

---

# BASIC GEOMETRY IN CG

CSE 872 FALL 2009

1

## INTRO

---

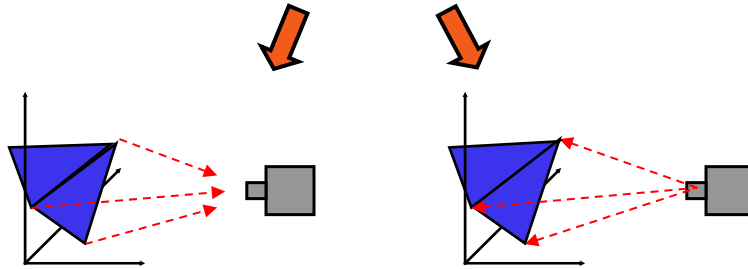
- 3D geometry in linear algebra
- Rigid body transformation
- Deformation in linear algebra
- One matrix to represent camera
- Quaternions

CSE 872 FALL 2009

2

# BASIC RENDERING MODEL

Models for objects and cameras?



[Slusallek'05]

**Rasterization:**  
Project geometry forward

**Ray Tracing:**  
Project image samples backwards

CSE 872 FALL 2009

3

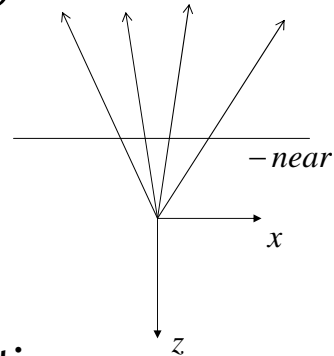
# PROJECTIVE PLANE \*

- Projective Space  $P(V)$

- Study lines
- Equivalence class
  - $(x, y, z) = (x/z, y/z, 1)$
  - $(x, y, z) = (sx, sy, sz)$

- Importance

- Camera, Transformations...



CSE 872 FALL 2009

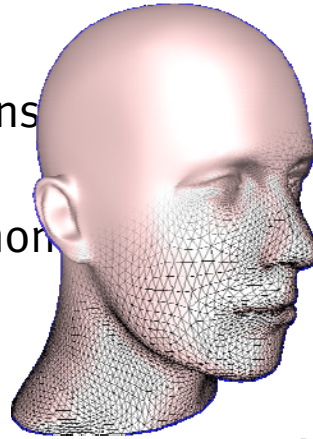
4

# REPRESENTATIONS

---

How do we represent an object?

- Points
  - $p = [x \ y \ z]$
- Mathematical Functions
  - $X^2 + Y^2 = R^2$
- Polygons (most common)
  - Points
  - Connectivity



CSE 872 FALL 2009

5

# POINTS AND VECTORS

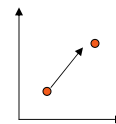
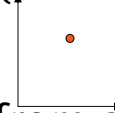
---

A 3D point  $p = [x \ y \ z]$

- Represents a location with respect to some coordinate system (affine)

A 3D vector  $v = [x \ y \ z]$

- Represents a displacement from  $\vec{a}$  position (linear)



CSE 872 FALL 2009

6

## VECTOR SPACES

---

Consists of a set of elements, called vectors and two operations that are defined on them, addition and scalar multiplication

## VECTOR ADDITION

---

Given  $V = [X \ Y \ Z]$  and  $W = [A \ B \ C]$

- $V+W = [X+A \ Y+B \ Z+C]$

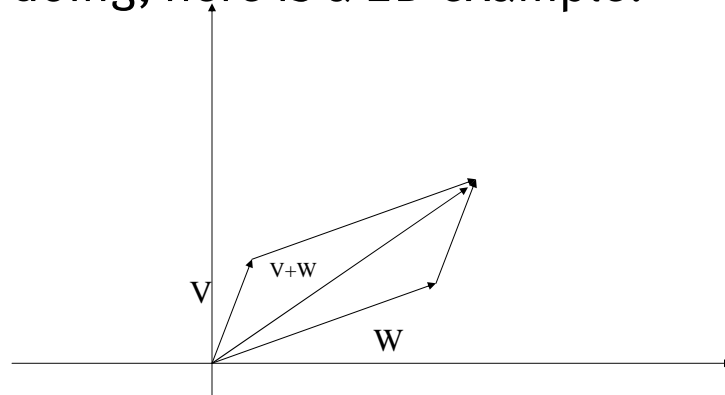
Properties of Vector addition

- Commutative:  $V+W=W+V$
- Associative  $(U+V)+W = U+(V+W)$
- Additive Identity:  $V+o = V$
- Additive Inverse:  $V+W = o, W=-V$

## PARALLELOGRAM RULE

---

To visualize what a vector addition is doing, here is a 2D example:



CSE 872 FALL 2009

9

## SCALAR PRODUCT

---

Given  $V = [X \ Y \ Z]$  and a Scalar  $s$  and  $t$

■  $sV = [sX \ sY \ sZ]$

Properties of Vector multiplication

Associative:  $(st)V = s(tV)$

Multiplicative Identity:  $1V = V$

Scalar Distribution:  $(s+t)V = sV+tV$

Vector Distribution:  $s(V+W) = sV+sW$

CSE 872 FALL 2009

10

## DOT PRODUCT AND DISTANCES

---

Given  $u = [x \ y \ z]$  and  $v = [a \ b \ c]$

- $v \cdot u = ax + by + cz$

The Euclidean distance of  $u$  from the origin is  $\sqrt{x^2 + y^2 + z^2}$  and is denoted by  $\|u\|$

- Notice that  $\|u\| = \sqrt{u \cdot u}$

The Euclidean distance between  $u$  and  $v$  is  $\sqrt{(x-a)^2 + (y-b)^2 + (z-c)^2}$  and is denoted by  $\|u - v\|$

## PROPERTIES OF THE DOT PRODUCT

---

Given a vector  $u, v, w$  and scalar  $s$

- The result of a dot product is a SCALAR value
- Commutative:  $v \cdot w = w \cdot v$
- Non-degenerate:  $v \cdot v = 0$  only when  $v = 0$
- Bilinear:  $v \cdot (u + sw) = v \cdot u + s(v \cdot w)$

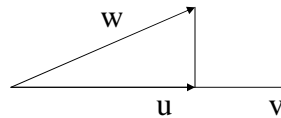
## ANGLES AND PROJECTION

---

Alternative view of the dot product  
 $v \cdot w = \|v\| \|w\| \cos(\theta)$  where  $\theta$  is the angle between  $v$  and  $w$

If  $v$  is a unit vector ( $\|v\| = 1$ ) then if we perpendicularly project  $w$  onto  $v$  we can call this newly projected vector  $u$  then

$$\|u\| = v \cdot w$$



## MATRICES

---

Linear operations on vector spaces

3x3 Matrix A looks like

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$a(i,j)$  refers to the element of matrix A

## MATRIX MULTIPLICATION

---

If  $A$  is an  $n \times k$  matrix and  $B$  is a  $k \times p$  then  $AB$  is a  $n \times p$  matrix with entries  $c(i,j)$  where

■  $c(i,j) = \sum a(i,s)b(s,j)$

Alternatively, if we took the rows of  $A$  and columns of  $B$  as individual vectors then  $c(i,j) = A_i^t \cdot B_j$  where the subscript refers to the row and column, respectively

## MATRIX MULTIPLICATION PROPERTIES

---

Associative:  $(AB)C = A(BC)$

Distributive:  $A(B+C) = AB+AC$

Multiplicative Identity:  $I = \text{diag}(1)$   
(square matrix)

NOT commutative:  $AB \neq BA$

## DETERMINANT

---

Defined on a square matrix (nxn)

$$\det A = |A| = \sum_{i=1}^n (-1)^{1+i} A_{1i}$$

Where  $A_{1i}$  determinant of  $(n-1) \times (n-1)$  submatrix  $A$  gotten by deleting the first row and the  $i$ th column

## RECURSIVE DEFINITION!!

---

The basis case

det of a 2x2 matrix is defined to be  $ad-bc$  where

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

## USES OF THE DETERMINANT?

---

Linear Independence of columns in a matrix

Cross Product

- Given 2 vectors  $v=[v_1 \ v_2 \ v_3]$ ,  $w=[w_1 \ w_2 \ w_3]$ , the cross product is defined to be the determinant of

$$\begin{vmatrix} i & j & k \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix} = \begin{pmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{pmatrix}$$

## CROSS PRODUCT PROPERTIES

---

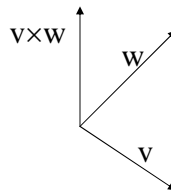
The Cross Product of  $v$  and  $w$  is denoted by  $v \times w$

Is a VECTOR, perpendicular to the plane defined by  $v$  and  $w$

$$\|v \times w\| = \|v\| \|w\| \sin \theta$$

- $\theta$  is the angle between  $v$  and  $w$

$$v \times w = -(w \times v)$$



## MATRIX TRANSPOSE AND INVERSE

---

The Transpose of a matrix A, denoted by  $A^T$  is defined as  $a_{ij}=a_{ji}$  (exchanging the rows and columns)

If A and B are nxn matrices and  $AB=BA=I$  then B is the inverse of A, denoted by  $A^{-1}$   
 $(AB)^{-1}=B^{-1}A^{-1}$  same applies for transpose

$$M^{T^{-1}} = M^{-1T}$$

## METHODS FOR FINDING THE INVERSE

---

### Explicit Methods

- Gaussian-Jordan Elimination
  - Create the Augmented matrix  $[A|I]$  and reduce the left side to the identity using elementary row operations and the right hand side will be the inverse. ie.  $[I|A^{-1}]$
- Cramer's Rule
  - Solve for  $A^{-1}$  where  $a'_{ij}=\det(\text{submatrix}(A_{ij}))$
  - $A^{-1}=(1/\det(A))(A')^T$

## IMPLICIT METHODS

---

Instead of explicitly calculating  $A^{-1}$ , there are techniques that solve equations of the form  $Ax=b$  (system of linear equations).

Clearly  $x=A^{-1}b$  but we do not need to explicitly calculate  $A^{-1}$  to calculate  $x$ .

- LU Decomposition
- QR Factorization
- Singular Value Decomposition (SVD)
- Conjugate Gradient if sparse...

## TRANSFORMATIONS

---

Why use transformations?

- Create object in convenient coordinates
- Reuse basic shape multiple times
- Hierarchical modeling
- System independent
- Virtual cameras

# TRANSLATION

---

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$T(t_x, t_y, t_z) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

# PROPERTIES OF TRANSLATION

---

$$T(0,0,0) \mathbf{v} = \mathbf{v}$$

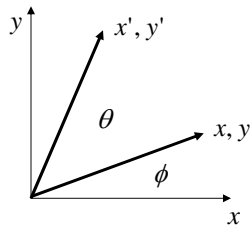
$$T(s_x, s_y, s_z) T(t_x, t_y, t_z) \mathbf{v} = T(s_x + t_x, s_y + t_y, s_z + t_z) \mathbf{v}$$

$$T(s_x, s_y, s_z) T(t_x, t_y, t_z) \mathbf{v} = T(t_x, t_y, t_z) T(s_x, s_y, s_z) \mathbf{v}$$

$$T^{-1}(t_x, t_y, t_z) \mathbf{v} = T(-t_x, -t_y, -t_z) \mathbf{v}$$

## ROTATIONS (2D)

---



$$\begin{cases} x' = x \cos \theta - y \sin \theta \\ y' = x \sin \theta + y \cos \theta \end{cases}$$

$$x = r \cos \phi$$

$$y = r \sin \phi$$

$$x' = r \cos(\phi + \theta)$$

$$y' = r \sin(\phi + \theta)$$

$$\cos(\phi + \theta) = \cos \phi \cos \theta - \sin \phi \sin \theta$$

$$\sin(\phi + \theta) = \cos \phi \sin \theta + \sin \phi \cos \theta$$

$$x' = (r \cos \phi) \cos \theta - (r \sin \phi) \sin \theta$$

$$y' = (r \cos \phi) \sin \theta + (r \sin \phi) \cos \theta$$

## ROTATIONS 2D

---

So in matrix notation

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

## ROTATIONS (3D)

---

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## PROPERTIES OF ROTATIONS

---

$$R_a(0) = I$$

$$R_a(\theta)R_a(\phi) = R_a(\phi + \theta)$$

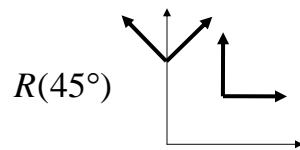
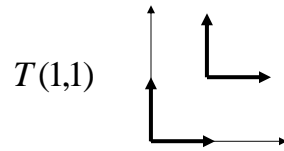
$$R_a(\theta)R_a(\phi) = R_a(\phi)R_a(\theta)$$

$$R_a^{-1}(\theta) = R_a(-\theta) = R_a^T(\theta)$$

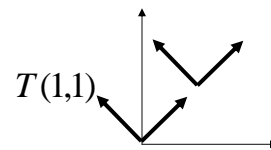
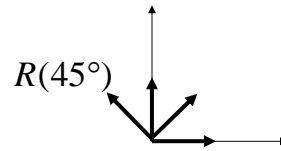
$$R_a(\theta)R_b(\phi) \neq R_b(\phi)R_a(\theta) \quad \text{order matters!}$$

## COMBINING TRANSLATION & ROTATION

---



CSE 872 FALL 2009



31

## COMBINING TRANSLATION & ROTATION

---

$$\mathbf{v}' = \mathbf{v} + T$$

$$\mathbf{v}'' = R\mathbf{v}'$$

$$\mathbf{v}'' = R(\mathbf{v} + T)$$

$$\mathbf{v}'' = R\mathbf{v} + RT$$

$$\mathbf{v}' = R\mathbf{v}$$

$$\mathbf{v}'' = \mathbf{v}' + T$$

$$\mathbf{v}'' = R\mathbf{v} + T$$

CSE 872 FALL 2009

32

# SCALING

---

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \end{bmatrix}$$

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

Uniform scaling **iff**  $s_x = s_y = s_z$

# AFFINE OPERATIONS

---

## Translation

- $x' = x + c$
- $y' = y + f$

## Scaling

- $x' = ax$
- $y' = dy$

## Rotation (CCW about o,o)

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

## Skew (or Shear)

- $x' = x + ay$
- $y' = y$



# HOMOGENEOUS COORDINATES

---

3D Projective Space: each point seen as a line in 4D

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ can be represented as } \begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix}$$

$$\text{where } x = \frac{X}{w}, \quad y = \frac{Y}{w}, \quad z = \frac{Z}{w}$$

# TRANSLATION REVISITED

---

$$T(t_x, t_y, t_z) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

## ROTATION & SCALING REVISITED

---

$$R_x(\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$S(s_x, s_y, s_z) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

## COMBINING TRANSFORMATIONS

---

$$\mathbf{v}' = S\mathbf{v}$$

$$\mathbf{v}'' = R\mathbf{v}' = RS\mathbf{v}$$

$$\mathbf{v}''' = T\mathbf{v}'' = TR\mathbf{v}' = TRS\mathbf{v}$$

$$\mathbf{v}''' = M\mathbf{v}$$

where  $M = TRS$

## TRANSFORMING TANGENTS

---

$$\mathbf{t} = \mathbf{p} - \mathbf{q}$$

$$\begin{aligned}\mathbf{t}' &= \mathbf{p}' - \mathbf{q}' \\ &= M\mathbf{p} - M\mathbf{q} \\ &= M(\mathbf{p} - \mathbf{q}) \\ &= M\mathbf{t}\end{aligned}$$

## TRANSFORMING NORMALS

---

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

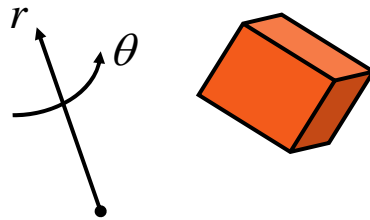
$$\begin{aligned}\mathbf{n}'^T \mathbf{t}' &= 0 \\ \mathbf{n}'^T M\mathbf{t} &= 0 \\ \mathbf{n}'^T M\mathbf{t} &= \mathbf{n}^T \mathbf{t} \\ \mathbf{n}'^T M &= \mathbf{n}^T \\ M^T \mathbf{n}' &= \mathbf{n}\end{aligned}$$

$$\mathbf{n}' = M^{T^{-1}} \mathbf{n} = M^{-1T} \mathbf{n}$$

## ROTATIONS ABOUT AN ARBITRARY AXIS

---

Rotate by  $\theta$  around a unit axis  $r$



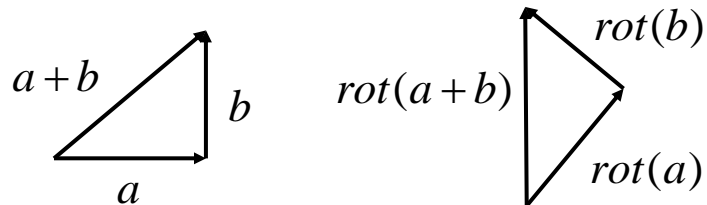
## ROTATIONS ABOUT AN ARBITRARY AXIS

---

Rotation is linear

$$\text{rot}(a + b) = \text{rot}(a) + \text{rot}(b)$$

$$\text{rot}(\alpha a) = \alpha \text{rot}(a)$$



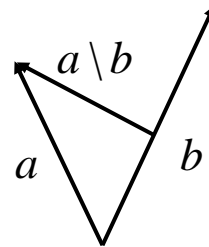
## ROTATIONS ABOUT AN ARBITRARY AXIS

---

Projection operator:

$$a \setminus b = a - \alpha b$$

$$\alpha = \frac{a \cdot b}{b \cdot b}$$



## ROTATIONS ABOUT AN ARBITRARY AXIS

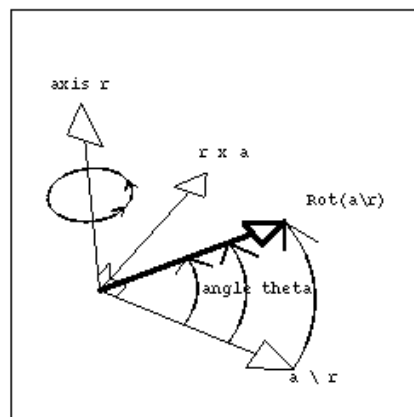
---

Create an orthonormal basis:

$$e_3 = r$$

$$e_1 = \frac{a \setminus r}{|a \setminus r|}$$

$$e_2 = e_3 \times e_1$$



## ROTATIONS ABOUT AN ARBITRARY AXIS

---

Decompose before rotating:  $a = a \setminus r + \alpha r$

$$\begin{aligned} \text{rot}(a) &= \text{rot}(a \setminus r + \alpha r) \\ &= \text{rot}(a \setminus r) + \text{rot}(\alpha r) \\ &= \text{rot}(|a \setminus r| e_1) + \alpha r \\ &= |a \setminus r| \text{rot}(e_1) + \alpha r \\ &= |a \setminus r| (\cos \theta e_1 + \sin \theta e_2) + \alpha r \\ &= \cos \theta (a \setminus r) + \sin \theta (r \times a) + (a \cdot r)r \end{aligned}$$

## AN ALTERNATIVE VIEW

---

We can view the rotation around an arbitrary axis as a set of simpler steps

We know how to rotate and translate around the world coordinate system

Can we use this knowledge to perform the rotation?

## ROTATION ABOUT AN ARBITRARY AXIS

---

Translate the space so that the origin of the unit vector is on the world origin

Rotate such that the extremity of the vector now lies in the xz plane (x-axis rotation)

Rotate such that the point lies in the z-axis (y-axis rotation)

Perform the rotation around the z-axis

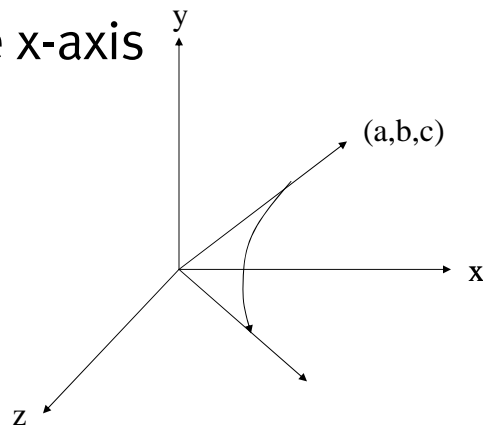
Undo the previous transformations

## ROTATION ABOUT AN ARBITRARY AXIS

---

Step 1

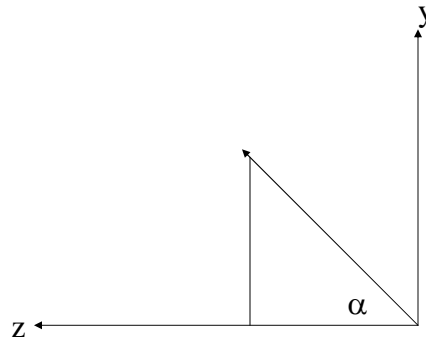
Rotate x-axis



## CLOSER LOOK AT Y-Z PLANE

---

Need to rotate  $\alpha$  degrees around the x-axis



## EQUATIONS FOR $\alpha$

---

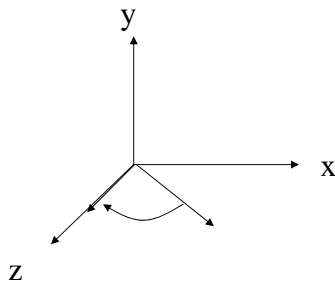
$$\sin(\alpha) = \frac{\| (0,0,1) \times (0,b,c) \|}{\| (0,0,1) \| \| (0,b,c) \|}$$

$$\cos(\alpha) = \frac{(0,b,c) \cdot (0,0,1)}{\| (0,b,c) \| \| (0,0,1) \|}$$

## ROTATION ABOUT THE Y-AXIS

---

Using the same analysis as before,  
we need to rotate  $\beta$  degrees  
around the Y-axis



## ROTATION ABOUT THE Z-AXIS

---

Now, it is aligned with the Z-axis,  
thus we can simply rotate  $\theta$   
degrees around the Z-axis.

Then undo all the transformations  
we just did

## EQUATION SUMMARY

---

$$rot_{axis}(\theta) = \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = T^{-1}R_x^{-1}(\alpha)R_y^{-1}(\beta)R_z(\theta)R_y(\beta)R_x(\alpha)T \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

## DEFORMATIONS

---

Transformations that do not preserve shape

- Non-uniform scaling
- Shearing
- Tapering
- Twisting
- Bending

# SHEARING

---

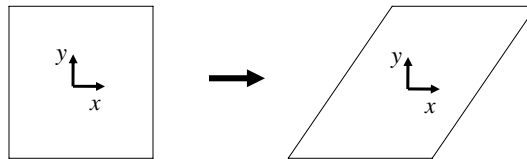
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s_{xy} & s_{xz} & 0 \\ s_{yx} & 1 & s_{yz} & 0 \\ s_{zx} & s_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$s_{xy} = 1$$

$$s_{xz} = 0$$

$$s_{yx} = s_{yz} = 0$$

$$s_{zx} = s_{zy} = 0$$



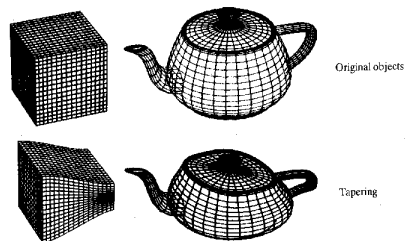
CSE 872 FALL 2009

55

# TAPERING

---

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & f(x) & 0 & 0 \\ 0 & 0 & f(x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



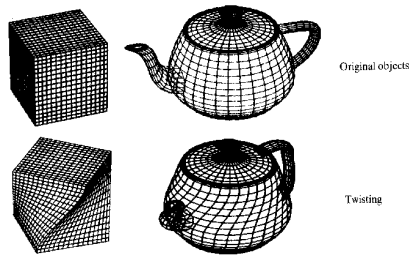
CSE 872 FALL 2009 Image courtesy of Watt, 3D Computer Graphics

56

# TWISTING

---

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



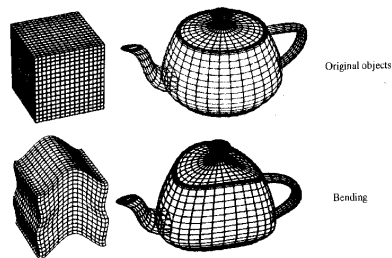
CSE 872 FALL 2009 Image courtesy of Watt, 3D Computer Graphics

57

# BENDING

---

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & f(y) & g(y) & 0 \\ 0 & h(y) & k(y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



CSE 872 FALL 2009 Image courtesy of Watt, 3D Computer Graphics

58

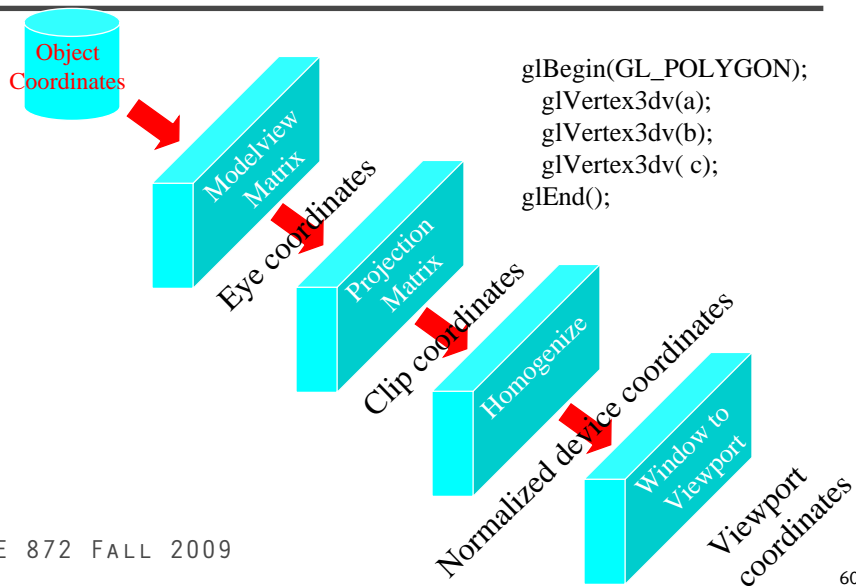
# COORDINATE SYSTEMS

---

Object coordinates  
World coordinates  
Camera coordinates  
Normalized device coordinates  
Window coordinates

# WITHIN OPENGL

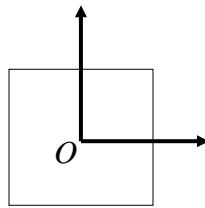
---



## OBJECT COORDINATES

---

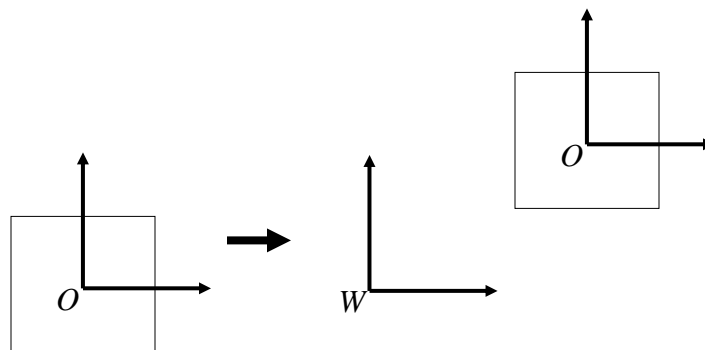
Convenient place to model the object



## WORLD COORDINATES

---

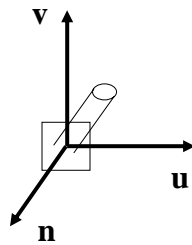
Common coordinates for the scene



## CAMERA COORDINATES

---

Coordinate system with the camera  
in a convenient pose



$$M = \begin{bmatrix} u_x & u_y & u_z & -\mathbf{r} \cdot \mathbf{u} \\ v_x & v_y & v_z & -\mathbf{r} \cdot \mathbf{v} \\ n_x & n_y & n_z & -\mathbf{r} \cdot \mathbf{n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## NORMALIZED DEVICE COORDINATES

---

Device independent coordinates  
Visible coordinate usually range  
from:

$$-1 \leq x \leq 1$$

$$-1 \leq y \leq 1$$

$$-1 \leq z \leq 1$$

## WINDOW COORDINATES

---

Adjusting the NDC to fit the window

$(x_0, y_0)$  is the lower left of the window

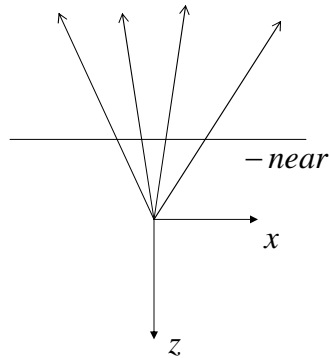
$$x_w = (x_{nd} + 1) \left( \frac{\text{width}}{2} \right) + x_0$$

$$y_w = (y_{nd} + 1) \left( \frac{\text{height}}{2} \right) + y_0$$

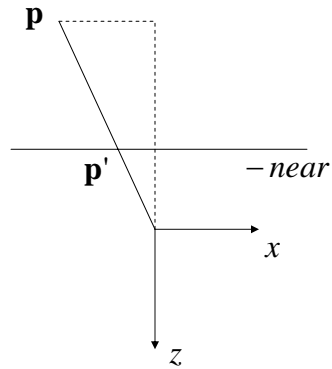
## PERSPECTIVE PROJECTION

---

Taking the camera coordinates to NDC



# PERSPECTIVE PROJECTION



$$\frac{x'}{-near} = \frac{x}{z}$$

$$x' = -near \frac{x}{z}$$

# PERSPECTIVE PROJECTION

Map (left,right) to (-1,1) when  $z = -near$

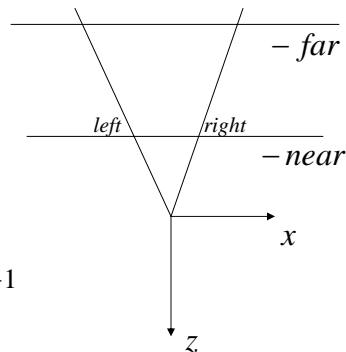
$$x' = -near \frac{x}{z}$$

$$-near \frac{x}{z} - left$$

$$\frac{2}{right - left} \left( -near \frac{x}{z} - left \right)$$

$$\frac{2}{right - left} \left( -near \frac{x}{z} - left \right) - 1$$

$$\frac{-2near}{right - left} \frac{x}{z} - \frac{right + left}{right - left}$$



## PSEUDODEPTH

---

Map (-near,-far) to (-1,1)

$$z' = \frac{far + near}{far - near} + \frac{2 far \cdot near}{far - near} \frac{1}{z}$$

Lines are preserved through the transformation

See Newman and Sproull '81 for the full derivation

## PERSPECTIVE PROJECTION

---

$$P = \begin{bmatrix} \frac{2near}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2near}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & -\frac{2 far \cdot near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

## PERSPECTIVE PROJECTION PRESERVES STRAIGHT LINES

---

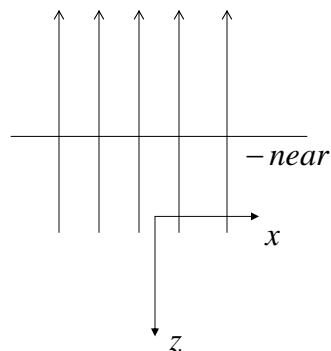
$$\begin{bmatrix} a_x + b_x t \\ a_y + b_y t \\ a_z + b_z t \\ 1 \end{bmatrix} \text{ maps to } \begin{bmatrix} A(a_x + b_x t) + B(a_z + b_z t) \\ C(a_y + b_y t) + D(a_z + b_z t) \\ E(a_z + b_z t) + F \\ -(a_z + b_z t) \end{bmatrix}$$

Compute  $x(t_2)-x(t_1)$ ,  $y(t_2)-y(t_1)$ ,  $z(t_2)-z(t_1)$ .  
All have numerators that depend on  $(t_2-t_1)$ .  
All have the same denominator.

Therefore, the ratios  $\frac{x(t_2)-x(t_1)}{y(t_2)-y(t_1)}$  ... are constants.

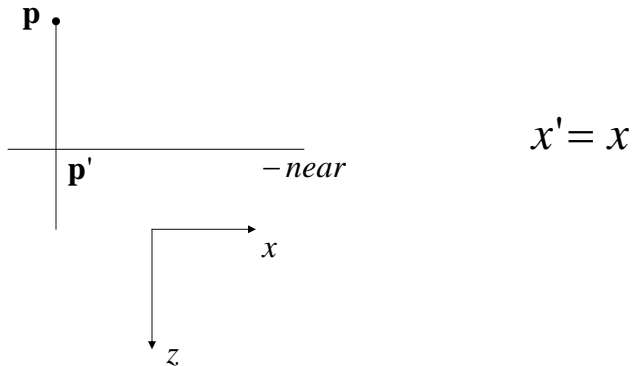
## ORTHOGRAPHIC PROJECTION

---



# ORTHOGRAPHIC PROJECTION

---



CSE 872 FALL 2009

73

# ORTHOGRAPHIC PROJECTION

---

Map (left,right) to (-1,1)

$$x' = x$$

$$x - left$$

$$\frac{2}{right - left}(x - left)$$

$$\frac{2}{right - left}(x - left) - 1$$

$$\frac{2}{right - left}x - \frac{right + left}{right - left}$$

CSE 872 FALL 2009

74

## ORTHOGRAPHIC PROJECTION

---

Map (near, far) to (-1,1)

$$\frac{z}{z - \text{near}}$$
$$\frac{2}{\text{far} - \text{near}}(z - \text{near})$$
$$\frac{2}{\text{far} - \text{near}}(z - \text{near}) - 1$$
$$\frac{2}{\text{far} - \text{near}}z - \frac{\text{far} + \text{near}}{\text{far} - \text{near}}$$

## ORTHOGRAPHIC PROJECTION

---

$$P = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{top} - \text{bottom}} & 0 & -\frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} \\ 0 & 0 & \frac{2}{\text{far} - \text{near}} & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## ORTHOGRAPHIC PROJECTIONS PRESERVE STRAIGHT LINES

---

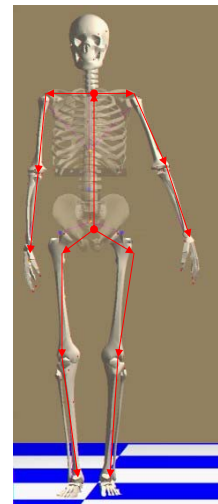
$$\begin{bmatrix} a_x + b_x t \\ a_y + b_y t \\ a_z + b_z t \\ 1 \end{bmatrix} \text{ maps to } \begin{bmatrix} A(a_x + b_x t) + B \\ C(a_y + b_y t) + D \\ E(a_z + b_z t) + F \\ 1 \end{bmatrix}$$

## WHY QUATERNIONS?

---

The motion of the skeleton is underlying most of character animation

Most skeletons are articulated:  
A figure made up of a series of links (bones) connected at joints (rotations)



## ROTATIONS USING QUATERNIONS\*

---

Quaternions

$$q = q_1 + i q_2 + j q_3 + k q_4$$

$$\begin{array}{l} i^2 = j^2 = k^2 = -1 \\ i j k = -1 \end{array}$$

$$q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \begin{pmatrix} s \\ v \end{pmatrix}$$

## ROTATIONS USING QUATERNIONS\*

---

Quaternion multiplication

$$q p = \begin{pmatrix} s_q s_p - v_q \cdot v_p \\ s_q v_p + s_p v_q + v_q \times v_p \end{pmatrix}$$

$$(p q) r = p (q r) \quad \text{Associative}$$

$$p q \neq q p \quad \text{Not Commutative}$$

## ROTATIONS USING QUATERNIONS\*

---

Quaternion magnitude

$$|q| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} = \sqrt{s_q^2 + v_q \cdot v_q}$$

Quaternion inverse

$$q^{-1} = \frac{1}{|q|^2} \begin{pmatrix} s \\ -v \end{pmatrix}$$

## ROTATIONS USING QUATERNIONS\*

---

Quaternion rotation:

$$q = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)r \end{pmatrix}$$

$$\text{rot} \begin{pmatrix} 0 \\ a \end{pmatrix} = q \begin{pmatrix} 0 \\ a \end{pmatrix} q^{-1}$$

## ROTATIONS USING QUATERNIONS\*

---

Combining rotations:

$$\text{rot}_1 \begin{pmatrix} 0 \\ a \end{pmatrix} = q_1 \begin{pmatrix} 0 \\ a \end{pmatrix} q_1^{-1}$$

$$\text{rot}_2 \left( \text{rot}_1 \begin{pmatrix} 0 \\ a \end{pmatrix} \right) = q_2 q_1 \begin{pmatrix} 0 \\ a \end{pmatrix} q_1^{-1} q_2^{-1}$$

$$\text{rot}_2 \left( \text{rot}_1 \begin{pmatrix} 0 \\ a \end{pmatrix} \right) = (q_2 q_1) \begin{pmatrix} 0 \\ a \end{pmatrix} (q_2 q_1)^{-1}$$

## ROTATIONS USING QUATERNIONS\*

---

Matrix Form:

$$q = w + i x + j y + k z$$

$$\begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2xz + 2wy & 0 \\ 2xy + 2wz & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx & 0 \\ 2xz - 2wy & 2yz + 2wx & w^2 - x^2 - y^2 + z^2 & 0 \\ 0 & 0 & 0 & w^2 + x^2 + y^2 + z^2 \end{bmatrix}$$

## SUMMARY

---

- Points/Vectors represented by
  - Vectors
  - Described in coordinate systems
- Transformations
  - Objects, Change of coord's
  - Projection matrix for camera
- Quaternions for rotation