Transforming Normals

- What is a normal?
  - Vector
    - Orthogonal (perpendicular) to plane/surface
  - Do standard transformations preserve orthogonality?

Computing Normals

- polygon: \( N = (P_2 - P_1) \times (P_3 - P_1) \)
- 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

Planes and Normals

- Plane - all points where \( N \cdot P = 0 \)
- Implicit form
  \[
  P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad N = \begin{bmatrix} A \\ B \\ C \\ 0 \end{bmatrix}
  \]
  
  \[
  Plane = A \cdot x + B \cdot y + C \cdot z + D
  \]

Finding Correct Normal Transform

- transform a plane
  \[
  P^N \quad P^0 = MP
  \]
  Given \( M \), find \( Q \)

  
  \[
  N^T \quad P^N = 0
  \]
  
  \[
  (QN)^T (MP) = 0
  \]
  
  \[
  N^T Q^T MP = 0
  \]
  
  \[
  Q^T M = I
  \]
  
  \[
  Q = (M^{-1})^T
  \]

Normal transformed by transpose of the inverse of the modeling transformation
Transformations in OpenGL

The Rendering Pipeline

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

Modeling Transformation
- Purpose:
  - Map geometry from local object coordinate system into a global world coordinate system
  - Same as placing objects
- Transformations:
  - Arbitrary affine transformations are possible
  - More complex transformations may be desirable
    - Freeform deformations
    - Not available in hardware

Transformations in OpenGL

Copyright 2011, A. Sheffer, UBC
Transformations in OpenGL

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Transformations in OpenGL

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

```
glMatrixMode(GL_MODELVIEW);
gLoadIdentity();
gRotatef(3,1,0);
gScale(2,2,2);
DrawHouse();
```

Composing Transformations

```
P_t = Trans(2,3,0)Rot(z,-90)P_i
```

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
gTranslatef( x, y, z ); // M= T(x,y,z) * R(\alpha) * id
gMatrixMode( GL_PROJECTION );
gRotatef( ... ); // P= ...
```

Copyright 2011, A. Sheffer, UBC
Interpreting Composite OpenGL Transformations

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

```c
// initialization of matrix
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(4, 3);
glRotatef(30, 0.0, 0.0, 1.0);
glTranslatef(-4, -3);
```

Top-to-bottom: transf. of coordinate frame

Bottom-to-top: transf. of object

Transformation Hierarchies

- scene may have a hierarchy of coordinate systems
- Multiple objects, multiple joint links, ...
- stores matrix at each level with incremental transform from parent's coordinate system

Check out: Brown Applets

```
http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/scenegraphs.html
```

Have a look later

Matrix Stacks

- Avoiding unnecessary computation when incremental processing makes no sense
- Using inverse to return to origin costs...

Copyright 2011, A. Sheffer, UBC
Matrix Stacks

gPushMatrix()
gScale(2,2,2)
gTranslate(1,0,0)

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

Matrix Stacks

Advantages
- No need to compute inverse matrices all the time
- Modularize changes to pipeline state
- Avoids incremental changes to coordinate systems
  - Accumulation of numerical errors

Practical issues
- In graphics hardware, depth of matrix stacks is limited
  - Typically 16 for model/view and ~4 for projective matrix

Modularization

- Drawing a scaled square
  - Push/pop ensures no coord system change

void drawBlock(float k) {
  glPushMatrix();
  glScalef(k,k,k);
  glBegin(GL_LINE_LOOP);
  glVertex3f(0,0,0);
  glVertex3f(1,0,0);
  glVertex3f(1,1,0);
  glVertex3f(0,1,0);
  glEnd();
  glPopMatrix();
}

Hierarchical Modeling

Advantages
- Define object once, instantiate multiple copies
- Transformation parameters often good control knobs
- Maintain structural constraints if well-designed

Limitations
- Expressivity: not always the best controls
- Can't do closed kinematic chains
- Keep hand on hip
Assignment 2

- Out last week, due **4pm Fri Oct 14, 2011**
  - Start very soon!
  - Build and animate a giraffe made out of spheres and 4x4 matrices
    - think cartoon, not beauty
  - Template code - program shell, Makefile

Advice

- **Draw one section at a time**
  - Ensure you're constructing hierarchy correctly
  - Use body as scene graph root
  - Continue with attached parts
  - Finish all required parts before...
    - ...Adding extra links or DOFs
    - ...Going for extra credit
  - Visual debugging
    - Draw the current coord system

Advanced transformations example

- Deformation Transfer [Sumner'05]
  - Use transformation gradients (transformation without translation) as per-triangle encoding of motion