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

University of British Columbia

University of British Columbia

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:

$$\begin{pmatrix} s_x & & & \\ & s_y & & \\ & & s_z & \\ & & & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & & \sin\alpha & \\ & 1 & & \\ -\sin\alpha & & \cos\alpha & \\ & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & t_x \\ & 1 & & t_y \\ & & 1 & t_z \\ & & & 1 \end{pmatrix}$$

University of British Columbia

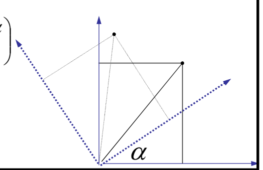
Clarification

- Why is this a rotation matrix? $R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$

$$Rv \bullet v = \begin{pmatrix} v_x \cos\alpha - v_y \sin\alpha \\ v_x \sin\alpha + v_y \cos\alpha \end{pmatrix} \bullet \begin{pmatrix} v_x \\ v_y \end{pmatrix} = v_x^2 \cos\alpha + v_y^2 \cos\alpha = \cos\alpha \|v\|^2$$

$$\forall v \in R^2 \ \exists a,b \text{ s.t. } v = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$Rv = aR\begin{pmatrix} 1 \\ 0 \end{pmatrix} + bR\begin{pmatrix} 0 \\ 1 \end{pmatrix} = a\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} + b\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}$$



University of British Columbia

Clarification

- Why does this matrix transform between frames?

$$U = u_x X + u_y Y + u_z Z$$
$$V = v_x X + v_y Y + v_z Z$$
$$W = w_x X + w_y Y + w_z Z$$

$$R = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

$$v_{UVW} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \rightarrow v = \alpha U + \beta V + \gamma W =$$

$$= \alpha(u_x X + u_y Y + u_z Z) + \beta(v_x X + v_y Y + v_z Z) + \gamma(w_x X + w_y Y + w_z Z) =$$

$$= (\alpha u_x + \beta v_x + \gamma w_x)X + (\alpha u_y + \beta v_y + \gamma w_y)Y + (\alpha u_z + \beta v_z + \gamma w_z)Z = v_{XYZ}$$

$$Rv_{UVW} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \alpha u_x + \beta v_x + \gamma w_x \\ \alpha u_y + \beta v_y + \gamma w_y \\ \alpha u_z + \beta v_z + \gamma w_z \end{pmatrix}$$

University of British Columbia

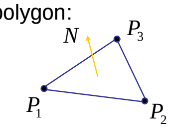
University of British Columbia

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

Transforming Normals

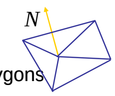
Computing Normals

- 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



Transforming Normals

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

Planes and Normals

- Plane - all points where $N \cdot P = 0$

$$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, N = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$$

- Implicit form

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

Finding Correct Normal Transform

- transform a plane

$$\begin{matrix} P & & P' = MP \\ N & \longrightarrow & N' = QN \end{matrix}$$

Given M, find Q

$N'^T P' = 0$ **stay perpendicular**

$(QN)^T (MP) = 0$ **substitute from above**

$N^T \underbrace{Q^T M}P = 0$ $\quad (\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$

$Q^T M = I$ $\quad N^T P = 0$

$$\boxed{Q = \left(M^{-1}\right)^T}$$

Normal transformed by *transpose* of *the inverse* of the modeling transformation

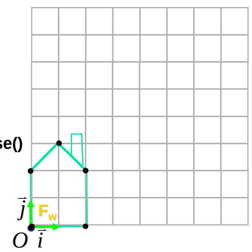
Transformations in OpenGL

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()



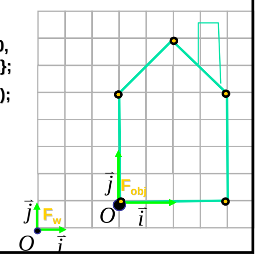
Transformations in OpenGL

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

```
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();
```



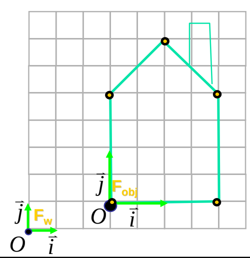
Transformations in OpenGL

- An easier way to do the same thing….

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

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

DrawHouse();
```



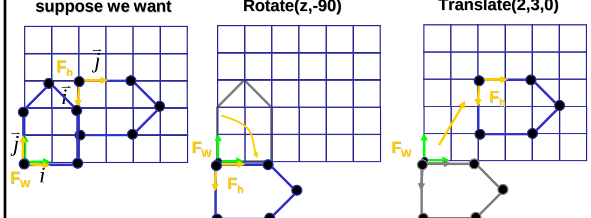
Matrix Operations in OpenGL

- 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= …
```


Composing Transformations

suppose we want | Rotate(z,-90) | Translate(2,3,0)



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

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

Composing Transformations

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

- R-to-L: interpret operations wrt fixed coords
  - moving object
- L-to-R: interpret operations wrt local coords
  - changing coordinate system
- OpenGL (L-to-R, local coords)

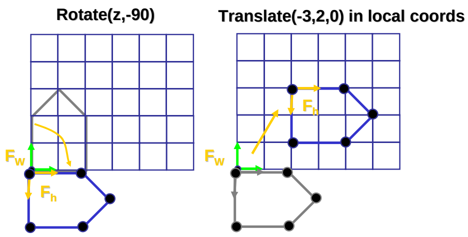
glTranslatef(2,3,0);
glRotatef(-90,0,0,1);
DrawHouse();

$M_{MV} = Trans(2,3,0) \cdot M_{MV}$

$M_{MV} = Rot(z,-90) M_{MV}$

updates current transformation matrix by postmultiplying

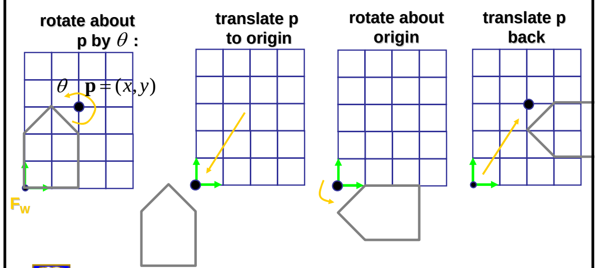
Composing Transformations



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

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

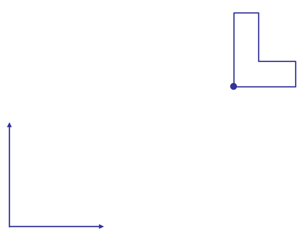

Rotation About a Point: Moving Object



$$\mathbf{T}(x,y,z)\mathbf{R}(z,\theta)\mathbf{T}(-x,-y,-z)$$

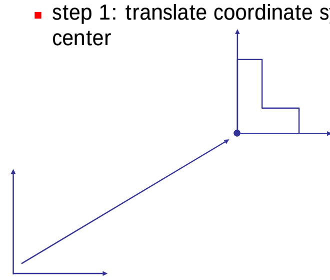
Rotation: Changing Coordinate Systems

- same example: rotation around arbitrary center



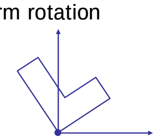
Rotation: Changing Coordinate Systems

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



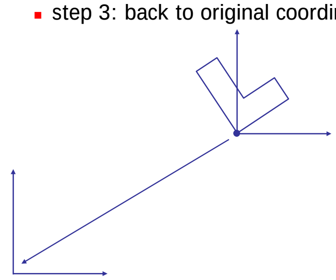
Rotation: Changing Coordinate Systems

- 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

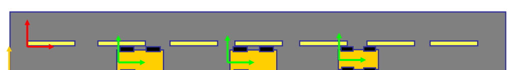
University of British Columbia

University of British Columbia

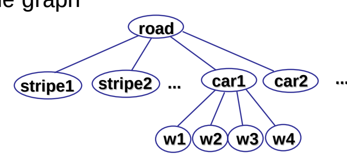
Transformation Hierarchies

Transformation Hierarchies

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

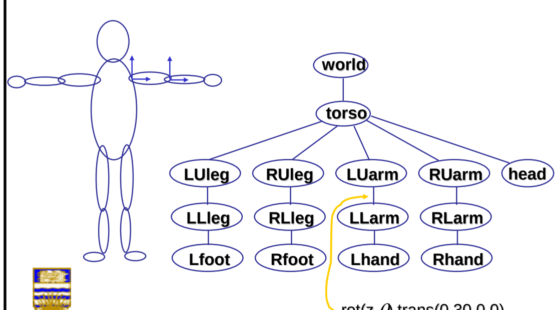


- scene graph



University of British Columbia

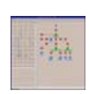
Transformation Hierarchies



University of British Columbia

Demo: Brown Applets

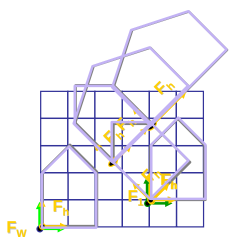
http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/scenegraphs.html



University of British Columbia

Composing Transformations

- OpenGL example

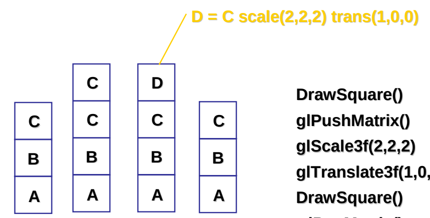


```
glLoadIdentity();
glTranslatef(4,1,0);
glPushMatrix();
glRotatef(45,0,0,1);
glTranslatef(0,2,0);
glScalef(2,1,1);
glTranslate(1,0,0);
glPopMatrix();
```

University of British Columbia

Transformation Hierarchies

- Matrix Stack




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

University of British Columbia

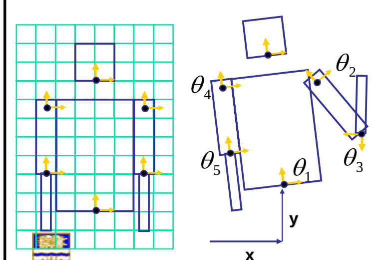
Matrix Stacks

- 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

- 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

University of British Columbia

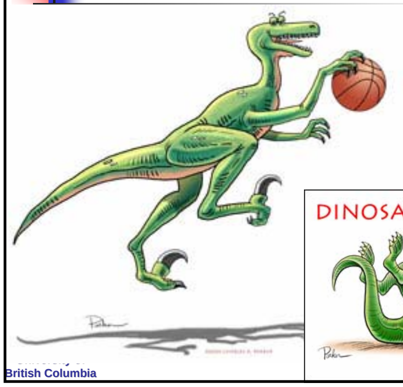
Assignment 1

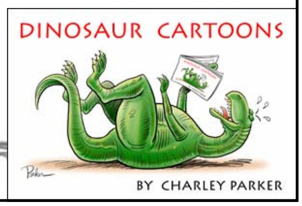
Assignment 1

- 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
    - http://www.ugrad.cs.ubc.ca/~cs314/Vsep2005/a1/a1.tar.gz

University of British Columbia

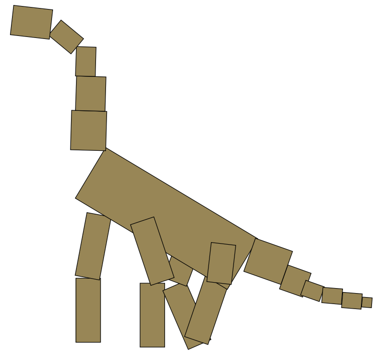
Dinosaurs





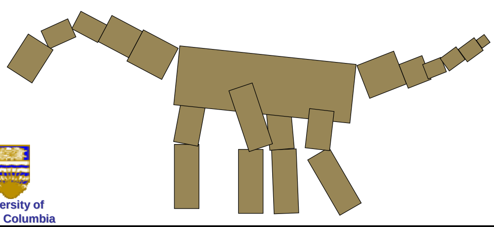
University of British Columbia

Articulated Dino



University of British Columbia

Articulated Dino



University of British Columbia

Demo

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

University of British Columbia

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

University of British Columbia

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 - .5 to +.5
    - corner of object, range 0 to 1

University of British Columbia

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

University of British Columbia

More Advice

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

University of British Columbia