3D Transformations (Java3D)

10-21-05
2D Transformations

- Rotations in 2D can be performed using the 2x2 rotation matrix \( M \):
  
  \[
  \begin{bmatrix}
  \cos t, & \sin t \\
  -\sin t, & \cos t
  \end{bmatrix}
  \]

- To rotate a point \( p = (x, y) \) by \( t \), use \( p' = Mp \):

  \[
  \begin{bmatrix}
  \cos t, & \sin t \\
  -\sin t, & \cos t
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]

  \[
  x' = x \cos t + y \sin t \\
  y' = -x \sin t + y \cos t
  \]

- This can be derived directly from trigonometric identities.
Scaling can be performed similarly:

\[
\begin{bmatrix}
 sx & 0 & x \\
 0 & sy & y \\
 0 & 0 & 1
\end{bmatrix}
\]

Translation can be performed using a 3x3 matrix:

\[
\begin{bmatrix}
 1 & 0 & tx \\
 0 & 1 & ty \\
 0 & 0 & 1
\end{bmatrix}
\]
2D Transformations (cont.)

- Rotation, scaling, and translation can be performed using a single 3x3 matrix:
  \[
  \begin{bmatrix}
  sx\cos t & \sin t & tx \\
  -\sin t & sy\cos t & ty \\
  0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y \\
  1
  \end{bmatrix}
  \]

- The third vector coordinate is called the homogenous (w) coordinate.

- Points use a homogeneous coordinate of 1.0, vectors use a homogeneous coordinate of 0.0.
3D Transformations

- 3D Scaling and translation can be performed similarly using 4x4 matrices:

\[
\begin{bmatrix}
sx & 0 & 0 & tx \\
0 & sy & 0 & ty \\
0 & 0 & sz & tz \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

- 3D Rotations are a little trickier...
3D Rotations

- Rotations can be performed around the x, y, or z axis (or any arbitrary axis for that matter).
- The x-axis rotation matrix:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos t & \sin t & 0 \\
0 & -\sin t & \cos t & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
3D Rotations (cont.)

- The y-axis rotation matrix:
  
  $\begin{bmatrix}
  \cos t, & 0, & -\sin t, & 0 \\
  0, & 1, & 0, & 0 \\
  \sin t, & 0, & \cos t, & 0 \\
  0, & 0, & 0, & 1 
  \end{bmatrix}$

- The z-axis rotation matrix:
  
  $\begin{bmatrix}
  \cos t, & \sin t, & 0, & 0 \\
  -\sin t, & \cos t, & 0, & 0 \\
  0, & 0, & 1, & 0 \\
  0, & 0, & 0, & 1 
  \end{bmatrix}$
Java3D Transformations

- 3D Transformations are handled by the Transform3D object.
  - Set the matrix to the identity using the identity method.
  - Set the matrix to a x-axis rotation matrix using the rotX method.
  - Set the matrix to a y-axis rotation matrix using the rotY method.
  - Set the matrix to a z-axis rotation matrix using the rotZ method.
  - Set the scale of a matrix using the setScale methods.
  - Set the translational component of a matrix using the setTranslation methods.
Matrix Concatenation

- Matrices can be multiplied to chain transformations.
  - Order is important: $m_1 \cdot m_2 = m_3$, $m_3 \cdot p$ is equivalent to $m_2 \cdot p$ followed by $m_1 \cdot p$ (or $m_1 \cdot (m_2 \cdot p)$)
  - Transform3D: multiply matrices using the mul methods.

- Matrix multiplications are implicit in the Scene Graph paradigm (using TransformGroup nodes in Java3D).

- Rotation about Euler angles is common, but may result in Gimbal Lock.
Arbitrary Rotations

- An arbitrary 3D rotation can be expressed using 4 coordinates:
  - 3 coordinates to represent the rotational axis \((x, y, z)\).
  - 1 coordinate to represent the magnitude of the rotation \((w)\).
- Java3D provides an AxisAngle class.
- Transform3D: set the matrix to an arbitrary rotation using the set method.
Quaternions

- Quaternions provide an alternate representation for arbitrary rotations.
- Quaternions are represented using 4 coordinates (like axis angles).
- Mathematical conversions between quaternions, axis angles, and 3x3 matrices are well defined.
Quaternions (cont.)

- Matrices and Quaternions are well defined under multiplication, axis angles are not.
- Matrix multiplications are expensive \( (n^3) \), Quaternion multiplication is less expensive \( (n^2) \).
- Quaternions also obviate Gimbal Lock by providing Spherical Linear Interpolation (SLERP).