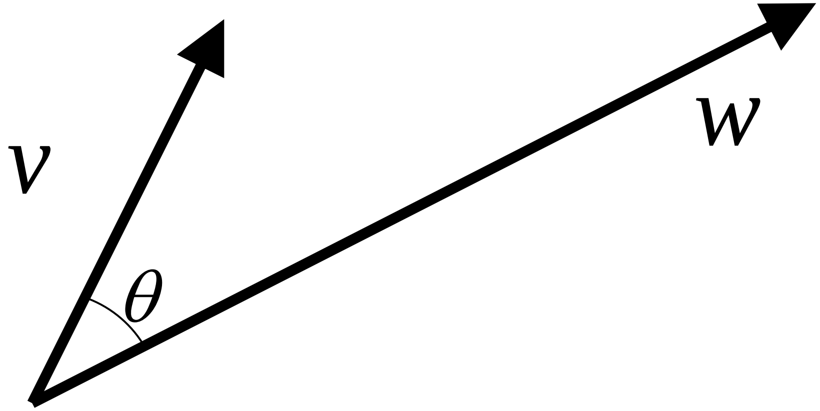
3D Transformations

Dr. Scott Schaefer

TAMU Computer Science

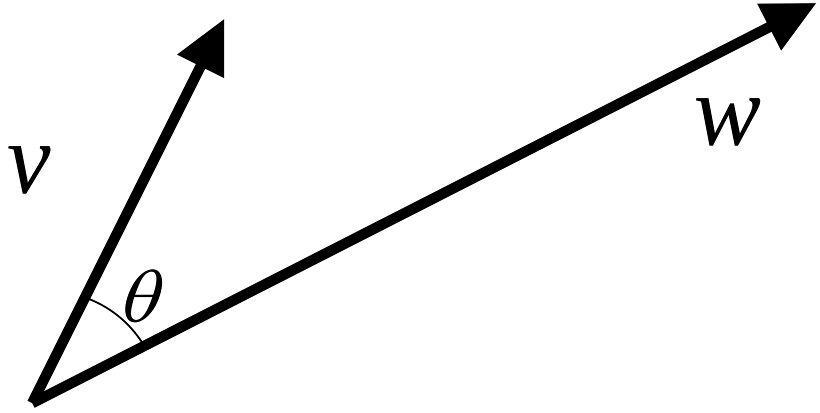
Review – Vector Operations

- Dot Product



Review – Vector Operations

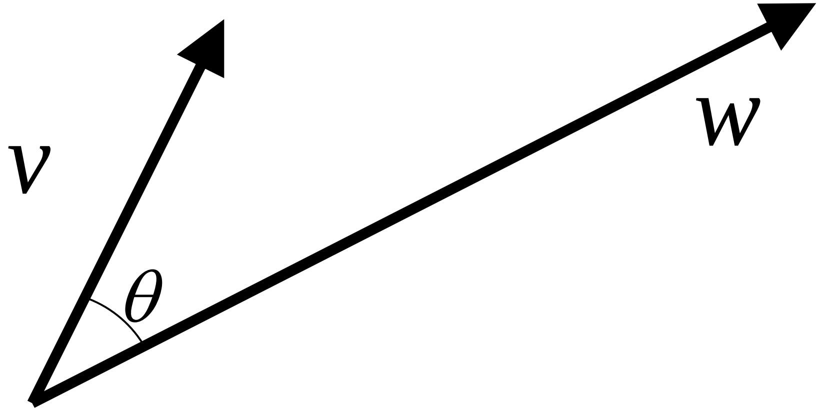
- Dot Product



$$v \cdot w = |v||w|\cos(\theta)$$

Review – Vector Operations

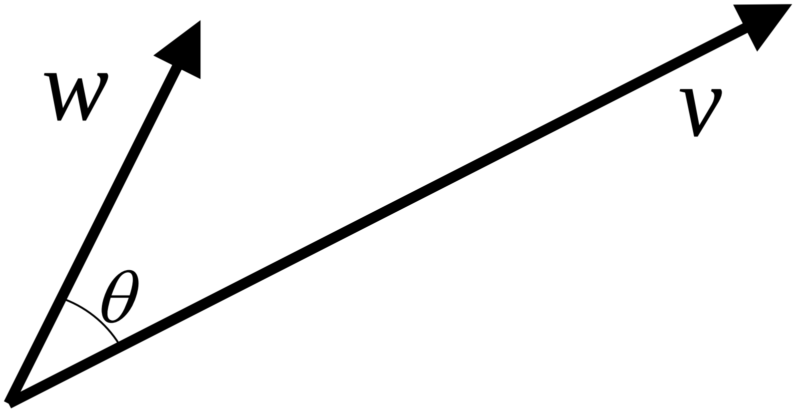
- Dot Product



$$v \cdot w = v_x w_x + v_y w_y + v_z w_z$$

Review – Vector Operations

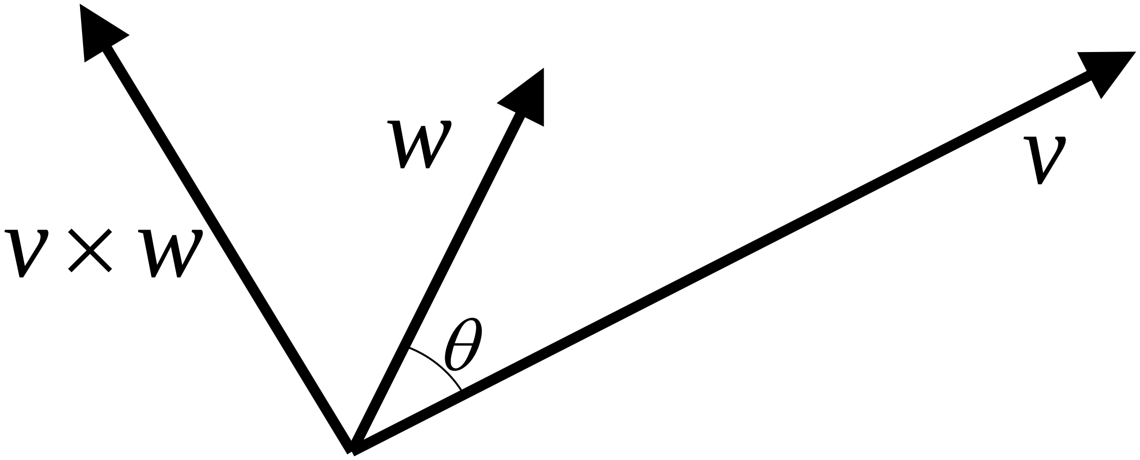
- Cross Product



$$v \times w = ?$$

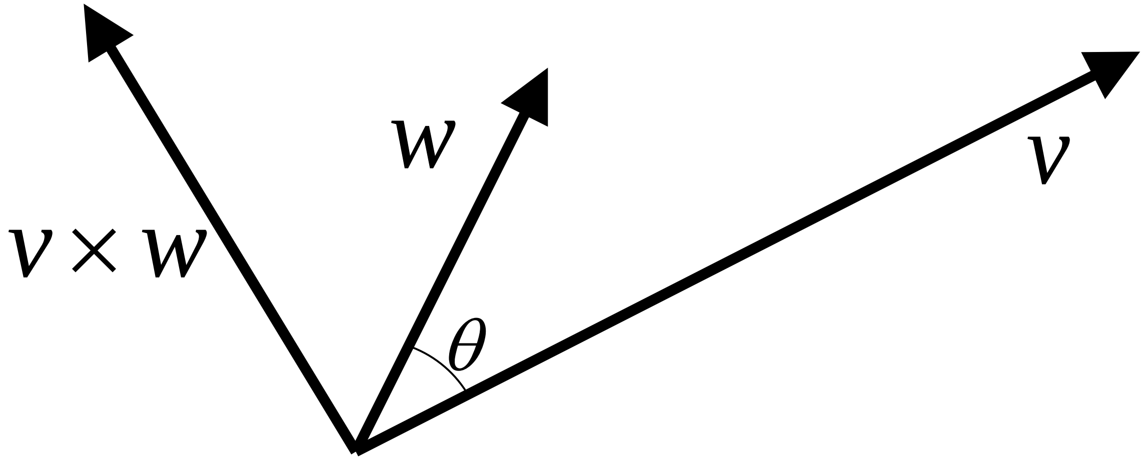
Review – Vector Operations

- Cross Product



Review – Vector Operations

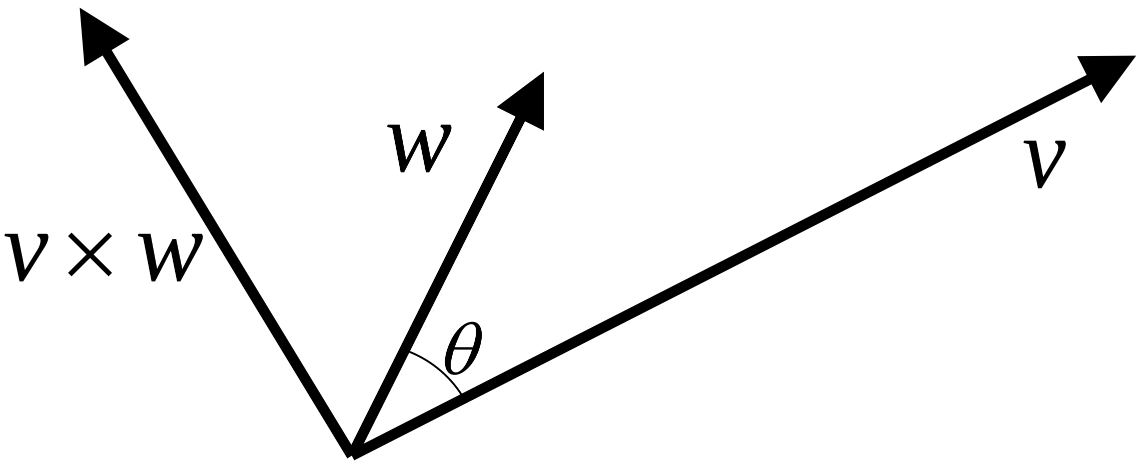
- Cross Product



$$v \cdot (v \times w) = 0$$

Review – Vector Operations

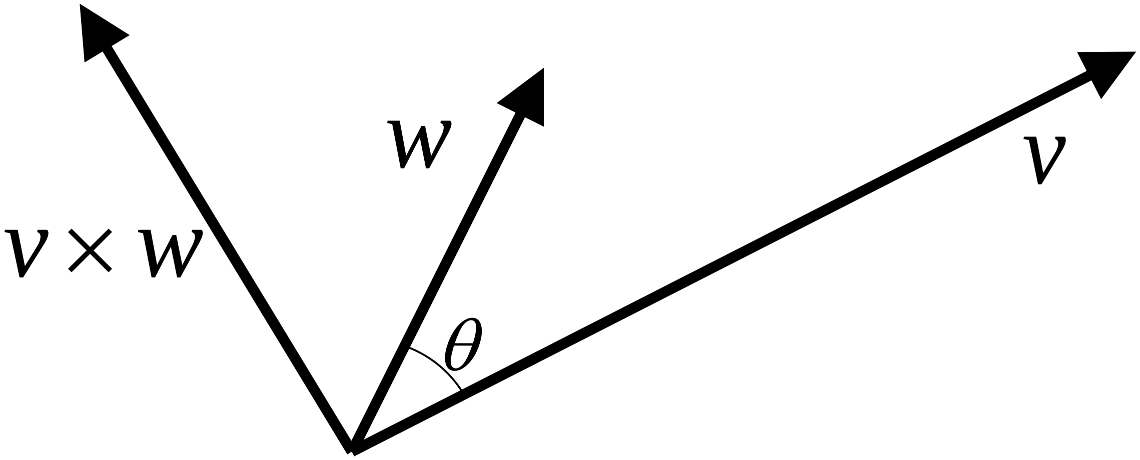
- Cross Product



$$w \cdot (v \times w) = 0$$

Review – Vector Operations

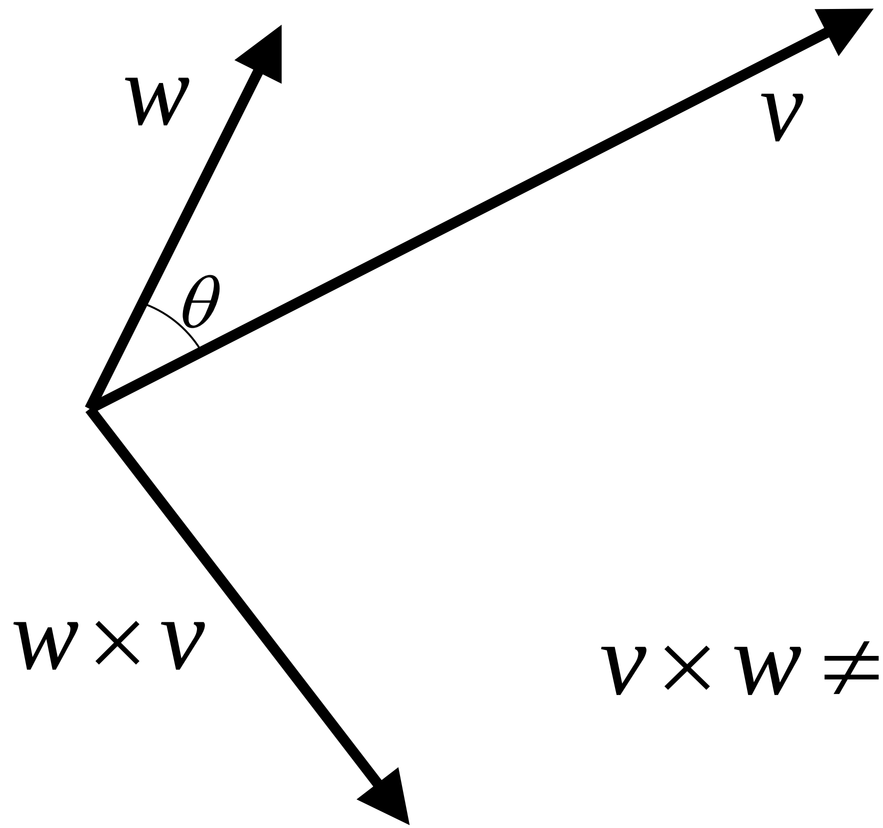
- Cross Product



$$v \times w \neq w \times v$$

Review – Vector Operations

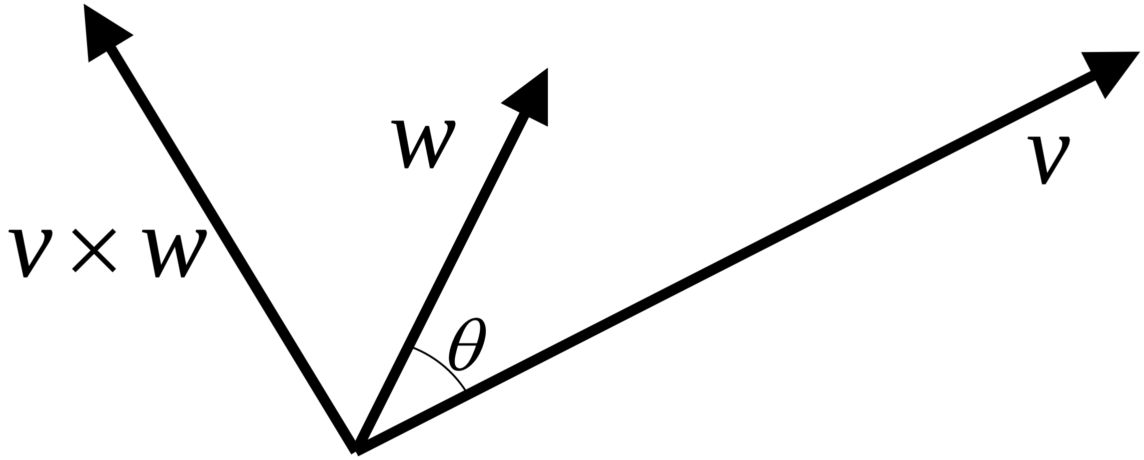
- Cross Product



$$v \times w \neq w \times v$$

Review – Vector Operations

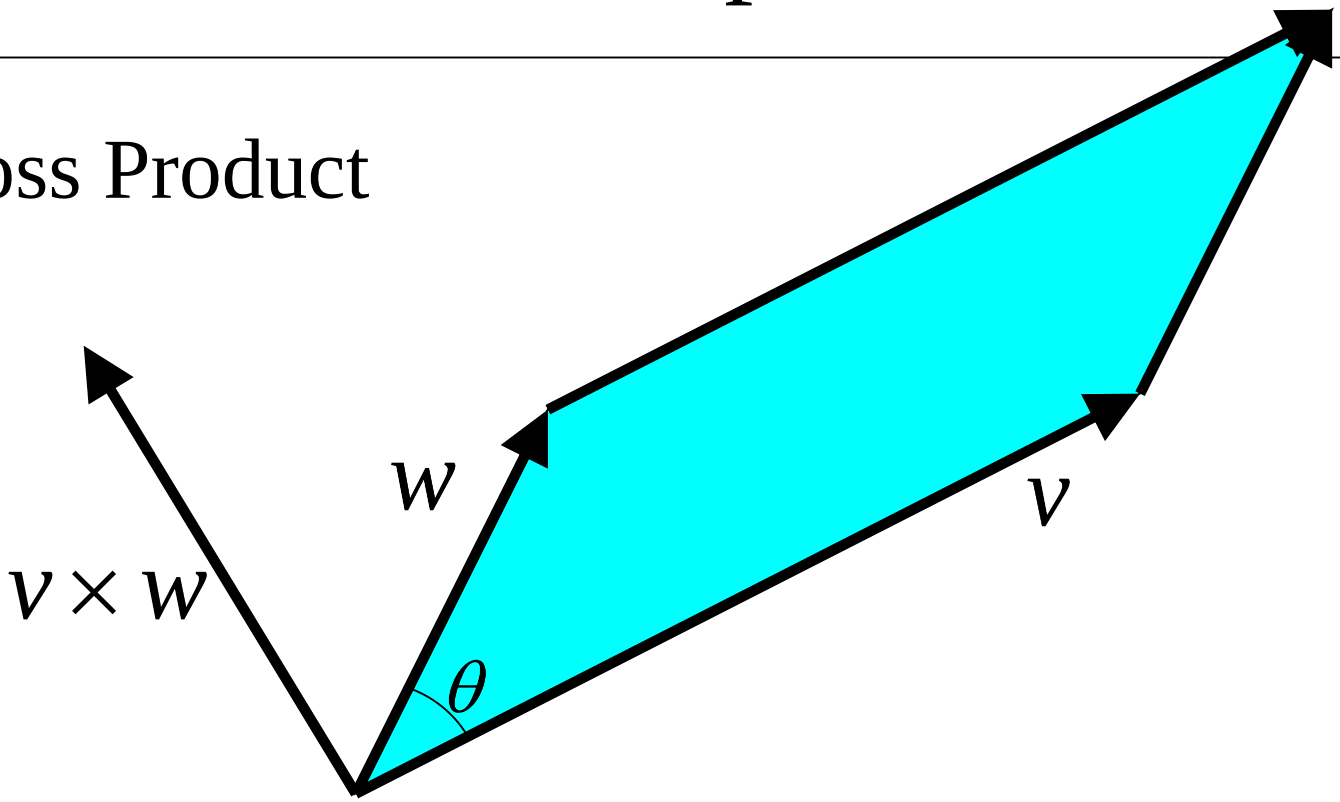
- Cross Product



$$|v \times w| = ?$$

Review – Vector Operations

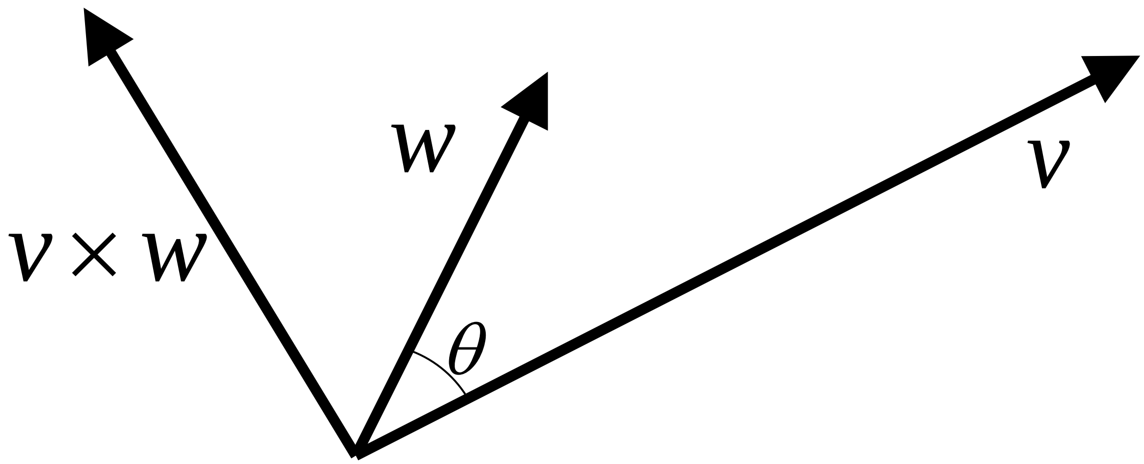
- Cross Product



$$|v \times w| = |v|\,|w|\sin(\theta)$$

Review – Vector Operations

- Cross Product

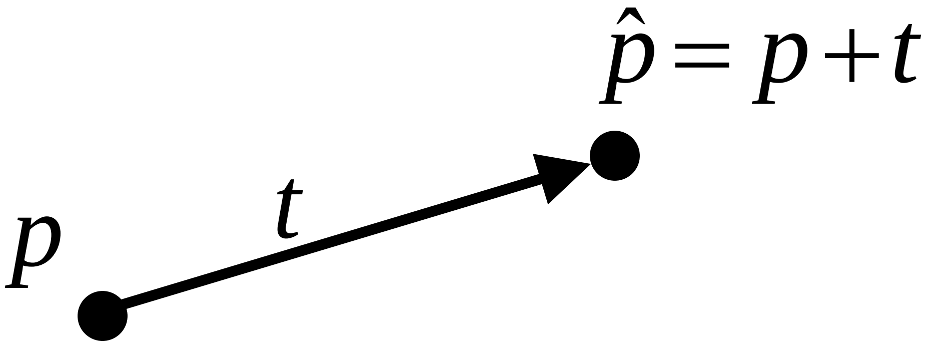


$$v \times w = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{vmatrix} = (v_y w_z - v_z w_y, v_z w_x - v_x w_z, v_x w_y - v_y w_x)$$

Transformations

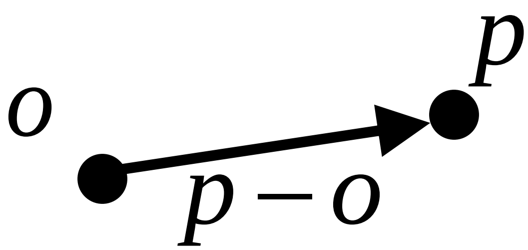
- Affine Transformations
  - Translation
  - Uniform Scaling
  - Non-uniform Scaling
  - Rotation
  - Mirror Image
- Projections
  - Orthogonal
  - Perspective

Translation

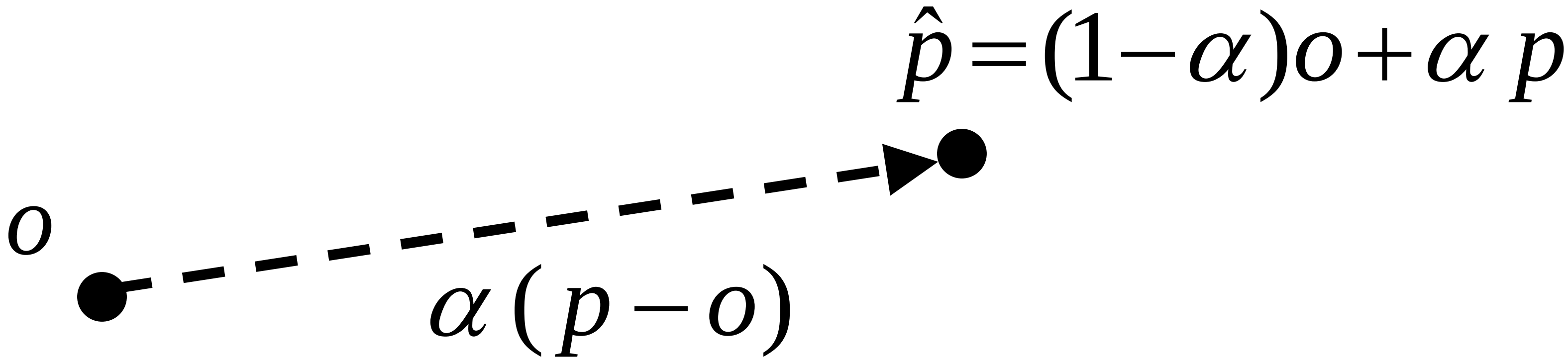


Uniform Scaling

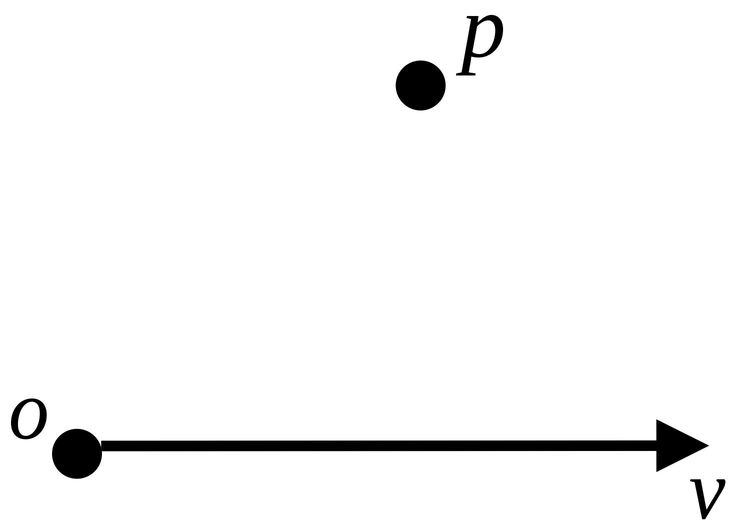
Uniform Scaling



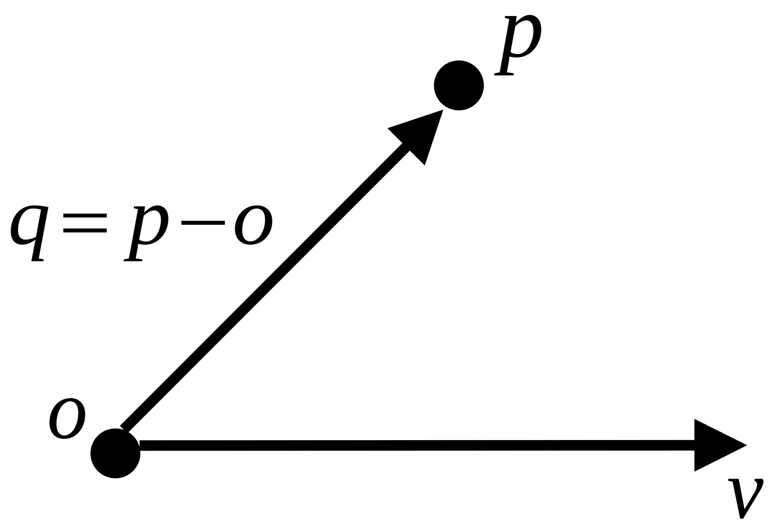
Uniform Scaling



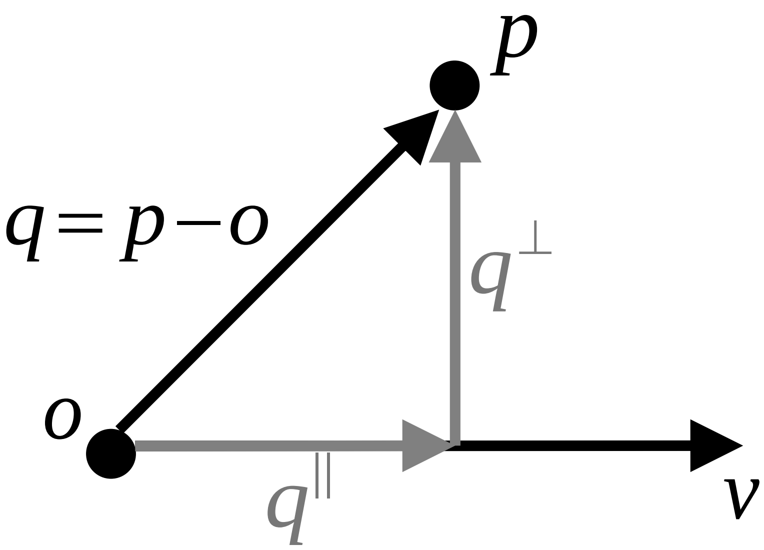
Non-Uniform Scaling



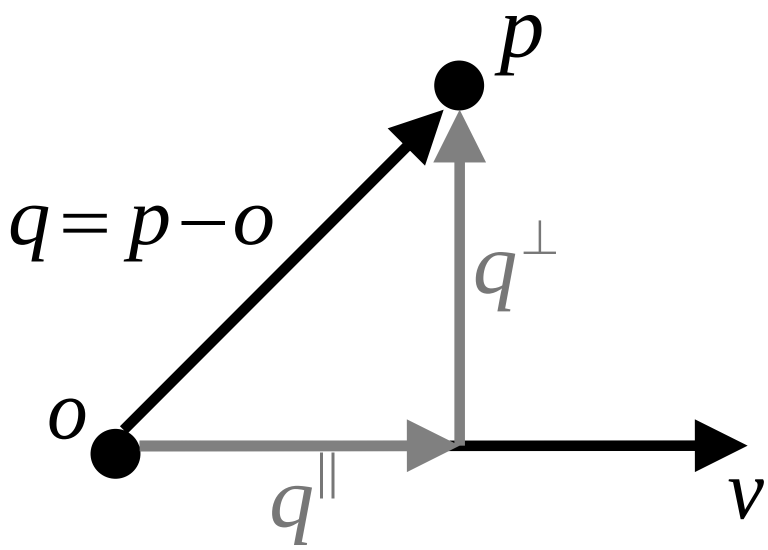
Non-Uniform Scaling



Non-Uniform Scaling



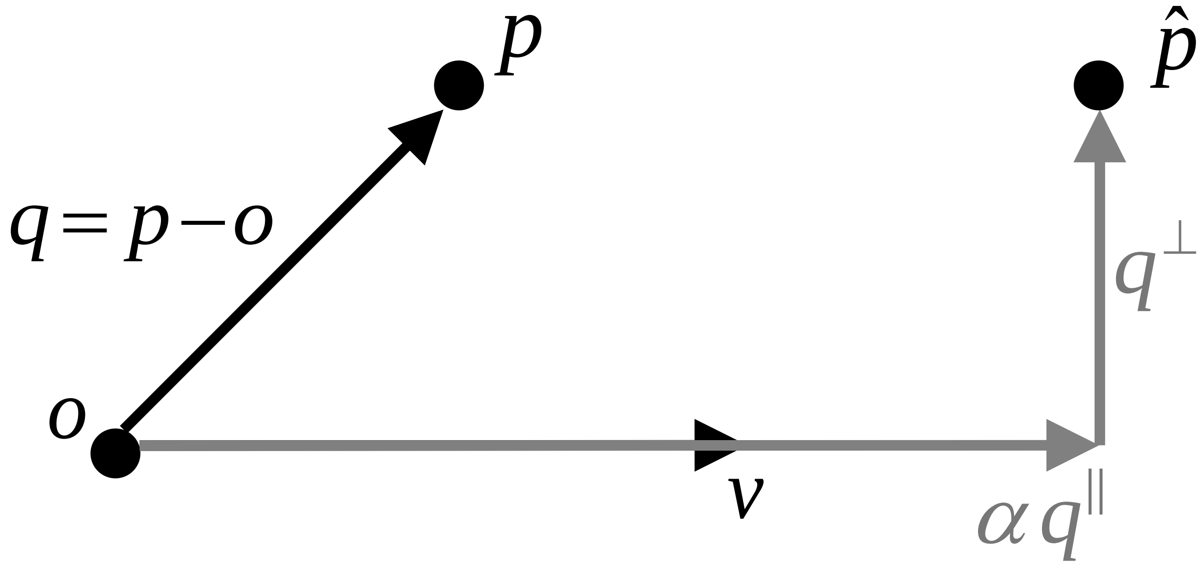
Non-Uniform Scaling



$$q^{\parallel} = (v \cdot q)v$$

$$q^{\perp} = q - (v \cdot q)v$$

Non-Uniform Scaling

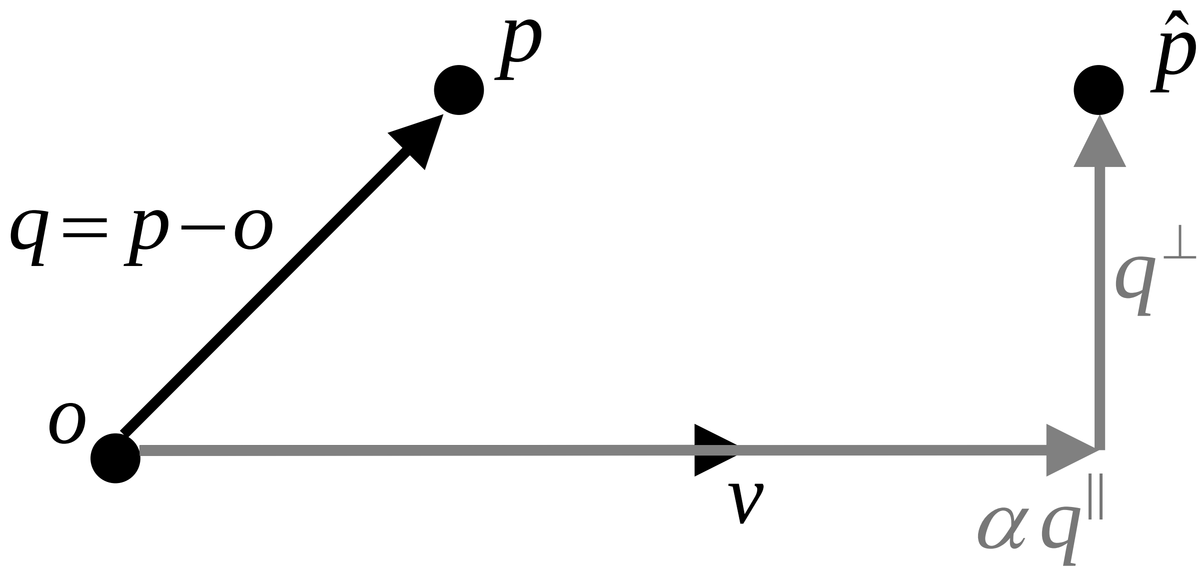


$$q^{\parallel} = (v \cdot q)v$$

$$q^{\perp} = q - (v \cdot q)v$$

$$\hat{p} = o + \alpha q^{\parallel} + q^{\perp}$$

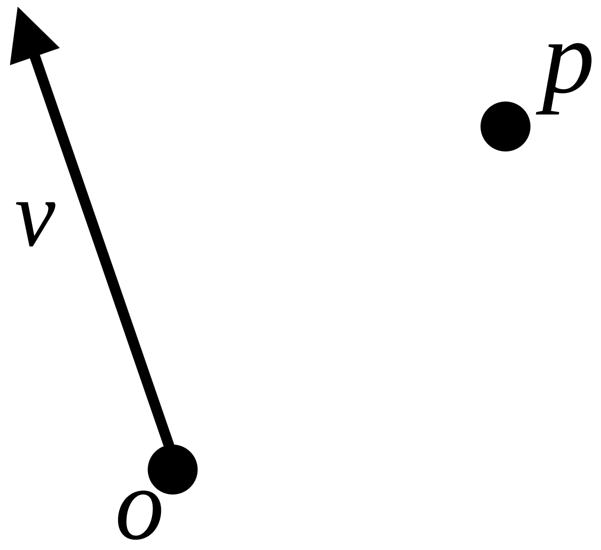
Non-Uniform Scaling



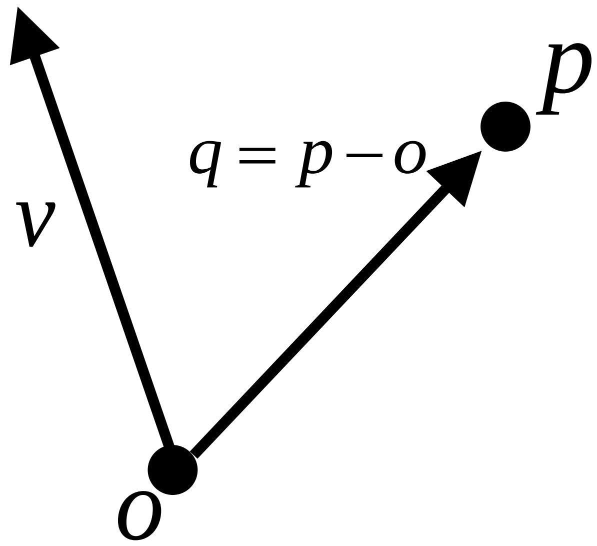
$$q^{\parallel} = (v \cdot q)v$$
$$q^{\perp} = q - (v \cdot q)v$$

$$\hat{p} = p + (\alpha - 1)(v \cdot (p - o))v$$

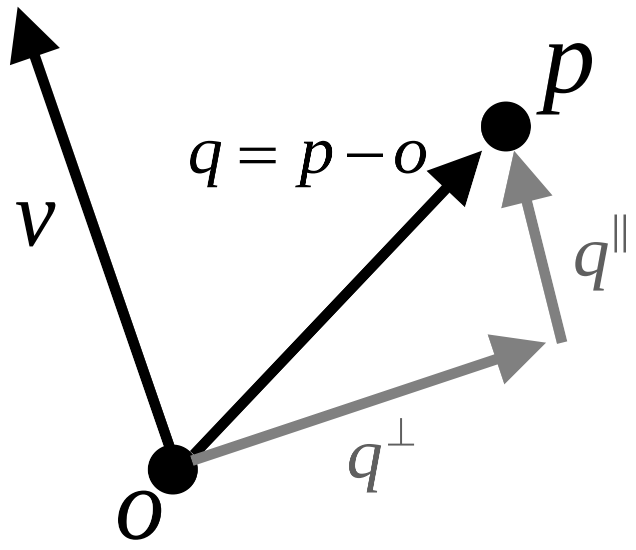
Rotation



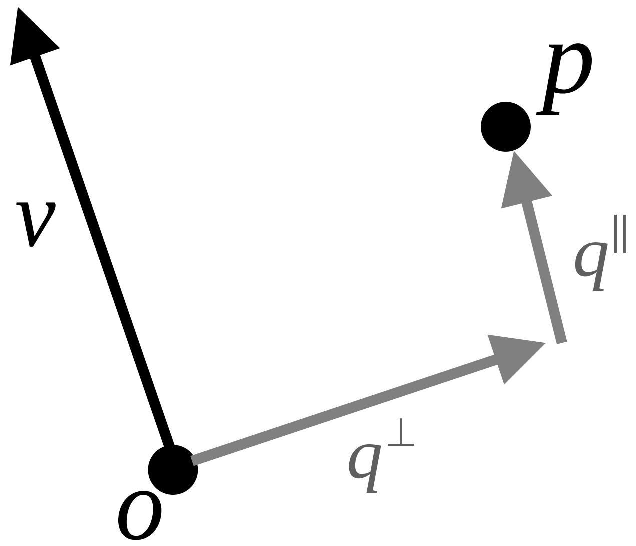
Rotation



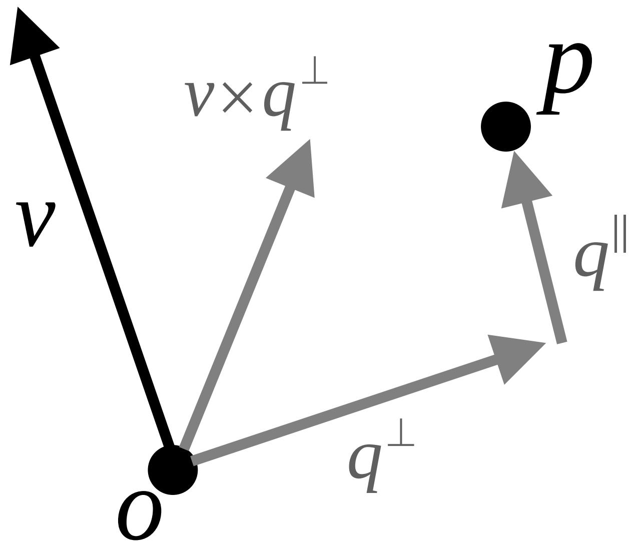
Rotation



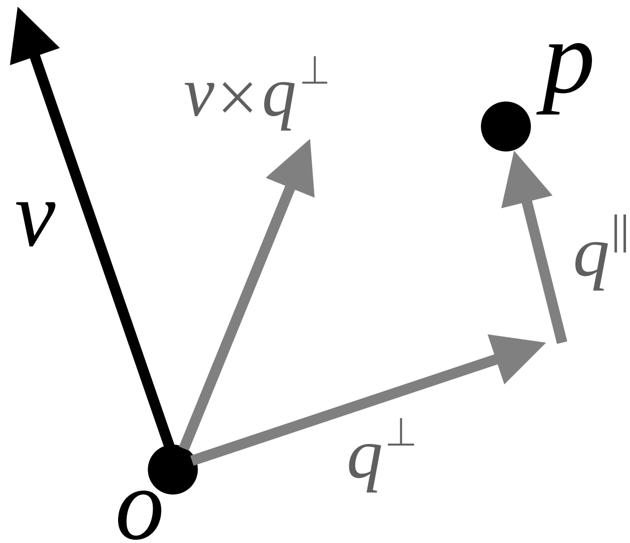
Rotation



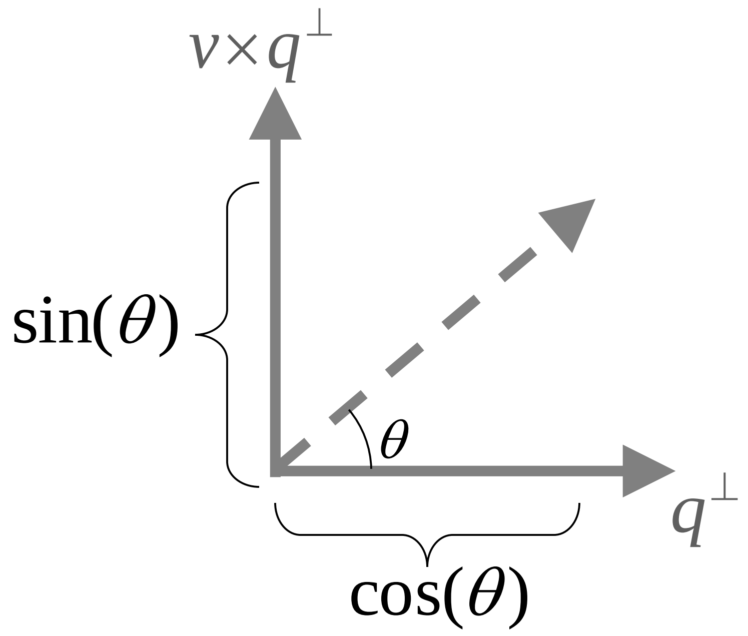
Rotation



Rotation

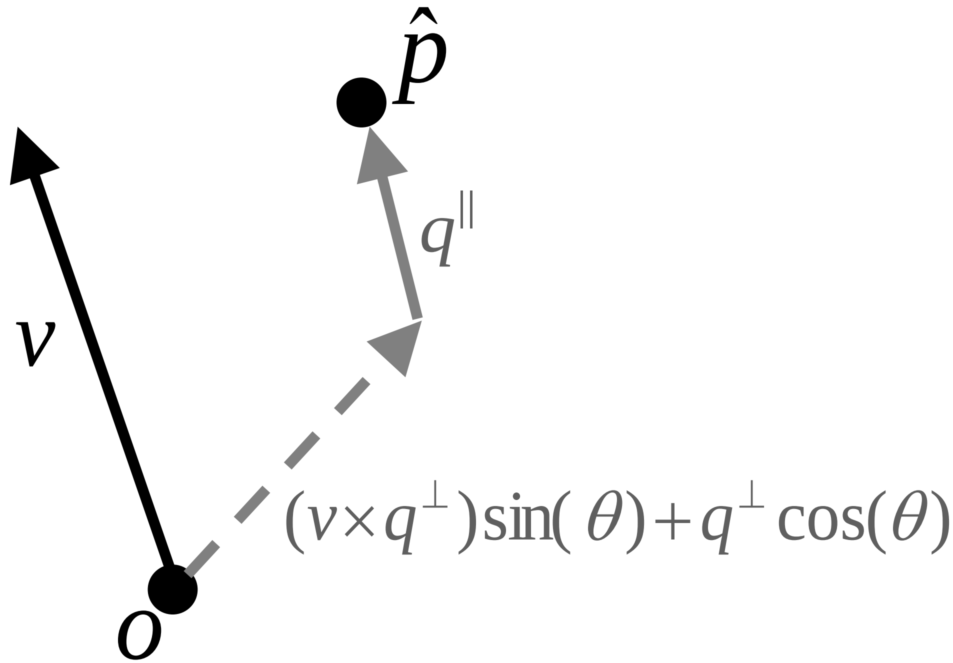


v
$v \times q^{\perp}$
p
$q^{\parallel}$
$q^{\perp}$
o



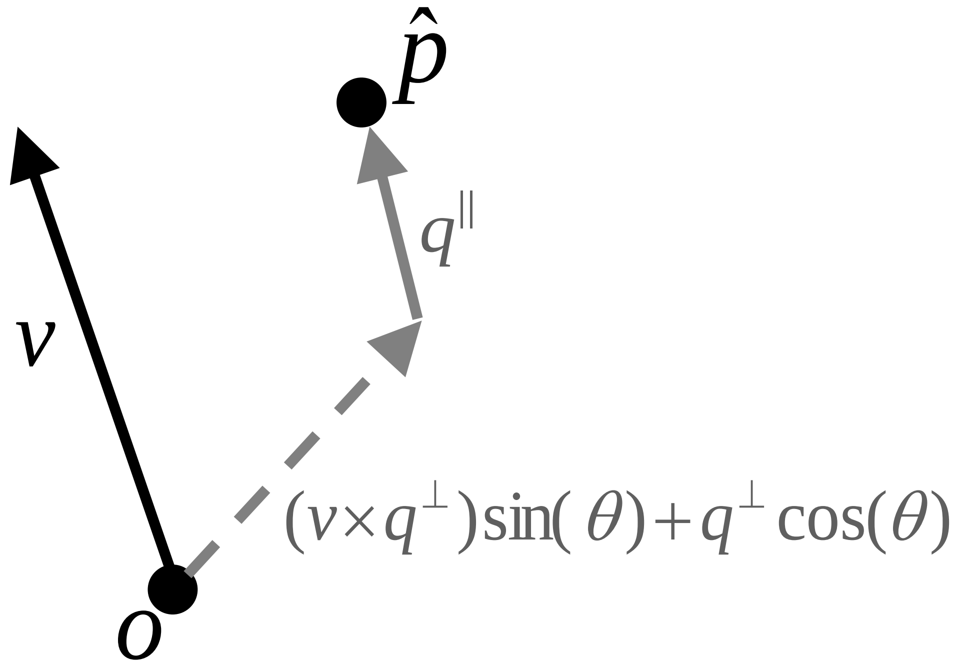
$v \times q^{\perp}$
$\sin(\theta)$
θ
$q^{\perp}$
$\cos(\theta)$

Rotation



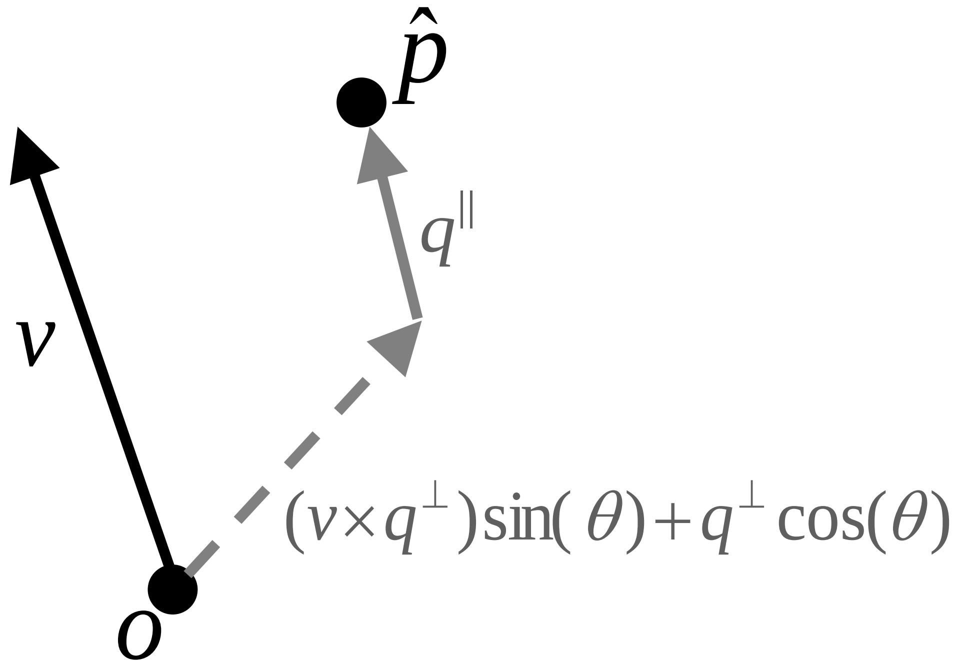
$$\hat{p} = o + q^{\|} + (v \times q^{\perp})\sin(\theta) + q^{\perp}\cos(\theta)$$

Rotation



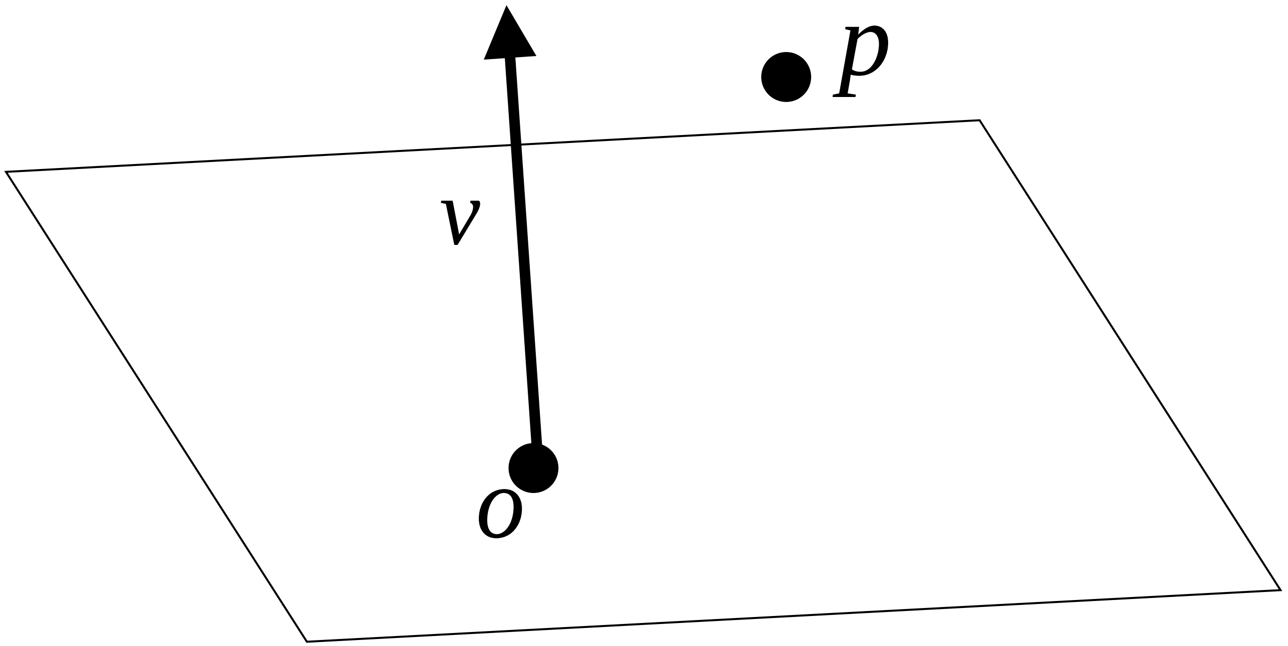
$$\hat{p}=o+q^{\parallel}+(v\times(q-q^{\parallel}))\sin(\theta)+(q-q^{\parallel})\cos(\theta)$$

Rotation

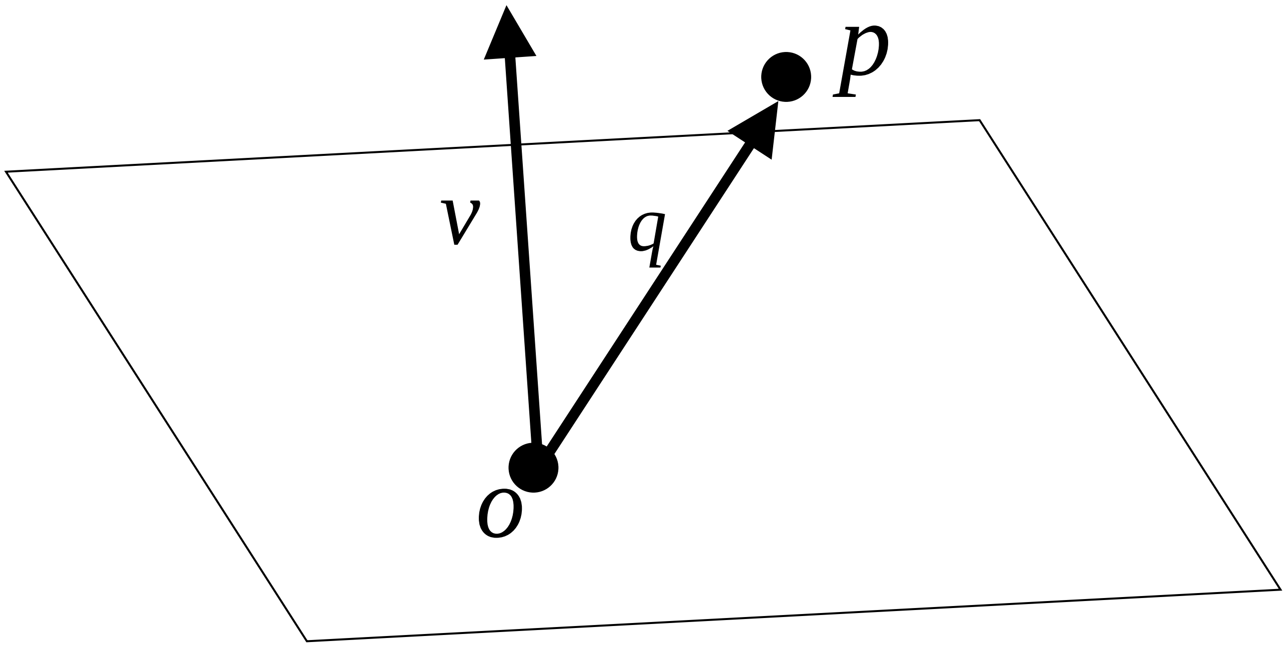


$$\hat{p} = o + (1 - \cos(\theta))q^{\parallel} + (v \times q)\sin(\theta) + q\cos(\theta)$$

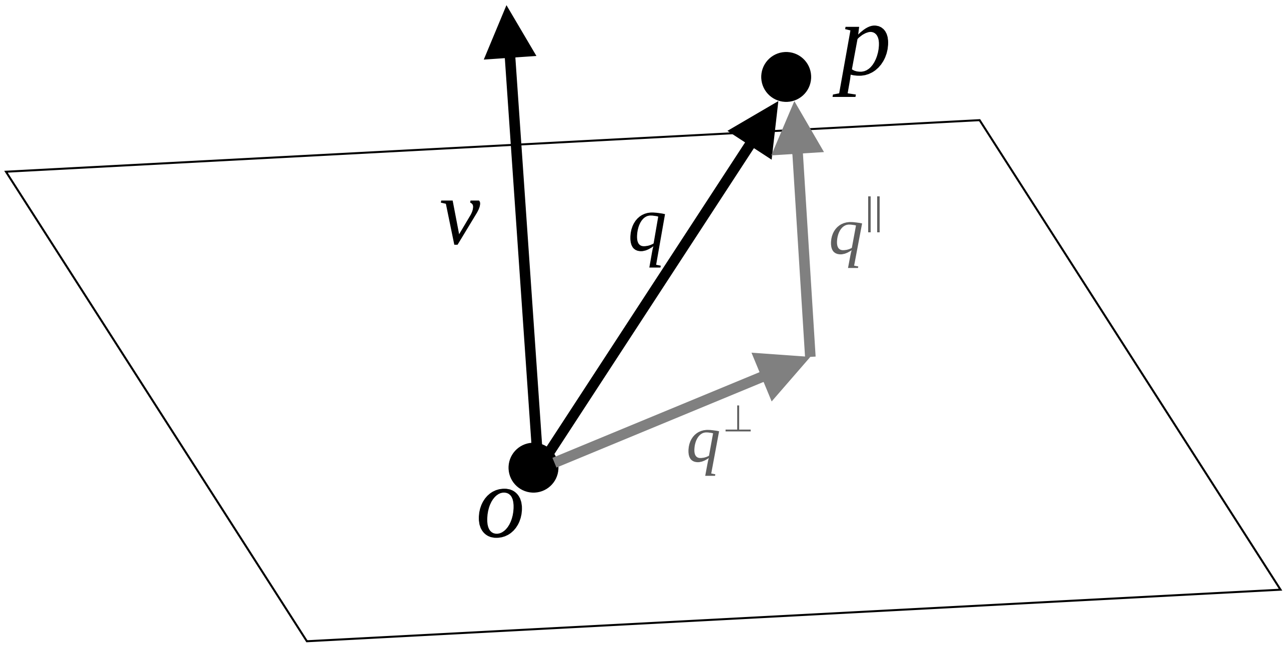
Mirror Image



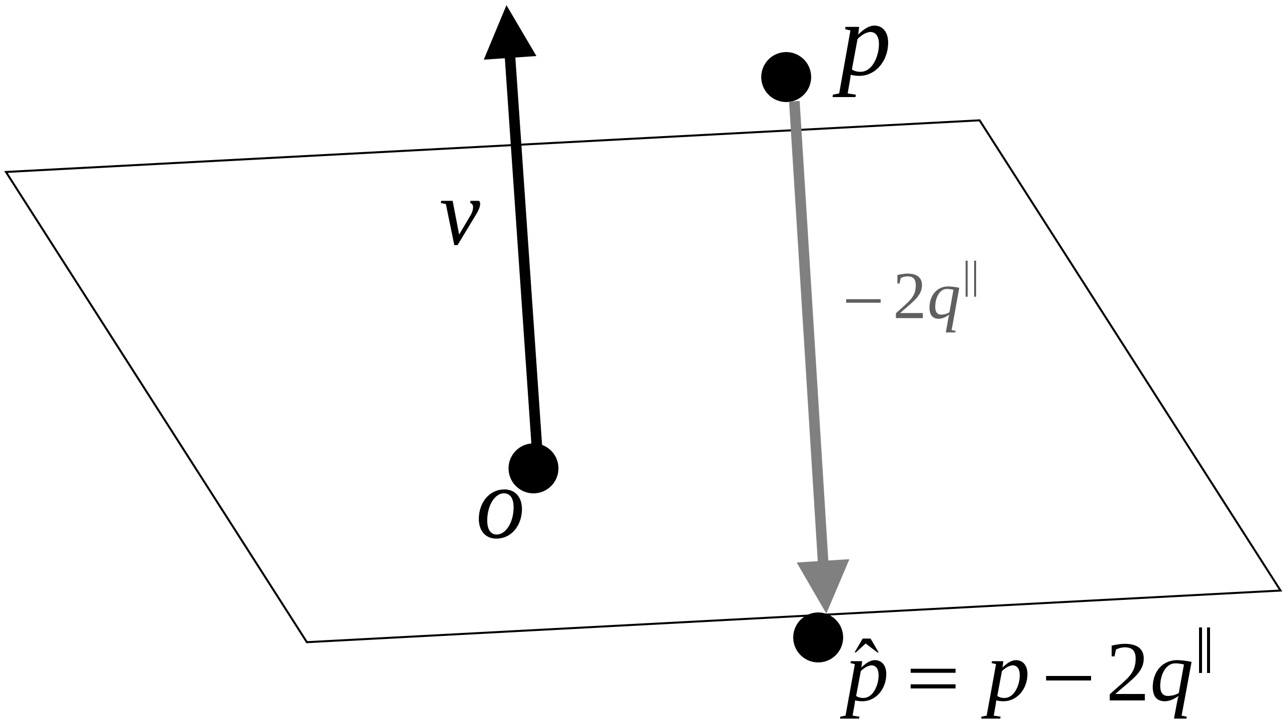
Mirror Image



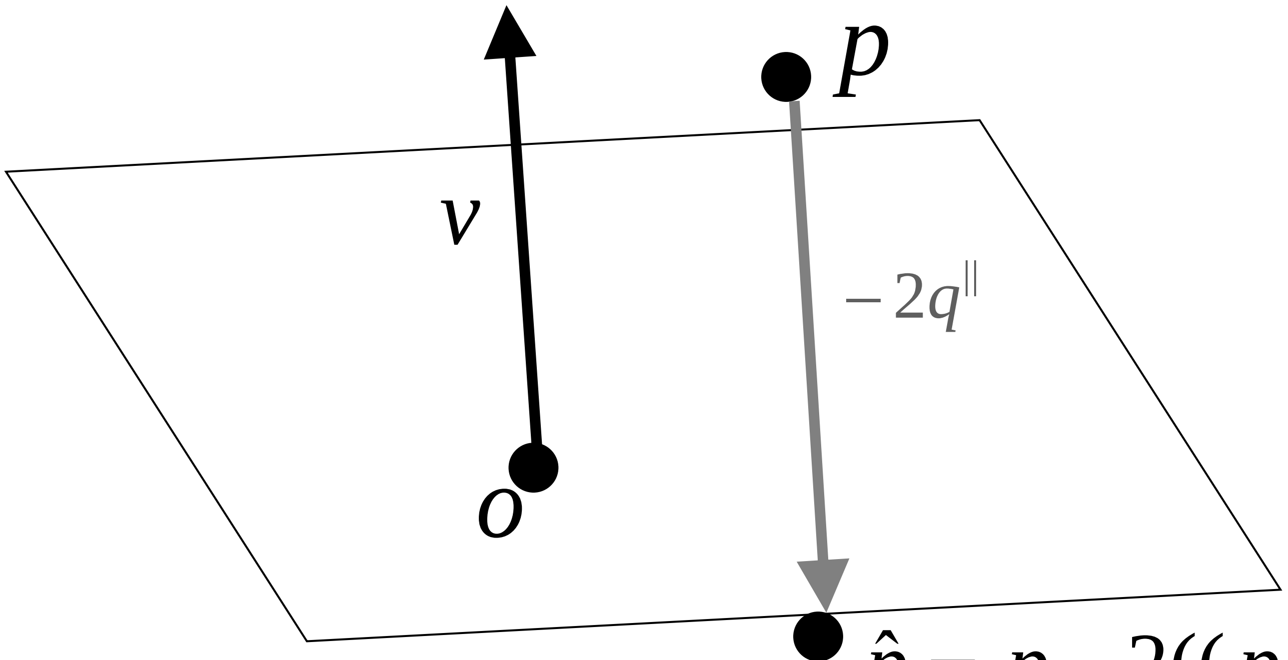
Mirror Image



Mirror Image

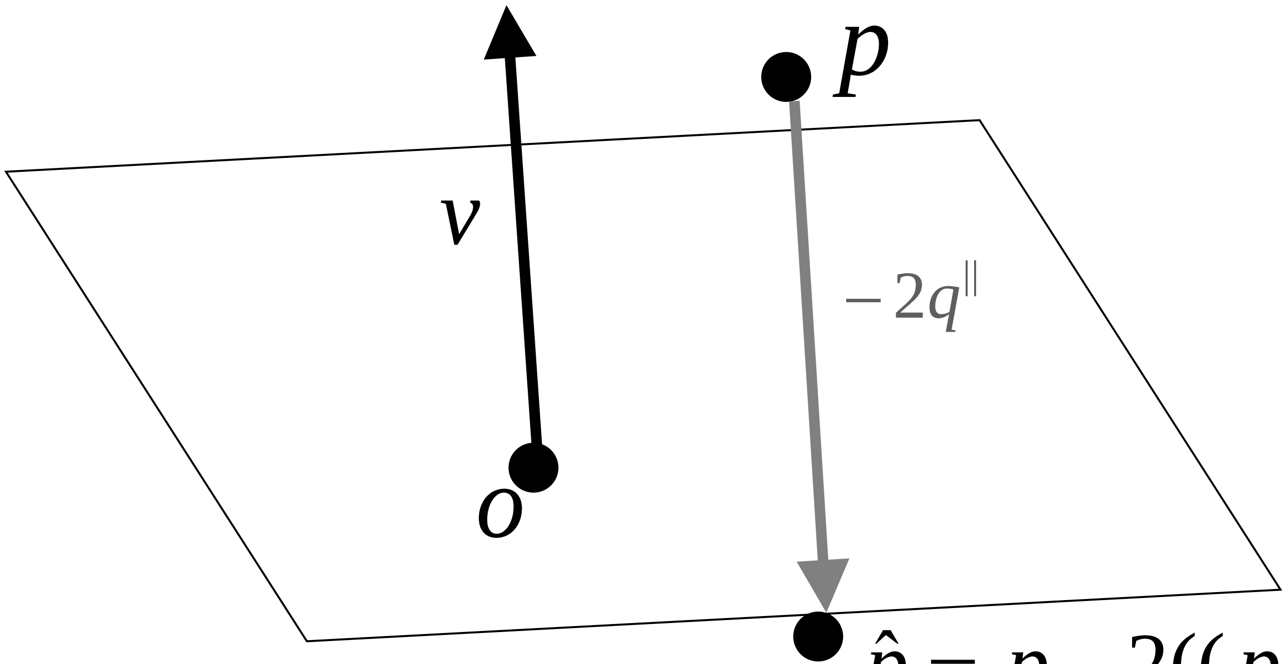


Mirror Image



$$\hat{p} = p - 2((p - o) \cdot v)v$$

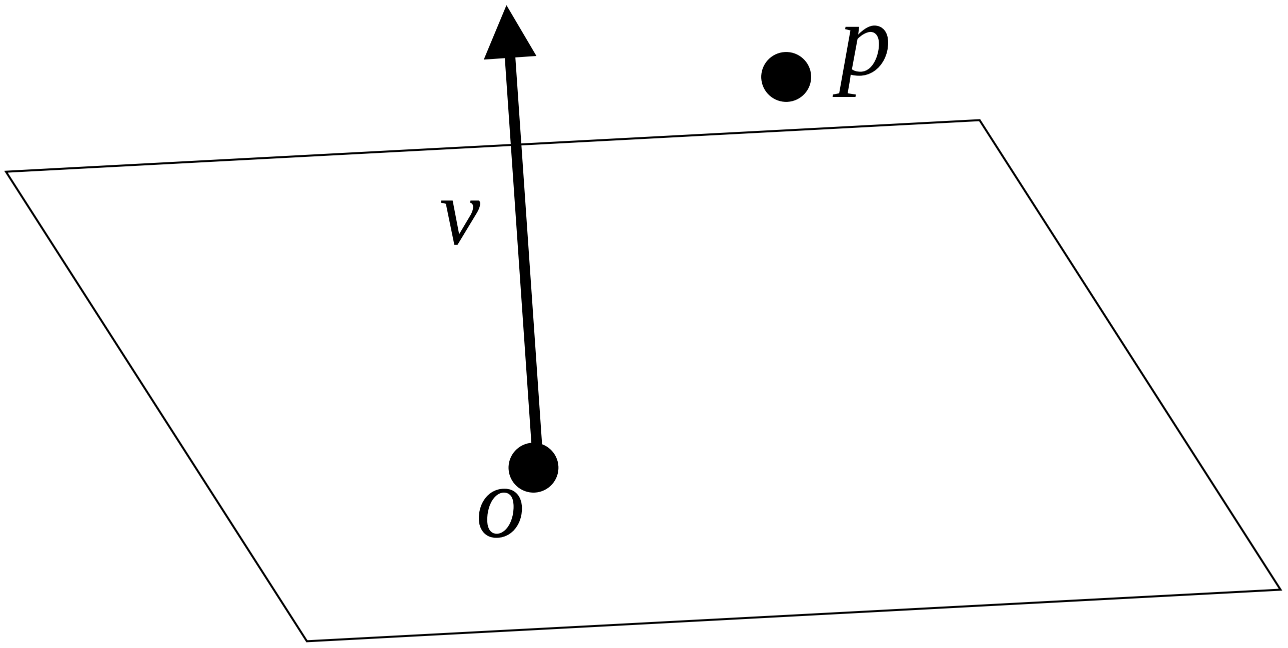
Mirror Image



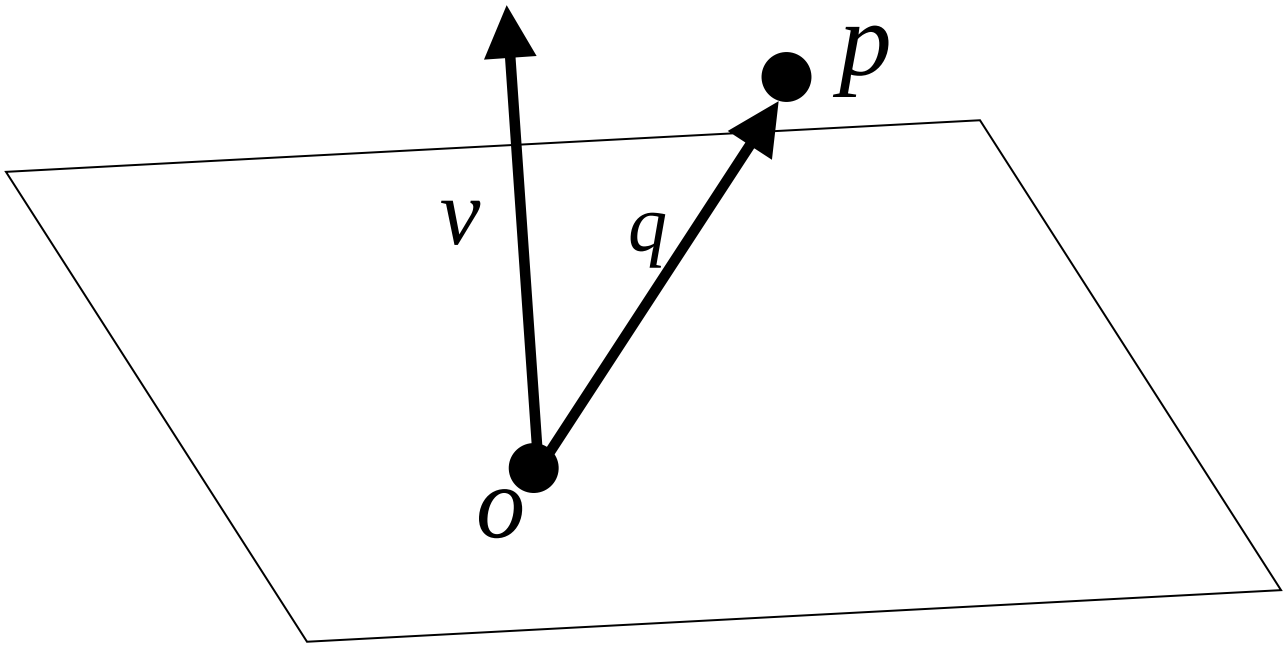
$$\hat{p} = p - 2((p - o) \cdot v)v$$

Exactly the same as non-uniform scaling with $\alpha = -1$!!!

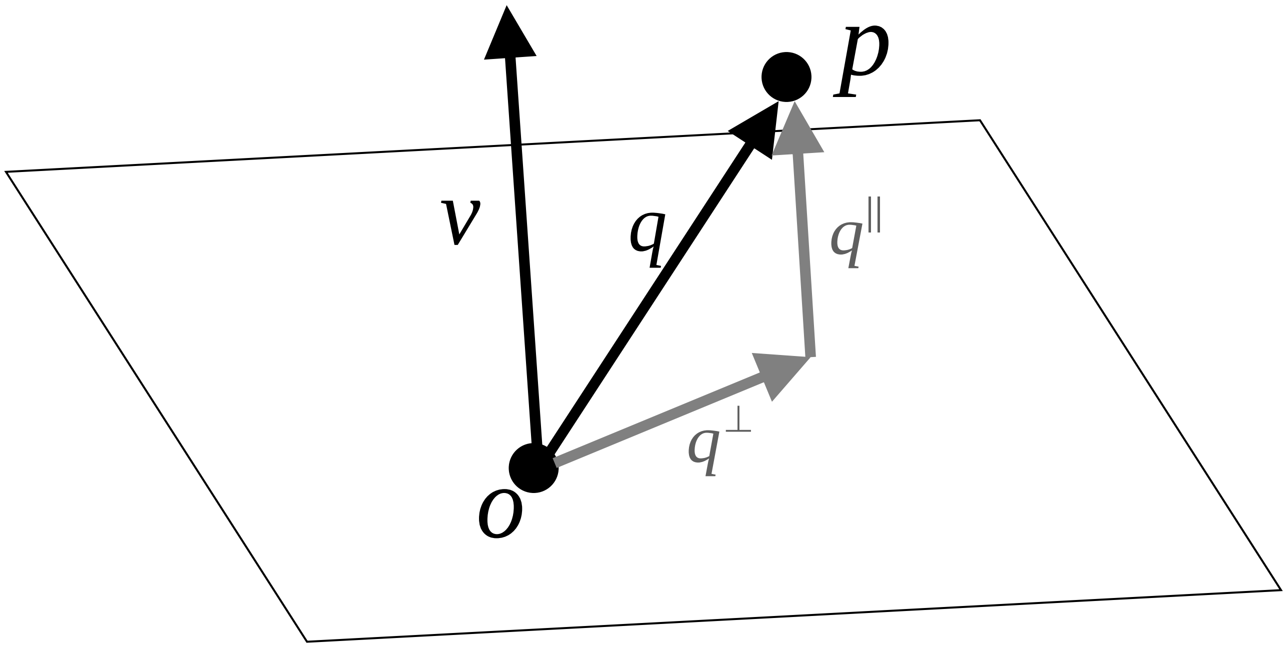
Orthogonal Projection



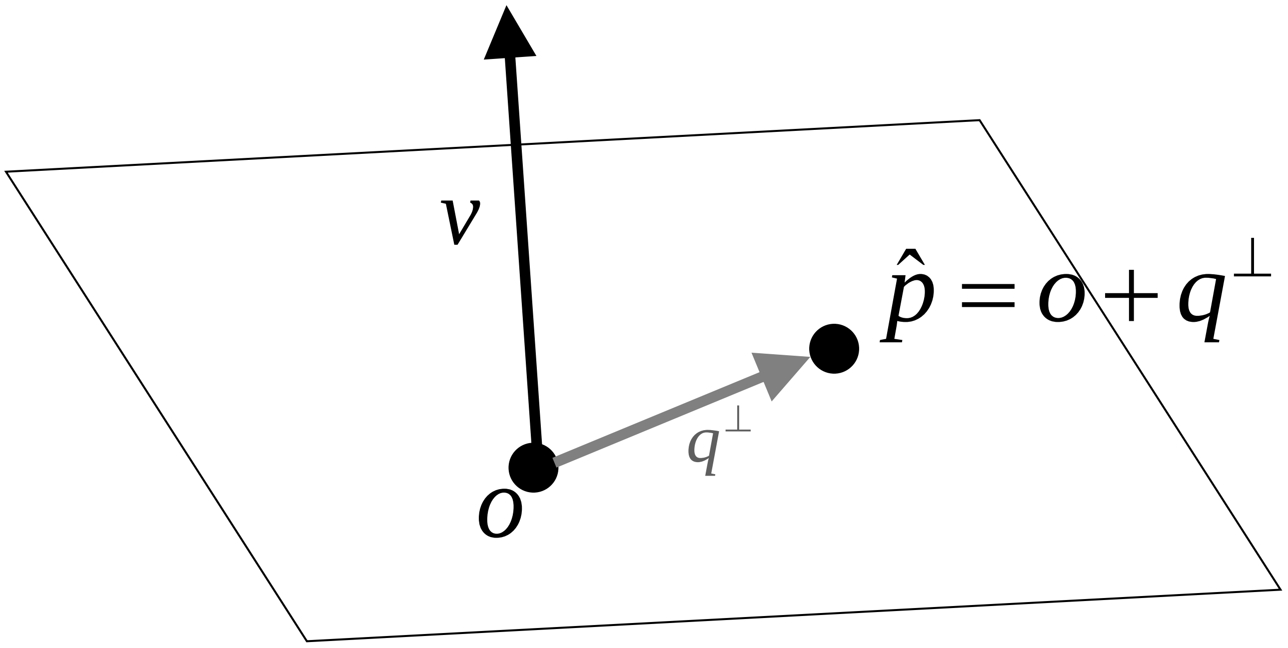
Orthogonal Projection



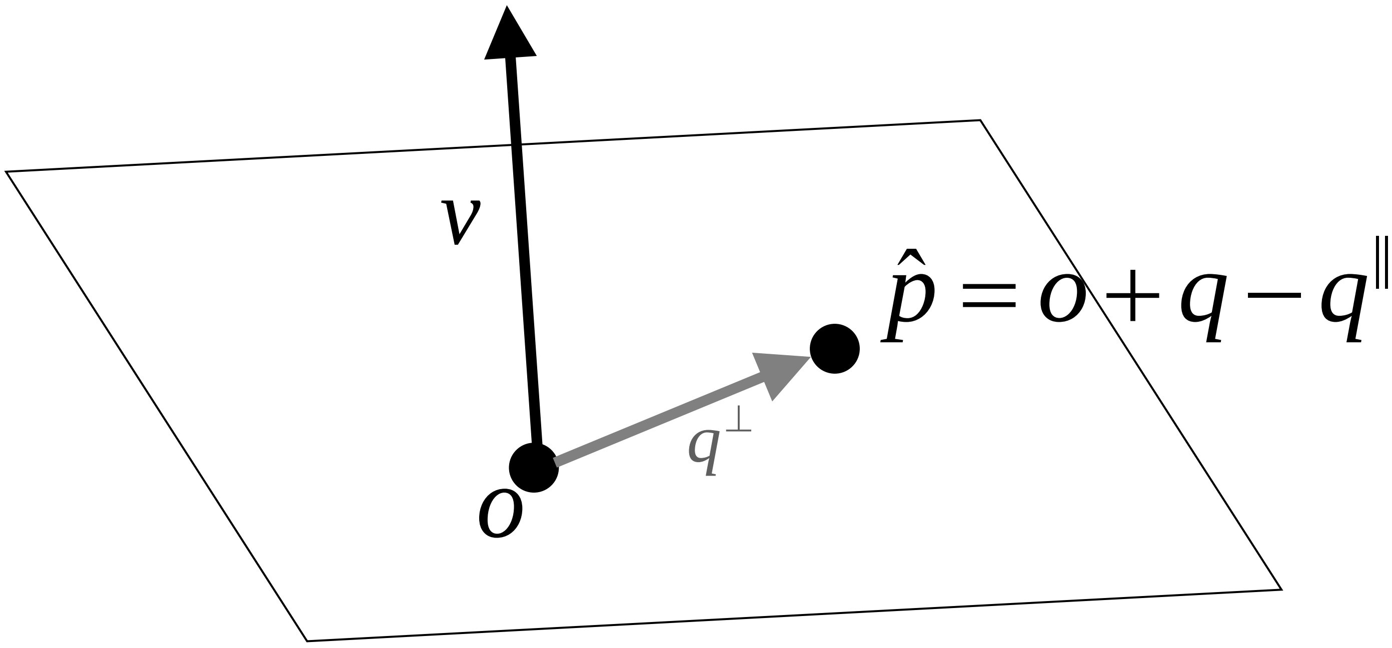
Orthogonal Projection



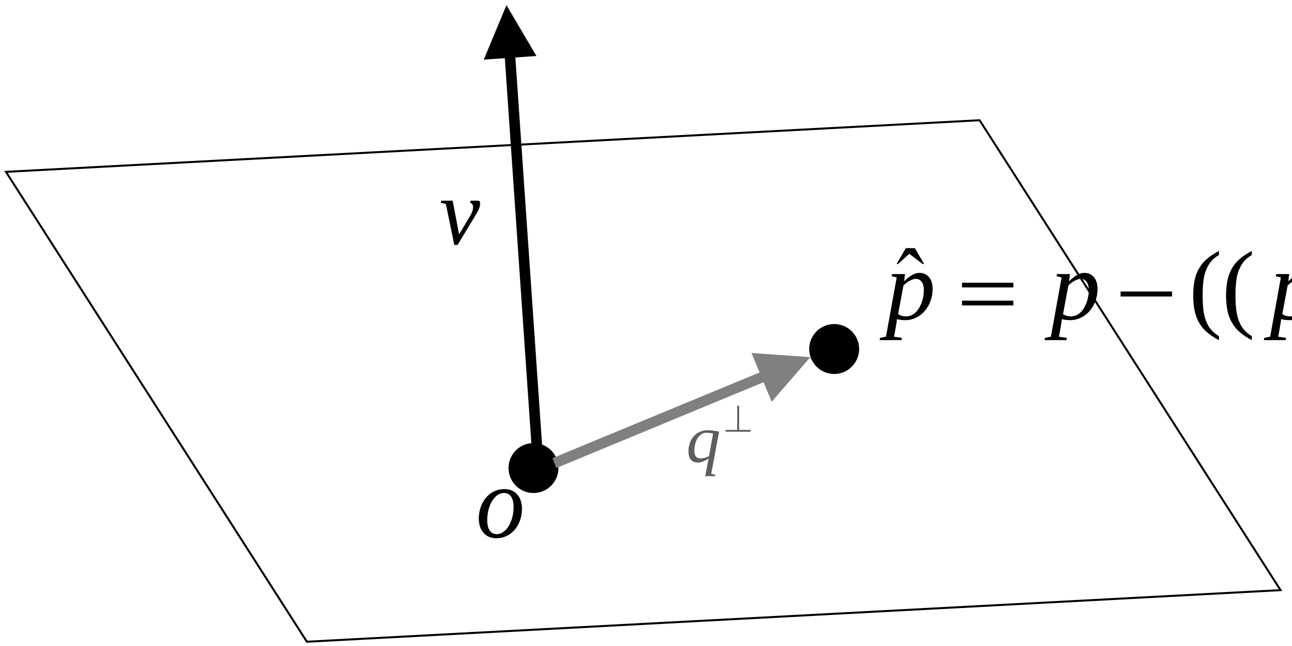
Orthogonal Projection



Orthogonal Projection

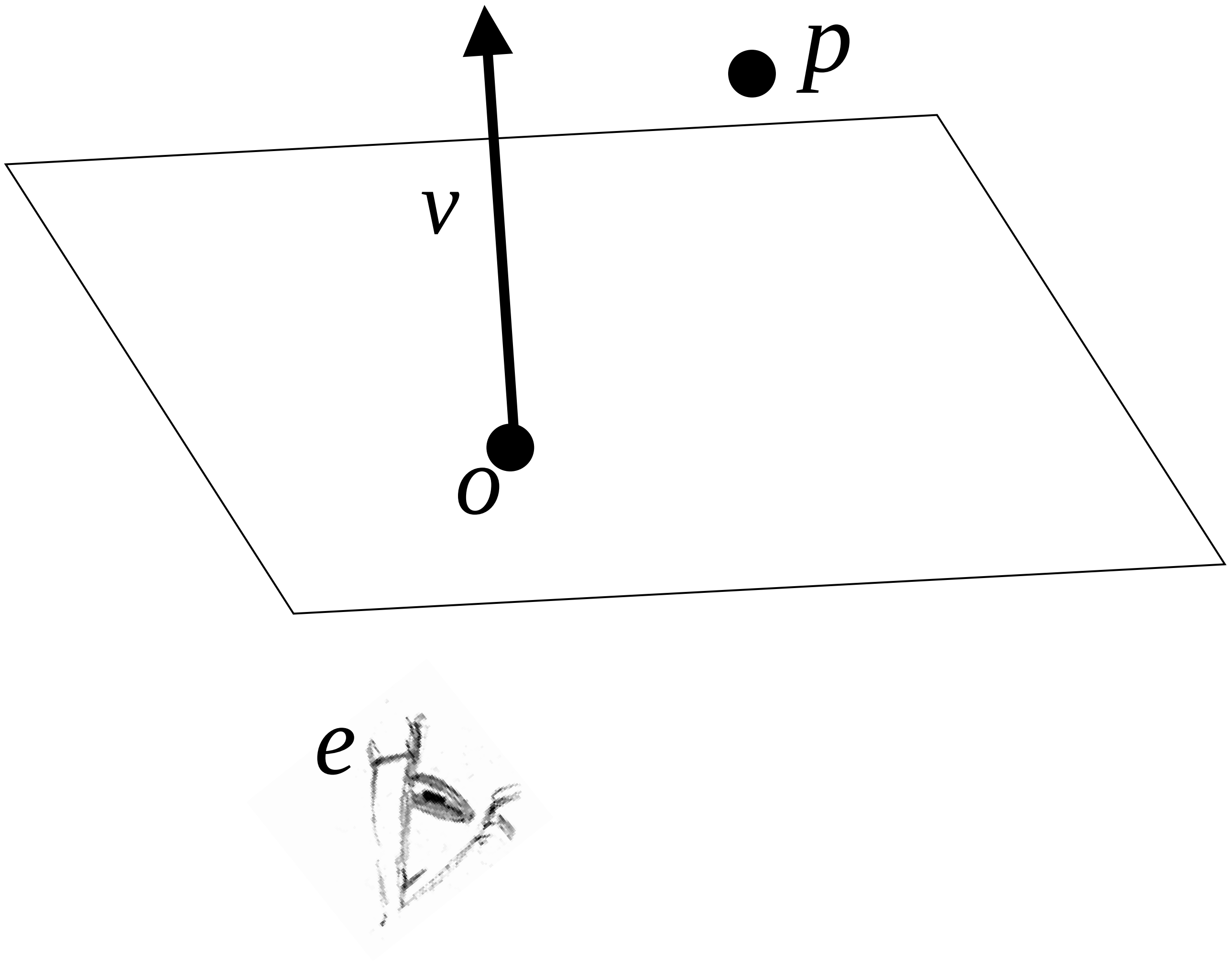


Orthogonal Projection

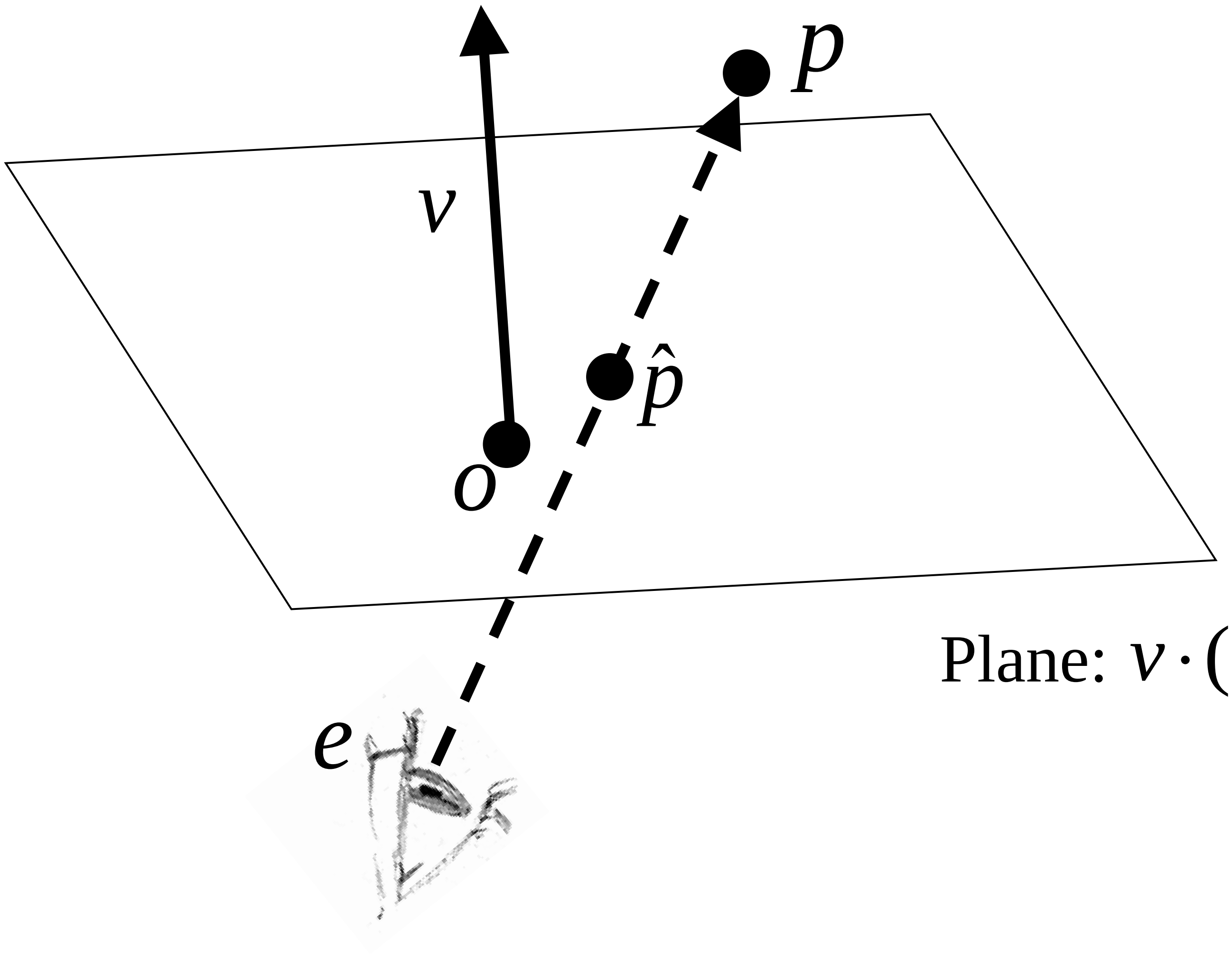


$$\hat{p} = p - ((p - o) \cdot v)v$$

Perspective Projection

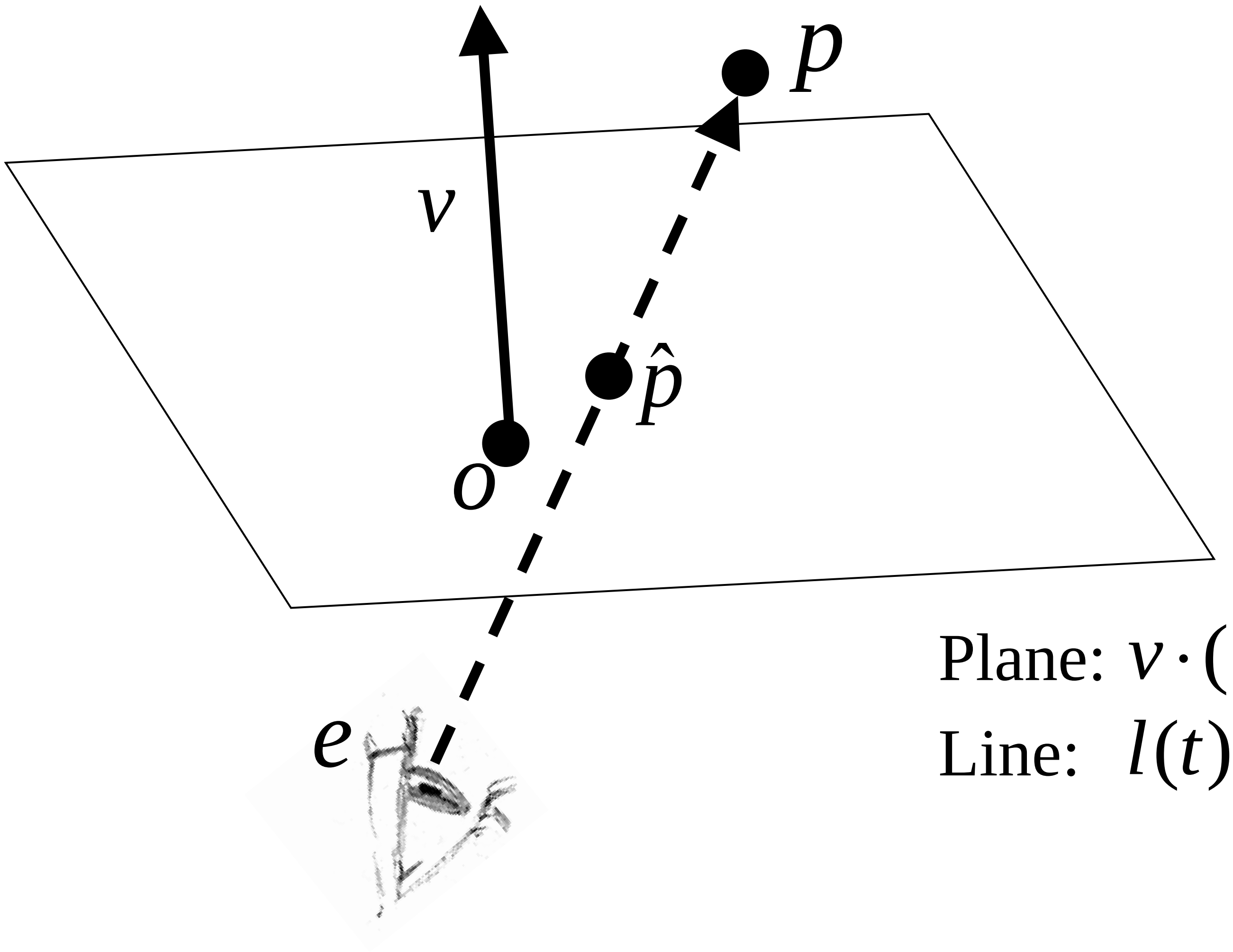


Perspective Projection



Plane: $v \cdot (x - o) = 0$

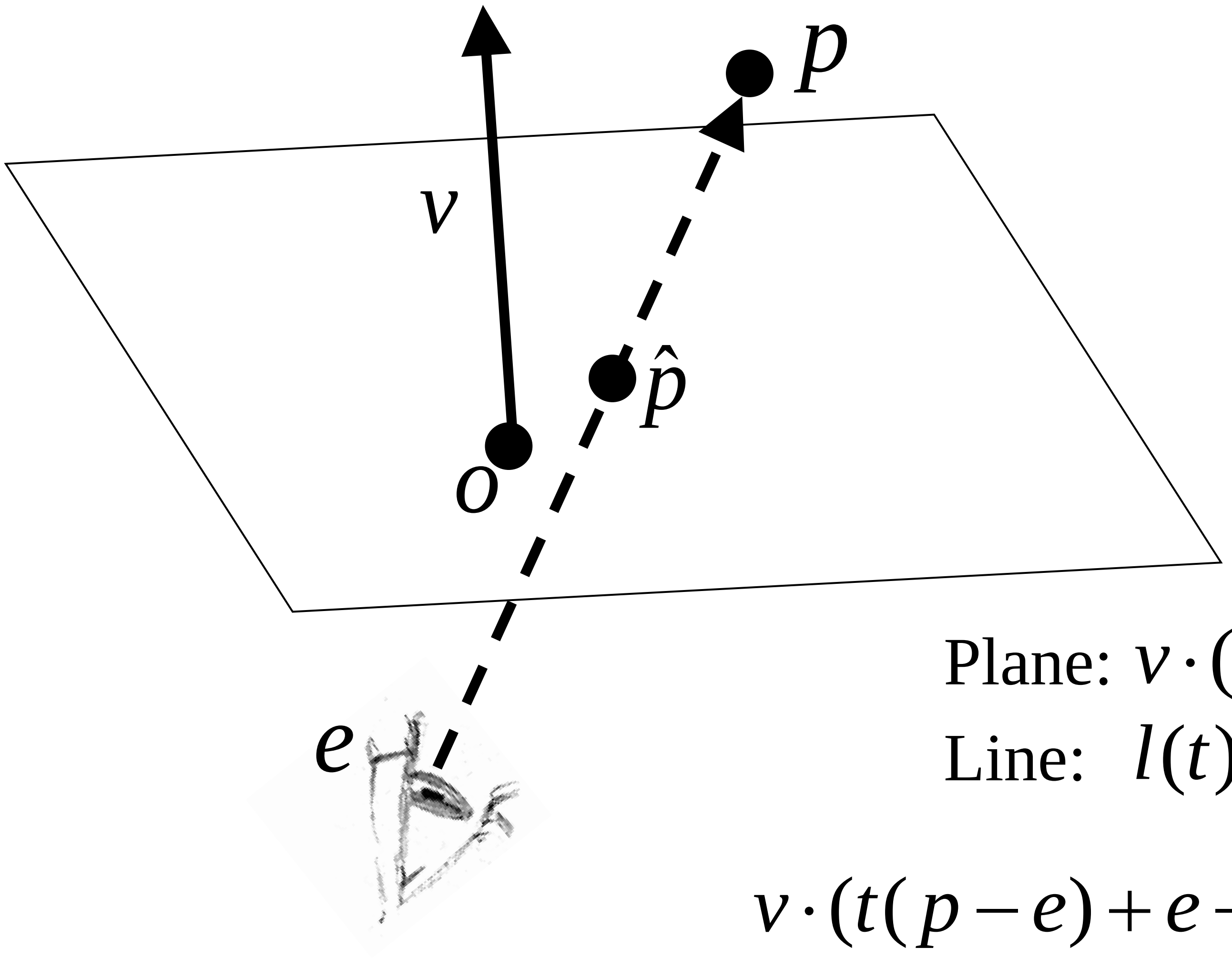
Perspective Projection



Plane: $v \cdot (x - o) = 0$

Line: $l(t) = (1-t)e + t\,p$

Perspective Projection

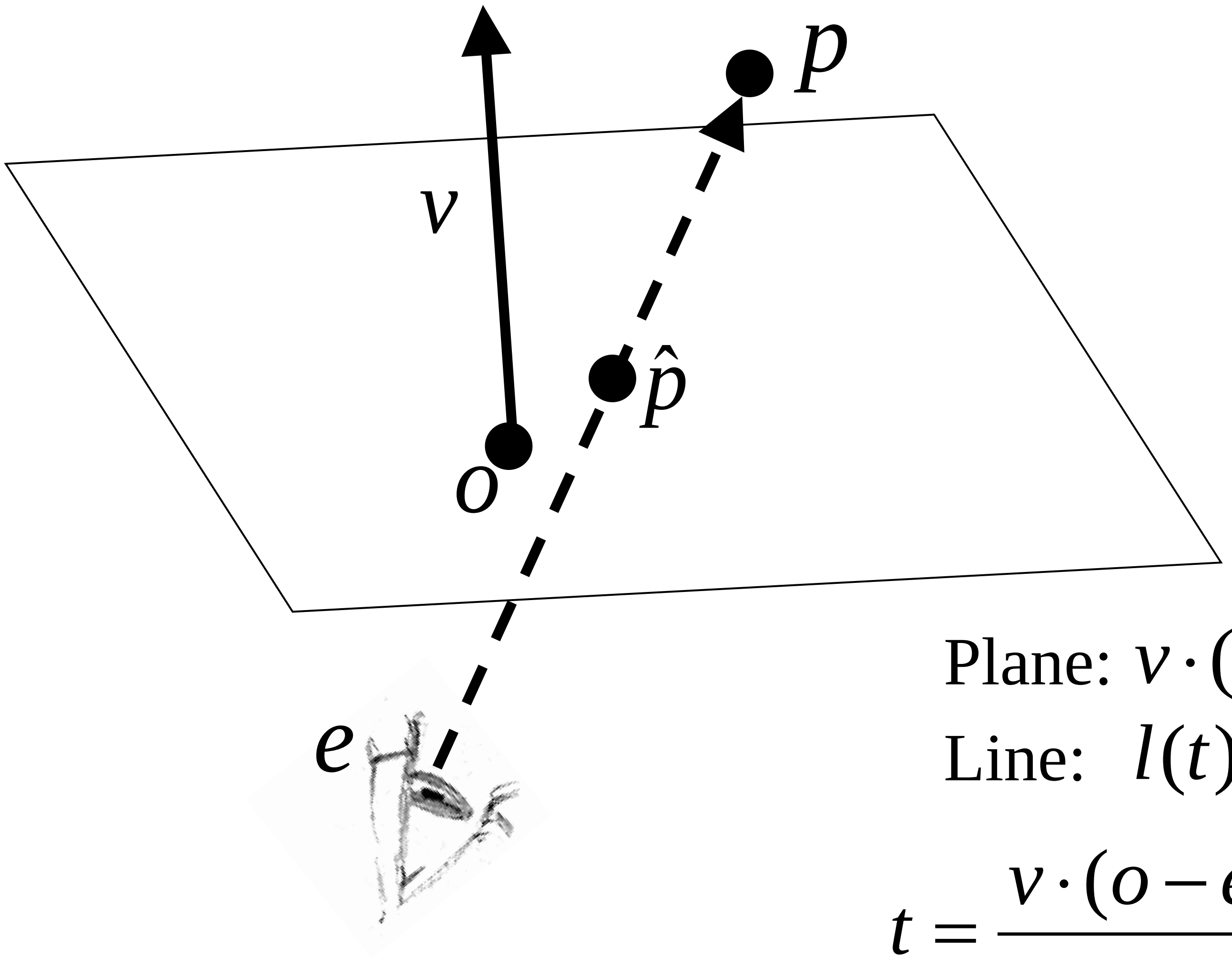


Plane: $v \cdot (x - o) = 0$

Line: $l(t) = (1 - t)e + t\,p$

$$v \cdot (t(p - e) + e - o) = 0$$

Perspective Projection

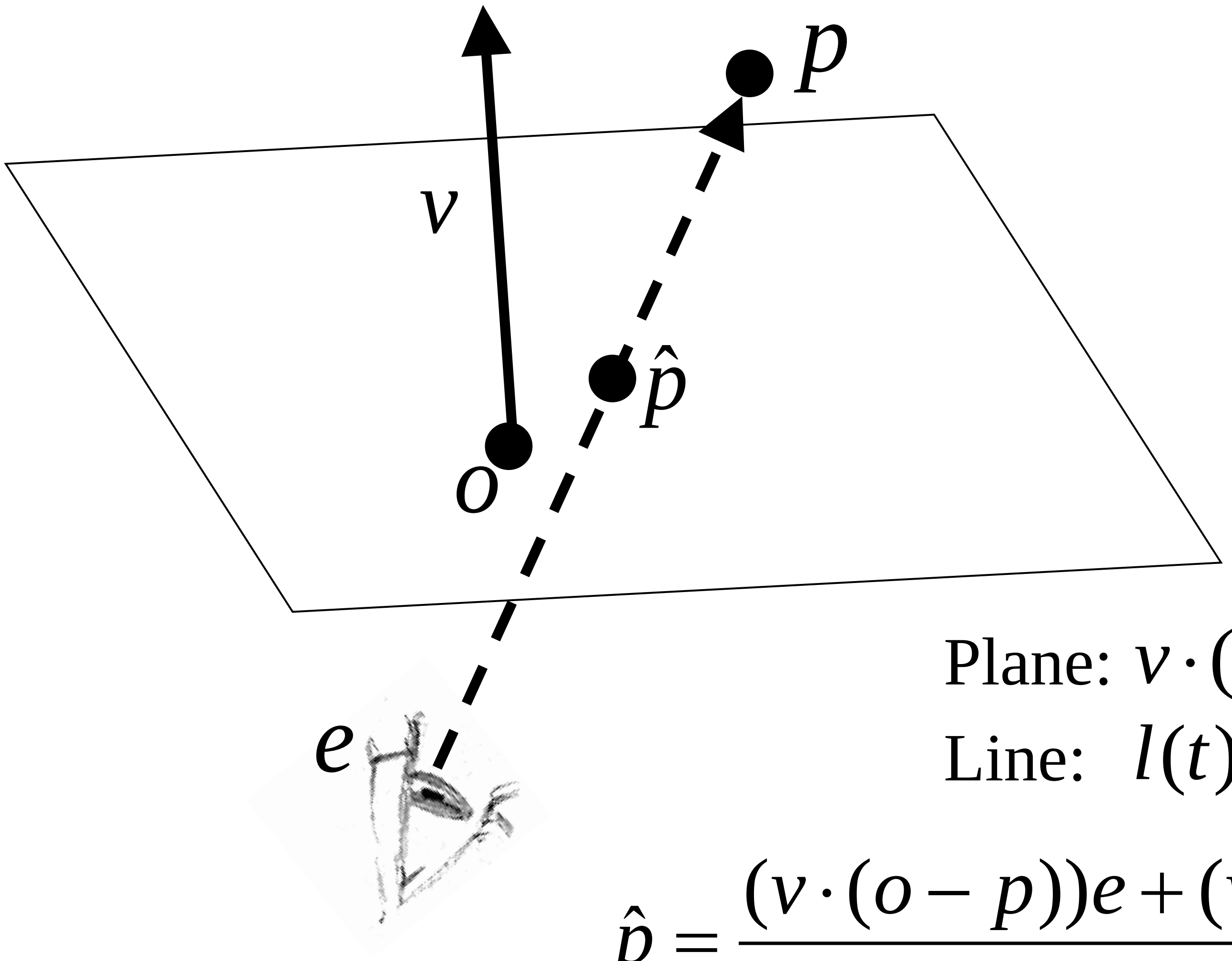


Plane: $v \cdot (x - o) = 0$

Line: $l(t) = (1 - t)e + t\,p$

$$t = \frac{v \cdot (o - e)}{v \cdot (p - e)}$$

Perspective Projection

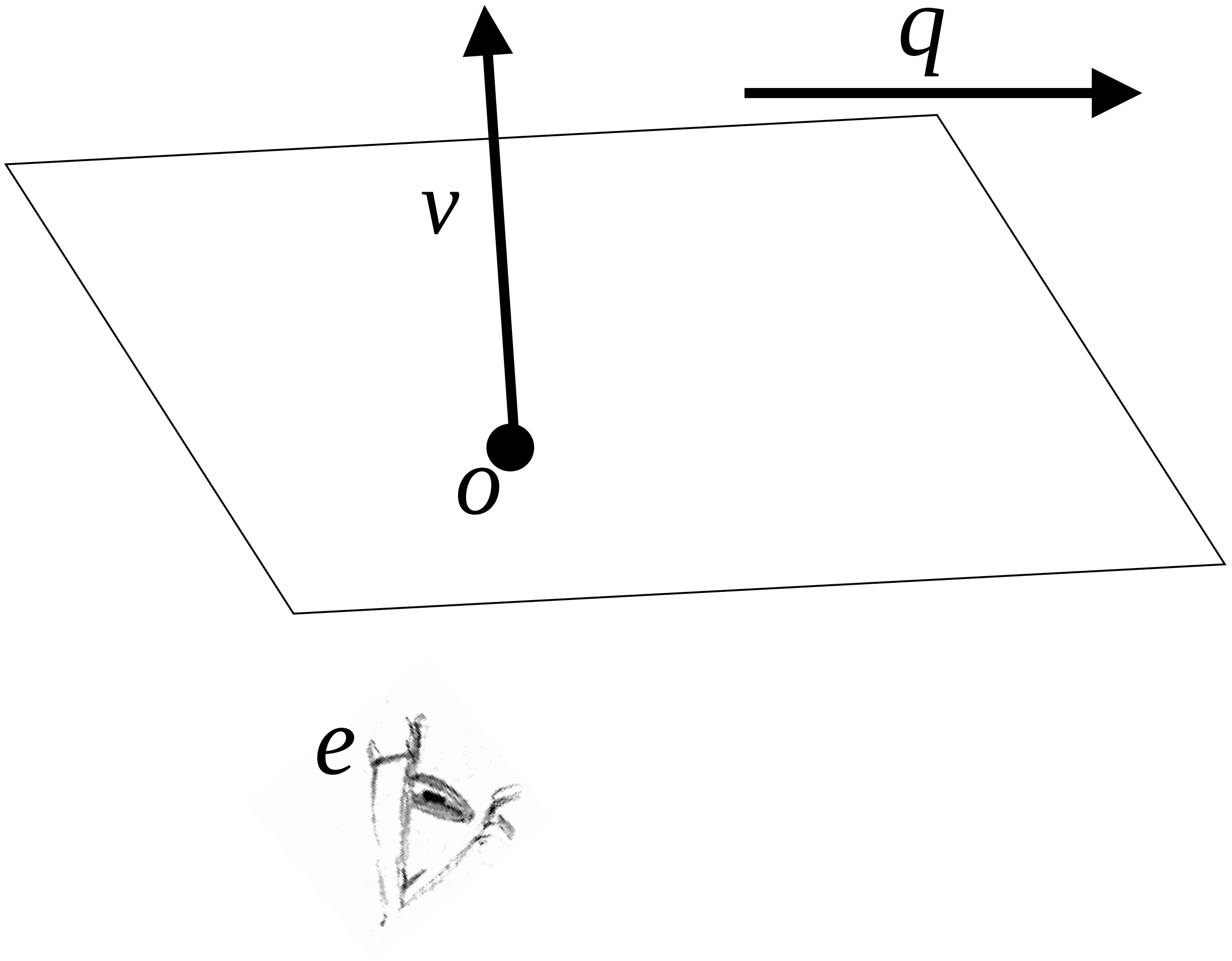


Plane: $v \cdot (x - o) = 0$

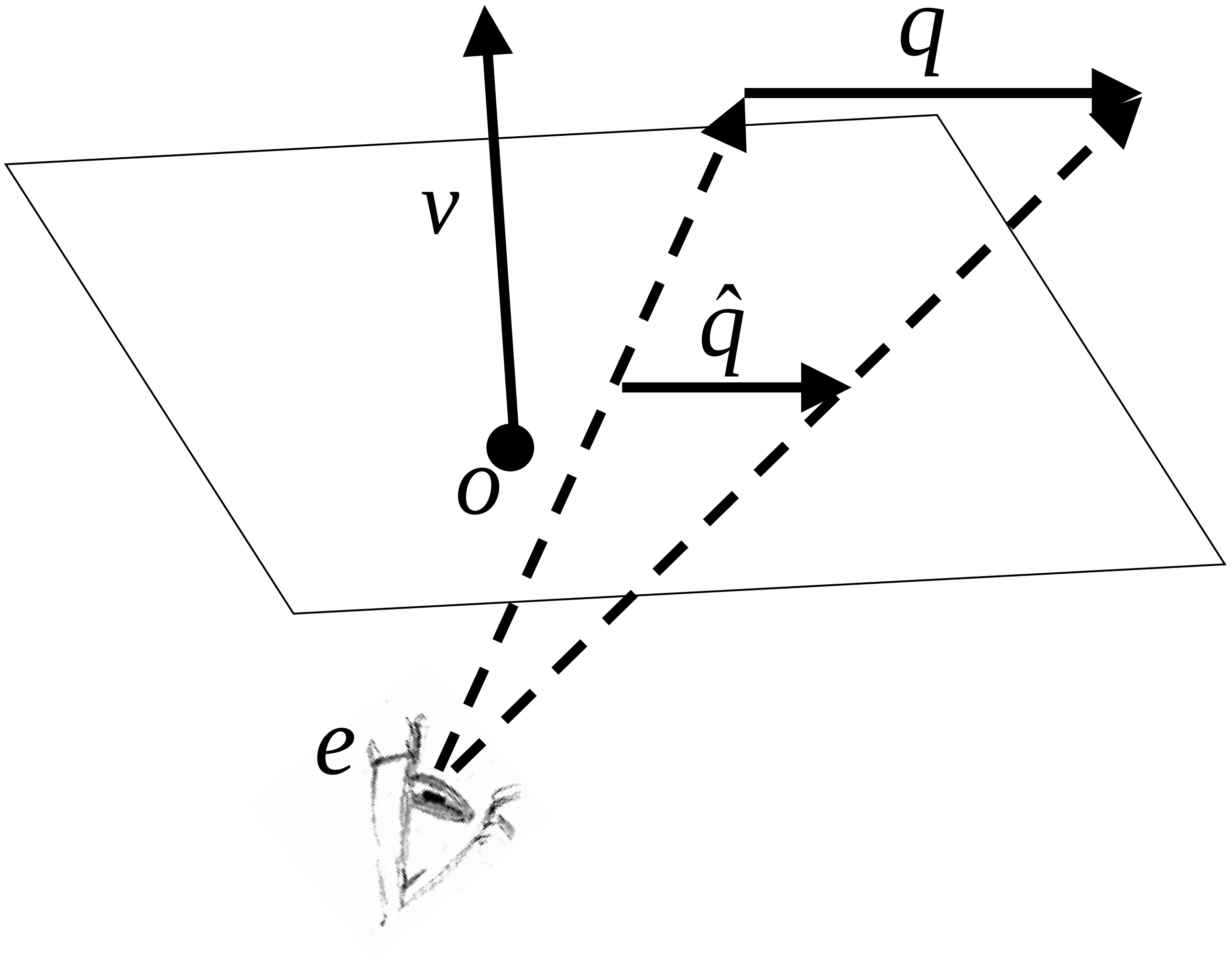
Line: $l(t) = (1 - t)e + t\,p$

$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))\,p}{v \cdot (e - p)}$$

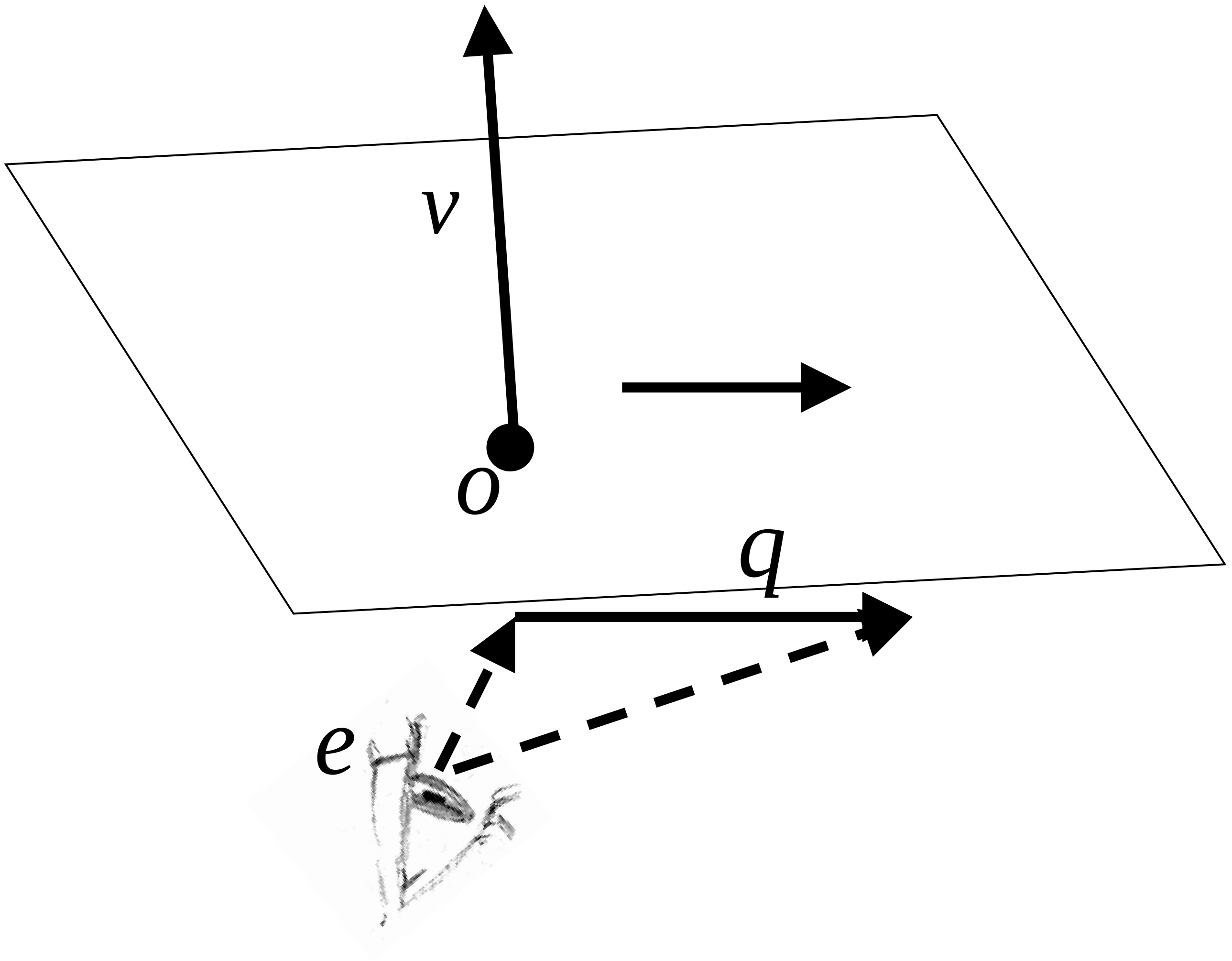
Perspective Projection



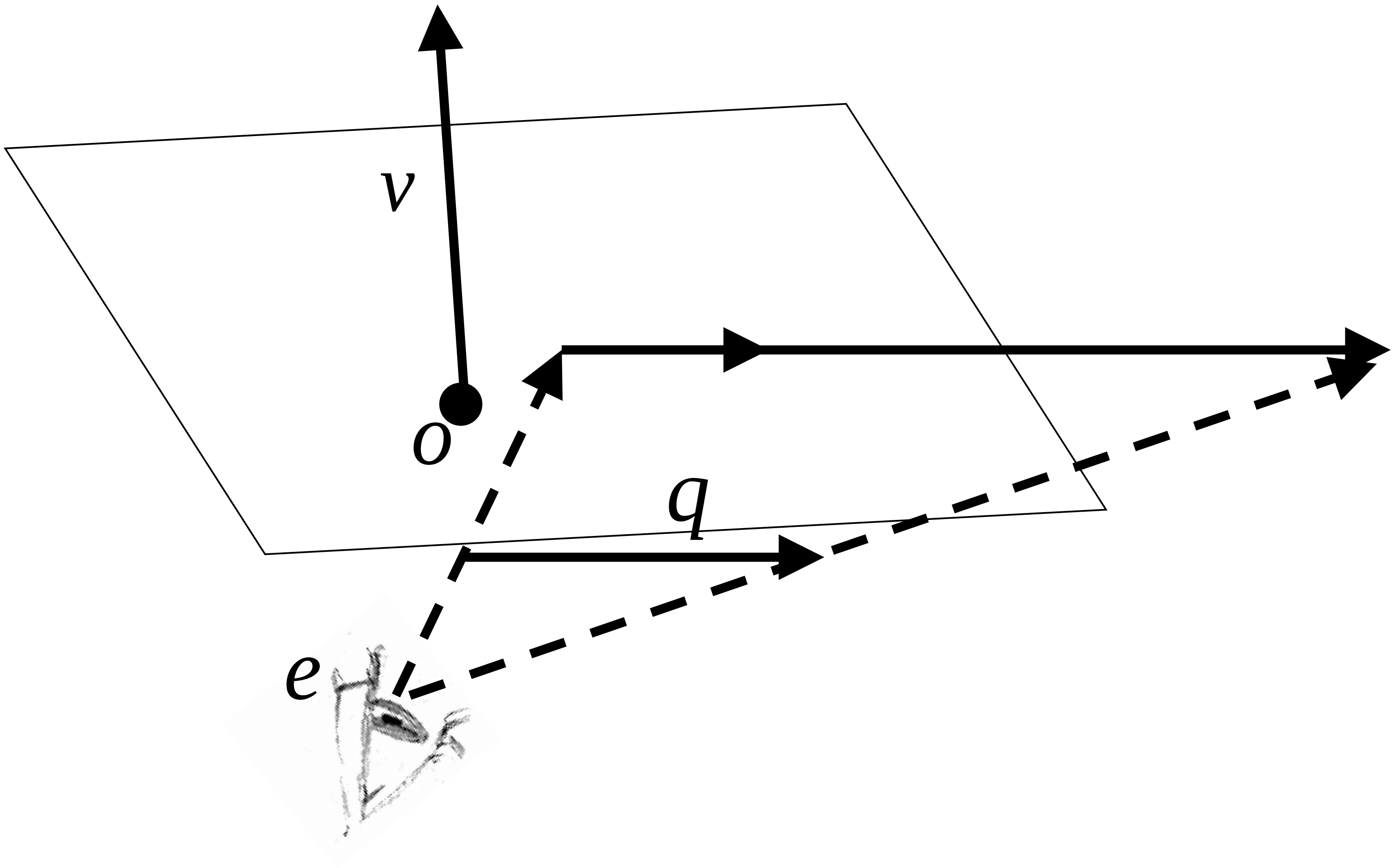
Perspective Projection



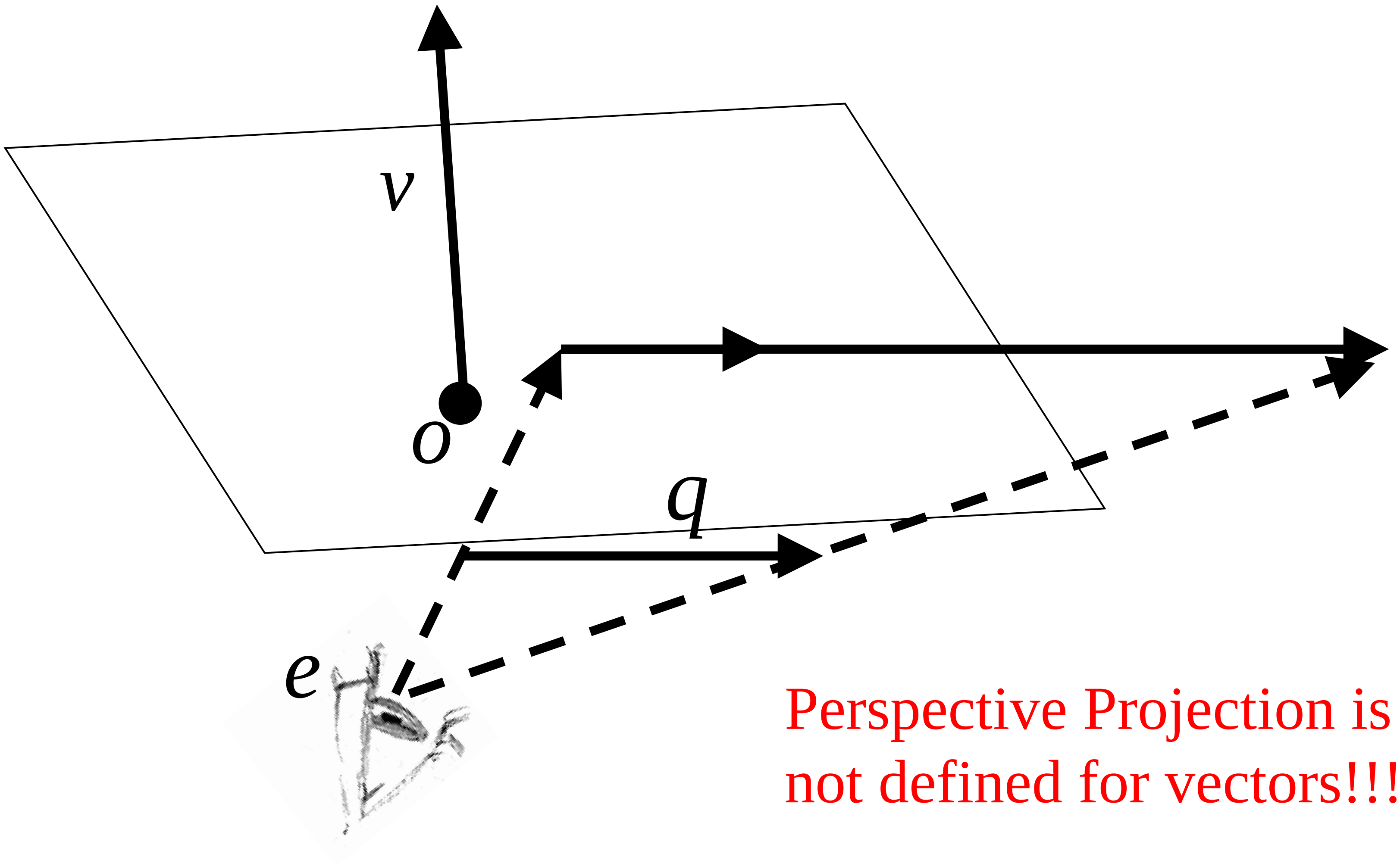
Perspective Projection



Perspective Projection



Perspective Projection



Perspective Projection is not defined for vectors!!!

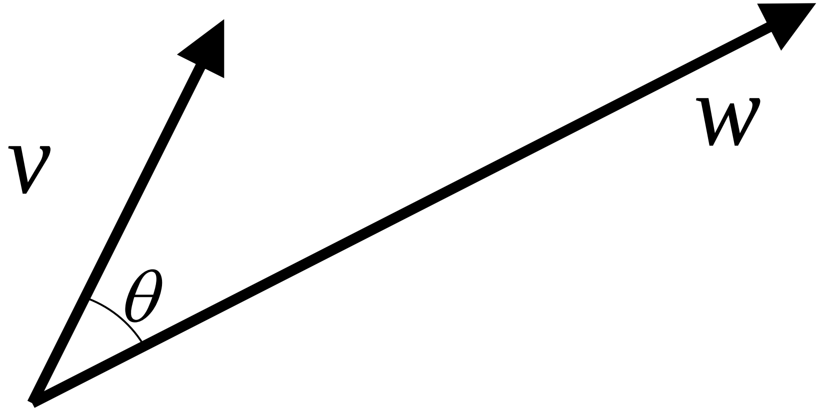
Transformations as Matrices

- Exactly like 2D
- 4x4 matrices
- Requires coordinates…. ☹

$$M = \begin{pmatrix} L & t \\ 0 & 1 \end{pmatrix}$$

Review – Vector Operations

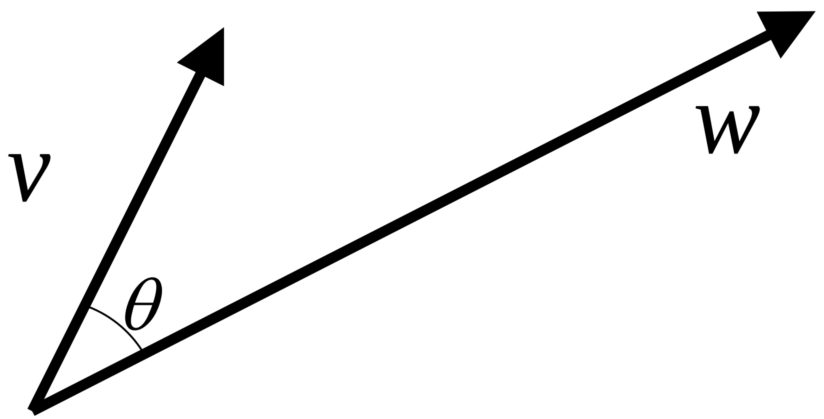
- Dot Product



$$v \cdot w = v_x w_x + v_y w_y + v_z w_z$$

Review – Vector Operations

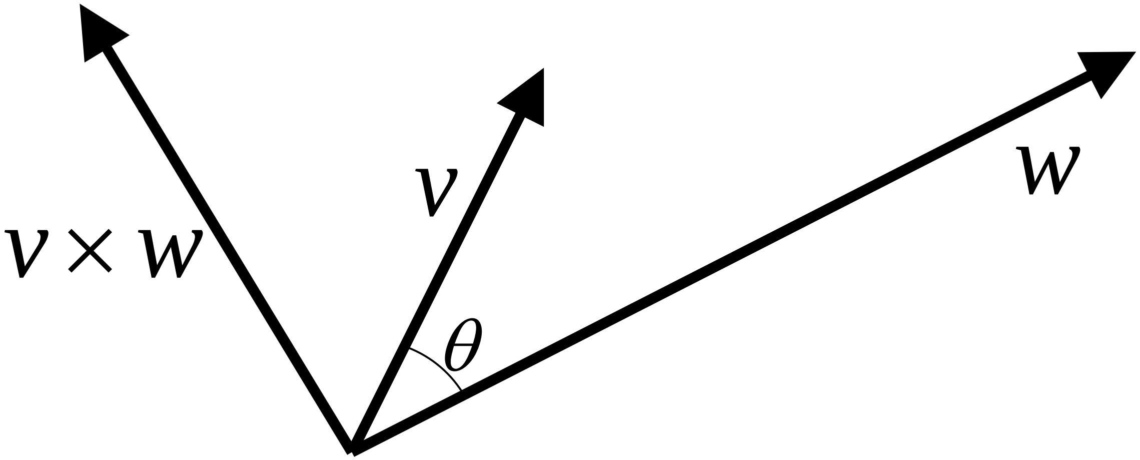
- Dot Product



$$v \cdot w = v^T w = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix} \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$$

Review – Vector Operations

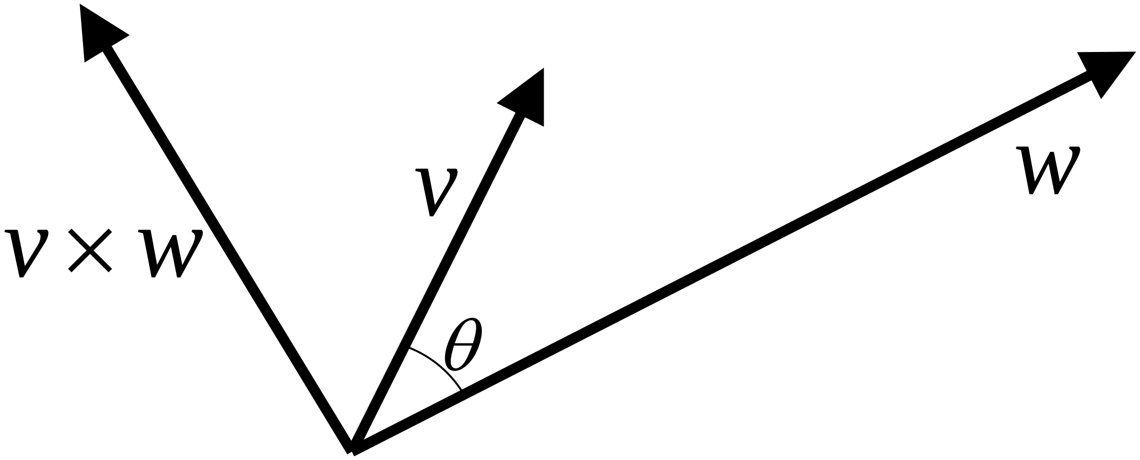
- Cross Product



$$v \times w = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{vmatrix} = (v_y w_z - v_z w_y, v_z w_x - v_x w_z, v_x w_y - v_y w_x)$$

Review – Vector Operations

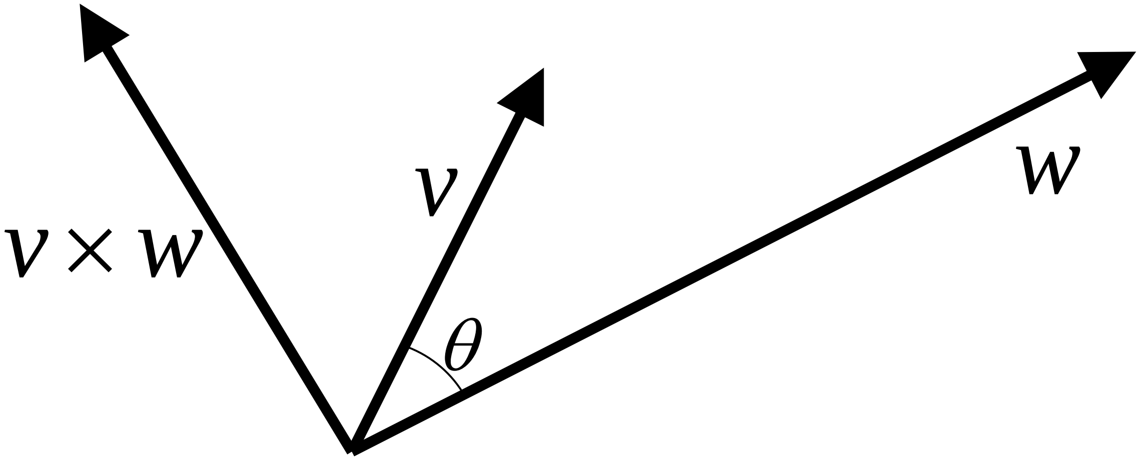
- Cross Product



$$v \times w = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} = (v_y w_z - v_z w_y, v_z w_x - v_x w_z, v_x w_y - v_y w_x)$$

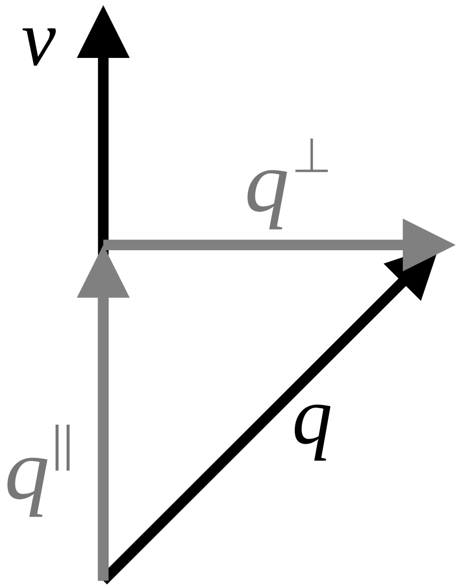
Review – Vector Operations

- Cross Product

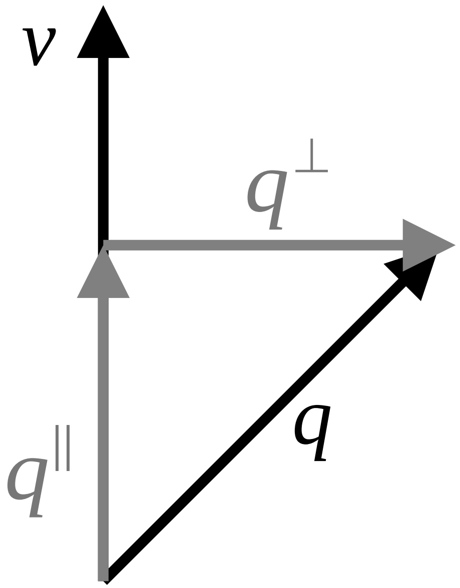


$$v \times \_ = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix}$$

Review – Vector Operations

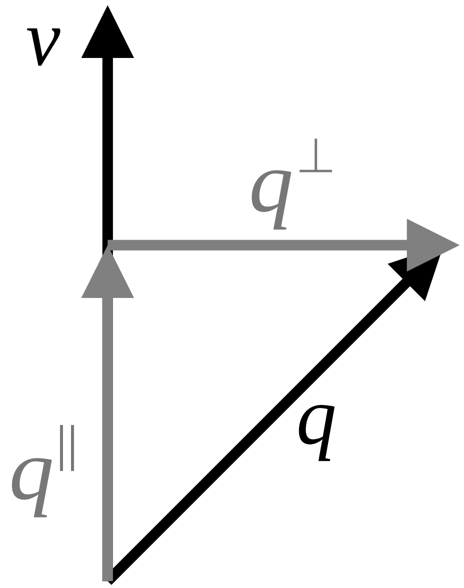


Review – Vector Operations



$$q^{\parallel} = v(v \cdot q)$$
$$q^{\perp} = q - q^{\parallel} = q - v(v \cdot q)$$

Review – Vector Operations



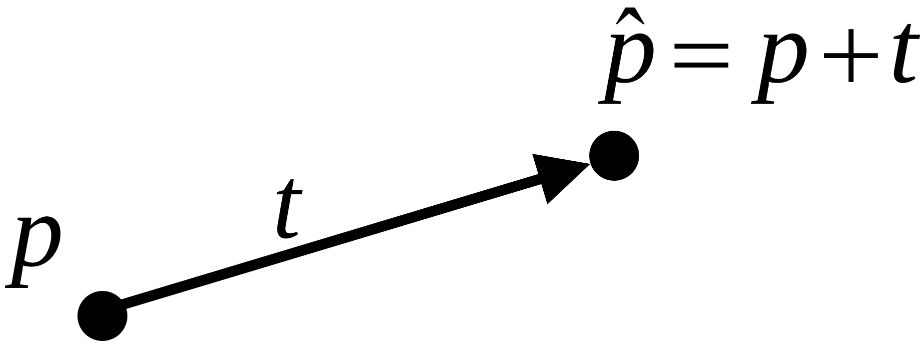
$$q^{\parallel} = v(v \cdot q)$$

$$q^{\perp} = q - q^{\parallel} = q - v(v \cdot q)$$

$$q^{\parallel} = v v^T q$$

$$q^{\perp} = (I - v v^T) q$$

Translation



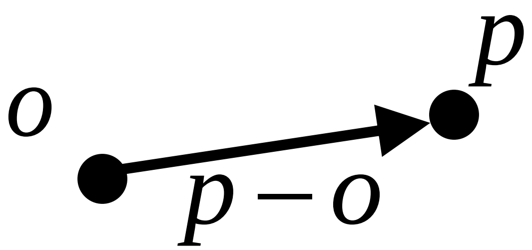
$$\begin{pmatrix} I & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Translation

$$\begin{pmatrix} I & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

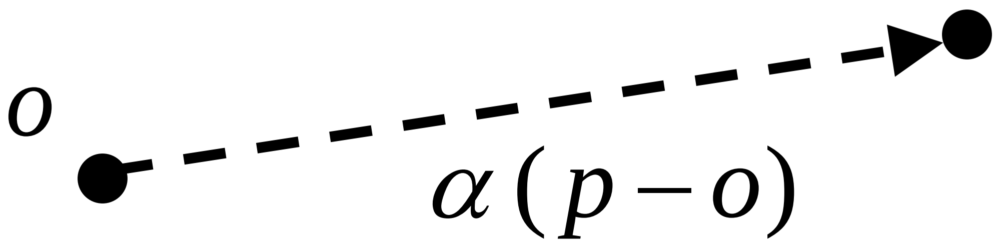
Uniform Scaling

Uniform Scaling



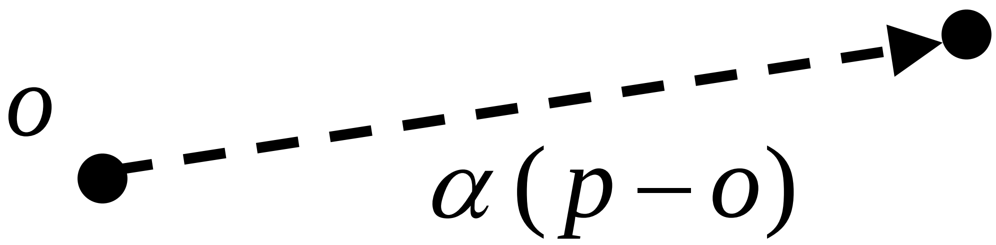
Uniform Scaling

$$\hat{p} = (1-\alpha)o + \alpha p$$



Uniform Scaling

$$\hat{p} = (1-\alpha)o + \alpha p$$



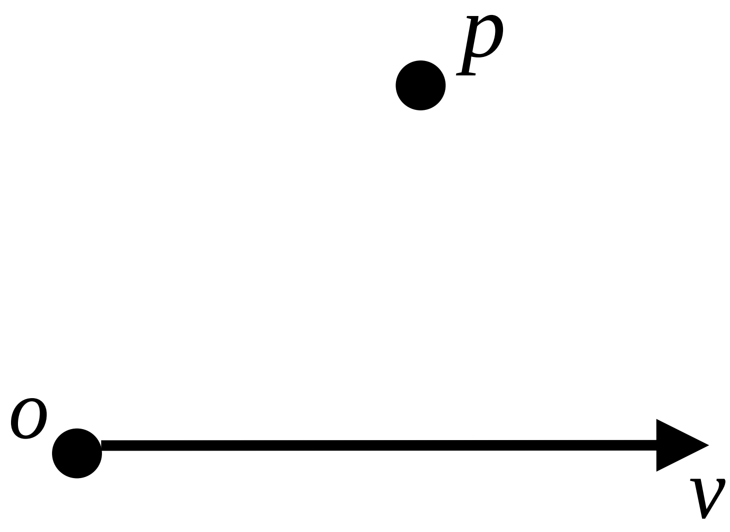
$$\begin{pmatrix} \alpha I & (1-\alpha)o \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Uniform Scaling

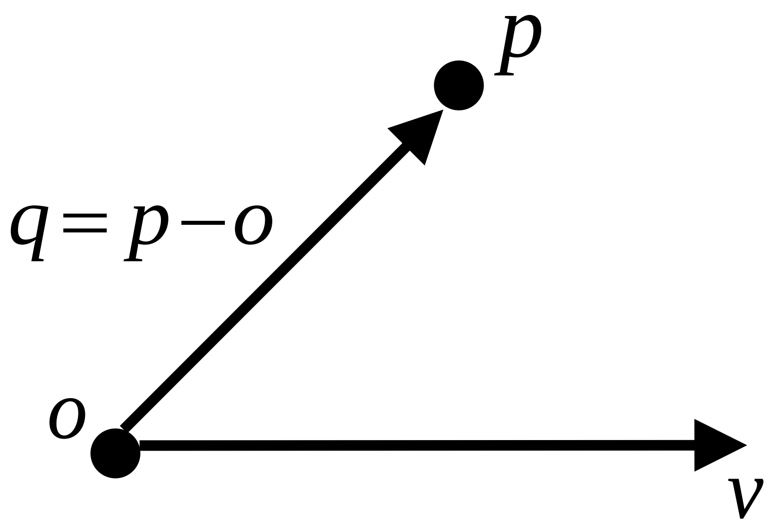
- $o = (0,0,0)^T$

$$\begin{pmatrix} \alpha I & (1-\alpha)o \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

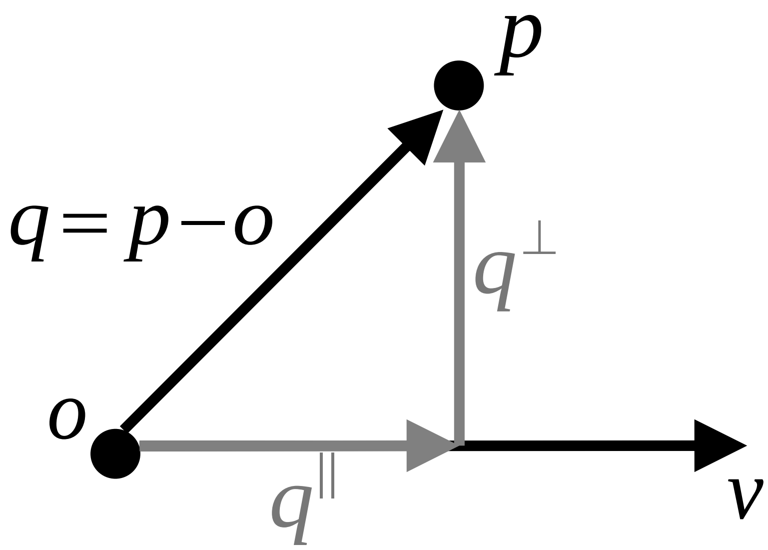
Non-Uniform Scaling



Non-Uniform Scaling

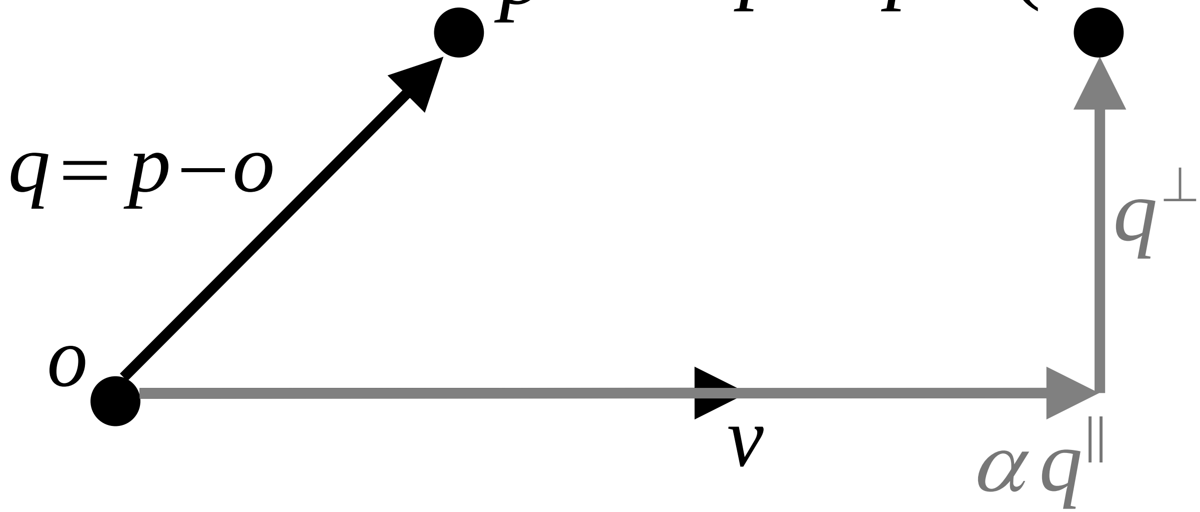


Non-Uniform Scaling



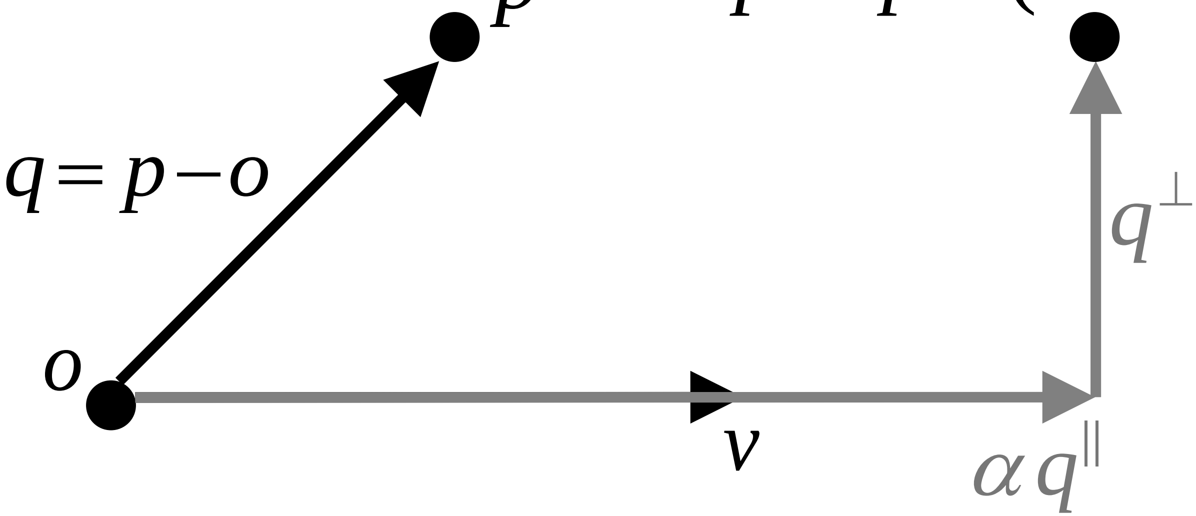
Non-Uniform Scaling

$$\hat{p} = p + (\alpha - 1)(v \cdot (p - o))v$$



Non-Uniform Scaling

$$\hat{p} = p + (\alpha - 1)(v \cdot (p - o))v$$



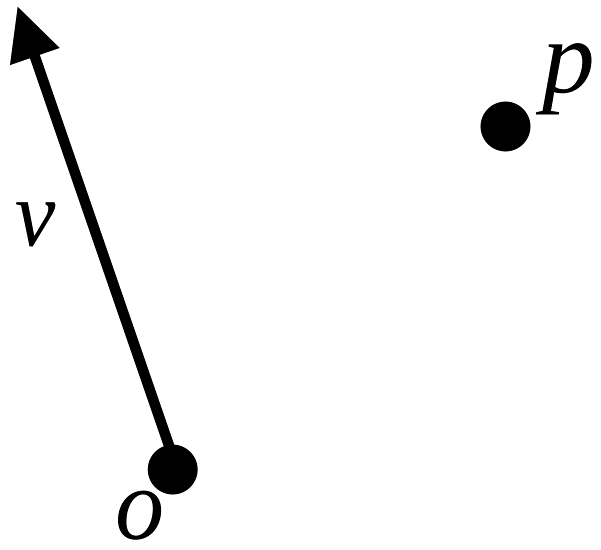
$$\begin{pmatrix} I + (\alpha - 1)vv^T & (1 - \alpha)vv^T o \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Non-Uniform Scaling

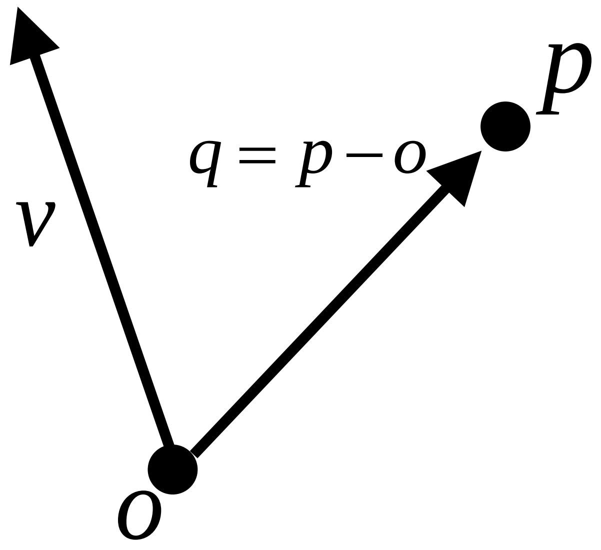
- $v = (1,0,0)^T$, $o = (0,0,0)^T$

$$\begin{pmatrix} I + (\alpha - 1) v v^T & (1-\alpha) v v^T o \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

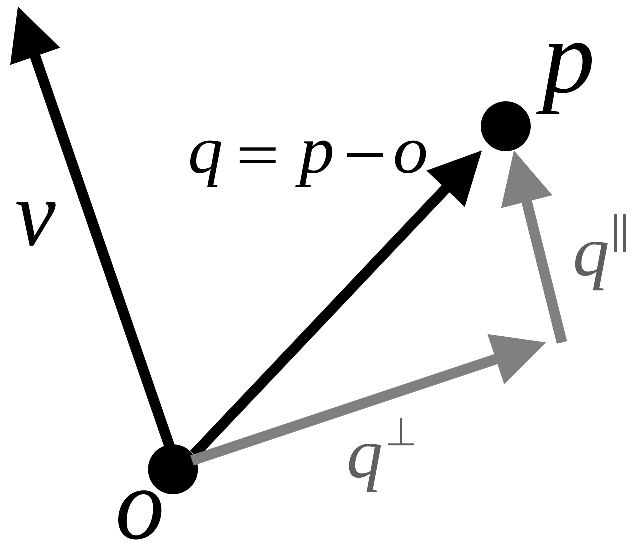
Rotation



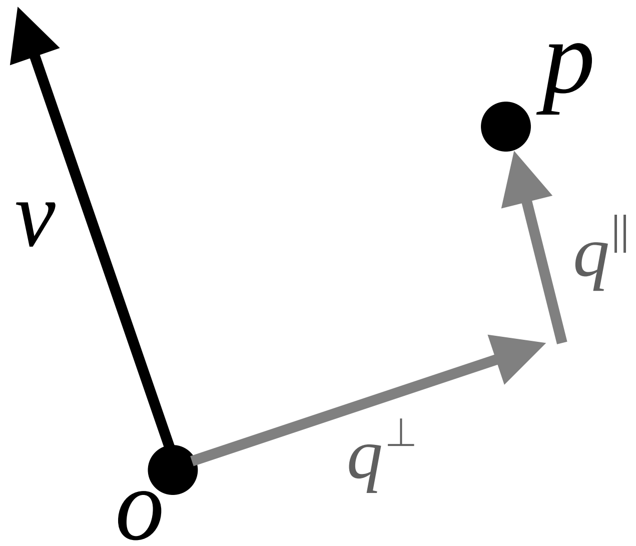
Rotation



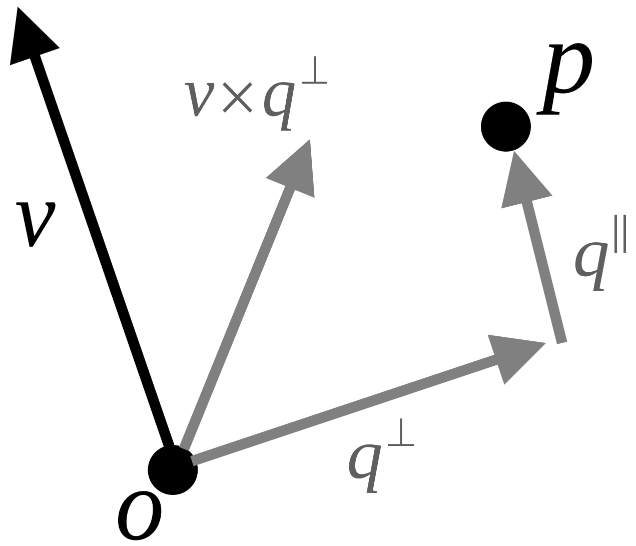
Rotation



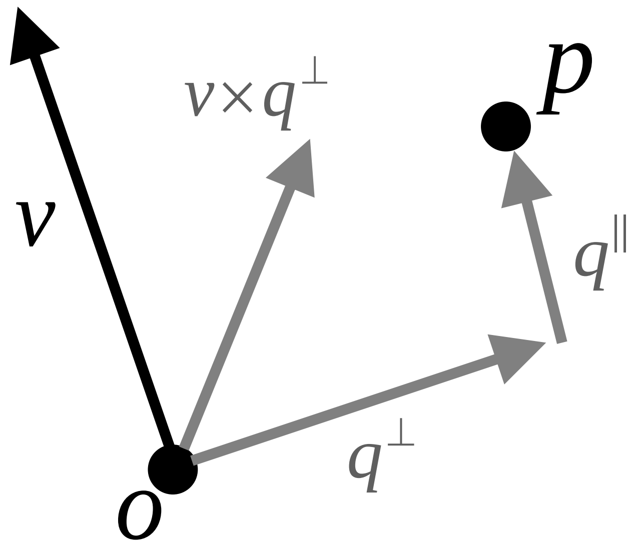
Rotation

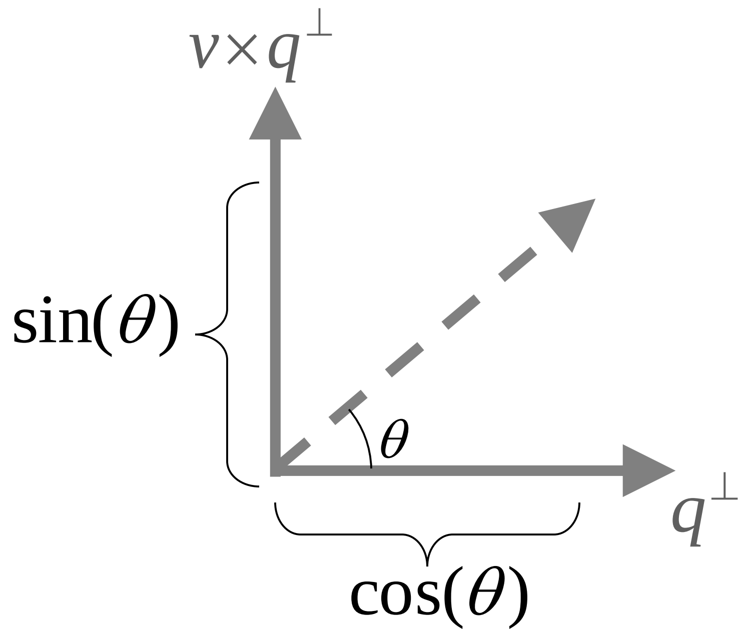


Rotation

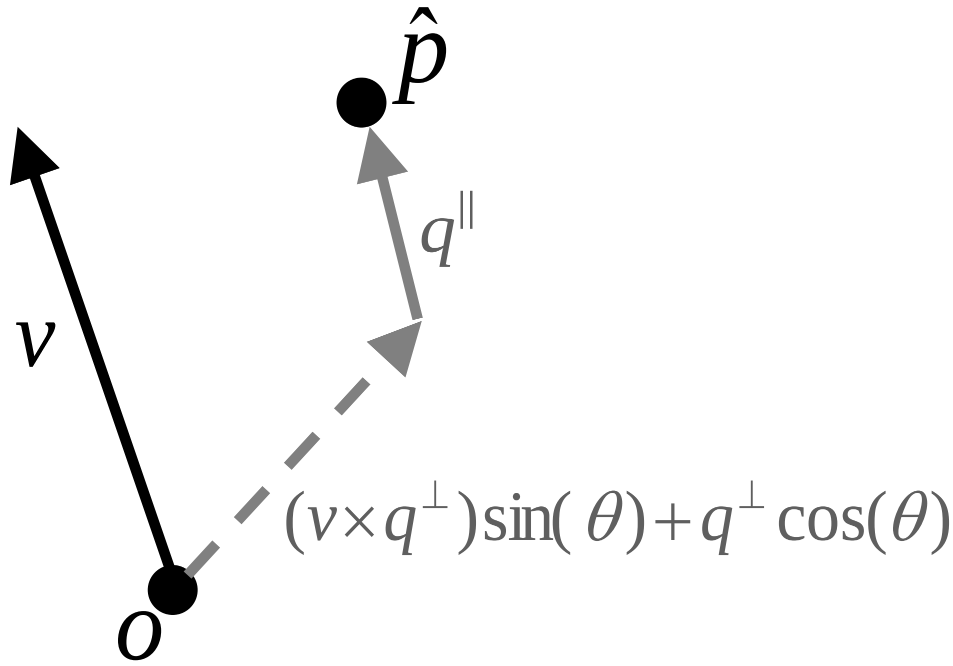


Rotation


v
$v \times q^{\perp}$
p
$q^{\parallel}$
$q^{\perp}$
o

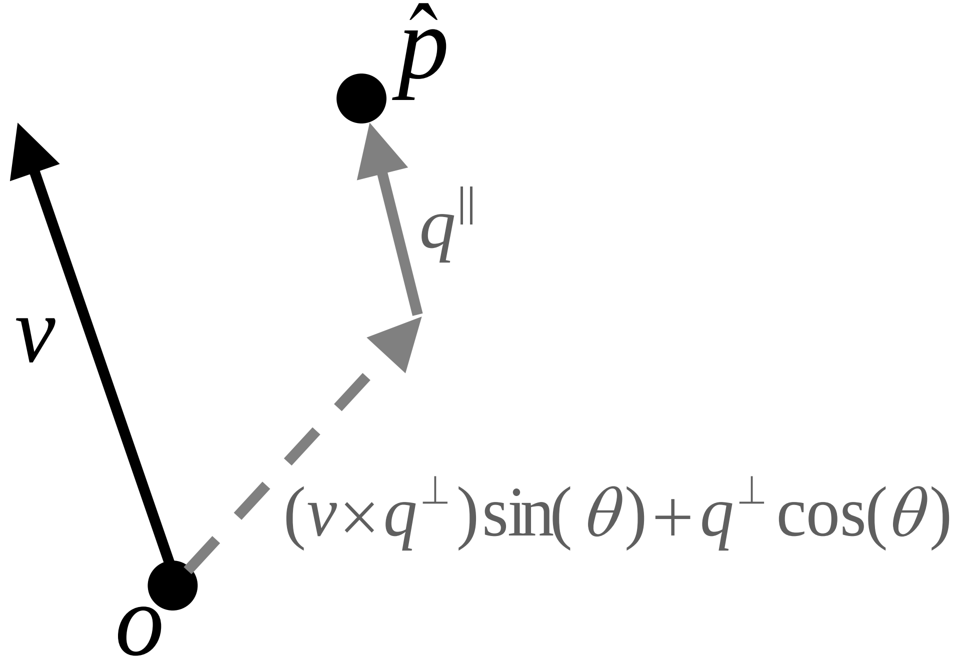

$v \times q^{\perp}$
$\sin(\theta)$
θ
$q^{\perp}$
$\cos(\theta)$

Rotation



$$\hat{p}=o+(1-\cos(\theta))(v\cdot q)v+(v\times q)\sin(\theta)+q\cos(\theta)$$

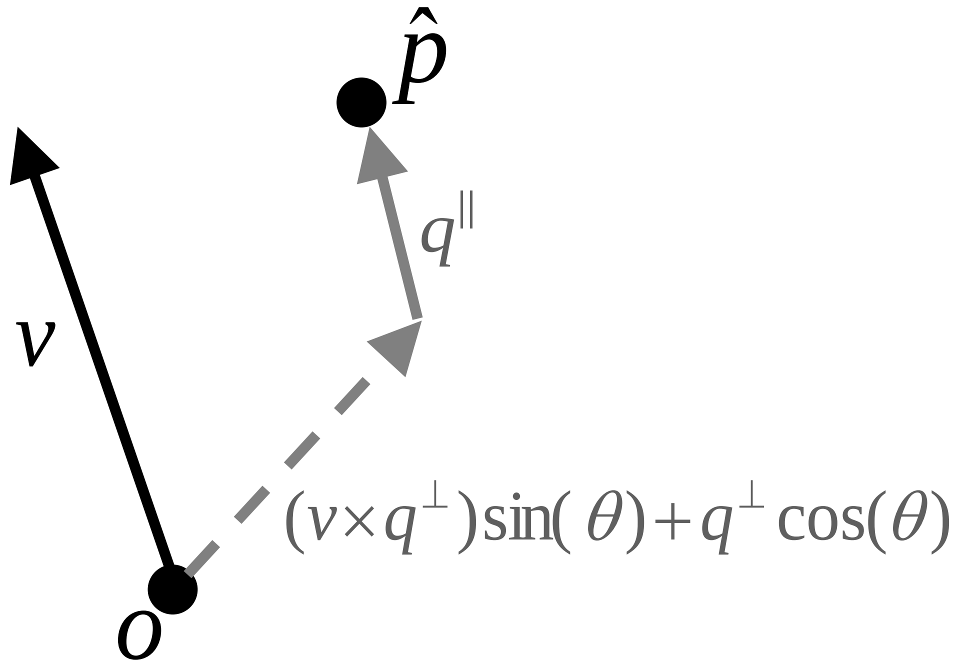
Rotation



$$\hat{p}=o+(1-\cos(\theta))(v\cdot q)v+(v\times q)\sin(\theta)+q\cos(\theta)$$

$$\begin{pmatrix} I & o \\ 0 & 1 \end{pmatrix}\begin{pmatrix} (1-\cos(\theta))vv^T+\sin(\theta)v\times\_+\cos(\theta)I & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} I & -o \\ 0 & 1 \end{pmatrix}\begin{pmatrix} p \\ 1 \end{pmatrix}=\begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Rotation



$$\hat{p}=o+(1-\cos(\theta))(v\cdot q)v+(v\times q)\sin(\theta)+q\cos(\theta)$$

$$\begin{pmatrix} I & o \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c+(1-c)v_x^{\;2} & (1-c)v_x v_y - s\,v_z & (1-c)v_x v_z + s\,v_y & 0 \\ (1-c)v_x v_y + s\,v_z & c+(1-c)v_y^{\;2} & (1-c)v_y v_z - s\,v_x & 0 \\ (1-c)v_x v_z - s\,v_y & (1-c)v_y v_z + s\,v_x & c+(1-c)v_z^{\;2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} I & -o \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Rotation

- Rotation about x

$$\begin{pmatrix} c+(1-c)v_x^{\,2} & (1-c)v_xv_y - s\,v_z & (1-c)v_xv_z + s\,v_y & 0 \\ (1-c)v_xv_y + s\,v_z & c+(1-c)v_y^{\,2} & (1-c)v_yv_z - s\,v_x & 0 \\ (1-c)v_xv_z - s\,v_y & (1-c)v_yv_z + s\,v_x & c+(1-c)v_z^{\,2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

- Rotation about y

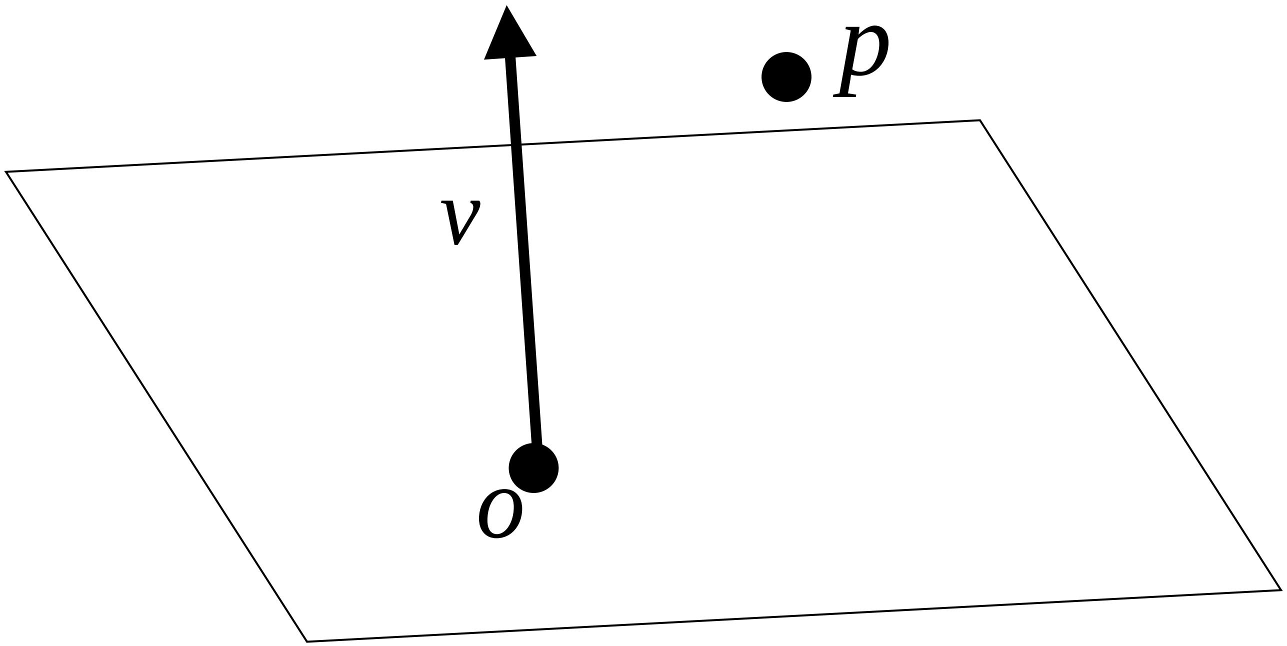
$$\begin{pmatrix} c+(1-c)v_x^{\;2} & (1-c)v_xv_y - s\,v_z & (1-c)v_xv_z + s\,v_y & 0 \\ (1-c)v_xv_y + s\,v_z & c+(1-c)v_y^{\;2} & (1-c)v_yv_z - s\,v_x & 0 \\ (1-c)v_xv_z - s\,v_y & (1-c)v_yv_z + s\,v_x & c+(1-c)v_z^{\;2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

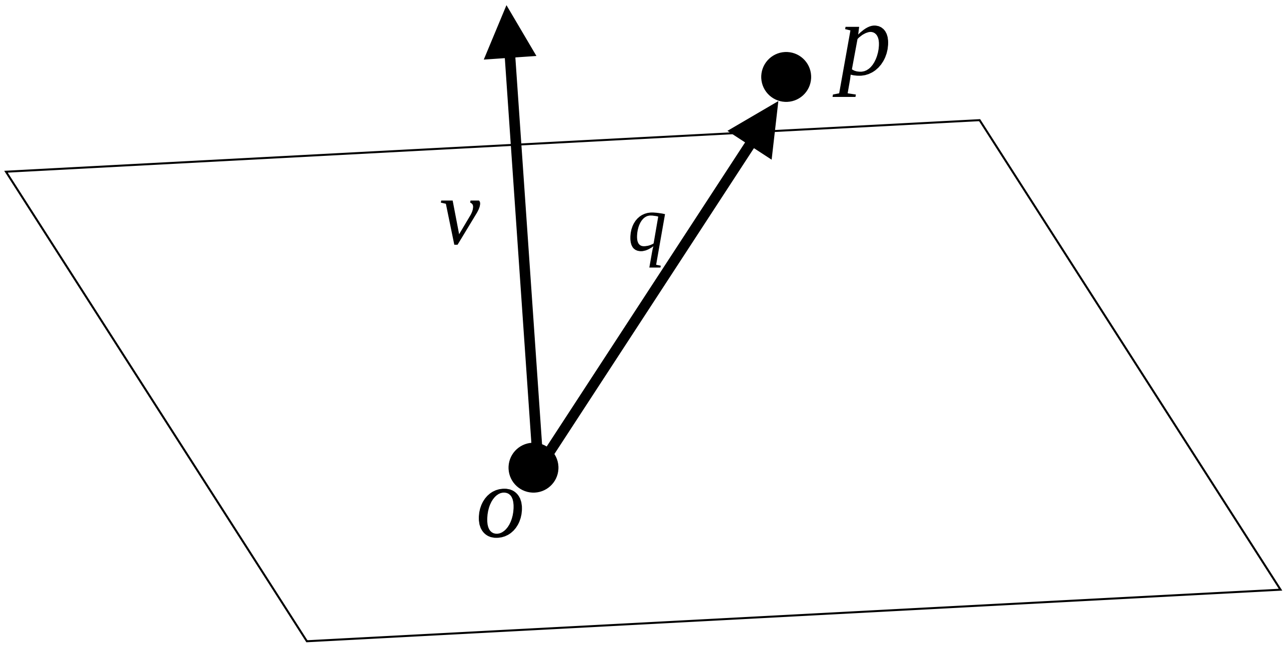
- Rotation about z

$$\begin{pmatrix} c+(1-c)v_x^{\ 2} & (1-c)v_xv_y - s\,v_z & (1-c)v_xv_z + s\,v_y & 0 \\ (1-c)v_xv_y + s\,v_z & c+(1-c)v_y^{\ 2} & (1-c)v_yv_z - s\,v_x & 0 \\ (1-c)v_xv_z - s\,v_y & (1-c)v_yv_z + s\,v_x & c+(1-c)v_z^{\ 2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

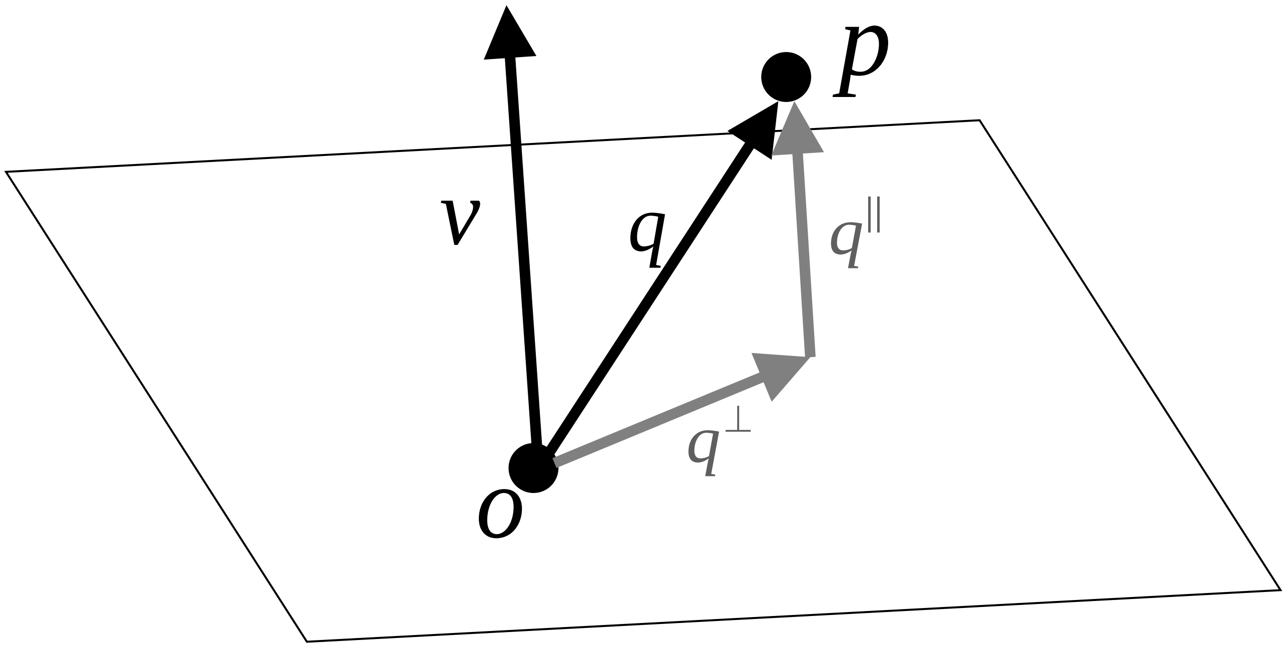
Mirror Image



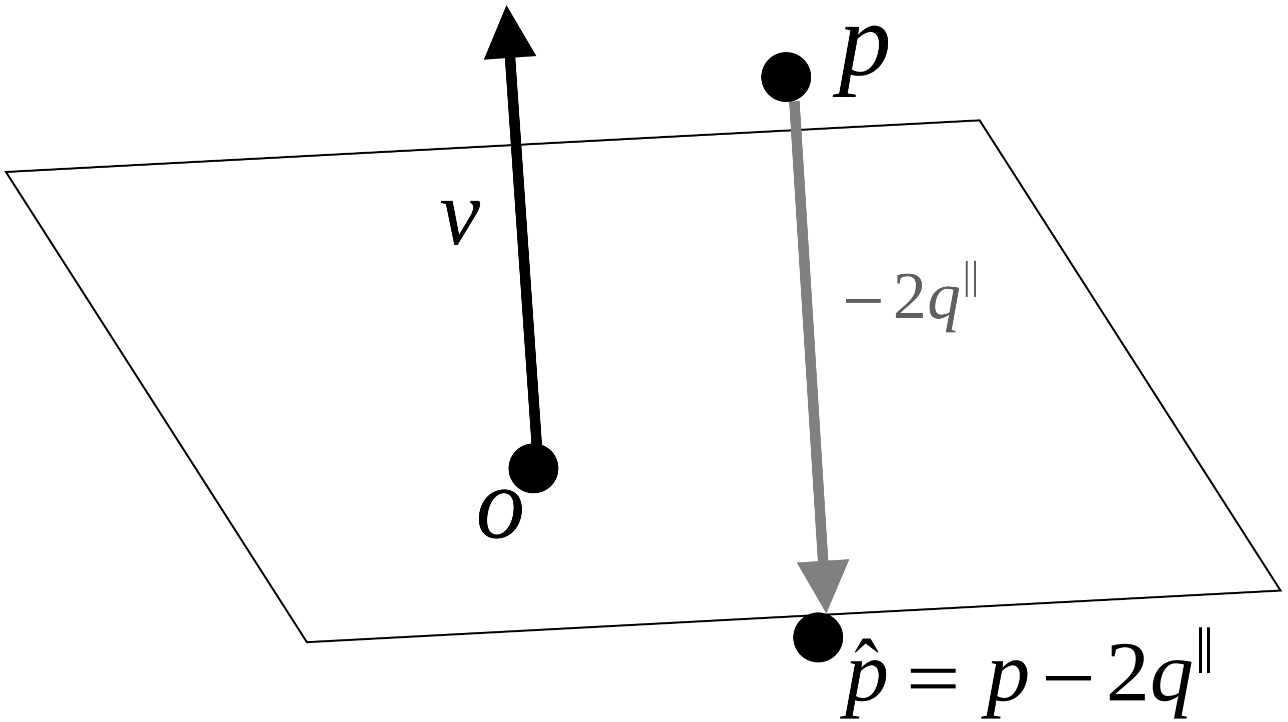
Mirror Image



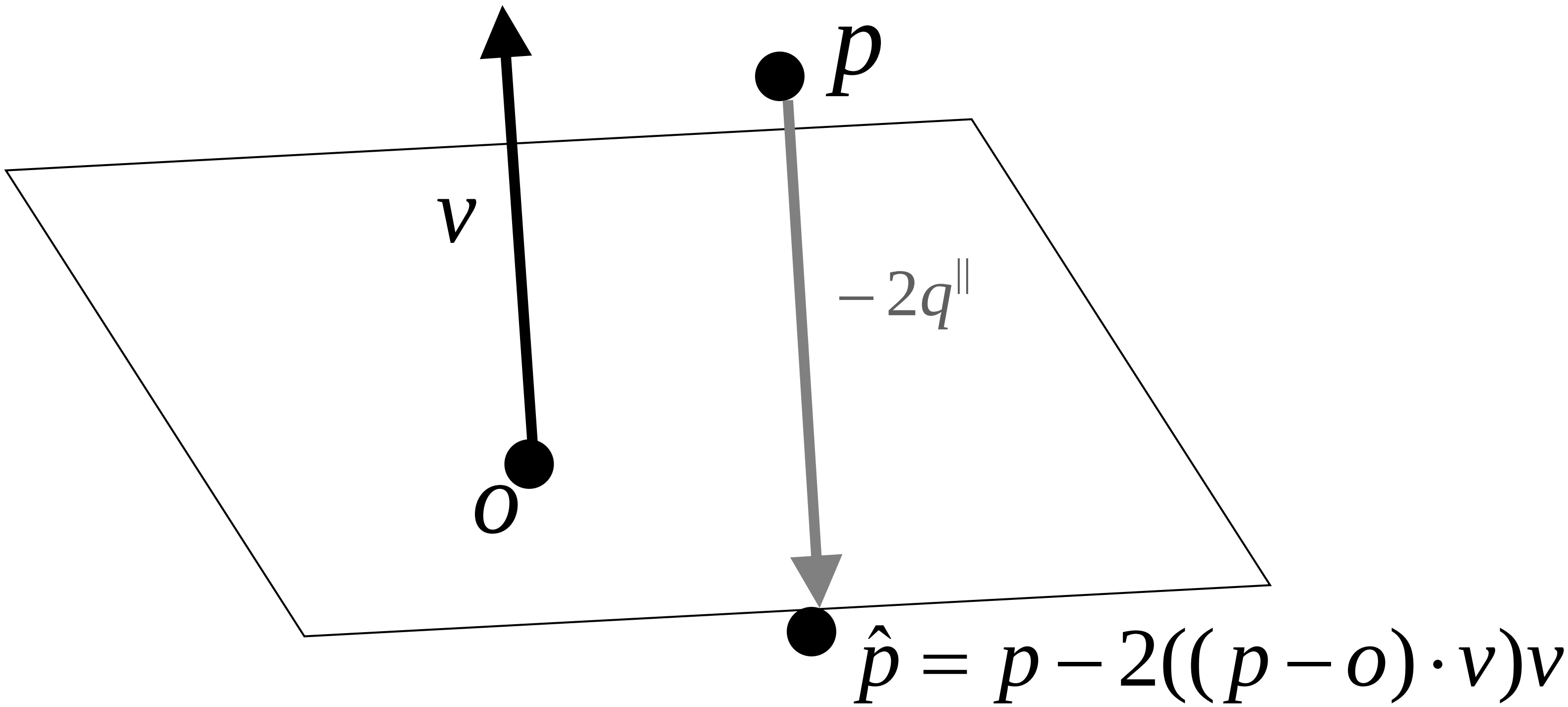
Mirror Image



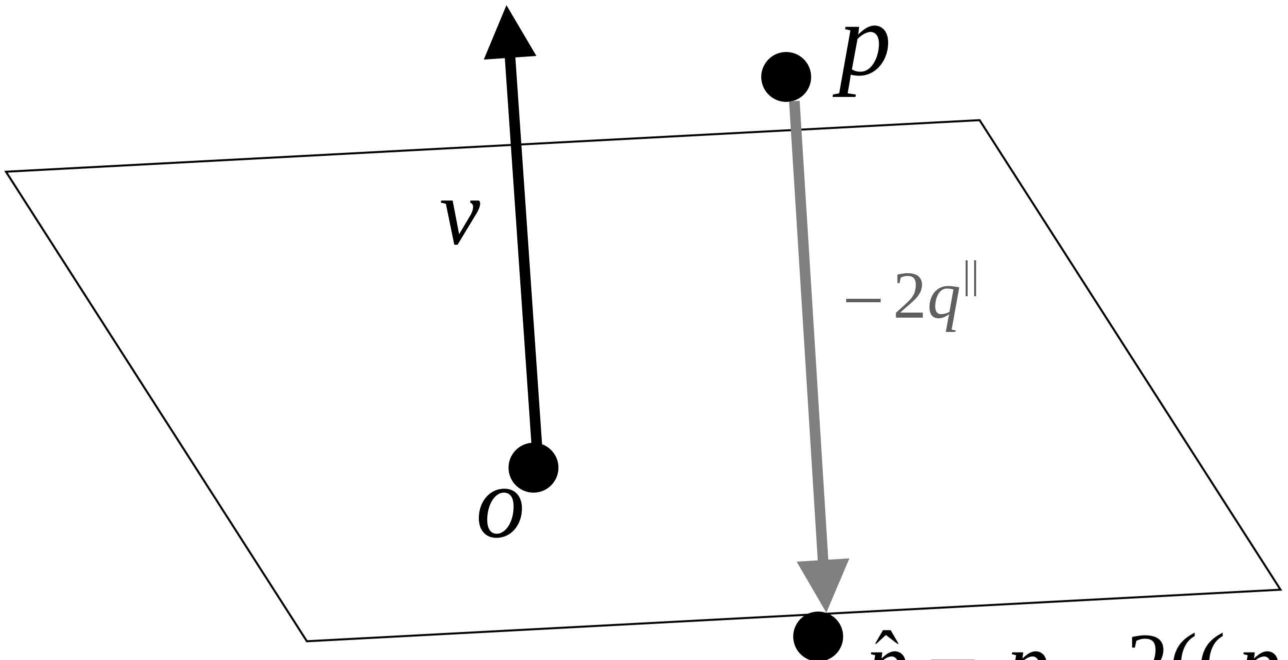
Mirror Image



Mirror Image



Mirror Image



$$\hat{p} = p - 2((p - o)\cdot v)v$$

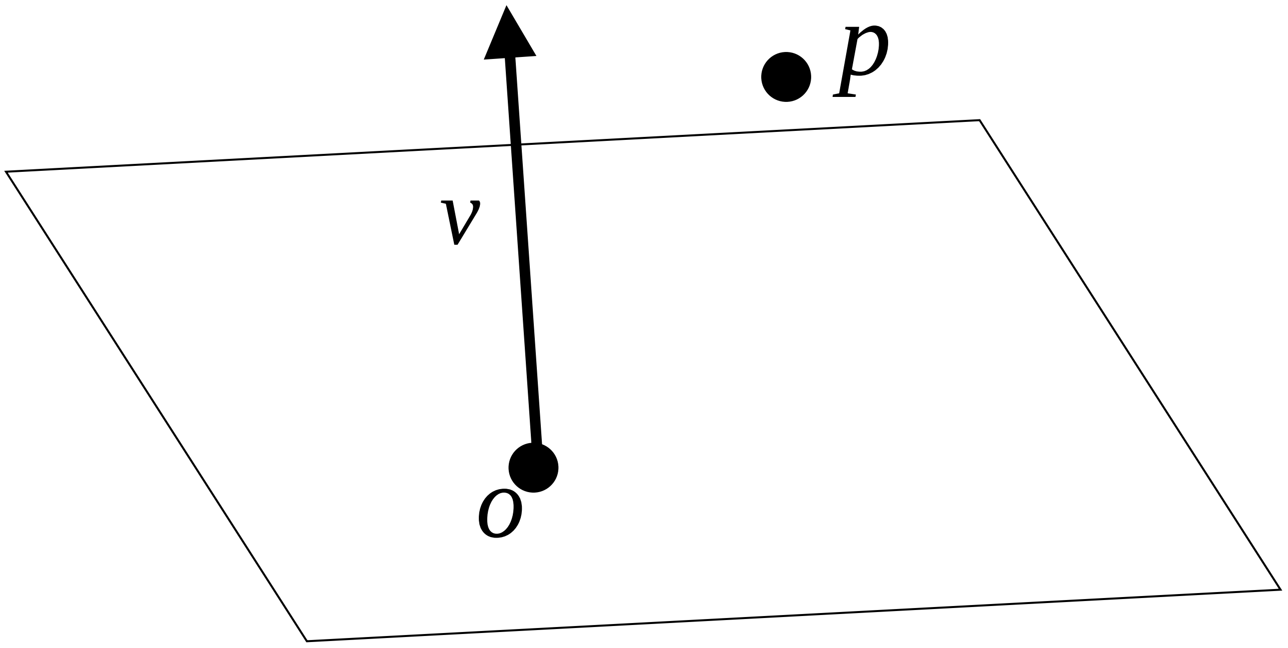
$$\begin{pmatrix} I - 2vv^T & 2vv^T o \\ 0 & 1 \end{pmatrix}\begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Mirror Image

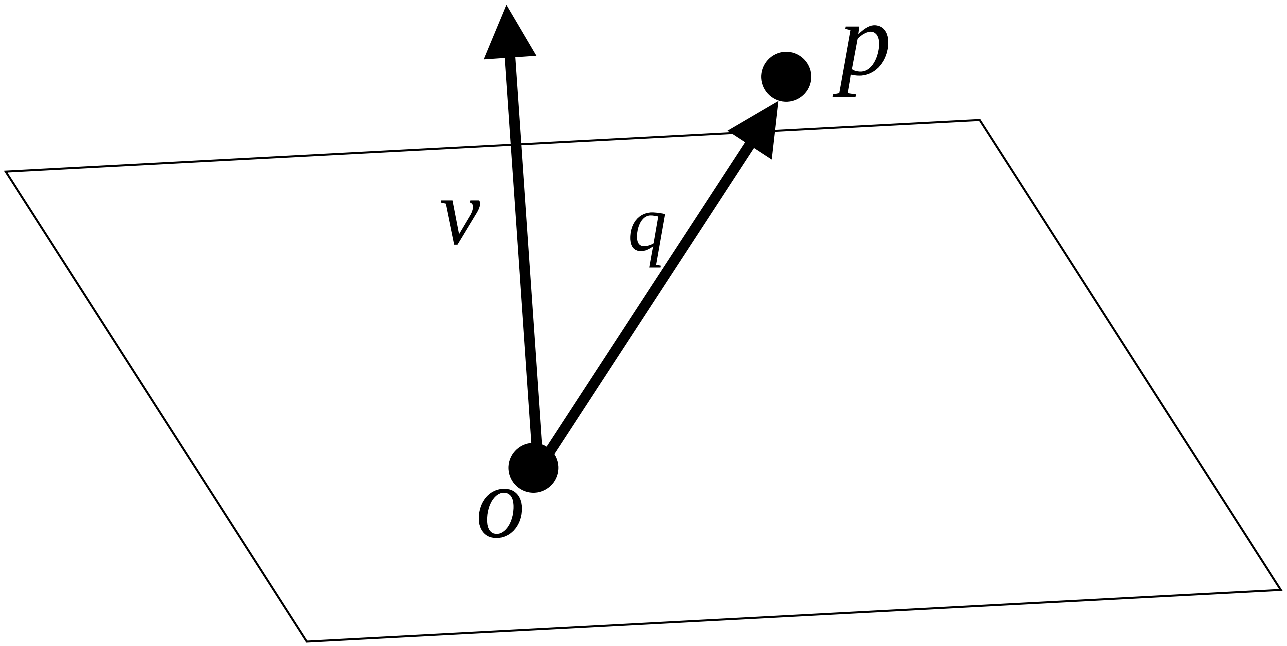
- $v = (0,0,1)^T$, $o = (0,0,0)^T$

$$\begin{pmatrix} I - 2vv^T & 2vv^T o \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

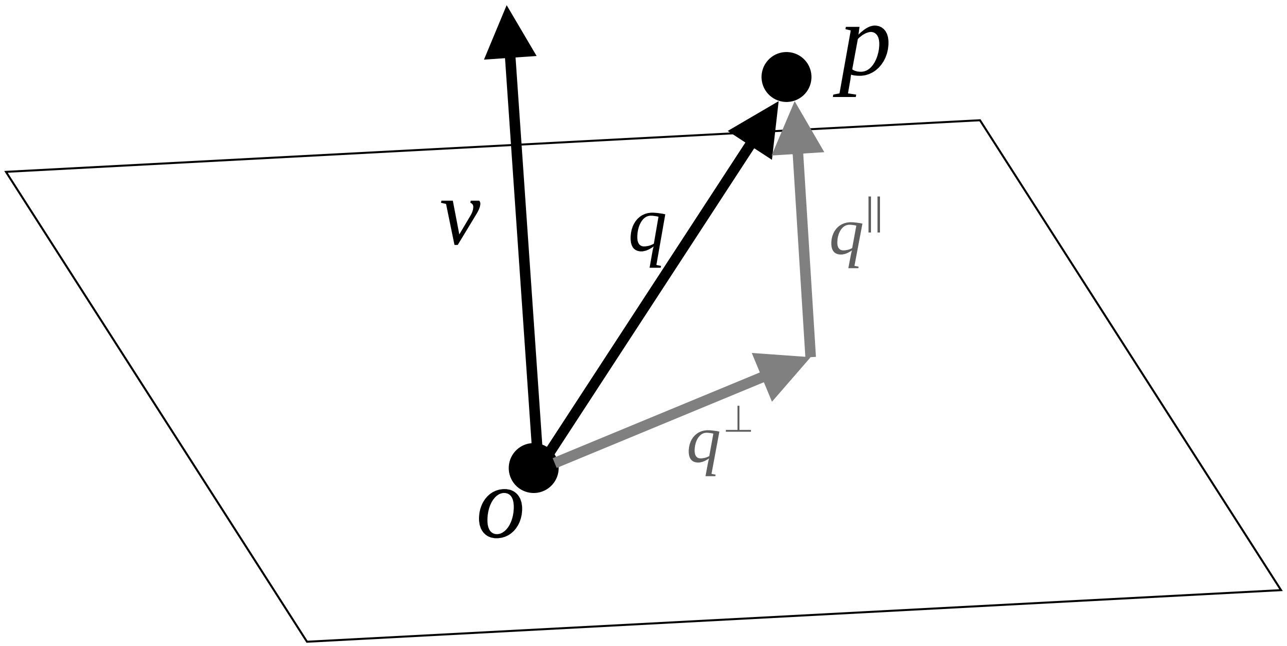
Orthogonal Projection



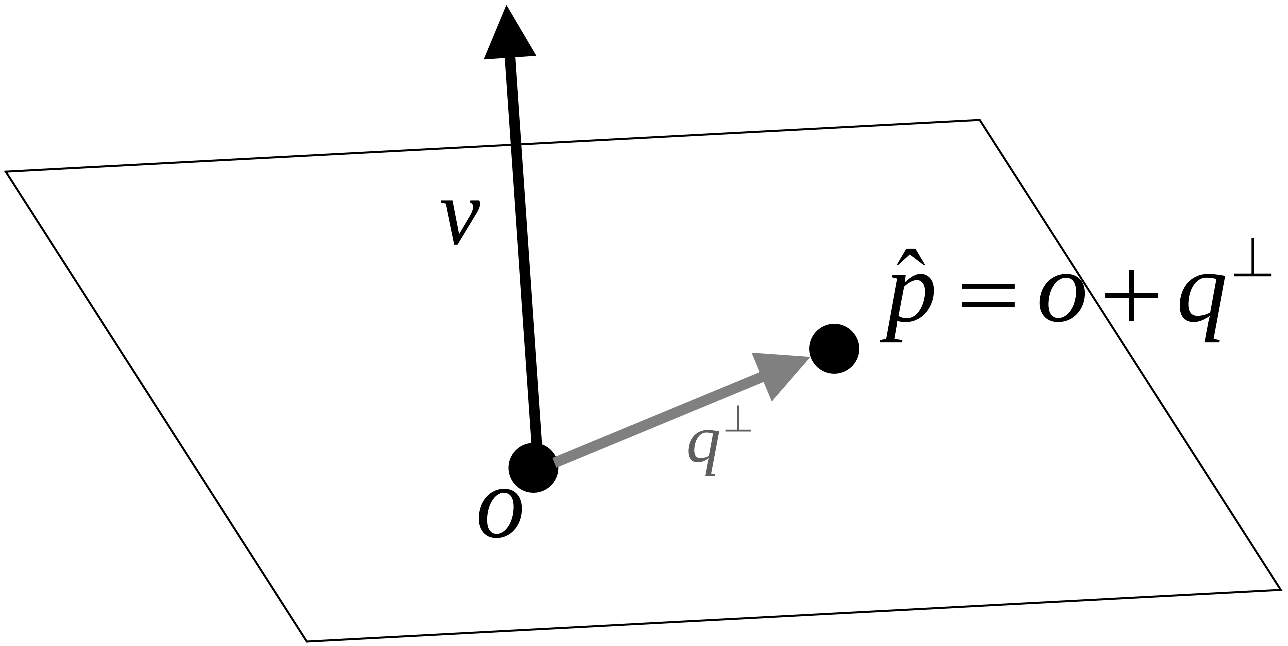
Orthogonal Projection



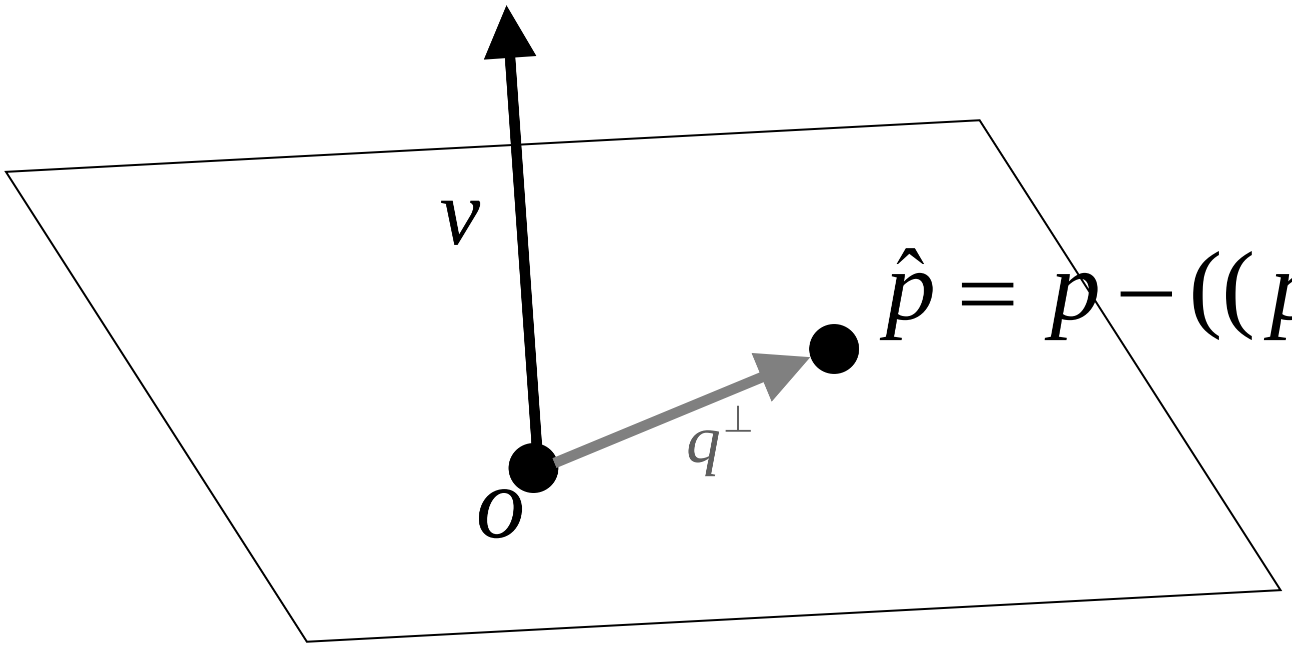
Orthogonal Projection



Orthogonal Projection

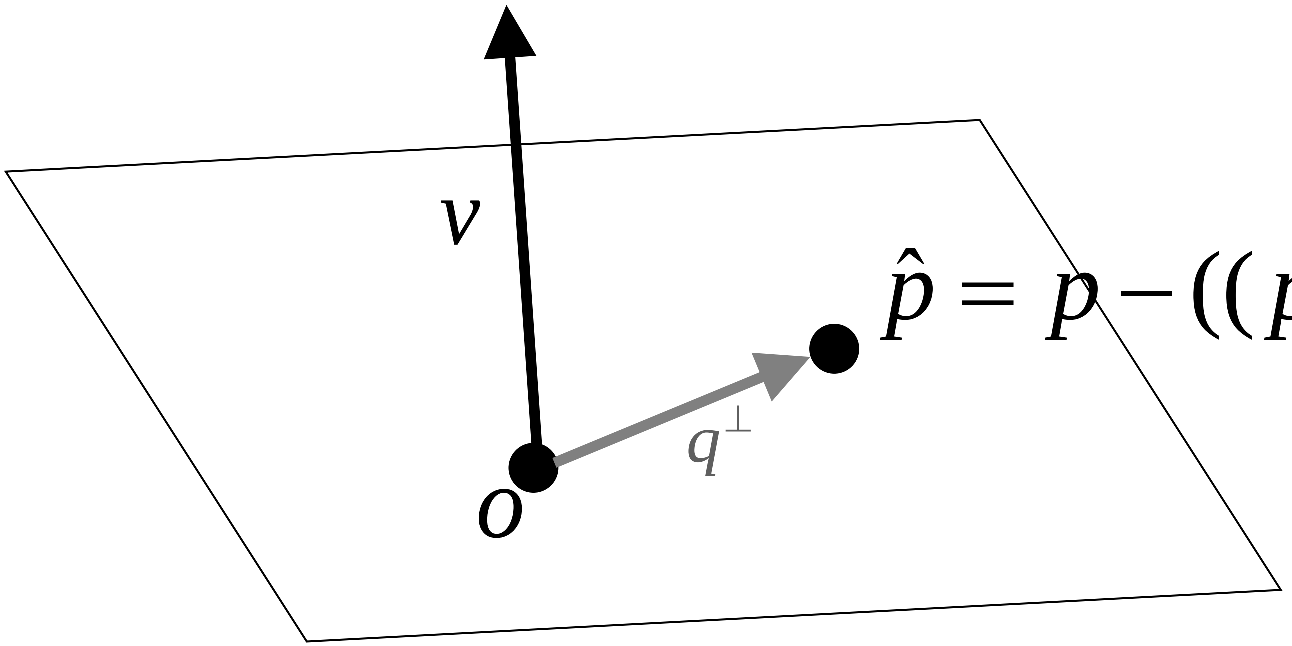


Orthogonal Projection



$$\hat{p} = p - ((p - o) \cdot v)v$$

Orthogonal Projection



$$\hat{p} = p - ((p - o) \cdot v)v$$

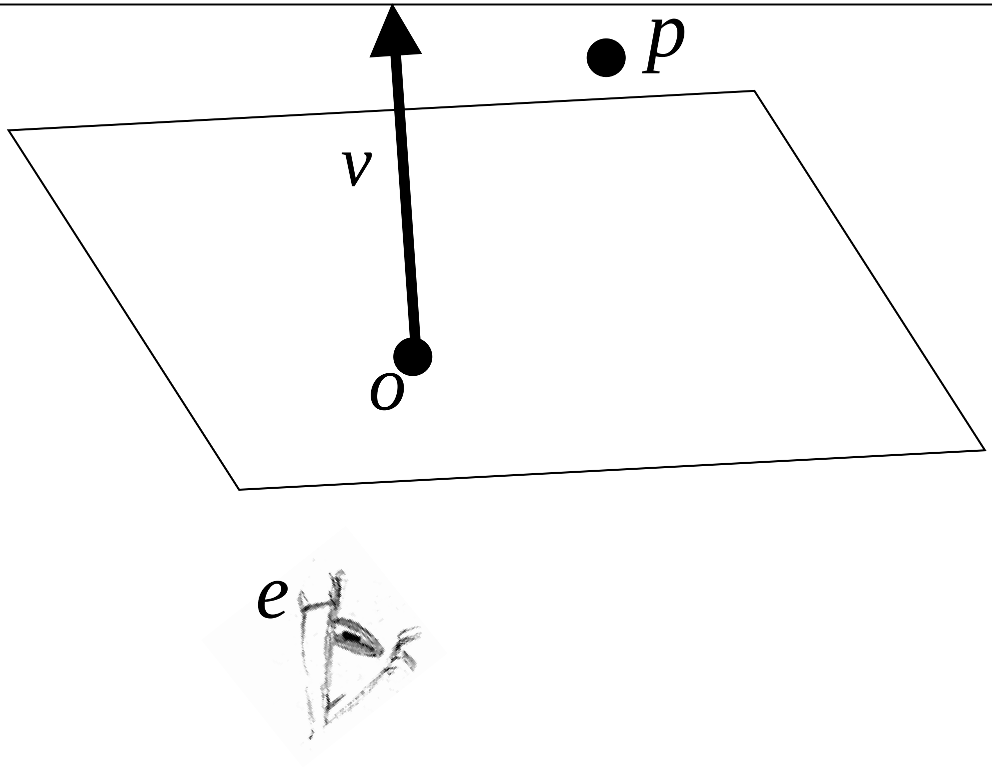
$$\begin{pmatrix} I - vv^T & +vv^T o \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Orthogonal Projection

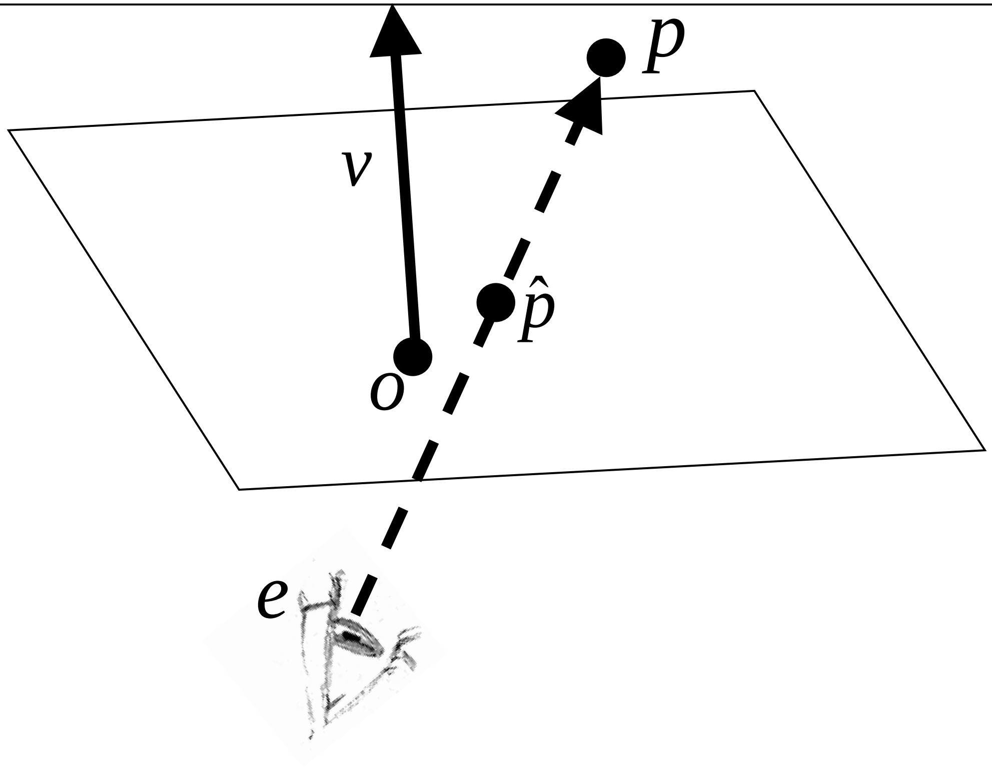
- $v = (0,0,1)^T$, $o = (0,0,0)^T$

$$\begin{pmatrix} I - vv^T & vv^T o \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

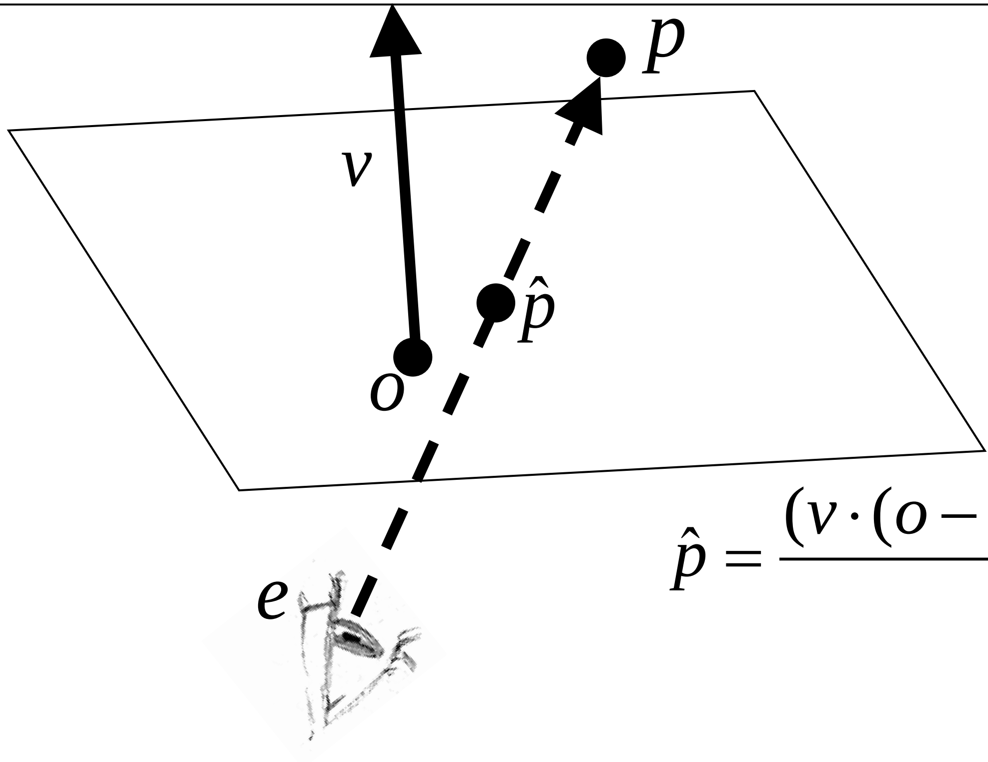
Perspective Projection



Perspective Projection



Perspective Projection



$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))\,p}{v \cdot (e - p)}$$

Perspective Projection

$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))p}{v \cdot (e - p)}$$

- Perspective transformations are not Affine!!!
- Rational expression requires a slight modification (homogeneous coordinates)

Perspective Projection

$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))p}{v \cdot (e - p)}$$

- Perspective transformations are not Affine!!!
- Rational expression requires a slight modification (homogeneous coordinates)

$$\begin{pmatrix} L & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ 1 \end{pmatrix}$$

Perspective Projection

$$\hat{p} = \frac{(v\cdot(o-p))e + (v\cdot(e-o))p}{v\cdot(e-p)}$$

- Perspective transformations are not Affine!!!
- Rational expression requires a slight modification (homogeneous coordinates)

$$\begin{pmatrix} L & t \\ c & d \end{pmatrix}\begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ \hat{w} \end{pmatrix}$$

Perspective Projection

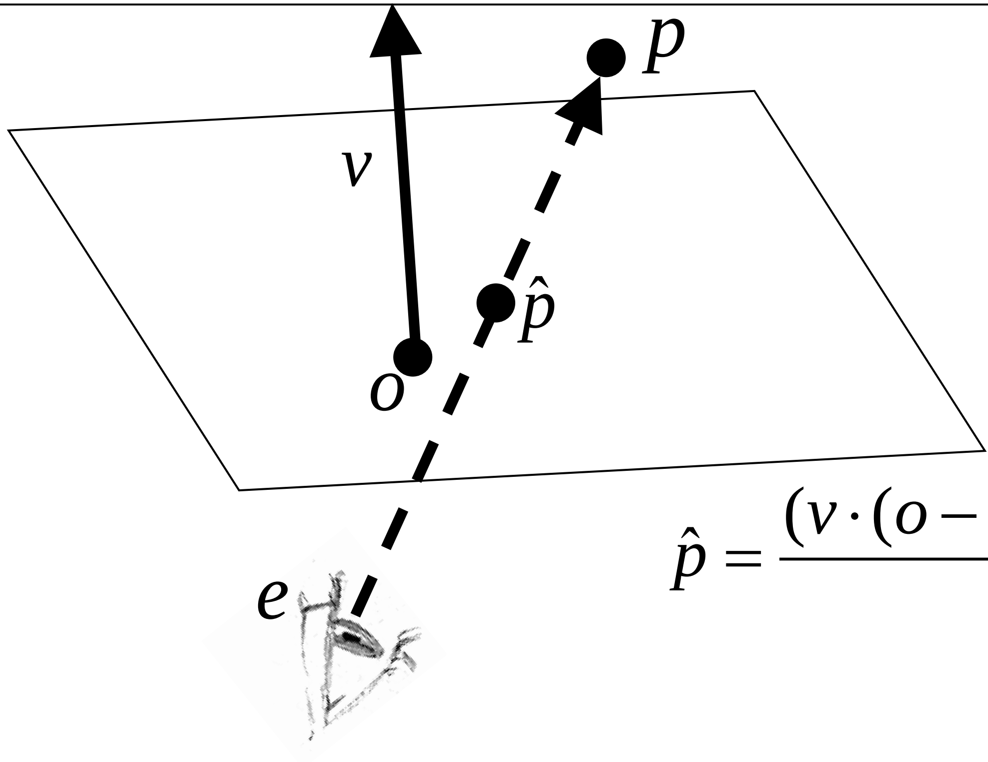
$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))\,p}{v \cdot (e - p)}$$

- Perspective transformations are not Affine!!!
- Rational expression requires a slight modification (homogeneous coordinates)

$$\begin{pmatrix} L & t \\ c & d \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ \hat{w} \end{pmatrix}$$

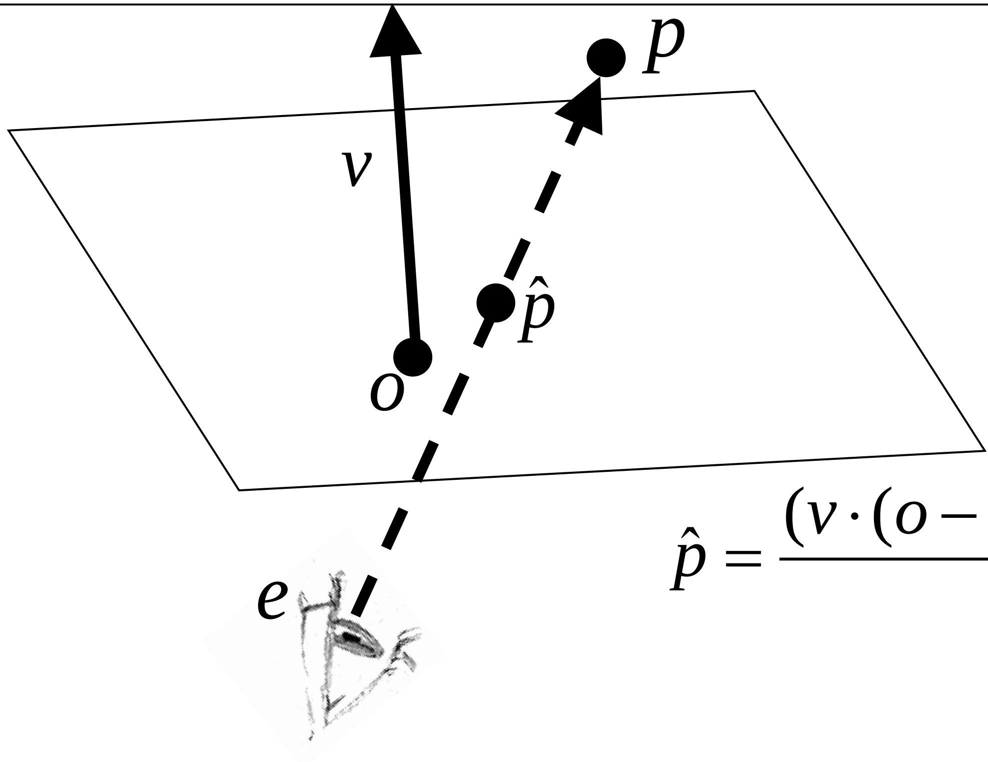
3D location is $\hat{p} / \hat{w}$

Perspective Projection



$$\hat{p} = \frac{(v \cdot (o - p))e + (v \cdot (e - o))\, p}{v \cdot (e - p)}$$

Perspective Projection



$$\hat{p} = \frac{(v\cdot(o-p))e + (v\cdot(e-o))\,p}{v\cdot(e-p)}$$

$$\begin{pmatrix} (v^T(e-o))I - e\,v^T & e(v^T o) \\ -v^T & v^T e \end{pmatrix}\begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{p} \\ \hat{w} \end{pmatrix}$$

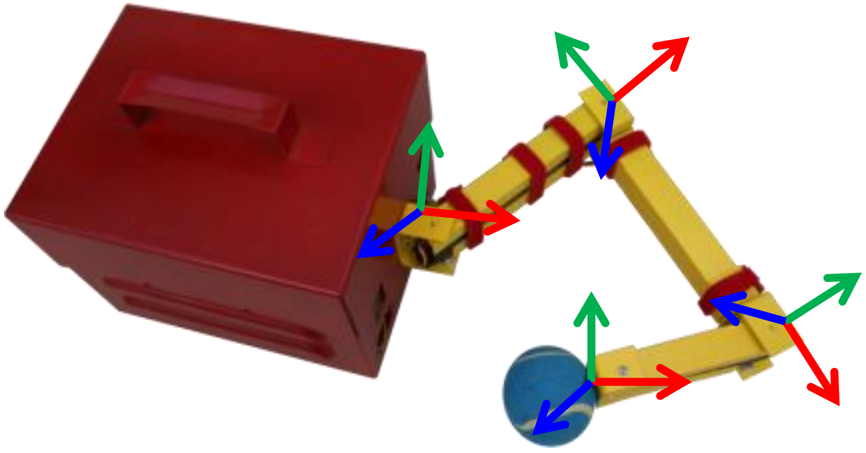
Perspective Projection

- $e = (0,0,1)^T$, $o = (0,0,0)^T$, $v = (0,0,1)^T$

$$\begin{pmatrix} (v^T(e-o))I - ev^T & e(v^T o) \\ -v^T & v^T e \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

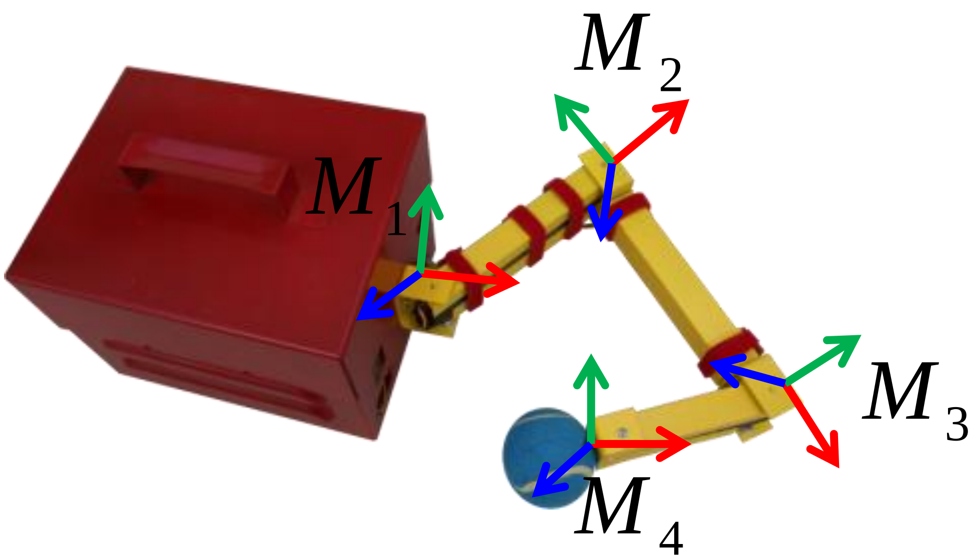
Hierarchical Animation

- Tree of transformation matrices
- Each node stores a local transformation with respect to its parent



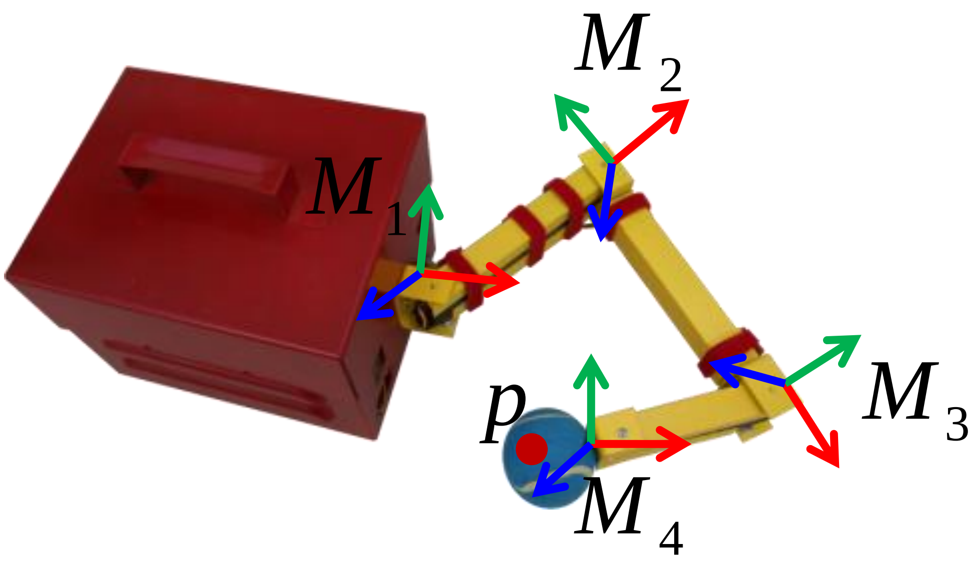
Hierarchical Animation

- Tree of transformation matrices
- Each node stores a local transformation with respect to its parent



Hierarchical Animation

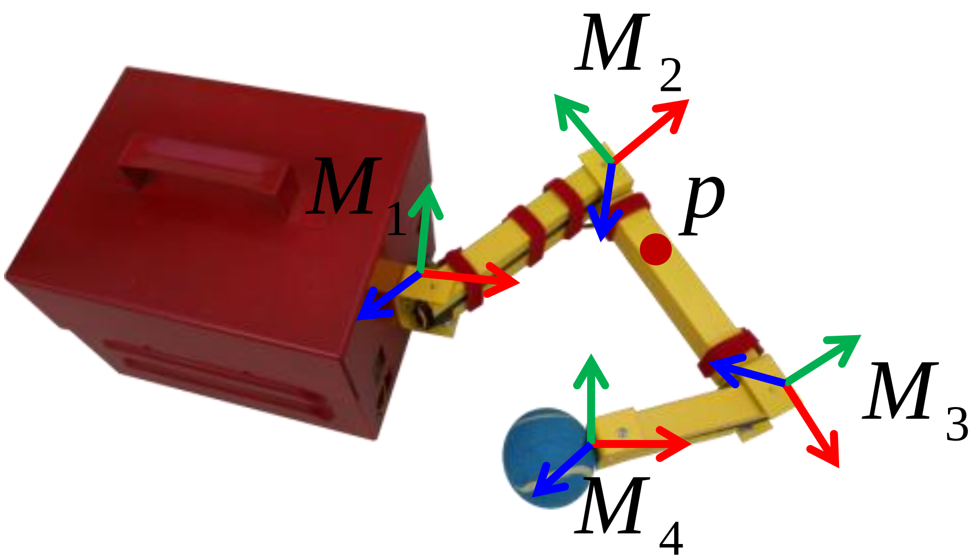
- Tree of transformation matrices
- Each node stores a local transformation with respect to its parent



$$\hat{p} = M_1 M_2 M_3 M_4 p$$

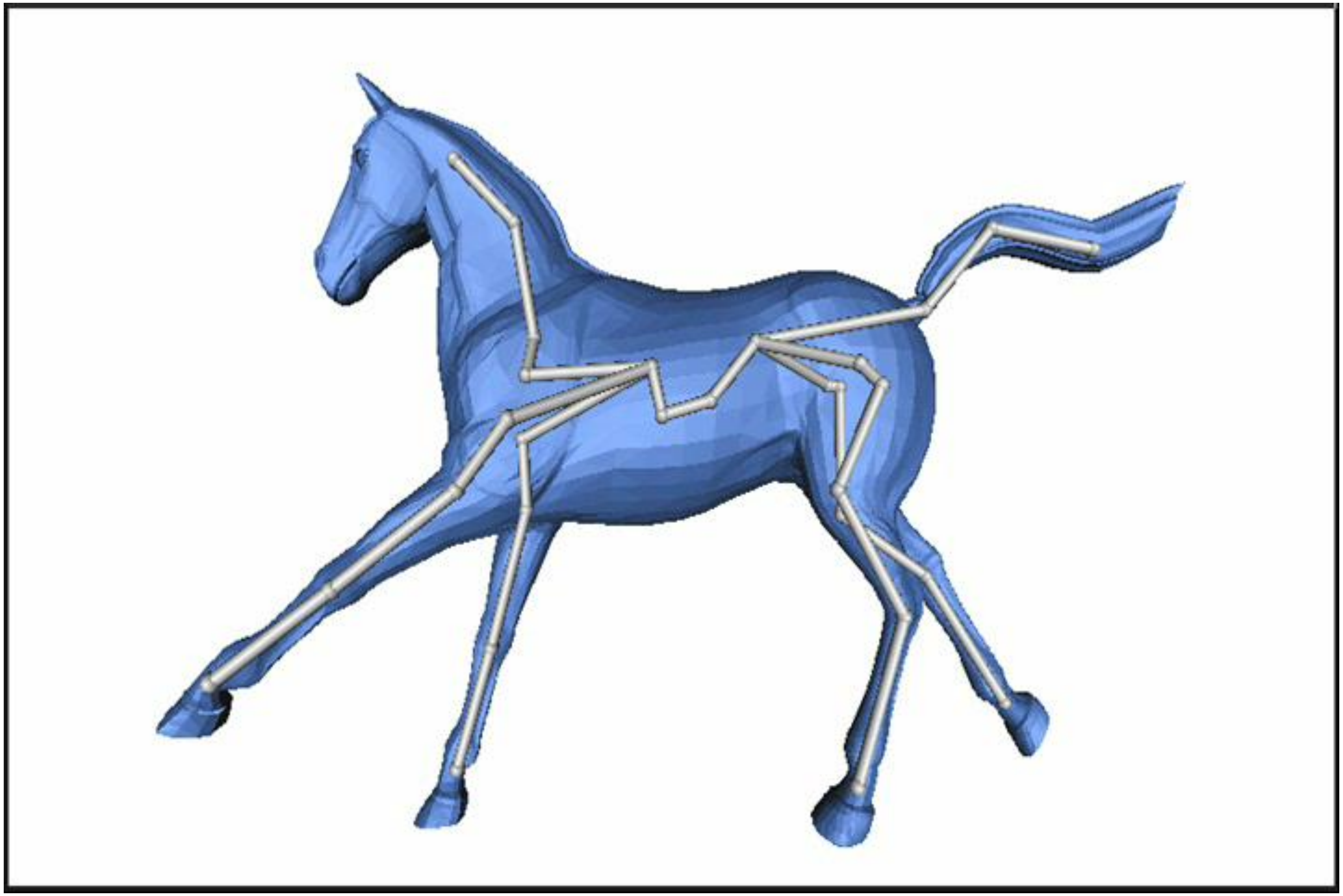
Hierarchical Animation

- Tree of transformation matrices
- Each node stores a local transformation with respect to its parent



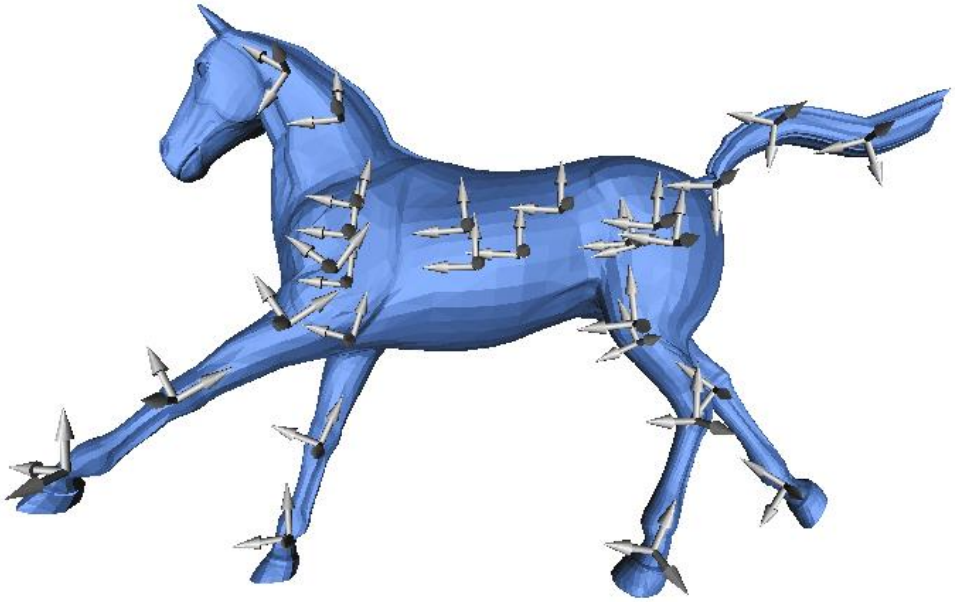
$$\hat{p} = M_1 M_2 p$$

Skeletal Animation



Skeletal Animation

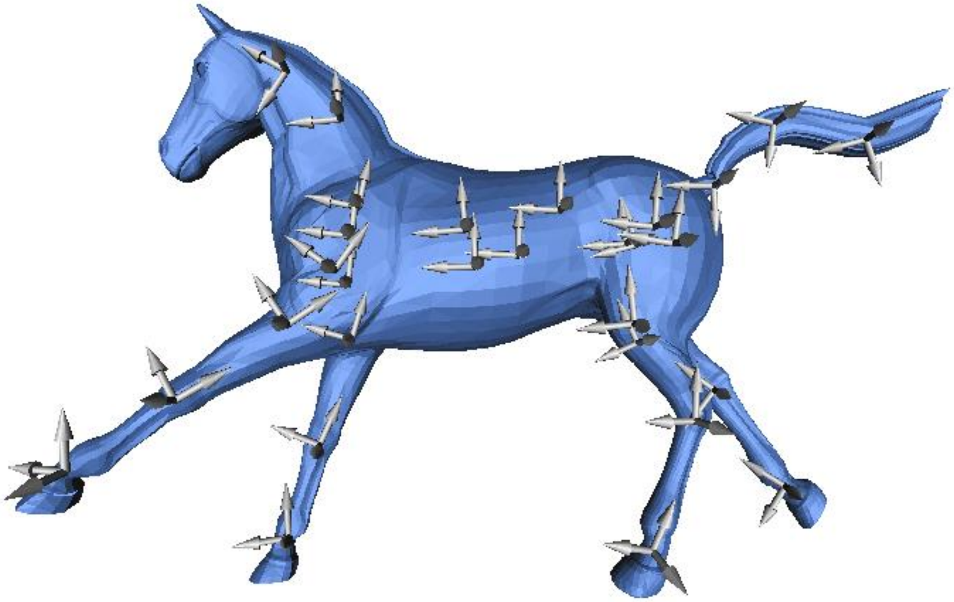
$$\hat{p} = \sum_i \alpha_i M_i p$$



Skeletal Animation

$$\hat{p} = \sum_i \alpha_i M_i p$$

M_i : Bone Transformation

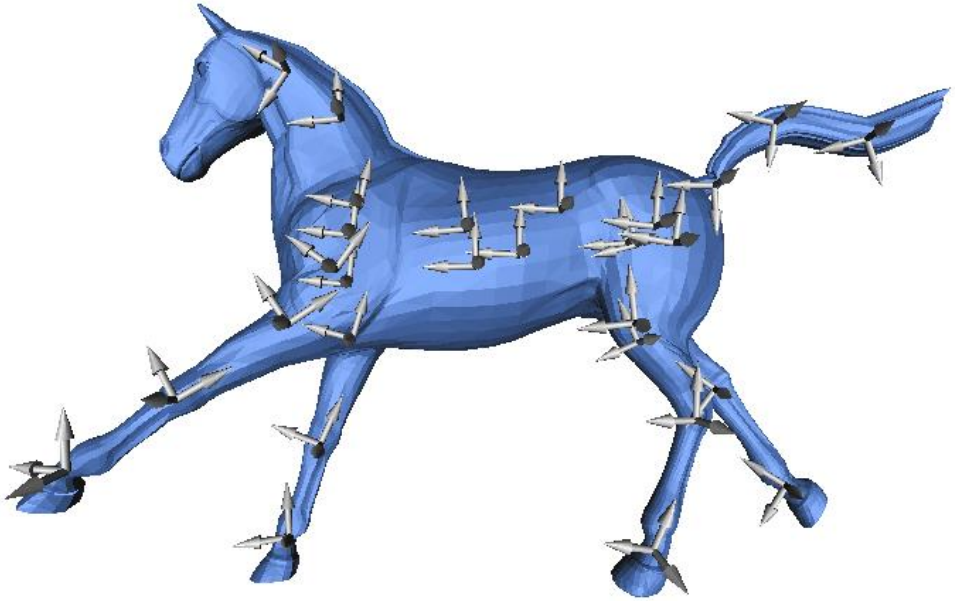


Skeletal Animation

$$\hat{p} = \sum_i \alpha_i M_i p$$

M_i : Bone Transformation

α_i : Skin Weights



Skeletal Animation

$$\hat{p} = \sum_i \alpha_i M_i p$$

M_i : Bone Transformation

α_i : Skin Weights

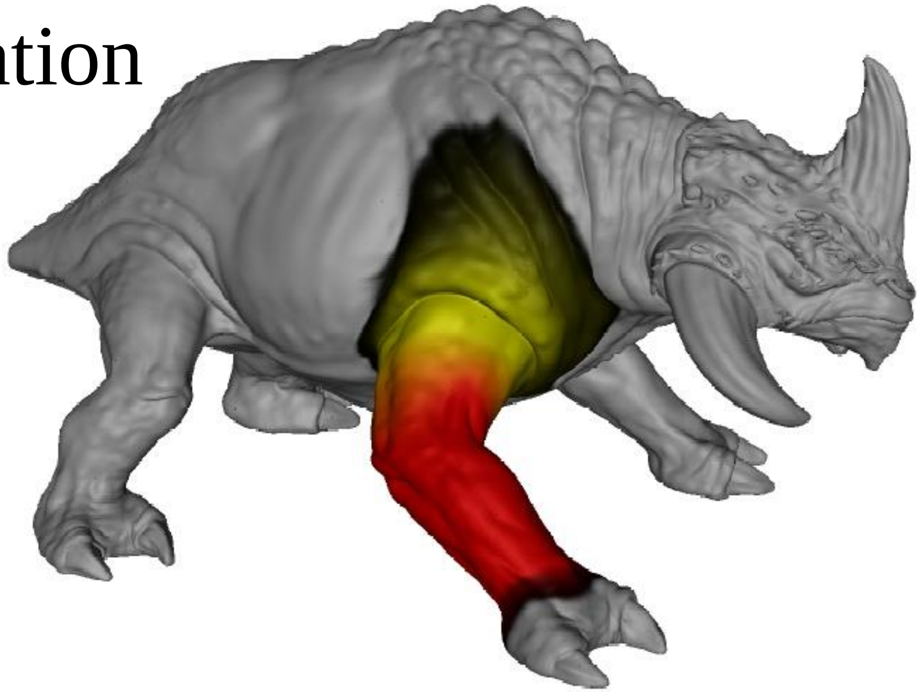
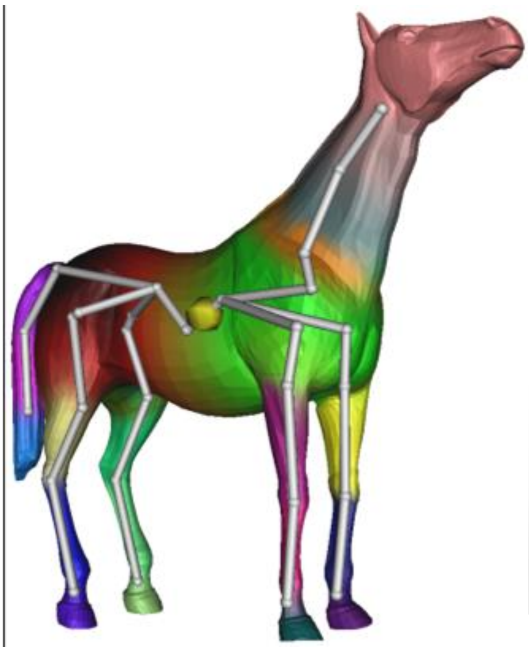
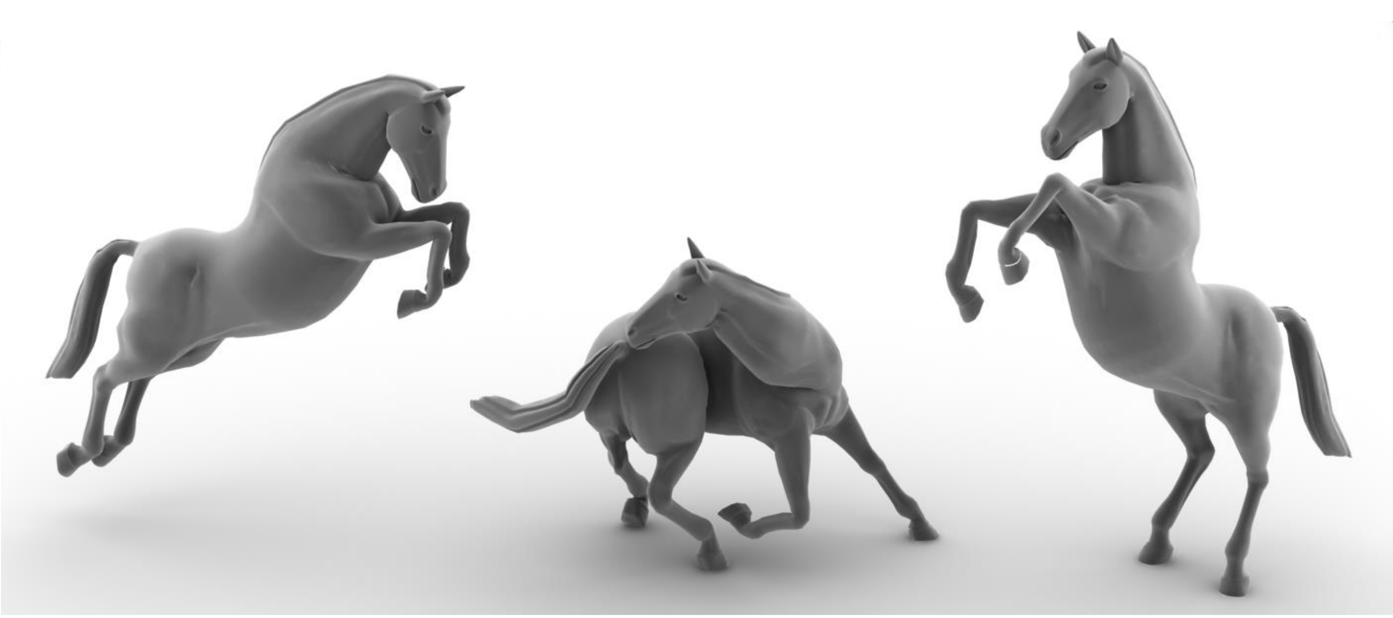


Image taken from [Wang et al. 2002]

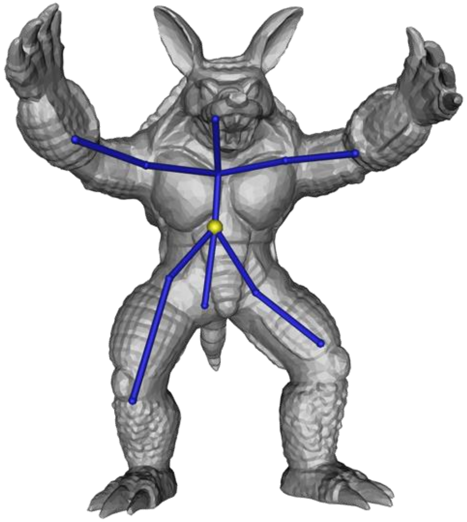
Skeletal Animation Examples





Skeletal Animation Examples

Skeletal Animation Examples



Transformations in OpenGL

- Types of transformation matrices
  - View
  - Model
  - Projection
  - Viewport

Transformations in OpenGL

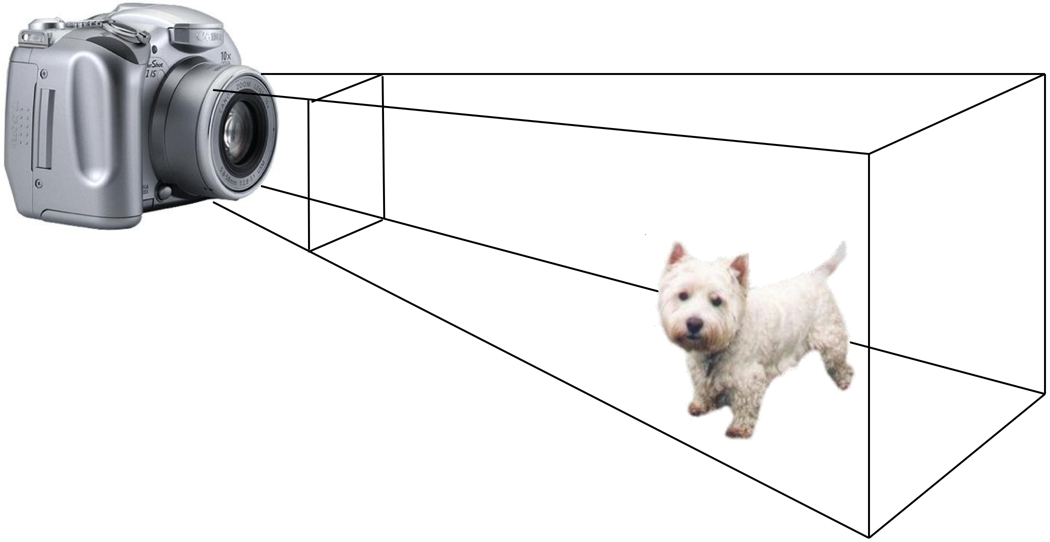
- Types of transformation matrices
  - View
  - Model
  - Projection
  - Viewport

Transformations in OpenGL

- Types of transformation matrices
  - View
  - Model
  - Projection
  - Viewport

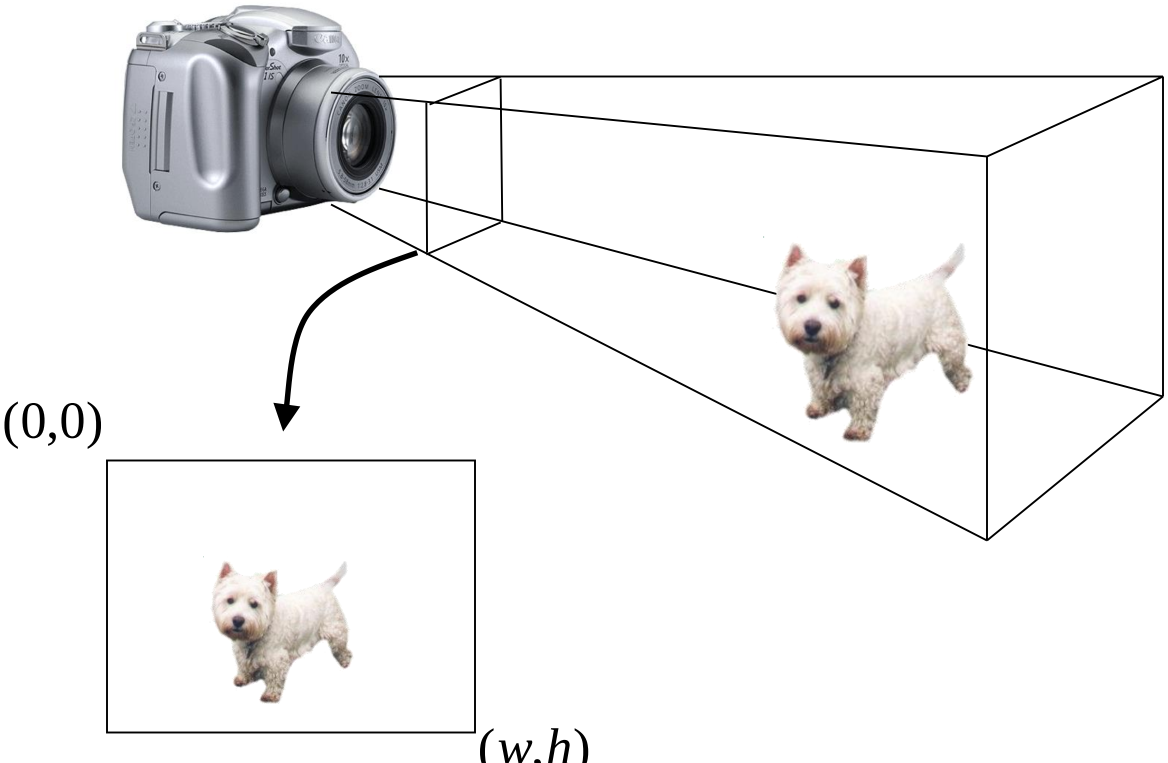
Transformations in OpenGL

- Types of transformation matrices
  - View
  - Model
  - Projection
  - Viewport



Transformations in OpenGL

- Types of transformation matrices
  - View
  - Model
  - Projection
  - Viewport



OpenGL: Matrix Commands

- glTranslatef(*x*, *y*, *z*)
- glScalef(*x*, *y*, *z*)
- glRotatef(*theta*, *vx*, *vy*, *vz*)
- glLoadIdentity(void)
- glPushMatrix(void)/glPopMatrix(void)

OpenGL: Matrix Example

glTranslatef(0, 1, 3) $\longleftarrow$ M_1

$$M_1 P$$

OpenGL: Matrix Example

glTranslatef(0, 1, 3) $\longleftarrow M_1$

glRotatef(45, 0, 1, 0) $\longleftarrow M_2$

$$M_1 M_2 P$$

OpenGL: Matrix Example

glTranslatef(0, 1, 3) ⟵ M_1

glRotatef(45, 0, 1, 0) ⟵ M_2

glScalef(1, 1, 2) ⟵ M_3

$$M_1 M_2 M_3 P$$

OpenGL: Matrix Example

glTranslatef(0, 1, 3) ⟵ M_1

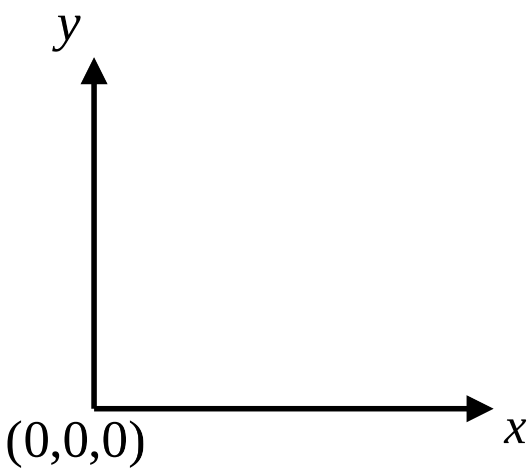
glRotatef(45, 0, 1, 0) ⟵ M_2

glScalef(1, 1, 2) ⟵ M_3

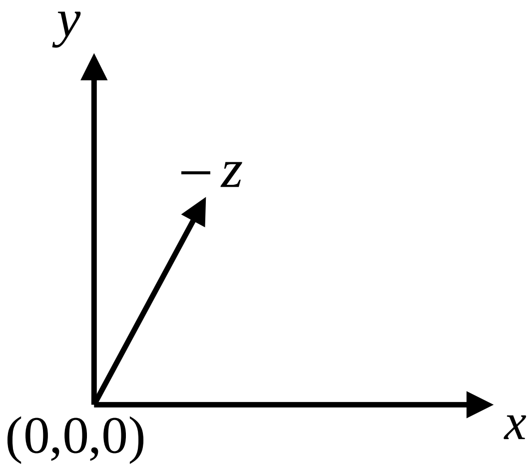
OpenGL multiplies matrices in the opposite order of what you might expect

$$M_1 M_2 M_3 P$$

Coordinate Systems in OpenGL

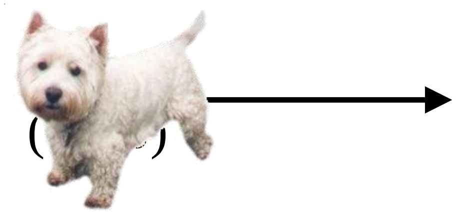


Coordinate Systems in OpenGL



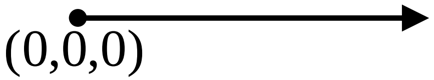
Coordinate Systems in OpenGL

Coordinate Systems in OpenGL

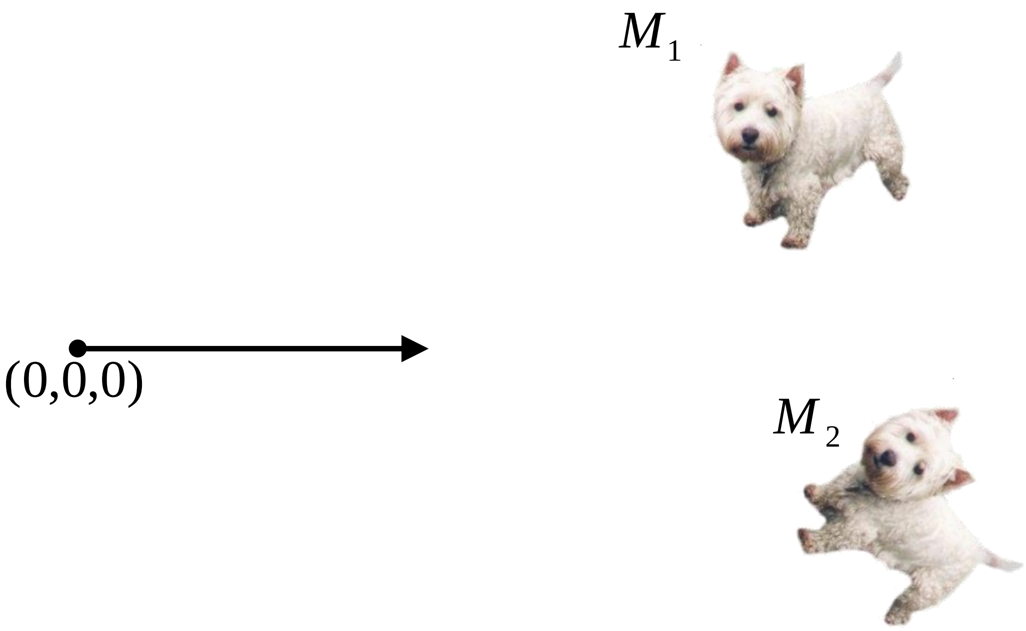


Coordinate Systems in OpenGL

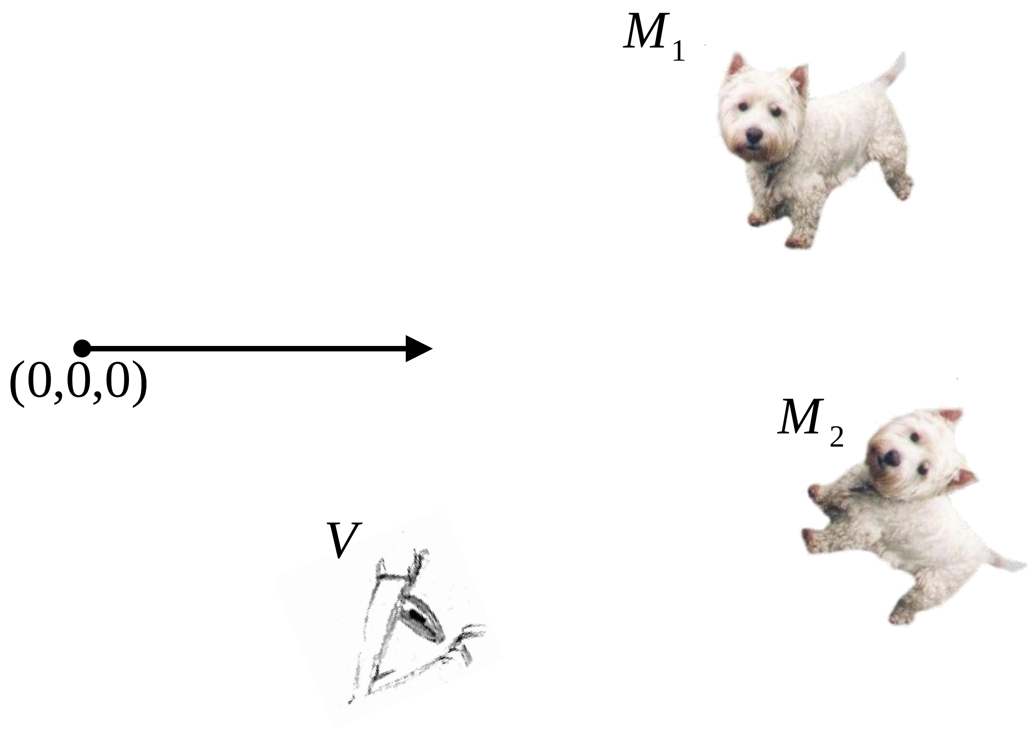
M_1



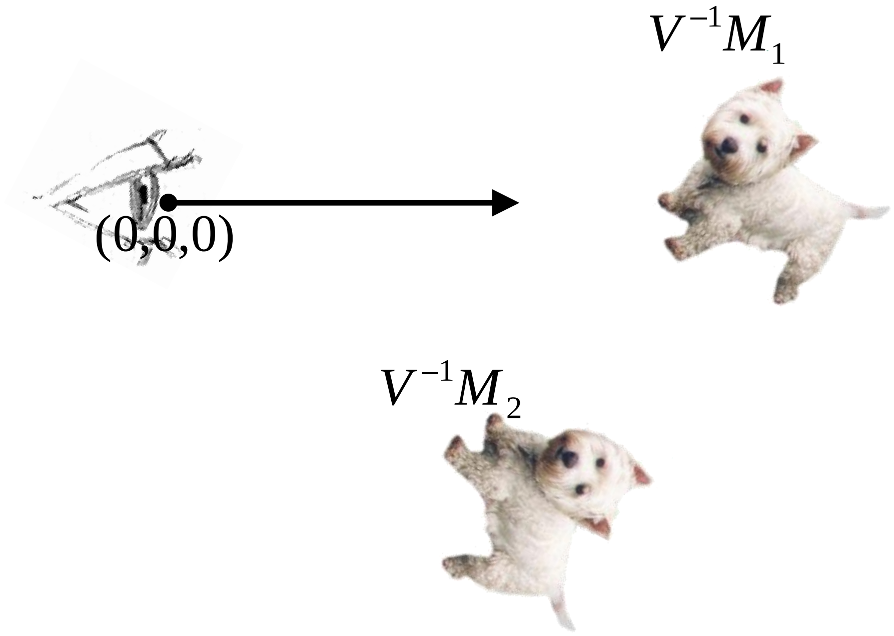
Coordinate Systems in OpenGL



Coordinate Systems in OpenGL

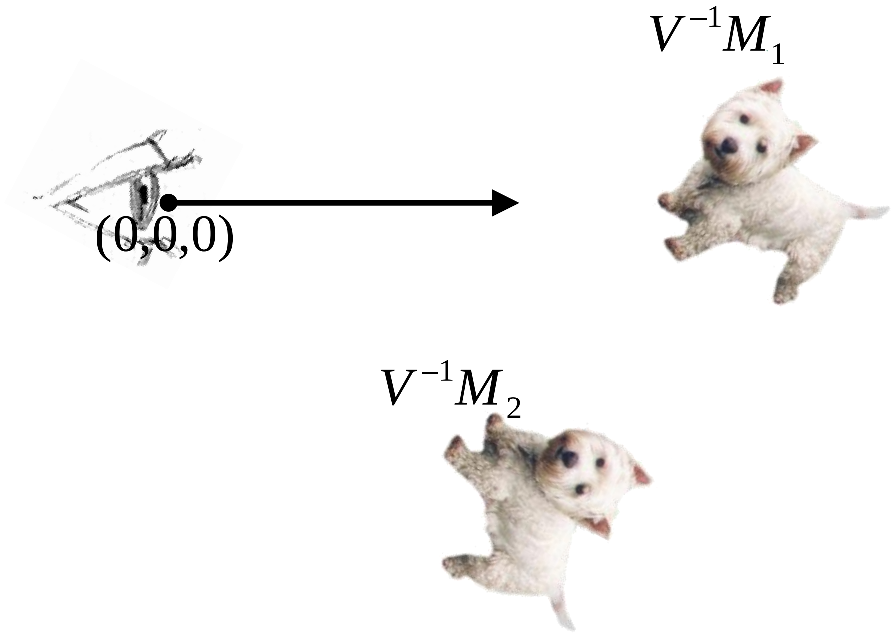


Coordinate Systems in OpenGL



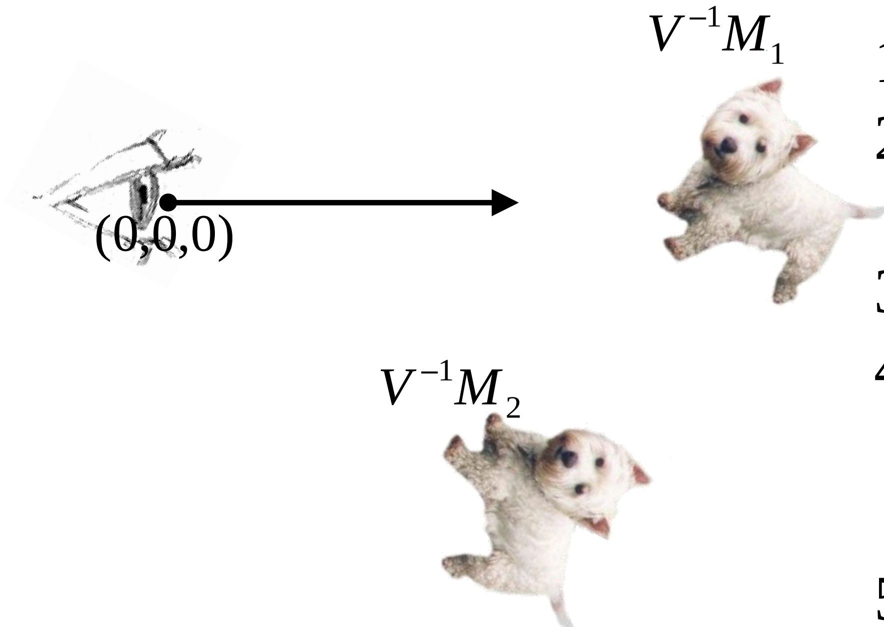
Coordinate Systems in OpenGL

$$\text{ModelView} = V^{-1}M$$



Coordinate Systems in OpenGL

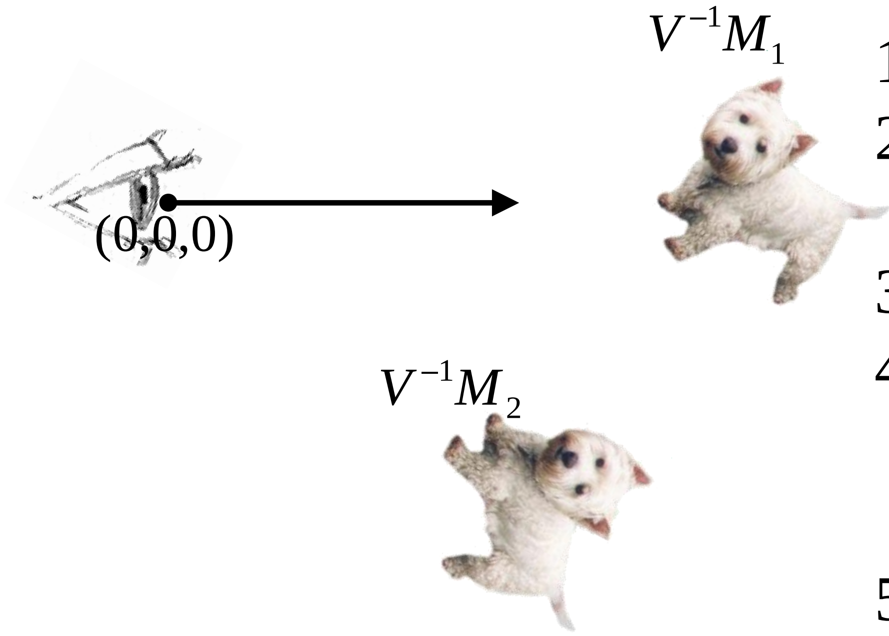
$$\text{ModelView} = V^{-1}M$$



1.PushMatrix
2.Specify viewer using inverse of view transformation
3. PushMatrix
4. Position object with transformations in opposite order of what you would expect
5. PopMatrix
6. PopMatrix

Coordinate Systems in OpenGL

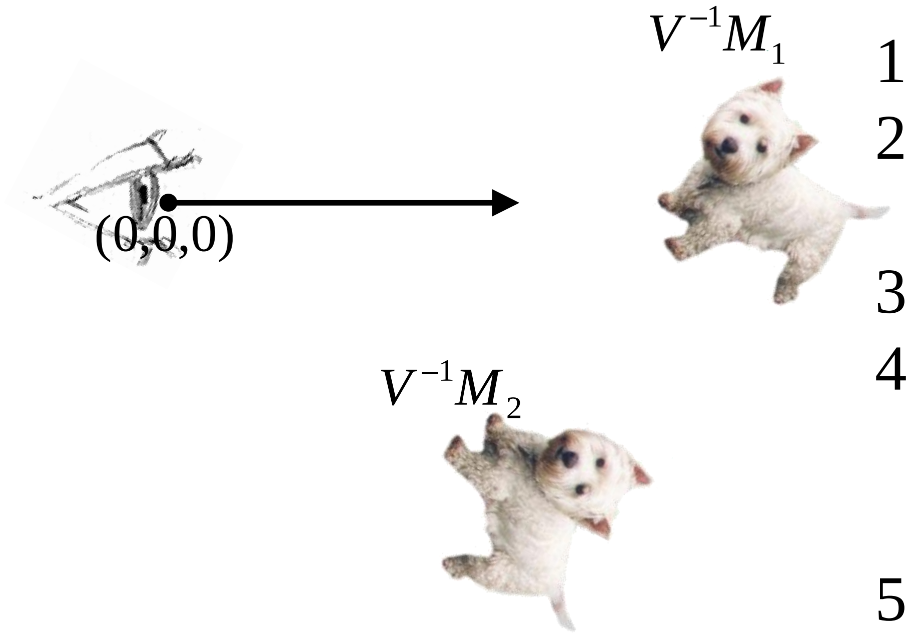
$$\text{ModelView} = (RT)^{-1} M$$



1. PushMatrix
2. Specify viewer using inverse of view transformation
3. PushMatrix
4. Position object with transformations in opposite order of what you would expect
5. PopMatrix
6. PopMatrix

Coordinate Systems in OpenGL

$$\text{ModelView} = T^{-1}R^{-1}M$$



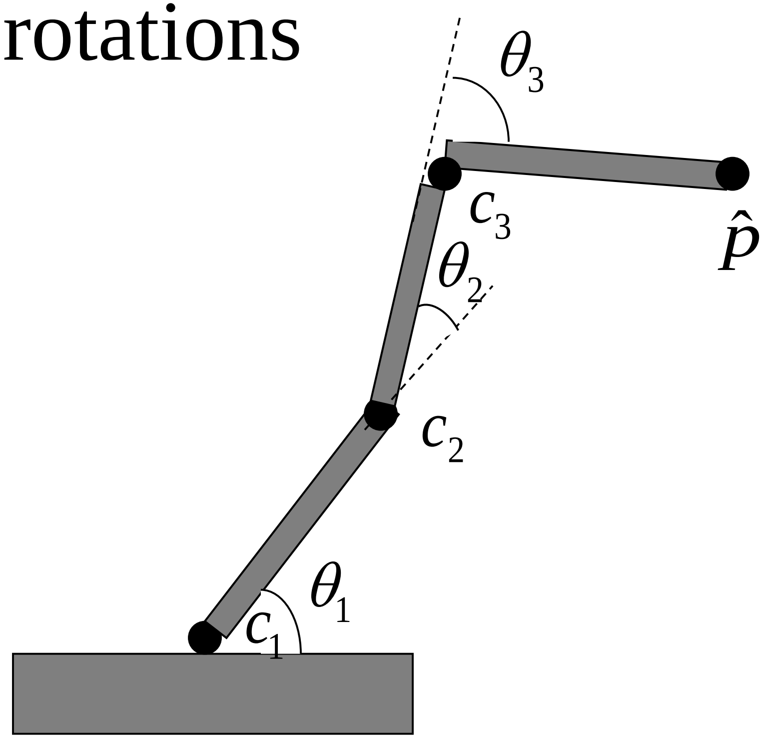
1.PushMatrix
2.Specify viewer using inverse of view transformation
3. PushMatrix
4. Position object with transformations in opposite order of what you would expect
5. PopMatrix
6. PopMatrix

OpenGL: Special Matrix Commands

- glMatrixMode(GL_MODELVIEW / GL_PROJECTION)
- gluLookAt(*ex*, *ey*, *ez*, *cx*, *cy*, *cz*, *ux*, *uy*, *uz*)
- gluPerspective(*fov*, *aspect*, *near*, *far*)
- glOrtho(*left*, *right*, *bottom*, *top*, *near*, *far*)
- glViewport(*x*, *y*, *w*, *h*)

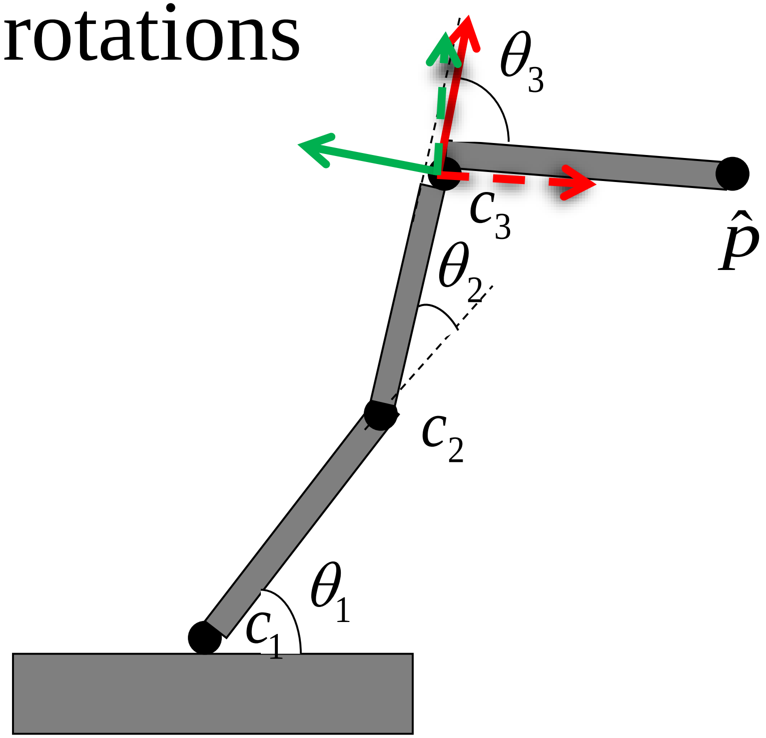
Hierarchical Animation

- Typically represented as relative joint locations and rotations



Hierarchical Animation

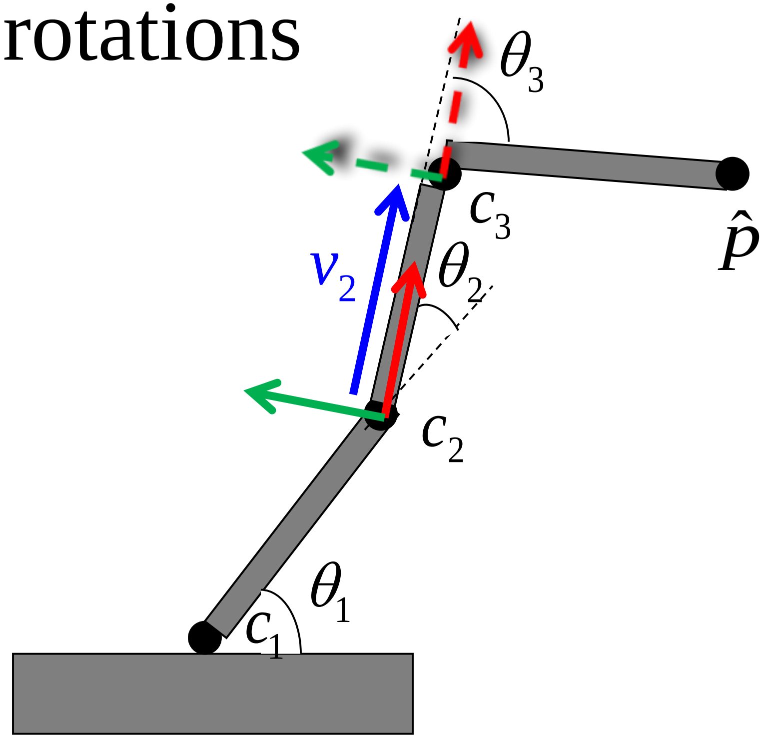
- Typically represented as relative joint locations and rotations



$$\hat{p} = \qquad R(\theta_3)\, p$$

Hierarchical Animation

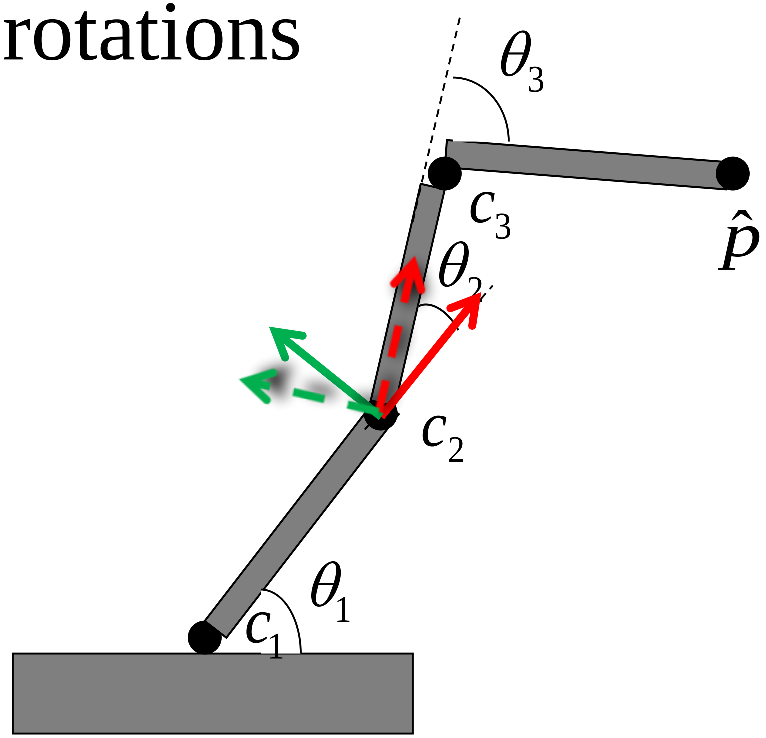
- Typically represented as relative joint locations and rotations



$$\hat{p} = \qquad T(v_2) R(\theta_3) p$$

Hierarchical Animation

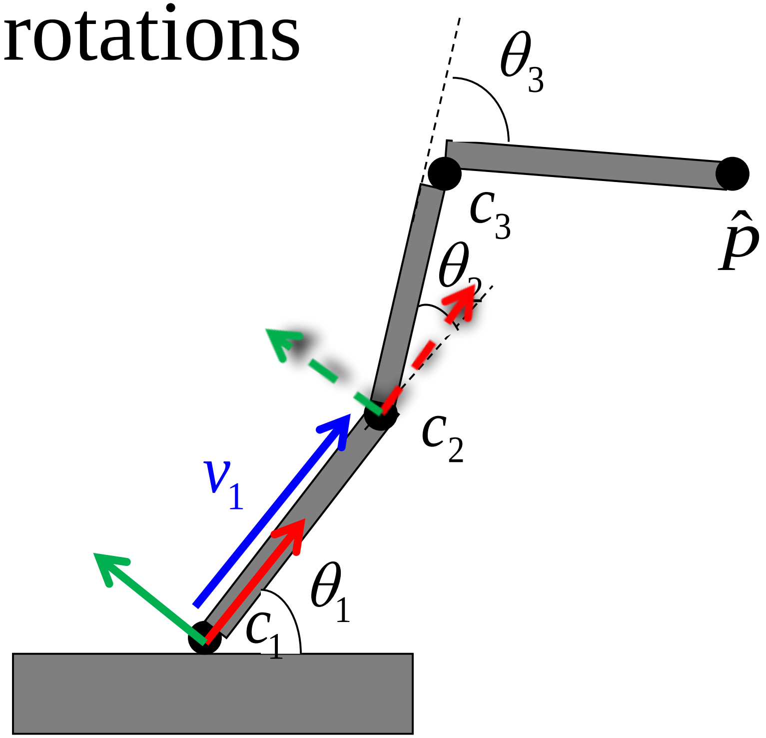
- Typically represented as relative joint locations and rotations



$$\hat{p} = \quad R(\theta_2) T(v_2) R(\theta_3) p$$

Hierarchical Animation

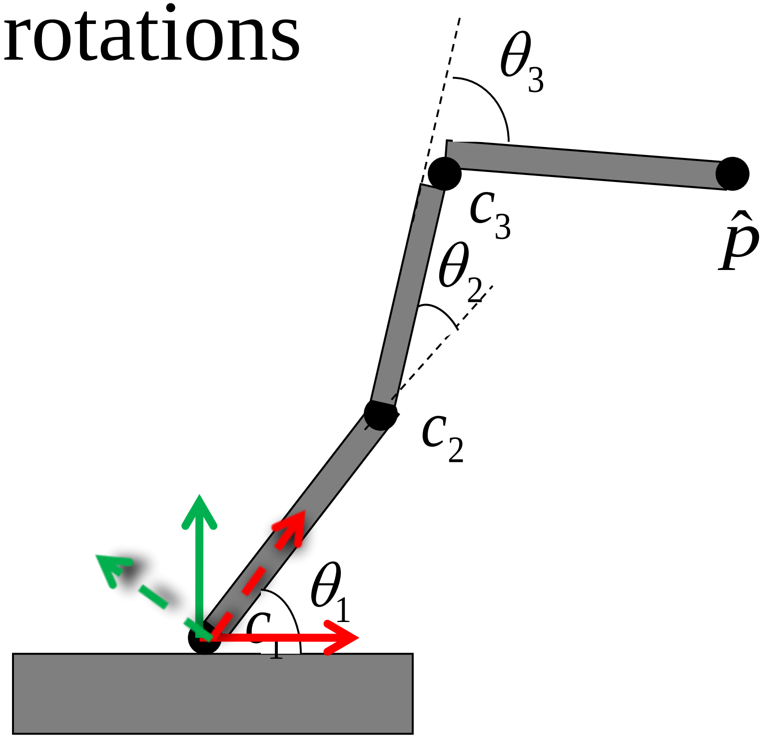
- Typically represented as relative joint locations and rotations



$$\hat{p} = \quad T(v_1)R(\theta_2)T(v_2)R(\theta_3)p$$

Hierarchical Animation

- Typically represented as relative joint locations and rotations



$$\hat{p} = R(\theta_1)T(v_1)R(\theta_2)T(v_2)R(\theta_3)p$$