Computer Graphics

Given a model of the world, what does the model look like from a particular point of view?
  • Add interactivity to build creative tools and viewer controlled animations
  • Add motion to agents in the model to get more traditional animations

Models in graphics include both geometric models and appearance models.
  • We want to be able to build and manipulate geometric models quickly and easily
  • Linear algebra gives us a set of tools with which we can do that

Transformations

Things we want to do with models in graphics
  • Translate them
  • Scale them
  • Rotate them
  • Smoosh, stretch, shear, and reflect them

If we want to translate an object around the screen (2D) do we...
  • A) change the coordinates by hand,
  • B) Add an offset to the current coordinates
  \[
  \begin{bmatrix}
  x_{\text{new}} \\
  y_{\text{new}}
  \end{bmatrix} = \begin{bmatrix}
  x_{\text{old}} \\
  y_{\text{old}}
  \end{bmatrix} + \begin{bmatrix}
  t_x \\
  t_y
  \end{bmatrix}
  \]
  • C) Develop a general system of representing and applying translations?

If we want to rotate an object around the screen do we...
  • A) change the coordinates by hand,
  • B) Transform the points along the edge of a circle
  \[
  \begin{bmatrix}
  x_{\text{new}} \\
  y_{\text{new}}
  \end{bmatrix} = \begin{bmatrix}
  x_{\text{old}} \cos \Theta - y_{\text{old}} \sin \Theta \\
  y_{\text{old}} \cos \Theta + x_{\text{old}} \sin \Theta
  \end{bmatrix}
  \]
  • C) Develop a general system of representing and applying rotations?

If we want to scale an object on the screen do we...
  • A) change the coordinates by hand,
  • B) Multiply each point by a scalar value
  \[
  \begin{bmatrix}
  x_{\text{new}} \\
  y_{\text{new}}
  \end{bmatrix} = \begin{bmatrix}
  x_{\text{old}} k_x \\
  y_{\text{old}} k_y
  \end{bmatrix}
  \]
  • C) Develop a general system of representing and apply scales?

If you said C on all three, you are correct.

Matrix Representations of:

Homogeneous Coordinates: \( x = x_h / h, \ y = y_h / h, \) let \( h = 1 \) for convenience.
  • Translations
• Rotations

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

• Scales

\[
\begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Concatenation of transformations

With column vectors, the procedure moves from right to left, so the right-most matrix is enacted first.

• Concatenating translations, order doesn’t matter (add translations).
• Concatenating 2-D rotations, order doesn’t matter (add rotations).
• Concatenating scales, order doesn’t matter (multiply scale factors).
• Concatenating arbitrary transformations, order matters.

Example: Take a unit square (0,0) to (1,1), translate it so the center is at the origin T(-.5, -.5), then scale it by 2 in all directions S(2,2), then rotate it by 45 degrees R_z(45), then translate it so the center is at (3,2), T(3.5, 2.5).

The single matrix is: X = T(3.5, 2.5) R_z(45) S(2,2) T(-.5, -.5)

General pivot point rotations

• Translate to origin (-x_0, -y_0)
• Rotate about the origin
• Translate back to the original position (x_0, y_0)

We can precalculate this matrix as well:

\[
\begin{bmatrix}
1 & 0 & x_0 \\
0 & 1 & y_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & -x_0 \\
0 & 1 & -y_0 \\
0 & 0 & 1
\end{bmatrix}
= 
\begin{bmatrix}
\cos \theta & -\sin \theta & x_0(1 - \cos \theta) + y_0 \sin \theta \\
\sin \theta & \cos \theta & y_0(1 - \cos \theta) + x_0 \sin \theta \\
0 & 0 & 1
\end{bmatrix}
\]

General fixed-point scaling

Same idea as arbitrary point rotations.

• Translate the fixed scale point to the origin
• Scale about the origin
• Translate back to the fixed point
• We can precalculate these as well
\[
\begin{bmatrix}
1 & 0 & x_0 \\
0 & 1 & y_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & -x_0 \\
0 & 1 & -y_0 \\
0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
s_x & 0 & x_0(1 - s_x) \\
0 & s_y & y_0(1 - s_y) \\
0 & 0 & 1
\end{bmatrix}
\]

General scaling directions and fixed points
Concatenate translations, rotations, and scales
• Translate the fixed point to the origin
• Rotate to the desired scaling axes
• Scale about the origin
• Rotate back to the original axes
• Translate back to the fixed point

Reflection
• Reflect about y-axis, x-axis, origin, line y =x, or line y = -x.
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad\text{or}\quad
\begin{bmatrix}
-1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad\text{or}\quad
\begin{bmatrix}
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad\text{or}\quad
\begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad\text{or}\quad
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Shear
A shear moves different parts of the coordinate system differently.
\[
\begin{bmatrix}
1 & sh_x & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

3-D Transformations
• 4x4 matrices
• 4 element homogeneous coordinates
• 3-D Translation: T(x, y, z)
• 3-D Scale: S(sx, sy, sz)
• 3-D Rotation
  • Z-axis
  • X-axis
  • Y-axis (flip sine term)
• Scale about arbitrary axes
  • Use u, v, w representation of the new axes, p is the new origin
  • Translate p to the origin
  • If u, v, and w are orthogonal unit vectors, then set up the rotation matrix using u as the top row, v and the second row, and w as the third row
• Scale
• Rotate back by the inverse of the previous rotation matrix (transpose)
• Translate back to the origin
• Orienting two coordinate systems
  • if you have an orthogonal coordinate system M defined by three unit vectors: u, v, w you can write, by inspection, the transformation that will bring M in line with the coordinate system axes in which u, v, w are defined.

\[
\begin{bmatrix}
u_x & u_y & u_z & 0 \\
v_x & v_y & v_z & 0 \\
w_x & w_y & w_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Hierarchical Modeling Systems

What really makes this powerful is setting it up as a hierarchical modeling system.
  • Use different coordinate systems for defining different stuff

**Modeling coordinates:** whatever is appropriate for developing the model  
**World coordinates:** where stuff lives in the world  
**Viewing coordinates:** transformation based on how we want to look at the world  
**Normalized viewing coordinates:** device independent coordinates  
**Screen coordinates:** where in the image does a particular graphic primitive go  

We can also make use of hierarchies within the modeling coordinates

Example: 2-D Planar Robot Arm

Hand structure
  • draw line from (0,1) to (3,0)  
  • draw line from (3,0) to (0,-1)  
  • draw line from (0,-1) to (0,1)

Forearm structure
  • draw line from (0,0) to (10,0)  
  • rotate by end-effector angle
By changing the joint angles, the robot appears to move in a physically correct manner.

If we figure out the physical equations for the joint angles based on external/internal forces, we can animate the figure in a physically realistic manner.