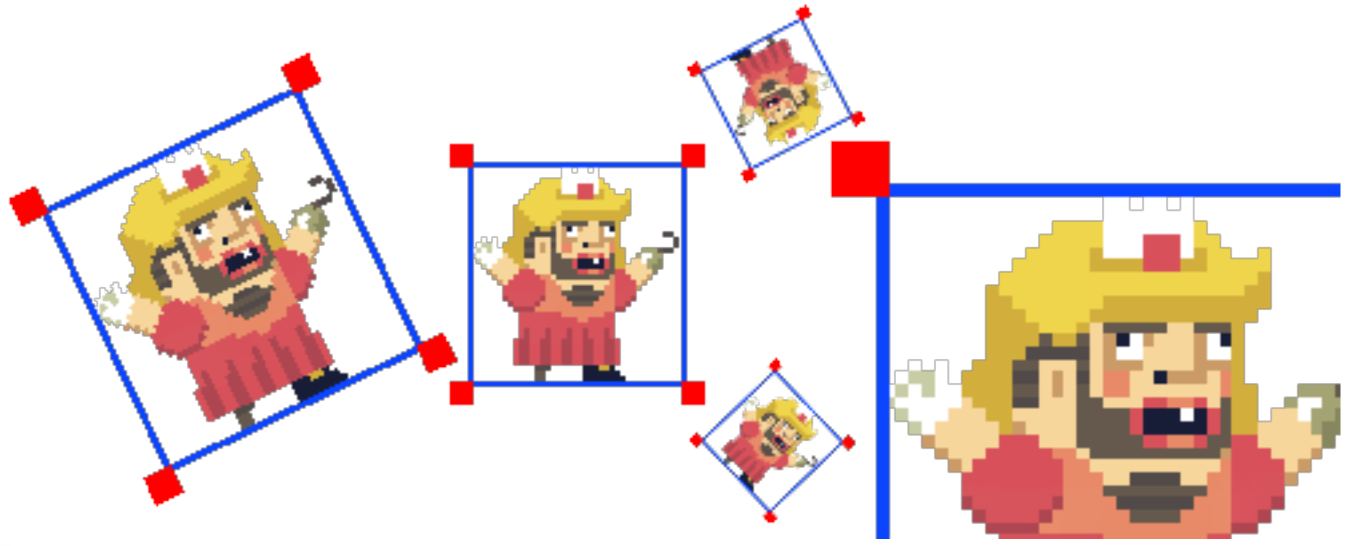
Fall 2018

CSCI 420: **Computer Graphics**

2.2 Transformations

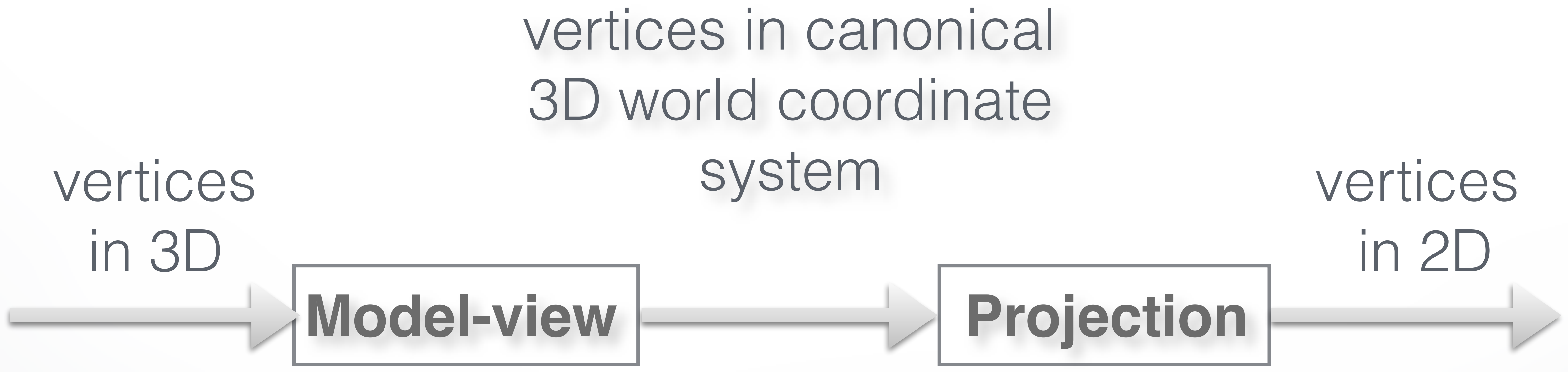


Hao Li

http://cs420.hao-li.com

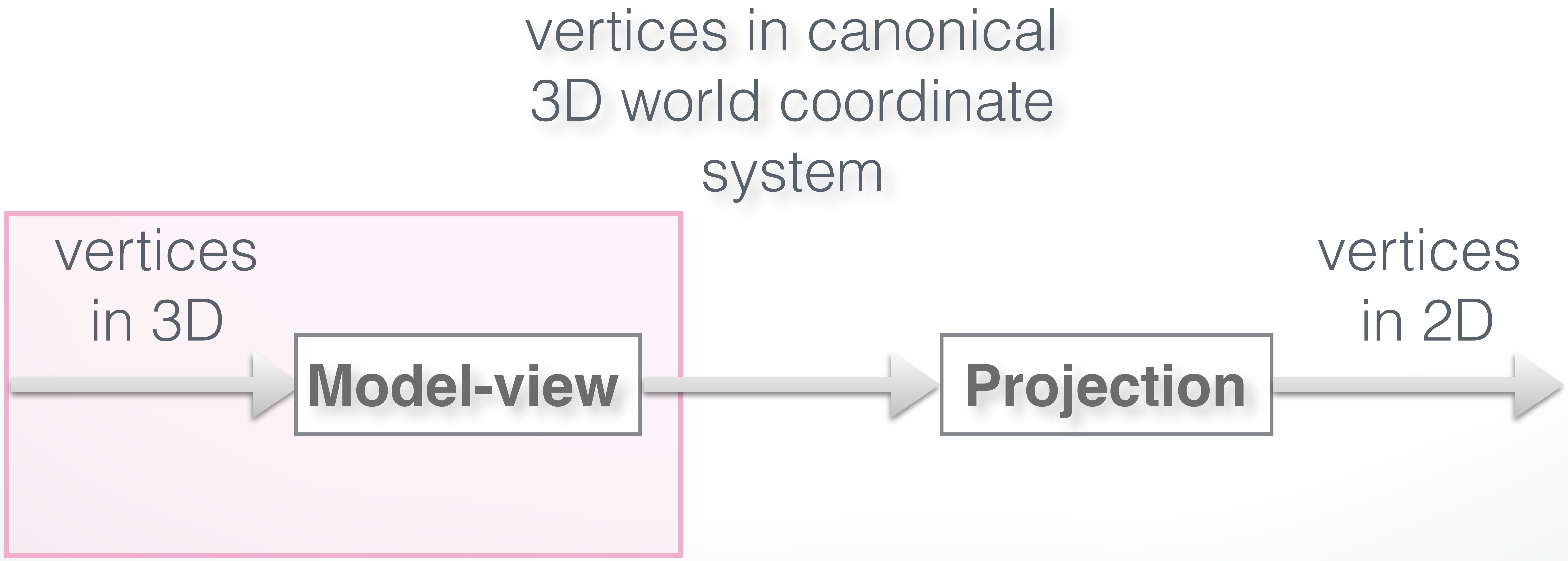
OpenGL Transformations Matrices

- Model-view matrix (4x4 matrix)
- Projection matrix (4x4 matrix)



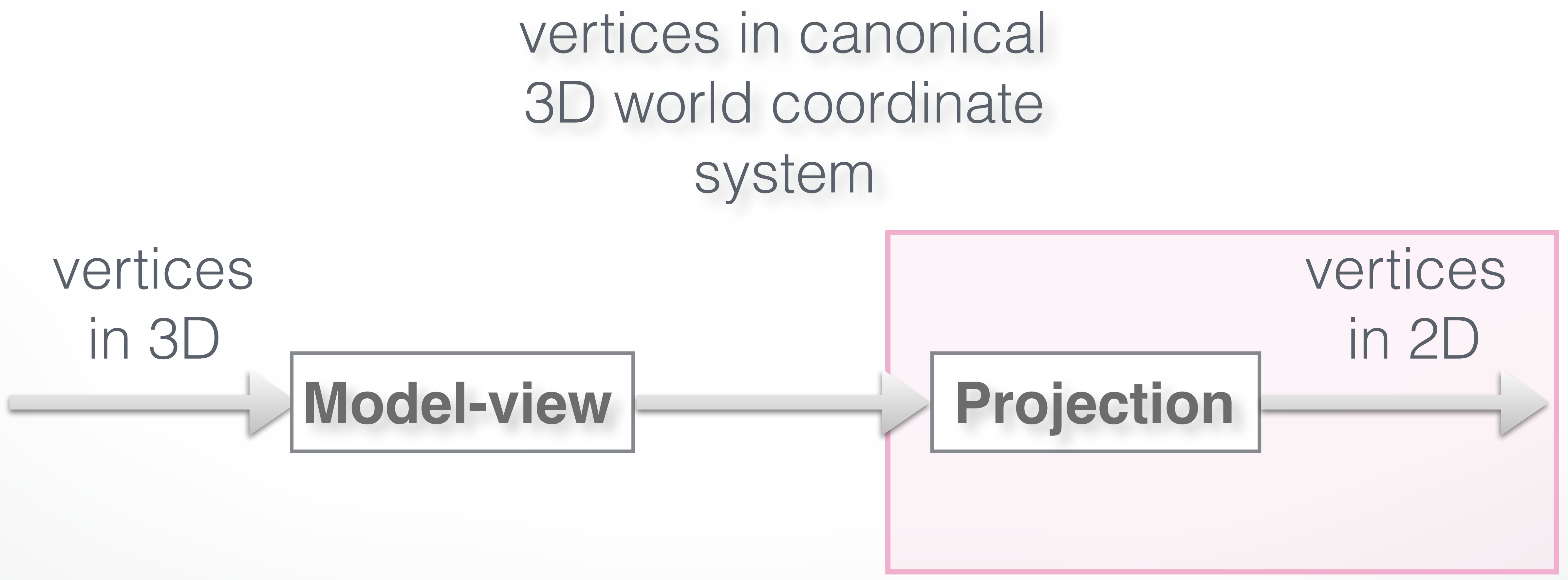
4x4 Model-view Matrix (this lecture)

- Translate, rotate, scale objects
- Position the camera

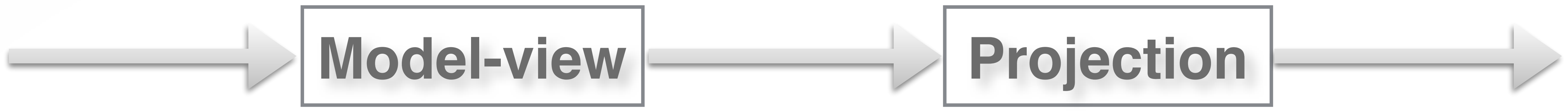


4x4 Model-view Matrix (next lecture)

- Projection from 3D to 2D



OpenGL Transformation Matrices



- Manipulated separately in OpenGL

  (must set matrix mode) :

```
glMatrixMode (GL_MODELVIEW);
glMatrixMode (GL_PROJECTION;
```

Setting the Current Model-view Matrix

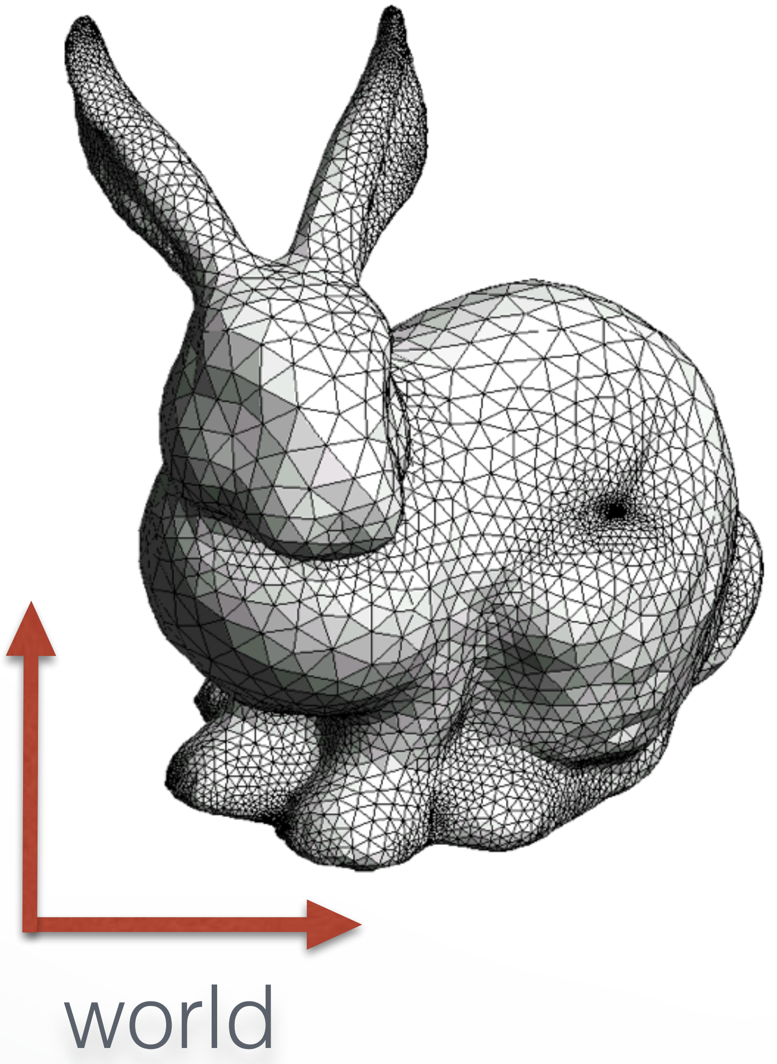
- Load or post-multiply

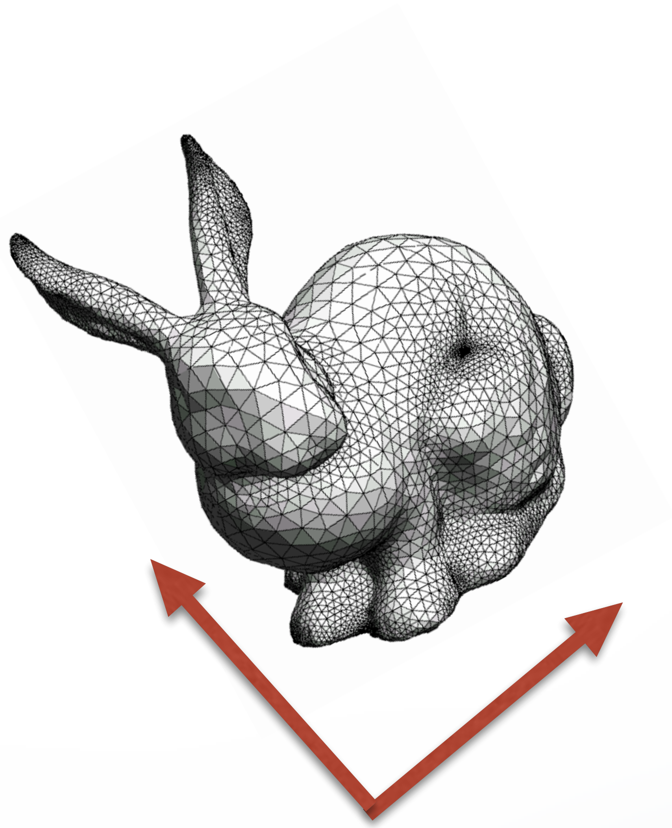
```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity(); // very common usage
float m[16] = { … };
glLoadMatrixf(m); // rare, advanced
glMultMatrixf(m); // rare, advanced
```

- Use library functions

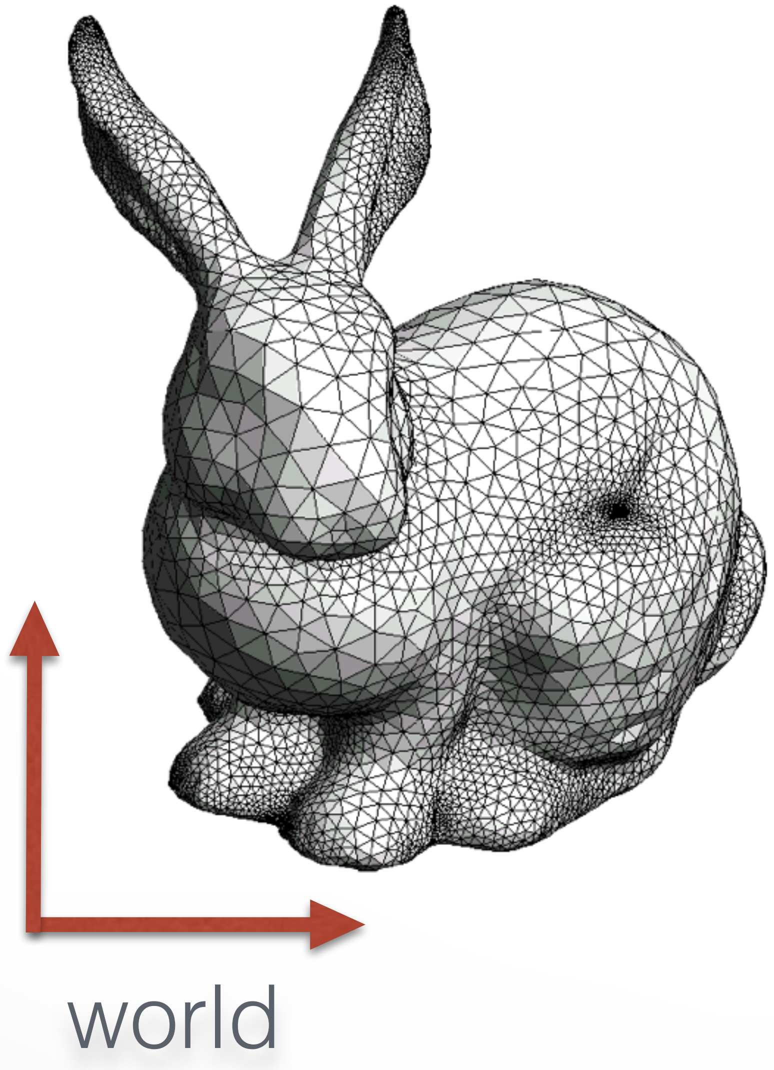
```
glTranslatef(dx, dy, dz);
glRotatef(angle, vx, vy, vz);
glScalef(sx, sy, sz);
```

Translated, rotated, scaled object





The *rendering* coordinate system

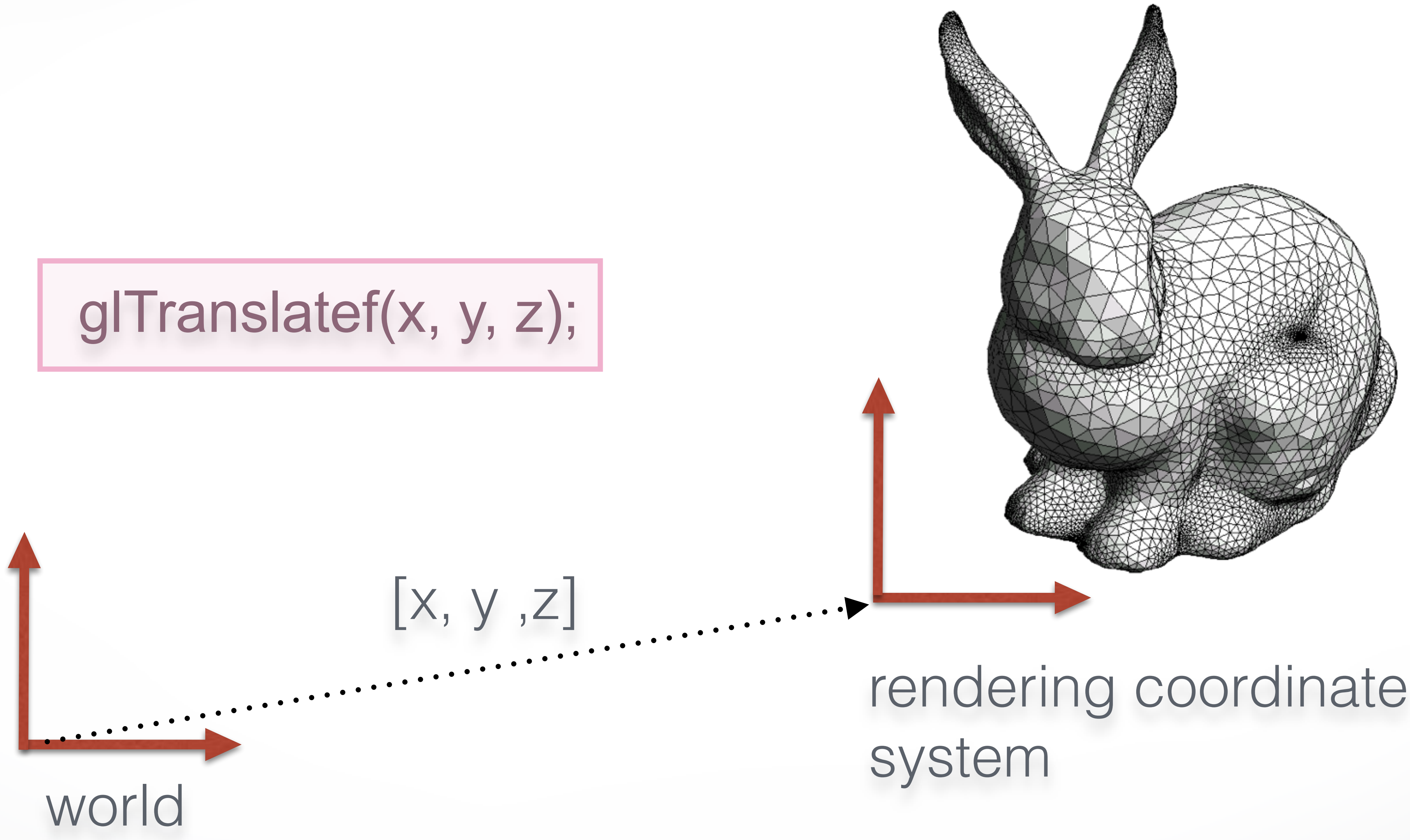


Initially (after glLoadIdentity()) :

rendering coordinate system = world coordinate system

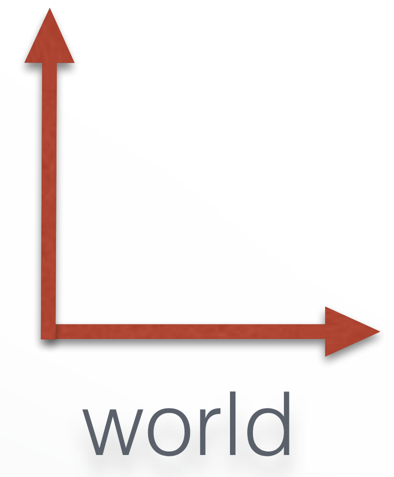
The *rendering* coordinate system

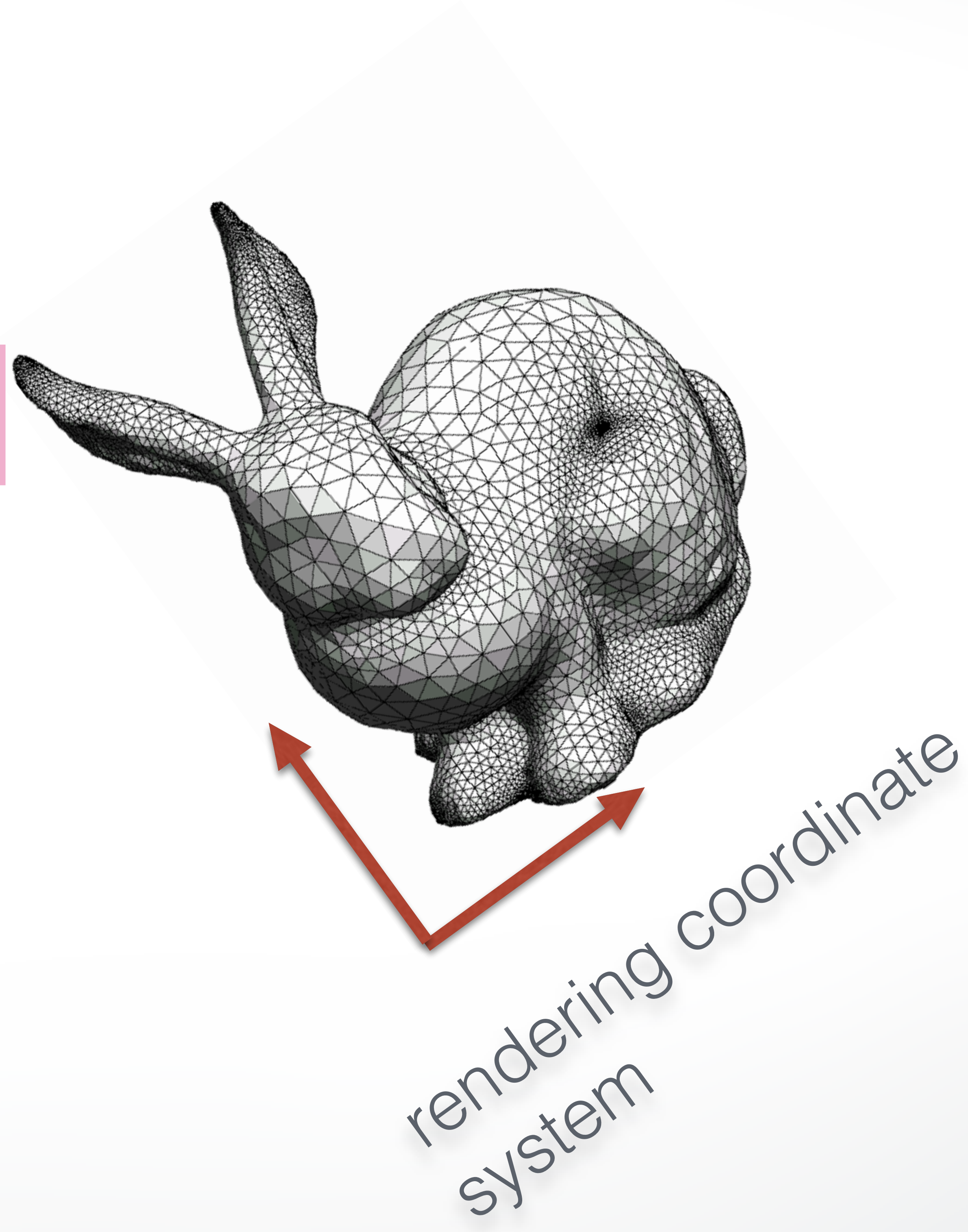
```
glTranslatef(x, y, z);
```



The *rendering* coordinate system

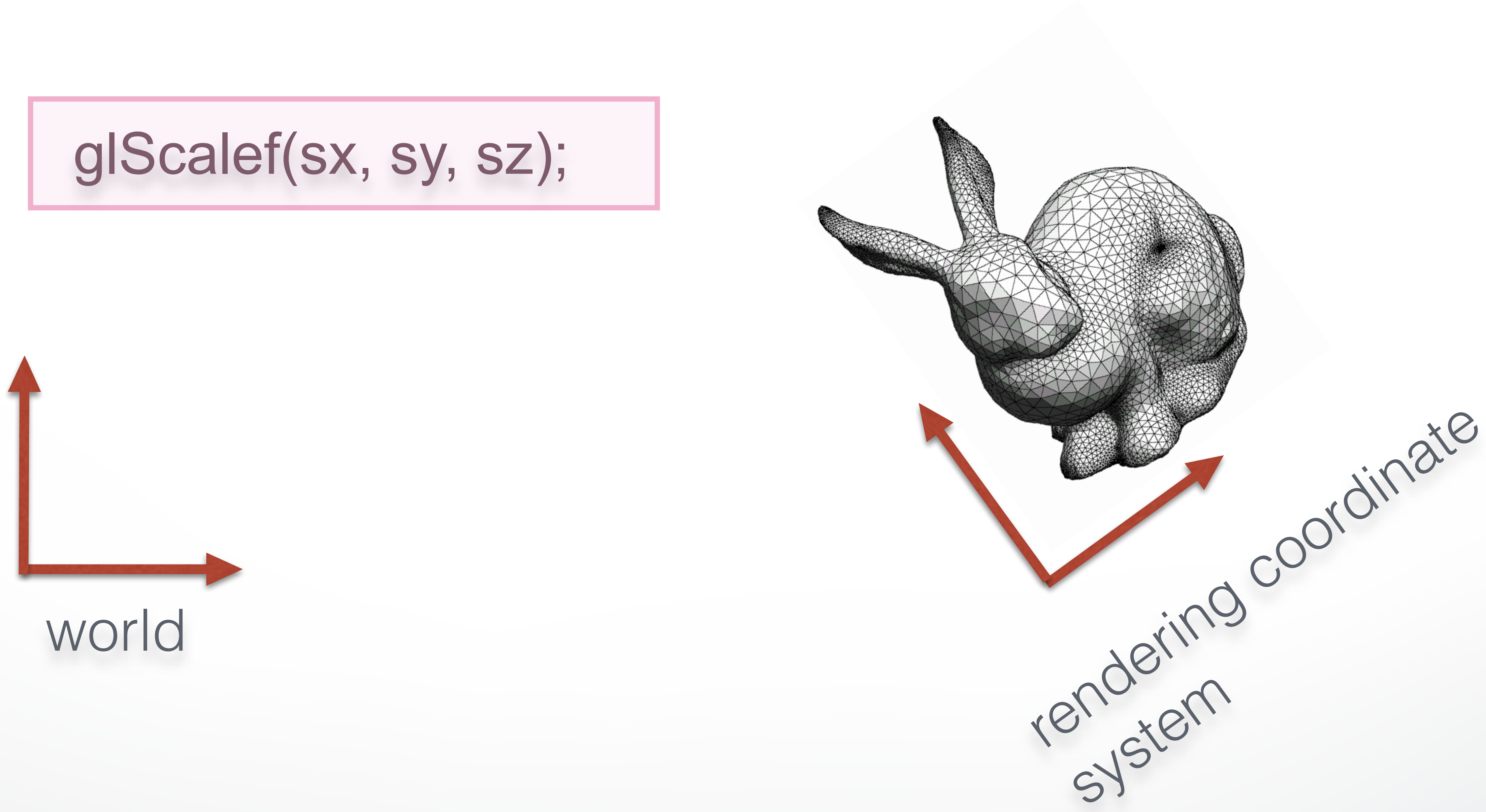
```
glRotatef(angle,ax, ay, az);
```





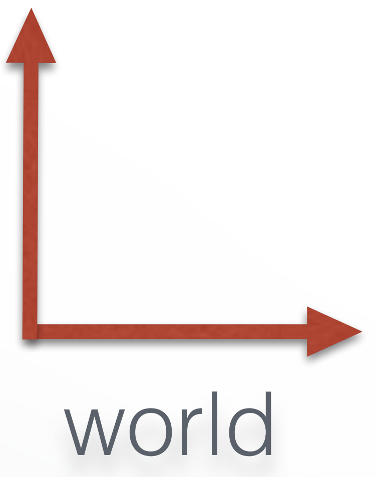
The *rendering* coordinate system

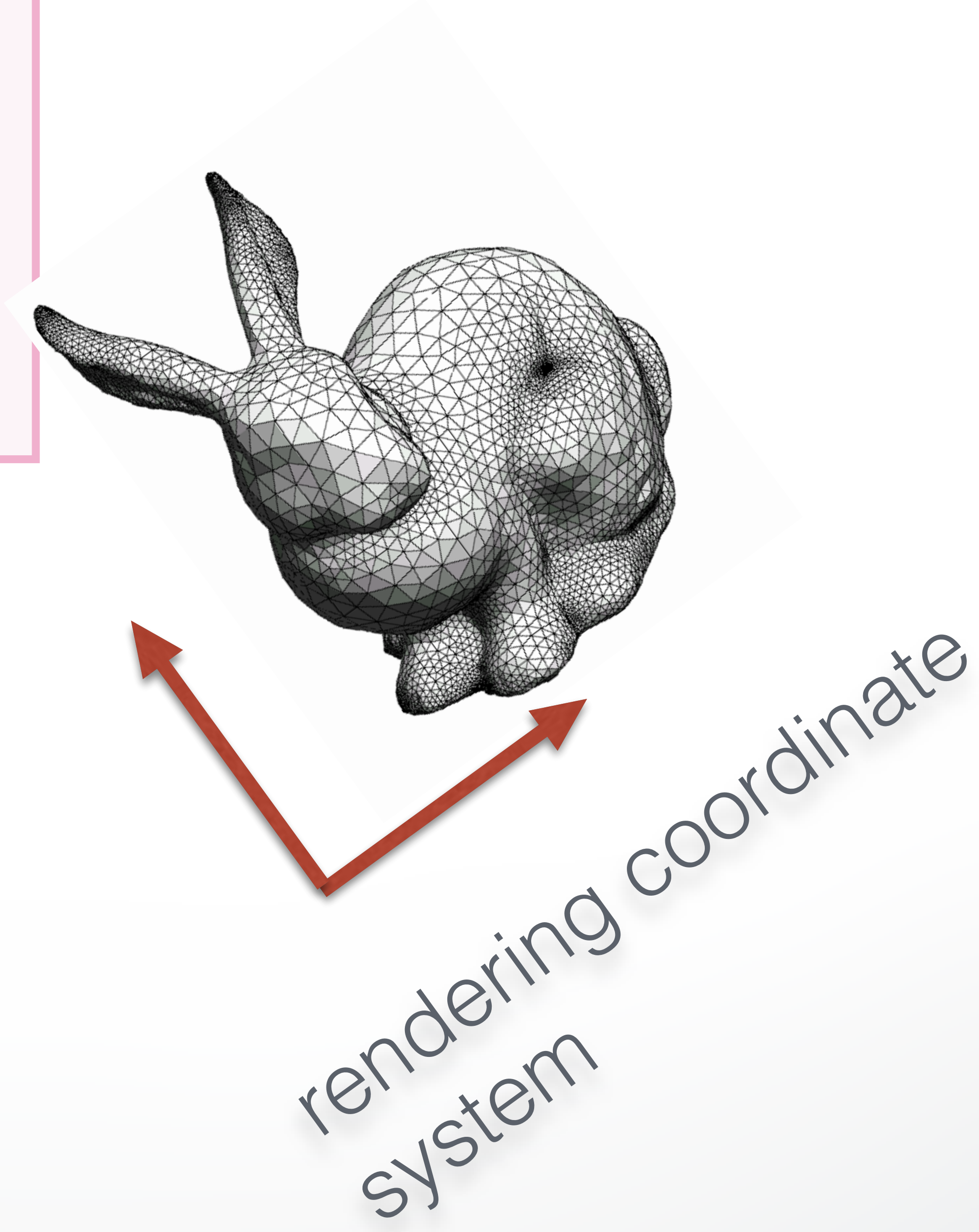
```
glScalef(sx, sy, sz);
```



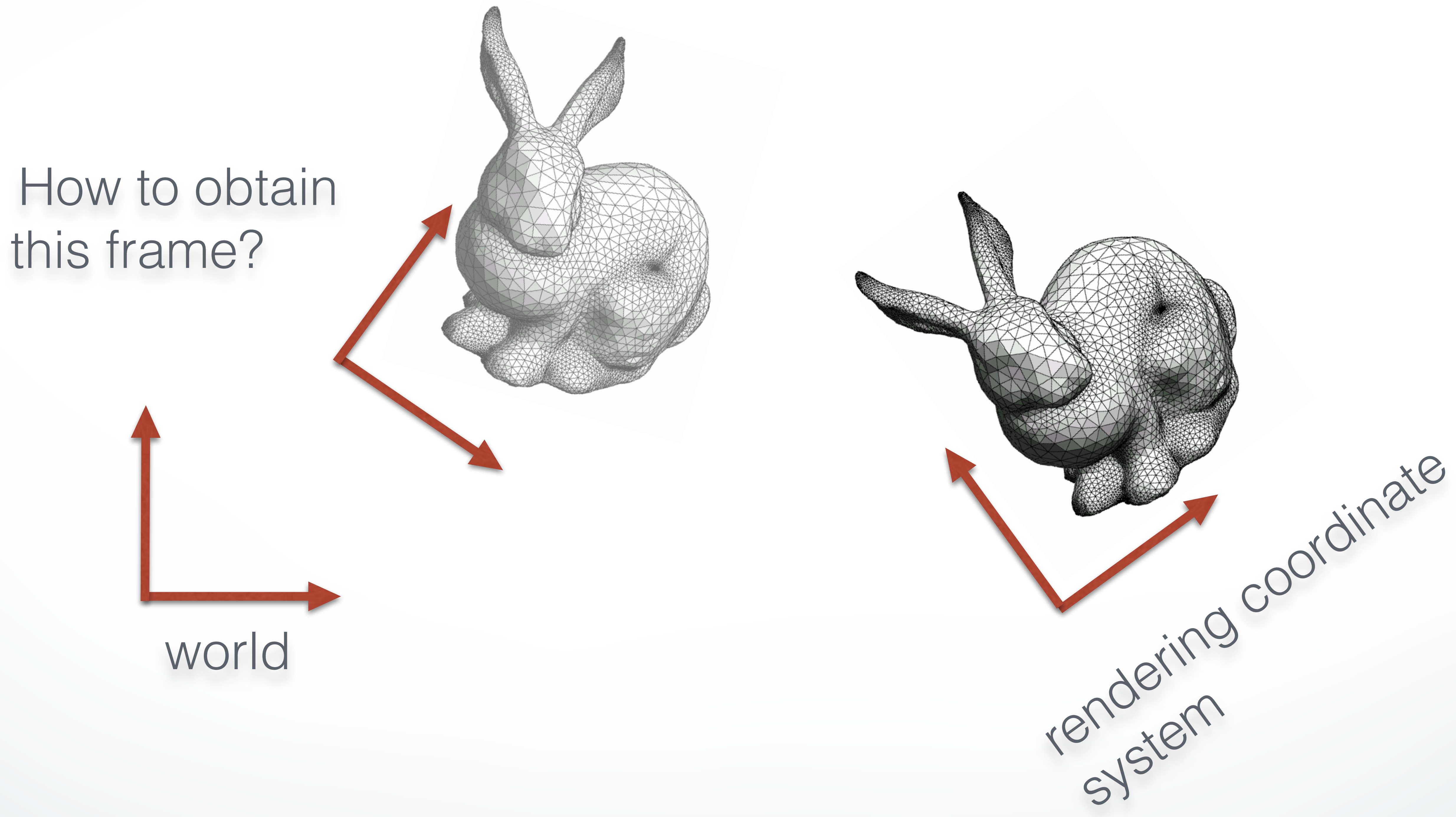
OpenGL code

```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef(x, y, z);
glRotatef(angle, ax, ay, az);
glScalef(sx, sy, sz);
renderBunny();
```





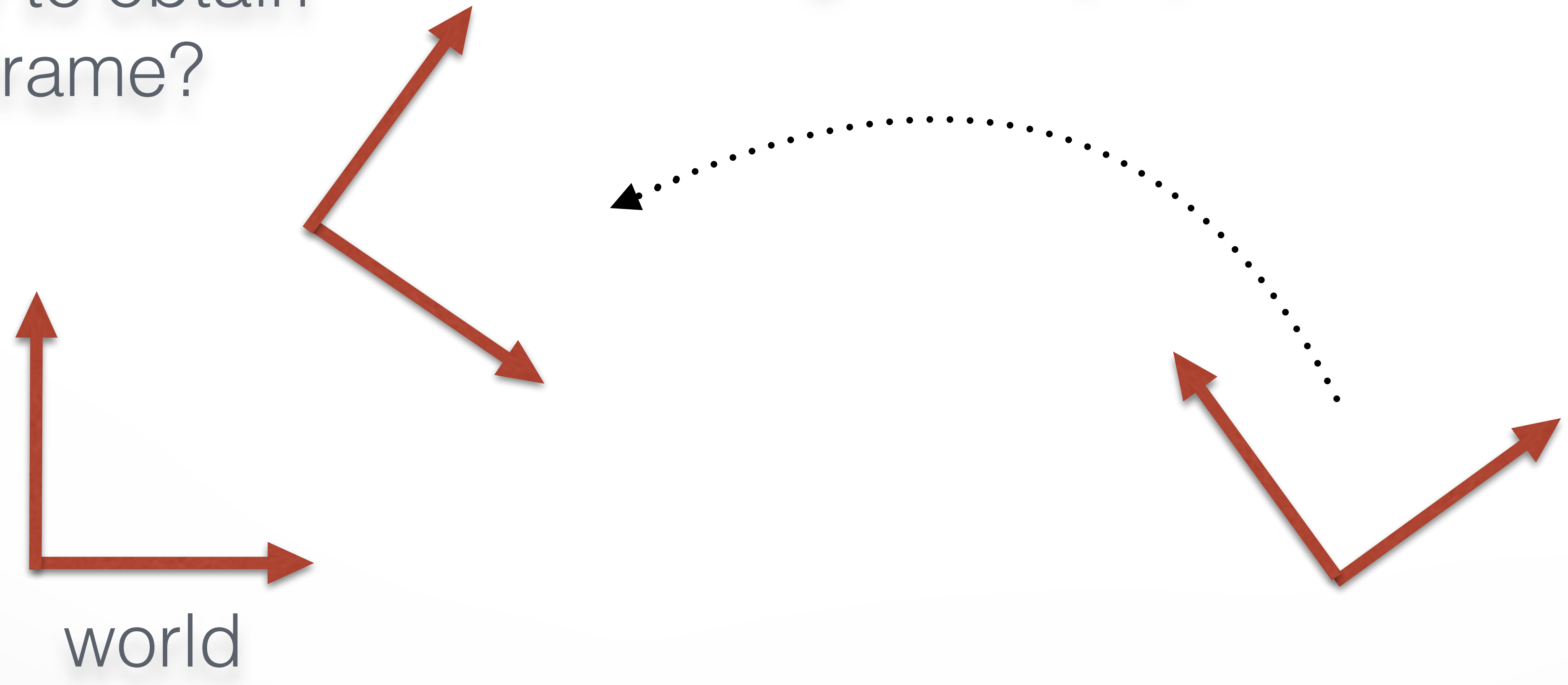
Rendering more objects



Solution 1

Find glTranslate(…), glRotatef(…), glScalef(…)

How to obtain this frame?



Solution 2: gl{Push,Pop}Matrix

```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();

// render first bunny
glPushMatrix(); // store current matrix
glTranslate3f(…);
glRotatef(…);
renderBunny();
glPopMatrix(); // pop matrix

// render second bunny
glPushMatrix(); // store current matrix
glTranslate3f(…);
glRotatef(…);
renderBunny();
glPopMatrix(); // pop matrix
```

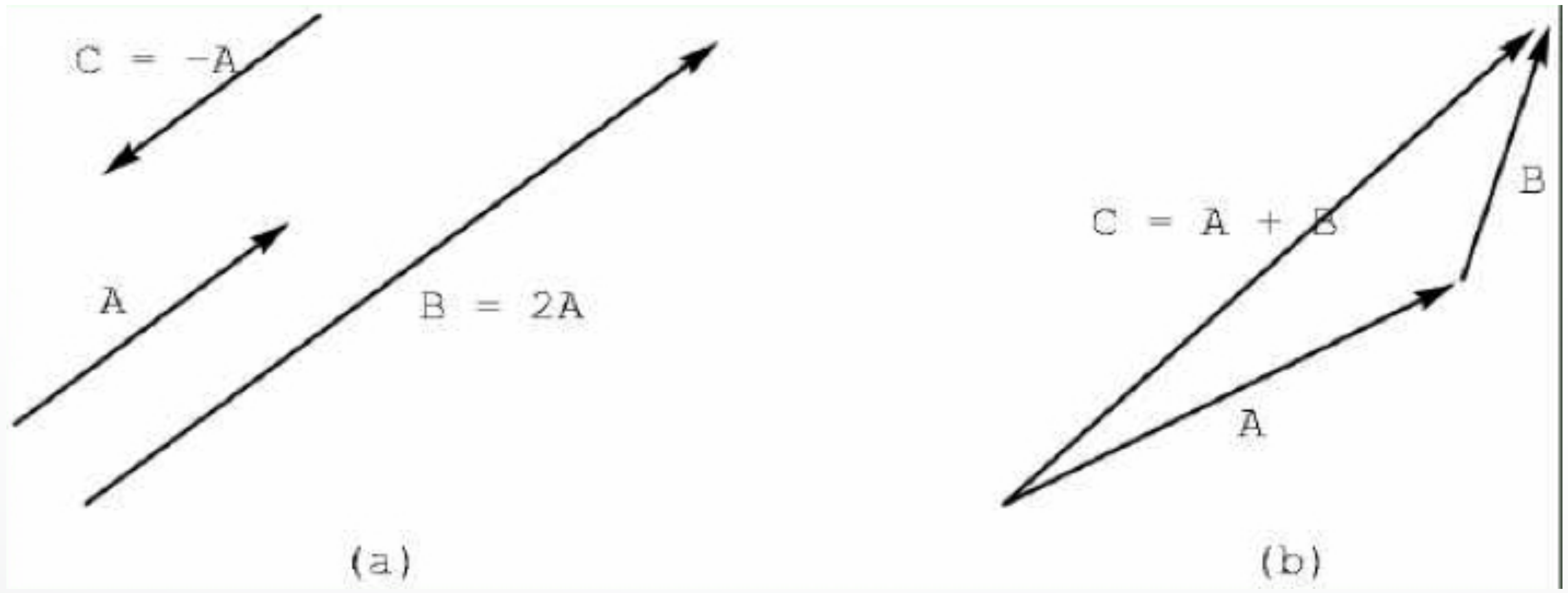
Recall: Linear Algebra

Scalars

- Scalars α , β , γ from a *scalar field*
- Operations $\alpha + \beta$, $\alpha\beta$, 0 , 1 , $-\alpha$, $()^{-1}$
- “Expected” laws apply
- Examples: rationals or reals with addition and multiplication

Vectors

- Vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ from a *vector space*
- Vector addition $\mathbf{u} + \mathbf{v}$, subtraction $\mathbf{u} - \mathbf{v}$
- Zero vector $\mathbf{0}$
- Scalar multiplication $\alpha\mathbf{v}$

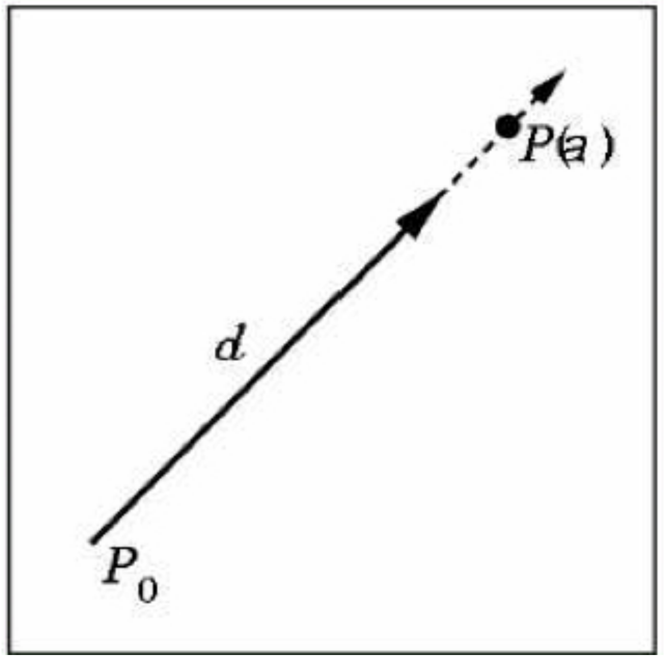


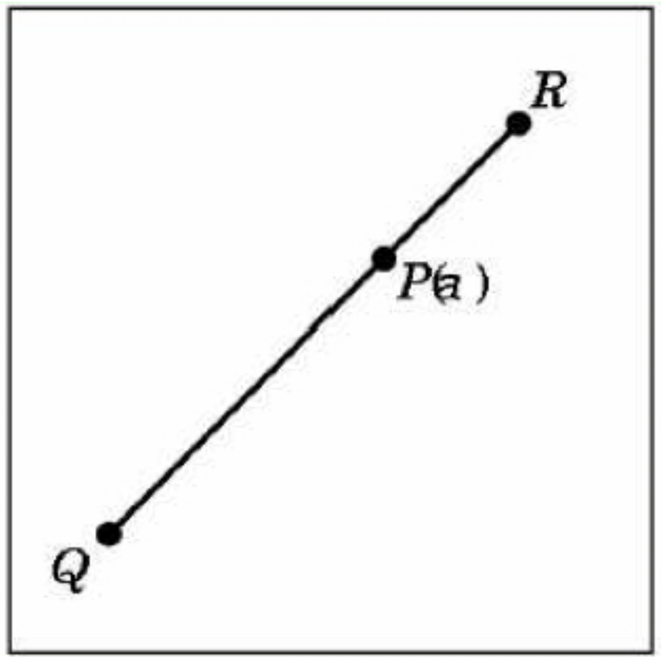
Euclidean Space

- Vector space over real numbers
- Three-dimensional in computer graphics
- Dot product: $\alpha = \mathbf{u}^\top \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3$
- $\mathbf{0}^\top \mathbf{0} = 0$,
- $\mathbf{u}, \mathbf{v}$ are orthogonal if $\mathbf{u}^\top \mathbf{v} = 0$
- $||\mathbf{v}||^2 = \mathbf{v}^\top \mathbf{v}$ defines $||\mathbf{v}||$, the *length* of $\mathbf{v}$

Lines and Line Segments

- Parametric form of line: $\mathbf{p}(\alpha) = \mathbf{p}_0 + \alpha\mathbf{d}$





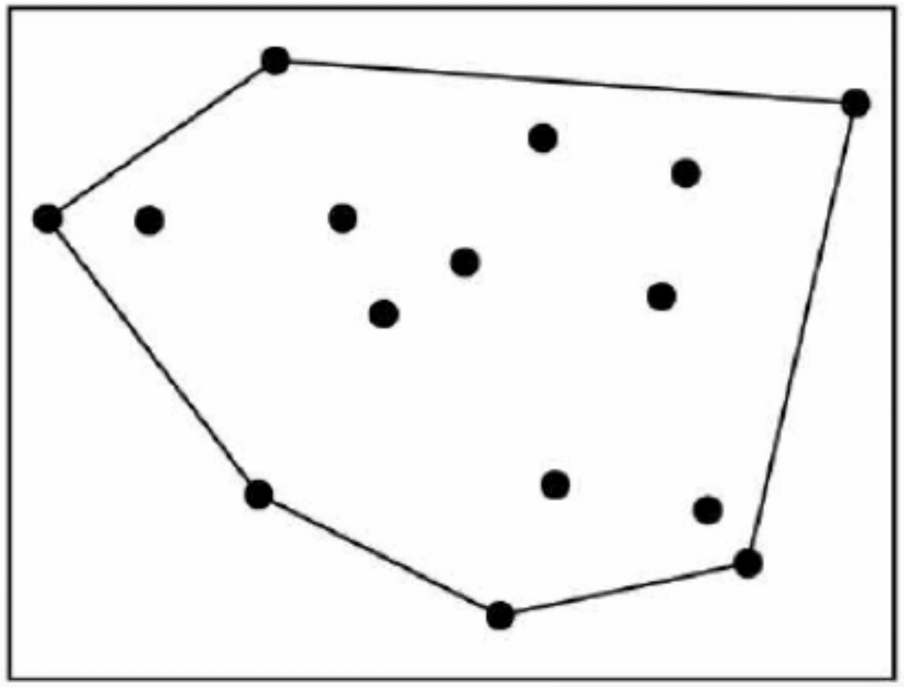
- Line segment between $\mathbf{q}$ and $\mathbf{r}$:

$\mathbf{p}(\alpha) = (1-\alpha)\mathbf{q} + \alpha\mathbf{r}$ for $0 \le \alpha \le 1$

Convex Hull

- Convex hull defined by

  $\mathbf{p} = \alpha_1 \mathbf{p}_1 + \ldots + \alpha_n \mathbf{p}_n$
  for $\alpha_1 + \ldots + \alpha_n = 1$
  and $0 \leq \alpha_1 \leq 1, i = 1 \ldots n$

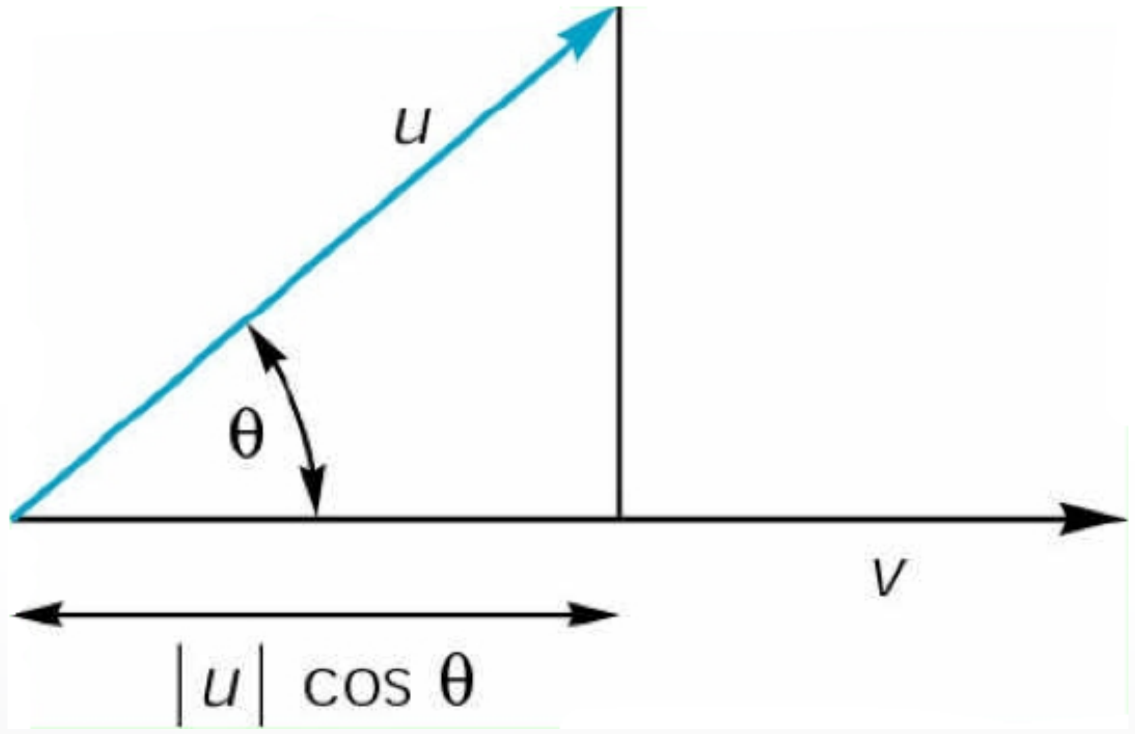


Projection

- Dot product projects one vector onto another vector

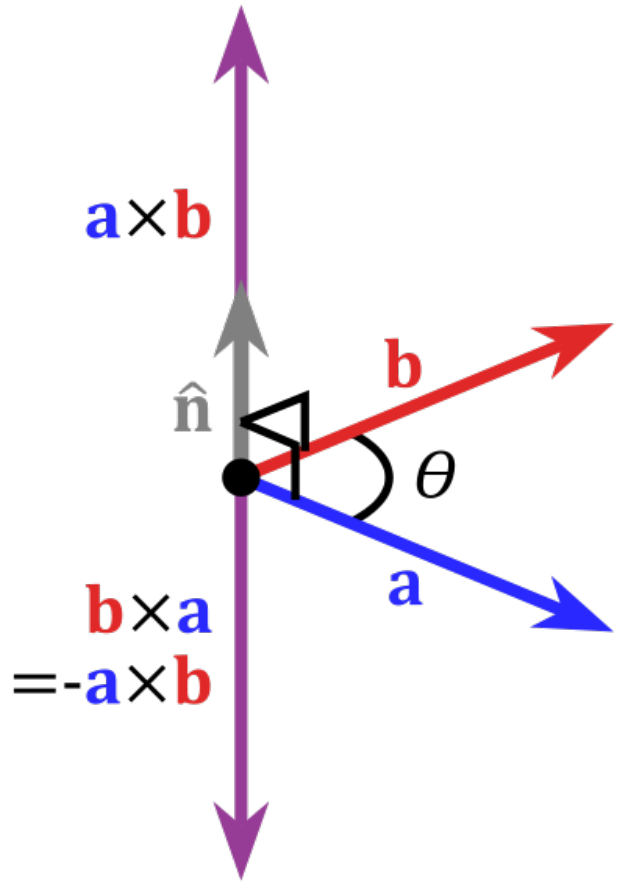
$$\mathbf{u}^\top \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3 = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\theta)$$

$$\pi_{\mathbf{v}}(\mathbf{u}) = (\mathbf{u}^\top \mathbf{v})\mathbf{v} / \|\mathbf{v}\|^2$$



Cross Product

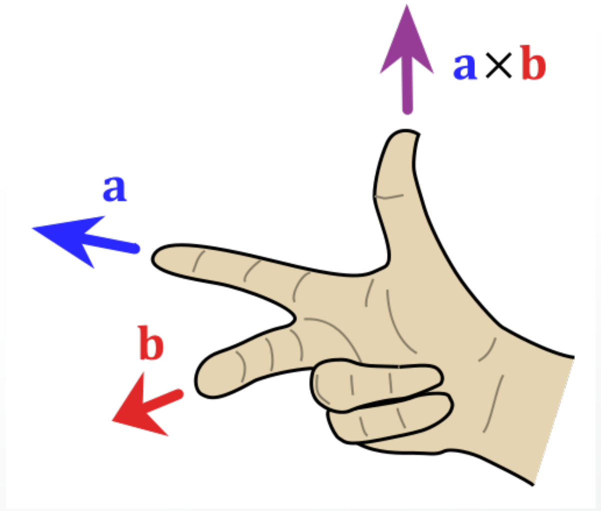
$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$



$$\|\mathbf{a} \times \mathbf{b}\| = |a||b|| \sin(\theta)|$$

- Cross product is perpendicular to both $\mathbf{a}$ and $\mathbf{b}$

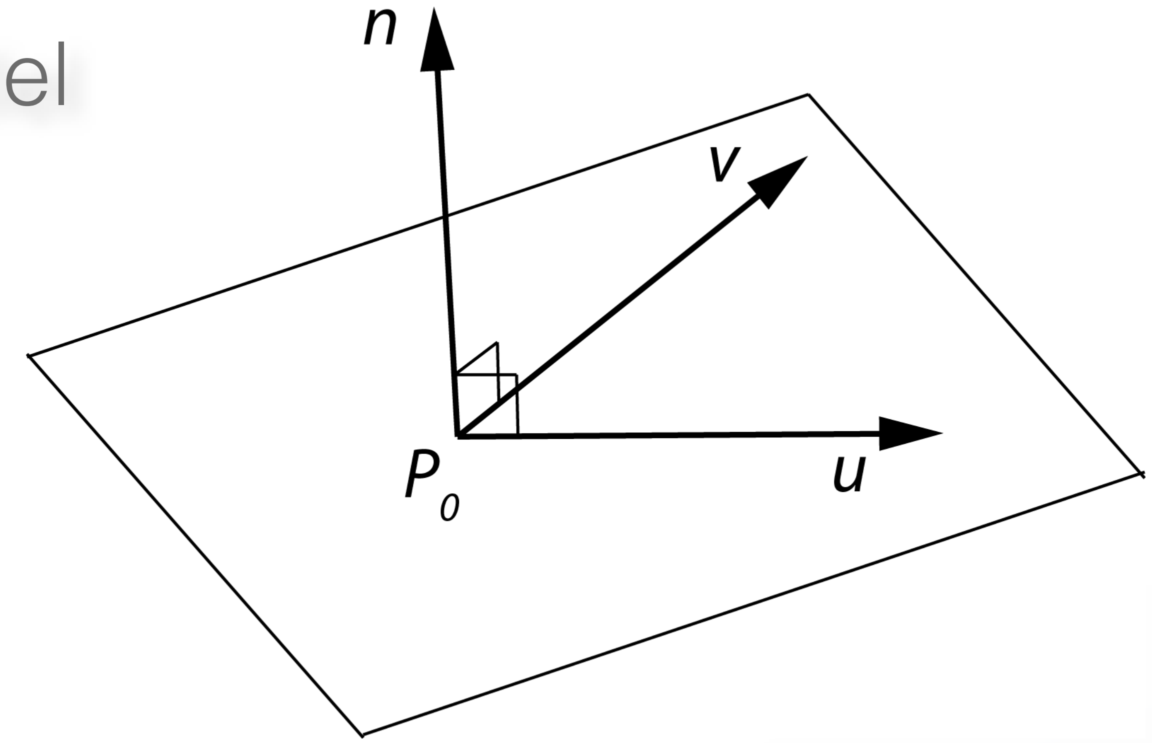
- Right-hand rule



Plane

- Plane defined by point $\mathbf{p}_0$ and vectors $\mathbf{u}$ and $\mathbf{v}$
- $\mathbf{u}$ and $\mathbf{v}$ should not be parallel
- Parametric form:

$$\mathbf{t}(\alpha, \beta) = \mathbf{p}_0 + \alpha\mathbf{u} + \beta\mathbf{v}$$

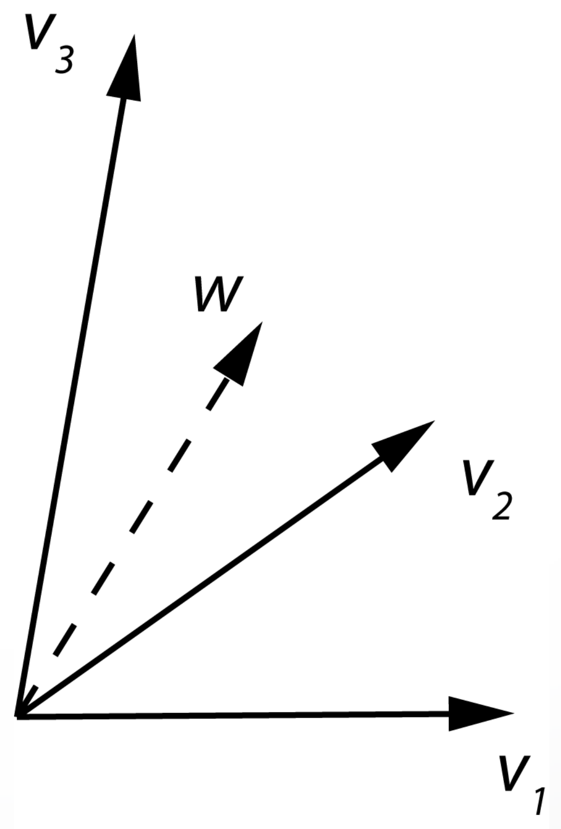


- $\mathbf{n} = \mathbf{u} \times \mathbf{v} / \|\mathbf{u} \times \mathbf{v}\|$ is the normal
- $\mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0) = 0$ if and only if $\mathbf{p}$ lies in plane

Coordinate Systems

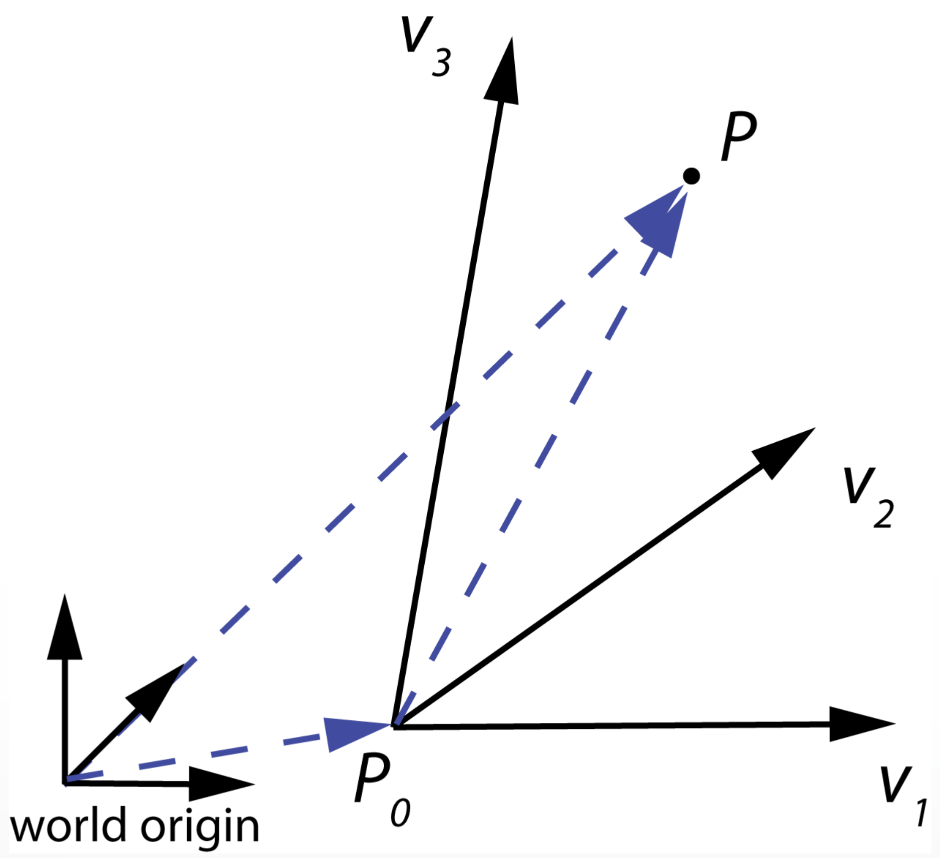
- Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ be three linearly independent vectors in a 3-dimensional vector space

- Can write *any* vector $\mathbf{w}$ as
  $$\mathbf{w} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3$$
  for some scalars $\alpha_1, \alpha_2, \alpha_3$

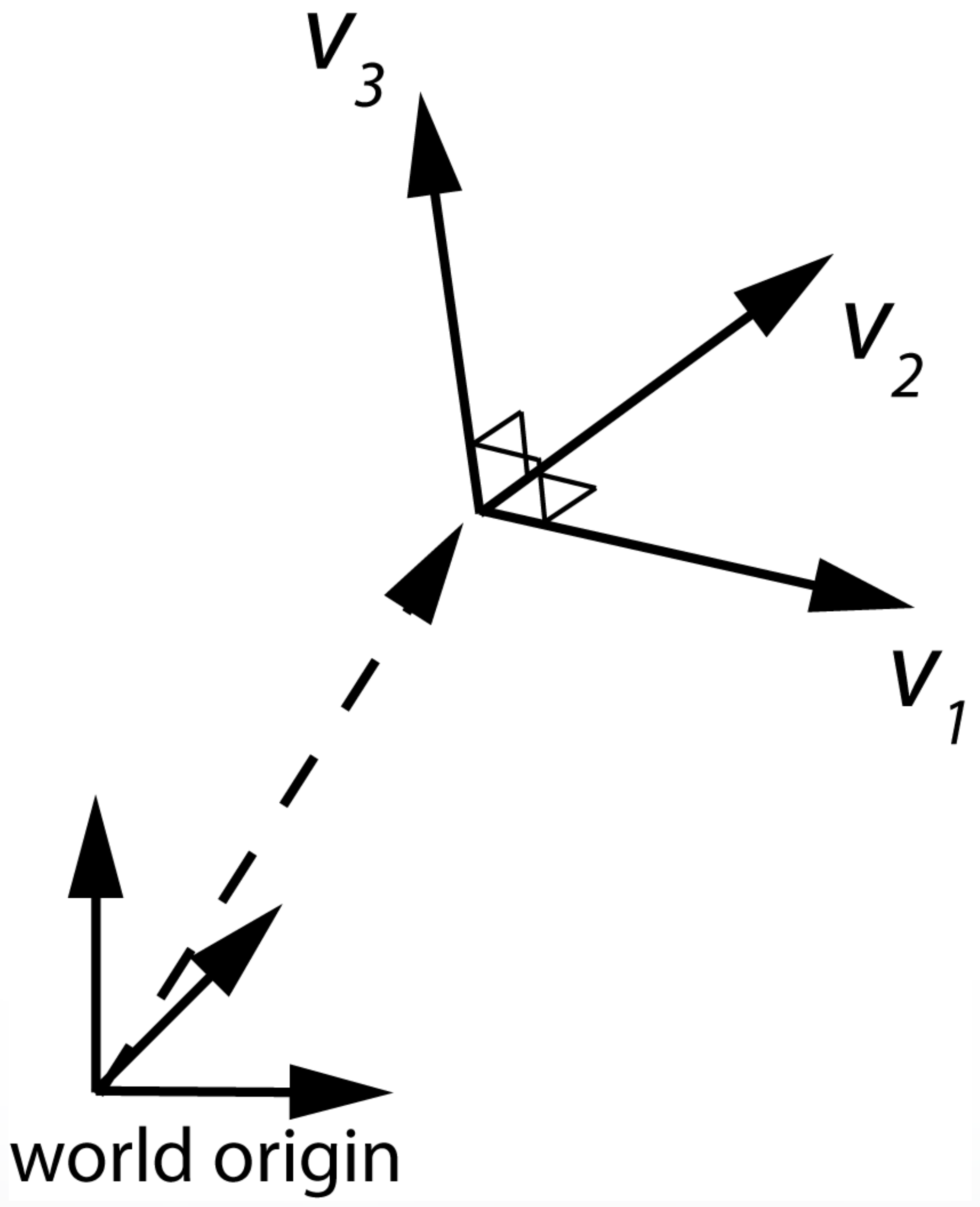


Frames

- Frame = origin $\mathbf{p}_0$ + coordinate system
- Any point $\mathbf{p} = \mathbf{p}_0 + \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$



In Practice, Frames are Often Orthogonal



Change of Coordinate System

- Bases $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ and $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$

- Express basis vectors $\mathbf{u}_i$ in terms of $\mathbf{v}_j$

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$

$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$

$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$

- Represent in matrix form:

$$\begin{bmatrix} \mathbf{u}_1^\top \\ \mathbf{u}_2^\top \\ \mathbf{u}_3^\top \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \mathbf{v}_3^\top \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

Representing
3D transformations
(and model-view matrices)

Linear Transformations

- 3 x 3 matrices represent linear transformations

$$\mathbf{a} = \mathbf{M}\mathbf{b}$$

- Can represent rotation, scaling, and reflection
- Cannot represent translation

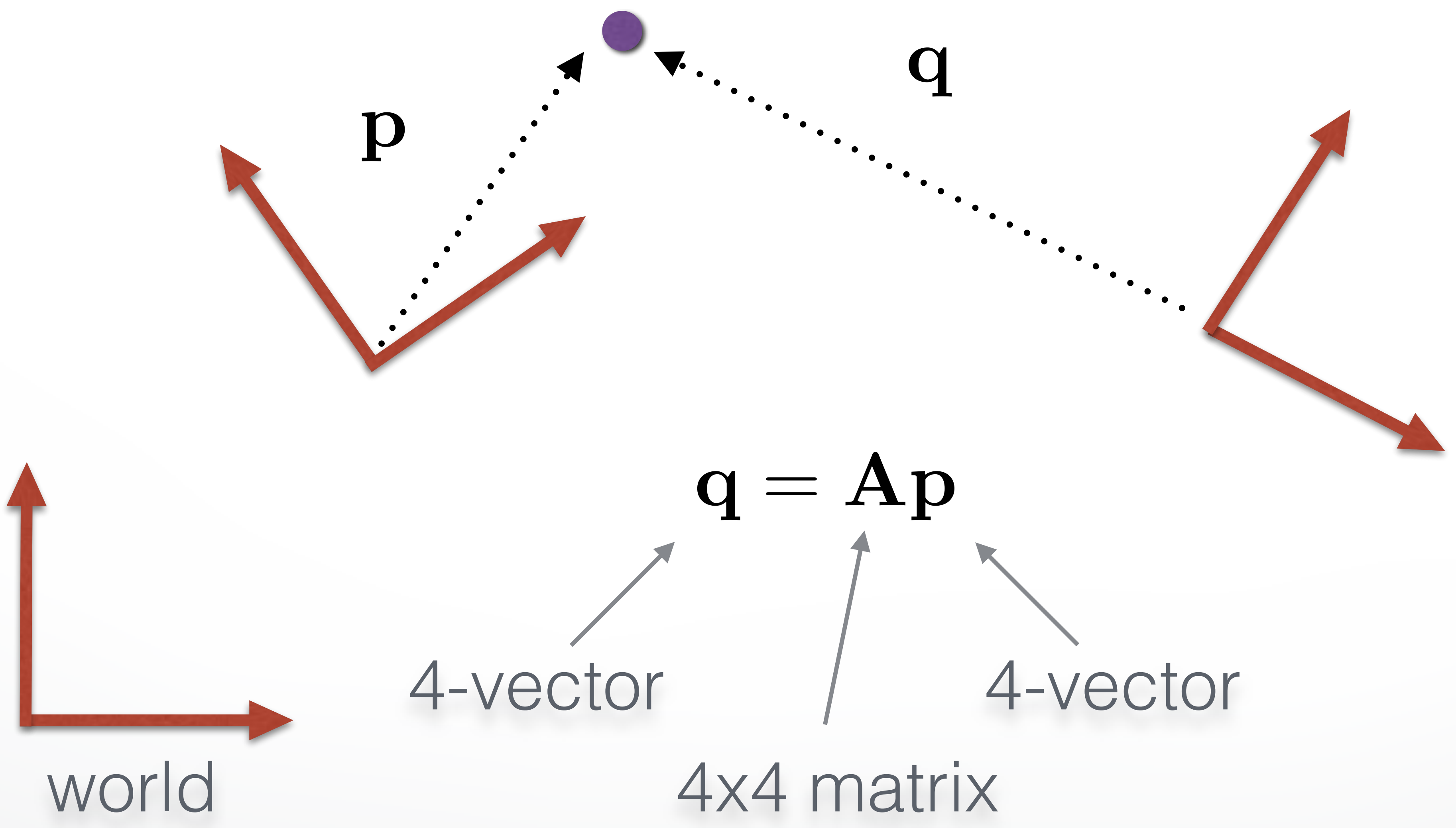
$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

Homogeneous Coordinates

- In order to represent rotations, scales AND translations
- Augment $[\alpha_1, \alpha_2, \alpha_3]^\top$ by adding a fourth component (1):
  $\mathbf{p} = [\alpha_1, \alpha_2, \alpha_3, 1]^\top$
- Homogeneous property:
  $\mathbf{p} = [\alpha_1, \alpha_2, \alpha_3, 1]^\top = [\alpha_1, \alpha_2, \alpha_3]^\top$ for any scalar $\neq 0$

Homogeneous Coordinates

- Homogeneous coordinates are transformed by 4x4 matrices



$$\mathbf{q} = \mathbf{A}\mathbf{p}$$

4-vector

4x4 matrix

4-vector

Affine Transformations (4x4 matrices)

- Translation
- Rotation
- Scaling
- Any composition of the above
- Later: projective (perspective) transformations

  -Also expressible as 4 x 4 matrices!

Translation

- $\mathbf{q} = \mathbf{p} + \mathbf{d}$ where $\mathbf{d} = [\alpha_x, \alpha_y, \alpha_z, 0]^\top$,
- $\mathbf{p} = [x, y, z, 1]^\top$,
- $\mathbf{q} = [x', y', z', 1]^\top$,

- Express in matrix form $\mathbf{q} = \mathbf{T}\mathbf{p}$ and solve for $\mathbf{T}$

$$T = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$x' = \beta_x x$$

$$y' = \beta_y y$$

$$z' = \beta_z z$$

- Express as $\mathbf{q} = \mathbf{S}\mathbf{p}$ and solve for $\mathbf{S}$

$$S = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation in 2 Dimensions

- Rotation by θ about the origin

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

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

- Express in matrix form:

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

- Note that the determinant is 1

Rotation in 3 Dimensions

- Orthogonal matrices:

$$\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}$$

$$\det(\mathbf{R}) = 1$$

- Affine transformation:

$$A = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Affine Matrices are Composed by Matrix Multiplication

$$\mathbf{A} = \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3$$

- Applied from right to left

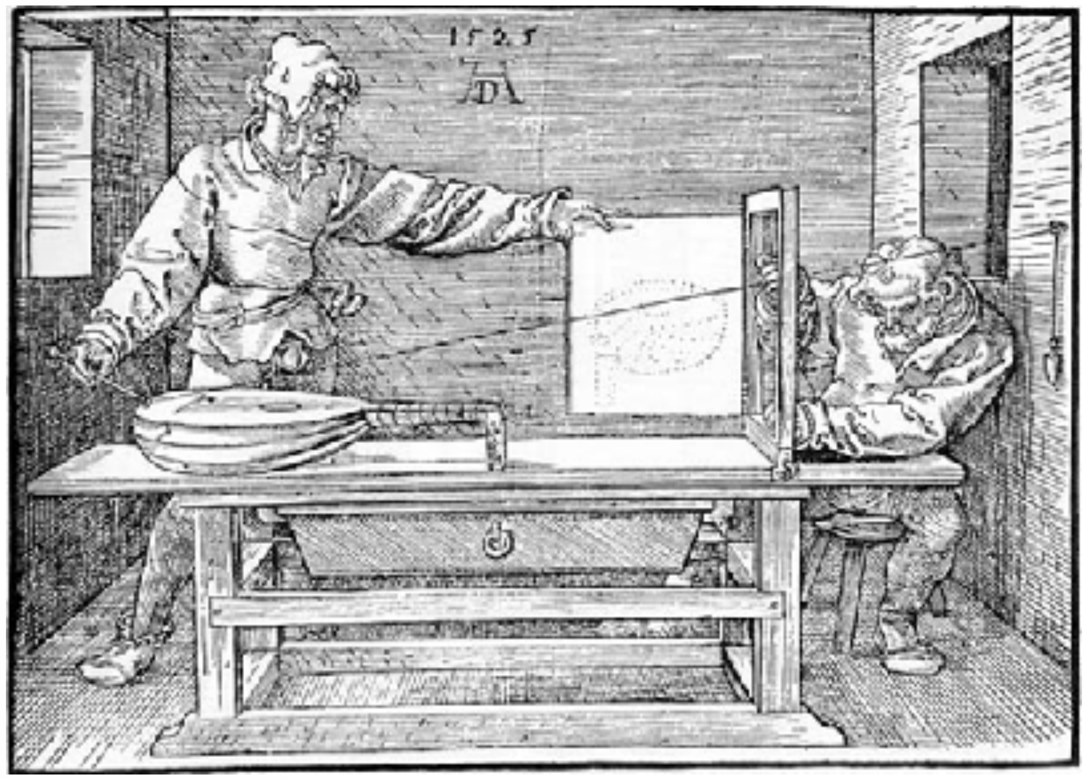
$$\mathbf{A}\mathbf{p} = (\mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3)\mathbf{p} = \mathbf{A}_1(\mathbf{A}_2(\mathbf{A}_3 \mathbf{p}))$$

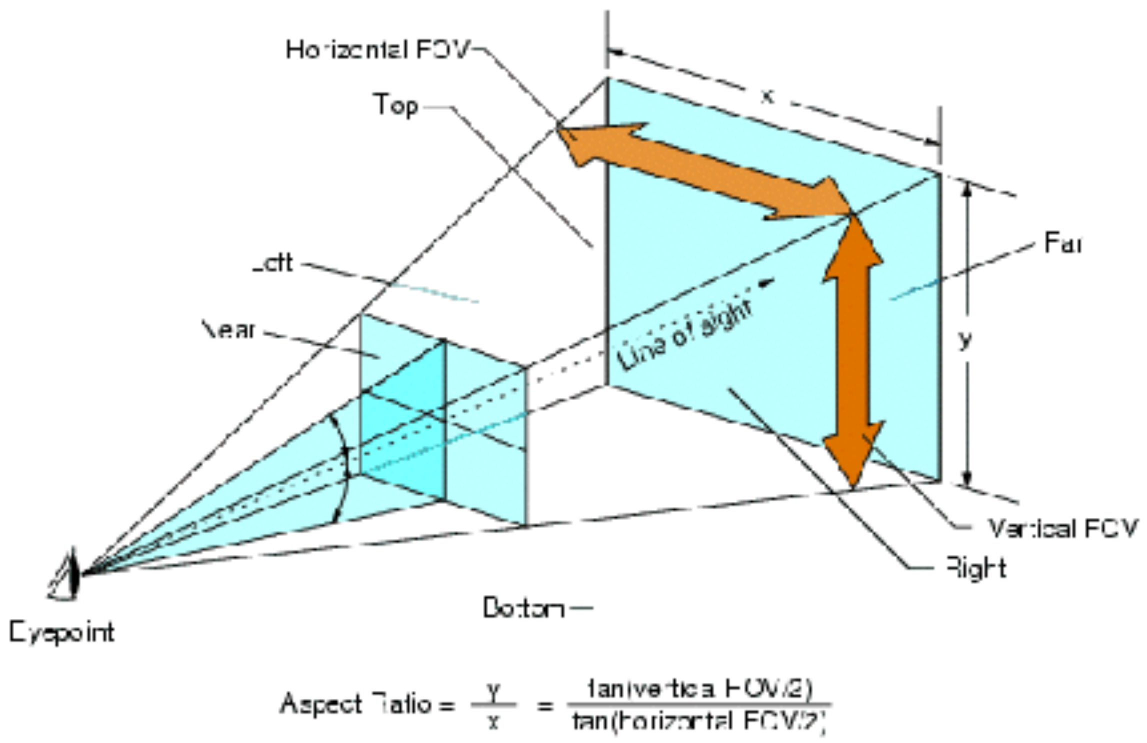
- When calling glTranslate3f, glRotatef, or glScalef, OpenGL forms the corresponding 4x4 matrix, and multiplies the current modelview matrix with it.

Summary

- OpenGL Transformation Matrices
- Vector Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices

Next Time: Viewing & Projection





http://cs420.hao-li.com

Thanks!