Arbitrary 3D rotations slide #8

We wish to rotate about an arbitrary vector a to form a new basis \( \mathbf{u, v, w} \) with \( \mathbf{w} = \mathbf{a} \).

1. Rotate that basis to the world coordinates \( \mathbf{x, y, z} \).
2. Rotate by \( \theta \) about the \( z \) axis.
3. Rotate back to the \( \mathbf{u, v, w} \) basis.

\[
\begin{bmatrix}
    x_u & x_v & x_w \\
    y_u & y_v & y_w \\
    z_u & z_v & z_w
\end{bmatrix}
= \begin{bmatrix}
    \cos \theta & -\sin \theta & 0 \\
    \sin \theta & \cos \theta & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_u & y_u & z_u \\
    x_v & y_v & z_v \\
    x_w & y_w & z_w
\end{bmatrix}
\]

Remember:

\[
R^T = R^{-1}
\]

\( \mathbf{w} \) is aligned with \( \mathbf{a} \).

Where did \( \mathbf{u, v} \) come from?

\[\mathbf{w} = \frac{\mathbf{a}}{||\mathbf{a}||}\]

Addition: \( \mathbf{t} \) that is not co-linear with \( \mathbf{w} \)

\[\mathbf{t} = \mathbf{w} + \text{change smallest magnitude component to } 1\]

\[\mathbf{w} = \left( \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0 \right) \Rightarrow \mathbf{t} = \left( \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 1 \right)\]

\[\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{||\mathbf{t} \times \mathbf{w}||}, \quad \mathbf{v} = \mathbf{w} \times \mathbf{u}\]
Arbitrary 3D rotations: Slide # 8

I wish to rotate about an arbitrary vector a

form a new basis \( u \) \( v \) \( w \) with \( w = a \)

rotate that basis to the world coordinate

\( x, y, z \)

rotate by \( \theta \) about the \( z \) axis

rotate back to the \( u, v, w \) basis

\[
\begin{bmatrix}
x_u & x_v & x_w \\
y_u & y_v & y_w \\
z_u & z_v & z_w
\end{bmatrix}
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_u & y_u & z_u \\
x_v & y_v & z_v \\
x_w & y_w & z_w
\end{bmatrix}
\]

\[R^T_{uvw} \]

remember:

\[R^T = R^{-1}\]

\(w\) is aligned with \(a\)

where did \(v, v\) come from?

\[w = \frac{a}{||a||}\]

\[t = w + \text{change smallest magnitude component to } 1\]

\[u = (x_2, -\frac{x_1}{||x_2||}, 0) \Rightarrow t = (\frac{x_2}{||x_2||}, -\frac{x_1}{||x_2||}, 1)\]

\[a = \frac{tx_w}{||tx_w||} \quad v = a \times u\]
Canonical View Volume

-1 < x < 1
-1 < y < 1
-1 < z < 1

cube is all x, y, z points \( \mathbb{R}^3 \)

screen is \( n_x \times n_y \) pixels

\[ n_x \text{ might not equal } n_y \text{ in general} \]

\[
\begin{bmatrix}
\text{X canonical} \\
\text{Y canonical} \\
1
\end{bmatrix} = \begin{bmatrix}
\frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\
0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\text{X pixel} \\
\text{Y pixel} \\
1
\end{bmatrix}
\]

\[ X_{pixel} = \frac{n_x}{2} x + \frac{n_x-1}{2} \]

\[
\begin{align*}
x = 0 & \Rightarrow \frac{n_x-1}{2} \\
x = -1 & \Rightarrow -n_x + \frac{n_x-1}{2} = \frac{n_x}{2} - \frac{1}{2} \\
x = 1 & \Rightarrow \frac{n_x + n_x-1}{2} = n_x - \frac{1}{2}
\end{align*}
\]

could also carry z along for z buffering
Orthographic Projection

- take 3D line with endpoints \( a \) \& \( b \)
- use matrix \( M \) to take these points
  to \( Ma, Mb \) in the canonical view volume

Viewer is looking along the minus \( z \) axis with
\( y \) axis pointing up (right hand coordinate system)

Needs a transform to take

\[
\begin{align*}
y &= b \Rightarrow y = 1 \\
y &= t \Rightarrow y = 1 \\
x &= l \Rightarrow x = -1 \\
x &= r \Rightarrow x = 1 \\
z &= n \Rightarrow z = 1 \\
z &= f \Rightarrow z = -1
\end{align*}
\]

\( f \) is more negative than \( n \)
(confusing - caused by looking down negative \( z \) axis)

What operation will do this for us?
move + scale \& more intuitive
or
scale + more
Now have points in canonical view volume
Add in generalization of previous equation = canonical view volume

Combine 3 matrices to get

\[
\begin{bmatrix}
X_{\text{pixel}} \\
Y_{\text{pixel}} \\
Z_{\text{canonical}}
\end{bmatrix}
= M_0
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
\]

\( Z \) is now \([-1, 1]\)
ignored for projection
used for \( Z \) buffer.

For each line segment \((a, b)\)
\( p = M_0a; \)
\( q = M_0b; \)
Draw line \((x_p, y_p, x_q, y_q)\)
Arbitrary View Positions

eye position e (center of eye/lens)

screen direction s

view-up vector t (bisection camera/head +
points up)

\[ w = -\frac{s}{\|s\|} \]

\[ u = \frac{t \times w}{\|t \times w\|} \]

\[ v = w \times u \]

remember: cross product is

3D vector \(-1\) to same orig. vector

\[ \|a \times b\| = \|a\| \|b\| \sin \theta \]

What do we need to add to our pipeline?

a conversion from the coordinate system

\( x', y', z' \@ e \Rightarrow (u, v, w) \@ e \)

Both calls this canonical
we'll call it world

more \( e \Rightarrow o + \) align \(uvw \rightarrow xyz\)

\[ M_r = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
compute \( M_r \)
compute \( M_0 \)
\[ M = M_0 M_r \]
for each line segment \((a_i, b_i)\)
\[ p = M a_i \]
\[ q = M b_i \]
draw line \((x_p, y_p, x_q, y_q)\)
would like to add another matrix to our chain

but how to handle the divide by $z$?

$$y = \frac{\xi}{z} y'$$

trick: use that extra coordinate from last class ($\omega$ or $h$)

& let it take on values other than 1

$$\begin{bmatrix}
\xi x \\
\xi y \\
\xi z \\
\xi \eta
\end{bmatrix} \rightarrow \begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} \quad \text{by dividing by } \xi$$

perspective matrix:

$$M_p = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{c_2}{f} & -1
\end{bmatrix}$$

points $m/z = n$ plane are unchanged:

$$\frac{n (n+f)}{n+f} \rightarrow \frac{n+f}{n+f}$$

$z = \frac{z'}{n+f}$ homogenize $n+f$
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ \frac{y}{z+n} \\ \frac{z}{z+n} \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z+n} \\ 1 \end{bmatrix}$$

**Good properties:**

- Pts on $z=n$ plane are unchanged
  - $z=1 \quad x, y, z$ unchanged

- Pts on $z=n$ plane
  - $z$ unchanged, $x, y$ squished appropriately

Both $n + z$ (inside the view volume) are negative so no clips in $x, y$

- Preserves relative $z$ values (can be used for $z$ buffer)

- Map lines $\rightarrow$ lines
- Map planes $\rightarrow$ planes
So we can multiply $M$ by an arbitrary constant (it will get divided out later).

$$M_p = \begin{bmatrix}
  n & 0 & 0 & 0 \\
  0 & n & 0 & 0 \\
  0 & 0 & n + f & -fn \\
  0 & 0 & 1 & 0 \\
\end{bmatrix}$$

A more attractive version of $M_p$.

$M_p^{-1}$ is also important — select in on screen. What is 2D mouse pointing to in 3D space

$$M_p^{-1} = \begin{bmatrix}
  \frac{1}{n} & 0 & 0 & 0 \\
  0 & \frac{1}{n + f} & 0 & 0 \\
  0 & 0 & \frac{1}{n + f} & 0 \\
  0 & 0 & 0 & \frac{1}{n + f} \\
\end{bmatrix}$$

Equivalently,

$$M_p^{-1} = \begin{bmatrix}
  \frac{1}{n} & 0 & 0 & 0 \\
  0 & \frac{1}{n + f} & 0 & 0 \\
  0 & 0 & \frac{1}{n + f} & 0 \\
  0 & 0 & 0 & \frac{1}{n + f} \\
\end{bmatrix}$$
\[ M = M_o \quad M_p \quad M_v \]

\[ \uparrow \quad \uparrow \quad \uparrow \]

- Take one point to origin
- squash + align
- oxw to x'y'z'

- Canonical view volume
- perspective

- Compute \( M_0 \)
- Compute \( M_v \)
- Compute \( M_p \)

\[ M = M_o \quad M_p \quad M_v \]

For each line segment \((a_i, b_i)\) do

\[ p = M a_i \]
\[ q = M b_i \]

drawline \((qx, qy)\)