

## Transformations

Vector Spaces  
Euclidean Spaces  
Frames  
Homogeneous Coordinates  
Transformation Matrices

January 24, 2011

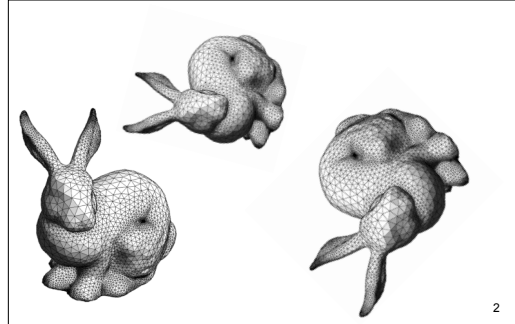
Jernej Barbic  
University of Southern California

[Angel, Ch. 4]

<http://www.bcf.usc.edu/~jbarbic/cs480-s11/>

1

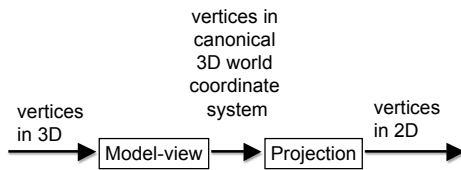
## OpenGL Transformations



2

## OpenGL Transformation Matrices

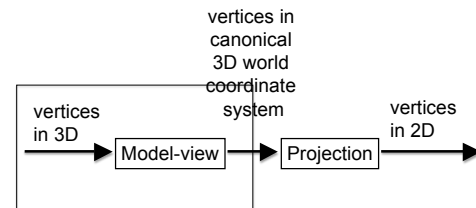
- Model-view matrix (4x4 matrix)
- Projection matrix (4x4 matrix)



3

## 4x4 Model-view Matrix (this lecture)

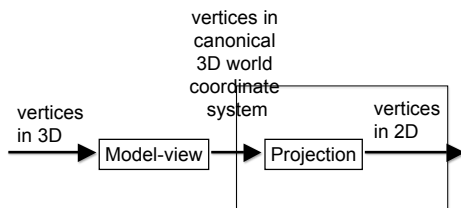
- Render translated, rotated, scaled objects
- Position the camera



4

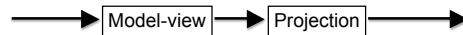
## 4x4 Projection Matrix (next lecture)

- Project from 3D to 2D



5

## OpenGL Transformation Matrices



- Manipulated separately in OpenGL (must set matrix mode):

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

6

## 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
```

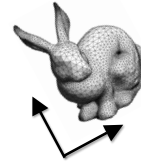
- Use library functions

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

7

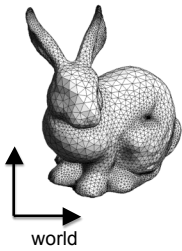
## The OpenGL rendering code

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



8

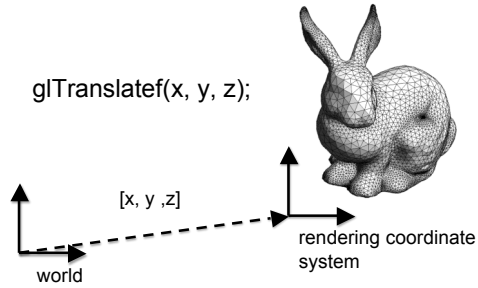
## The rendering 3D coordinate system



Initially (after `glLoadIdentity()`):  
rendering coordinate system =  
world coordinate system

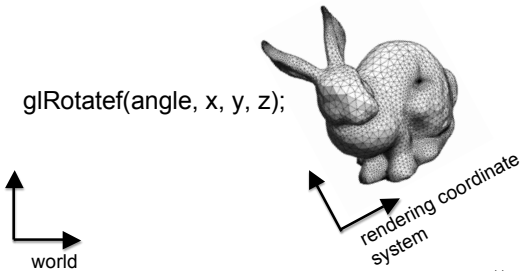
9

## The rendering 3D coordinate system



10

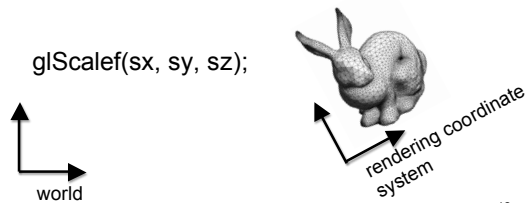
## The rendering 3D coordinate system



`glRotatef(angle, x, y, z);`

11

## The rendering 3D coordinate system

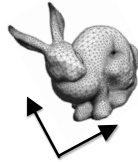


`glScalef(sx, sy, sz);`

12

## The OpenGL rendering code

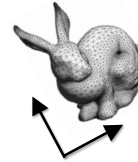
```
glMatrixMode (GL_MODELVIEW);  
glLoadIdentity();  
glTranslatef(x, y, z);  
glRotatef(angle, x, y, z);  
glScalef(sx, sy, sz);  
renderBunny();
```



13

## Rendering more objects

How to obtain  
this frame?

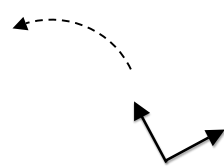


14

## Solution 1:

Find `glTranslatef(...)`, `glRotatef(...)`,  
`glScalef(...)`

How to obtain  
this frame?



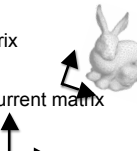
15

## Solution 2: `gl{Push,Pop}Matrix`

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

```
// render first bunny  
glPushMatrix(); // store current matrix  
glTranslatef3f(...);  
glRotatef(...);  
renderBunny();  
glPopMatrix(); // pop matrix
```

```
// render second bunny  
glPushMatrix(); // store current matrix  
glTranslatef3f(...);  
glRotatef(...);  
renderBunny();  
glPopMatrix(); // pop matrix
```



world

16

## 3D Math Review

17

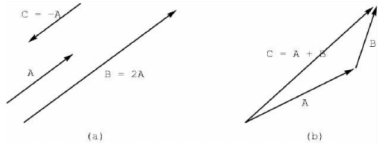
## Scalars

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

18

## Vectors

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



19

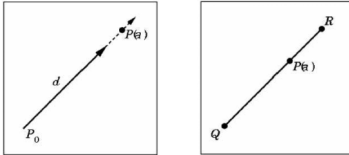
## Euclidean Space

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

20

## Lines and Line Segments

- Parametric form of line:  $P(\alpha) = P_0 + \alpha d$



- Line segment between  $Q$  and  $R$ :  
 $P(\alpha) = (1-\alpha)Q + \alpha R$  for  $0 \leq \alpha \leq 1$

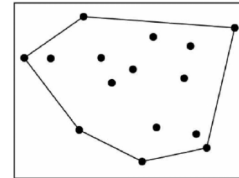
21

## Convex Hull

- Convex hull defined by

$$P = \alpha_1 P_1 + \dots + \alpha_n P_n$$

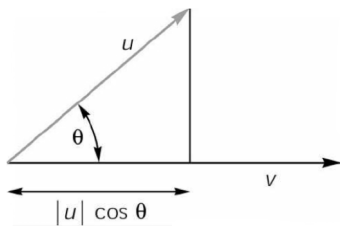
for  $\alpha_1 + \dots + \alpha_n = 1$   
and  $0 \leq \alpha_i \leq 1, i = 1, \dots, n$



22

## Projection

- Dot product projects one vector onto another vector  
 $u \cdot v = |u| |v| \cos(\theta)$

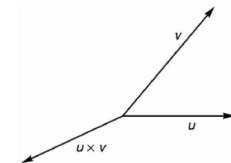


23

## Normal Vector

- Cross product defines normal vector  
 $u \times v = n$   
 $|u \times v| = |u| |v| |\sin(\theta)|$

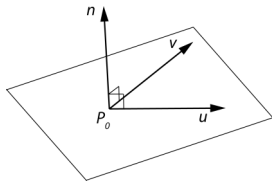
- Right-hand rule



24

## Plane

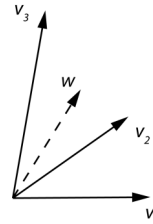
- Plane defined by point  $P_0$  and vectors  $u$  and  $v$
- $u$  and  $v$  cannot be parallel
- Parametric form:  
 $T(\alpha, \beta) = P_0 + \alpha u + \beta v$   
 ( $\alpha$  and  $\beta$  are scalars)
- $n = u \times v / |u \times v|$  is the normal
- $n \cdot (P - P_0) = 0$  if and only if  $P$  lies in plane



25

## Coordinate Systems

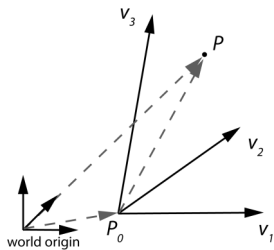
- Let  $v_1, v_2, v_3$  be three linearly independent vectors in a 3-dimensional vector space
- Can write any vector  $w$  as  
 $w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$   
 for scalars  $\alpha_1, \alpha_2, \alpha_3$



26

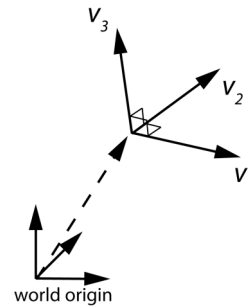
## Frames

- Frame = origin  $P_0$  + coordinate system
- Any point  $P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$



27

## In Practice Frames Often Orthogonal



28

## Change of Coordinate System

- Bases  $\{u_1, u_2, u_3\}$  and  $\{v_1, v_2, v_3\}$
- Express basis vectors  $u_i$  in terms of  $v_j$

$$u_1 = \gamma_{11}v_1 + \gamma_{12}v_2 + \gamma_{13}v_3$$

$$u_2 = \gamma_{21}v_1 + \gamma_{22}v_2 + \gamma_{23}v_3$$

$$u_3 = \gamma_{31}v_1 + \gamma_{32}v_2 + \gamma_{33}v_3$$

- Represent in matrix form:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = M \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

29

## Representing 3D transformations (and model-view matrices)

30

## Linear Transformations

- 3 x 3 matrices represent linear transformations  
**a = Mb**
- Can represent rotation, scaling, and reflection
- Cannot represent translation

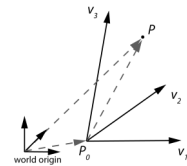
$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

31

## In order to represent rotations, scales AND translations: Homogeneous Coordinates

- $P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0$
- Then

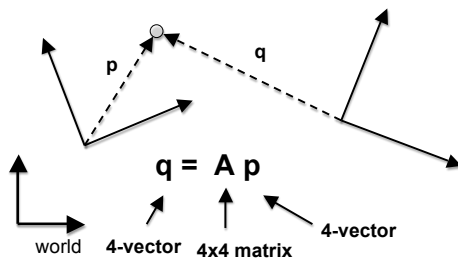
$$P = [\alpha_1 \ \alpha_2 \ \alpha_3 \ 1] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$



- Homogeneous coordinates:  
 $\mathbf{p} = [\alpha_1 \ \alpha_2 \ \alpha_3 \ 1]^T$

32

## Homogeneous coordinates are transformed by 4x4 matrices



33

## Affine Transformations (4x4 matrices)

- Translation
- Rotation
- Scaling
- Any composition of the above
- Later: projective (perspective) transformations  
- Also expressible as 4 x 4 matrices!

34

## Translation

- $\mathbf{q} = \mathbf{p} + \mathbf{d}$  where  $\mathbf{d} = [\alpha_x \ \alpha_y \ \alpha_z \ 0]^T$
- $\mathbf{p} = [x \ y \ z \ 1]^T$
- $\mathbf{q} = [x' \ y' \ z' \ 1]^T$
- Express in matrix form  $\mathbf{q} = \mathbf{T} \mathbf{p}$  and solve for  $\mathbf{T}$

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

35

## Scaling

- $x' = \beta_x x$
- $y' = \beta_y y$
- $z' = \beta_z z$
- Express as  $\mathbf{q} = \mathbf{S} \mathbf{p}$  and solve for  $\mathbf{S}$

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

36

## Rotation in 2 Dimensions

- Rotation by  $\theta$  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

37

## Rotation in 3 Dimensions

- Orthogonal matrices:

$$RR^T = R^T R = I$$
$$\det(R) = 1$$

- Affine transformation:

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

38

## Affine Matrices are Composed by Matrix Multiplication

- $A = A_1 A_2 A_3$
- Applied from right to left
- $A \mathbf{p} = (A_1 A_2 A_3) \mathbf{p} = A_1 (A_2 (A_3 \mathbf{p}))$
- This is what happens in OpenGL when calling `glTranslate3f`, `glRotatef`, ...

39

## Summary

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

40