Transformations 4

Readings for Transformations 1-5

- Shirley/Marschner
  - Ch 6: Transformation Matrices
    - except 6.1.6, 6.3.1
  - Sect 12.2 Scene Graphs

- Gortler
  - Ch 2: Linear, Sec 2.5-2.6
  - Ch 3: Affine
  - Ch 4: Respect
  - Ch 5: Frames in Graphics, 5.3-5.4
Correction: Composing Transformations

\[ p' = TRp \]

- which direction to read?
  - right to left
    - interpret operations \textit{wrt} fixed global coordinates
  - moving object
    - draw thing
    - rotate thing by 45 degrees \textit{wrt} fixed global coords
    - translate it \((2, 3)\) over \textit{wrt} fixed global coordinates
Correction: Composing Transformations

\[ p' = TRp \]

- which direction to read?
  - left to right
    - interpret operations \(\text{wrt}\) local coordinates
  - changing coordinate system
    - translate coordinate system \((2, 3)\) over
    - rotate coordinate system 45 degrees \(\text{wrt}\) LOCAL origin
    - draw object in current coordinate system
Practice: Composing Transformations
Transformation Hierarchies
Rotation About a Point: Moving Object

rotate about $p$ by $\theta$ :

$\theta$  
$p = (x, y)$

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

\[ T(x, y, z) R(z, \theta) T(-x, -y, -z) \]
Rotation: Changing Coordinate Systems

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

\[ T(x, y, z) R(z, \theta) T(-x, -y, -z) \]
Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 2: perform rotation

\[ T(x, y, z)R(z, \theta)T(-x, -y, -z) \]
Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 3: back to original coordinate system

\[ T(x, y, z) R(z, \theta) T(-x, -y, -z) \]
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
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
Transformation Hierarchy Example 1

![Transformation Hierarchy Diagram]

- **world**
  - **torso**
    - **LUleg**
    - **LLLeg**
    - **Lfoot**
    - **RUleg**
    - **RLLeg**
    - **Rfoot**
    - **LUarm**
    - **LLArm**
    - **Lhand**
    - **RUarm**
    - **RLArm**
    - **Rhand**
    - **head**

Transformation: \( \text{trans}(0.30,0,0) \) \( \text{rot}(z, \theta) \)
Transformation Hierarchy Example 2

- draw same 3D data with different transformations: instancing
Transformation Hierarchies Demo

• transforms apply to graph nodes beneath

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

- challenge of avoiding unnecessary computation
  - using inverse to return to origin
  - computing incremental $T_1 \rightarrow T_2$

$T_1(x)$  $T_2(x)$  $T_3(x)$

Object coordinates

World coordinates
Matrix Stacks

pushMatrix()

popMatrix()

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

drawSquare()

pushMatrix()

scale(2,2,2)

translate(1,0,0)

drawSquare()

popMatrix()
Modularization

• drawing a scaled square
  • push/pop ensures no coord system change

```c
void drawBlock(float k) {
    pushMatrix();
    scale(k,k,k);
    drawBox();
    popMatrix();
}
```
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
Transformation Hierarchy Example 3

loadIdentity();
translate(4,1,0);
pushMatrix();
rotate(45,0,0,1);
translate(0,2,0);
scale(2,1,1);
translate(1,0,0);
popMatrix();
Transformation Hierarchy Example 4

```plaintext
translate(x,y,0);
rotate(θ,0,0,1);
DrawBody();
pushMatrix();
translate(0,7,0);
DrawHead();
popMatrix();
pushMatrix();
translate(2.5,5.5,0);
rotate(θ,0,0,1);
DrawUArm();
translate(0,-3.5,0);
rotate(θ,0,0,1);
DrawLArm();
popMatrix();
... (draw other arm)
```
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
Arbitrary Rotation

- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems \( XYZ \) and \( ABC \)
    - \( A \)'s location in the \( XYZ \) coordinate system is \((a_x, a_y, a_z, 1), \ldots\)
Arbitrary Rotation

- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems $XYZ$ and $ABC$
  - $A$’s location in the $XYZ$ coordinate system is $(a_x, a_y, a_z, 1)$, ...
Arbitrary Rotation

- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems $XYZ$ and $ABC$
    - $A$’s location in the $XYZ$ coordinate system is $(a_x, a_y, a_z, 1)$, ...
  - transformation from one to the other is matrix $R$ whose columns are $A,B,C$:

$$R(X) = \begin{bmatrix}
a_x & b_x & c_x & 0 \\
a_y & b_y & c_y & 0 \\
a_z & b_z & c_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} = (a_x, a_y, a_z, 1) = A$$