Chapter 3.5

Transformations – OpenGL
Composition of Transformations

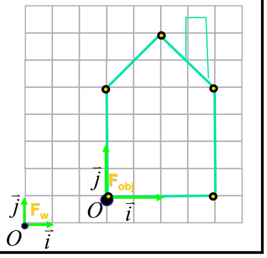
Transformations in OpenGL

- An easier way to do the same thing....

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

glTranslatef(3,1,0);
glScale(2,2,2);

DrawHouse();
```

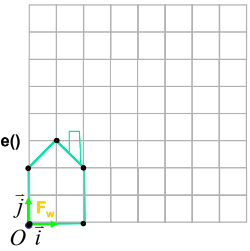


Transformations in OpenGL

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

glBegin(GL_LINE_LOOP);
glVertex2f(0,0);
glVertex2f(2,0);
glVertex2f(2,2);
glVertex2f(1,3);
glVertex2f(0,2);
glEnd();
```

DrawHouse()



$$P' = PT_{MV}$$

Matrix Operations in OpenGL

- 2 Matrices:
  - Model/view matrix M
  - Projective matrix P
- Example:

```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity(); // M=Id
glRotatef( angle, x, y, z ); // M= R(α)*Id
glTranslatef( x, y, z ); // M= T(x,y,z)*R(α)*Id
glMatrixMode( GL_PROJECTION );
glRotatef( … ); // P= …
```

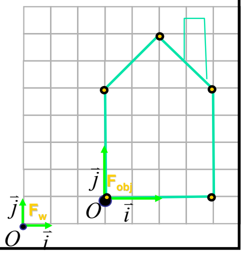
Transformations in OpenGL

$$[x \quad y \quad z \quad 1]_w = [x \quad y \quad z \quad 1]_{obj} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 3 & 1 & 0 & 1 \end{bmatrix}$$

```
GLfloat  T[16] = { 2,0,0,0,  0,2,0,0,
                   0,0,2,0  3,1,0,1};

glMatrixMode(GL_MODELVIEW);
glLoadMatrixf(T);

DrawHouse();
```

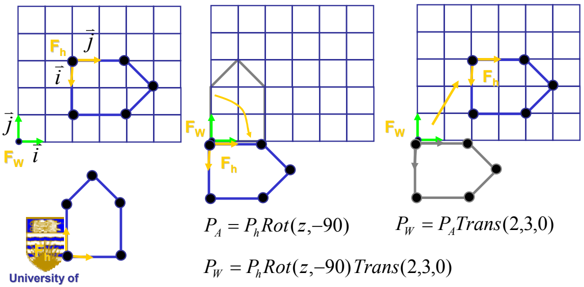


Composing Transformations

suppose we want

Rotate(z,-90)

Translate(2,3,0)



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

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

Composing Transformations

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

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

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

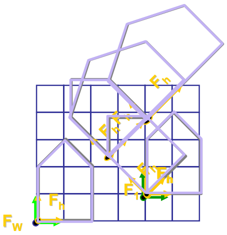
$$M_{MV} = Trans(2,3,0) \cdot M_{MV}$$
$$M_{MV} = Rot(z,-90) M_{MV}$$

updates current transformation matrix by postmultiplying

University of British Columbia

Composing Transformations

- OpenGL example

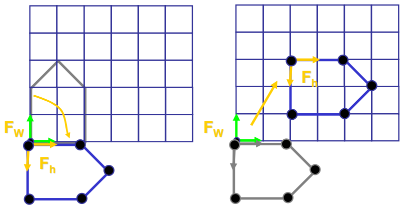


```
glLoadIdentity();
glTranslatef(4,1,0);
glPushMatrix();
glRotatef(45,0,0,1);
glTranslatef(0,2,0);
glScalef(2,1,1);
glTranslate(1,0,0);
glPopMatrix();
```

University of British Columbia

Composing Transformations

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



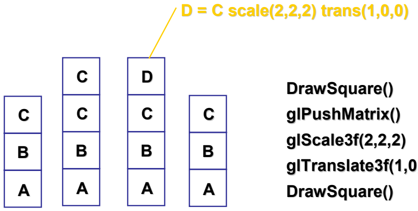
$$P_W = P_h Trans(-3,2,0)\,Rot(z,-90)$$

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

University of British Columbia

Transformation Hierarchies

- Matrix Stack



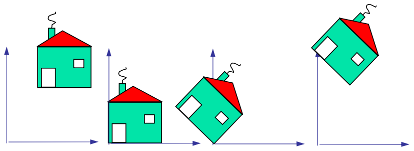
D = C scale(2,2,2) trans(1,0,0)

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

University of British Columbia

Why

- Simplify local operations
- Avoid move to origin



$$T^{(-p_x,-p_y)} R^{\theta} T^{(p_x,p_y)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -p_x & -p_y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{bmatrix}$$

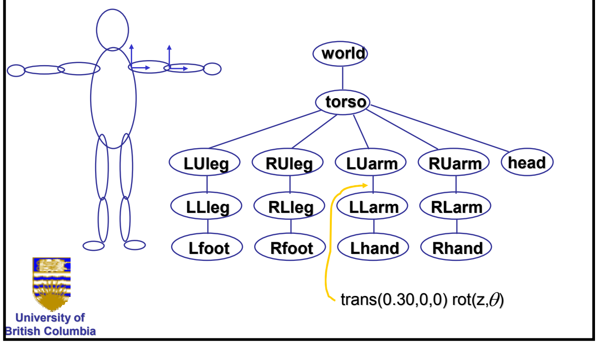
University of British Columbia

Matrix Stacks

- Means of returning to previously-used coordinate system
  - Support several models or model parts
    - Natural hierarchical structure
- depth of matrix stacks limited in hardware
  - typically: 16 for ModelView, 4 for Projection

University of British Columbia

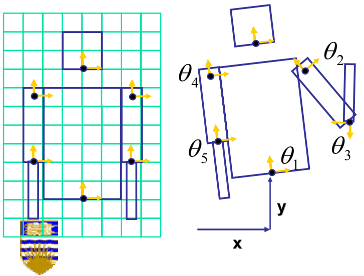
Transformation Hierarchies



University of British Columbia

Transformation Hierarchies

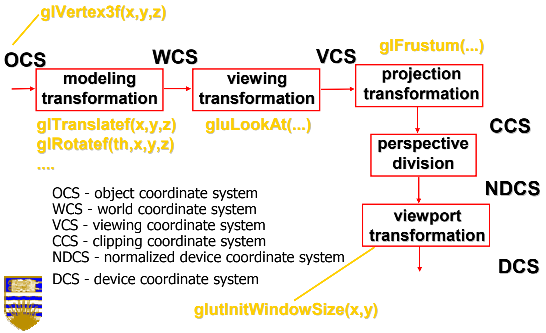
- Example



```
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)
```

University of British Columbia

Projective Rendering Pipeline



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

University of British Columbia