Object in world coordinates (after modeling transforms)

Transform object from world coordinates to standard camera coordinates

Clip against canonical view frustum

Project using standard camera model

Transform object from world coords to camera coords

Further transform so that frustum is canonical frustum.
Step 1. Translate the camera at VRP to the world origin. Call this $T_1$.

Translation vector is simply negative VRP.

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to change so that the camera location becomes the origin).
Transform object from world coords to camera coords

Step 2. Rotate camera coordinate frame (in w.c.) so that so that \( \mathbf{u} \) is \( x \), \( \mathbf{v} \) is \( y \), and \( \mathbf{n} \) is \( z \). The matrix is ?

(We are changing the coordinate system of the world, which is the same thing mathematically as moving the camera. We want object world coordinates to change so that the camera axis becomes the standard axis—e.g, \( \mathbf{u} \) becomes \((1,0,0)\), \( \mathbf{v} \) becomes \((0,1,0)\) and \( \mathbf{n} \) becomes \((0,0,1)\)).
Step 2. Rotate camera coordinate frame (in w.c.) so that so that $\mathbf{u}$ is $\mathbf{x}$, $\mathbf{v}$ is $\mathbf{y}$, and $\mathbf{n}$ is $\mathbf{z}$. The matrix is:

$$\begin{vmatrix}
\mathbf{u}^T & 0 \\
\mathbf{v}^T & 0 \\
\mathbf{n}^T & 0 \\
0 & 0 & 0 & 1
\end{vmatrix}$$

(why?)
Transform object from world coords to camera coords

\[
\begin{bmatrix}
  u^T & 0 \\
  v^T & 0 \\
  n^T & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  u \\
  0
\end{bmatrix}
= \begin{bmatrix}
  1 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

In the current coords (world shifted so that VPR is at origin): \( u \) maps into the X-axis unit vector \((1,0,0,0)\) which is what we want.

(Similarly, \( v \rightarrow \) Y-axis unit vector, \( n \rightarrow \) Z-axis unit vector)
Object in world coordinates (after modeling transforms)

Transform object from world coordinates to standard camera coordinates

Clip against canonical view frustum

Project using standard camera model

Transform object from world coords to camera coords

Further transform so that frustum is canonical frustum.
Mapping the view frustum to the canonical view frustum
Further transform so that frustum is canonical frustum.

Since we are now in camera coordinates, we will often refer to them as \((x,y,z)\) not \((u,v,n)\).

1. Translate focal point to origin
2. Shear so that central axis of frustum lies along the \(z\) axis
3. Scale \(x, y\) so that faces of frustum lie on conical planes
4. Isotropic scale so that back clipping plane lies at \(z=-1\)
Step 1: Translate focal point (PRP) to origin; call this translation $T_2$. Since we have PRP in camera coordinates (where we now are), the translation vector is simply negative PRP. In particular, in the very common case where PRP is $(0,0,f)$, the translation vector is $(0,0,-f)$.

Window center is now:

\[
\left( \frac{u_{\text{max}} + u_{\text{min}}}{2}, \frac{v_{\text{max}} + v_{\text{min}}}{2}, -f \right)
\]
Step 2: Shear this volume so that the central axis lies on the z-axis. This is a shear, because rectangles on planes $z=\text{constant}$ must stay rectangles. Call this shear $S_1$
Shear $S_1$ takes previous window midpoint

\[
\frac{1}{2}(u_{\text{max}} + u_{\text{min}}), \quad \frac{1}{2}(v_{\text{max}} + v_{\text{min}}), \quad \text{to (0, 0, -f)}
\]

- this means that matrix is
Shear $S_1$ takes previous window midpoint $\begin{pmatrix} \frac{1}{2}(u_{\text{max}} + u_{\text{min}}) \\ \frac{1}{2}(v_{\text{max}} + v_{\text{min}}) \end{pmatrix}$ to $(0, 0, -f)$ - this means that the matrix is:

\[
\begin{pmatrix}
0 & \frac{(u_{\text{min}} + u_{\text{max}})}{2f} \\
1 & \frac{(v_{\text{min}} + v_{\text{max}})}{2f} \\
0 & 1 \\
0 & 0 \\
0 & 0
\end{pmatrix}
\]
3. Scale $x, y$ so that planes are on $z=x$, $z=-x$ and $z=y$ and $z=-y$.
Call this scale $S_{c_1}$

4. Isotropic scale so that far clipping plane is $z=-1$; call this scale $S_{c_2}$
3. Scale x, y so that planes are on z=x, z=-x and z=y and z=-y. Call this scale $S_{c_1}$

Diagram for $S_y$
(ignoring x coordinate)
4. Scale x, y so that planes are on z=x,
z=-x and z=y and z=-y. Call this
scale $S_{c_1}$

\[
\begin{align*}
&\frac{1}{2}(v_{\text{max}} \pm v_{\text{min}}), \quad \text{if} \quad y=-z \\
&k_y \frac{1}{2}(v_{\text{max}} \pm v_{\text{min}}) = f \\
&k_y = \frac{2f}{(v_{\text{max}} \pm v_{\text{min}})} \quad (k_y \text{ is y scale factor})
\end{align*}
\]
\[ \text{Sc}_1 = \begin{bmatrix} 
\frac{2f}{(u_{\text{max}} \square u_{\text{min}})} & 0 & 0 & 0 \\
0 & \frac{2f}{(v_{\text{max}} \square v_{\text{min}})} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 
\end{bmatrix} \]
5. Now isotropic scale so that far clipping plane is $z=-1$; call this scale $S_{c_2}$

Now $y=-z$

$z=-f+B$

Camera plane
(may be on either side of near clipping plane)
5. Now isotropic scale so that far clipping plane is $z=-1$; call this scale $S_{c_2}$

Currently, at far clipping plane, $z=-f+B$

Want a factor $k$ so that $k(-f+B)=-1$

So, $k = -1 / (-f + B) = 1 / (f - B)$

(Note that $B$ is negative, and $k$ is positive)
3D Viewing Pipeline

\[
\begin{pmatrix}
\text{Point in canonical camera coordinates}
\end{pmatrix}
\rightarrow
Sc_2 Sc_1 S_1 T_2 R_1 T_1
\rightarrow
\begin{pmatrix}
\text{Point in world coordinates}
\end{pmatrix}
\]
Object in world coordinates (after modeling transforms)

- Transform object from world coordinates to standard camera coordinates
  - Clip against canonical view frustum
    - Project using standard camera model

Plan A: Clip against canonical frustum (relatively easy—we chose the canonical frustum so that it would be easy!)

Plan B: Be even more clever. Further transform to cube and clip in homogenous coordinates.
Plan A: Clipping against the canonical frustum

2D algorithms are easily extended. For example, for Cohen Sutherland we use the following 6 out codes:

\[
y > -z \quad y < z \quad x > -z \quad x < z \quad z < -1 \quad z > z_{\text{min}}
\]

\[
( z_{\text{min}} = \frac{(f-F)}{(B-f)} )
\]

Recall C.S for segments

Compute out codes for endpoints

While not trivial accept and not trivial reject:

- Clip against a problem edge (one point in, one out)
- Compute out codes again

Return appropriate data structure
Clipping against the canonical frustum

Clipping polygons in 3D against canonical frustum planes is simpler and more efficient than the general case.

Recall the S.C. gives four cases:

- Polygon edge crosses clip plane going from out to in
  - emit crossing, next vertex
- Polygon edge crosses clip plane going from in to out
  - emit crossing
- Polygon edge goes from out to out
  - emit nothing
- Polygon edge goes from in to in
  - emit next vertex

(The above is from before, just change “edge” to “plane”)
Plan B: Clipping in homogenous coords

- For any camera, can turn the view frustrum into a regular parallelepiped (box). We will use the box bounded by $x = \pm 1$, $y = \pm 1$, $z = -1$, and $z = 0$.

- Advantages
  - Simplified clipping in homogenous coordinates
  - Extends to cases where we use homogenous coordinates to represent additional information (and $w$ could be negative).
  - Can simplify visibility algorithms.

- Approach: clever use of homogenous coordinates
Polygon in 3D

Rotate and translate to place camera at origin

Orthographic case

Translate and scale to box

Perspective case

Translate, shear and scale to canonical frustum

Transform frustum to standard box

Divide and clip against box or clip in h.c.’s and then divide

Projection is now trivial
Transforming canonical frustum to box

\[ x = -z \]
\[ y = -z \]
\[ z = d \]

\[ y = 1 \]

\[ z = -1 \]
Transforming canonical frustum to box

$y = -z$

Camera plane

$z = -1$

$-z$
Transforming canonical frustum to box

The picture should suggest an appropriate scaling for y.

It is z = z_{\text{min}}

Camera plane

y = -z

z = -1

\(-z\)
Transforming canonical frustum to box

On top, \( y \to 1 \), so scaling is \( \frac{1}{y} \)
Recall that \( y=-z \) there.

On bottom, \( y \to -1 \) so scaling is \( \frac{-1}{y} \). Recall that \( y=z \) there.

So scaling is \( y' = \frac{y}{-z} \)

Similarly, \( x' = \frac{x}{-z} \)

Transformation is **non-linear**, but in h.c., we can make \( w = (-z) \).
Transforming canonical frustum to box

For $z$, we translate near plane to origin. But now box is too small. Specifically it has $z$ dimension $(1!+ z_{\text{min}})$ (recall $z_{\text{min}}$ is negative)

So we have an extra scale factor $1 / (1 + z_{\text{min}})$ and thus

$$z' = \frac{z - z_{\text{min}}}{1 + z_{\text{min}}}$$

But we want $x$ and $y$ to work nicely in h.c., with $w=-z$, so we use

$$z' = \frac{(z - z_{\text{min}}) / (1 + z_{\text{min}}) / (-z)}$$

(Thus in our box, depth transforms non-linearly)
In h.c.,

\[
x \mapsto x \\
y \mapsto y \\
z \mapsto \frac{(z - z_{\text{min}})}{(1 + z_{\text{min}})} \\
1 \mapsto -z
\]

So, the matrix is ？
In h.c.,

\[ x \mapsto x \]
\[ y \mapsto y \]
\[ z \mapsto (z - z_{\text{min}}) / (1 + z_{\text{min}}) \]
\[ 1 \mapsto -z \]

So, the matrix is

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 + z_{\text{min}} & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\]
Mapping to standard view volume (additional comments)

- The mapping from \([z_{\min}, -1]\) to \([0,-1]\) is non-linear. (Of course, there exists a linear mapping, but not if we want everything else to work out nicely in h.c.).
- So a change in depth of \(\triangle D\) at the near plane maps to a larger depth difference in screen coordinates than the same \(\triangle D\) at the far plane.
- But order is preserved (important!); the function is monotonic (proof?).
- And lines are still lines (proof?) and planes are still planes (important!).
Transforming canonical frustum to box

$y = -z$

Same

$z = -1$

Different

$z = d$