University of British Columbia

Composing Geometric Transformations

- composition of transformations
- scene graphs

Transformations

University of British Columbia

Affine transformations

- linear transformation + translations
- can be expressed as a 3x3 matrix + 3 vector

$$P' = M \cdot P + T$$

4x4 matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

h=1

Transformations

University of British Columbia

translate(a,b,c)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

scale(a,b,c)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$Rotate(z,\theta)$

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

Simple Compositions

University of British Columbia

translate(a,b,c) translate(d,e,f)

$$\begin{bmatrix} 1 & & & a+d \\ & 1 & & b+e \\ & & 1 & c+f \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & d \\ & 1 & & e \\ & & 1 & f \\ & & & 1 \end{bmatrix}$$

scale(a,b,c) scale(d,e,f)

$$\begin{bmatrix} ad & & & \\ & be & & \\ & & cf & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} d & & & \\ & e & & \\ & & f & \\ & & & 1 \end{bmatrix}$$

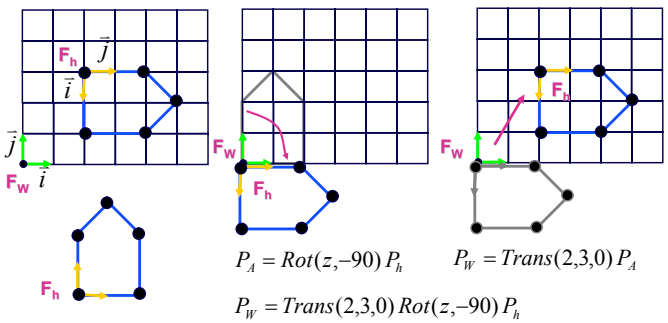
$Rotate(z,\theta_1)\ Rotate(z,\theta_2)$

$$\begin{bmatrix} \cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & & \\ \sin\theta_1 & \cos\theta_1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & & \\ \sin\theta_2 & \cos\theta_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Composing Transformations

University of British Columbia

suppose we want | **Rotate(z,-90)** | **Translate(2,3,0)**



$P_A = Rot(z,-90)\,P_h$

$P_W = Trans(2,3,0)\,P_A$

$P_W = Trans(2,3,0)\,Rot(z,-90)\,P_h$

Composing Transformations

University of British Columbia

$P_W = Trans(2,3,0)\,Rot(z,-90)\,P_h$

- R-to-L: interpret operations wrt fixed coords
- L-to-R: interpret operations wrt local coords
- OpenGL (L-to-R, local coords)

```
glTranslatef(2,3,0);
glRotatef(-90,0,0,1);
draw_house();
```

$M_{MV} = M_{MV} \cdot Trans(2,3,0)$

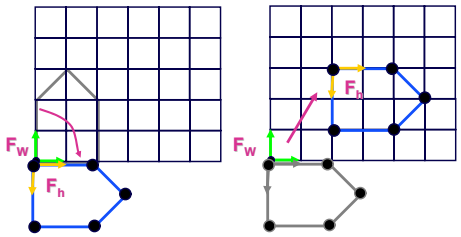
$M_{MV} = M_{MV} \cdot Rot(z,-90)$

updates current transformation matrix by postmultiplying

Composing Transformations

University of British Columbia

Rotate(z,-90) | Translate(-3,2,0) in local coords



$P_W = Rot(z,-90)\,Trans(-3,2,0)\,P_h$

```
glRotatef(-90,0,0,1);
glTranslatef(-3,2,0);
draw_house();
```


Composing Transformations

University of British Columbia

Equivalence

$P_W = Trans(2,3,0)\,Rot(z,-90)\,P_h$

$P_W = Rot(z,-90)\,Trans(-3,2,0)\,P_h$

Undoing Transformations

$Trans(x,y,z)^{-1} = Trans(-x,-y,-z)$

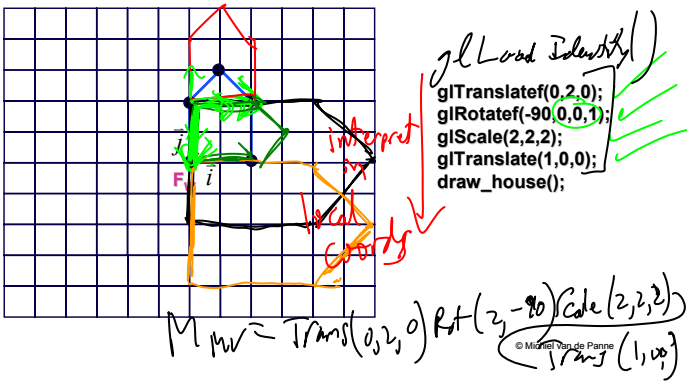
$Trans(x,y,z)\;Trans(-x,-y,-z) = I$

$Rot(z,\theta)^{-1} = Rot(z,-\theta)$

$Rot(z,\theta)\;Rot(z,-\theta) = I$

Test yourself ...

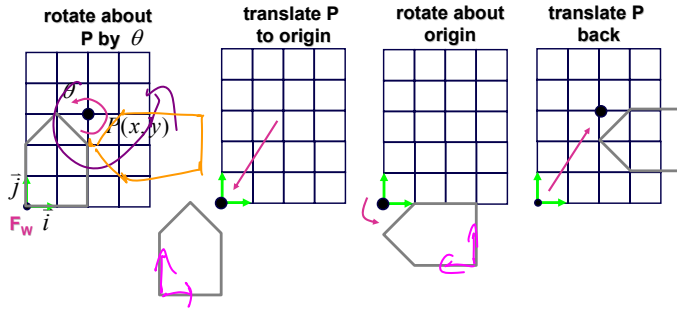
University of British Columbia




```
glTranslatef(0,2,0);
glRotatef(-90,0,0,1);
glScale(2,2,2);
glTranslate(1,0,0);
draw_house();
```


Rotation about a point

University of British Columbia



$Trans(x,y,z)\,Rot(z,\theta)\,Trans(-x,-y,-z)$

Rotation about an arbitrary axis

University of British Columbia

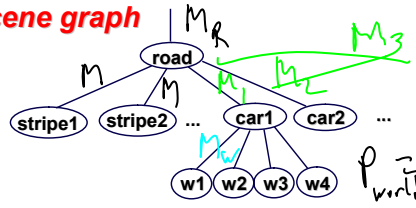
- axis defined by two points
- translate point to the origin
- rotate to align axis with z-axis (or x or y)
- perform rotation
- undo aligning rotations
- undo translation

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


Transformation Hierarchies

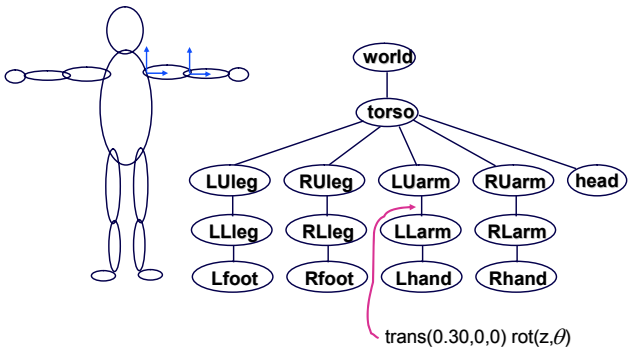
University of British Columbia

scene graph



Transformation Hierarchies

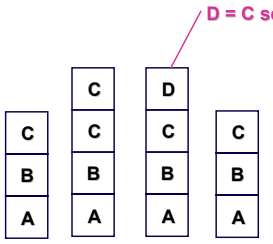
University of British Columbia



Transformation Hierarchies

University of British Columbia

Matrix Stack




```
DrawSquare()
glPushMatrix()
glScale3f(2,2,2)
glTranslate3f(1,0,0)
DrawSquare()
popmatrix()
```


Hierarchical Modeling with Matrix Stacks

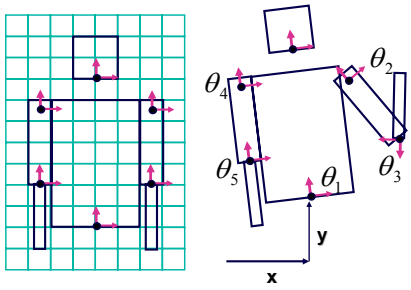
University of British Columbia

- provides a means of returning to a previously-used coordinate system
- graphical scenes and characters have a natural hierarchical structure
- depth of matrix stacks is limited in hardware
  - *typically: 16 for ModelView, 4 for Projection*

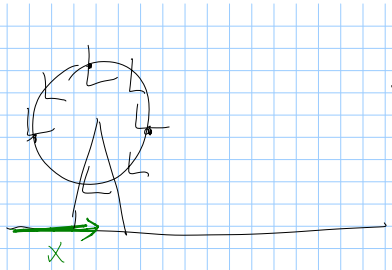
Transformation Hierarchies

University of British Columbia

Example



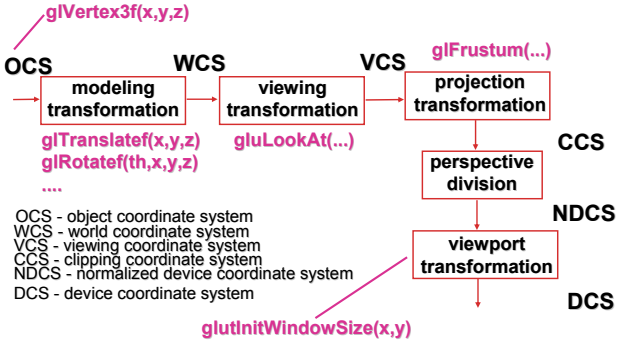

```
glTranslatef(x,y,0);
glRotatef(θ1,0,0,1);
DrawBody();
glPushMatrix();
  glTranslatef(0,7,0);
  DrawHead();
glPopMatrix();
glPushMatrix();
  glTranslate(2.5,5.5,0);
  glRotatef( θ2,0,0,1);
  DrawUArm();
  glTranslate(0,-3.5,0);
  glRotatef( θ3,0,0,1);
  DrawLArm();
glPopMatrix();
... (draw other arm)
```


```
glPushMatrix();
glTranslatef( x,0, 0);
draw_supports();
glTranslate(0,5,0);
for (θ = 0; θ < 360; θ += 45){
    glPushMatrix();
    dx = r cosθ
    dy = r sinθ
    glTranslate( dx,dy,0);
    draw_seat();
    glPopMatrix();
}
glPopMatrix();
```

Projective Rendering Pipeline

University of British Columbia



OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

Coming Up…

University of
British Columbia

- animation
- projection transformations
- viewing transformations