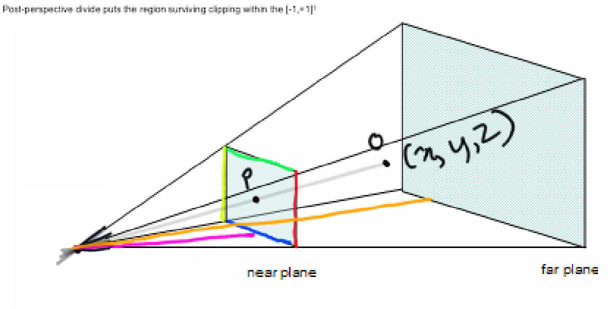
Deriving Perspective projection formulas

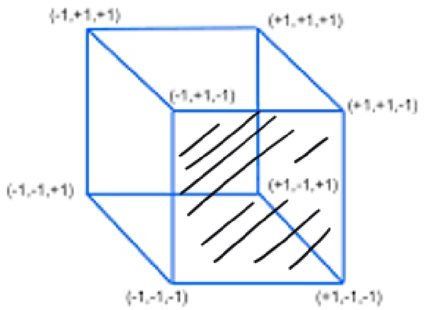
Frustum :-



- Near plane (n)
- Far plane (f)
- Right side (r)
- Left side (l)
- Top side (t)
- Bottom side (b)

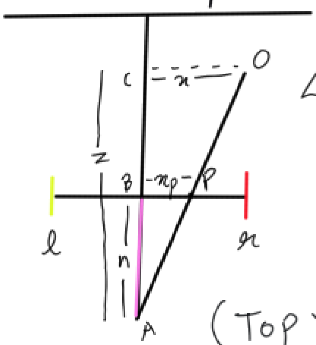
We need to find equations that map a point O(x, y, z) inside the frustum to its projection on the near plane P(x_p, y_p, z_p)

NDC Space Clip Cube



The projected point P(x_p, y_p, z_p) needs to be mapped to OpenGL's 'Normalized Device Coordinates'. (The shaded plane maps to your monitor screen.)

x to x_p

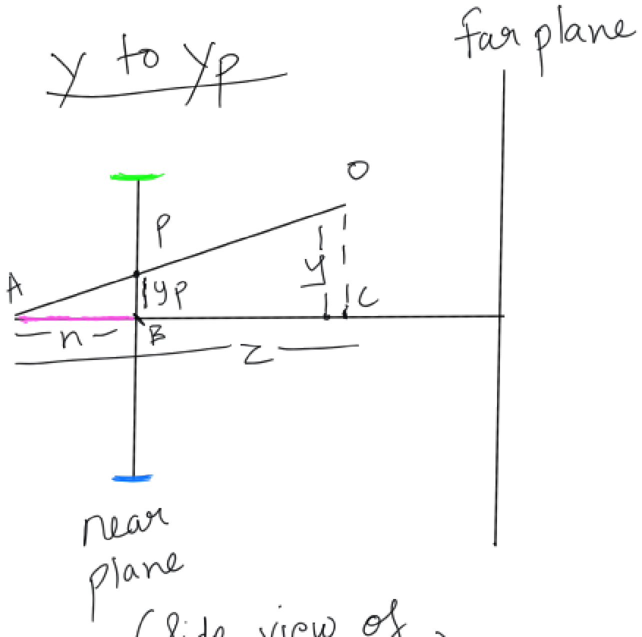


(Top view of frustum)

Δ ACO and Δ ABP are similar triangles

$$\frac{x}{z} = \frac{x_p}{n} \rightarrow x_p = \frac{nx}{z}$$

y to y_p



(Side view of frustum)

$\Delta APB \sim \Delta AOC$

$$\frac{y}{z} = \frac{y_p}{n} \rightarrow y_p = \frac{ny}{z}$$

Z to Z_p : When we do a perspective projection on near plane, we are getting rid of the z coord as we are mapping to a flat plane.

i.e $z_p = n$ always.

Converting projected points to NDC space

x_p to x_n

We want to map x_p to $-1, 1$ range.

When $x_p = l$, the point would be on left side of the screen (-1) & when $x_p = r$ the point would be on right side of screen $(+1)$

$$A x_p + B = x_n$$

We know, $Al + B = -1$ & $Ar + B = 1$

Solving these: $A(r-l) = 1-(-1) \rightarrow A = 2/(r-l)$

$$B = -1 - Al = -1 - \frac{2l}{(r-l)} = \frac{-r+l-2l}{(r-l)}$$

$$B = -\left(\frac{r+l}{r-l}\right)$$

$$\therefore \frac{2x_p}{(r-l)} - \left(\frac{r+l}{r-l}\right) = x_n$$

$$\therefore \frac{2nx}{(r-l)z} - \left(\frac{r+l}{r-l}\right) = x_n$$

Doing the same exercise for y_p we get

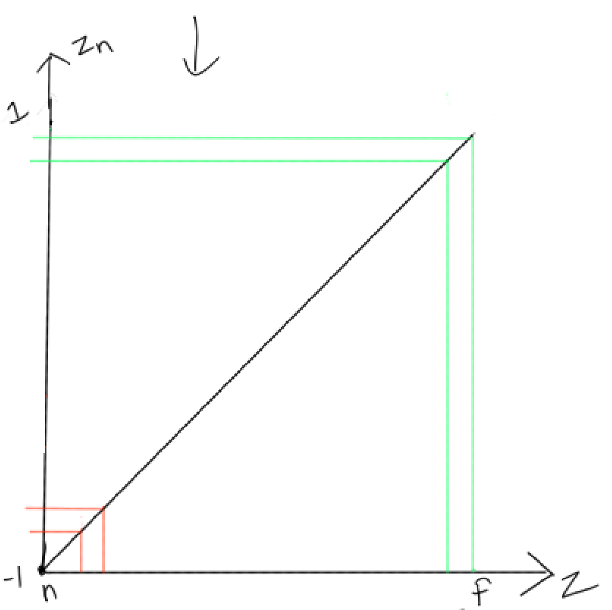
$$\boxed{\frac{2ny}{(t-b)z} - \left(\frac{t+b}{t-b}\right) = y_n}$$

z to z_n

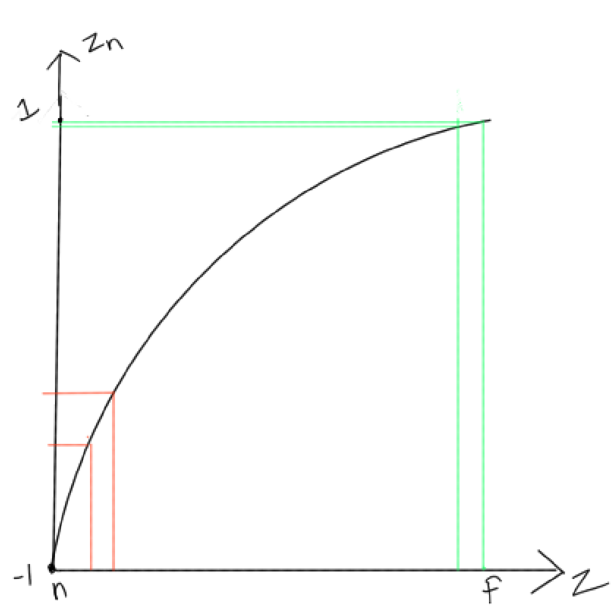
Tho it doesn't make sense to map z to a range of $(-1, 1)$ as we projecting on a flat plane, the z_n value would be used by OpenGL for depth testing (nearer objects would be drawn over farther objects)

There are 2 ways we can map z to z_n

$$Az + B = z_n \quad \& \quad \frac{Az+B}{z} = z_n$$



Change in z_n for change in z closer to n = Change in z_n for the same change in z closer to f.
i.e Depth precision is evenly distributed for all objects across the near-far plane.



Change in z_n for change in z closer to n > Change in z_n for the same change in z closer to f.
i.e There is more depth precision for objects close to near plane compared to objects close to far plane.

Choosing the second option we get lower probability of z-fighting for nearby objects.

$$\frac{Az+B}{z} = z_n \qquad \text{when } z = n, \; z_n = -1$$
$$z = f, \; z_n = 1$$

$$An + B = -n \quad \text{and} \quad Af + B = f \longrightarrow A(f-n) = f+n$$
$$A = (f+n)/(f-n)$$

$$B = -n - An = -n(1+A) = -n\left[1 + \frac{(f+n)}{(f-n)}\right] = -n\left[\frac{f-n+f+n}{f-n}\right] = -\frac{2fn}{(f-n)}$$

$$\therefore \frac{\frac{(f+n)}{(f-n)}z - \frac{2fn}{(f-n)}}{z} = z_n$$

Thus the final equations for mapping $O(x, y, z)$ in camera space to OpenGL NDC space (x_n, y_n, z_n) are:

$$x_n = \frac{2nx}{(r-l)z} - \left(\frac{r+l}{r-l}\right)$$

$$y_n = \frac{2ny}{(t-b)z} - \left(\frac{t+b}{t-b}\right)$$

$$z_n = \left[\frac{\left(\frac{f+n}{f-n}\right)z - \left(\frac{2fn}{f-n}\right)}{z}\right]$$

The equations in the last page are enough to render a scene with perspective projection. Whatever that is described here on out is used just to vectorize these equations using OpenGL's vertex pipeline.

(x_l, y_l, z_l) : Local space coordinates

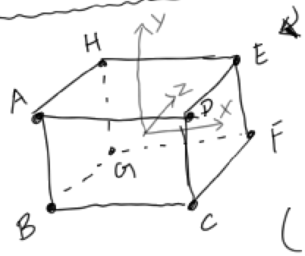
↓

Vertex shader

This is the part you can code, rest is controlled by OpenGL

your code that transforms (x_l, y_l, z_l) (Local space) to (x_q, y_q, z_q, w_q) in "w_q Scaled Device Coordinates" (WQSDC)

This is a made up term by me for my own understanding



A = $-w_q \;\; w_q \;\; -w_q$ E = $w_q \;\; w_q \;\; w_q$

B = $-w_q \;\; -w_q \;\; -w_q$ F = $w_q \;\; -w_q \;\; w_q$

C = $w_q \;\; -w_q \;\; -w_q$ G = $-w_q \;\; -w_q \;\; w_q$

D = $w_q \;\; w_q \;\; -w_q$ H = $-w_q \;\; w_q \;\; w_q$

(if $w_q = 1$, WQDC is NDC)

gl_Position = (x_q, y_q, z_q, w_q)

↓

Clip test : Is $-w_q \le z_q \le w_q$ && $-w_q \le y_q \le w_q$ && $-w_q \le x_q \le w_q$? — No → Discard Vertex

Clip test is basically checking whether your point will be within the NDC cube i.e. -1 to 1 range for all dimensions.

Yes ↓

$$\text{NDC coord} = \left(\frac{x_q}{w_q}, \frac{y_q}{w_q}, \frac{z_q}{w_q}, \frac{w_q}{w_q}\right) = (x_n, y_n, z_n, 1)$$

If you take a look at perspective equations, some terms are divided by z; we cannot execute a divide operation using matrix multiplications. Luckily if you take a look at the vertex pipeline the very last step divides all the coordinates by w_q. Hence we can leverage this to execute a divide by manipulating the value of w_q.

Vectorizing perspective equations for Right Handed Coordinate System where camera looks down the positive Z Axis

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

If we take a look at $z \to z_n$ equation, we are doing a divide by z, if we set $w_q = z$ we can perform that divide

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Since we have set $w_q = z$, equations for $x \rightarrow x_n$, $y \rightarrow y_n$, $z \rightarrow z_n$ get divided by z.

In order to balance these equations, we will muliply $x \rightarrow x_n, y \rightarrow y_n, z \rightarrow z_n$ equations with z

$$x_q = z x_n = z\left[\frac{2nx}{(r-l)z} - \left(\frac{r+l}{r-l}\right)\right]$$

$$x_q = \frac{2nx}{(r-l)} - z\left(\frac{r+l}{r-l}\right)$$

Similarly for

$$y_q = z(y_n) \quad \frac{2ny}{(t-b)} - z\left(\frac{t+b}{t-b}\right)$$

$$z_q = z(z_n)$$

$$z_q = \left(\frac{f+n}{f-n}\right)z - \left(\frac{2fn}{f-n}\right)$$

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix} = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & -\left(\frac{r+l}{r-l}\right) & 0 \\ 0 & \frac{2n}{(t-b)} & -\left(\frac{t+b}{t-b}\right) & 0 \\ 0 & 0 & \frac{(f+n)}{(f-n)} & \frac{-2fn}{(f-n)} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Consider $n = 1$, $f = 10$ since we are viewing along $+z$ Axis. Thus for all points within the frustum z will +ve → hence w_q will be +ve.

Assume $(x, y, 1)$ pt.

$$(x_q, y_q, z_q, w_q) = \left(x_q, y_q, \frac{11}{9}(1) - \frac{20}{9}, 1\right)$$

$$= \left(x_q, y_q, \frac{-9}{9}, 1\right)$$

$$= (x_q, y_q, -1, 1)$$

Opengl will perform a clip test

$$-w_q \le x_q \le w_q \qquad\qquad -1 \le x_q \le 1$$
$$-w_q \le y_q \le w_q \quad \Rightarrow \quad -1 \le y_q \le 1$$
$$-w_q \le z_q \le w_q \qquad\qquad -1 \le 1 \le 1$$

$\downarrow$

So if x_q, y_q are also in the "w_q Scaled Device Coordinates" then this point will be rendered.

Vectorize perspective projection equations for RHS where camera looks down the $-ve$ Z axis

~~Let~~ Let's consider the matrix we derived for looking down $+ve$ Z axis in RHS.

Since we are looking down the $-ve$ Z axis, let's consider $n = -1$, $f = -10$. All points viewable (i.e points in frustum) have z between -1 to -10.

Consider point $(x, y, -1)$

$$(x_q, y_q, z_q, w_q) = \left(x_q, y_q, \frac{-11}{9} - \left(\frac{20}{-9}\right), -1\right)$$
$$= \left(x_q, y_q, \frac{-11+20}{9}, -1\right)$$
$$= \left(x_q, y_q, 9/9, -1\right) = (x_q, y_q, 1, -1)$$

OpenGL will perform a clip test :-

$$-w_q \le x_q \le w_q \longrightarrow 1 \le x_q \le -1$$
$$-w_q \le y_q \le w_q \longrightarrow 1 \le y_q \le -1$$
$$-w_q \le z_q \le w_q \longrightarrow 1 \le 1 \le -1$$

Here the comparison $1 \le (\text{any value}) \le -1$ doesn't make any sense. A value cannot be greater than 1 AND less than -1 at the same time. This happened because w_q is negative. If we use the current matrix w_q will always be -ve for all viewable points. Due to above mentioned reason all such points will fail the clipping test & we won't see any viewable objects.

Thus to fix this we need to make sure w_q is +ve for viewable points i.e set $w_q = -z$

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

We need to modify our $\begin{matrix} x \\ y \\ z \end{matrix} \rightarrow \begin{matrix} x_n \\ y_n \\ z_n \end{matrix}$ equations to account for $(w_q = -z)$ divide.

For $\begin{matrix} x \rightarrow x_n \\ y \rightarrow y_n \\ z \rightarrow z_n \end{matrix}$ equations, we need to balance them by multiplying by $-z$

$$x_q = -z\,x_n = -z\left(\frac{2nx}{(r-l)z} - \frac{(r+l)}{(r-l)}\right)$$

$$x_q = \frac{-2nx}{(r-l)} + \frac{z(r+l)}{(r-l)}$$

Similarly for

$$y_q = -z y_n = \frac{-2ny}{(t-b)} + z\left(\frac{t+b}{t-b}\right)$$

$$z_q = -z(z_n) = -\left(\frac{f+n}{f-n}\right)z + \left(\frac{2fn}{f-n}\right)$$

$$\therefore \begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix} = \begin{bmatrix} \frac{-2n}{(r-l)} & 0 & \left(\frac{r+l}{r-l}\right) & 0 \\ 0 & \frac{-2n}{(t-b)} & \left(\frac{t+b}{t-b}\right) & 0 \\ 0 & 0 & -\left(\frac{f+n}{f-n}\right) & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Thus,

For RHS, looking down +ve Z axis

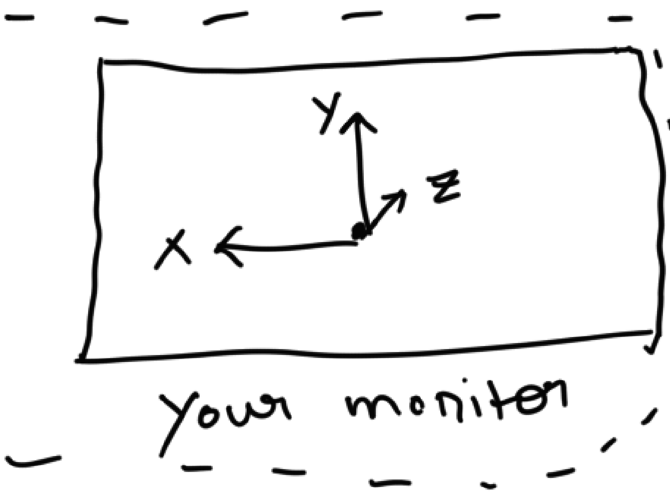
$$\begin{bmatrix} x_a \\ y_a \\ z_a \\ w_a \end{bmatrix} = \begin{bmatrix} \frac{2n}{(r-l)} & 0 & -\left(\frac{r+l}{r-l}\right) & 0 \\ 0 & \frac{2n}{(t-b)} & -\left(\frac{t+b}{t-b}\right) & 0 \\ 0 & 0 & \left(\frac{f+n}{f-n}\right) & \frac{-2fn}{(f-n)} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

For RHS, looking down -ve Z axis

$$\begin{bmatrix} x_a \\ y_a \\ z_a \\ w_a \end{bmatrix} = \begin{bmatrix} \frac{-2n}{(r-l)} & 0 & \left(\frac{r+l}{r-l}\right) & 0 \\ 0 & \frac{-2n}{(t-b)} & \left(\frac{t+b}{t-b}\right) & 0 \\ 0 & 0 & -\left(\frac{f+n}{f-n}\right) & \frac{2fn}{(f-n)} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In RHS viewing along +ve Z Axis

+ve X Axis would be along the left side of your screen.

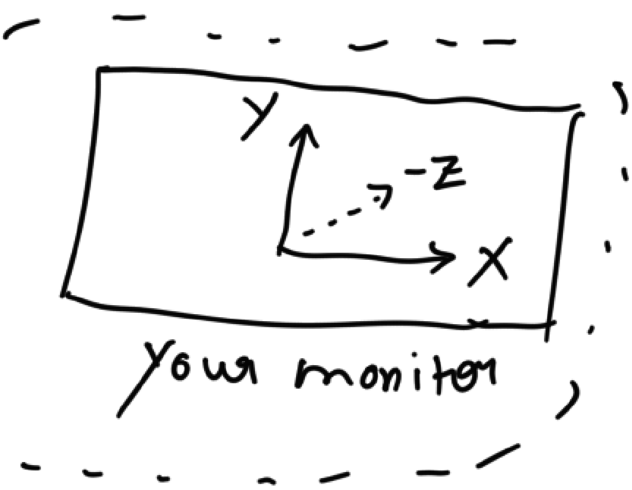


Hence +ve displacement along X will move the objects to the left. This can be counterintuitive for many people as we are used to assuming +ve X along Right hand side i.e many would expect the object to go Right for a +ve displacement along X. Same thing applies for camera movement. We would have to have +ve displacement along X when pressing 'A' to move camera to the left.

Thus,

Movement along Z would work as expected but movement along X would be opposite ~~to~~ our intuition

For RHS, -ve Z Axis



+ve displacement along X = Moves Right

-ve displacement along X = Moves Left

+ve displacement along Z would move objects closer to screen, whereas -ve displacement along Z would move them farther. This might be counterintuitive to some as one might expect the object to move farther for +ve displacement along Z. This applies to camera movement as well. We would need to have a -ve displacement when pressing 'W' so that camera 'zooms' into the scene.

Thus,

Movement along X would be as expected. However movement along Z would be opposite to our intuition