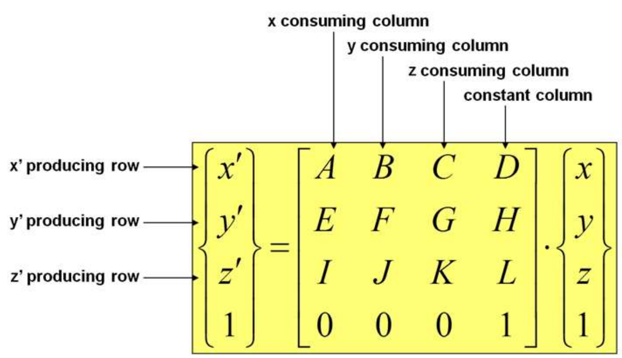
# Transformations

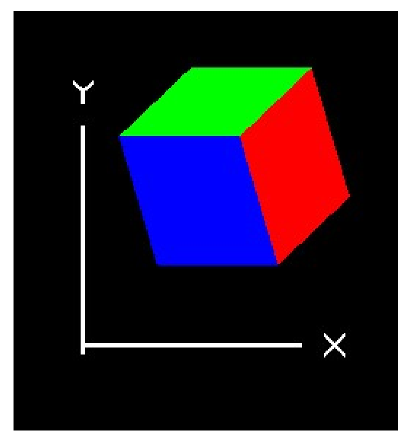


**Mike Bailey**

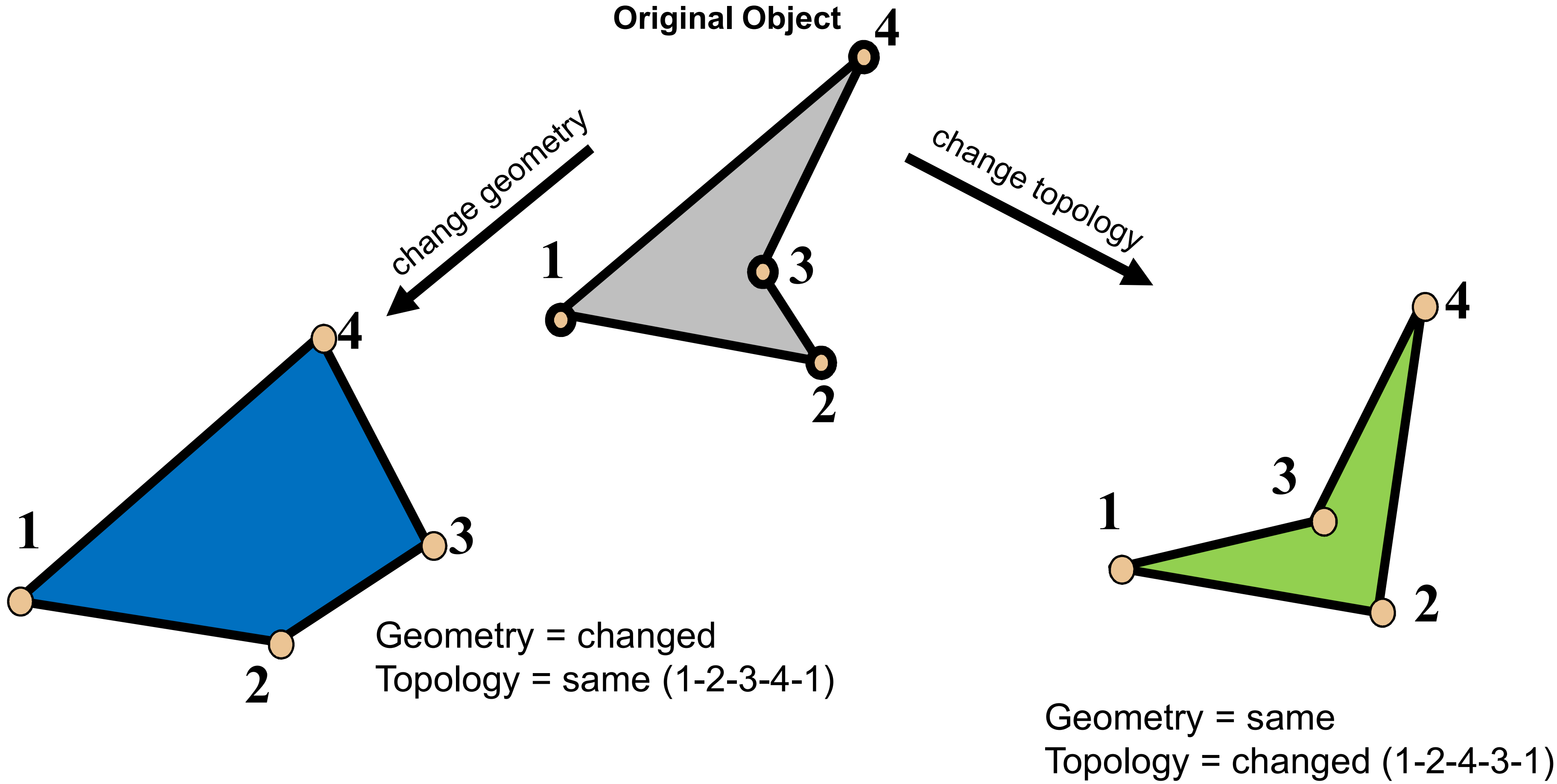
**mjb@cs.oregonstate.edu**




$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$



# Geometry vs. Topology



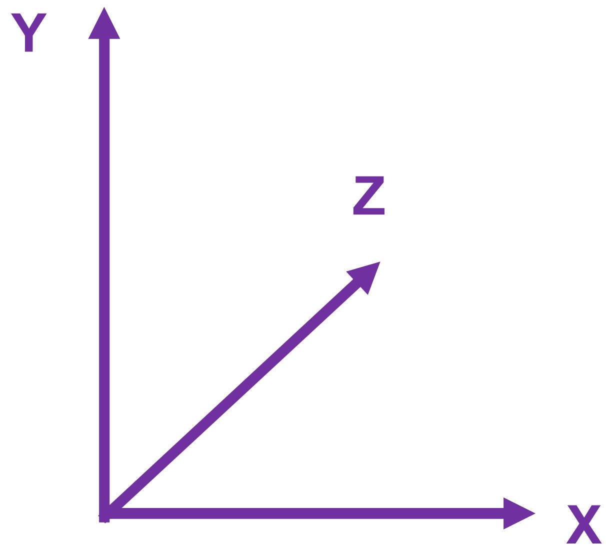

## Geometry:

**Where things are (e.g., coordinates)**

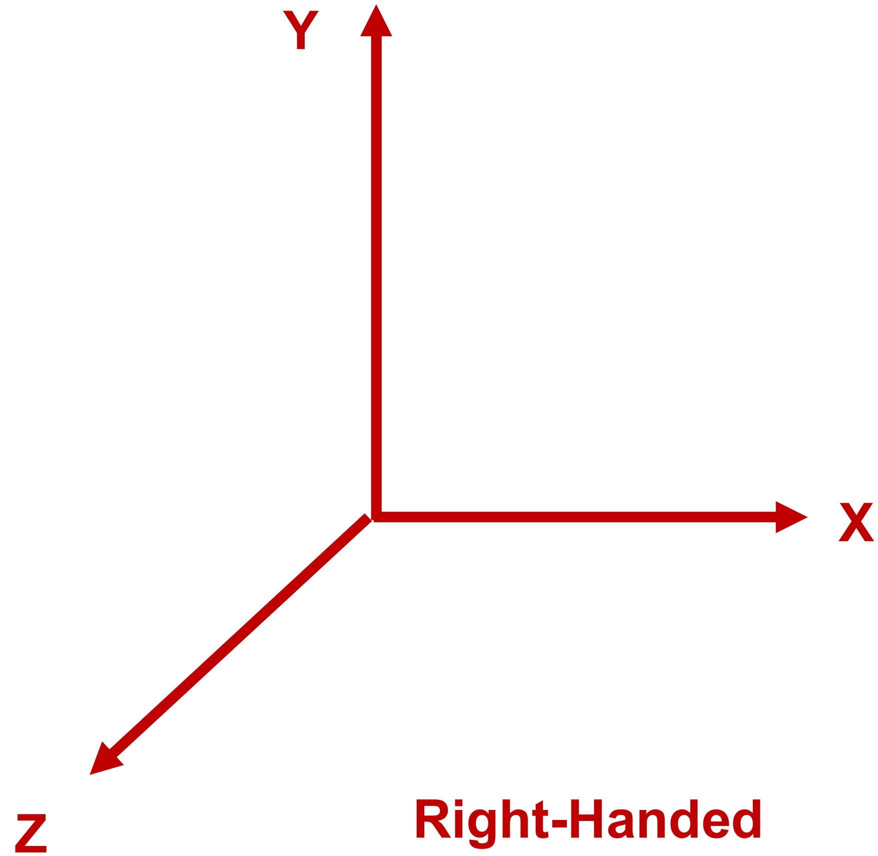
## Topology:

**How things are connected**

# 3D Coordinate Systems

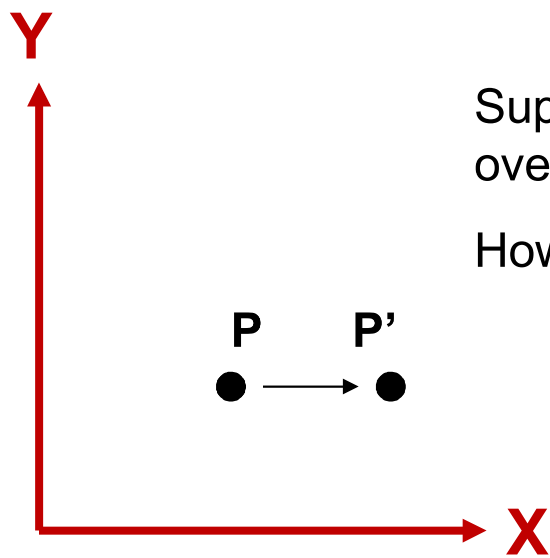



Left-Handed




Right-Handed

# Transformations




Suppose you have a point P and you want to move it over by 2 units in X.

How would you change P's coordinates?

$$P' = (P'_x, P'_y) = (P_x + 2., P_y)$$

This is known as a coordinate *transformation*

# General Form of 3D Linear Transformations

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

It's called a "Linear Transformation" because all of the coordinates are raised to the 1st power, that is, there are no $x^2$, $x^3$, etc. terms.

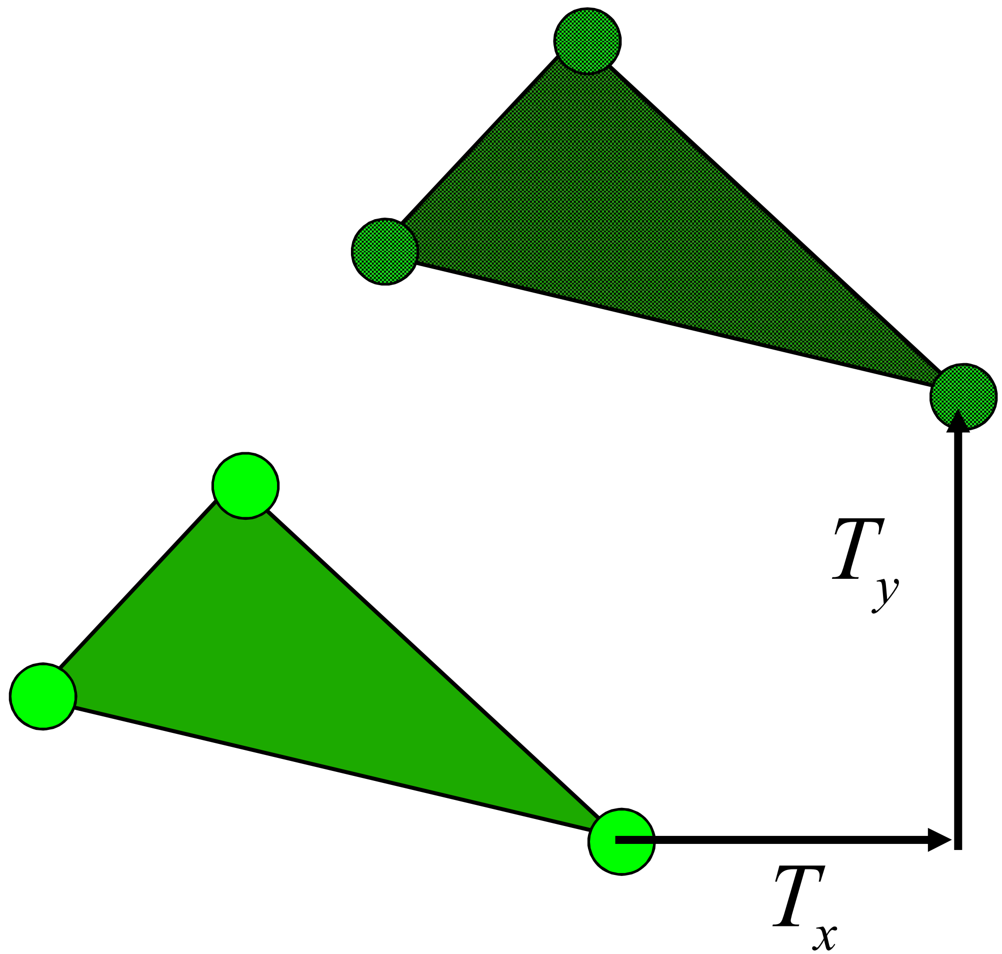
**Transform the geometry – leave the topology as is**

# Translation Equations

$$x' = x + T_x$$

$$y' = y + T_y$$

$$z' = z + T_z$$

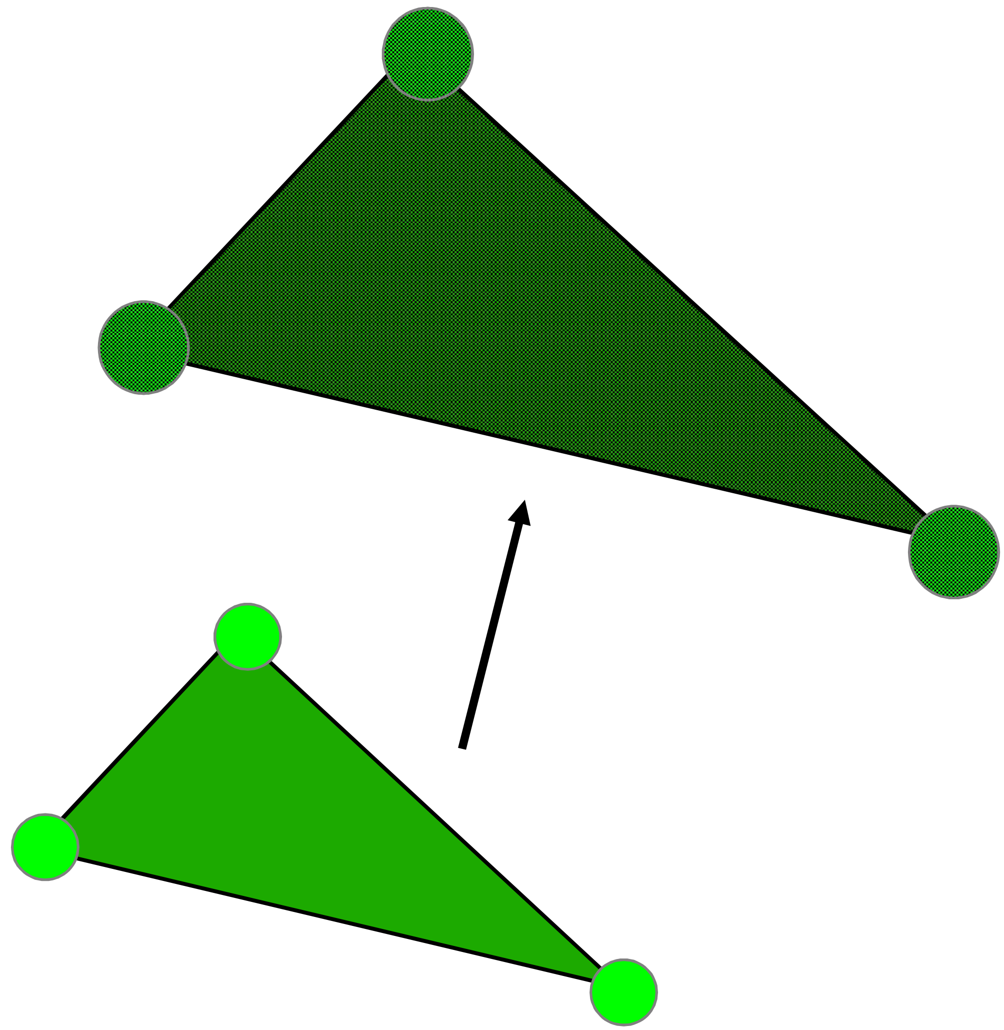


# Scaling About the Origin

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

$$z' = z \cdot S_z$$

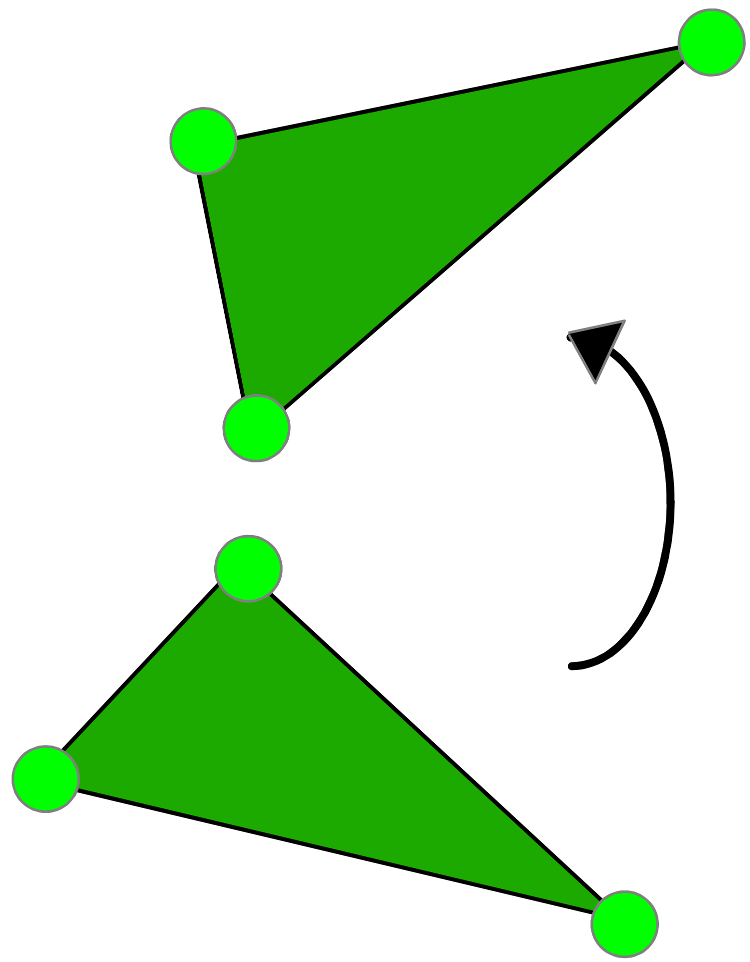


Oregon State University
Computer Graphics

# 2D Rotation About the Origin

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

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



# Linear Equations in Matrix Form

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column

y consuming column

z consuming column

constant column

x' producing row

y' producing row

z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

# Identity Matrix ( [ I ] )

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column, y consuming column, z consuming column, constant column

x' producing row, y' producing row, z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$x' = x$$
$$y' = y$$
$$z' = z$$

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

[ I ] signifies that “Nothing has changed”

# Matrix Inverse

$$[M] \bullet [M]^{-1} = [I]$$

$$[M] \bullet [M]^{-1} = \text{“Nothing has changed”}$$

“Whatever [M] does, $[M]^{-1}$ undoes”

# Translation Matrix

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column
y consuming column
z consuming column
constant column

x' producing row
y' producing row
z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

Quick! What is the inverse of this matrix?

# Scaling Matrix

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column
y consuming column
z consuming column
constant column

x' producing row
y' producing row
z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

Quick! What is the inverse of this matrix?

# 3D Rotation Matrix About Z

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column, y consuming column, z consuming column, constant column

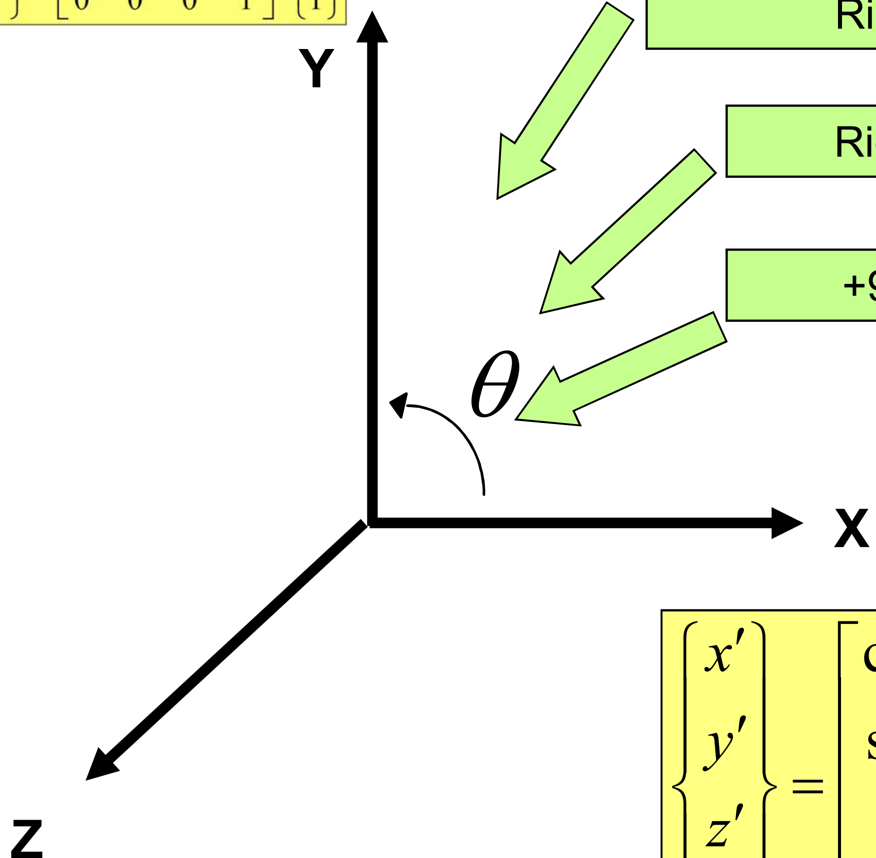
x' producing row, y' producing row, z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

Right-handed coordinates

Right-handed positive rotation rule

+90º rotation gives:  y' = x




$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

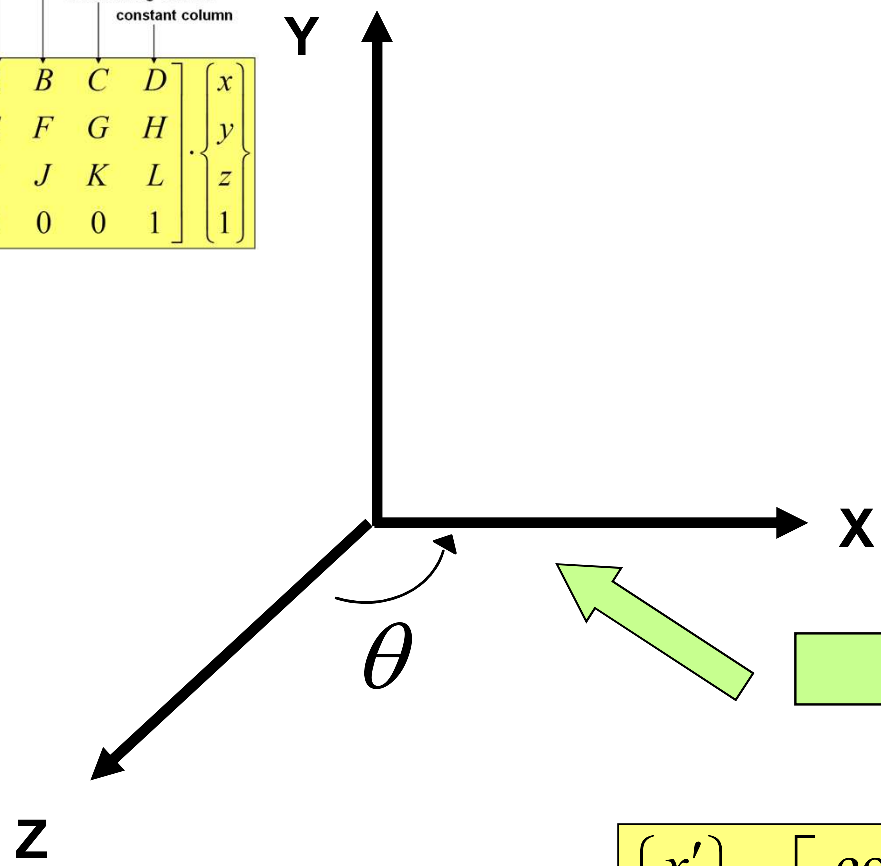
# 3D Rotation Matrix About Y

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column
y consuming column
z consuming column
constant column

x' producing row
y' producing row
z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$




+90° rotation gives: x' = z

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

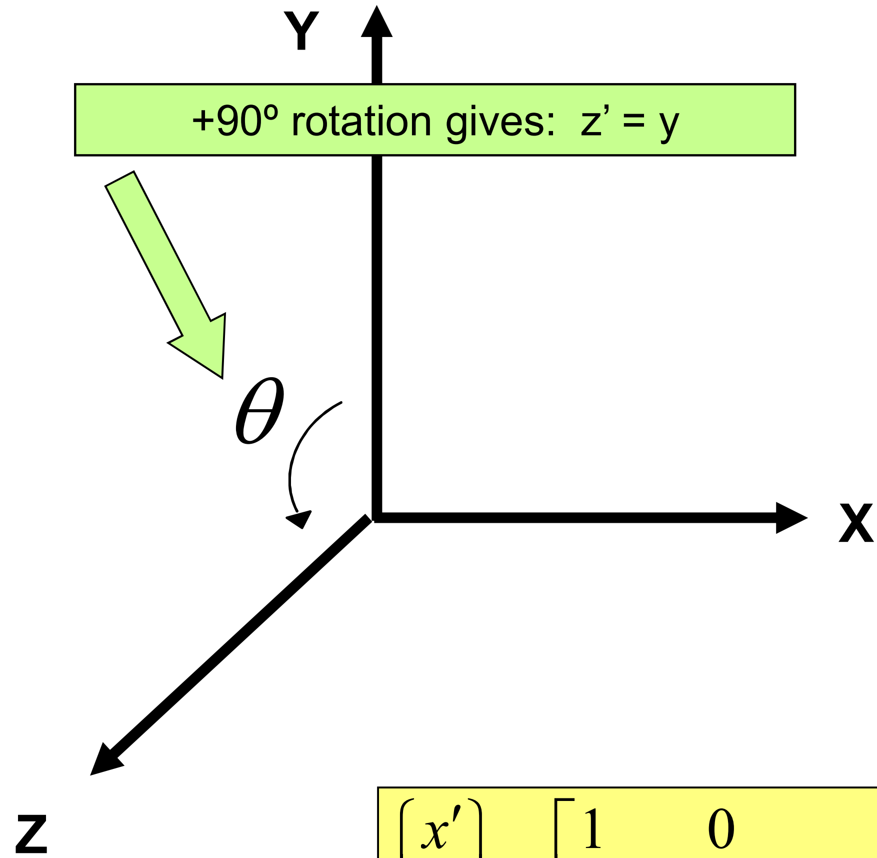
# 3D Rotation Matrix About X

$$x' = Ax + By + Cz + D$$
$$y' = Ex + Fy + Gz + H$$
$$z' = Ix + Jy + Kz + L$$

x consuming column
y consuming column
z consuming column
constant column

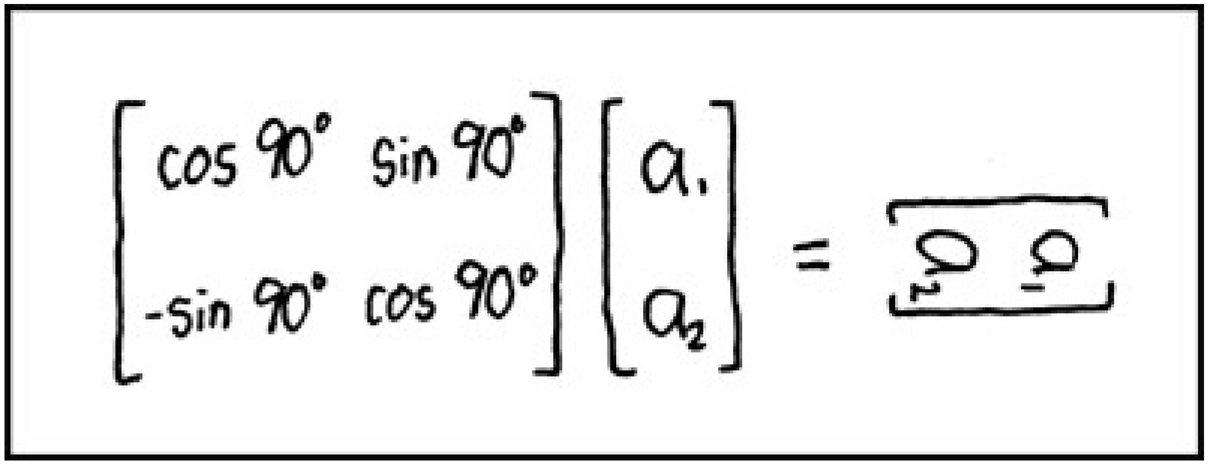
x' producing row
y' producing row
z' producing row

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$



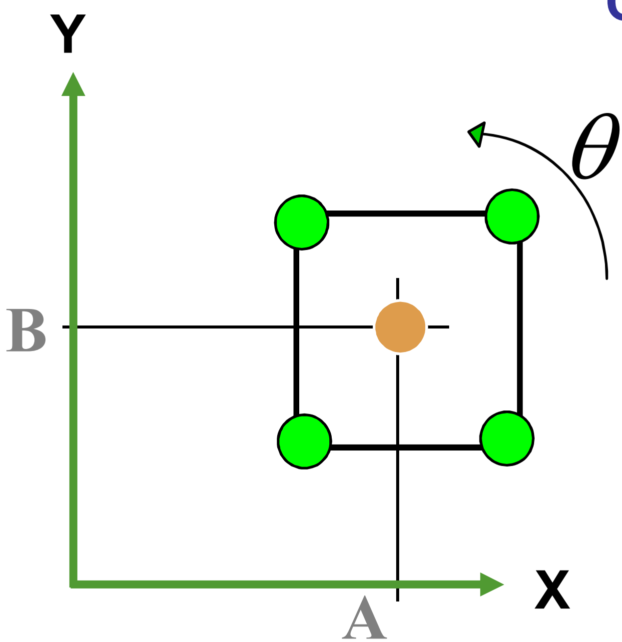

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

# How it Really Works :-)




http://xkcd.com

# Compound Transformations




**Q:** Our rotation matrices only work around the origin. What if we want to rotate about an arbitrary point (A,B)?

**A:** Use more than one matrix.

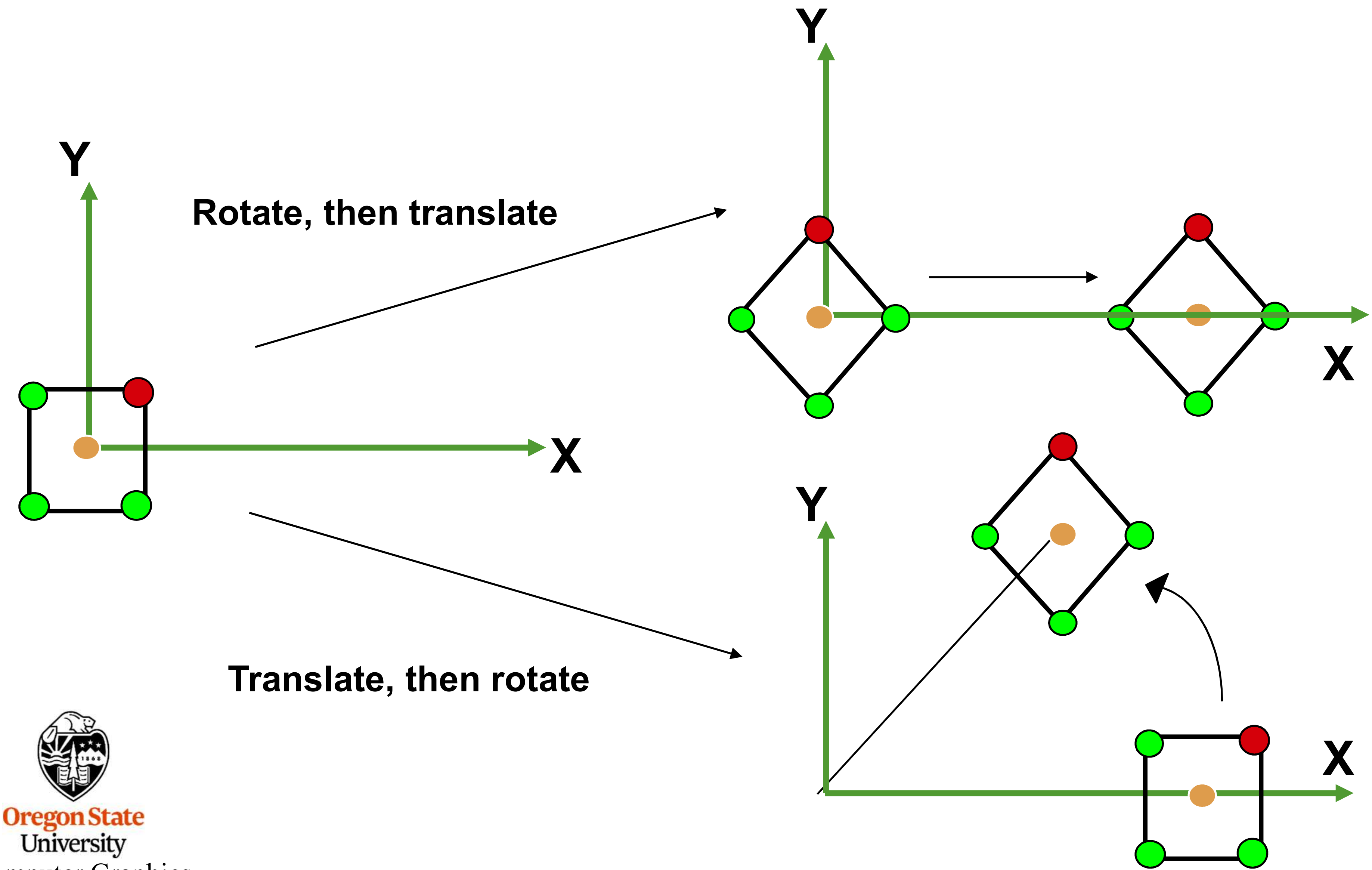
Write it →

3 2 1

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left( \left[T_{+A,+B}\right] \cdot \left( \left[R_\theta\right] \cdot \left( \left[T_{-A,-B}\right] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \right) \right) \right)$$

← Say it

# Matrix Multiplication *is not* Commutative




Oregon State University
Computer Graphics

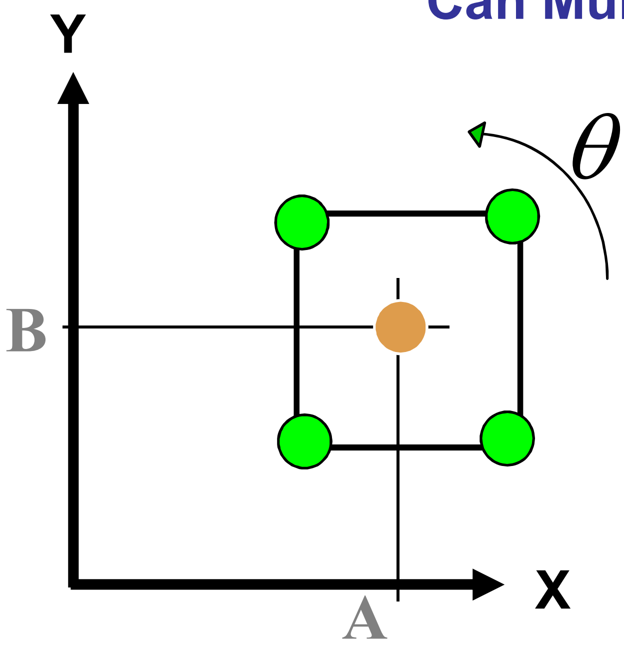
# Matrix Multiplication *is* Associative

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left( \left[ T_{+A,+B} \right] \cdot \left( \left[ R_\theta \right] \cdot \left( \left[ T_{-A,-B} \right] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \right) \right) \right)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \underbrace{\left( \left[ T_{+A,+B} \right] \cdot \left[ R_\theta \right] \cdot \left[ T_{-A,-B} \right] \right)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

One matrix –
the Current Transformation Matrix, or CTM

# Can Multiply All Geometry by One Matrix !

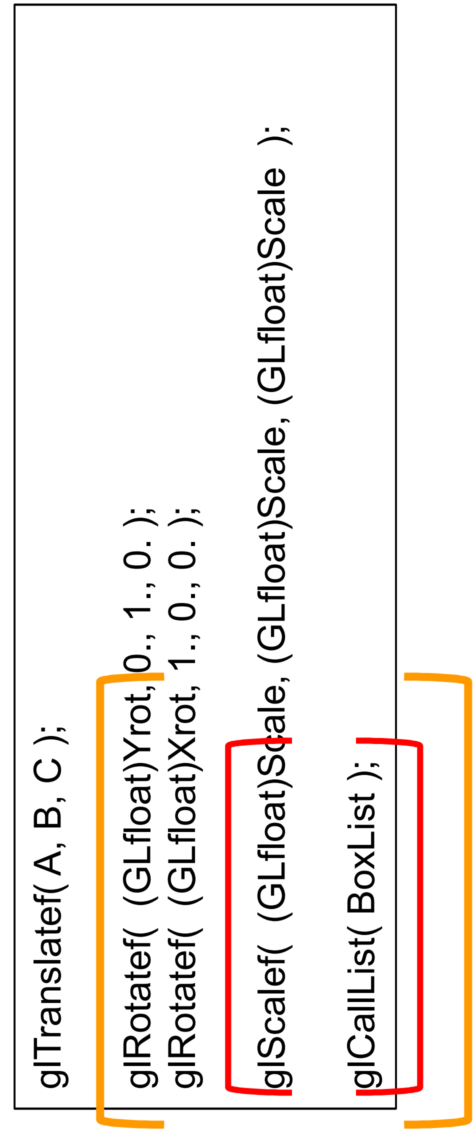



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = [M] \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

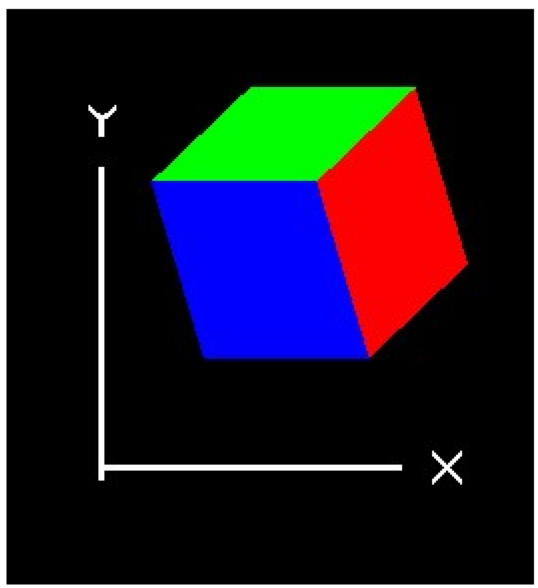
Graphics hardware can do this very quickly!

# OpenGL Will Do the Transformation Compounding for You !

```
glTranslatef( A, B, C );

glRotatef(  (GLfloat)Yrot, 0., 1., 0. );
glRotatef(  (GLfloat)Xrot, 1., 0., 0. );

glScalef(  (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale  );

glCallList( BoxList );
```

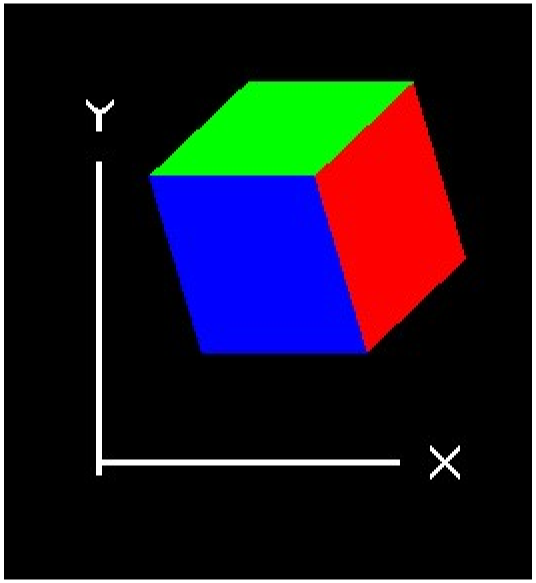



Typically objects are modeled around their own local origin, so the
glTranslate( -A, -B, -C )
step is unnecessary.

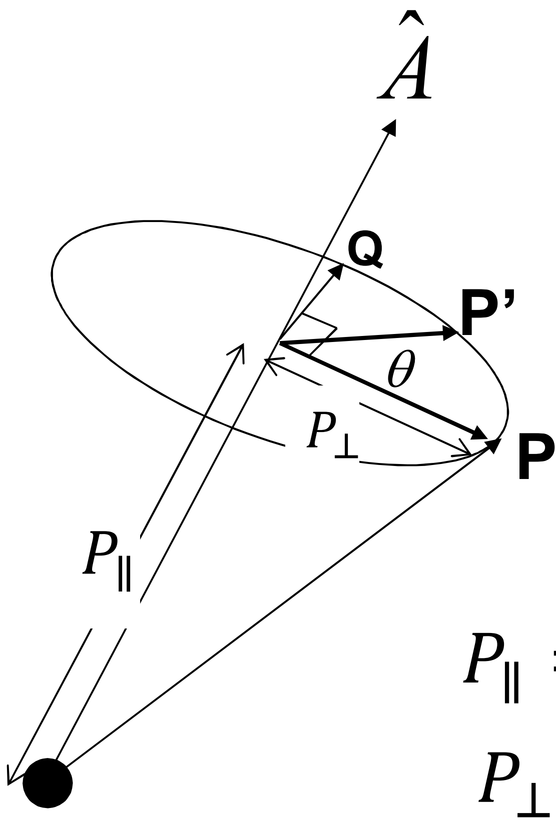


# OpenGL Will Do the Transformation Compounding for You !

```
for( ; ; )
{
        << Turn mouse position into Xrot and Yrot >>

        glLoadIdentity( );

        glTranslatef( A, B, C );

        glRotatef(  (GLfloat)Yrot, 0., 1., 0. );
        glRotatef(  (GLfloat)Xrot, 1., 0., 0. );

        glScalef(  (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale  );

        glCallList( BoxList );
}
```



# The Funky Rotation Matrix for an Arbitrary Axis and Angle




$$P_\parallel = \hat{A}(\hat{A} \bullet P)$$

$$P_\perp = P - P_\parallel = P - \hat{A}(\hat{A} \bullet P)$$

$$Q = \hat{A} \times P_\perp = \hat{A} \times P_\perp + 0 = \hat{A} \times P_\perp + \hat{A} \times P_\parallel = \hat{A} \times (P_\perp + P_\parallel) = \hat{A} \times P$$

# The Funky Rotation Matrix for an Arbitrary Axis and Angle

$$P_{\parallel}' = P_{\parallel} \qquad P_{\perp}' = P_{\perp}\cos\theta + Q\sin\theta$$

$$P' = P'_{\parallel} + P'_{\perp}$$

$$P' = \left[\hat{A}(\hat{A} \bullet P)\right] + \cos\theta\left[P - \hat{A}(\hat{A} \bullet P)\right] + \sin\theta\left[\hat{A} \times P\right]$$

# The Funky Rotation Matrix for an Arbitrary Axis and Angle

$$\hat{A}(\hat{A} \bullet P) = \begin{bmatrix} A_x A_x & A_x A_y & A_x A_z \\ A_y A_x & A_y A_y & A_y A_z \\ A_z A_x & A_z A_y & A_z A_z \end{bmatrix} \{P\}$$

$$\hat{A} \times P = \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix} \{P\}$$

# The Funky Rotation Matrix for an Arbitrary Axis and Angle

$$P' = \left[\hat{A}(\hat{A} \bullet P)\right] + \cos\theta \left[P - \hat{A}(\hat{A} \bullet P)\right] + \sin\theta \left[\hat{A} \times P\right]$$

$$P' = \left( \begin{bmatrix} A_xA_x & A_xA_y & A_xA_z \\ A_yA_x & A_yA_y & A_yA_z \\ A_zA_x & A_zA_y & A_zA_z \end{bmatrix} + \cos\theta \left[ [I] - \begin{bmatrix} A_xA_x & A_xA_y & A_xA_z \\ A_yA_x & A_yA_y & A_yA_z \\ A_zA_x & A_zA_y & A_zA_z \end{bmatrix} \right] + \sin\theta \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix} \right) \{P\}$$

$$[M] = \left( \begin{bmatrix} A_xA_x & A_xA_y & A_xA_z \\ A_yA_x & A_yA_y & A_yA_z \\ A_zA_x & A_zA_y & A_zA_z \end{bmatrix} + \cos\theta \begin{bmatrix} 1-A_xA_x & -A_xA_y & -A_xA_z \\ -A_yA_x & 1-A_yA_y & -A_yA_z \\ -A_zA_x & -A_zA_y & 1-A_zA_z \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix} \right)$$

# The Funky Rotation Matrix for an Arbitrary Axis and Angle

$$[M] = \left( \begin{bmatrix} A_xA_x & A_xA_y & A_xA_z \\ A_yA_x & A_yA_y & A_yA_z \\ A_zA_x & A_zA_y & A_zA_z \end{bmatrix} + \cos\theta \begin{bmatrix} 1-A_xA_x & -A_xA_y & -A_xA_z \\ -A_yA_x & 1-A_yA_y & -A_yA_z \\ -A_zA_x & -A_zA_y & 1-A_zA_z \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix} \right)$$

$$[M] = \begin{bmatrix} A_xA_x + \cos\theta(1-A_xA_x) & A_xA_y - \cos\theta\left(A_xA_y\right) - \sin\theta A_z & A_xA_z - \cos\theta\left(A_xA_z\right) + \sin\theta A_y \\ A_yA_x - \cos\theta\left(A_yA_x\right) + \sin\theta A_z & A_yA_y + \cos\theta\left(1-A_yA_y\right) & A_yA_z - \cos\theta\left(A_yA_z\right) - \sin\theta A_x \\ A_zA_x - \cos\theta\left(A_zA_x\right) - \sin\theta A_y & A_zA_y - \cos\theta\left(A_zA_y\right) + \sin\theta A_x & A_zA_z + \cos\theta\left(1-A_zA_z\right) \end{bmatrix}$$

For this to work, A must be a unit vector