15-462: Computer Graphics

Math for Computer Graphics
Transformations

- Translation, rotation, scaling
  - 2D
  - 3D
- Homogeneous coordinates
- Transforming normals
- Examples

Shirley Chapter 6
Uses of Transformations

• Modeling
  – build complex models by positioning simple components
  – transform from object coordinates to world coordinates
• Viewing
  – placing the virtual camera in the world
  – specifying transformation from world coordinates to camera coordinates
• Animation
  – vary transformations over time to create motion
Rigid Body Transformations

Rotation angle and line about which to rotate
Non-rigid Body Transformations

Distance between points on object do not remain constant
Basic 2D Transformations

Scale

Shear

Rotate

chalkboard
Composition of Transformations

- Created by stringing basic ones together, e.g.
  - “translate \( p \) to the origin, rotate, then translate back”

  can also be described as a rotation about \( p \)

- Any sequence of linear transformations can be collapsed into a single matrix formed by multiplying the individual matrices together

- Order matters!

- Can apply a whole sequence of transformations at once

*Translate to the origin, rotate, then translate back.*
3D Transformations

- 3-D transformations are very similar to the 2-D case
- Scale
- Shear
- Rotation is a bit more complicated in 3-D
  - different rotation axes
But what about translation?

- Translation is not linear--how to represent as a matrix?

chalkboard
But what about translation?

- Translation is not linear--how to represent as a matrix?

- Trick: add extra coordinate to each vector
- This extra coordinate is the *homogeneous* coordinate, or $w$
- When extra coordinate is used, vector is said to be represented in *homogeneous coordinates*
- We call these matrices *Homogeneous Transformations*
W? Where did that come from?

• Practical answer:
  - W is a clever algebraic trick.
  - Don’t worry about it too much. The w value will be 1.0 for the time being (until we get to perspective viewing transformations)

• More complete answer:
  - (x,y,w) coordinates form a 3D projective space.
  - All nonzero scalar multiples of (x,y,1) form an equivalence class of points that project to the same 2D Cartesian point (x,y).
  - For 3-D graphics, the 4D projective space point (x,y,z,w) maps to the 3D point (x,y,z) in the same way.
Homogeneous 2D Transformations

The basic 2D transformations become

\[
\text{Translate: } \quad \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\quad \text{Scale: } \quad \begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad \text{Rotate: } \quad \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Now any sequence of translate/scale/rotate operations can be combined into a single homogeneous matrix by multiplication.

3D transforms are modified similarly
Rigid Body Transformations

(a)  (b)

Rotation angle and line about which to rotate
Rigid Body Transformations

• A transformation matrix of the form

\[
\begin{bmatrix}
  x_x & x_y & t_x \\
  y_x & y_y & t_y \\
  0 & 0 & 1
\end{bmatrix}
\]

where the upper 2x2 submatrix is a rotation matrix and column 3 is a translation vector, is a rigid body transformation.

• Any series of rotations and translations results in a rotation and translation of this form (and no change in the distance between vertices)
Sequences of Transformations

Often the same transformations are applied to many points

Calculation time for the matrices and combination is negligible compared to that of transforming the points

Reduce the sequence to a single matrix, then transform
Collapsing a Chain of Matrices.

- Consider the composite function $ABCD$, i.e. $p' = ABCDp$
- Matrix multiplication isn’t commutative - the order is important
- But matrix multiplication is associative, so can calculate from right to left or left to right: $ABCD = (((AB) C) D) = (A (B (CD)))$
- Iteratively replace either the leading or the trailing pair by its product

<table>
<thead>
<tr>
<th>Premultiply</th>
<th>Postmultiply</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M \leftarrow D$</td>
<td>$M \leftarrow A$</td>
</tr>
<tr>
<td>$M \leftarrow CM$</td>
<td>$M \leftarrow MB$</td>
</tr>
<tr>
<td>$M \leftarrow BM$</td>
<td>$M \leftarrow MC$</td>
</tr>
<tr>
<td>$M \leftarrow AM$</td>
<td>$M \leftarrow MD$</td>
</tr>
</tbody>
</table>

both give the same result.
What is a Normal? – refresher

Indication of outward facing direction for lighting and shading

Order of definition of vertices in OpenGL

Right hand rule
Transforming Normals

- It’s tempting to think of normal vectors as being like porcupine quills, so they would transform like points.
- Alas, it’s not so.
- We need a different rule to transform normals.

Original Shape shown with normals

Transformed shape with normals treated as points

Transformed shape with correct normals
Examples

- Modeling with primitive shapes
Practice Problems

The slide on page 4 of this slide deck / pdf file shows three rigid body transformations. Refer to this slide for the problems below.

Give a transformation matrix for rigid body translation by vector $\mathbf{d}$ as shown in the first figure on page 4.

Give a transformation matrix (or sequence of matrices) for accomplishing the transformation shown in the second figure on page 4. This transformation results in a rotation of 45 degrees about the object center point $\mathbf{p}_f$. 
Give a set of OpenGL calls or pseudocode that may result in the transformation shown in the third figure on page 4. This is a rotation of angle $u$ about the line in direction $v$ passing through point $p_f$. You may assume you have a function call for rotation of an angle about a given axis. If you do this problem as multiple operations, be clear and careful of the ordering of these operations.

Practice constructing shear matrices to achieve desired effects.