Week 3, Mon Jan 21

University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2008

Tamara Munzner

Transformations III

Readings for Jan 16-25
- FCG Chap 6 Transformation Matrices
- except 6.16, 6.3.1
- FCG Sect 13.3 Scene Graphs
- RB Chap Viewing
- Viewing and Modeling Transforms until Viewing Transformations
- Examples of Composing Several Transformations through Building an Articulated Robot Arm
- RB Appendix Homogeneous Coordinates and Transformation Matrices
- until Perspective Projection
- RB Chap Display Lists

News
- Homework 1 out today

Review: Composing Transformations
Ta Ta Rb Rb != != Rb Rb Ta Ta

Matrix Composition
- matrices are convenient, efficient way to represent series of transformations
  - general purpose representation
  - hardware matrix multiply
  - matrix multiplication is associative
  - p' = (T*R*S)p
  - procedure
  - correctly order your matrices!
  - multiply matrices together
  - result is one matrix, multiply vertices by this matrix
  - all vertices easily transformed with one matrix multiply

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
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

Matrix Stacks

- challenges of avoiding unnecessary computation
  - using inverse to return to origin
  - computing incremental \( T_1 \rightarrow T_2 \)
- advantages
  - no need to compute inverse matrices all the time
  - modularity 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 about 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);
    glTranslatef(GL_LINES_LOOP);  
    glScalef(0, 1, 0);  
    glTranslatef(0, 1, 0);  
    glTranslatef(0, 1, 0);  
    glPopMatrix();
  }  

Hierarchical Modelling

- 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
  - can’t do other constraints
  - collision detection
    - self-intersection
      - walk through walls