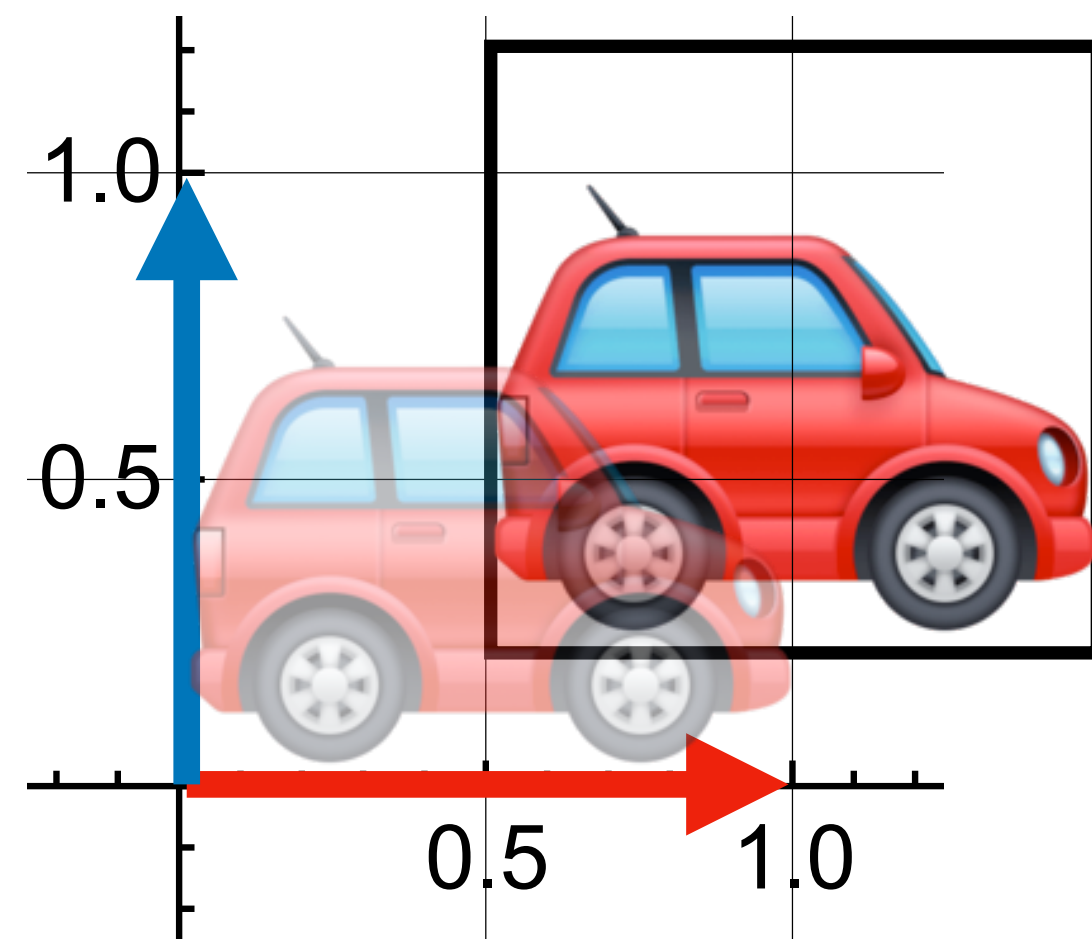
How to Specify Transformation?

Transform Points and Vectors

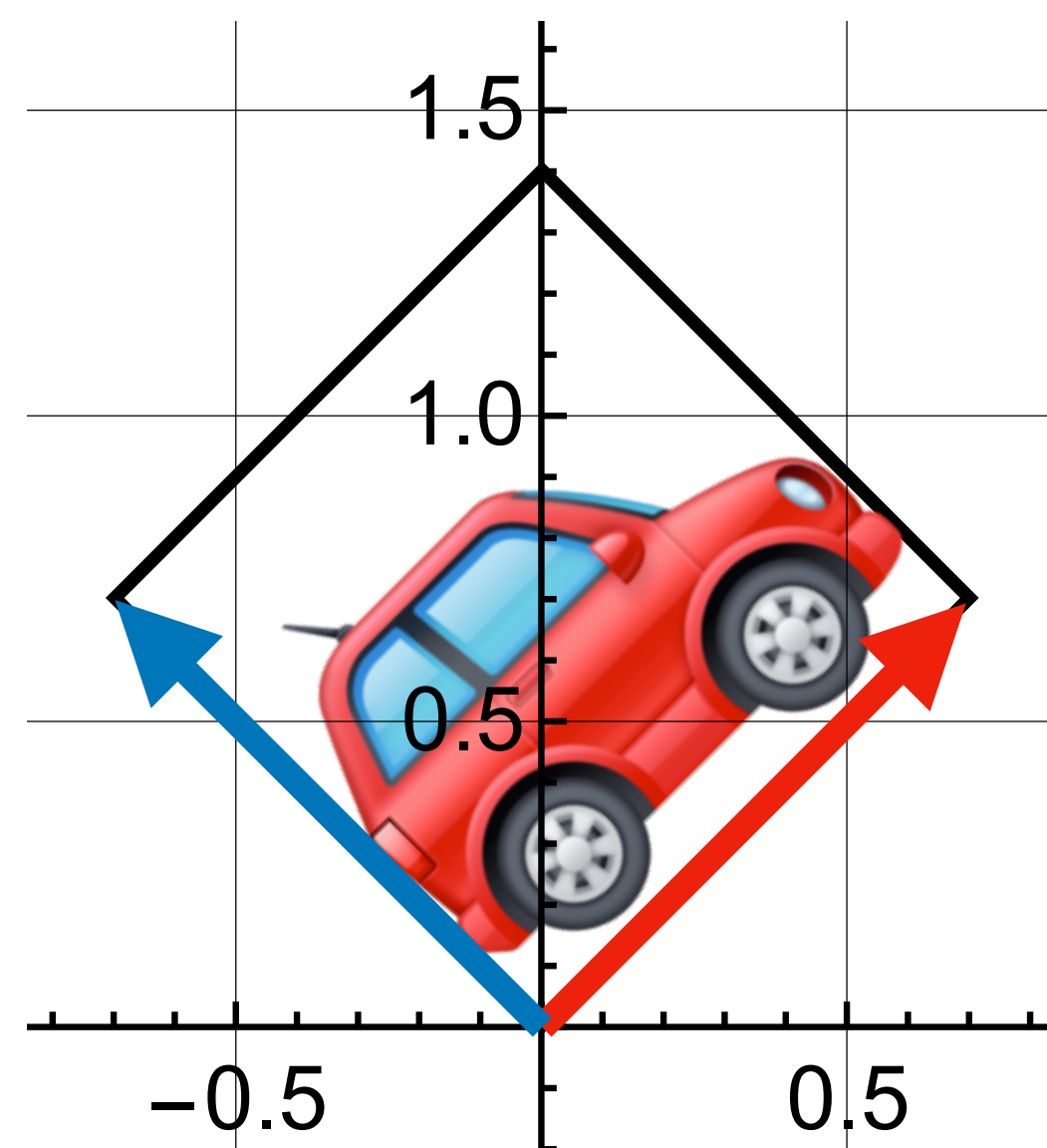
- $V_1 \rightarrow V_1'$?
- $(V_1, V_2) \rightarrow (V_1', V_2')$?
- 3x4 matrix has 12 elements ... map 4 vertices?
 - ➔ It's useful to think of constrained transformations

- Translate (no reshape)
- Rotate (no reshape)
- Resize (no reshape)
- Scale (reshapes)
- Shear (reshapes)
- Change basis (no reshape)

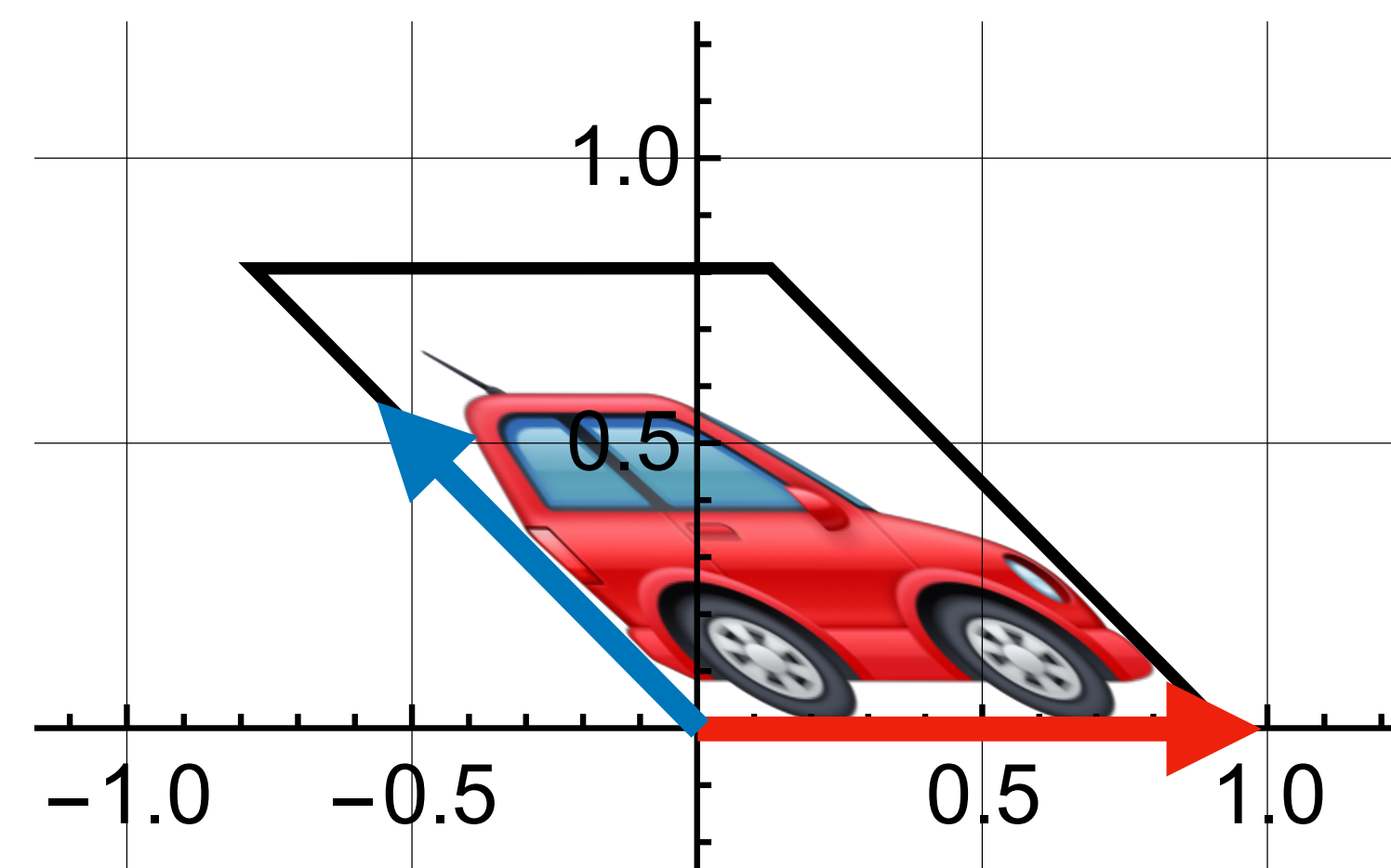




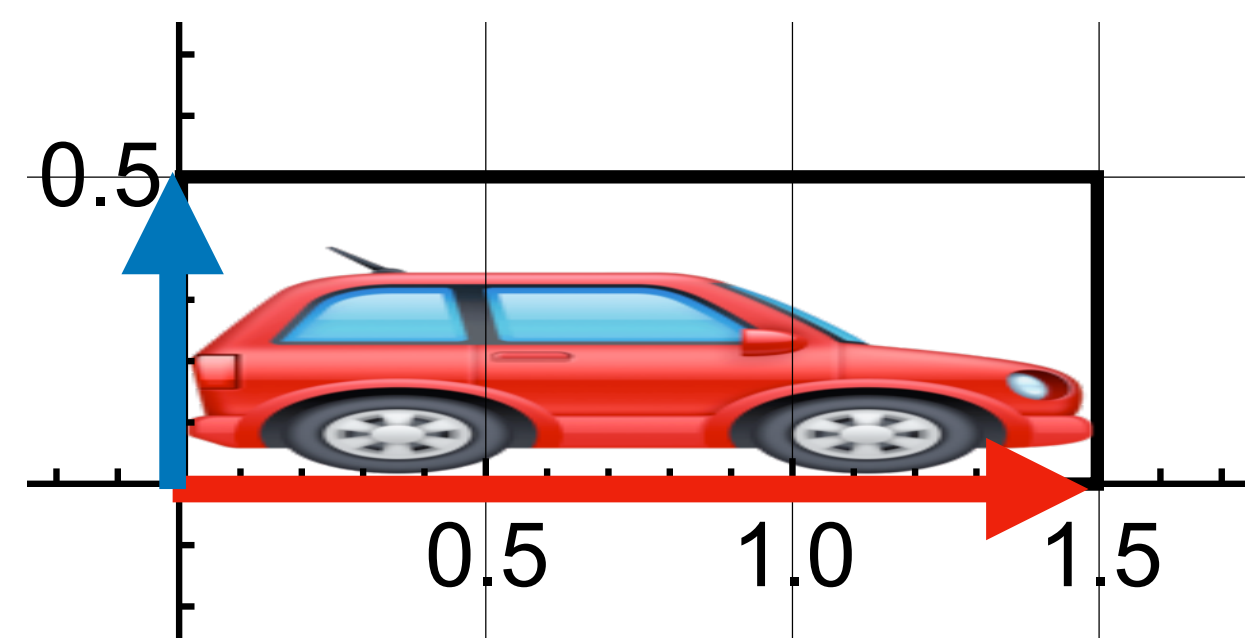
Translation



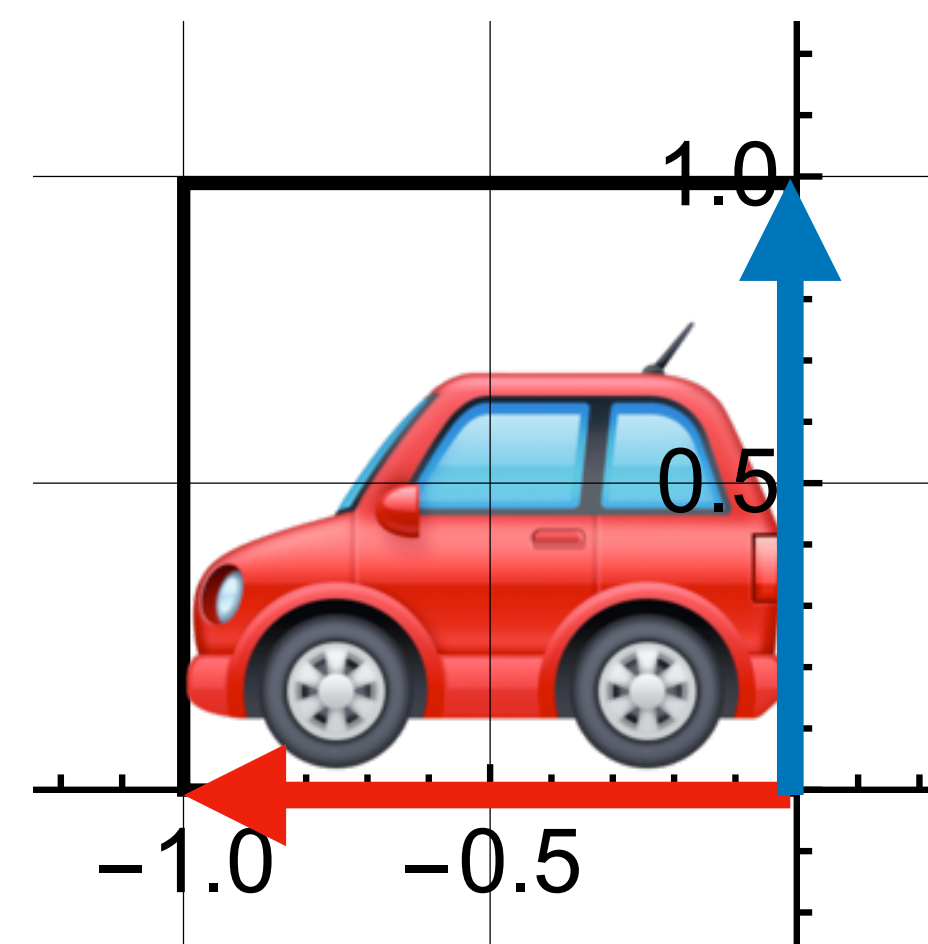
Rotation



Shear



Scaling

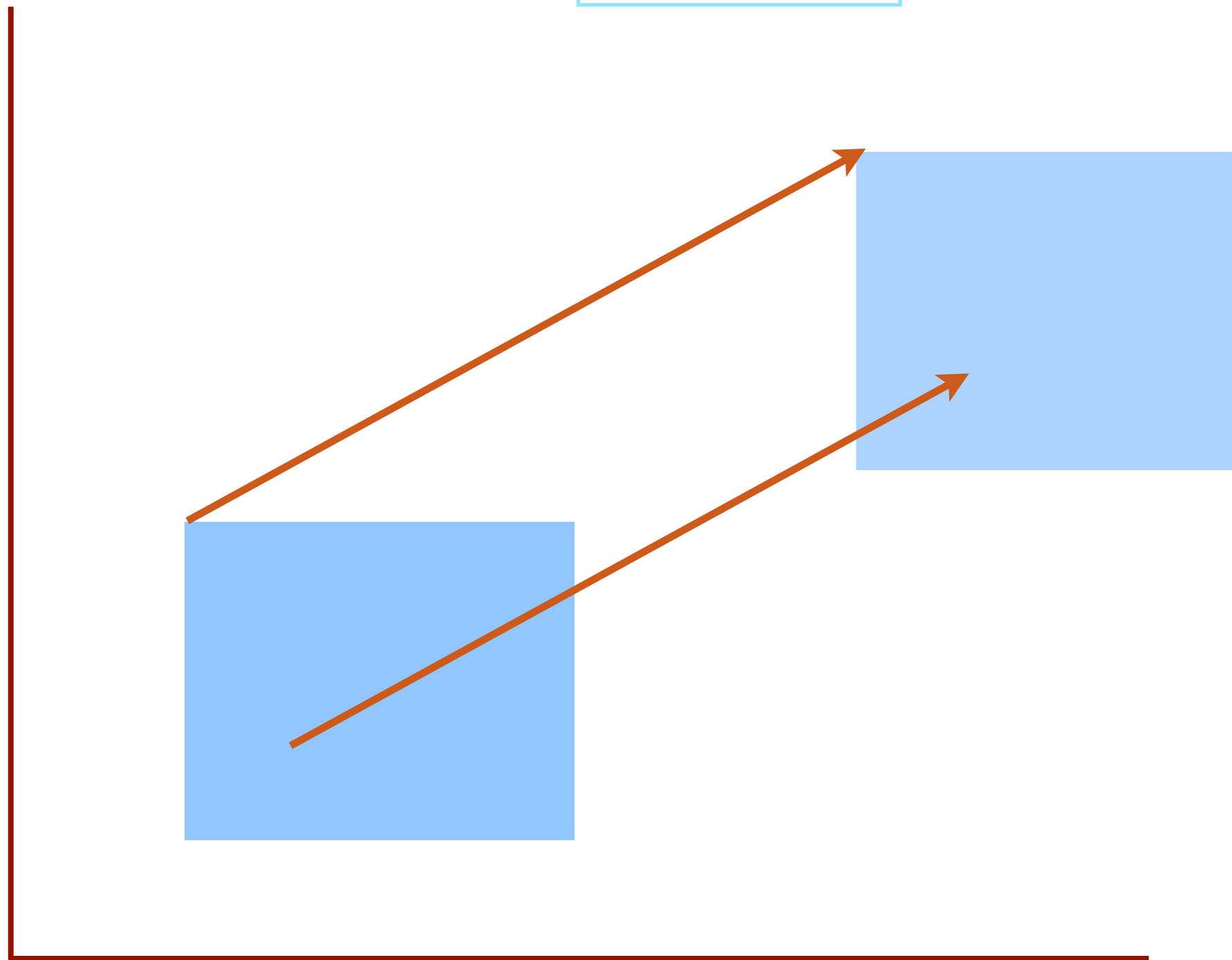


Reflection



Translation

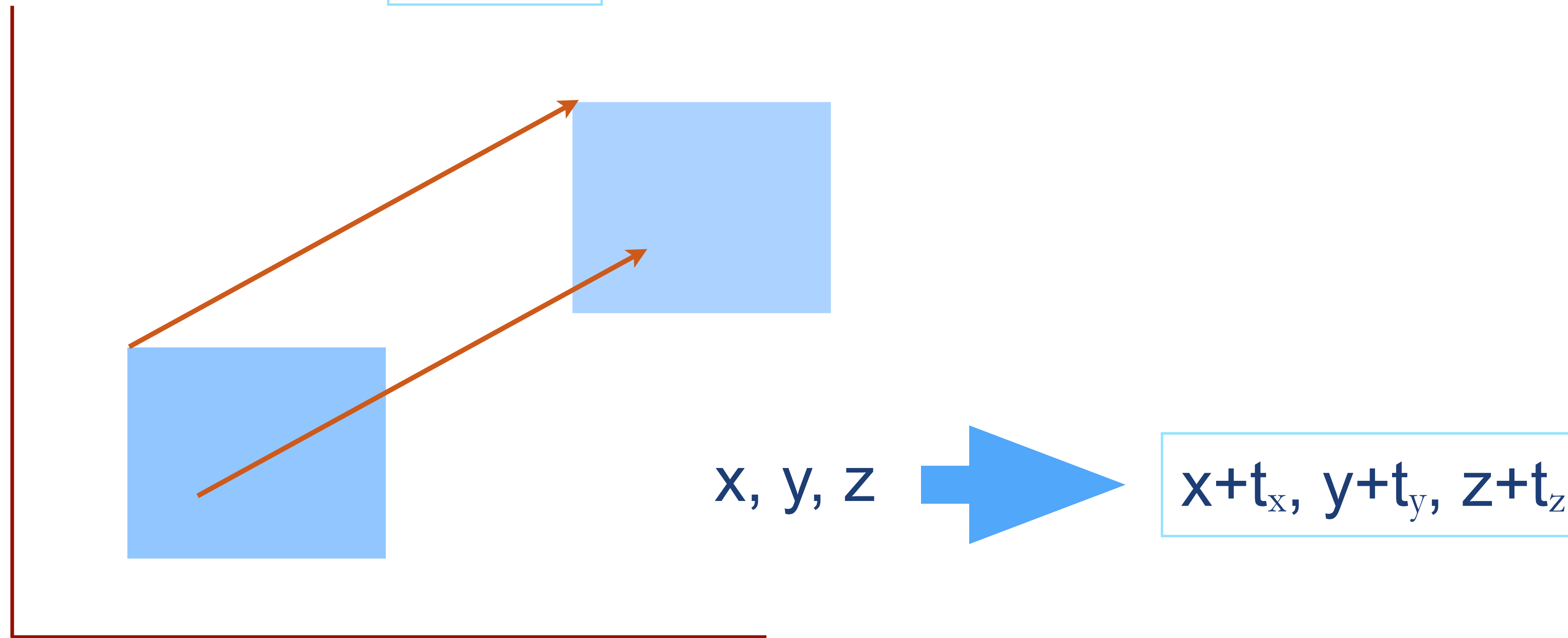
- Translate by a Vector t_x, t_y, t_z





Translation

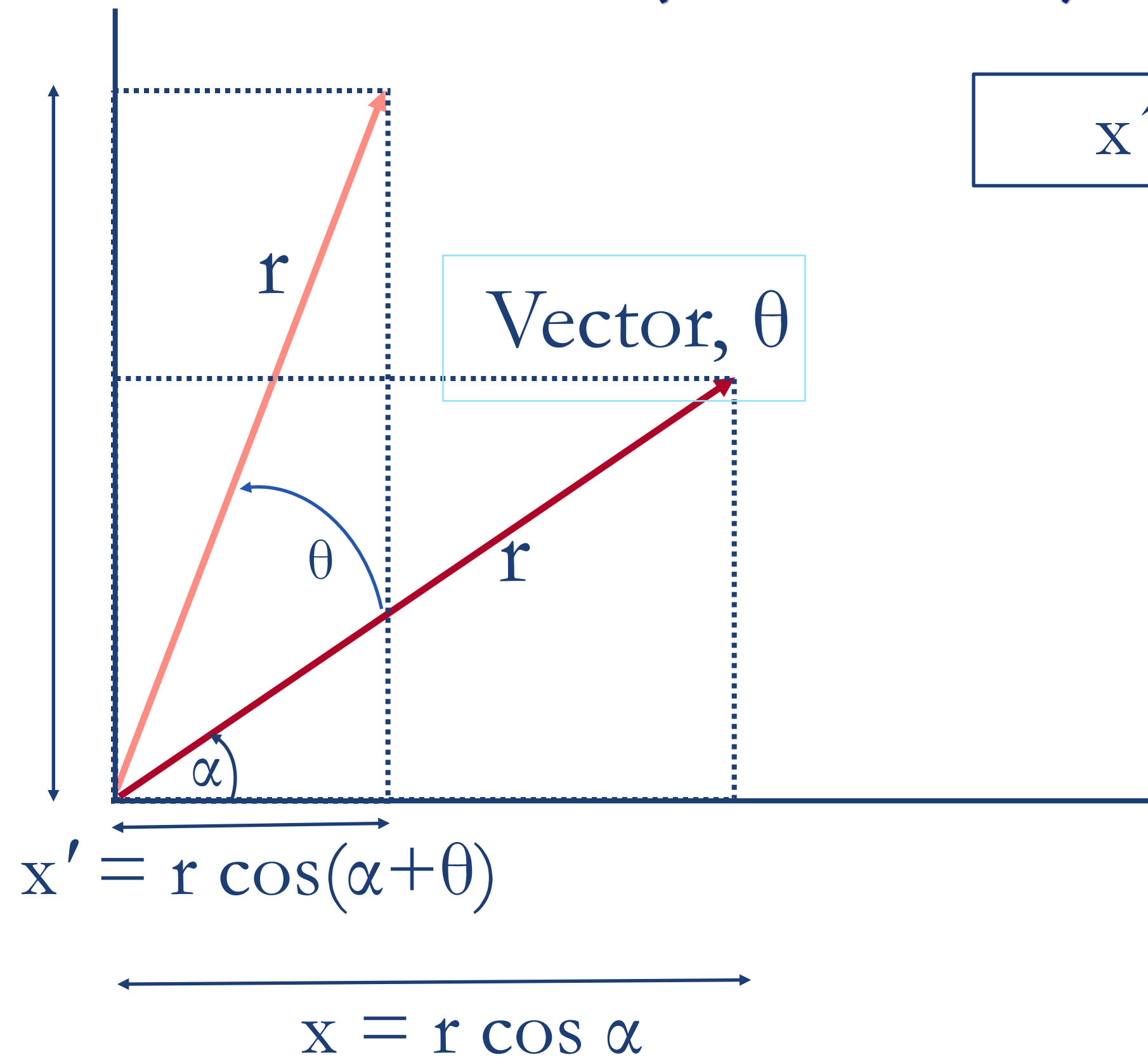
- Translate by a Vector t_x, t_y, t_z





Rotation

- Rotate about an axis (of rotation)



$$x' = r \cos \alpha \cos \theta - r \sin \alpha \sin \theta = x \cos \theta - y \sin \theta$$

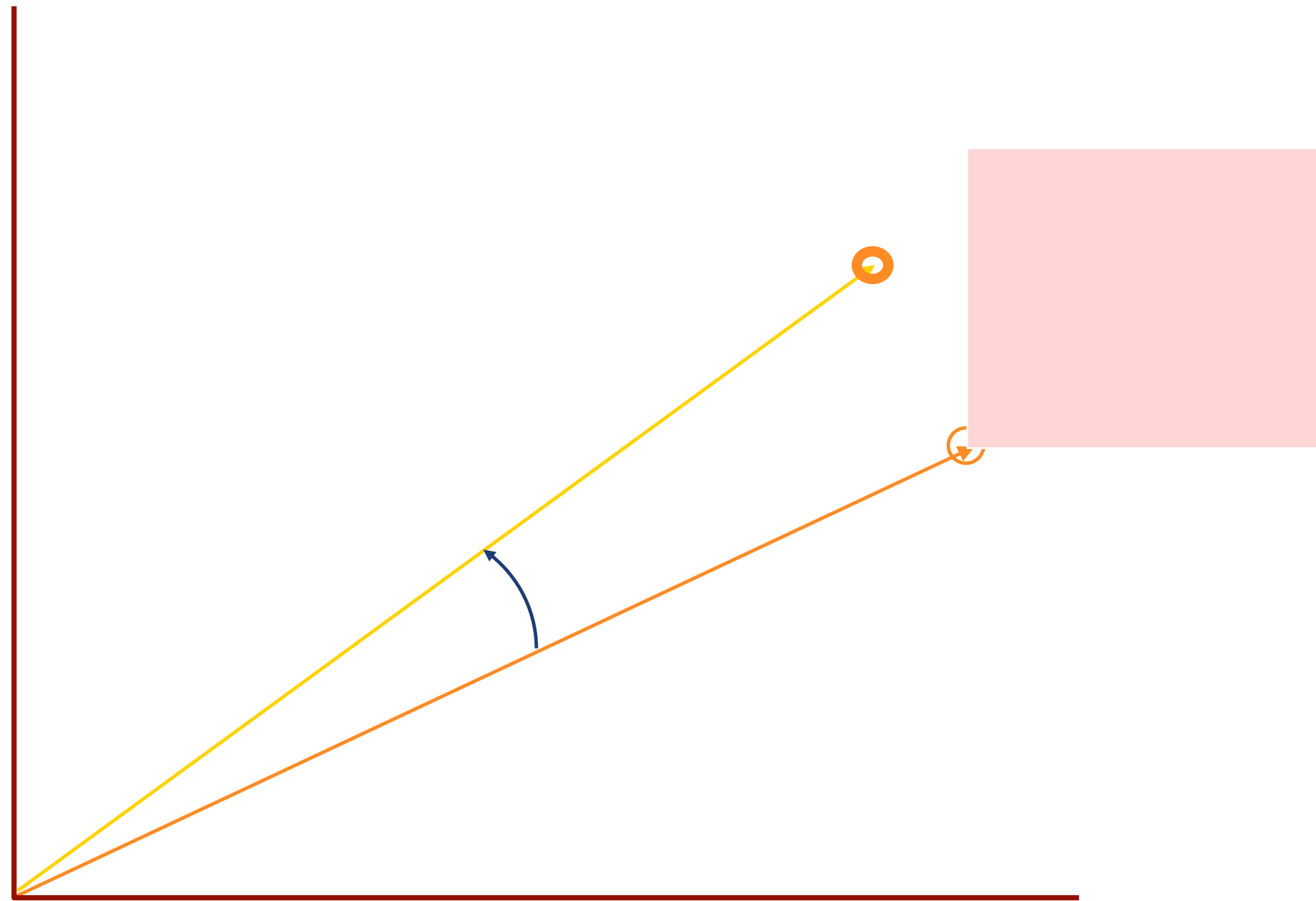
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

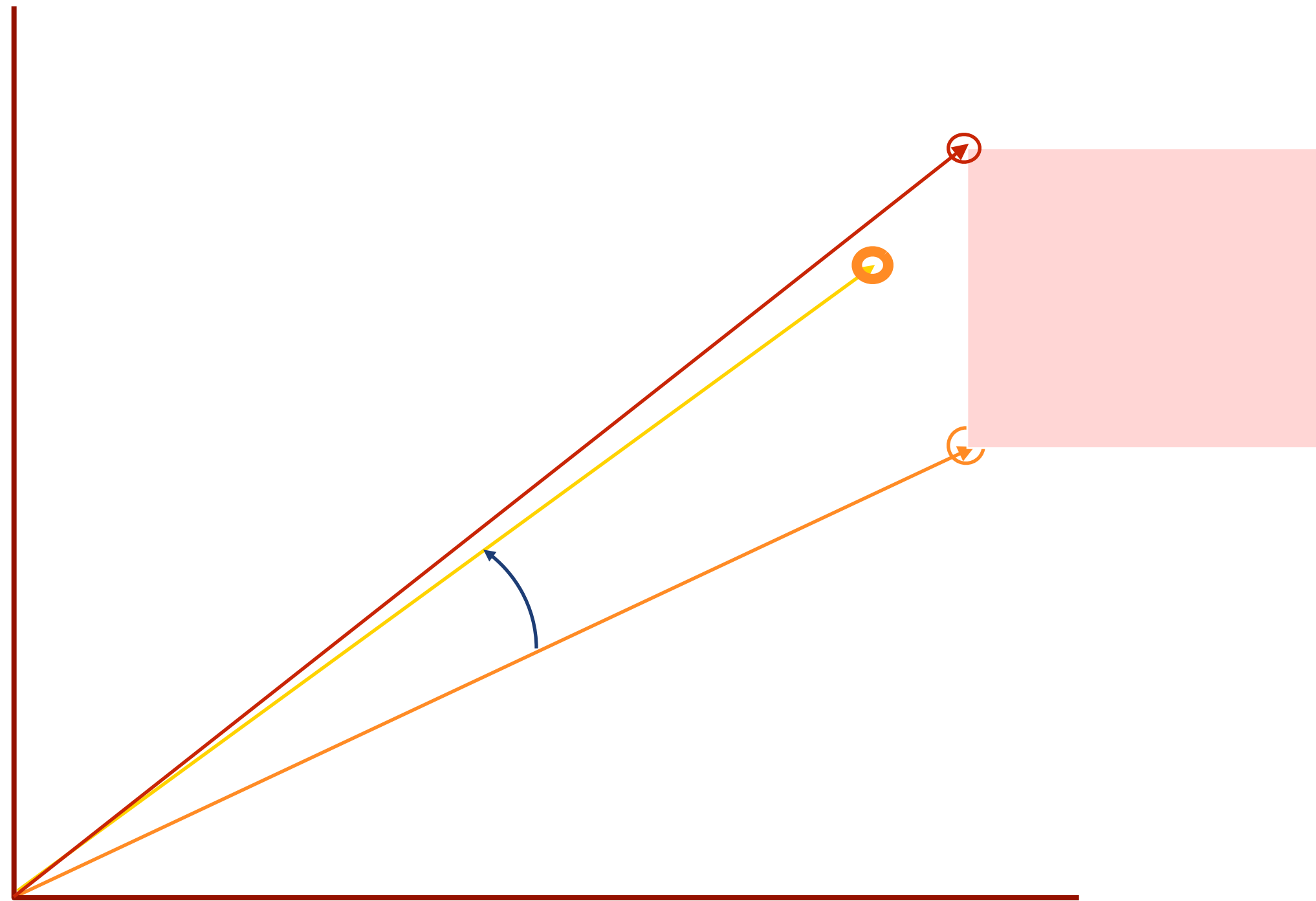
orthonormal

3D

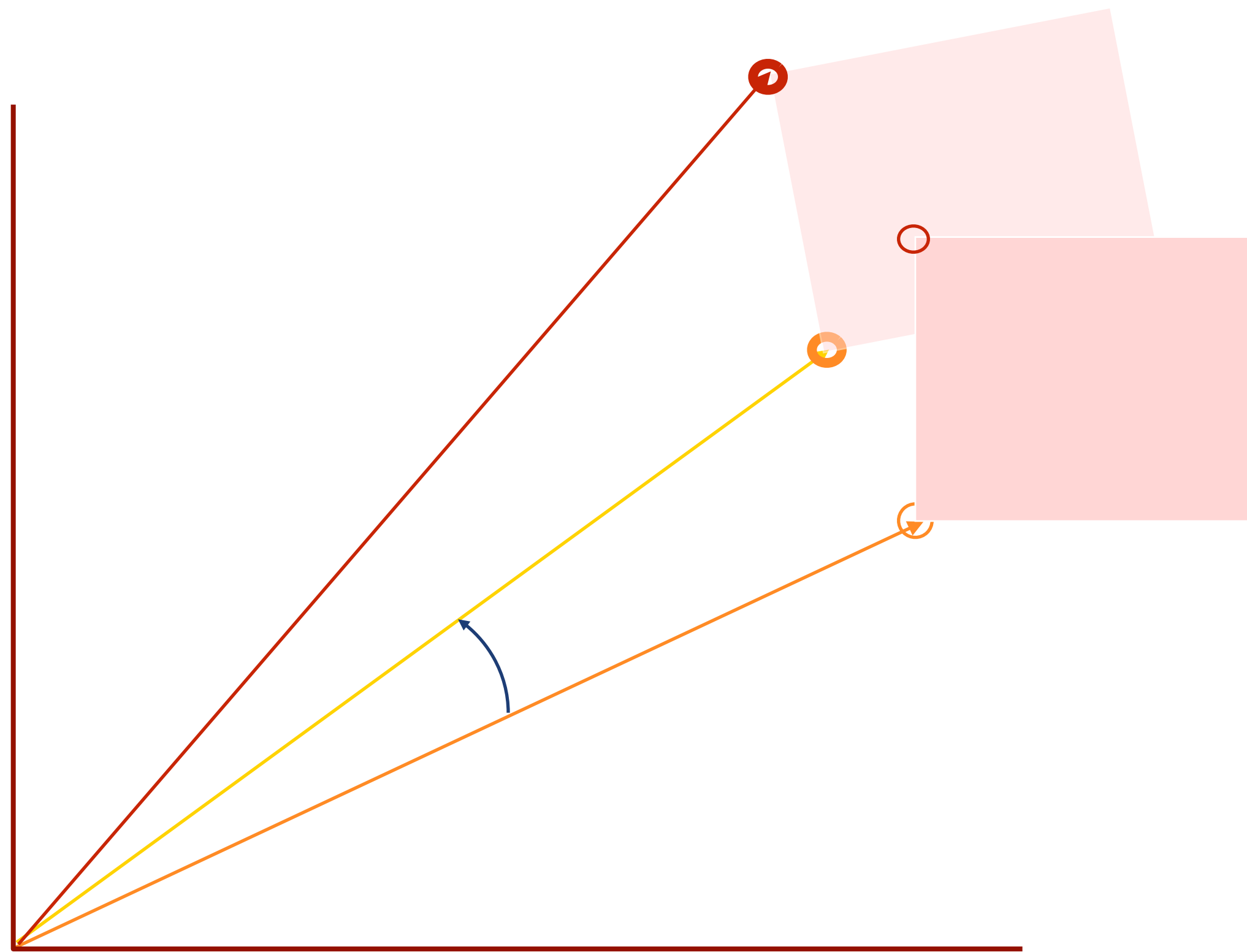
(Somewhat) General Rotation



(Somewhat) General Rotation



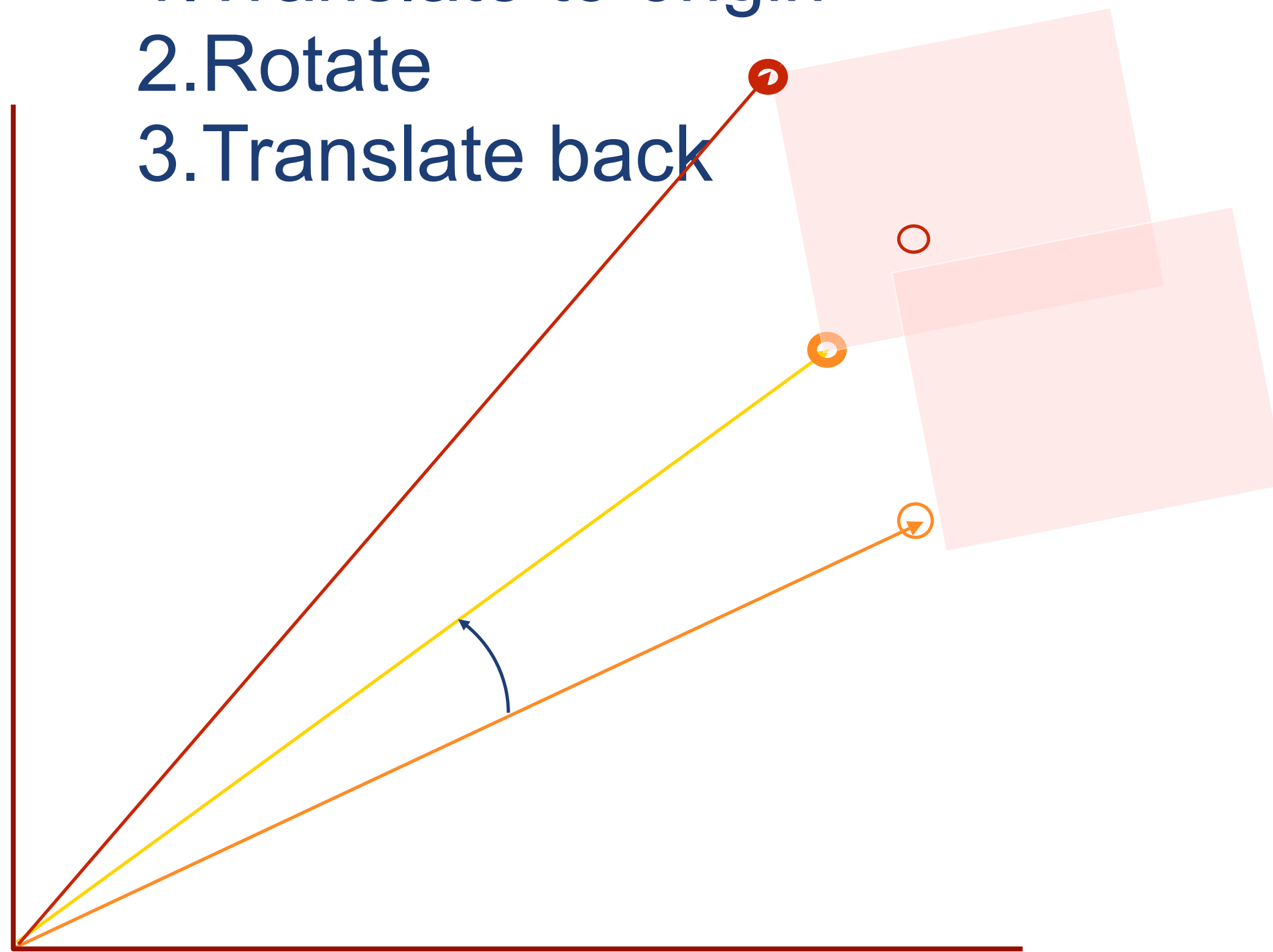
(Somewhat) General Rotation



(Somewhat) General Rotation



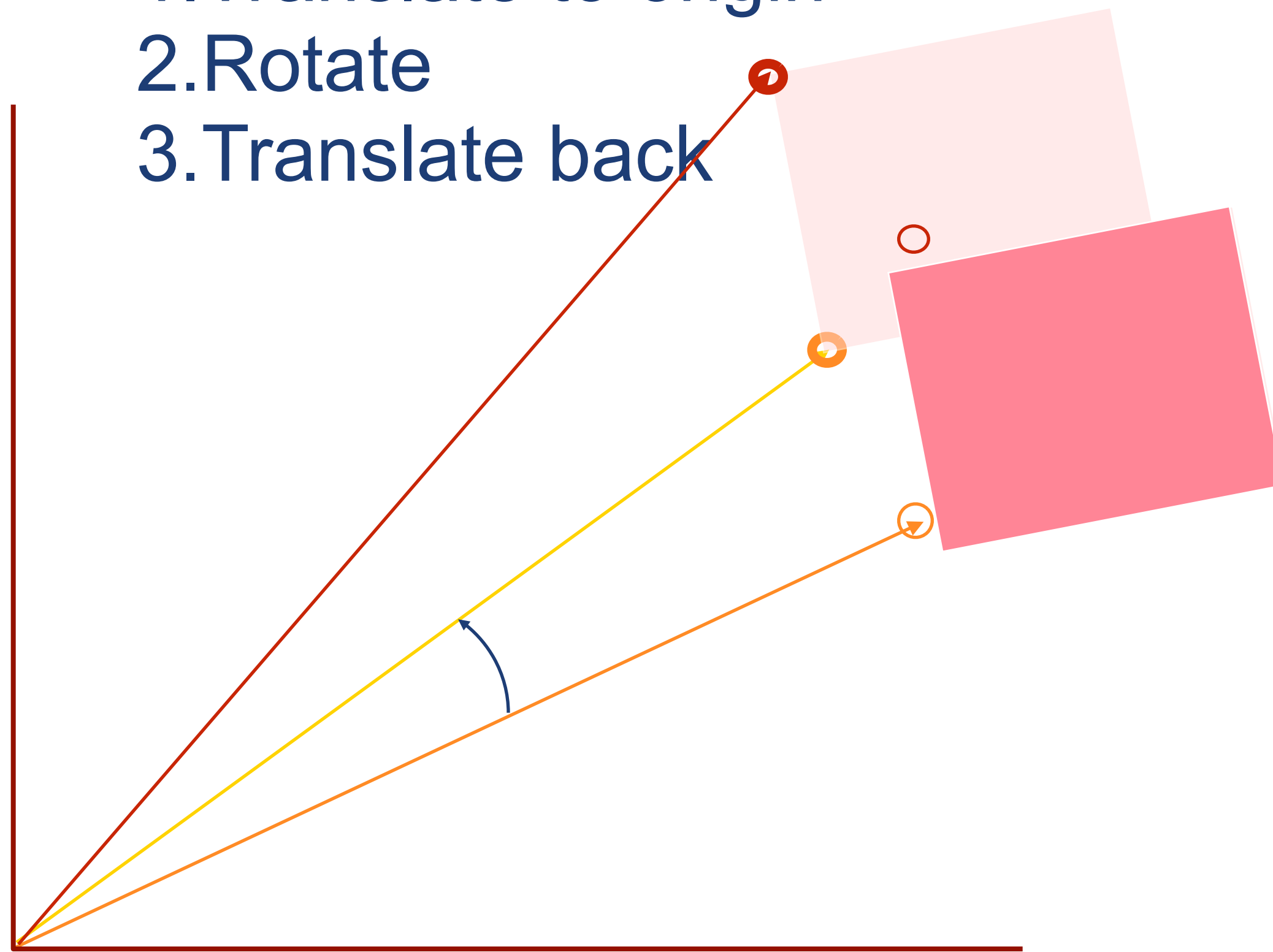
1. Translate to origin
2. Rotate
3. Translate back



(Somewhat) General Rotation



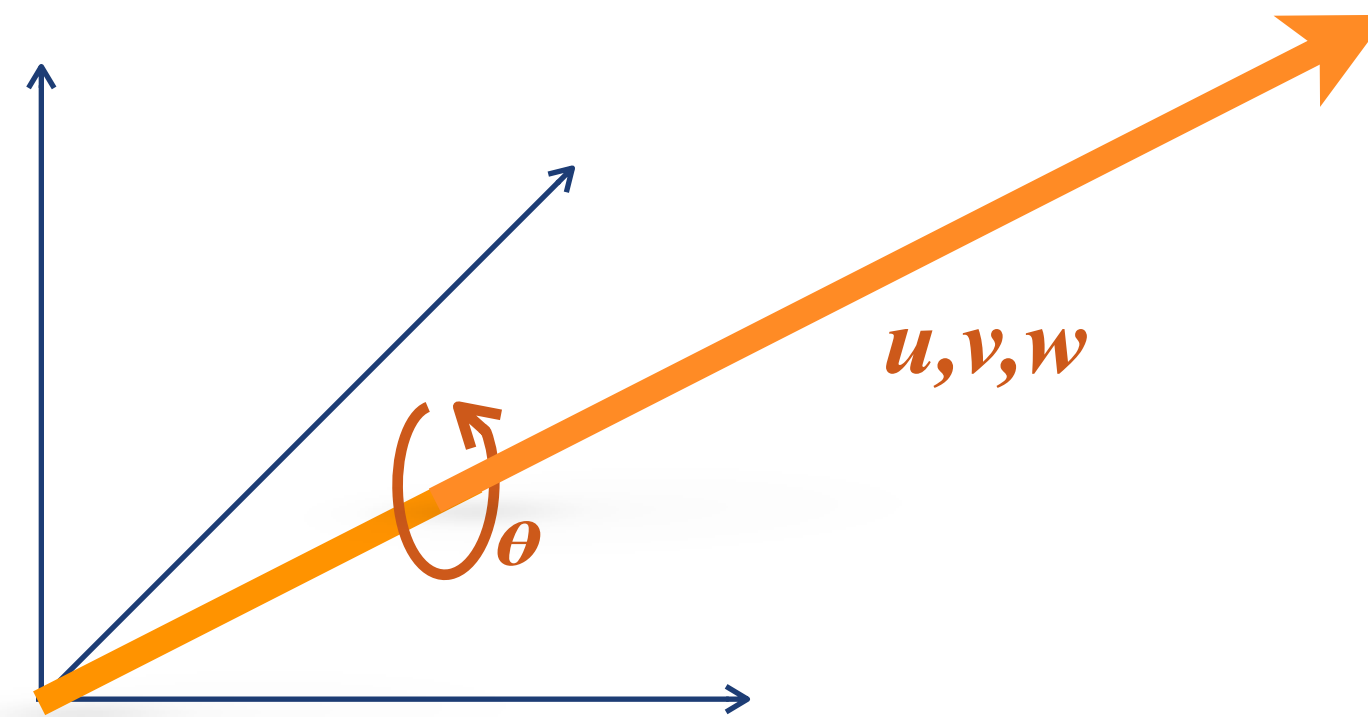
1. Translate to origin
2. Rotate
3. Translate back



Rotation About Arbitrary Axis



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} u.u(1 - \cos \theta) + \cos \theta & w.u(1 - \cos \theta) - w \sin \theta & u.w(1 - \cos \theta) - v \sin \theta \\ v.u(1 - \cos \theta) + w \sin \theta & w.u(1 - \cos \theta) + \cos \theta & v.w(1 - \cos \theta) - u \sin \theta \\ w.u(1 - \cos \theta) - v \sin \theta & w.u(1 - \cos \theta) - u \sin \theta & w.w(1 - \cos \theta) + \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$





Quaternions

- Relates two vectors
 - Just as a vector relates two points
- $q = \langle s, x, y, z \rangle = \langle s, \mathbf{v} \rangle = s + xi + yj + zk$
 - Scalar, **Vector**: relates length and orientation
 - $\cos(\theta/2), \mathbf{U} \sin(\theta/2)$

Quaternion addition: $(s_1, \mathbf{v}_1) + (s_2, \mathbf{v}_2) = (s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2)$

Quaternion Scaling: $a(s, \mathbf{v}) = (as, a\mathbf{v})$

Quaternion magnitude: $|(s, \mathbf{v})| = \sqrt{s^2 + |\mathbf{v}|^2}$

Quaternion multiplication: $q_1 q_2 = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$

Quaternion inverse: If $q = (s, \mathbf{v})$, then $q^{-1} = (s, -\mathbf{v})/|q|^2$

Vector $\mathbf{v}$ to quaternion V :

▸ $(0, \mathbf{v})$

Apply quaternion q to V

▸ $q V q^{-1} \equiv (0, \mathbf{v}')$

$(s, \mathbf{p})(0, \mathbf{v})(s, -\mathbf{p})$

$$\mathbf{v}' = s^2 \mathbf{v} + \mathbf{p}(\mathbf{v} \cdot \mathbf{p}) + 2s(\mathbf{p} \times \mathbf{v}) + \mathbf{p} \times (\mathbf{p} \times \mathbf{v})$$

Scale



- Scale by factor(s)

S $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$

Scale



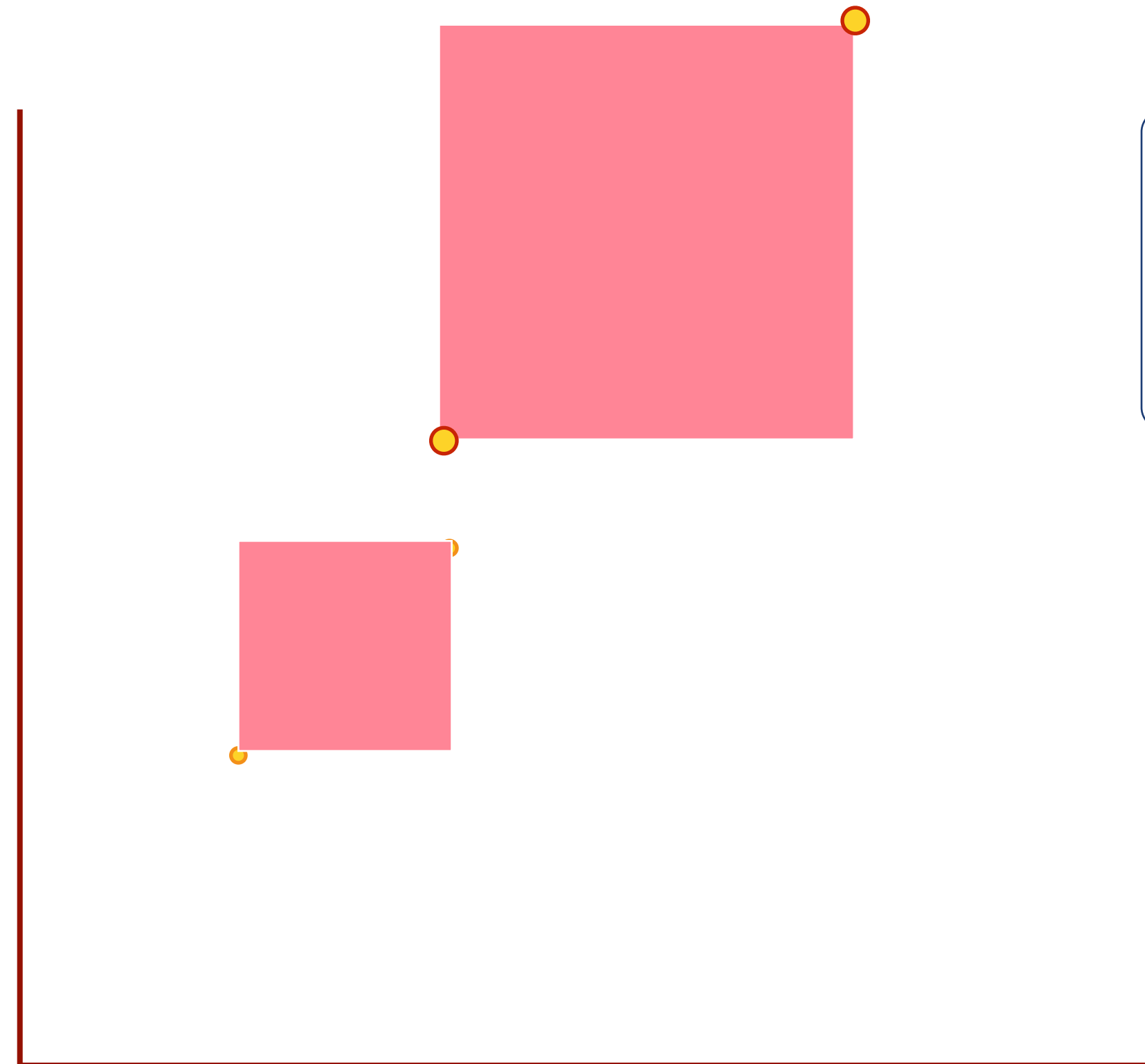
- Scale by factor(s)

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Scale

- Scale by factor(s)

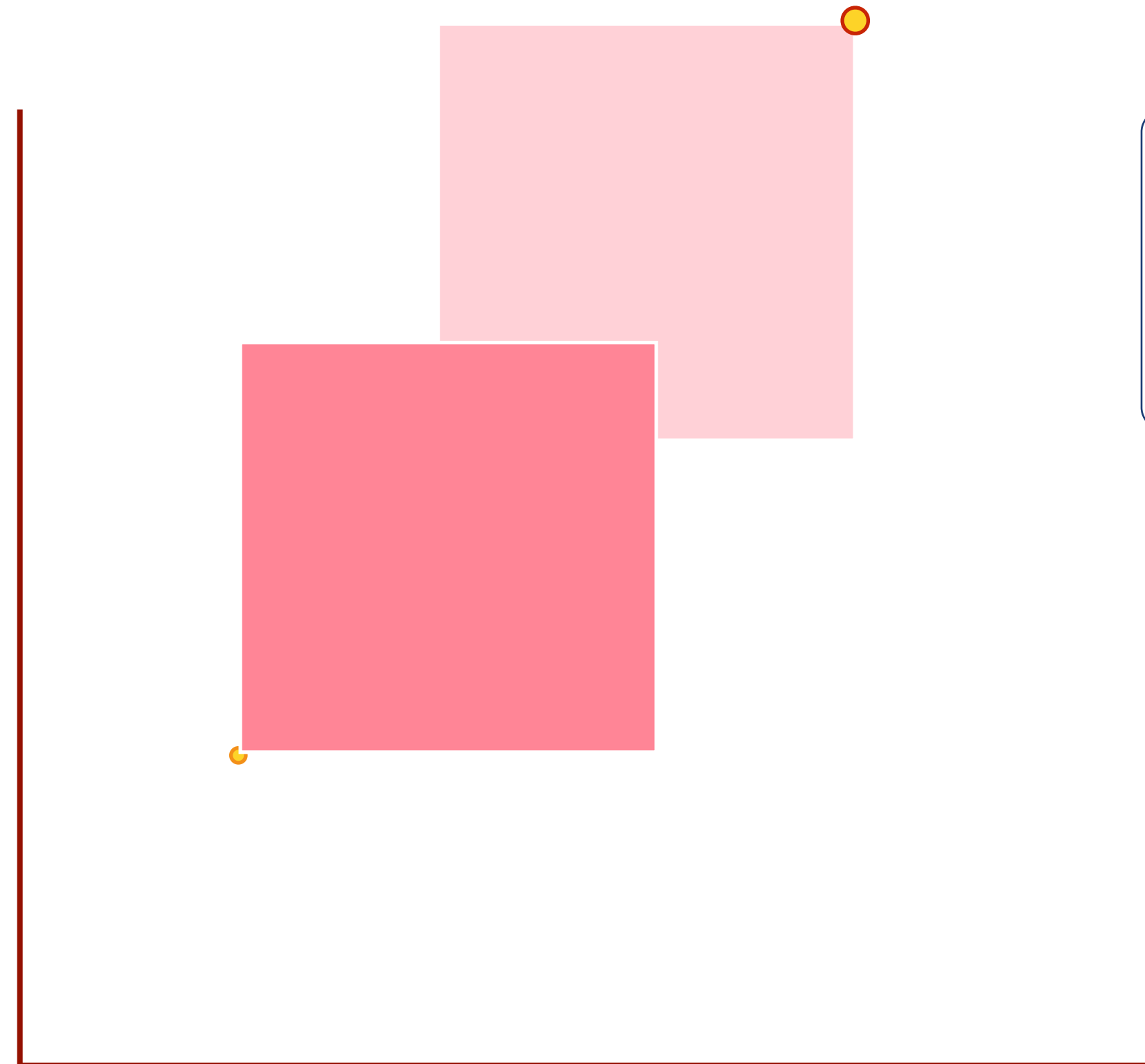


$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Scale

- Scale by factor(s)

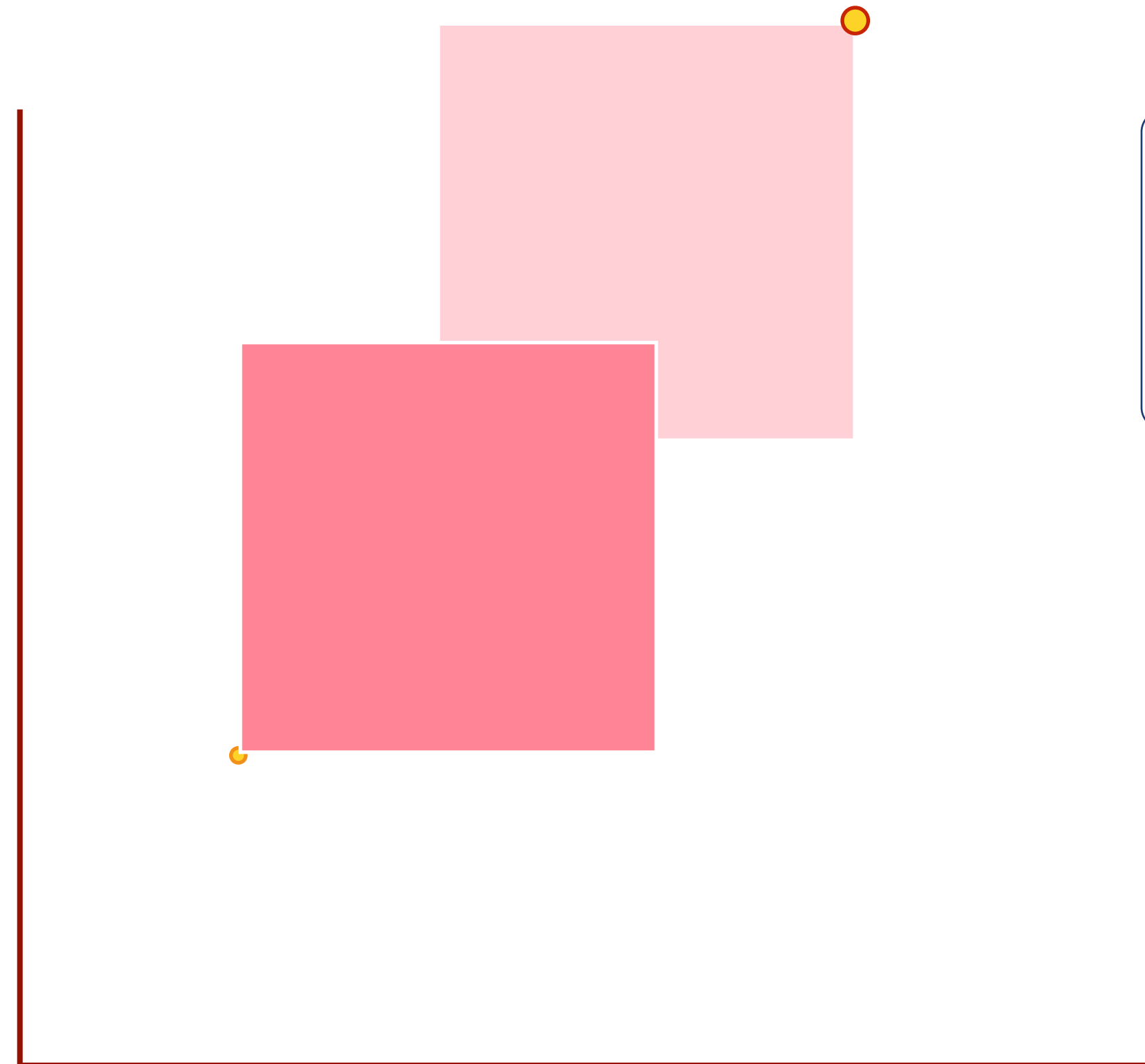


$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Scale

- Scale by factor(s)



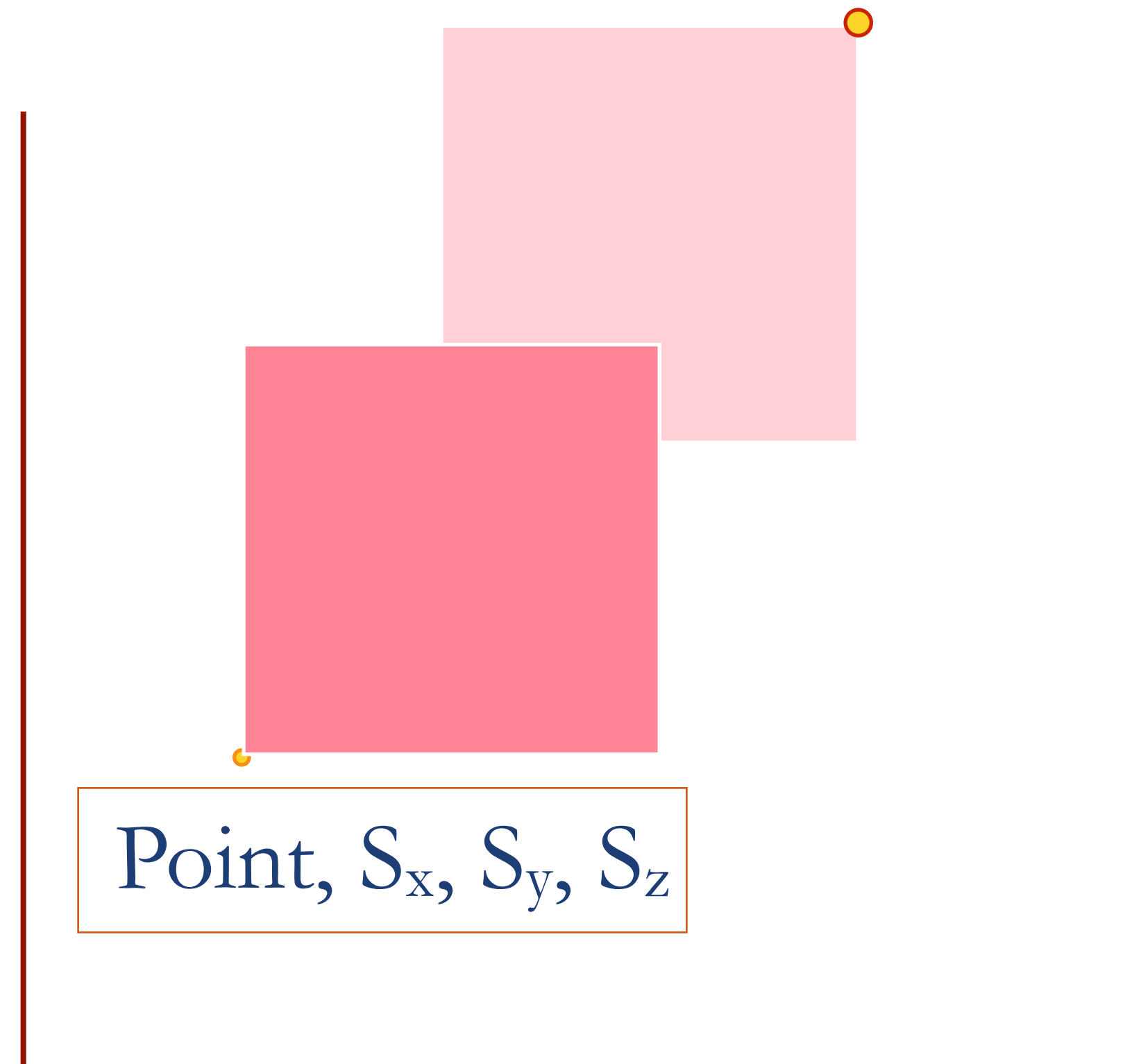
$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(About origin)



Scale

- Scale by factor(s)



$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(About origin)



Translation

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

4th coordinate = Homogeneous Coordinates

Affine transformation

- Maps line to line, plane to plane
- Preserves parallelism

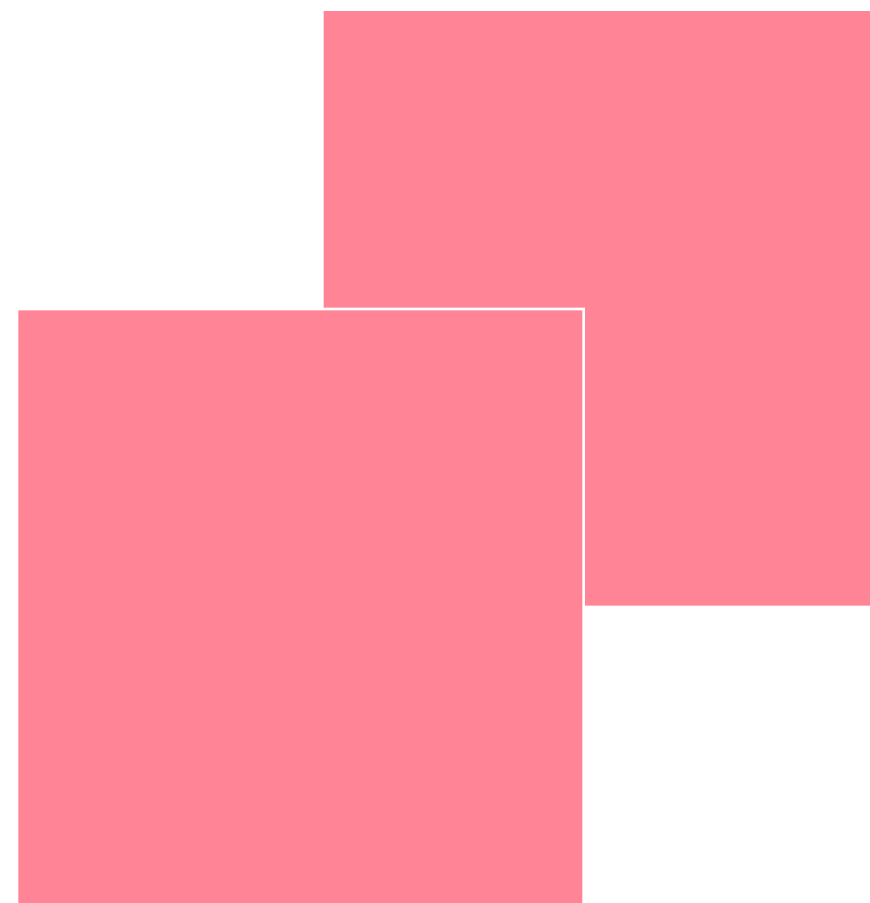


Using 4x4 Matrices

- Scale by factors about point

Point, S_x , S_y , S_z

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



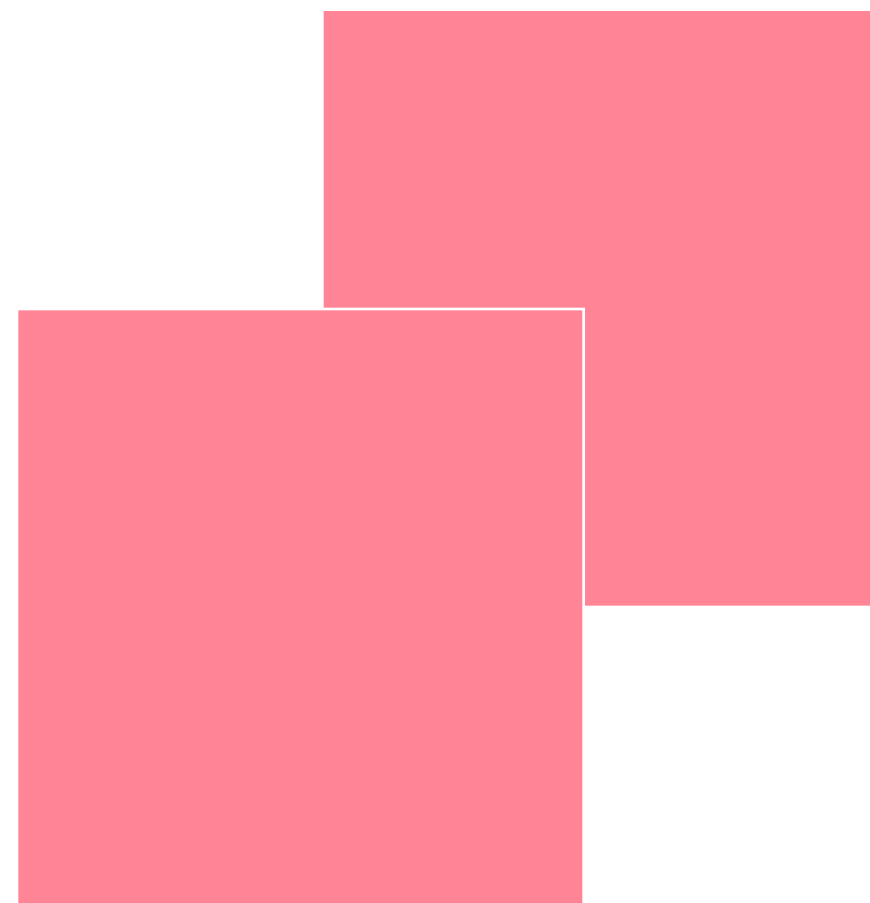


Using 4x4 Matrices

- Scale by factors about point

Point, S_x , S_y , S_z

$$\begin{pmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



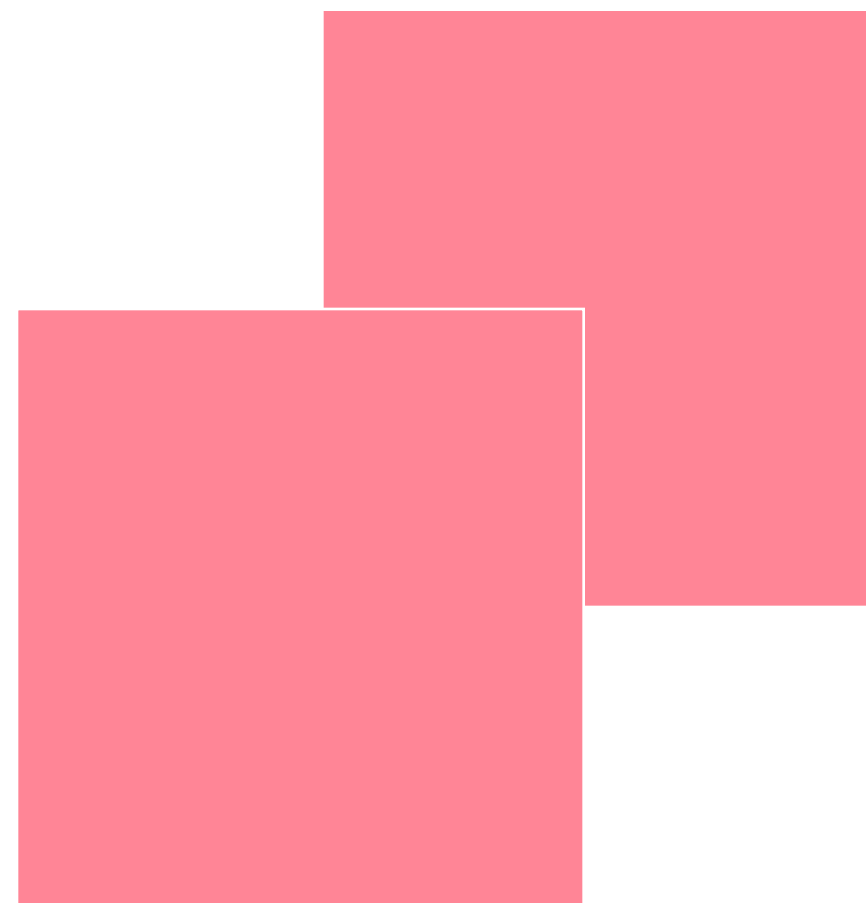


Using 4x4 Matrices

- Scale by factors about point

Point, S_x , S_y , S_z

$$\begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} S_x & 0 & 0 & t_x \\ 0 & S_y & 0 & t_y \\ 0 & 0 & S_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

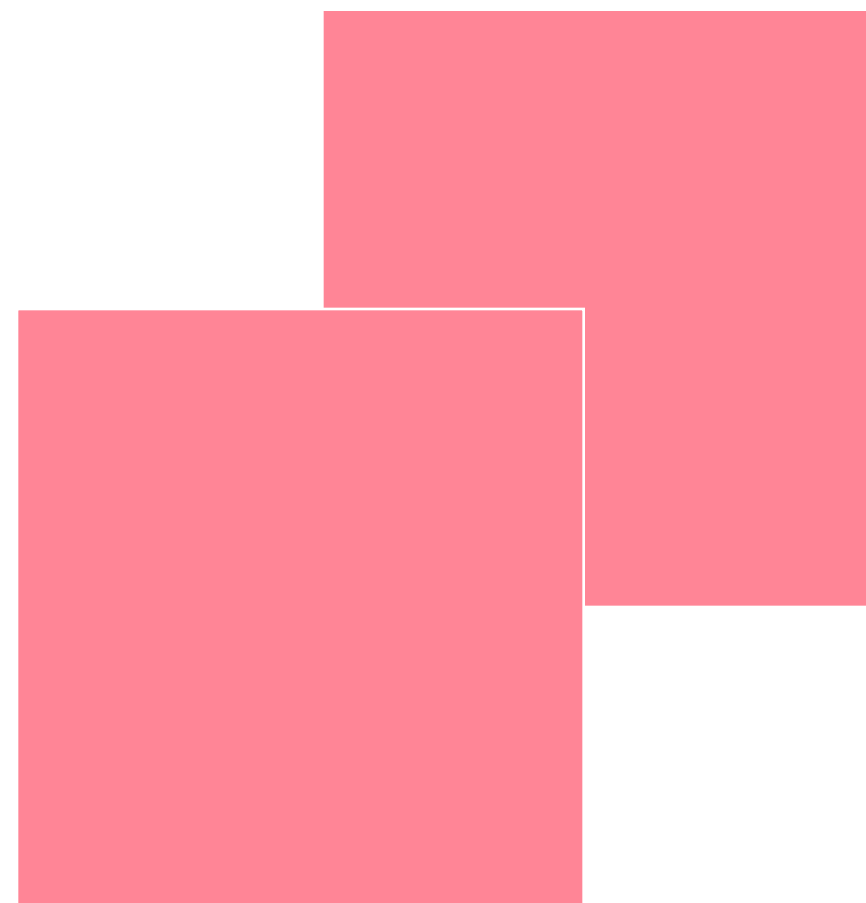


Using 4x4 Matrices

- Scale by factors about point

Point, S_x , S_y , S_z

$$\begin{pmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

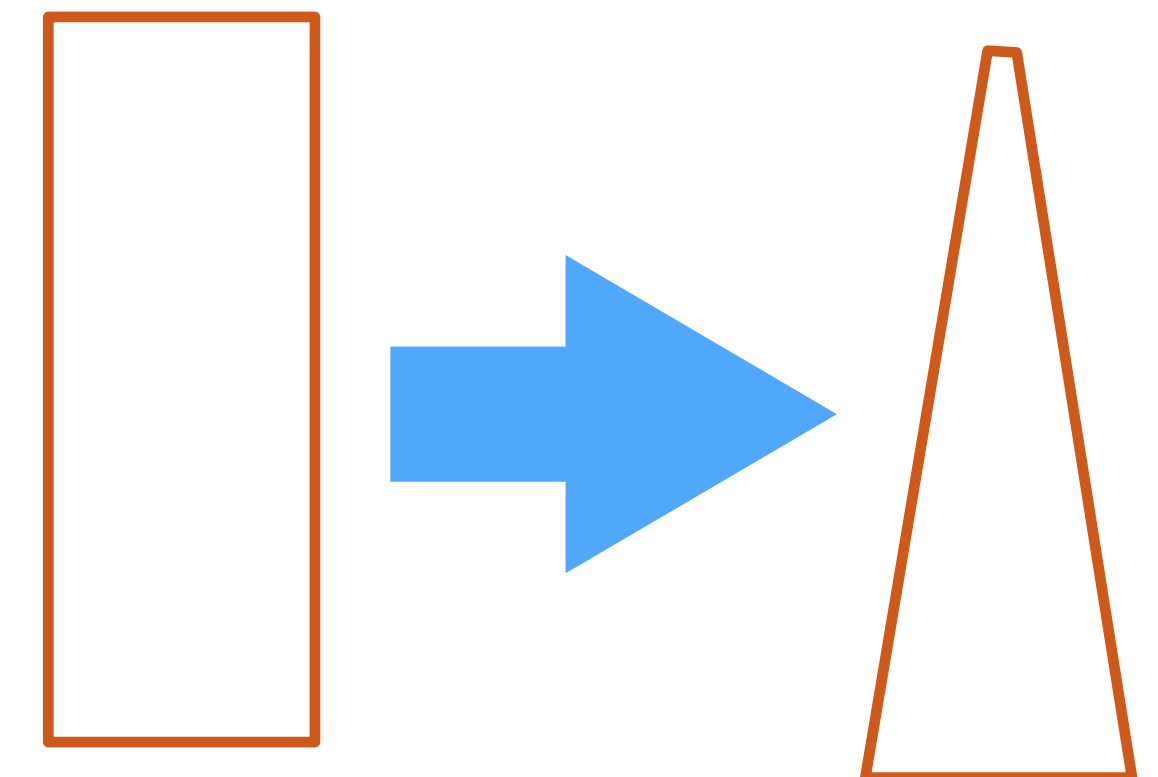
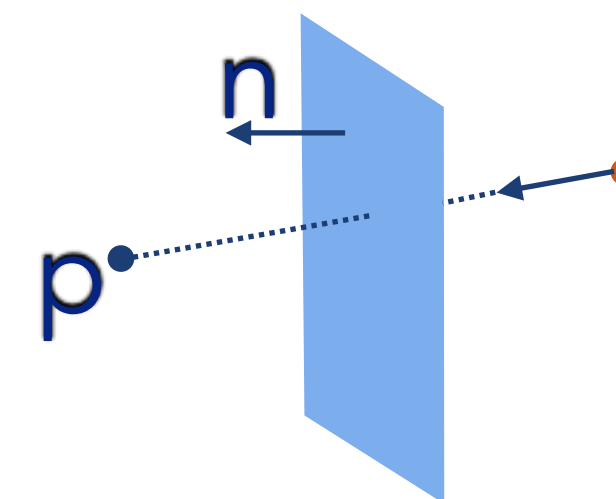
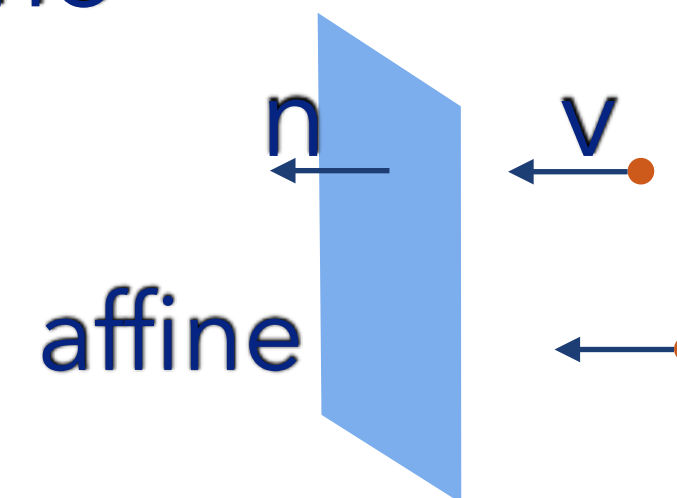


$$\begin{pmatrix} S_x & 0 & 0 & t_x \\ 0 & S_y & 0 & t_y \\ 0 & 0 & S_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$



Projection of Points

- Intersection of a path with a shape
 - ➔ Path originates at the point
 - We will focus on linear path (i.e., direction)
 - ➔ Planar projection if the shape is a plane
 - ➔ Parallel
 - Constant Vector
 - Orthographic (perpendicular to plane), Oblique
 - ➔ Perspective
 - Vector to a given 'center of projection'





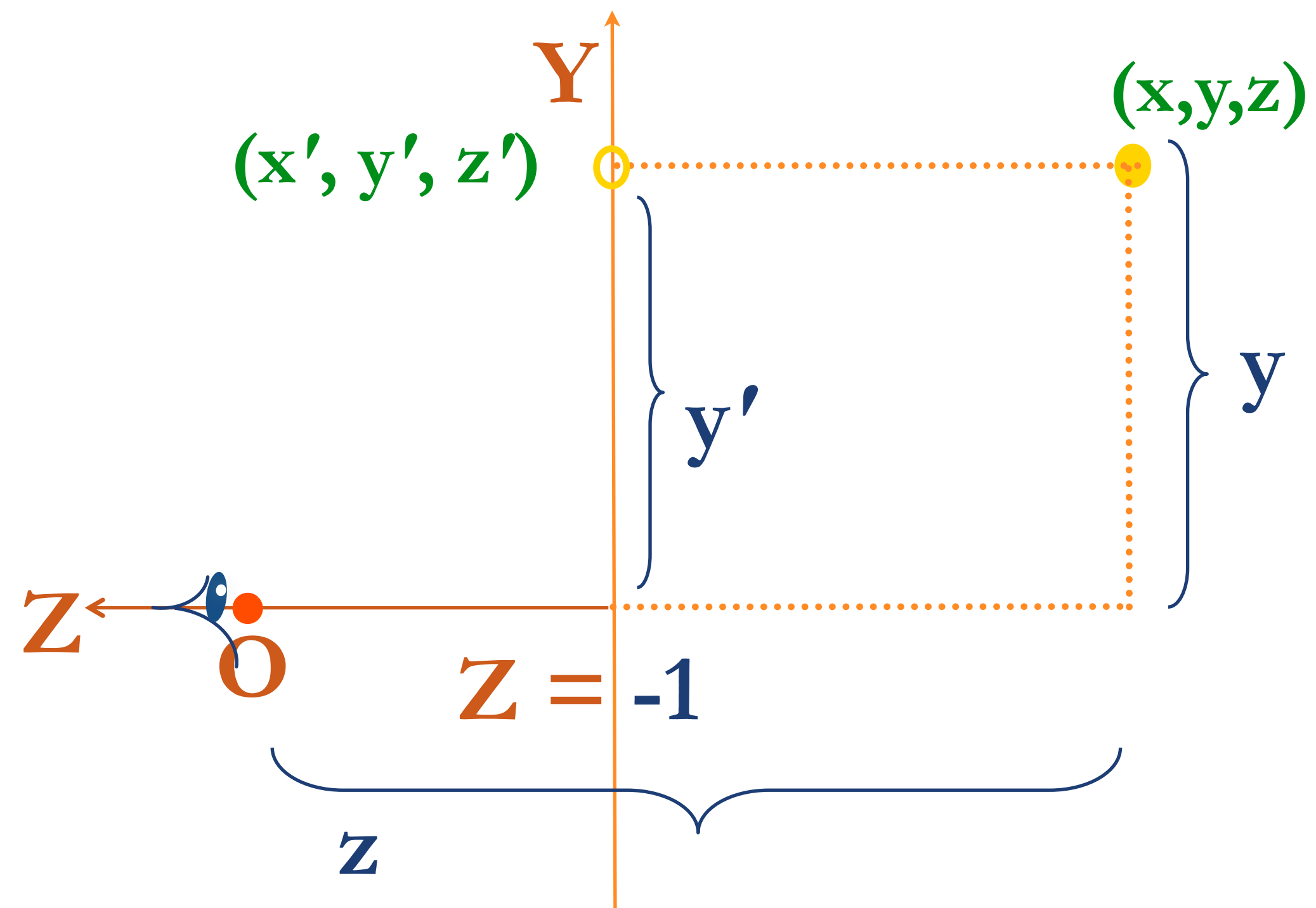
Parallel Projection

- Transform plane of projection to $Z = -1$

$$y' = y, z' = -1$$
$$x' = x$$

$$z' = -1?$$

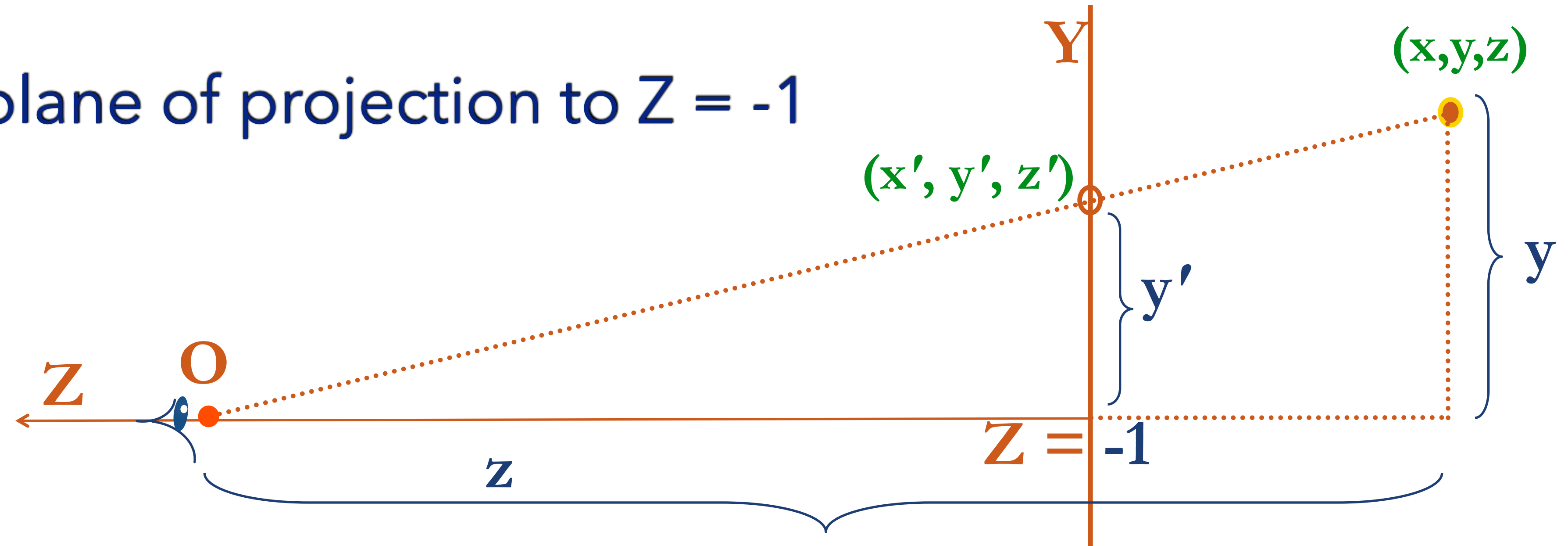
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$





Perspective Projection

- Transform plane of projection to $Z = -1$



$$y' / -1 = y / z \Rightarrow$$

$$y' = -y / z$$

$$x' = -x / z$$

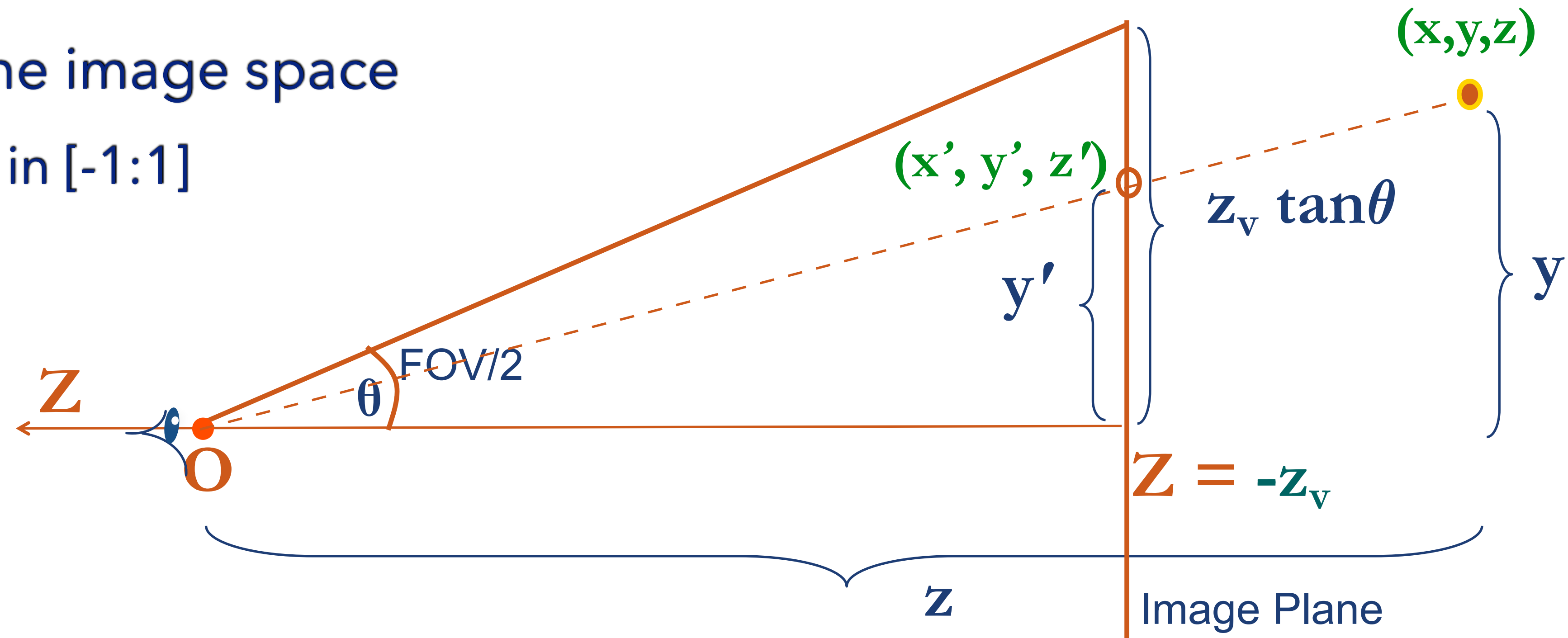
$$z' = -1?$$

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



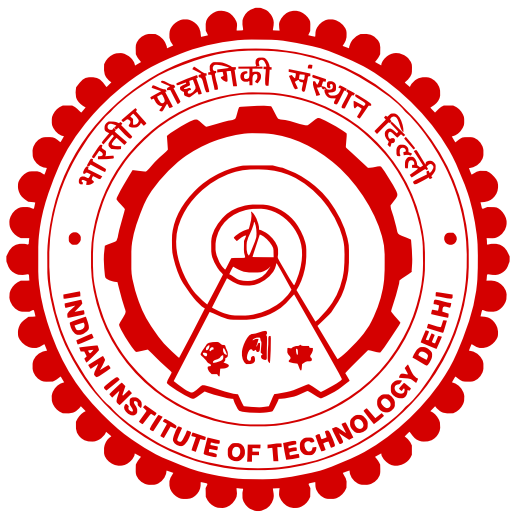
Perspective II

- Normalize the image space
 → Image X, Y lie in $[-1:1]$



$$y' = -z_v y / z$$

$$\text{Normalized } y': y'' = y' / (z_v \tan \theta) = \boxed{-y / z \cot \theta_y} \quad \boxed{x'' = -x / z \cot \theta_x}$$

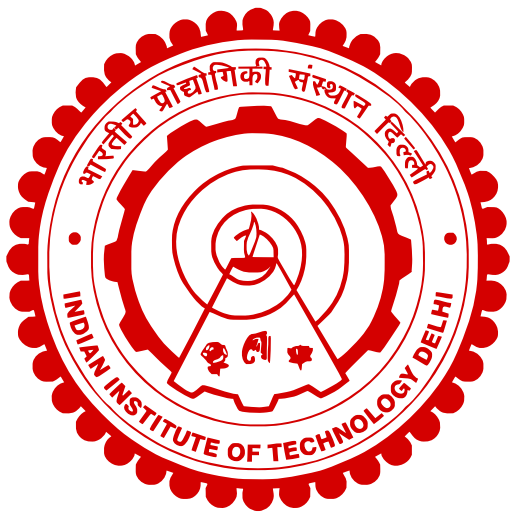


Viewport Transformation

- Further, if we want
 - ➔ X to lie in [0:Screen-Width]
 - ➔ Y to lie in [0:Screen-Height]

$$X_{\text{screen}} = (X_{\text{image}} + 1)/2 * \text{Width}$$

$$Y_{\text{screen}} = (Y_{\text{image}} + 1)/2 * \text{Height}$$

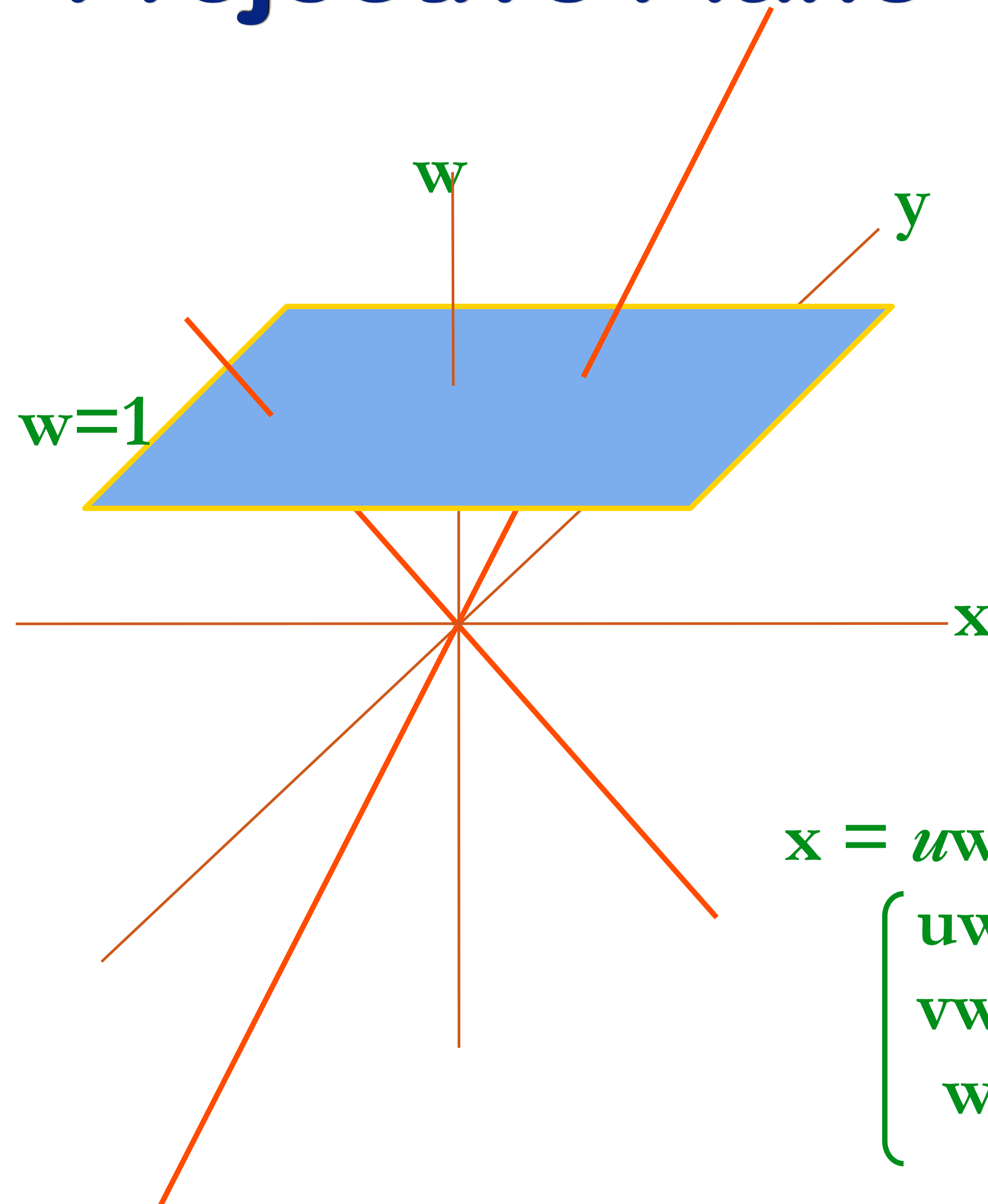


Projective Space

- **n** -dimensional projective space is constructed from an **$n+1$** -dimensional vector space
 - ➔ Where, $aP = P$
 - ➔ $(0,0,0,...)$ is undefined
- The original space is a projection of this **$n+1$** -dimensional vector space
 - ➔ $x, y, z, w = x/w, y/w, z/w, 1 =$ affine space
 - ➔ Except when $w=0$: $x, y, z, 0 = x/z, y/z, 1, 0$
 - Except when $z = 0 \dots$
 - ➔ w is called the homogeneous coordinate



Projective Plane



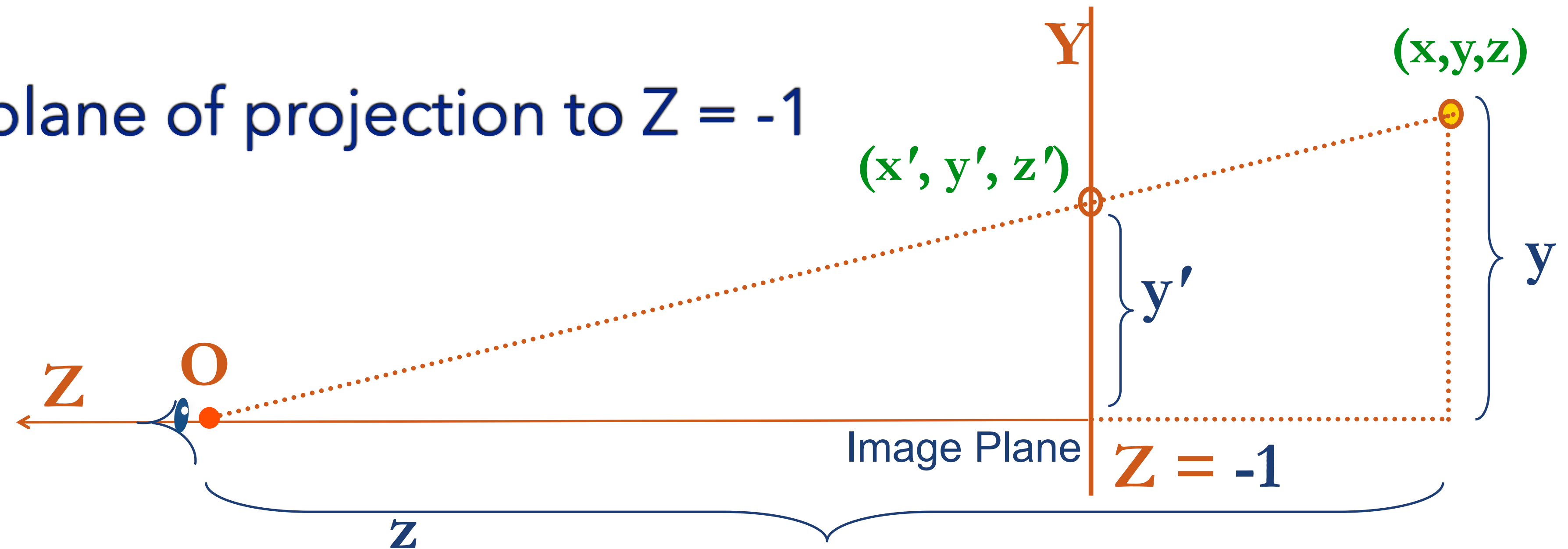
$$x = uw, y = vw, \text{ ie,}$$

$$\begin{bmatrix} uw \\ vw \\ w \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$



Perspective Projection

- Transform plane of projection to $Z = -1$



$$y' / -1 = y / z \Rightarrow$$

$$y' = -y / z$$

$$x' = -x / z$$

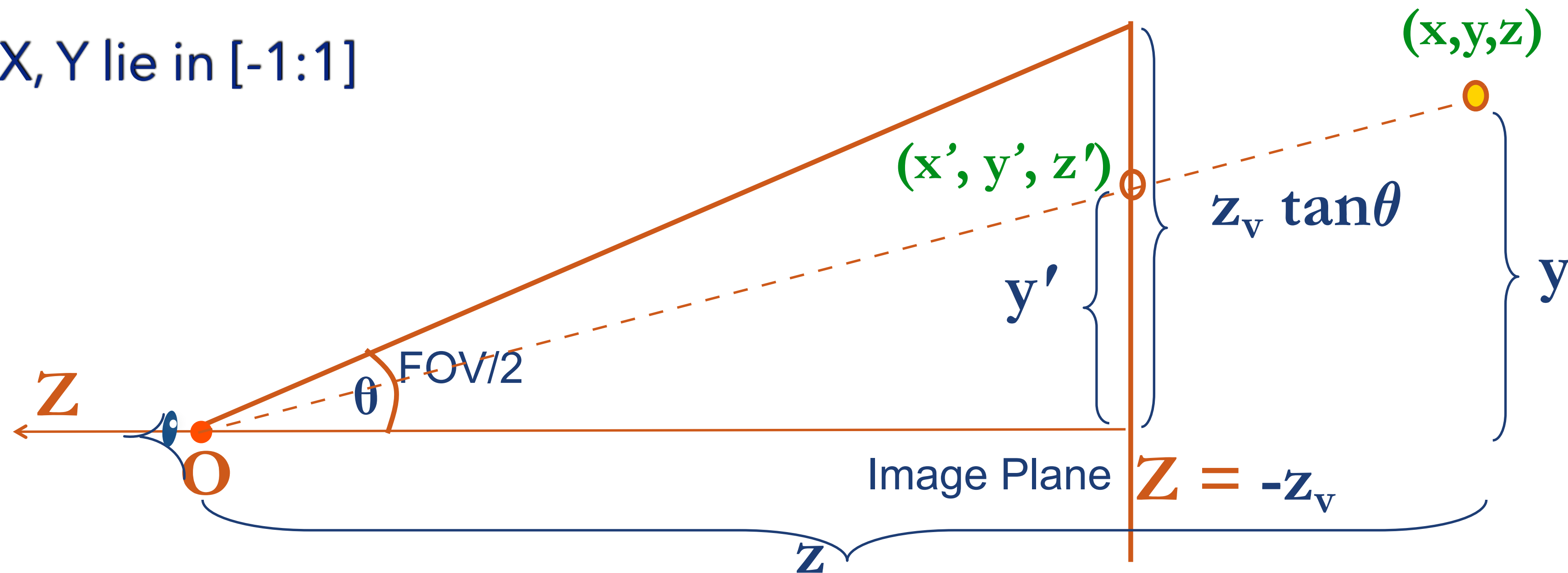
$$z' = -1?$$



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

Perspective

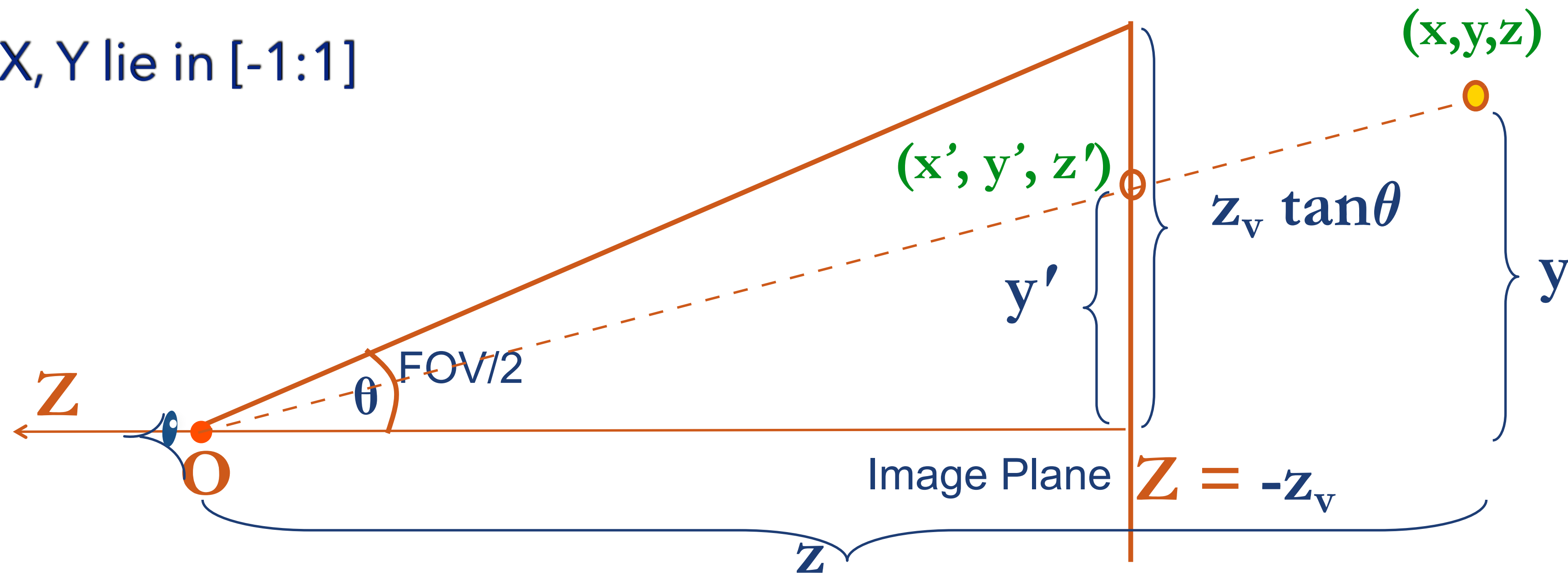
- Normalize the image space
 - Image X, Y lie in $[-1:1]$





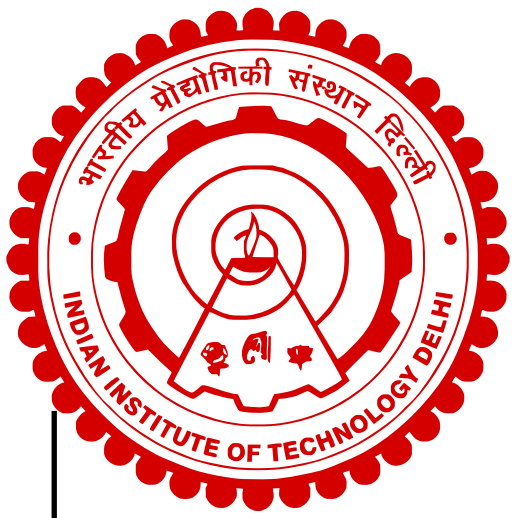
Perspective

- Normalize the image space
 → Image X, Y lie in [-1:1]



Normalized y' : $y'' = y' / (z_v \tan \theta) = \boxed{-y/z \cot \theta_y}$ $\boxed{x'' = -x/z \cot \theta_x}$

Perspective III

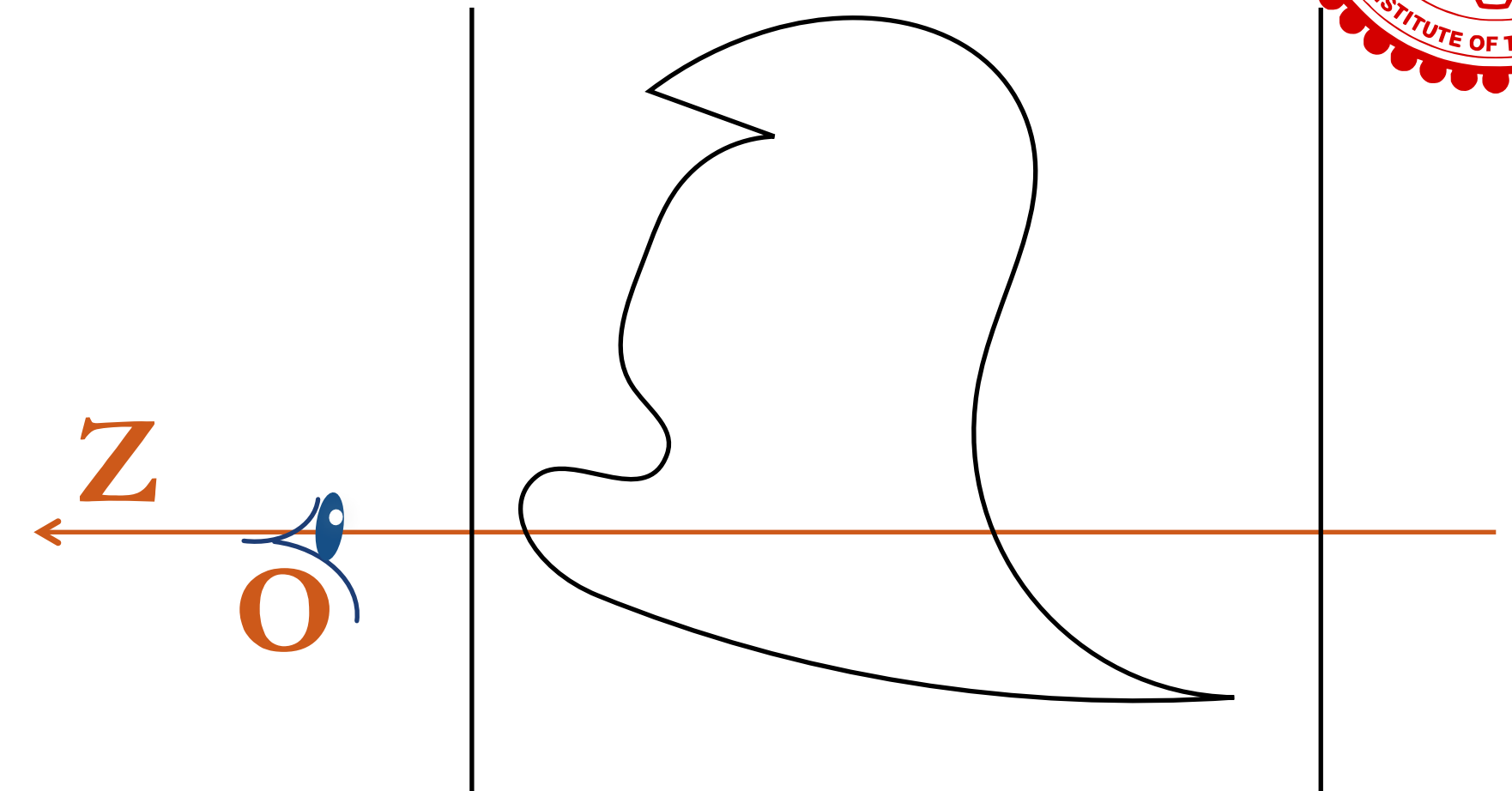


- We also want to not normalize Z

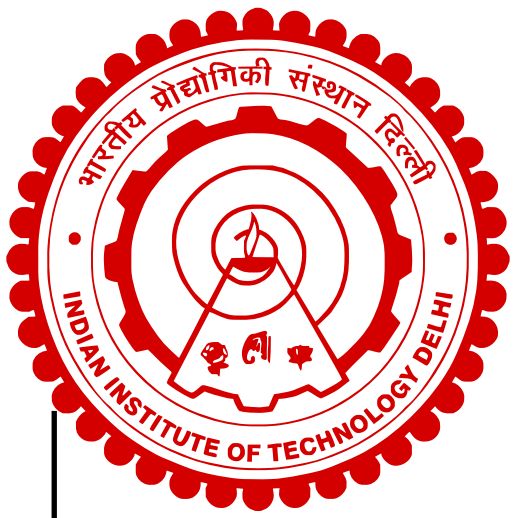
→ Near: $-Z_n \rightarrow -1$

→ Far: $-Z_f \rightarrow 1$

$$\begin{bmatrix} N \\ D \\ C \end{bmatrix} = \begin{bmatrix} \cot \frac{FOV_x}{2} & 0 & 0 & 0 \\ 0 & \cot \frac{FOV_y}{2} & 0 & 0 \\ 0 & 0 & \frac{Z_n + Z_f}{Z_n - Z_f} & \frac{2Z_n Z_f}{Z_n - Z_f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Perspective III

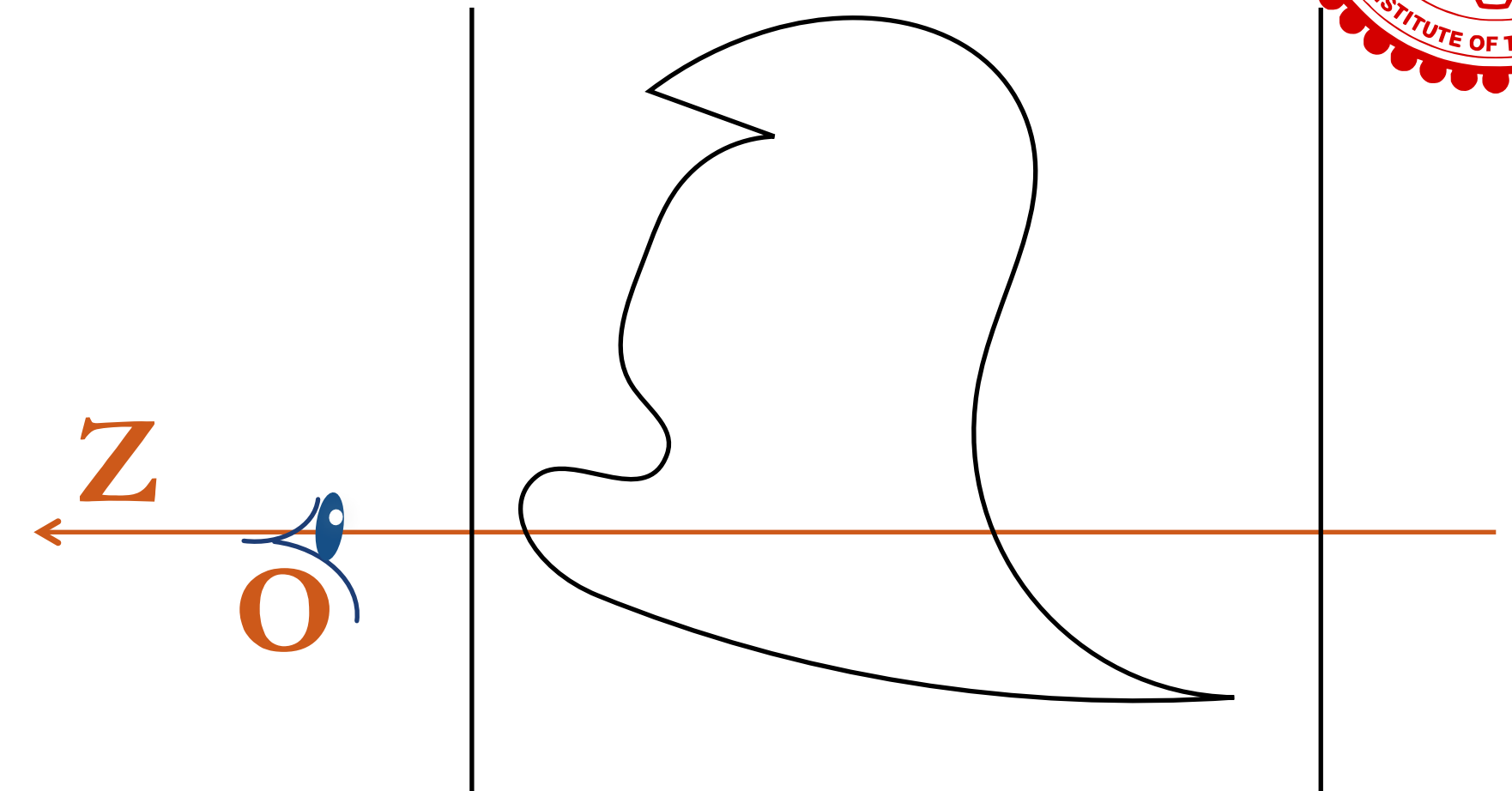


- We also want to not normalize Z

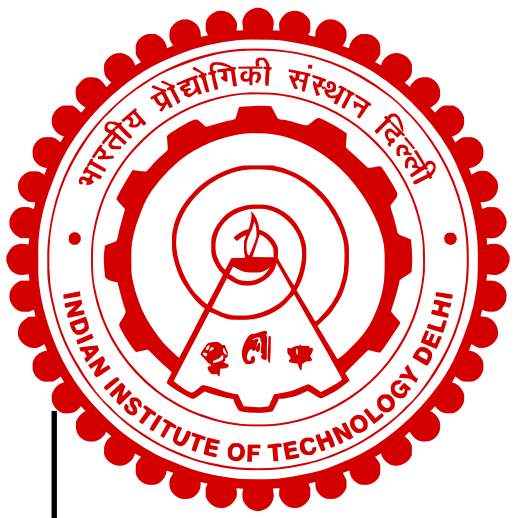
→ Near: $-Z_n \rightarrow -1$

→ Far: $-Z_f \rightarrow 1$

$$\begin{bmatrix} \text{C} \\ \text{L} \\ \text{I} \\ \text{P} \\ \text{C} \\ \text{o} \\ \text{o} \\ \text{r} \\ \text{d} \\ \text{s} \end{bmatrix} = \begin{bmatrix} \cot \frac{FOV_x}{2} & 0 & 0 & 0 \\ 0 & \cot \frac{FOV_y}{2} & 0 & 0 \\ 0 & 0 & \frac{Z_n + Z_f}{Z_n - Z_f} & \frac{2Z_n Z_f}{Z_n - Z_f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Perspective III

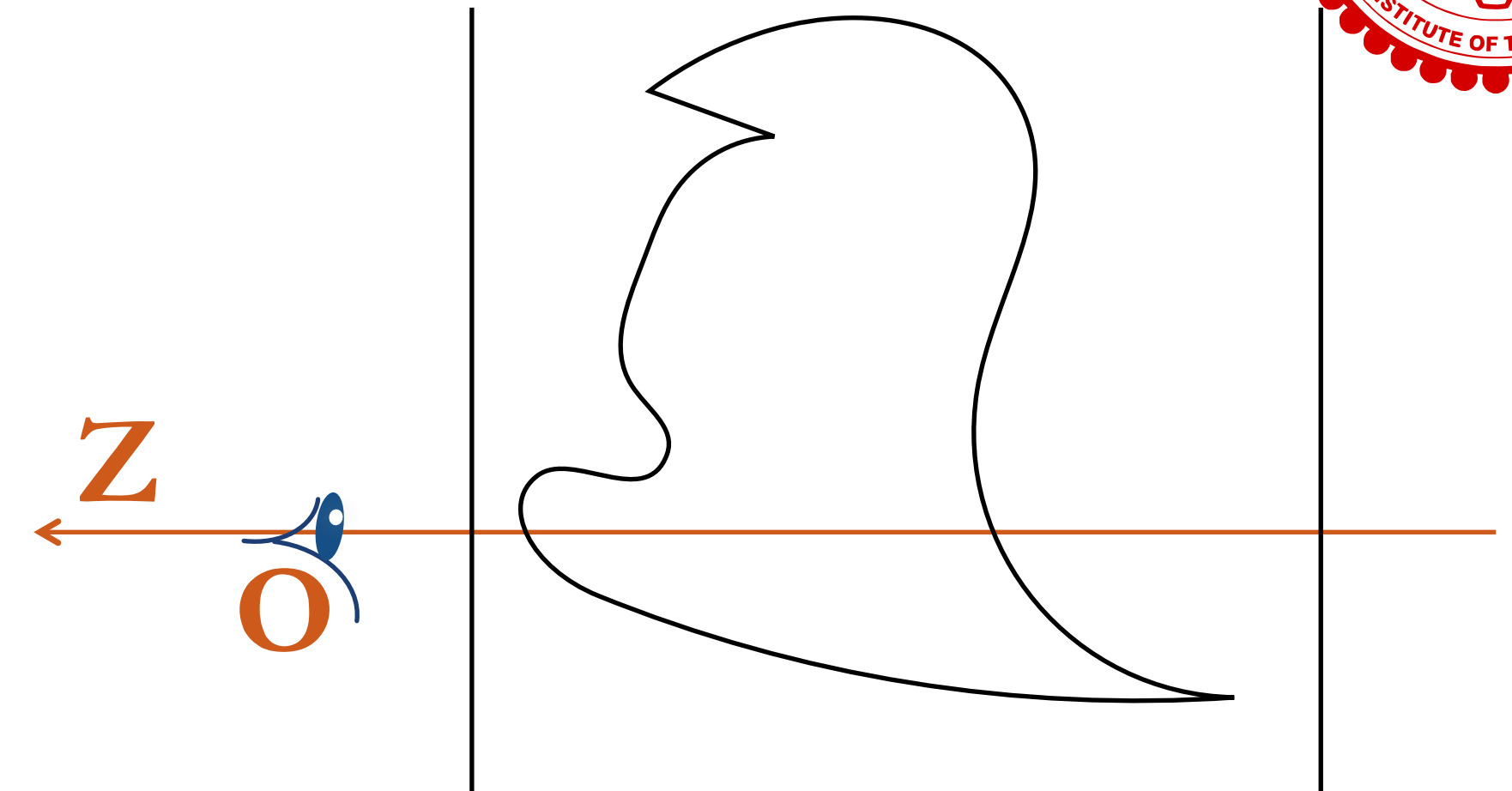


- We also want to not normalize Z

→ Near: $-Z_n \rightarrow -1$

→ Far: $-Z_f \rightarrow 1$

$$\begin{bmatrix} \text{C} \\ \text{L} \\ \text{I} \\ \text{P} \\ \text{C} \\ \text{o} \\ \text{o} \\ \text{r} \\ \text{d} \\ \text{s} \end{bmatrix} = \begin{bmatrix} \cot \frac{FOV_x}{2} & 0 & 0 & 0 \\ 0 & \cot \frac{FOV_y}{2} & 0 & 0 \\ 0 & 0 & \frac{Z_n + Z_f}{Z_n - Z_f} & \frac{2Z_n Z_f}{Z_n - Z_f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$





World to Camera

- Centre of projection/Camera position

→ O

- Reference point/Lookat point: L

→ $V3 = (O-L)/|O-L|$

- Camera orientation/Up vector

→ $T = U/|U|$

→ $V1 = T \times V3$

→ $V2 = V3 \times V1$

M: Model $\rightarrow$ World

V: World $\rightarrow$ Camera

P: Camera $\rightarrow$ Normalized Screen

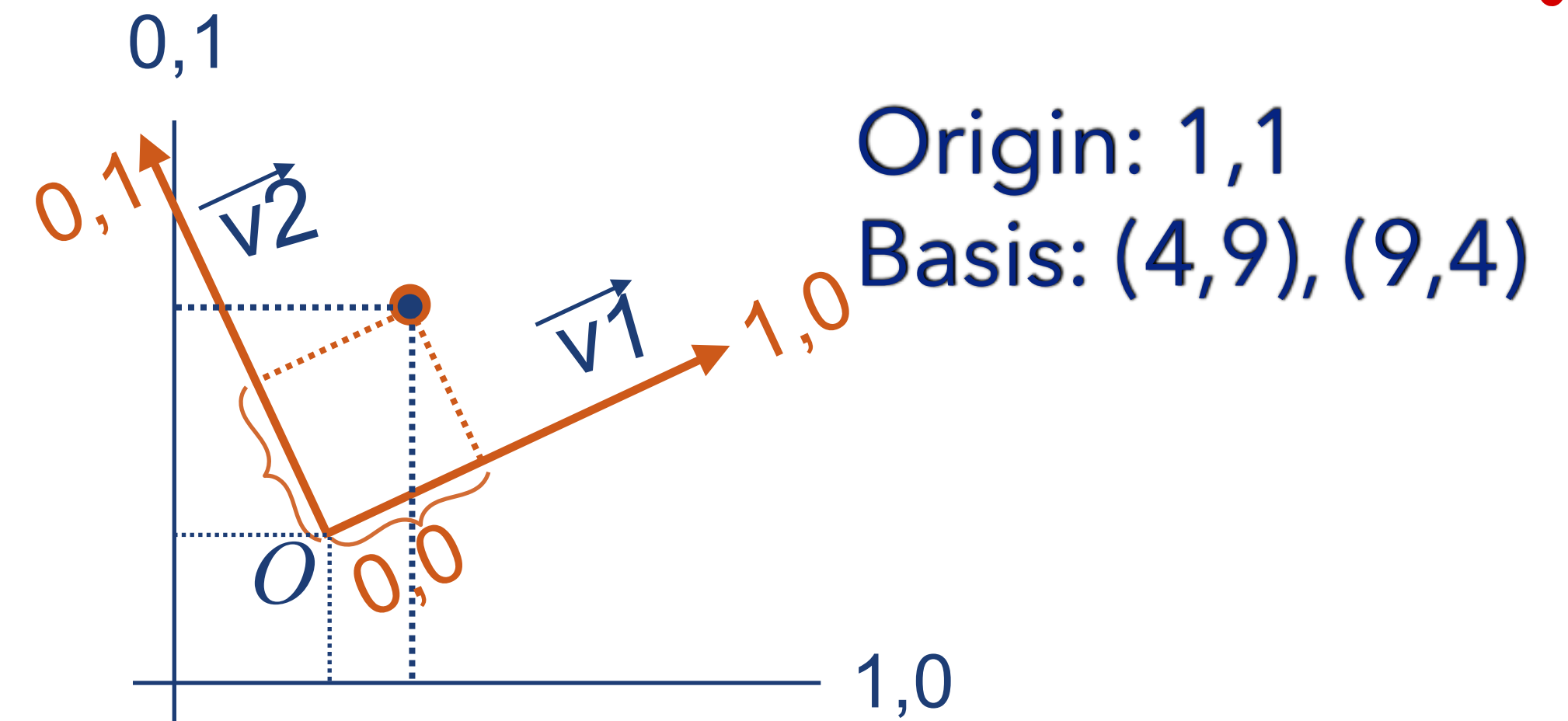
ModelView, VM: Model $\rightarrow$ Camera?

PVM: Model $\rightarrow$ Normalized Screen?



Change of Basis

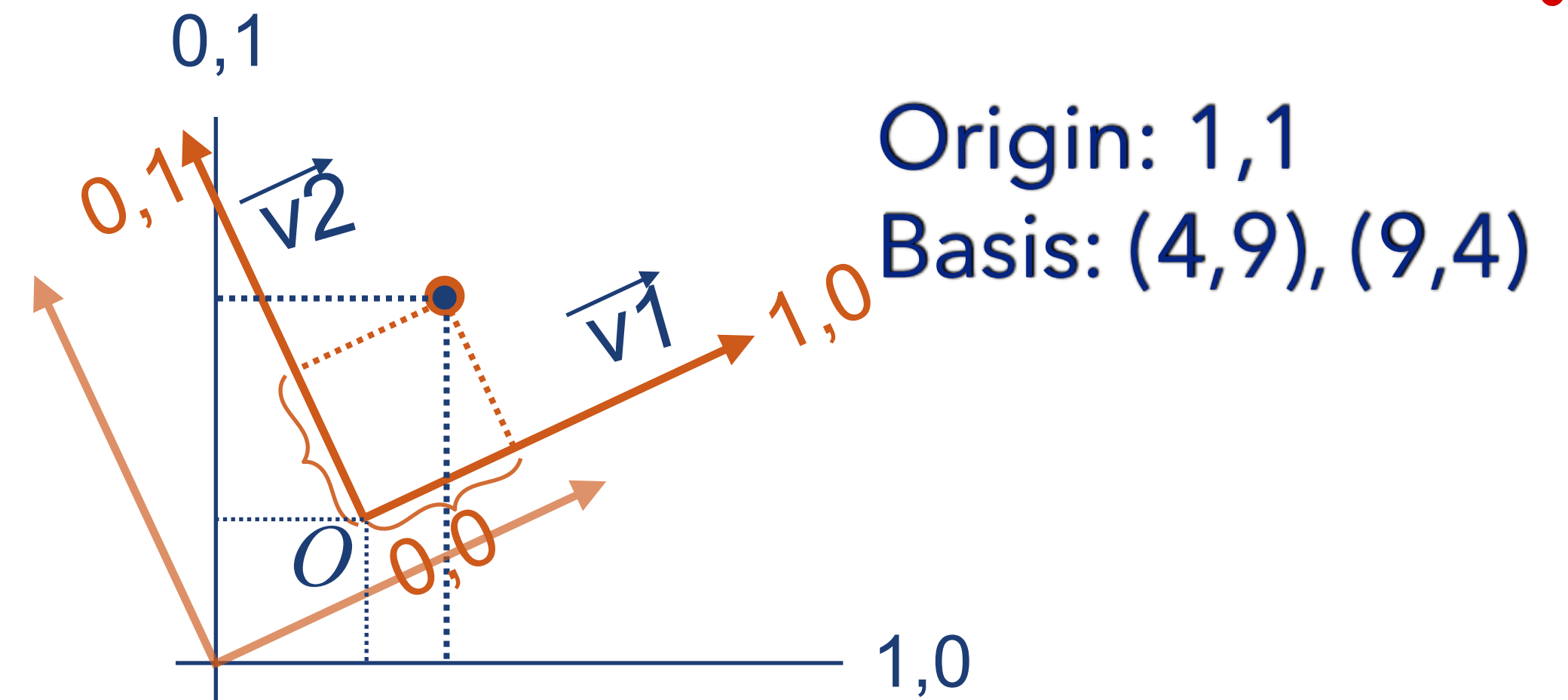
- New basis:
 - Orthonormal axes: $V1, V2, V3$
 - Origin: O





Change of Basis

- New basis:
 - ➔ Orthonormal axes: $V1, V2, V3$
 - ➔ Origin: O

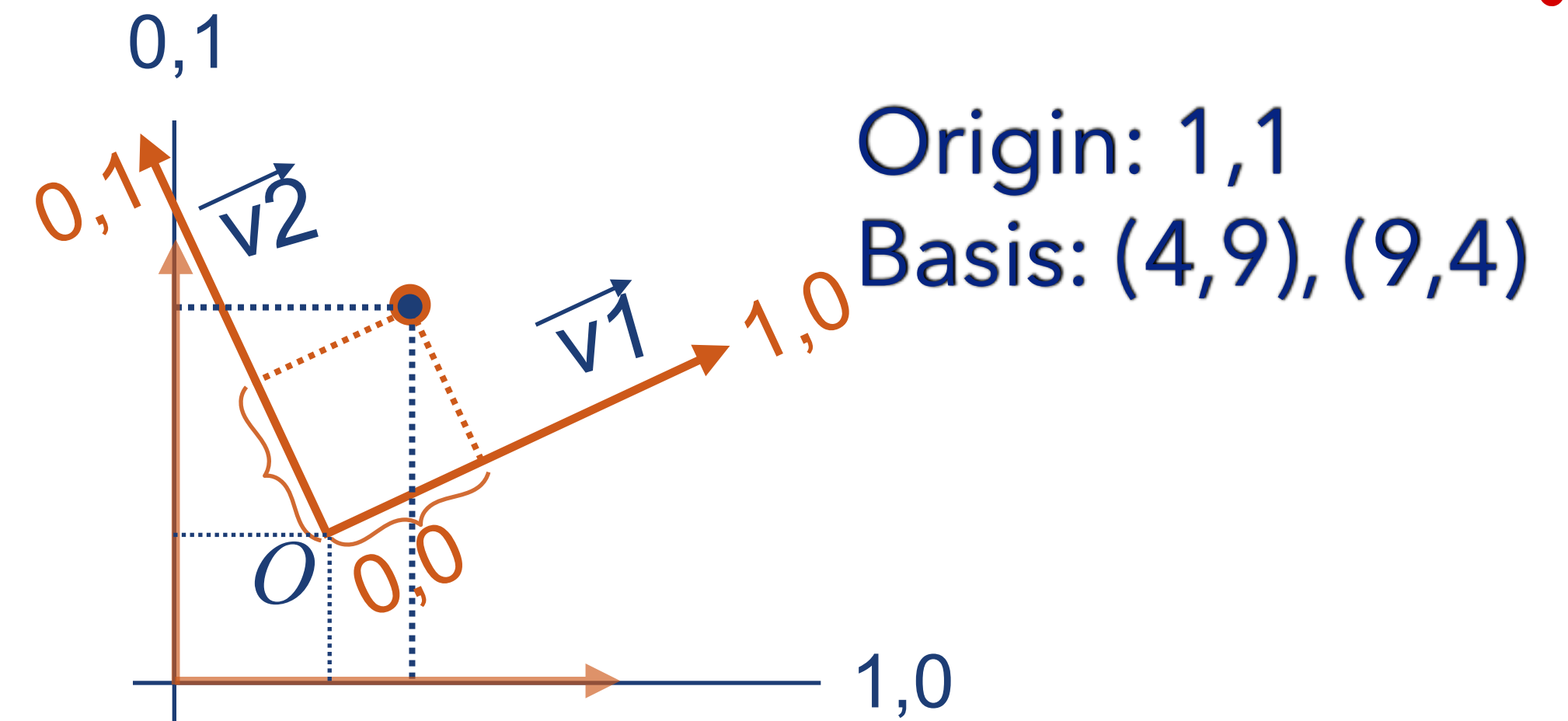


$$\begin{bmatrix} 1 & 0 & 0 & -O_x \\ 0 & 1 & 0 & -O_y \\ 0 & 0 & 1 & -O_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Change of Basis

- New basis:
 - ➔ Orthonormal axes: $V1, V2, V3$
 - ➔ Origin: O

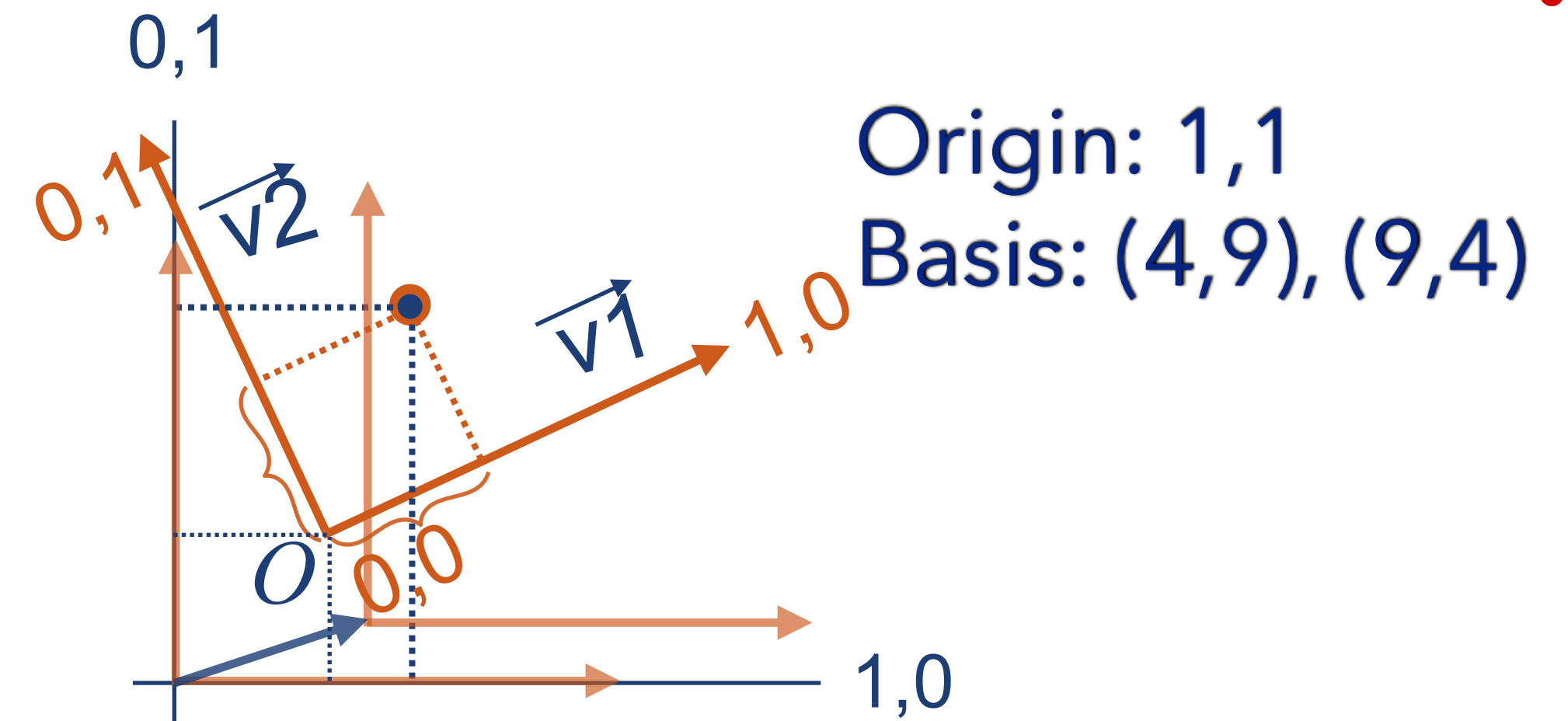


$$\begin{bmatrix} - & - & V1 & - & - & 0 \\ - & - & V2 & - & - & 0 \\ - & - & V3 & - & - & 0 \\ 0 & 0 & 0 & & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -O_x \\ 0 & 1 & 0 & -O_y \\ 0 & 0 & 1 & -O_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Change of Basis

- New basis:
 - ➔ Orthonormal axes: $V1, V2, V3$
 - ➔ Origin: O

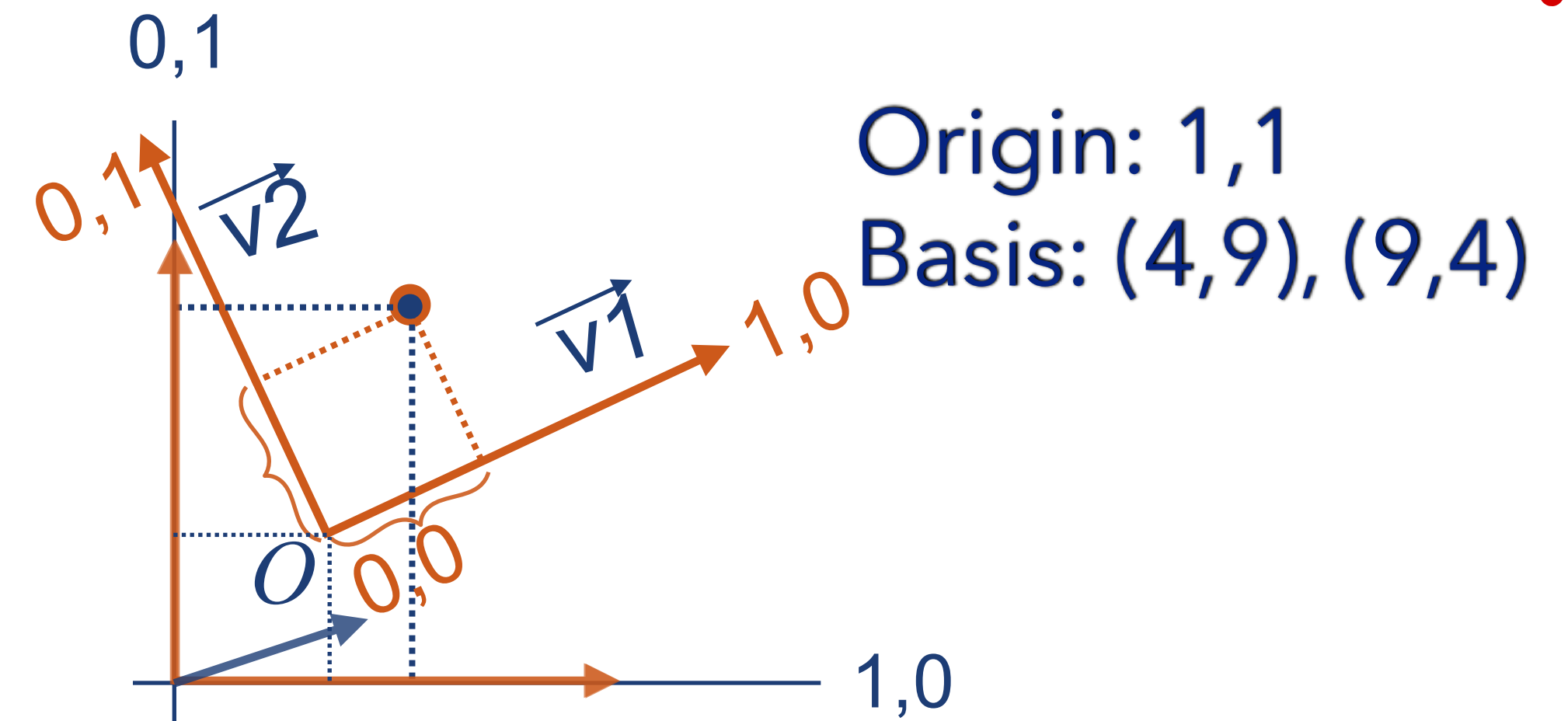


$$\begin{bmatrix} - & - & V1 & - & - & 0 \\ - & - & V2 & - & - & 0 \\ - & - & V3 & - & - & 0 \\ 0 & 0 & 0 & & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -O_x \\ 0 & 1 & 0 & -O_y \\ 0 & 0 & 1 & -O_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Change of Basis

- New basis:
 - ➔ Orthonormal axes: $V1, V2, V3$
 - ➔ Origin: O



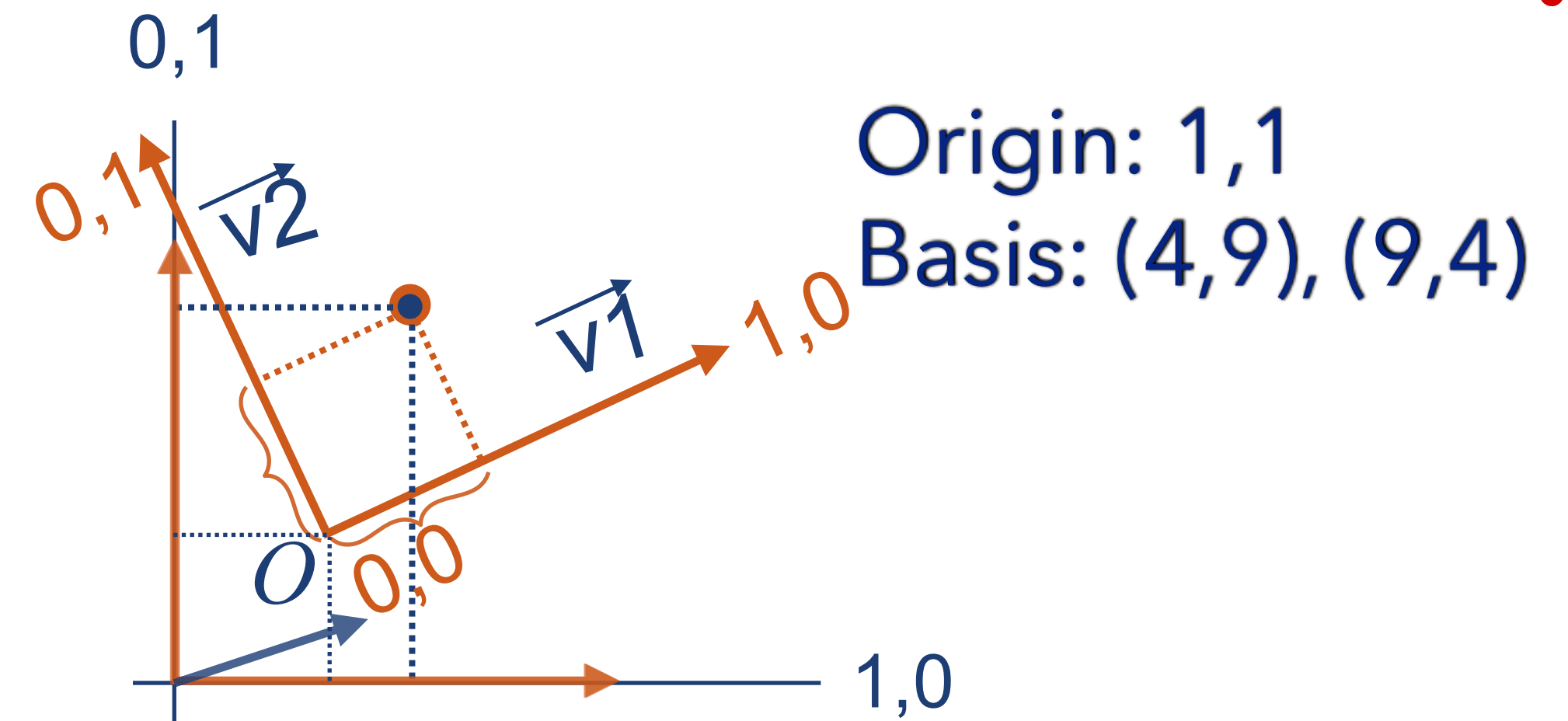
$$\begin{bmatrix} 1 & 0 & 0 & -O_x \\ 0 & 1 & 0 & -O_y \\ 0 & 0 & 1 & -O_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} - & - & V1 & - & - & 0 \\ - & - & V2 & - & - & 0 \\ - & - & V3 & - & - & 0 \\ 0 & 0 & 0 & & & 1 \end{bmatrix}$$



Change of Basis

- New basis:
 - ➔ Orthonormal axes: $V1, V2, V3$
 - ➔ Origin: O

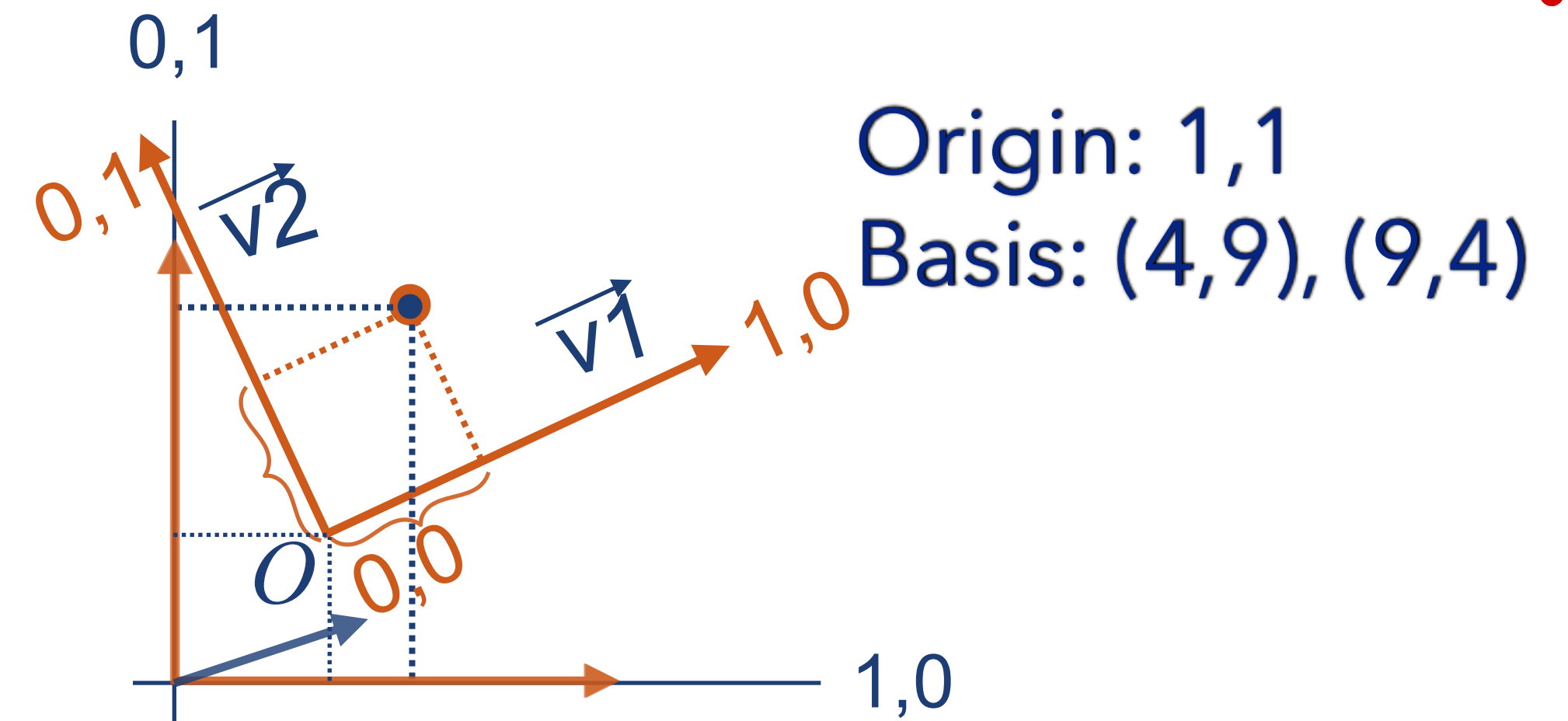


$$\begin{bmatrix} 1 & 0 & 0 & -V1 \cdot O \\ 0 & 1 & 0 & -V2 \cdot O \\ 0 & 0 & 1 & -V3 \cdot O \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} - & - & V1 & - & - & 0 \\ - & - & V2 & - & - & 0 \\ - & - & V3 & - & - & 0 \\ 0 & 0 & 0 & & & 1 \end{bmatrix}$$



Change of Basis

- New basis:
 - Orthonormal axes: $V1, V2, V3$
 - Origin: O

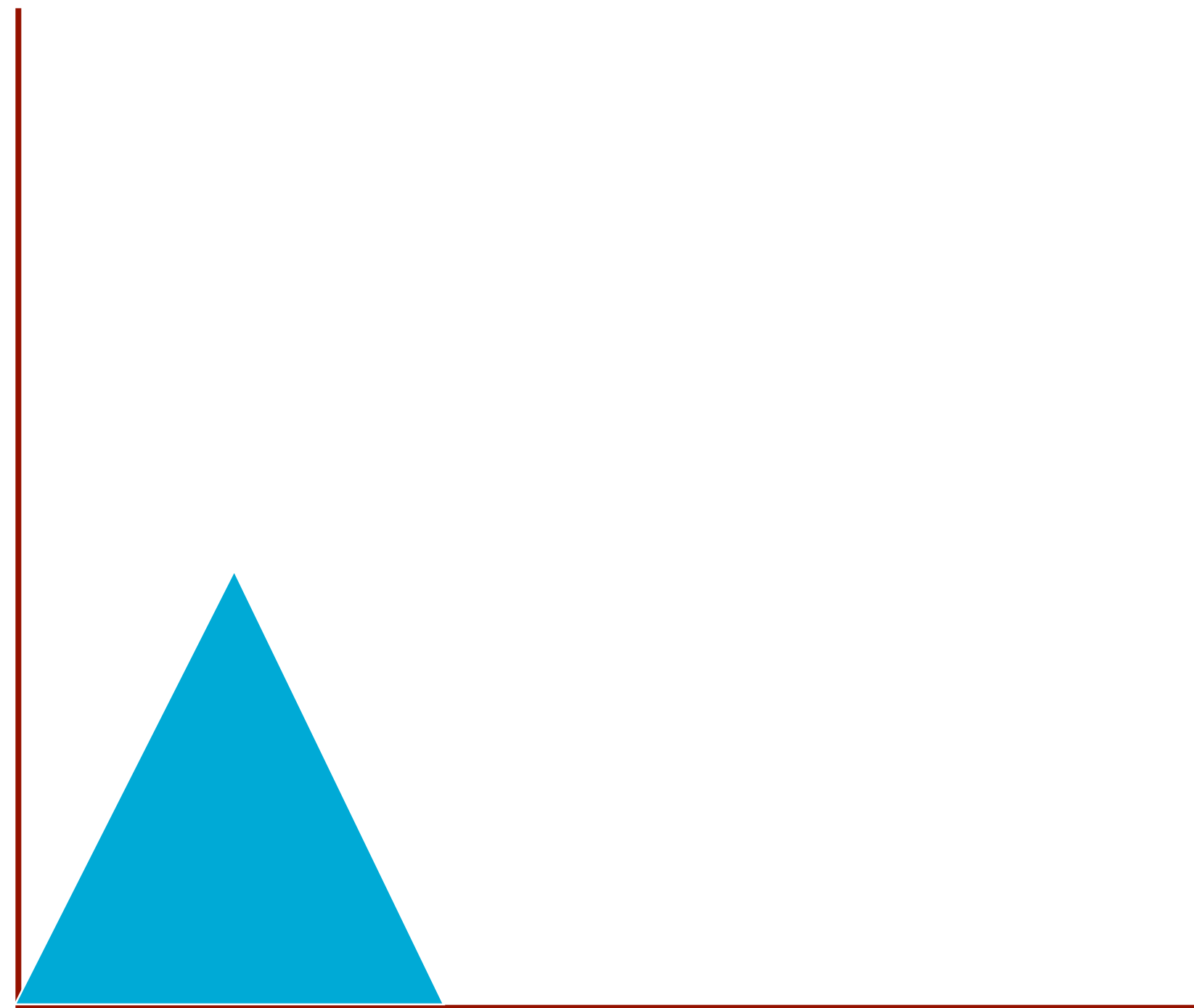


$$\begin{bmatrix} 1 & 0 & 0 & -V1 \cdot O \\ 0 & 1 & 0 & -V2 \cdot O \\ 0 & 0 & 1 & -V3 \cdot O \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} - & - & V1 & - & - & 0 \\ - & - & V2 & - & - & 0 \\ - & - & V3 & - & - & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} - & V1 & - & -V1 \cdot O \\ - & V2 & - & -V2 \cdot O \\ - & V3 & - & -V3 \cdot O \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composition: Fixed frame



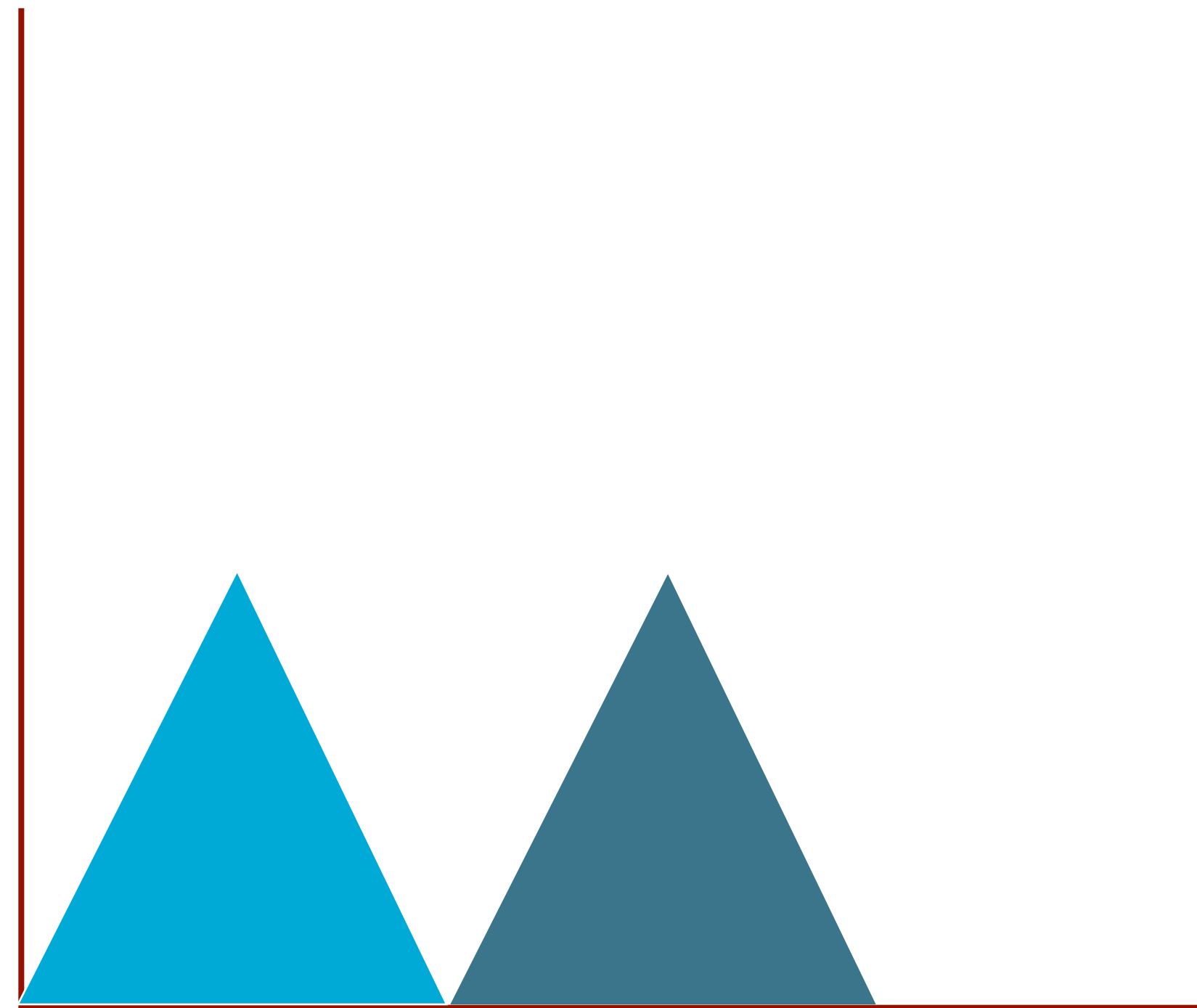
- $T(1,0)$
- $R(90)$
- RT_p





Composition: Fixed frame

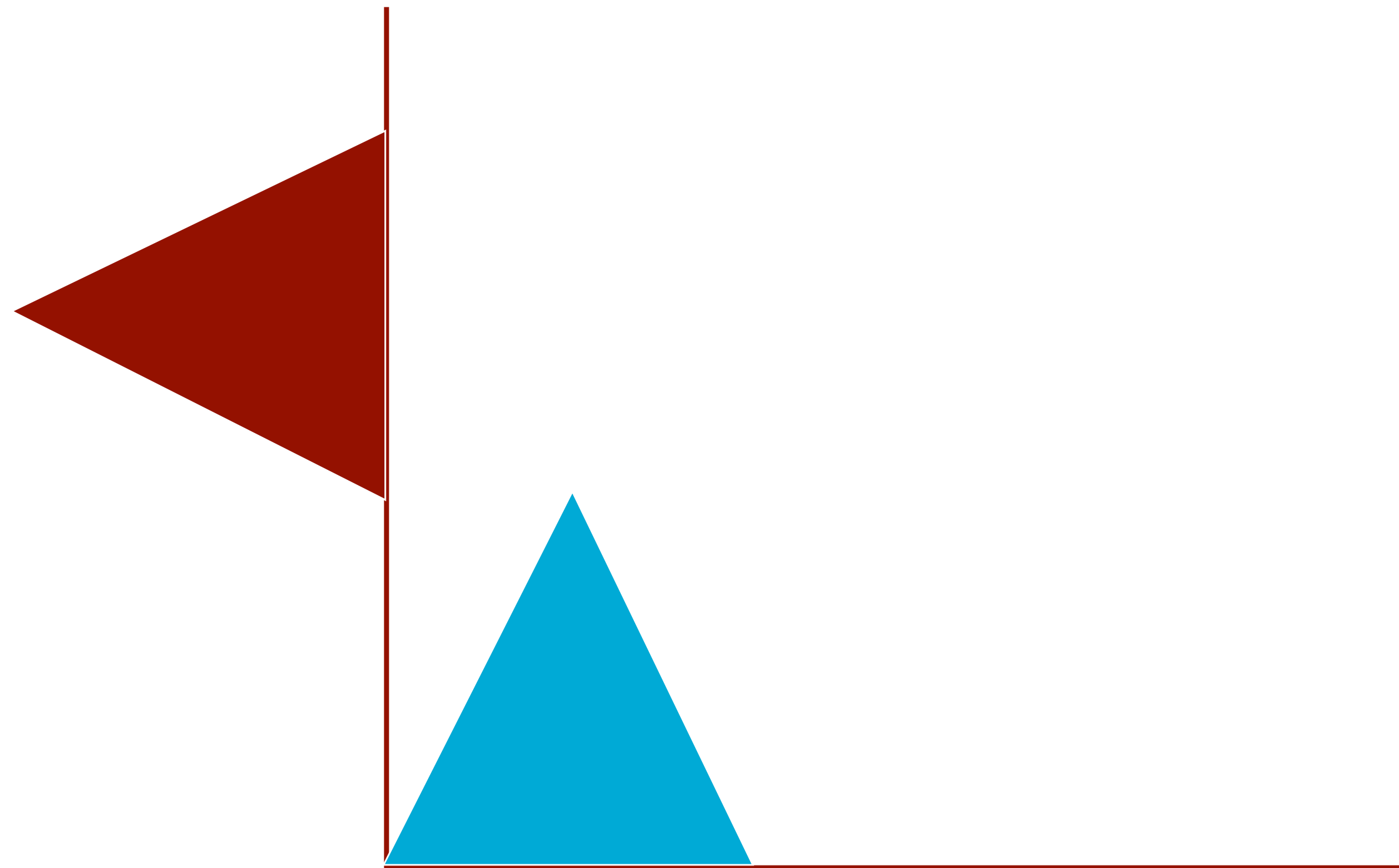
- $T(1,0)$ ←
- $R(90)$
- $R T_p$





Composition: Fixed frame

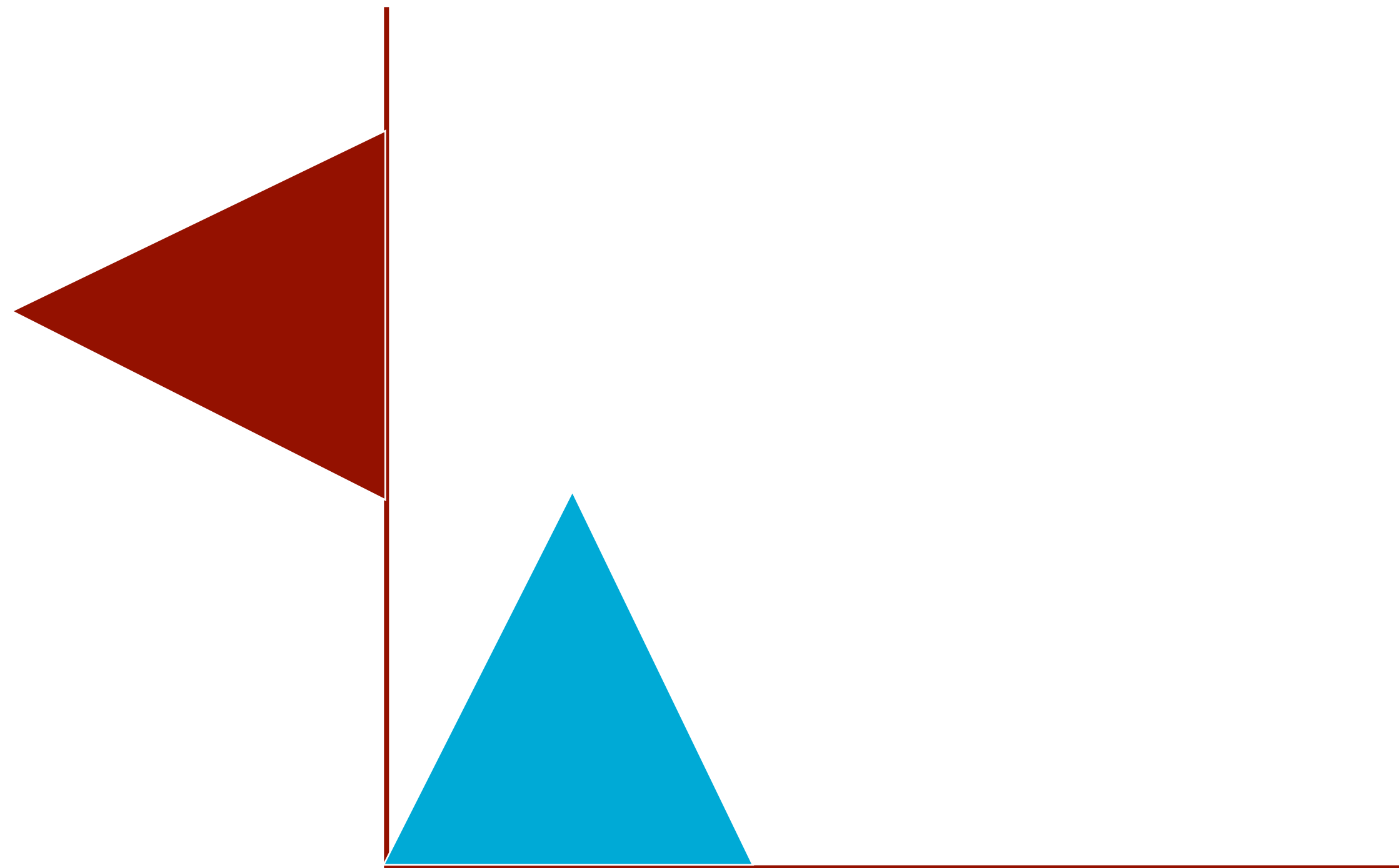
- $T(1,0)$
- $R(90)$ ←
- $R T_p$





Composition: Fixed frame

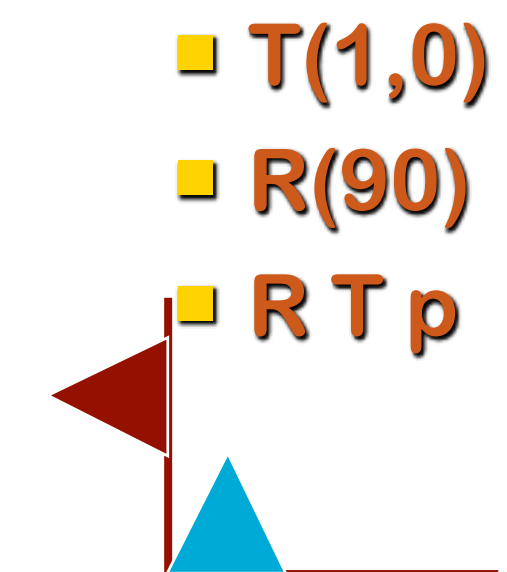
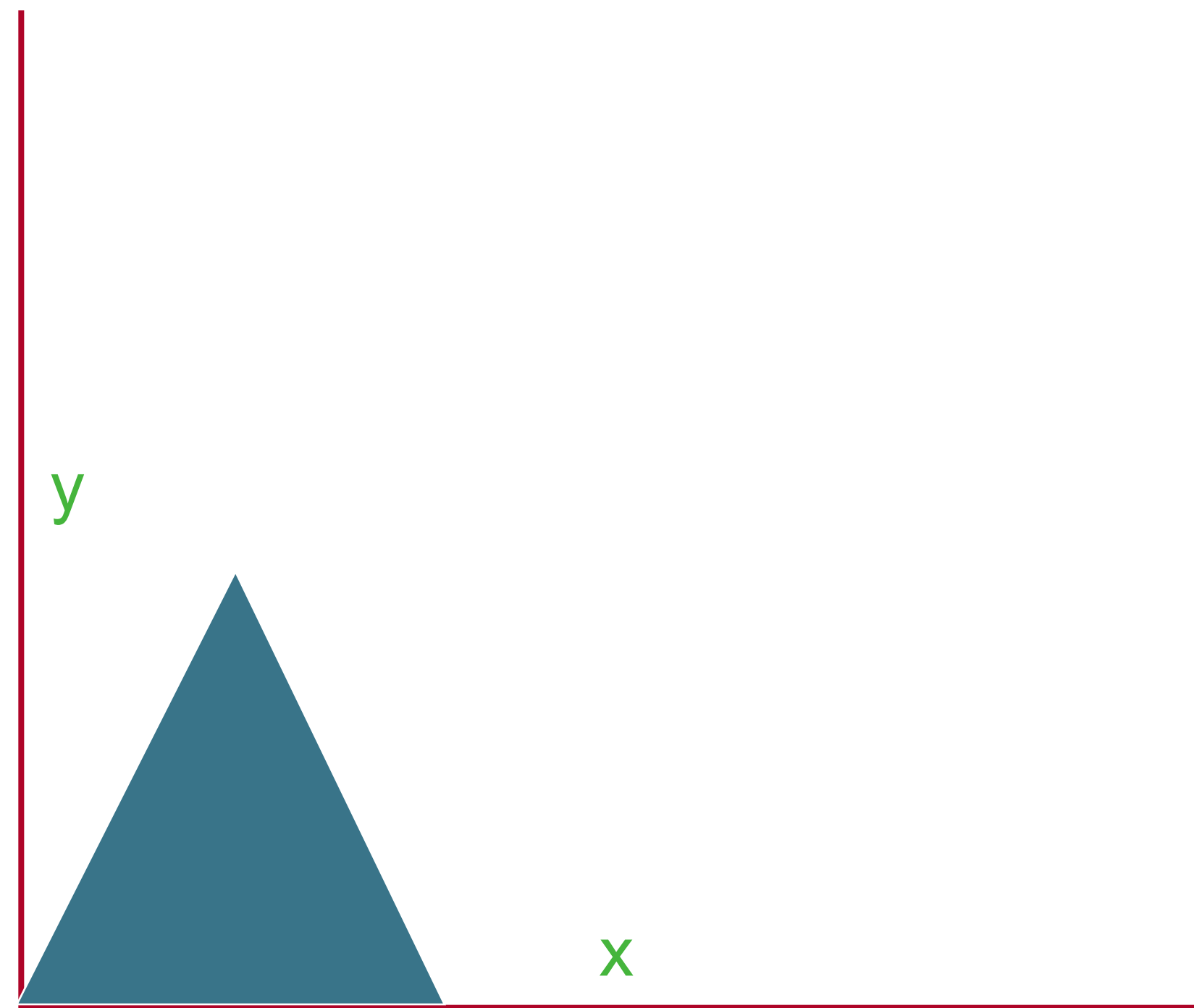
- $T(1,0)$
- $R(90)$ ←
- $R T_p$ ←





Composition: Local frame

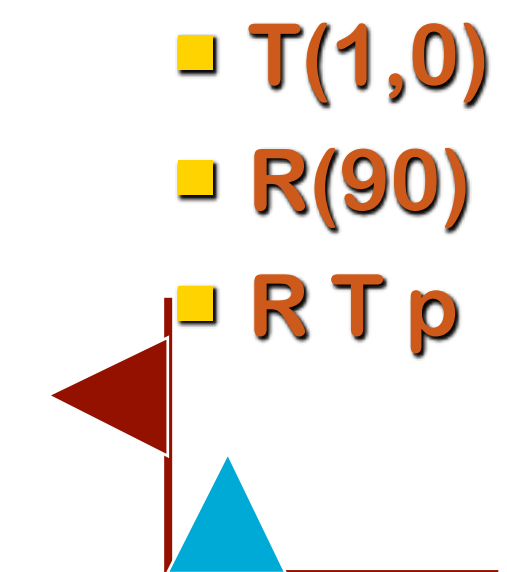
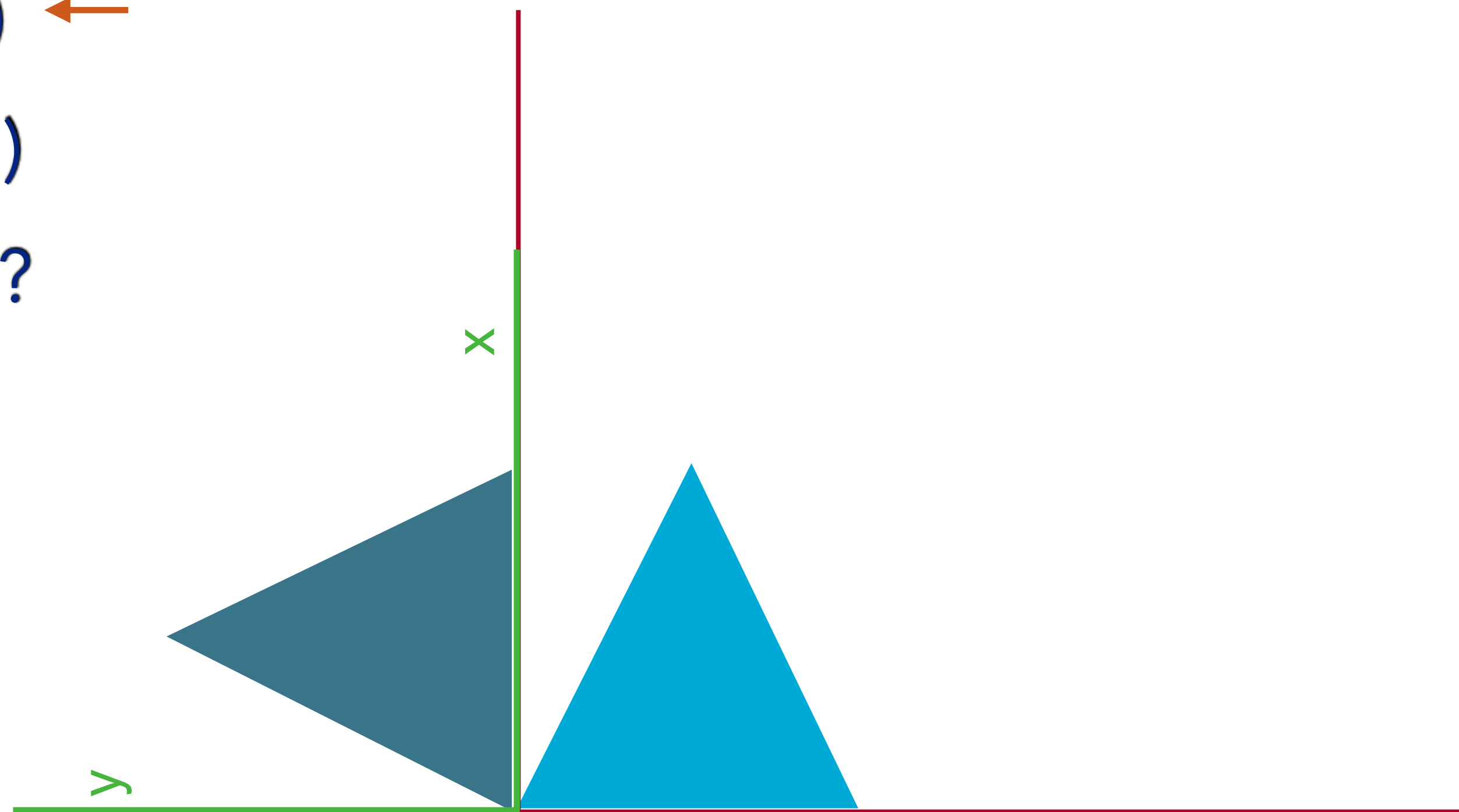
- $R(90)$
- $T(1,0)$
- $R T p?$





Composition: Local frame

- $R(90)$ ←
- $T(1,0)$
- $R T p?$



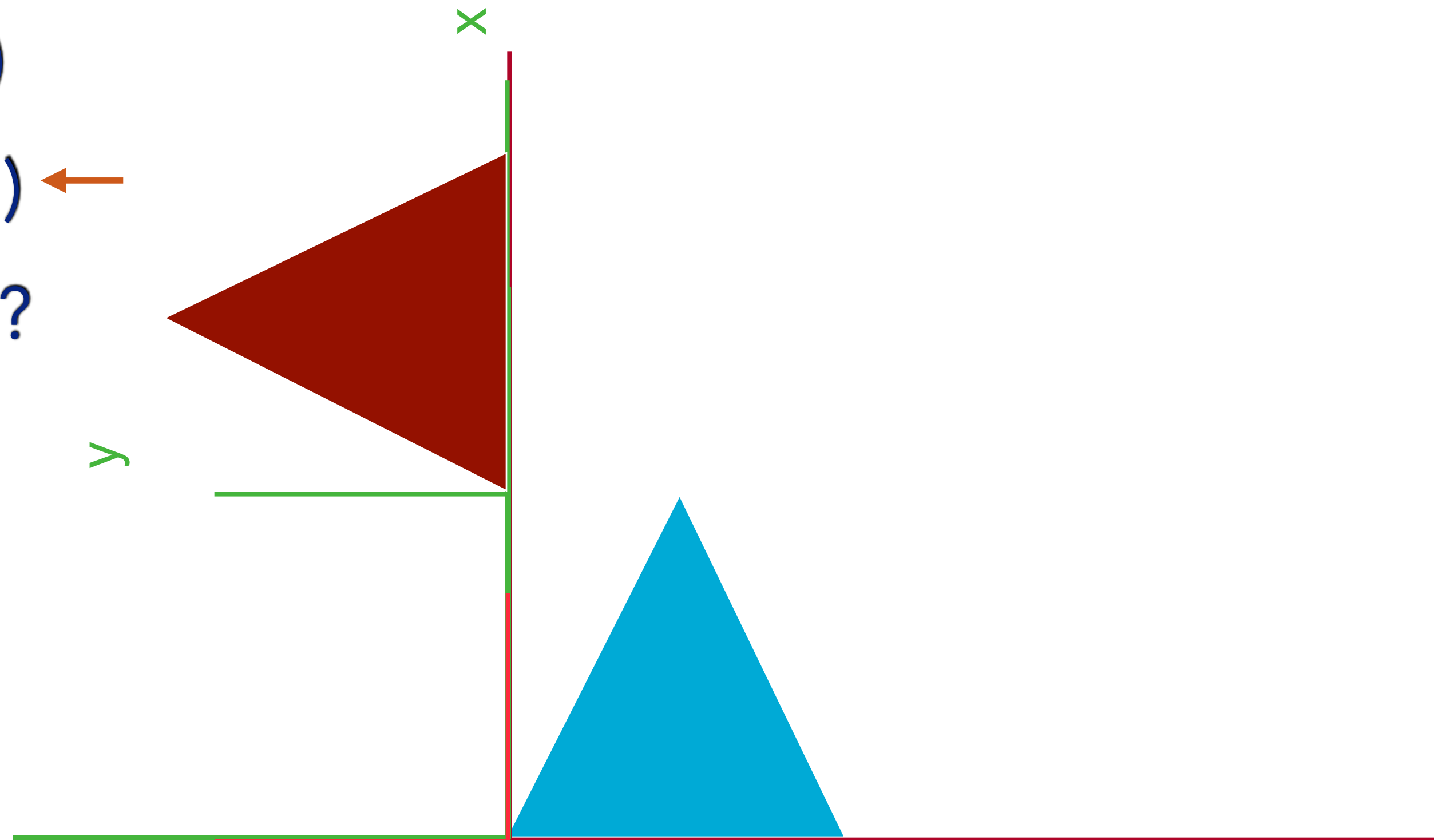


Composition: Local frame

- $R(90)$

- $T(1,0)$ ←

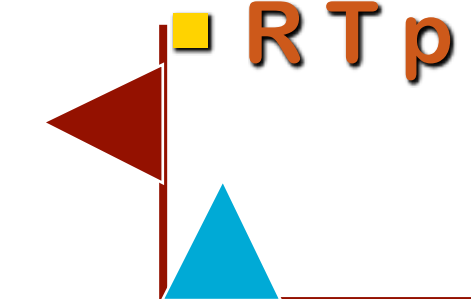
- $R T p?$ →



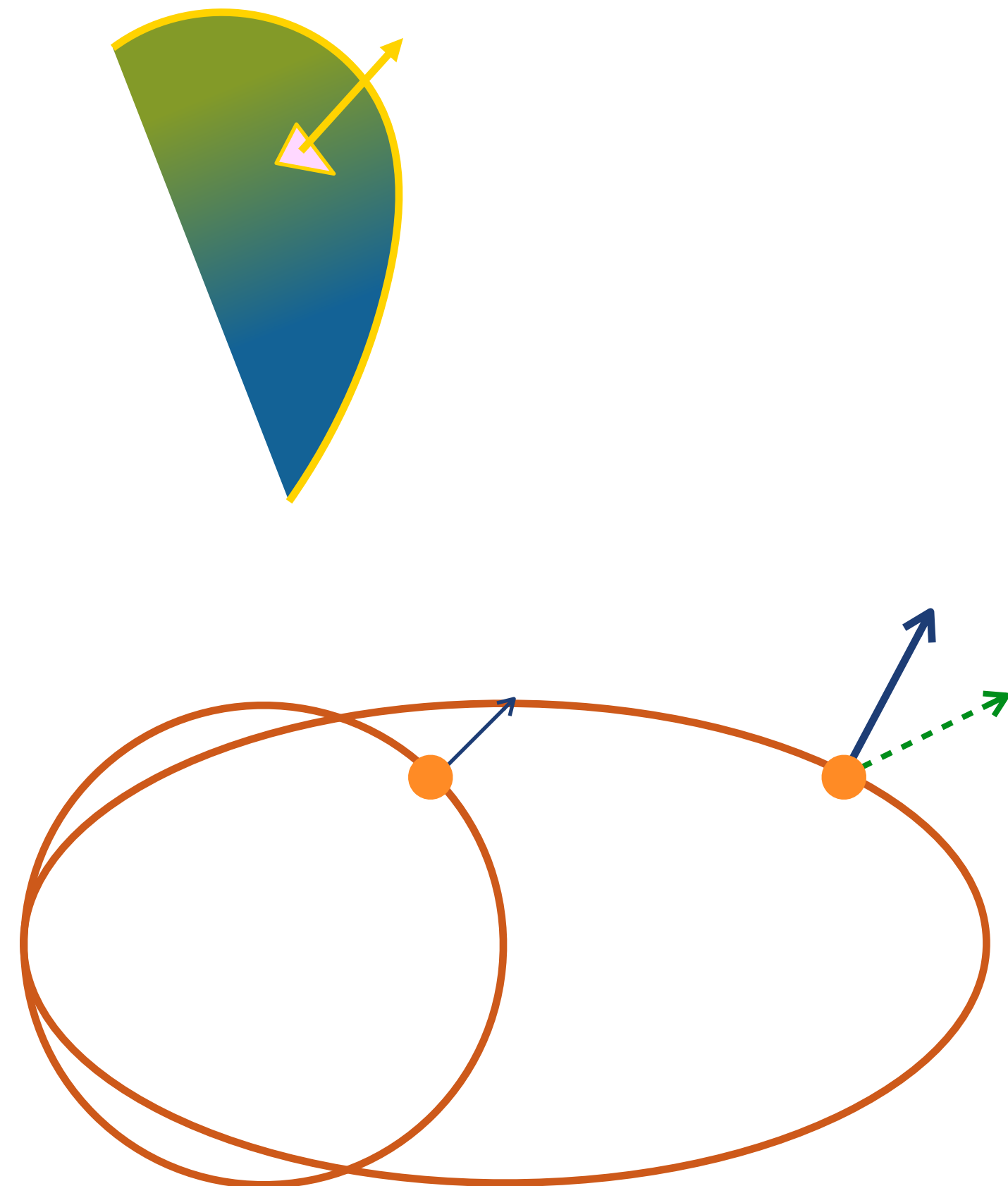
■ $T(1,0)$

■ $R(90)$

■ $R T p$

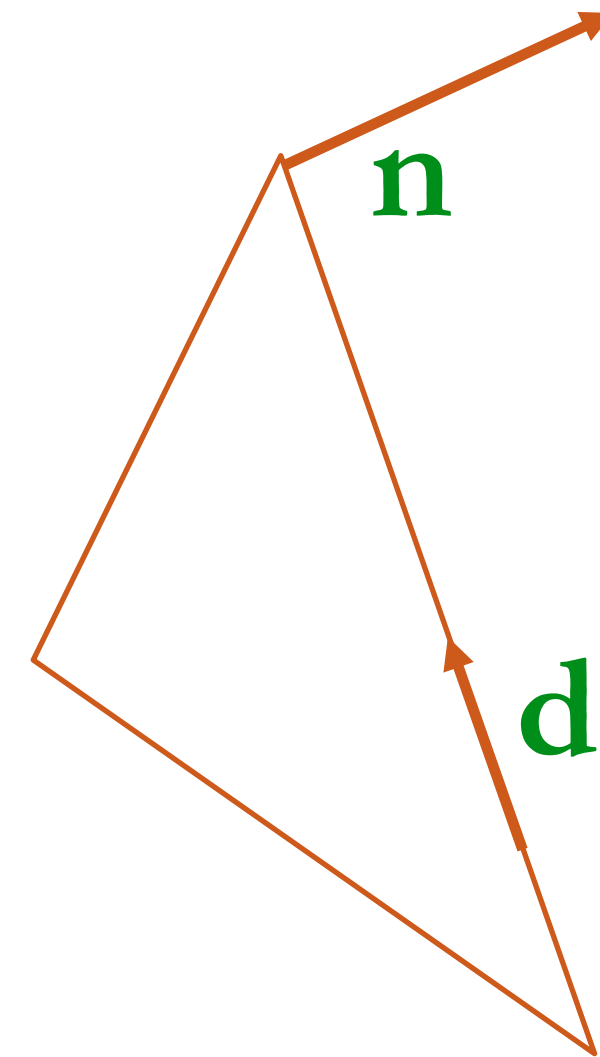
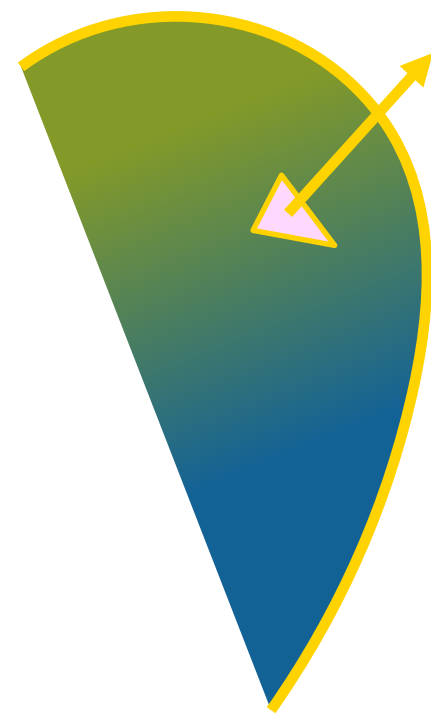


Transformation of Normals





Transformation of Normals



$$\vec{n} \cdot \vec{d} = 0$$

$$n^T d = 0$$

$$\Rightarrow n'^T M d \text{ must be } 0$$

$$\Rightarrow (\underline{M'} n)^T M d = 0$$

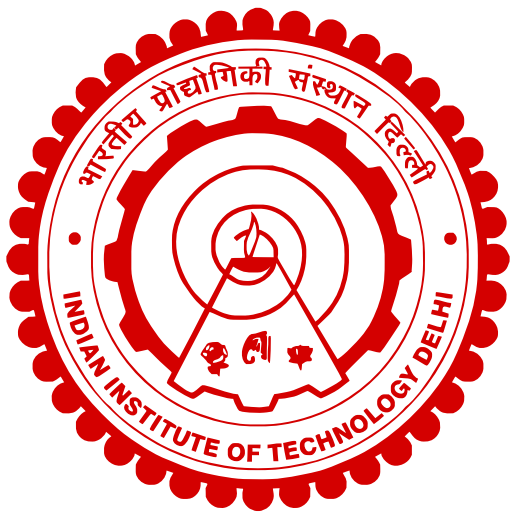
$$\Rightarrow n^T \underline{M'^T} M d = 0$$

$$\Rightarrow \underline{M'^T} M = k I$$

$$\Rightarrow \underline{M'^T} = k M^{-1}$$

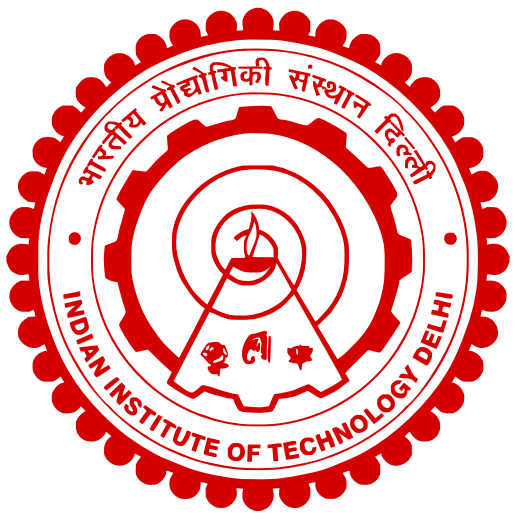
$$\Rightarrow \underline{M'} = (M^{-1})^T$$

(Choose $k = 1$)



Transformation Recap

- General transformation =
 - ➔ Composition of known special transformations
- Transformation is equivalent to an inverse change of basis
- Be mindful of space in which
 - ➔ the transformation is specified
 - ➔ the vertex/vector is specified
- $[M_1 \ M_2 \ M_3 \ \dots \ M_k] V =$
 - ➔ M_k then M_{k-1} until M_1 in world-coordinates
 - ➔ M_1 , then M_2 , until M_k in object-coordinates

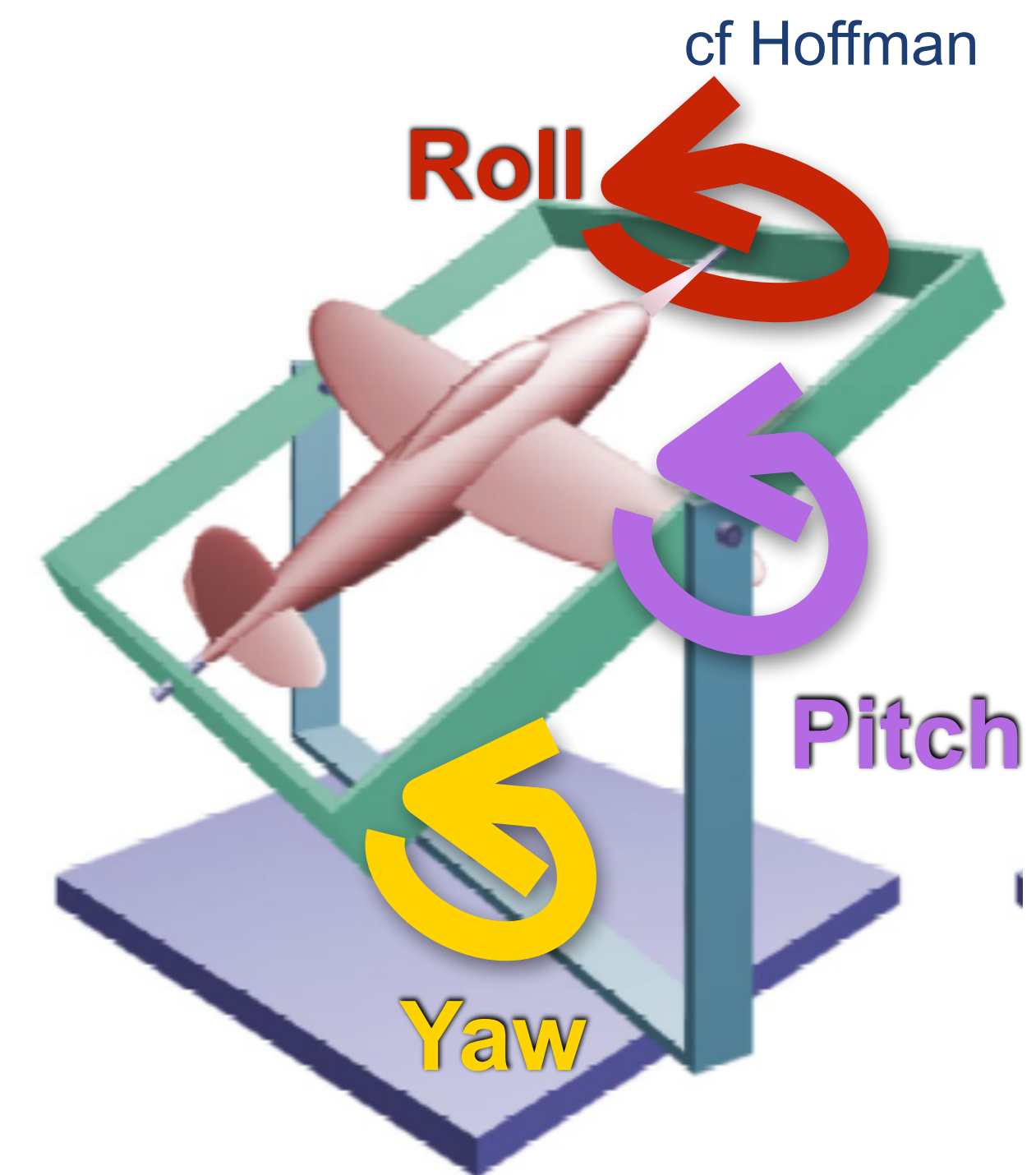


3D Interaction

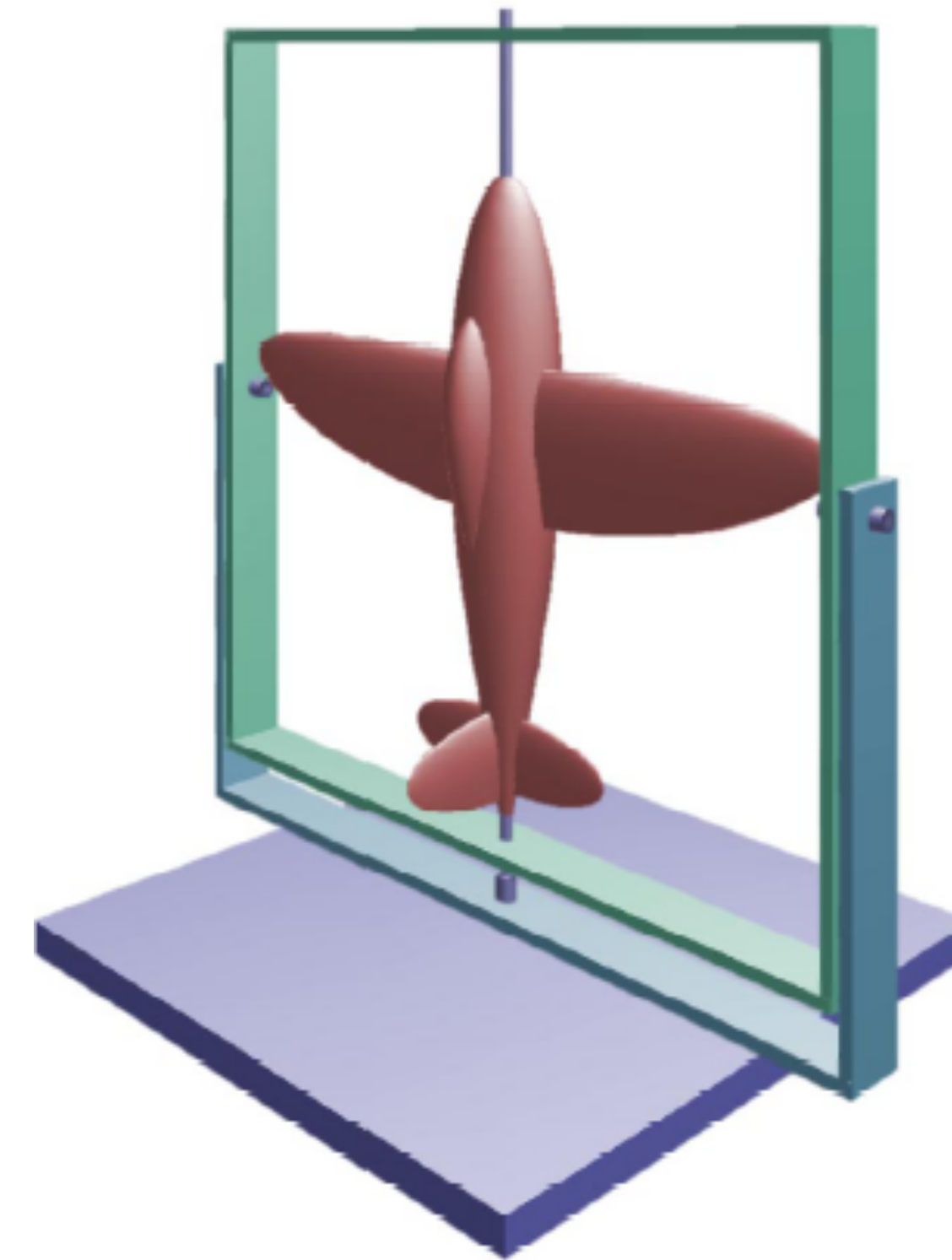
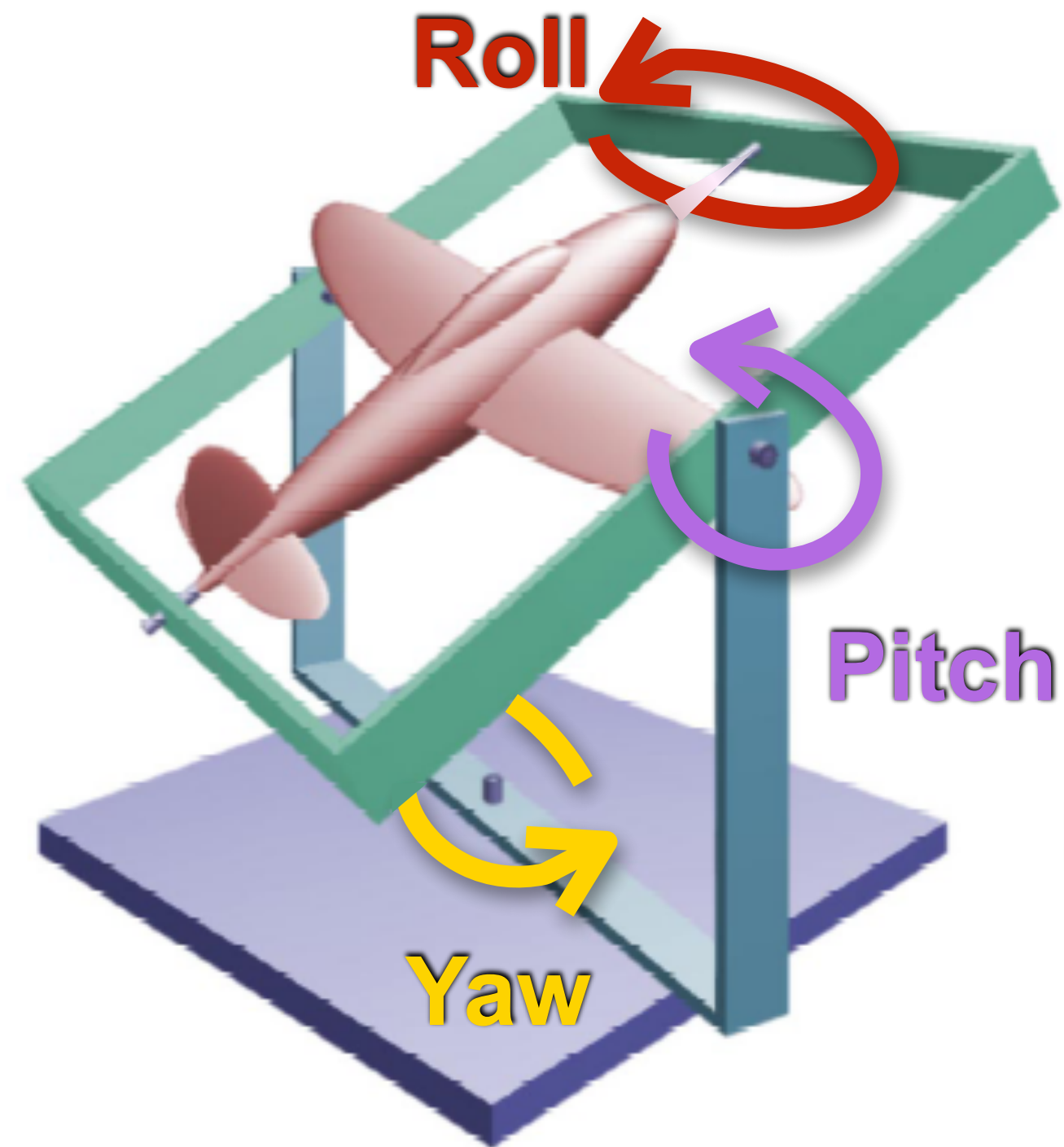
- Specify point (3 DOF)
- Specify orientation (3 DOF)
- Navigate (6 DOF)
 - ➔ Manipulate object
 - ➔ Manipulate eye/walk
 - ➔ Specify path
- Select an object or region in space
 - ➔ Find out its location and orientation

3D Interface

- Mouse provide a 2D input (+ buttons)
 - ➔ 2D location
 - ➔ Up-down, Left-right motion
 - ➔ 2D rotation
- 3D navigation has 6 DOF
 - ➔ Up / Down
 - ➔ Forward / Backward
 - ➔ Left / Right
 - ➔ Pitch (transverse axis)
 - ➔ Yaw (normal axis)
 - ➔ Roll (longitudinal axis)



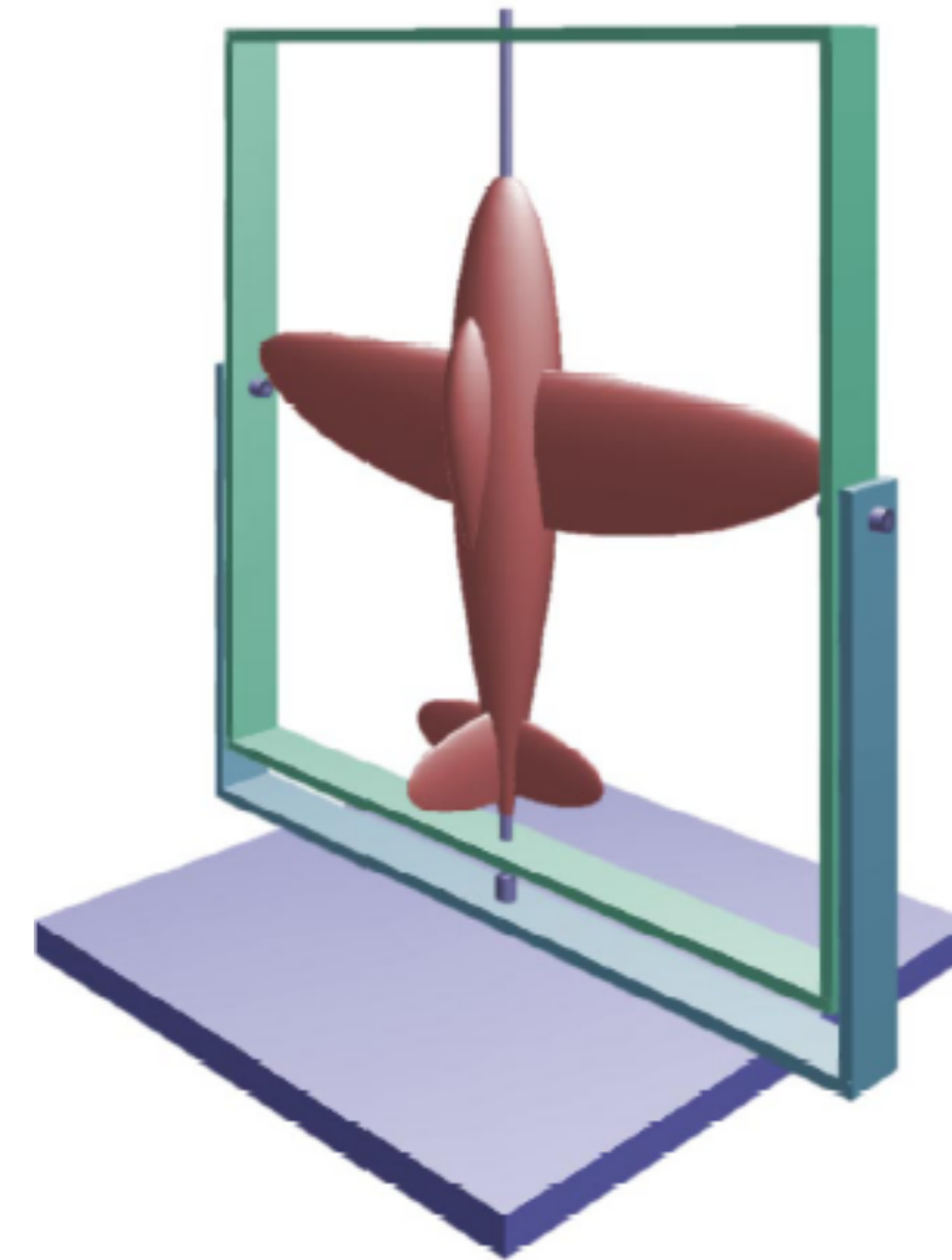
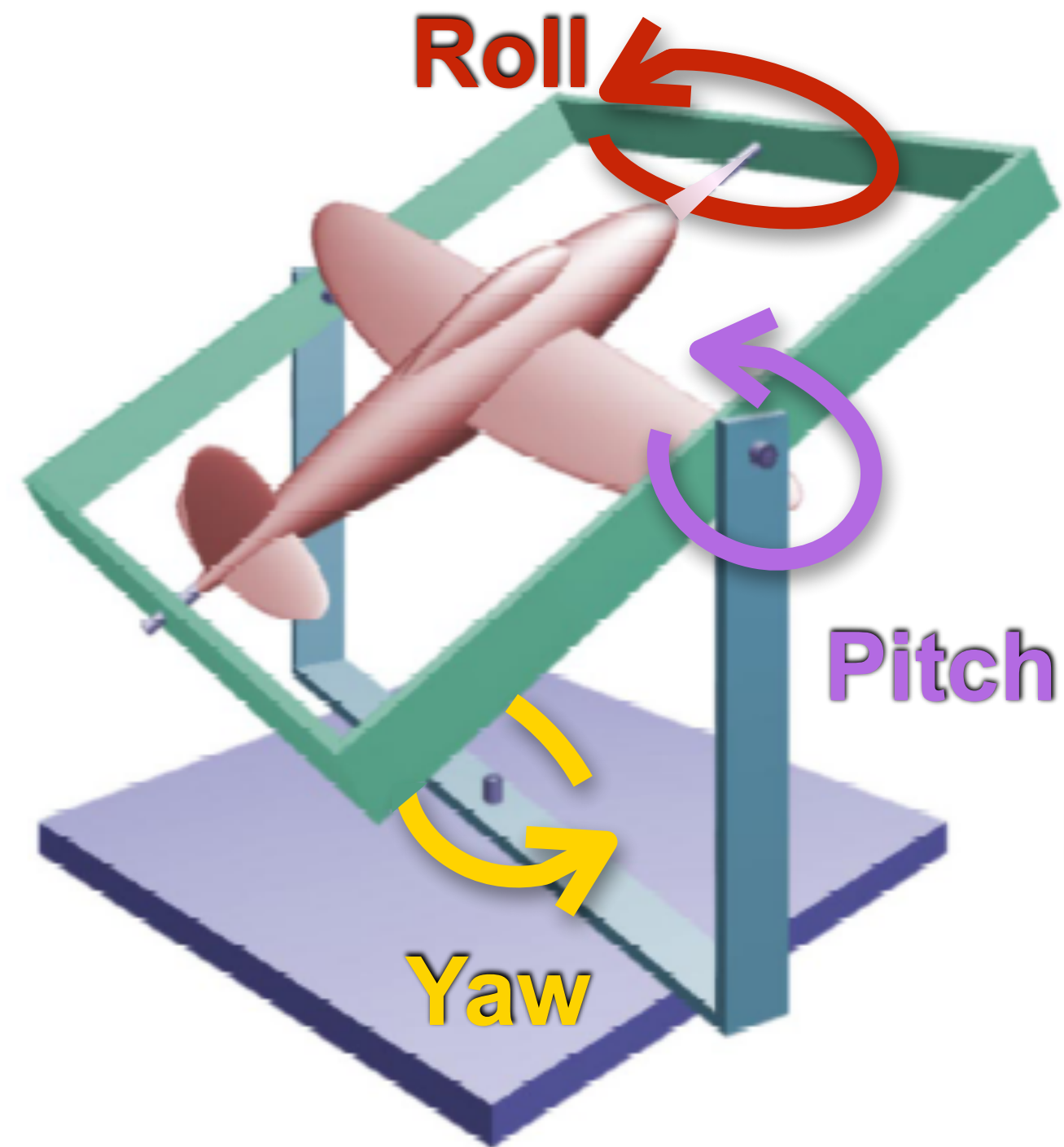
Rotation by Euler angles



Gimbal Lock

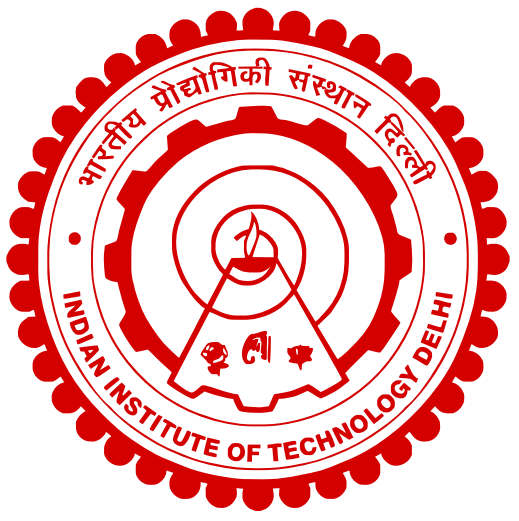


Rotation by Euler angles



Gimbal Lock

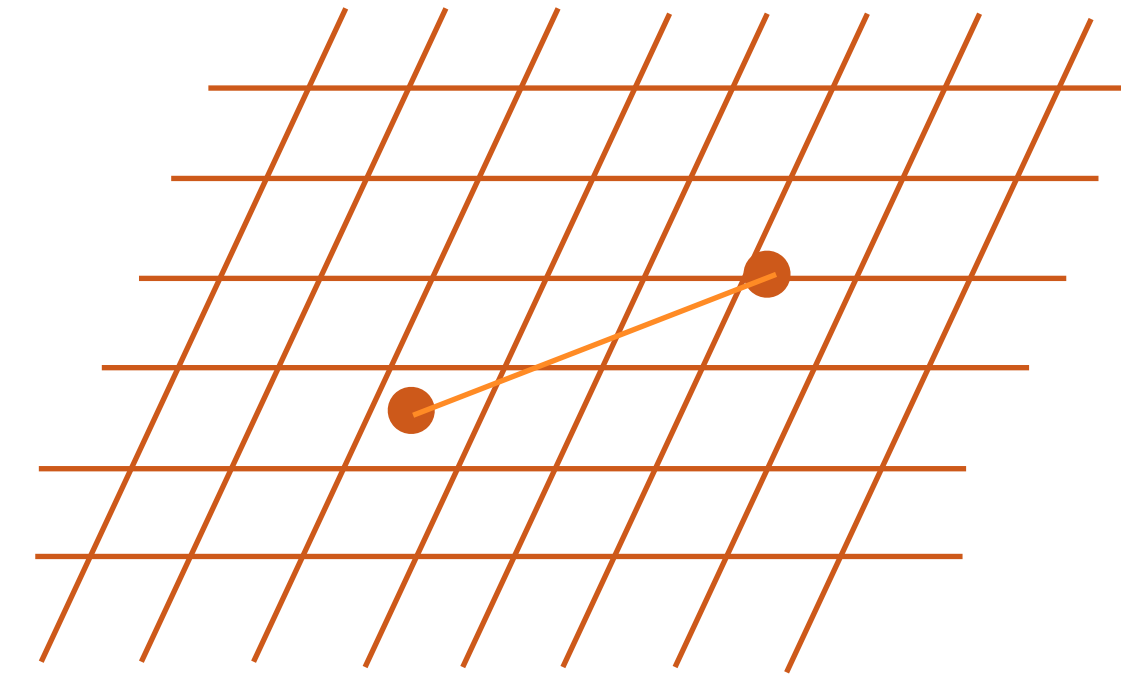
- Interpolation, Composition is non-intuitive



Trackball Interface

- Inspect an object
 - ➔ Imagine rotating a sphere by a handle
- Map planar 2D motion onto unit sphere
 - ➔ **Raise** old position $\langle x_0, z_0 \rangle$ to $\langle x_0, y_0, z_0 \rangle$
 - ➔ **Raise** new position $\langle x_1, z_1 \rangle$ to $\langle x_1, y_1, z_1 \rangle$
 - ➔ Find rotation that moves handle from old to new

Trackball



Trackball

- $p_1: x_1^2 + y_1^2 + z_1^2 = 1$
- $p_2: x_2^2 + y_2^2 + z_2^2 = 1$
- Ball rotates about the axis formed by p_1 and p_2
 - $p_1 \times p_2$
- Rotation angle proportional to the distance $p_1 p_2$
 - Or simply $\|p_1 \times p_2\|$
- What happens if the object is not at the origin?

