Chapter 4: Transformations - Transforming Normals, Hierarchies and OpenGL, Assignment 2

Transformations in OpenGL

The Rendering Pipeline

Modeling Transformation
- Purpose:
  - Map geometry from local object coordinate system into a global world coordinate system
- Same as placing objects
- Hardware support for arbitrary affine transformations

Viewing Transformation
- Purpose:
  - Map geometry from world coordinate system into camera coordinate system
    - Camera coordinate system is right-handed, viewing direction is negative $z$-axis
  - Same as placing camera
- Transformations:
  - Usually only rigid body transformations
  - Rotations and translations
  - Objects have same size and shape in camera and world coordinates

Model/View Transformation
- Combine modeling and viewing transform
  - Combine into single matrix
- Saves computation time
  - If many points are to be transformed
- Possible because viewing transformation directly follows modeling transformation without intermediate operations
Transformations in OpenGL

```c
void DrawHouse()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glBegin(GL_LINE_LOOP);
    glVertex2f(0,0);
    glVertex2f(2,0);
    glVertex2f(2,2);
    glVertex2f(1,3);
    glVertex2f(0,2);
    glEnd();
}
```

Transformations in OpenGL

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

Transformations in OpenGL

```c
void DrawHouse()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(3,1,0);
    glScale(2,2,2);
    DrawHouse();
}
```

Composing Transformations

```c
update current transformation matrix
by postmultiplying
```

Post Multiplication

- Composite transformation = matrix product
- Rather than multiply each point sequentially with 3 matrices, first multiply the matrices, then multiply each point with only one (composite) matrix
  - Much faster for large # of points!
  - Same reason to use homogeneous coordinates
Interpreting Composite OpenGL Transformations

- Example from earlier lectures:
  - Rotation around arbitrary center
- In OpenGL:

```
// initialization of matrix
glMatrixMode( GL_MODELVIEW );
gLoadIdentity();

// translation
glTranslatef( 4, 3 );
gBegin( GL_TRIANGLES );

// rotation
glRotatef( 30, 0.0, 0.0, 1.0 );

gBegin( GL_TRIANGLES );

// translation
glTranslatef( -4, -3 );
```

Top-to-bottom: transf. of coordinate frame
Bottom-to-top: transf. of object

Matrix Operations in OpenGL

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

```
gMatrixMode( GL_MODELVIEW );
gLoadIdentity(); // M=Id
gRotatef( angle, x, y, z ); // M= R(\alpha) * Id
gTranslate( x, y, z ); // M= T(x,y,z) * R(\alpha) * Id
gMatrixMode( GL_PROJECTION );
gRotatef( ... ); // P= ...```

Transformation Hierarchies

- Scenes have multiple coordinate systems
- Often strongly related
  - Parts of the body
  - Object on top of each other
    - Next to each other...
- Independent definition is bug prone
- Solution: Transformation Hierarchies

Transformation Hierarchy Examples

```
gTranslate3f(x,y,0);
gRotatef( \theta ,0,0,1);
DrawBody();
gTranslate2(2.5,5.5,0);
gRotatef( \theta ,0,0,1);
DrawUArm();
gTranslate(0,-3.5,0);
gRotatef( \theta ,0,0,1);
DrawLArm();
```

Matrix Stacks

```
gPushMatrix();
gPopMatrix()
gPushMatrix();

\[ D = C \text{ scale}(2,2,2) \text{ trans}(1,0,0) \]
```

Copyright 2013, A. Sheffer, UBC
Transformation Hierarchy Examples

Transformation Hierarchy Examples

Transformation Hierarchies

Matrix Stacks

Hierarchical Modeling

Transforming Normals

Copyright 2013, A. Sheffer, UBC
Computer Graphics

Transformations 2

Computing Normals
- polygon: 
  \[ N = \frac{(P_2 - P_1) \times (P_3 - P_1)}{\|P_2 - P_1\| \times \|P_3 - P_1\|} \]
  - assume vertices ordered CCW when viewed from visible side of polygon
  - normal for a vertex
  - used for lighting
  - supplied by model (i.e., sphere), or computed from neighboring polygons

Transforming Normals
- When transforming triangle(s) can we use the same transformation to transform the normal & avoid re-computation?
- What is a normal?
  - Vector
    - Orthogonal (perpendicular) to plane/surface
  - Do standard transformations preserve orthogonality?
    - Or angles in general?

Finding Correct Normal Transform
- transform a plane
  \[ \begin{align*}
  P' &= MP \\
  N' &= QN
  \end{align*} \]
  - Given M, find Q
  - \( N^T P' = 0 \)
  - \( (QN)^T (MP) = 0 \)
  - \( N^T Q^T MP = 0 \)
  - \( Q^T M = I \)
  - Normal transformed by transpose of the inverse of the modeling transformation

Planes and Normals
- Plane - all points where \( N \cdot P = 0 \)
  - Implicit form
    \[ \begin{align*}
    P &= \begin{bmatrix} x \\ y \\ z \end{bmatrix},
    N &= \begin{bmatrix} A \\ B \\ C \end{bmatrix},
    \end{align*} \]
    \[ \text{Plane} = A \cdot x + B \cdot y + C \cdot z + D \]

Transformation properties
- Straight lines
- Parallel lines
- Distance
- Angles

Copyright 2013, A. Sheffer, UBC