CS559: Computer Graphics

Lecture 8: 3D Transforms

Li Zhang

Spring 2008

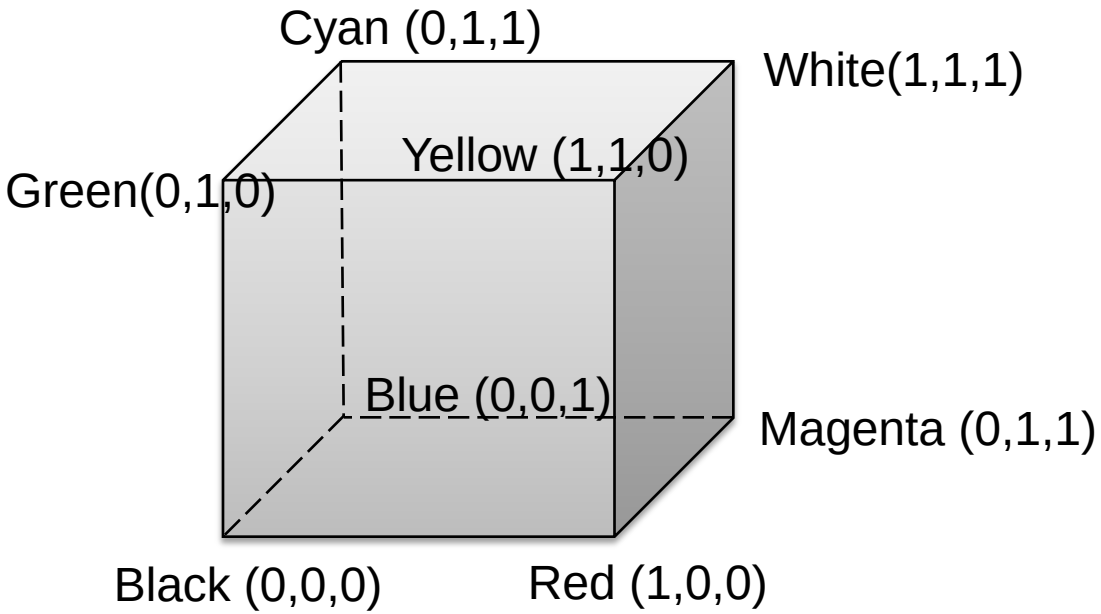
Most Slides from Stephen Chenney

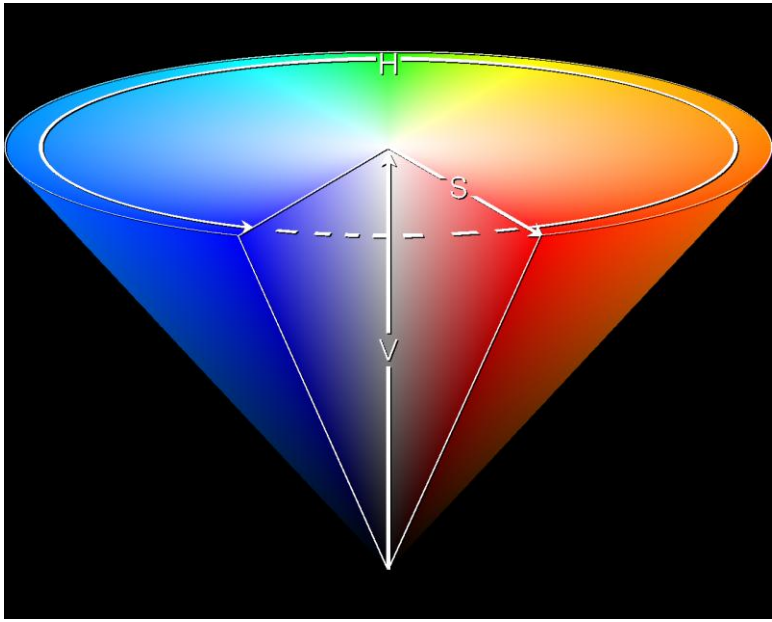
Today

- Finish Color space
- 3D Transforms and Coordinate system

- Reading:
  - Shirley ch 6

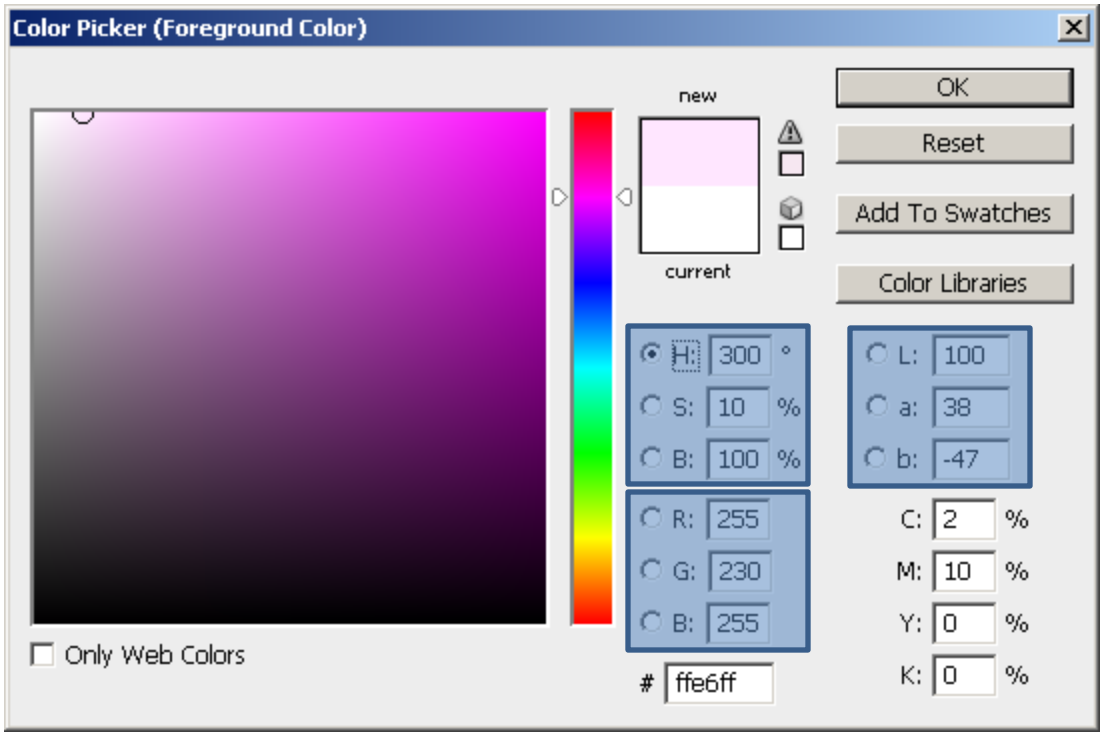
RGB and HSV





Different ways to represent/parameterize color

Photoshop Color Picker



L-A-B Color Space

- L-A-B
  - L: luminance/Brightness
  - A: position between magenta and green (negative values indicate green while positive values indicate magenta)
  - B: position between yellow and blue (negative values indicate blue and positive values indicate yellow)

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{aligned} L^* &= 116\, f(Y/Y_n) - 16 \\ a^* &= 500\,[f(X/X_n) - f(Y/Y_n)] \\ b^* &= 200\,[f(Y/Y_n) - f(Z/Z_n)] \end{aligned}$$

$$\begin{aligned} f(t) &= t^{1/3} \\ f(t) &= 7.787\,t + 16/116 \end{aligned}$$

http://en.wikipedia.org/wiki/Lab_color_space

http://en.wikipedia.org/wiki/CIE_1931_color_space

Spatial resolution and color

original

R

G

B

Blurring the G component

original

processed

R

G

B

Blurring the R component

original

processed

R

G

B

Blurring the B component

original

processed

R

G

B

Lab Color Component

L

a

b

A rotation of the color coordinates into directions that are more perceptually meaningful:
L: luminance,
a: magenta-green,
b: blue-yellow

Bluring L

original

processed

L

a

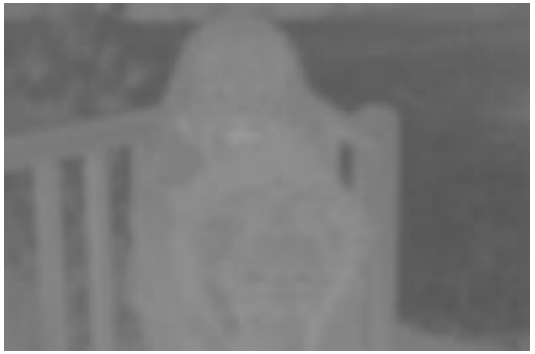
b

Bluring a

original

processed

L



a

b

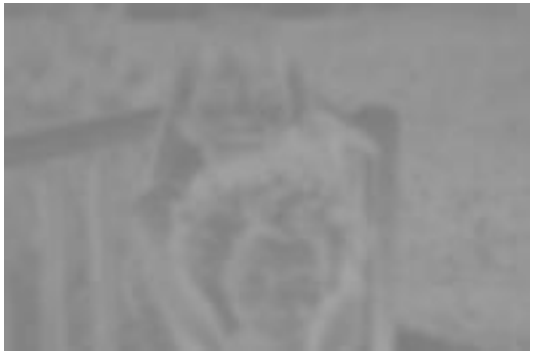
Bluring b

original

processed

L

a



b

Application to image compression

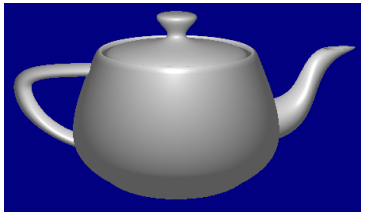
- (compression is about hiding differences from the true image where you can’t see them).

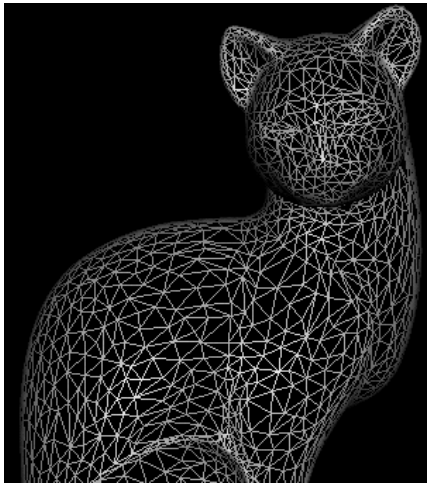
Where to now…

- We are now done with images
- We will spend several weeks on the mechanics of 3D graphics
  - 3D Transform
  - Coordinate systems and Viewing
  - Drawing lines and polygons
  - Lighting and shading
- We will finish the semester with modeling and some additional topics

3D Graphics Pipeline

Modeling
(Creating 3D Geometry)





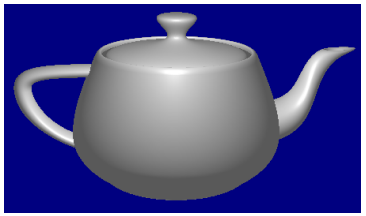
Rendering
(Creating, shading images from geometry, lighting, materials)

3D Graphics Pipeline

Modeling
(Creating 3D Geometry)

→

Rendering
(Creating, shading images from geometry, lighting, materials)



Want to place it at correct location in the world
Want to view it from different angles
Want to scale it to make it bigger or smaller
Need transformation between coordinate systems
-- Represent transformations using matrices and matrix-vector multiplications.

Recall: All 2D Linear Transformations

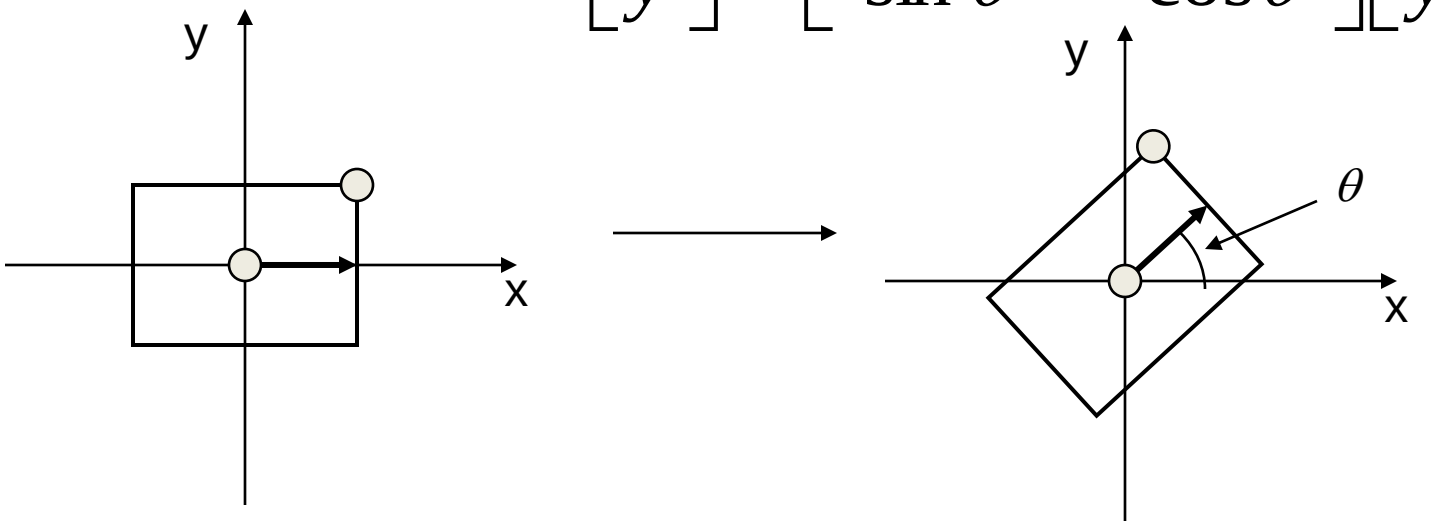
- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotation

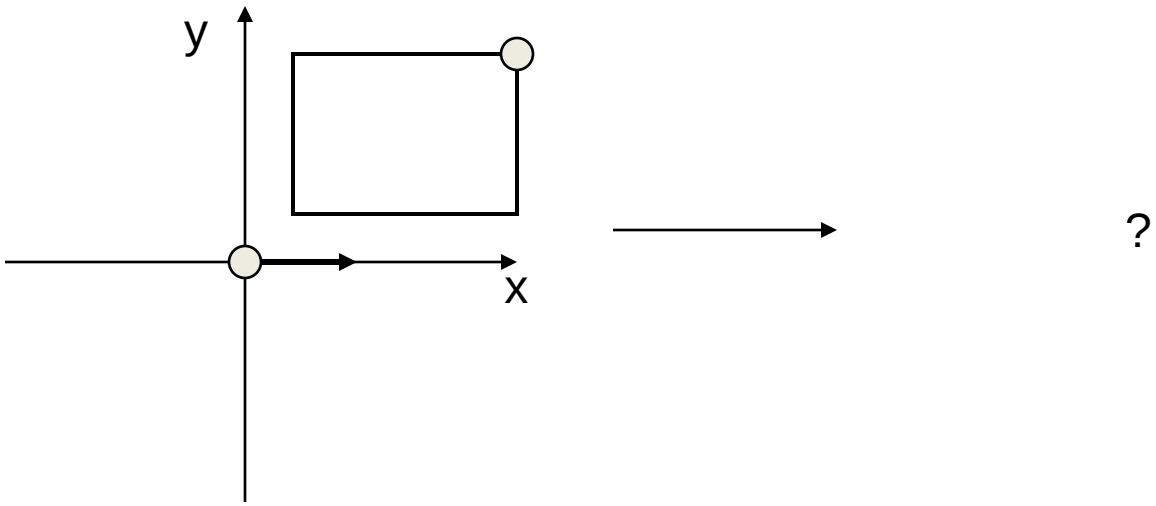
- Rotate counter-clockwise about the origin by an angle θ

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



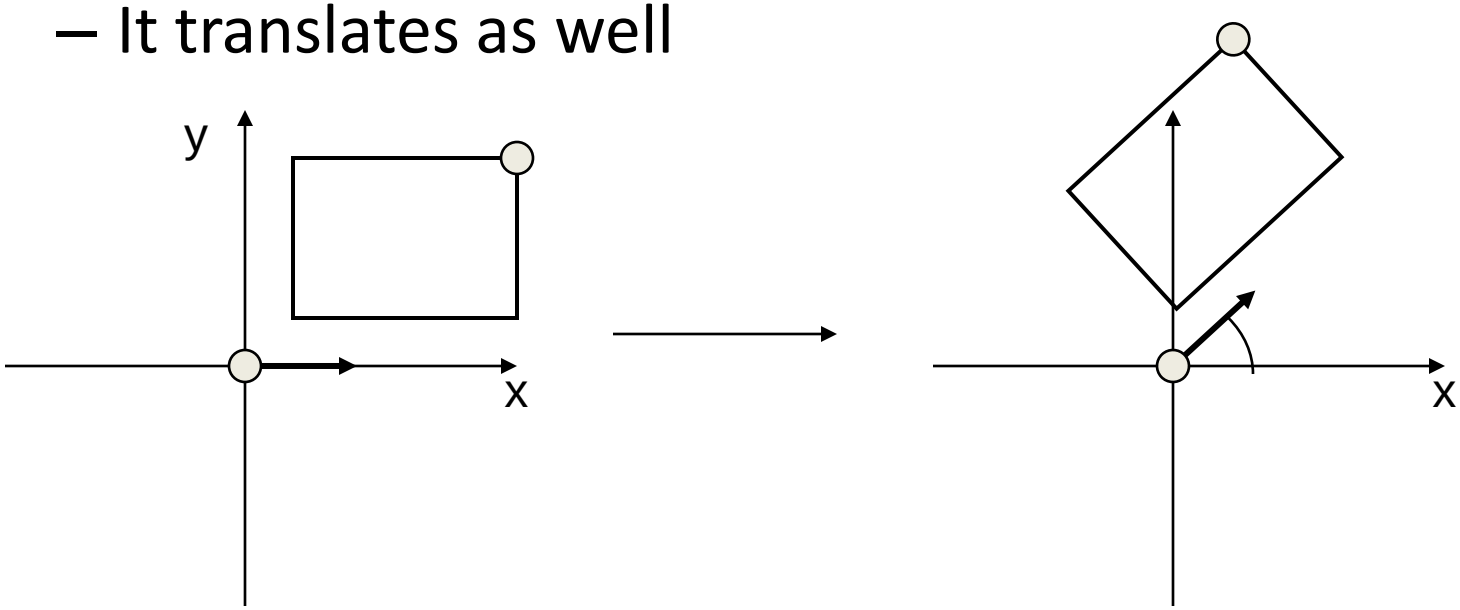
Rotating About An Arbitrary Point

- What happens when you apply a rotation transformation to an object that is not at the origin?



Rotating About An Arbitrary Point

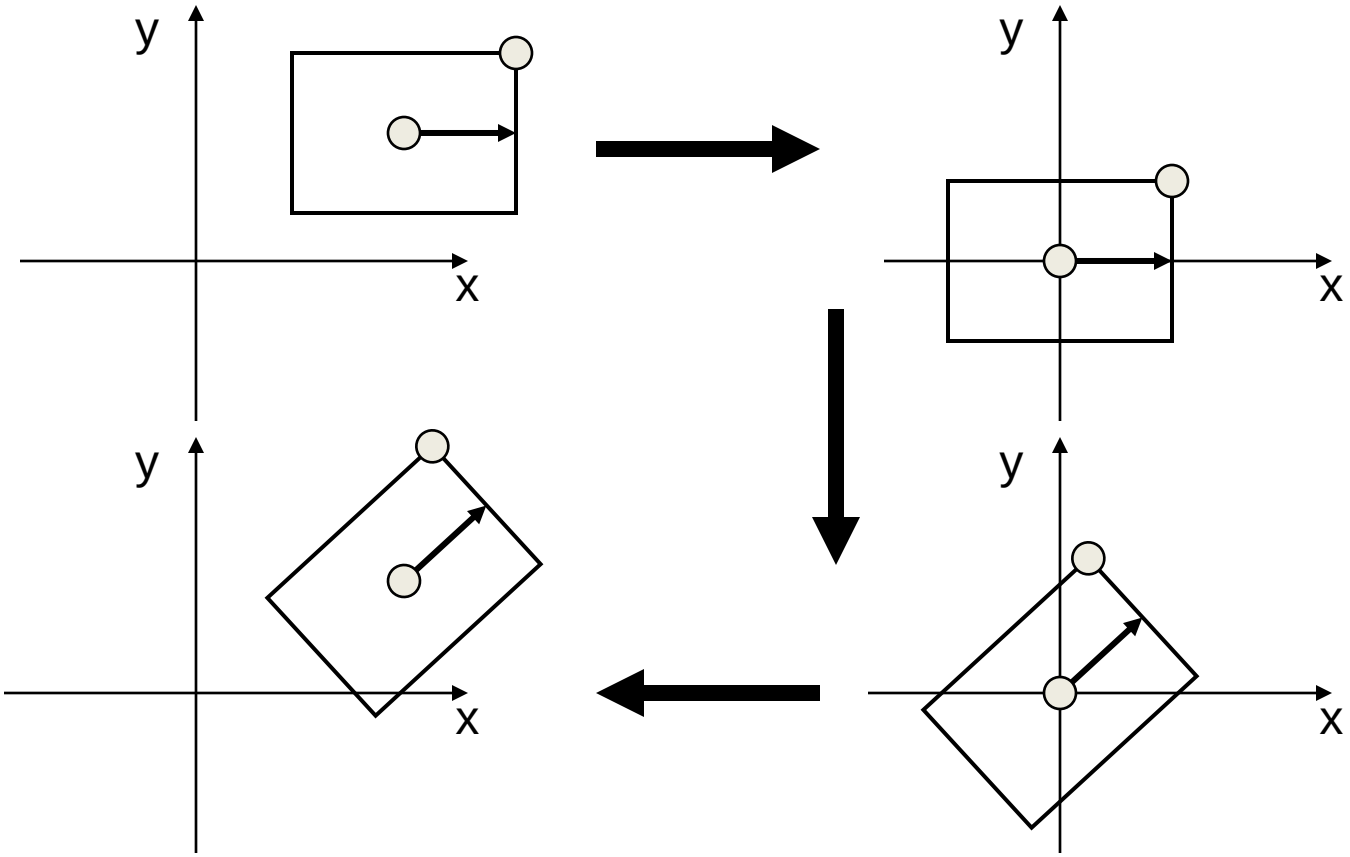
- What happens when you apply a rotation transformation to an object that is not at the origin?
  - It translates as well



How Do We Fix it?

- How do we rotate an about an arbitrary point?
  - Hint: we know how to rotate about the origin of a coordinate system

Rotating About An Arbitrary Point



Scaling an Object not at the Origin

- What happens if you apply the scaling transformation to an object not at the origin?
- Based on the rotating about a point composition, what should you do to resize an object about its own center?

Back to Rotation About a Pt

- Say ***R*** is the rotation matrix to apply, and ***p*** is the point about which to rotate
- Translation to Origin: $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{p}$
- Rotation: $\boldsymbol{x}'' = \boldsymbol{R}\boldsymbol{x}' = \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{p}) = \boldsymbol{R}\boldsymbol{x} - \boldsymbol{R}\boldsymbol{p}$
- Translate back: $\mathbf{x}''' = \mathbf{x}'' + \mathbf{p} = \mathbf{R}\mathbf{x} + \left(-\mathbf{R}\mathbf{p} + \mathbf{p}\right)$
- How to express all the transformation using matrix multiplication?

Homogeneous Coordinates

- Use three numbers to represent a point

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \text{ for any w } \neq 0, \text{usually w } = 1$$

$$\begin{bmatrix} x/w \\ y/w \end{bmatrix} \Leftarrow \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Translation can now be done with matrix multiplication!

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinates

- Use three numbers to represent a point

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \text{ for any w } \neq 0, \text{usually w } = 1$$

$$\begin{bmatrix} x/w \\ y/w \end{bmatrix} \Leftarrow \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Translation can now be done with matrix multiplication!

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M_1 M_2 M_3 M_4 M_5 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix}$$

Basic Transformations

- Translation:

$$\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation:

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

- Scaling:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composing rotations, scales

$$x_3 = Rx_2 \qquad x_2 = Sx_1$$

$$x_3 = R(Sx_1) = (RS)x_1$$

$$x_3 \neq SRx_1$$

Rotation and scaling are not commutative.

Inverting Composite Transforms

- Say I want to invert a combination of 3 transforms
- Option 1: Find composite matrix, invert
- Option 2: Invert each transform ***and swap order***

$$M = M_1 M_2 M_3$$

$$M^{-1} = M_3^{-1} M_2^{-1} M_1^{-1}$$

Inverting Composite Transforms

- Say I want to invert a combination of 3 transforms
- Option 1: Find composite matrix, invert
- Option 2: Invert each transform ***and swap order***
- Obvious from properties of matrices

$$M = M_1 M_2 M_3$$

$$M^{-1} = M_3^{-1} M_2^{-1} M_1^{-1}$$

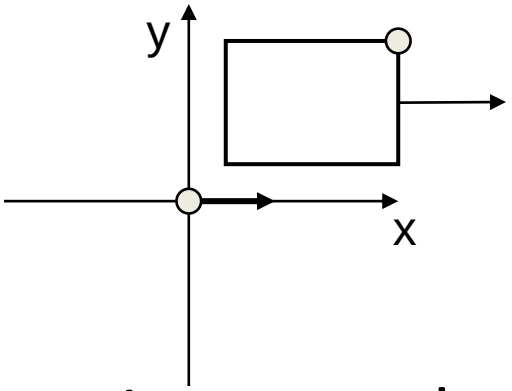
$$M^{-1} M = M_3^{-1}(M_2^{-1}(M_1^{-1} M_1) M_2) M_3$$

Homogeneous Transform Advantages

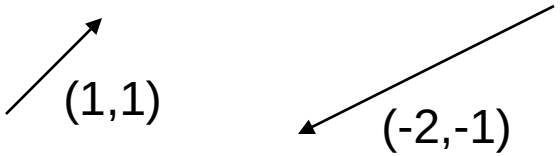
- Unified view of transformation as matrix multiplication
  - Easier in hardware and software
- To compose transformations, simply multiply matrices
  - Order matters: BA vs AB
- Allows for transforming directional vectors
- Allows for non-affine transformations:
  - Perspective projections!

Directions vs. Points

- We have been talking about transforming points



- Directions are also important in graphics
  - Viewing directions
  - Normal vectors
  - Ray directions



- Directions are represented by vectors, like points, and can be transformed, but not like points

Transforming Directions

- Say I define a direction as the difference of two points: $\boldsymbol{d=a-b}$
  - This represents the *direction* of the line between two points
- Now I translate the points by the same amount: $\boldsymbol{a'=a+t,\ b'=b+t}$
- $\boldsymbol{d'=a'-b'=d}$
- How should I transform $\boldsymbol{d}$?

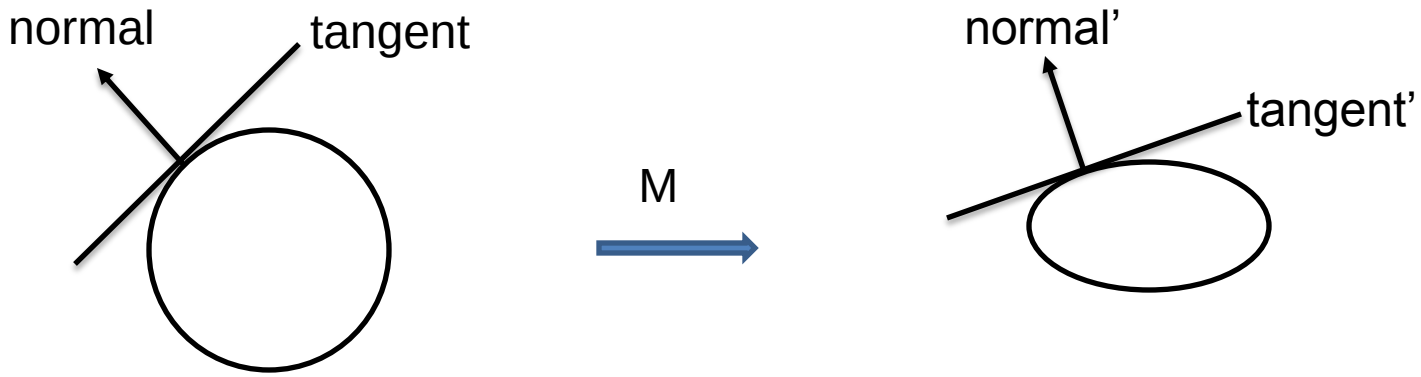
Homogeneous Directions

- Translation does not affect directions!
- Homogeneous coordinates give us a very clean way of handling this
- The direction *(x,y)* becomes the homogeneous direction *(x,y,0)*

$$\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

- The correct thing happens for rotation and scaling also
  - Scaling changes the length of the vector, but not the direction
  - Normal vectors are slightly different – we'll see more later

Transforming normal vectors



$$\mathbf{n}^{\mathrm{T}}\mathbf{t}=0$$

$$\mathbf{t}'=\boxed{\mathbf{M}\mathbf{t}}$$

$$\mathbf{n}'^{\mathrm{T}}\,\mathbf{t}'=0$$

$$(\mathbf{n}^{\mathrm{T}}\mathbf{M}^{-1})(\mathbf{M}\mathbf{t})=0$$

If M is a rotation,

$$(\mathbf{M}^{-1})^{\mathrm{T}}=\mathbf{M}$$

$$\mathbf{n}'=(\mathbf{n}^{\mathrm{T}}\mathbf{M}^{-1})^{\mathrm{T}}=\boxed{(\mathbf{M}^{-1})^{\mathrm{T}}\mathbf{n}}$$

3D Transformations

- Homogeneous coordinates: *(x,y,z)=(wx,wy,wz,w)*

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} \text{ for any w} \neq 0, \text{usually w} = 1$$

$$\begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix} \Leftarrow \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- Transformations are now represented as 4x4 matrices

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

3D Affine Transform

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Rotation

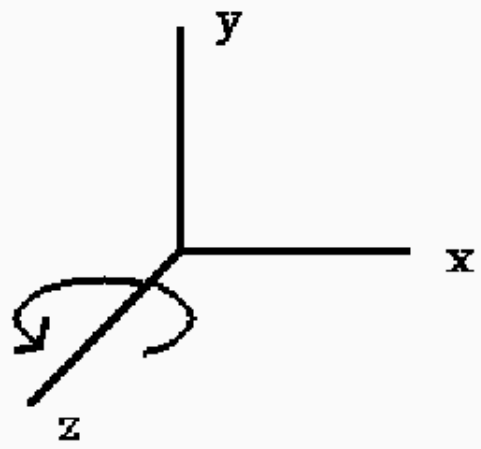
- Rotation in 3D is about an *axis* in 3D space passing through the origin
- Using a matrix representation, any matrix with an *orthonormal* top-left 3x3 sub-matrix is a rotation
  - Rows/columns are mutually orthogonal (0 dot product)
  - Determinant is 1
  - Implies columns are also orthogonal, and that the transpose is equal to the inverse

$$R = \begin{bmatrix} | & | & | & 0 \\ \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & 0 \\ | & | & | & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ then } \mathbf{r}_1 \bullet \mathbf{r}_2 = 0, \mathbf{r}_1 \bullet \mathbf{r}_3 = 0, \mathbf{r}_2 \bullet \mathbf{r}_3 = 0, \mathbf{r}_1 \bullet \mathbf{r}_1 = 1, \mathbf{r}_2 \bullet \mathbf{r}_2 = 1, \mathbf{r}_3 \bullet \mathbf{r}_3 = 1.$$

Specifying a rotation matrix

Z-Axis Rotation

Z-axis rotation is identical to the 2D case:



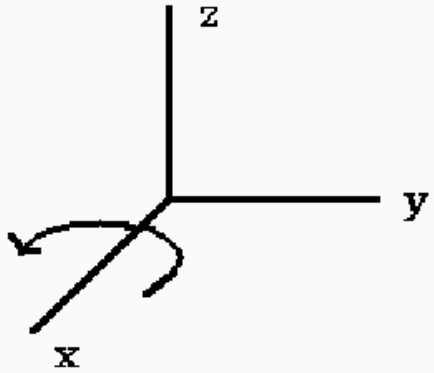
x' = x*cos q - y*sin q
y' = x*sin q + y*cos q
z' = z

$$Rz(q) = \begin{pmatrix} \cos q & \sin q & 0 & 0 \\ -\sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

X-Axis Rotation

X-axis rotation looks like Z-axis rotation if replace:

X axis with Y axis
Y axis with Z axis
Z axis with X axis



So we do the same replacement in the equations:

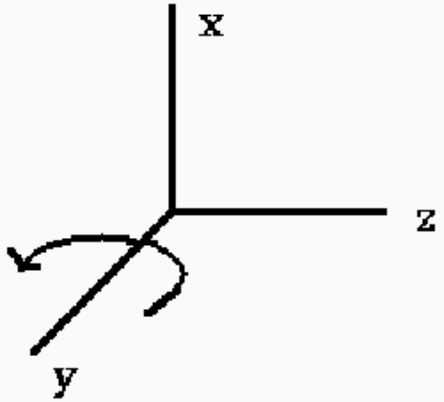
y' = y*cos q - z*sin q
z' = y*sin q + z*cos q
x' = x

$$Rx(q) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q & \sin q & 0 \\ 0 & -\sin q & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Y-Axis Rotation

Y-axis rotation looks like Z-axis rotation if replace:

X axis with Z axis
Y axis with X axis
Zaxis with Y axis



So we do the same replacement in equations :

z' = z*cos q - x*sin q
x' = z*sin q + x*cos q
y' = y

$$Ry(q) = \begin{pmatrix} \cos q & 0 & -\sin q & 0 \\ 0 & 1 & 0 & 0 \\ \sin q & 0 & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

http://pratt.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/3drota.htm

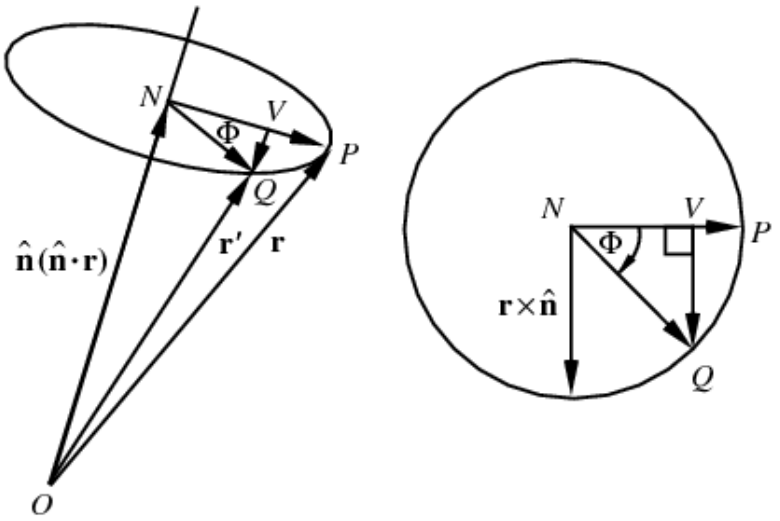
Specifying a rotation matrix

- *Euler angles:* Specify how much to rotate about X, then how much about Y, then how much about Z
  - Hard to think about, and hard to compose

$$[\mathbf{R}] = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Alternative Representations

- Specify the axis and the angle (OpenGL method)
  - Hard to compose multiple rotations



A rotation by an angle $\theta \in \mathbb{R}$ around axis specified by the unit vector $\hat{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ is given by $I + \tilde{\omega} \sin\theta + \tilde{\omega}^2 (1 - \cos\theta)$

$$\tilde{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

http://mathworld.wolfram.com/RodriguesRotationFormula.html

Non-Commutativity

- Not Commutative (unlike in 2D)!!
- Rotate by x, then y is not same as y then x
- Order of applying rotations does matter
- Follows from matrix multiplication not commutative
  - R1 * R2 is not the same as R2 * R1

Other Rotation Issues

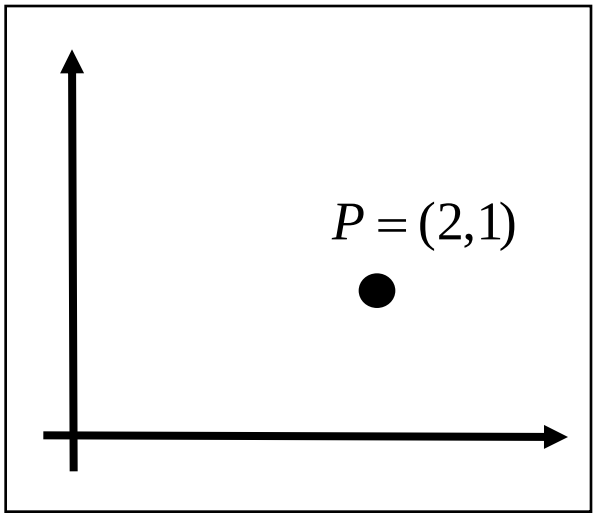
- Rotation is about an axis at the origin
  - For rotation about an arbitrary axis, use the same trick as in 2D: Translate the axis to the origin, rotate, and translate back again

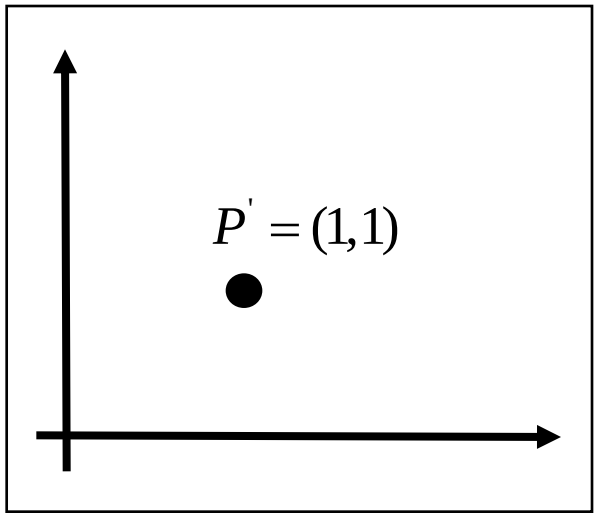
Transformation Leftovers

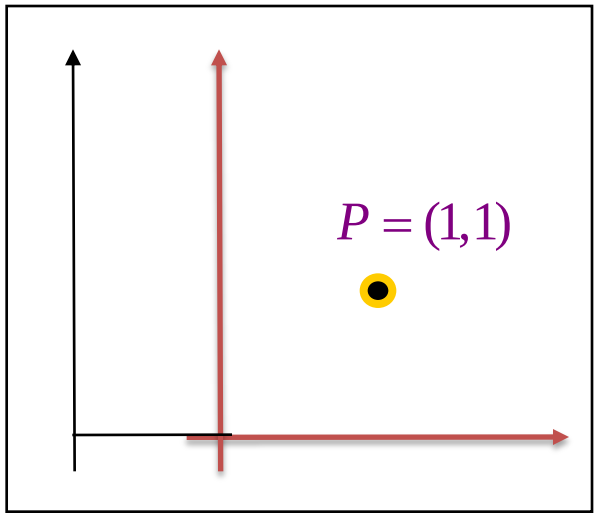
- Scale, shear etc extend naturally from 2D to 3D
- Rotation and Translation are the *rigid-body transformations:*
  - Do not change lengths or angles, so a body does not deform when transformed

Coordinate Frames

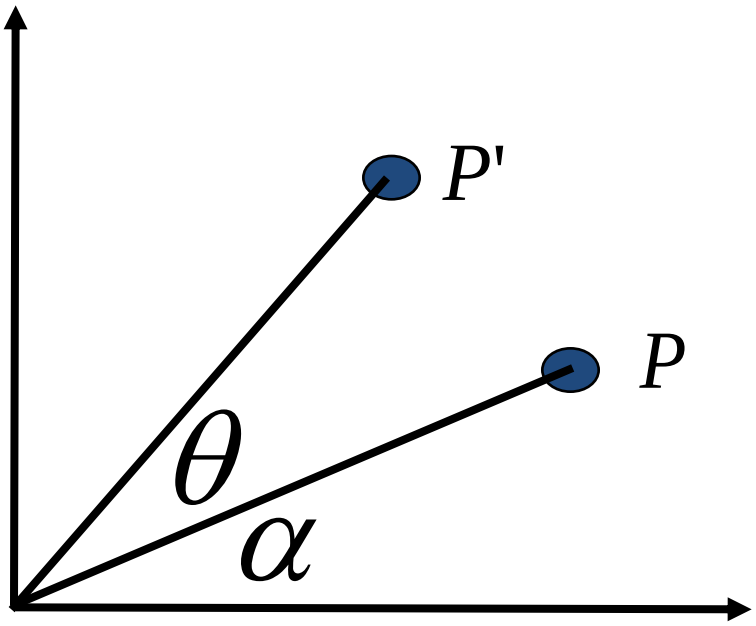
- All of discussion in terms of operating on points
- But can also change coordinate system
- Example, motion means either point moves backward, or coordinate system moves forward

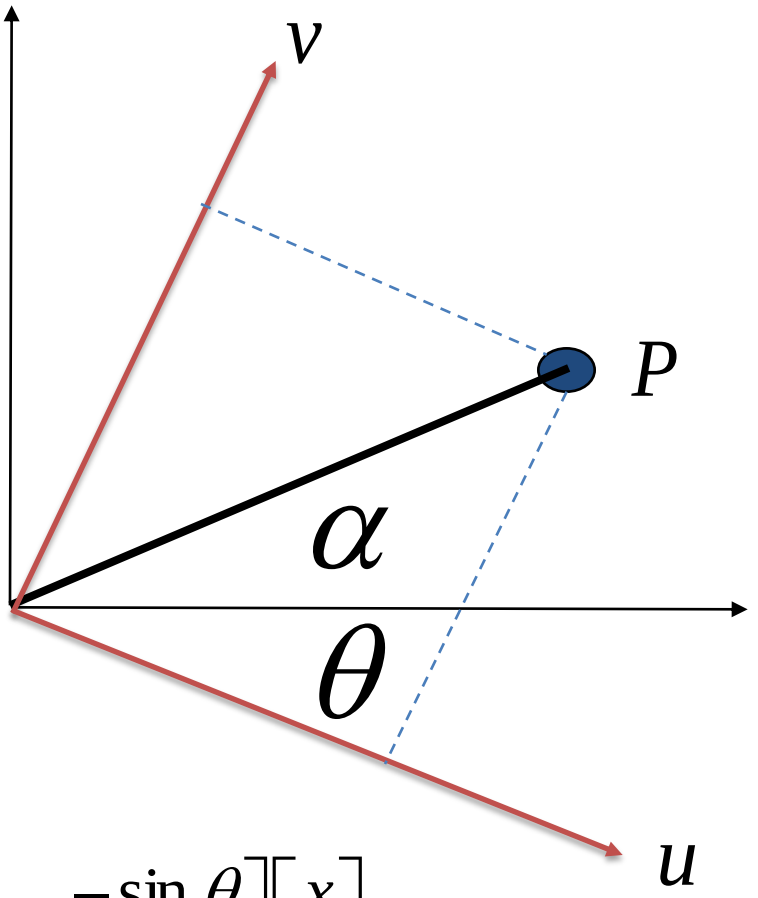






Coordinate Frames: Rotations





$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

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

Geometric Interpretation 3D Rotations

- Rows of matrix are 3 unit vectors of new coord frame
- Can construct rotation matrix from 3 orthonormal vectors
- Effectively, projections of point into new coord frame

$$R_{uvw} = \begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{pmatrix}$$

$$Rp = \begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = ? \quad \begin{pmatrix} u \bullet p \\ v \bullet p \\ w \bullet p \end{pmatrix}$$