Transformations

Geometry vs. Topology

Geometry: Where things are (e.g., coordinates)
Topology: How things are connected

Geometry = same
Topology = changed (1-2-4-3-1)

Geometry = changed
Topology = same (1-2-3-4-1)

Original Object

3D Coordinate Systems

Left-Handed
Right-Handed

Transformations

Suppose you have a point P and you want to move it over by 2 units in X.

How would you change P’s coordinates?

\[ P' = (P'_x, P'_y) = (P_x + 2, P_y) \]

This is known as a coordinate transformation
General Form of 3D Linear Transformations

It's called a "Linear Transformation" because all of the coordinates are raised to the 1st power, that is, there are no \( x^2, x^3 \), etc. terms.

\[
x' = A x + B y + C z + D
\]

\[
y' = E x + F y + G z + H
\]

\[
z' = I x + J y + K z + L
\]

Transform the geometry – leave the topology as is

Translation Equations

\[
x' = x + T_x
\]

\[
y' = y + T_y
\]

\[
z' = z + T_z
\]

Scaling About the Origin

\[
x' = x \cdot S_x
\]

\[
y' = y \cdot S_y
\]

\[
z' = z \cdot S_z
\]

2D Rotation About the Origin

\[
x' = x \cos \theta - y \sin \theta
\]

\[
y' = x \sin \theta + y \cos \theta
\]
Linear Equations in Matrix Form

\[
x' = Ax + By + Cz + D
\]
\[
y' = Ex + Fy + Gz + H
\]
\[
z' = Ix + Jy + Kz + L
\]

Matrix Inverse

\[
[M] \cdot [M]^{-1} = [I]
\]
\[
[M] \cdot [M]^{-1} = \text{“Nothing has changed”}
\]

Identity Matrix ([I])

\[
x' = x
\]
\[
y' = y
\]
\[
z' = z
\]

Translation Matrix

\[
x' = Ax + By + Cz + D
\]
\[
y' = Ex + Fy + Gz + H
\]
\[
z' = Ix + Jy + Kz + L
\]

Quick! What is the inverse of this matrix?
Quick! What is the inverse of this matrix?
Compound Transformations

Q: Our rotation matrices only work around the origin. What if we want to rotate about an arbitrary point (A,B)?

A: Use more than one matrix.

Write it

\[
\begin{pmatrix}
    x' \\
y' \\
z'
\end{pmatrix} = \begin{pmatrix} 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix}
    T_{+A,+B} \\
    R_y \\
    T_{-A,-B}
\end{pmatrix} \begin{pmatrix}
    x \\
y \\
z
\end{pmatrix}
\]

Say it

Matrix Multiplication is not Commutative

Matrix Multiplication is Associative

One matrix – the Current Transformation Matrix, or CTM
Can Multiply All Geometry by One Matrix!

\[
\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} M \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
\]

Graphics hardware can do this very quickly!

OpenGL Will Do the Transformation Compounding for You!

```
for( ; ; )
{
    << Turn mouse position into Xrot and Yrot >>
    glLoadIdentity( );
    glTranslatef( A, B, C );
    glRotatef( (GLfloat)Yrot, 0., 1., 0. );
    glRotatef( (GLfloat)Xrot, 1., 0., 0. );
    glScalef( (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale );
    glCallList( BoxList );
}
```

Typically objects are modeled around their own local origin, so the `glTranslate(-A, -B, -C)` step is unnecessary.

The Funky Rotation Matrix for an Arbitrary Axis and Angle

\[
\mathbf{Q} = \hat{A} \times \mathbf{P} = \hat{A} \times (\mathbf{P}_\parallel + \mathbf{P}_\perp) = \hat{A} \times \mathbf{P}_\perp + \hat{A} \times \mathbf{P}_\parallel = \hat{A} \times (\mathbf{P}_\perp + \mathbf{P}_\parallel) = \hat{A} \times \mathbf{P}
\]
The Funky Rotation Matrix for an Arbitrary Axis and Angle

\[ \mathbf{P}' = \mathbf{P}'_\parallel + \mathbf{P}'_\perp \]

\[ \mathbf{P}' = [\hat{\mathbf{A}}(\hat{\mathbf{A}} \cdot \mathbf{P})] + \cos \theta [\mathbf{P} - \hat{\mathbf{A}}(\hat{\mathbf{A}} \cdot \mathbf{P})] + \sin \theta [\hat{\mathbf{A}} \times \mathbf{P}] \]

For this to work, \( \mathbf{A} \) must be a unit vector