University of British Columbia

Composing Geometric Transformations

- composition of Transformations
- scene graphs

University of British Columbia

Transformations

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

University of British Columbia

Transformations

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)

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

$Rotate(z, \theta)$

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

University of British Columbia

$c\theta_1 \cdot c\theta_2 - s\theta_1 \cdot s\theta_2$
$= \cos(\theta_1 + \theta_2)$

Simple Compositions

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

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

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

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

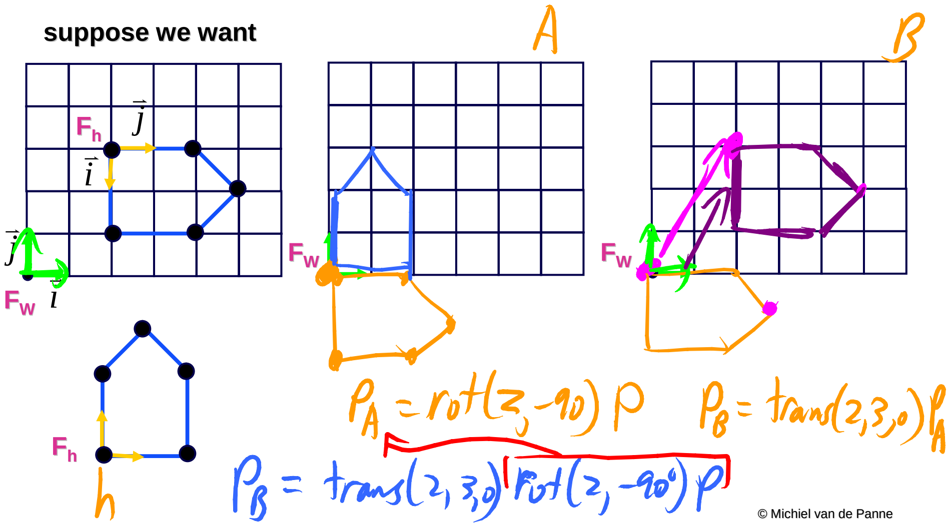
$Rotate(z, \theta_1)\ Rotate(z, \theta_2)$

$$\begin{bmatrix} c\theta_1 & -s\theta_1 & & \\ s\theta_1 & c\theta_1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & -s\theta_2 & & \\ s\theta_2 & c\theta_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} \boxed{} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Composing Transformations

University of British Columbia

suppose we want



$P_A = rot(z,-90)\,P$ $\qquad P_B = trans(2,3,0)\,P_A$

$P_B = trans(2,3,0)\,Rot(z,-90°)\,P$

Composing Transformations

University of British Columbia

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

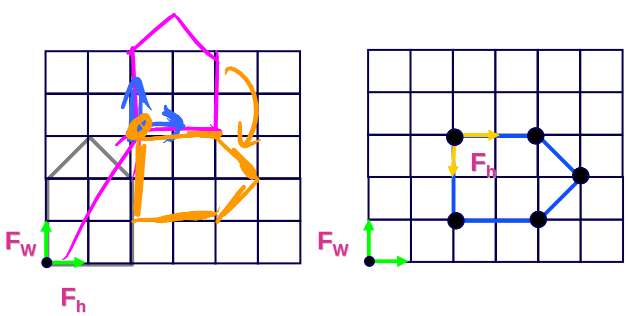
- R-to-L: interpret operations wrt fixed coords
- L-to-R: interpret operations wrt local coords
- OpenGL (L-to-R, local coords)

OpenGL

glTranslatef(2,3,0)
glRotatef(-90,0,0,1)
drawHouse()

Composing Transformations

University of British Columbia



In Local Coords: Trans(2,3,0) ✓
Local Coords Rot(z,-90°)

Composing Transformations

University of British Columbia

Equivalence

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

Undoing Transformations

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

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

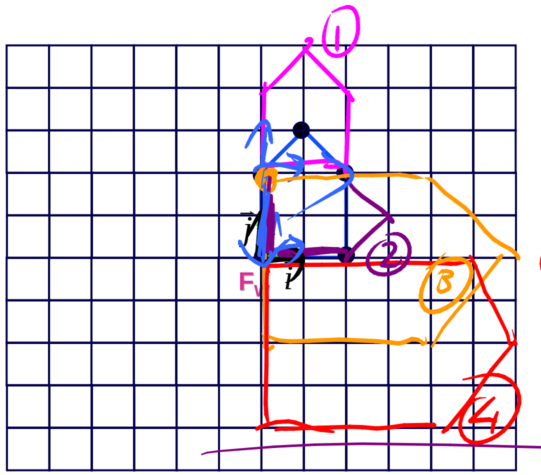
$Rot(z,\theta)^{-1} = Rot(z,-\theta)$

$= Rot(-z,\theta)$

$Scale(a,b,c)^{-1} = Scale(\frac{1}{a},\frac{1}{b},\frac{1}{c})$

Test yourself ...

University of British Columbia



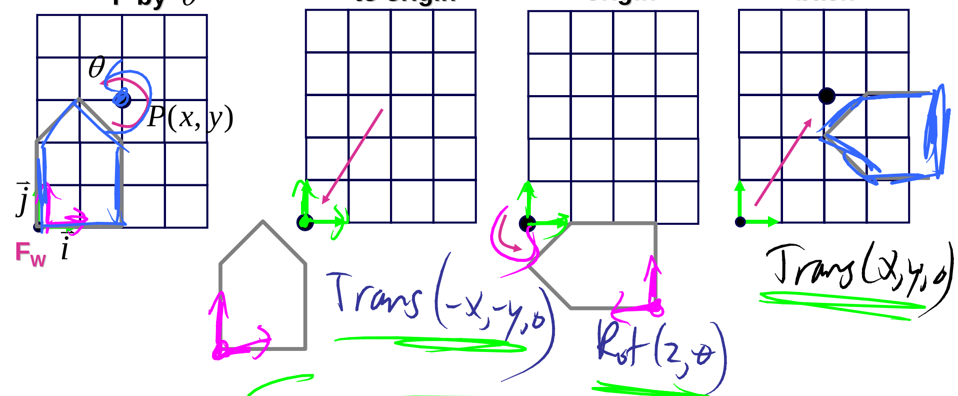
1. glTranslatef(0,2,0);
2. glRotatef(-90,0,0,1);
3. glScale(2,2,2);
4. glTranslate(1,0,0);

draw_house();

$M = \text{Trans}(0,2,0)\ \text{Rot}(z,-90)\ \text{Scale}(2,2,2)\ \text{Trans}(1,0,0)$

Rotation about a point

University of British Columbia



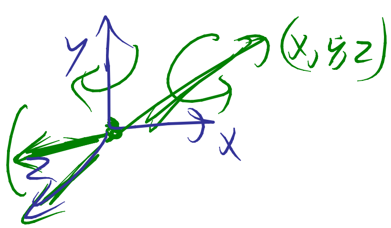
$M = \text{Trans}(x,y,0)\ \text{Rot}(z,\theta)\ \text{Trans}(-x,-y,0)$

Rotation about an arbitrary axis

University of British Columbia

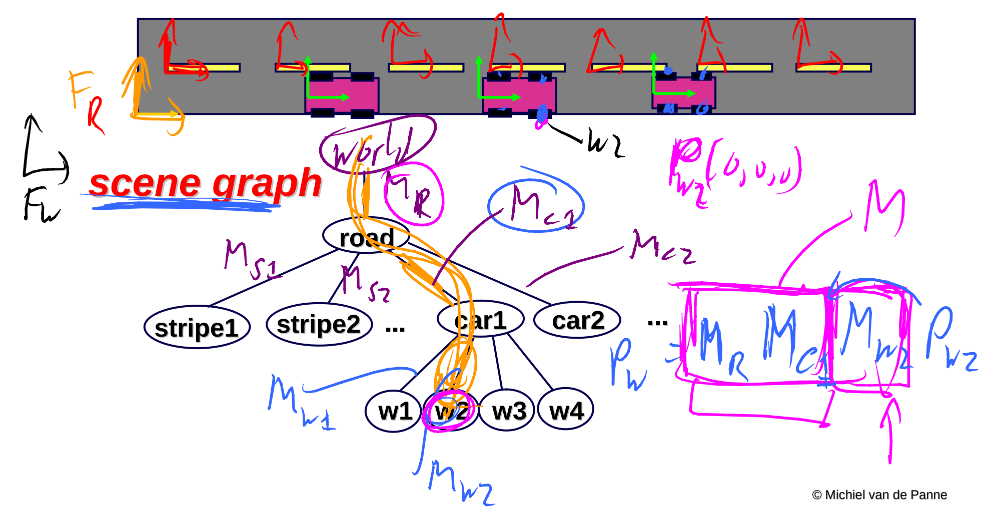
glRotatef(angle, x, y, z);

glRotatef(θ, 0, 0, 1)



Transformation Hierarchies

University of British Columbia

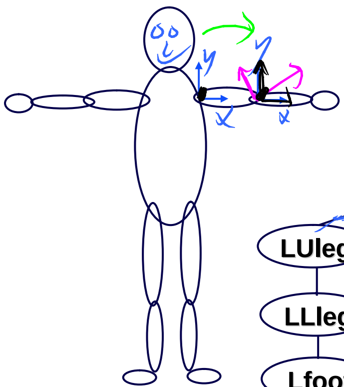


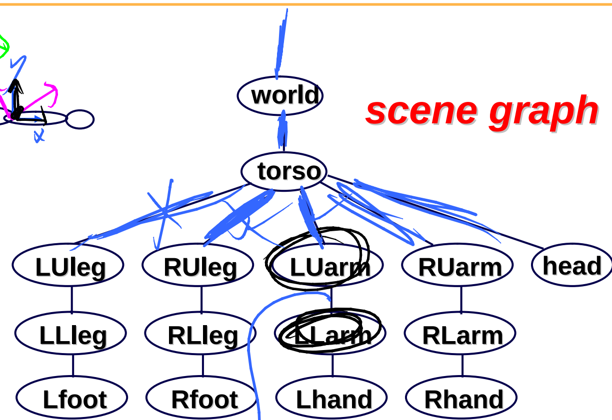
$P_W = M_R\, M_{C1}\, M_{W2}\, P_{W2}$

$P_{W2}(0,0,0)$

Transformation Hierarchies

University of
British Columbia





scene graph

glTranslate(30,0,0)
gl Rotate(θ,0,0,1)

Trans(30cm,0,0) Rot(z,θ)

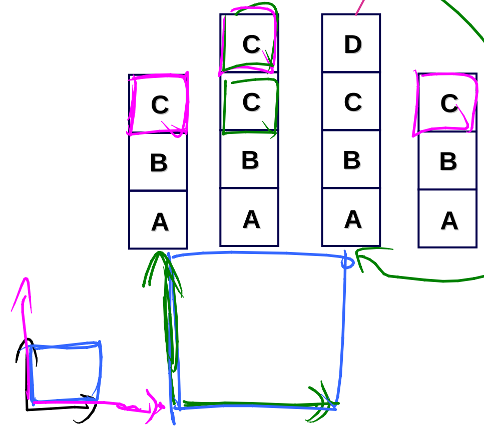
θ

Transformation Hierarchies

University of
British Columbia

Matrix Stack

D = C scale(2,2,2) trans(1,0,0)



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

Hierarchical Modeling with Matrix Stacks

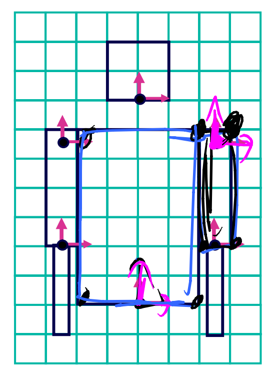
University of
British Columbia

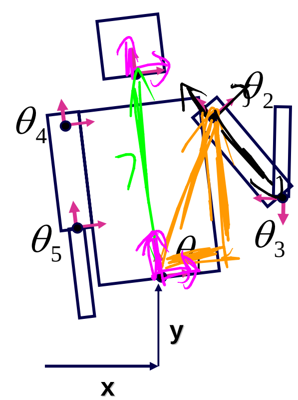
- provides a means of returning to a previously-used coordinate system
- graphical scenes and characters have a natural hierarchical structure
- depth of matrix stacks is limited in hardware
  - *typically: 16 for ModelView, 4 for Projection*

Transformation Hierarchies

University of
British Columbia

Example

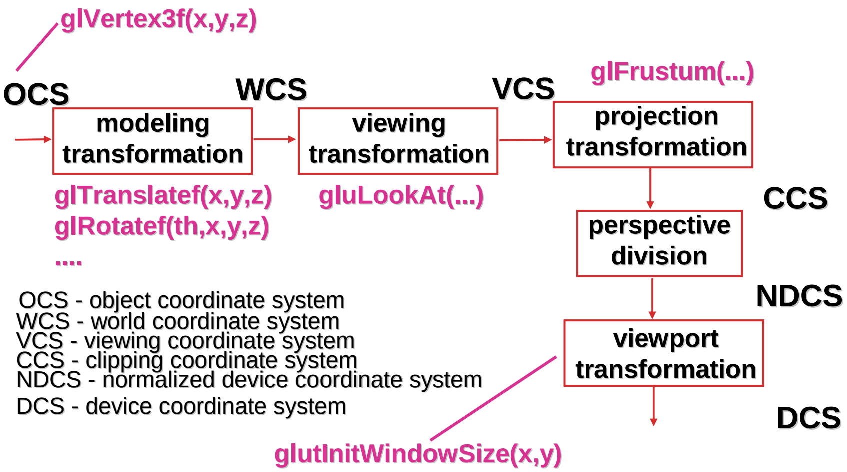





```
glTranslatef(x,y,0);
glRotatef(θ1,0,0,1);
DrawBody();
glPushMatrix();
  glTranslatef(0,7,0);
  DrawHead();
glPopMatrix();
glPushMatrix();
  glTranslate(2.5,5.5,0);
  glRotatef( θ2,0,0,1);
  DrawUArm();
  glTranslate(0,-3.5,0);
  glRotatef(θ3,0,0,1);
  DrawLArm();
glPopMatrix();
... (draw other arm)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

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

- projection transformations
- viewing transformations