Admin

• Assignments are due October 21 (start soon anyway!)
• Midterm data -- October 28
• (Rough) project milestones are available
Transform object from world coords to camera coords.

Clip against view frustrum.

Further transform so that frustum is canonical frustum.

Transform object from world coords to camera coords.

Project using standard camera model.
At this point, the camera coordinate frame with the origin at the specified center of the camera plane. The focal point is along the \textbf{n} axis, a distance $f$ from the origin.
Mapping the view frustum to the canonical view frustum
Further transform so that frustum is canonical frustum.

1. Translate focal point to origin
2. Shear so that central axis of frustum lies along the n 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 to origin; call translation $T_2$. This takes center of window to:

$$\left( \frac{1}{2}(u_{\text{max}} + u_{\text{min}}), \, \frac{1}{2}(v_{\text{max}} + v_{\text{min}}), \, 0 \right)$$
Step 2: Shear this volume so that the central axis lies on the n-axis. This is a shear, because rectangles on planes n=constant must stay rectangles. Call this shear $S_1$. 

\[ \text{Diagram showing shearing process.} \]
Shear $S_1$ takes previous window midpoint to $(0, 0, f)$ - this means that matrix is
Shear $S_1$ takes previous window midpoint \( \left[ \frac{1}{2} (u_{\text{max}} + u_{\text{min}}), \frac{1}{2} (v_{\text{max}} + v_{\text{min}}), f \right] \) to \((0, 0, f)\) - this means that matrix is:

\[
\begin{bmatrix}
0 & \frac{(u_{\text{min}} + u_{\text{max}})}{2f} & 0 \\
\frac{(v_{\text{min}} + v_{\text{max}})}{2f} & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
4. Scale x, y so that planes are on $z=x$, $z=-x$ and $z=y$ and $z=-y$. Call this scale $Sc_1$

5. Isotropic scale so that far clipping plane is $z=-1$; call this scale $Sc_2$
4. Scale x, y so that planes are on z=x, z=-x and z=y and z=-y. Call this scale $S_{c_1}$

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

\[
\frac{1}{2} (v_{\text{max}} - v_{\text{min}}), \quad f \rightarrow \quad y=-z
\]  

\[
k_y \left( \frac{1}{2} (v_{\text{max}} - v_{\text{min}}) \right) = f
\]

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

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

\[
\begin{pmatrix}
\text{Point in canonical camera coordinates} \\
Sc_2 Sc_1 S_1 T_2 R_1 T_1 \\
\text{Point in world coordinates}
\end{pmatrix}
\]
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:

\[ \begin{align*}
y &> -z \\
y &< z \\
x &> -z \\
x &< z \\
z &< -1 \\
z &> z_{min}
\end{align*} \]

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

Recall C.S

Compute out codes for endpoints
While not trivial accept and not trivial reject:
  - Clip against a problem edge (out code bit is not 0)
  - 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”)