Notes
- Drop-box is no. 14  You can hand in your assignments
- Assignment 0 due Fri. 4pm
- Assignment 1 is out
- Office hours today 16:00 - 17:00, in lab or in reading room

Chapter 4 - Reminder
Transformations

Reminder
- Linear transformation - combinations of
  - Shear, scale, rotate, reflect
- Affine transformation - Add translations
  - Closed under composition
- Use homogeneous coordinates to keep in matrix form
- General forms:
  \[
  \begin{pmatrix}
  s_x & 0 & 0 & 0 \\
  0 & s_y & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \begin{pmatrix}
  \cos \alpha & \sin \alpha & 0 & 0 \\
  -\sin \alpha & \cos \alpha & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \begin{pmatrix}
  1 & 0 & t_x & 0 \\
  0 & 1 & t_y & 0 \\
  0 & 0 & 1 & t_z \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \begin{pmatrix}
  v_x \\
  v_y \\
  v_z \\
  v_w
  \end{pmatrix}
  \]

Chapter 5: Transformations - Transforming Normals, Hierarchies and OpenGL, Assignment 1

Clarification
- Why does this matrix transform between frames?
  
  \[
  U = u_x, X + u_y, Y + u_z, Z \\
  V = v_x, X + v_y, Y + v_z, Z \\
  W = w_x, X + w_y, Y + w_z, Z
  \]
  
  \[
  R = \begin{pmatrix}
  u_x & v_x & w_x \\
  u_y & v_y & w_y \\
  u_z & v_z & w_z
  \end{pmatrix}
  \]
  
  \[
  v_{135} = \begin{pmatrix}
  a & \frac{ab}{c} & b & 0 \\
  y & x & z & 1
  \end{pmatrix}
  \rightarrow v = \alpha U + \beta V + \gamma W
  \]
  
  \[
  v_{s135} = (w_x, X + w_y, Y + w_z, Z) = a(w_x, X + w_y, Y + w_z, Z) + b(w_x, X + w_y, Y + w_z, Z) + c(w_x, X + w_y, Y + w_z, Z)
  \]

  \[
  R v_{s135} = \begin{pmatrix}
  a & \beta & \gamma & 0 \\
  w_x & w_y & w_z & 1
  \end{pmatrix}
  \]

  \[
  v_{s135} = \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \]
Transforming Normals

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

Finding Correct Normal Transform

- transform a plane
  \[ P' = MP \]
  \[ N' = QN \]
  Given \( M \), find \( Q \)

  \[ N^T P' = 0 \]
  \[ (QN)^T (MP) = 0 \]
  \[ N^T Q^T MP = 0 \]
  \[ Q^T M = I \]

  \[ Q = (M^{-1})^T \]

  Normal transformed by transpose of the inverse of the modeling transformation

Computing Normals

- polygon:
  \[ N = (P_3 - P_1) \times (P_2 - P_1) \]
- 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 \( N \cdot P = 0 \)

  \[ P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, N = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \]

  Implicit form

  \[ Plane = A \cdot x + B \cdot y + C \cdot z + D \]

Transformations in OpenGL
Transformations in OpenGL

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

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

```c
DrawHouse();
```

Transformations in OpenGL

```c
An easier way to do the same thing....
```n
```c
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(3,1,0);
glScale(2,2,2);
DrawHouse();
```

Matrix Operations in OpenGL

```c
2 Matrices:
- Model/view matrix M
- Projective matrix P

Example:
```n
```c
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);
```

Composing Transformations

```c
suppose we want
```

```c
```

```c
```

```c
```

Composing Transformations

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

```c
```

```c
```

```c
```

```c
```

```c
Updates current transformation matrix by postmultiplying
```
Composing Transformations

- \( \text{Rotate}(z,-90) \)
- \( \text{Translate}(-3,2,0) \) in local coords

\[ P_w = \text{Rot}(z,-90)\text{Trans}(-3,2,0)P \]

\[ \text{glRotatef}(-90,0,0,1); \]
\[ \text{glTranslatef}(-3,2,0); \]
\[ \text{draw\_house}(); \]

Rotation About a Point: Moving Object

- rotate about \( p \) by \( \theta \):
- translate \( p \) to origin
- rotate about origin
- translate \( p \) back

\[ T(x,y,z)R(z,\theta)T(-x,-y,-z) \]

Rotation: Changing Coordinate Systems

- same example: rotation around arbitrary center

Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 1: translate coordinate system to rotation center

Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 2: perform rotation

Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 3: back to original coordinate system
**General Transform Composition**
- transformation of geometry into coordinate system where operation becomes simpler
  - typically translate to origin
- perform operation
- transform geometry back to original coordinate system

**Rotation About an Arbitrary Axis**
- 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

**Transformation Hierarchies**
- scene may have a hierarchy of coordinate systems
  - stores matrix at each level with incremental transform from parent's coordinate system

**Demo: Brown Applets**
Composing Transformations

- OpenGL example

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

Transformation Hierarchies

- Matrix Stack

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

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

Assignment 1

- Out today, due 4pm Fri Oct 15, 2005
- Start very soon!
- Build dinosaur out of spheres and 4x4 matrices
  - Think cartoon, not beauty
- Template code - program shell, Makefile
  
```bash
http://www.ugrad.cs.ubc.ca/~cs314/Vsep2005/a1/a1.tar.gz
```
Dinosaurs

Articulated Dino

Articulated Dino

Demo

- Maybe in a couple weeks – Ask prof.
- Can view last years demos of dogs and birds

Advice

- Build then animate 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
  - Going for extra credit
  - Playing with lighting or viewing

More Advice

- OK to use glRotate, glTranslate, glScale
- OK to use glutSolidSphere, or build your own
  - where to put origin? your choice
    - center of object, range -0.5 to +0.5
    - corner of object, range 0 to 1
More Advice

- Visual debugging
  - Color sphere faces differently
  - Draw the current coord system
- Transformations - intuition
  - move physical objects around
  - play with demos
  - Brown scenegraph applets

More Advice

- Transitions
  - safe to linearly interpolate parameters for glRotate/glTranslate/glScale
  - do not interpolate individual elements of 4x4 matrix!