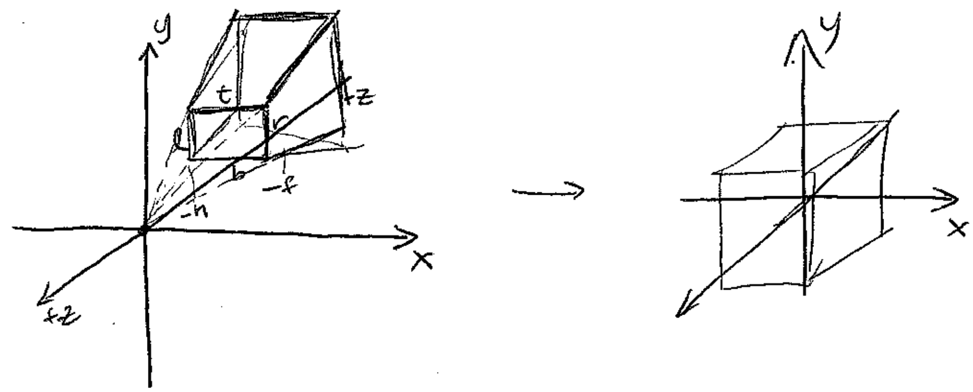
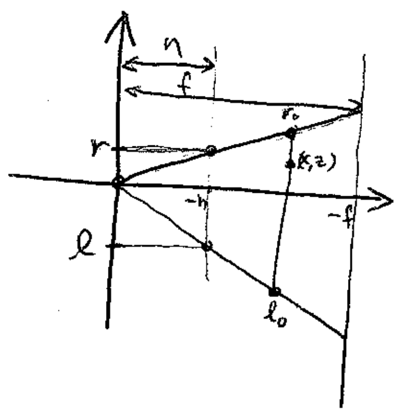
PERSPECTIVE PROJECTION



Task: Take a frustum defined by (l, r, t, b, n, f) and project it into a cube with sides ± 1

Again, first do this for the x-coordinate:



we want to transform

$$l_0 \le x \le x_0 \rightarrow -1 \le x' \le 1$$

what are r_0, l_0 in terms of what we know:

$$\frac{r_0}{-z} = \frac{r}{n} \Rightarrow r_0 = -\frac{rz}{n}$$

$$\frac{l_0}{-z} = \frac{l}{n} \Rightarrow l_0 = -\frac{lz}{n}$$

distance between r_0 and l_0 : $-\frac{rz}{n} + \frac{lz}{n} = \frac{(l-r)z}{n}$

midpt between r_0 and l_0 = $\frac{-rz - lz}{2n}$

Now write x':

$$x' = \frac{2n}{(l-r)z}\left(x - \frac{-rz-lz}{2n}\right) = \frac{2n}{(l-r)z}\left(x + \frac{(r+l)z}{2n}\right)$$

$$\boxed{x' = \frac{2n\,x}{(l-r)z} + \frac{r+l}{l-r}}$$

Hmm... there is $\frac{1}{z}$ here. How do we write this in matrix form? We'll deal with this later!

We can do same for y':

$$\boxed{y' = \frac{2n\,y}{(b-t)z} + \frac{t+b}{b-t}}$$

To get rid of that weird z in the denominator, let's multiply both sides by $-z$ (later we'll have to remember to divide)

$$\boxed{-zx' = \frac{2n}{r-l}x + \frac{r+l}{r-l}z}$$

$$\boxed{-zy' = \frac{2n}{t-b}y + \frac{t+b}{t-b}z}$$

we can divide out the $-z$ by putting $-z$ in the homogeneous coordinate w!

- Now for the z-component:

$$z' = f(z) = \frac{Az+B}{-z}$$

Now plug in what we know:

$$f(-n) = -1 \Rightarrow \frac{-An+B}{n} = -1 \Rightarrow -An+B = -n$$

$$f(-f) = 1 \Rightarrow \frac{-Af+B}{f} = 1 \Rightarrow -Af+B = f$$

$$-An + Af = -n - f$$

$$A(f-n) = -(n+f)$$

$$\boxed{A = -\frac{f+n}{f-n}}$$

plug in to get B

$$B \Rightarrow \frac{f+n}{f-n}\cdot n + B = -n$$

$$B = -\frac{f+n}{f-n}\cdot n - \frac{nf-n^2}{f-n}$$

$$= \frac{-fn-\cancel{n^2}-nf+\cancel{n^2}}{f-n}$$

$$\boxed{B = -\frac{2fn}{f-n}}$$

OpenGL perspective projection matrix

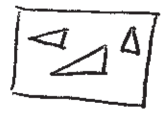
$$\boxed{M_{pers} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}}$$

to get a $-z$ in the w component!

Clipping

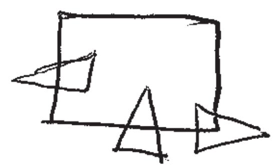
We want to draw triangles on the screen, but not all triangles will land nicely within the screen boundaries.

(drawn normally) • Some triangles will fall completely within the screen boundaries →



(culled - not drawn at all) • Some triangles will fall completely outside screen boundaries →

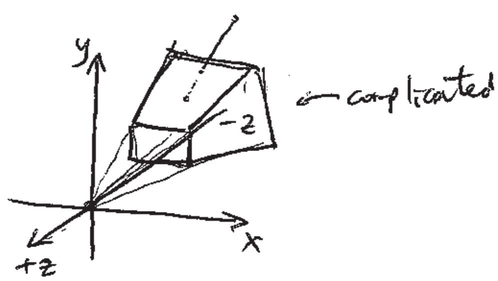
• However, some triangles will "straddle" the screen boundaries →

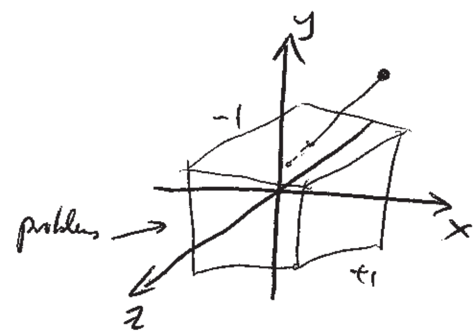


In order to render these triangles, we must break them up into triangles that are completely inside and triangles that are completely outside. This process is called clipping.

Seems like
Two potential options:

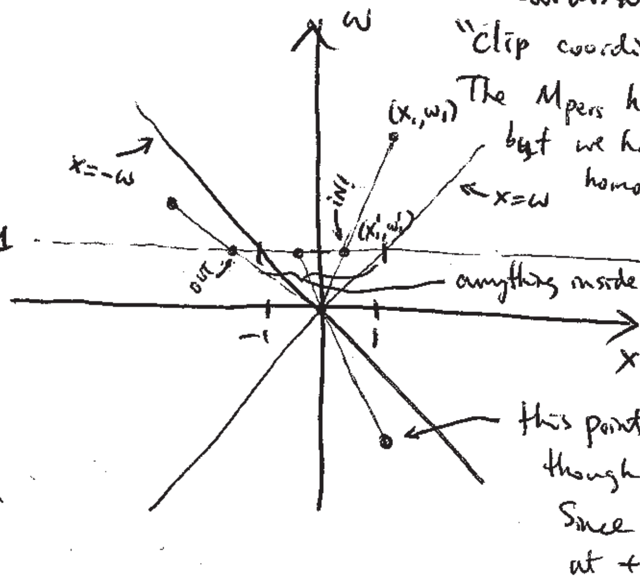
Clip using the original view frustum   or   after projection transformation.





This seem better because it's a simple test against ±1 planes. However, after homogenization there is a problem!

Post-perspective coordinates
"clip coordinates"
The M_{pers} has been applied but we have not performed homogenization ←



anything inside here would be inside our canonical view volume! ±1

negative w's mean

this point would be considered "in" even though it was behind the viewer (● w is negative ⇒ z is positive) Since viewer looking down the −z axis at the origin, anything at +z is behind them.

Comparison of clipping before/after Homogenization

After: If $\left\{\begin{array}{l} -1 \le x' \le 1 \text{ and} \\ -1 \le y' \le 1 \text{ and} \\ -1 \le z' \le 1 \end{array}\right\}$ then in else out

Pros: • easy to compute clip point

Cons: • Unnecessary division (expensive)
• Stuff from behind the user can come in front

Before: If $w_c > 0$ then

If $\left\{\begin{array}{l} -w_c \le x_c \le w_c \text{ and} \\ -w_c \le y_c \le w_c \text{ and} \\ -w_c \le z_e \le w_c \end{array}\right\}$ then in else out

else $(w_c < 0)$

If $\left\{\begin{array}{l} -w_c \ge x_c \ge w_c \text{ and} \\ -w_c \ge y_c \ge w_c \text{ and} \\ -w_c \ge z_c \ge w_c \end{array}\right\}$ then in else out.

However, computing the intersections with the clip plane are a little trickier...