Projections

- Mapping from d dimensional space to $d-1$ dimensional subspace
- Range of any projection $\mathcal{P} : R^3 \rightarrow R^2$ called a *projection plane*
- $\mathcal{P}$ maps lines to points
- The image of any point $\mathbf{p}$ under $\mathcal{P}$ is the intersection of a *projection line* through $\mathbf{p}$ with the projection plane.

Parallel Projections

- All projection lines are parallel.
- An *orthographic projection* has projection lines orthogonal to projection plane.
- Otherwise a parallel projection is an *oblique projection*
- Particularly interesting oblique projections are the *cabinet projection* and the *cavalier projection*.

Perspective Projection

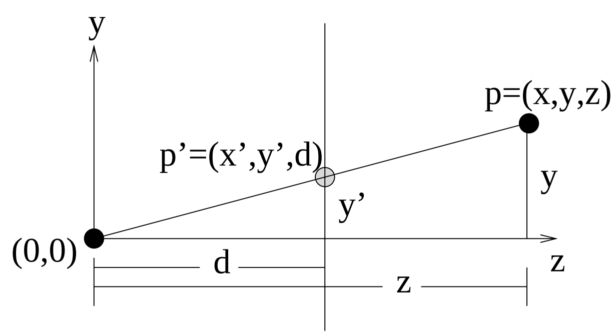
- All projection lines pass through the *center of projection* (eyepoint).
- Therefore also called *central projection*
- This is *not* affine, but rather a *projective transformation*.

Projective Transformation

- Does not preserve angles, distances, ratios of distances or affine combinations.
- *Cross ratios* are preserved.
- Incidence relationships are generally preserved.
- Straight lines are mapped to straight lines.

Perspective Transform in Eye Coordinates

- Given a point $\mathbf{p}$, find its projection $\mathcal{P}(\mathbf{p})$
- Convenient to do this in *eye coordinates*, with center of projection at origin and $z = d$ projection plane
- Note that eye coordinates are left-handed



Projection plane, z=d

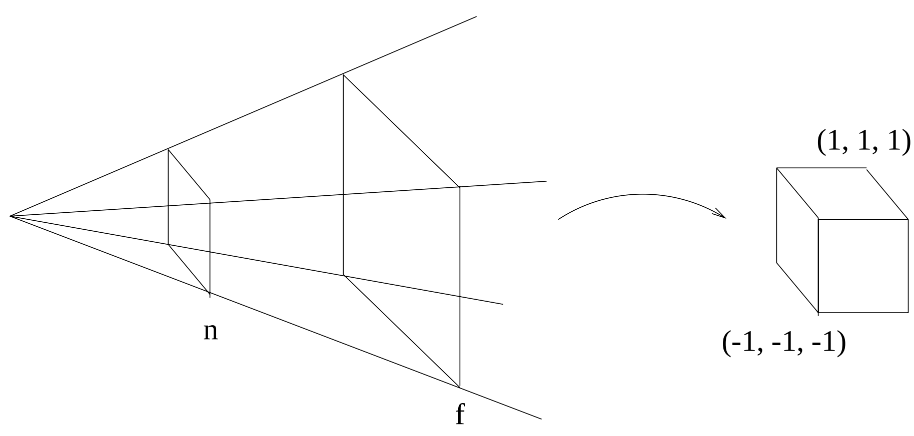
- Due to similar triangles $\mathcal{P}(\mathbf{p}) = (dx/z, dy/z, d)$
- For any other point $\mathbf{q} = (kx, ky, kz), k \neq 0$ on same projection line $\mathcal{P}(\mathbf{q}) = (dx/z, dy/z, d)$
- If we have surfaces, we need to know which ones occlude others from the eye position
- This projection loses all z information, so we cannot do occlusion testing after projection

Homogeneous Coordinates

- Homogeneous coordinates represent n-space as a subspace of $n+1$ space
- For instance, homogeneous 4-space embeds ordinary 3-space as the $w = 1$ hyperplane
- Thus, we can obtain the 3-d image of any homogeneous point $(wx, wy, wz, w), w \neq 0$ as $(x, y, z, 1) = (wx/w, wy/w, wz/w, w/w)$, that is, by dividing all coordinates by w.
- Lines in homogeneous space which intersect the $w = 1$ hyperplane project to 3-space points.
- Notice that this is just a perspective projection from 4-d homogeneous space to 3-space, instead of dividing by z, we are dividing by w.

The OpenGL Perspective Matrix

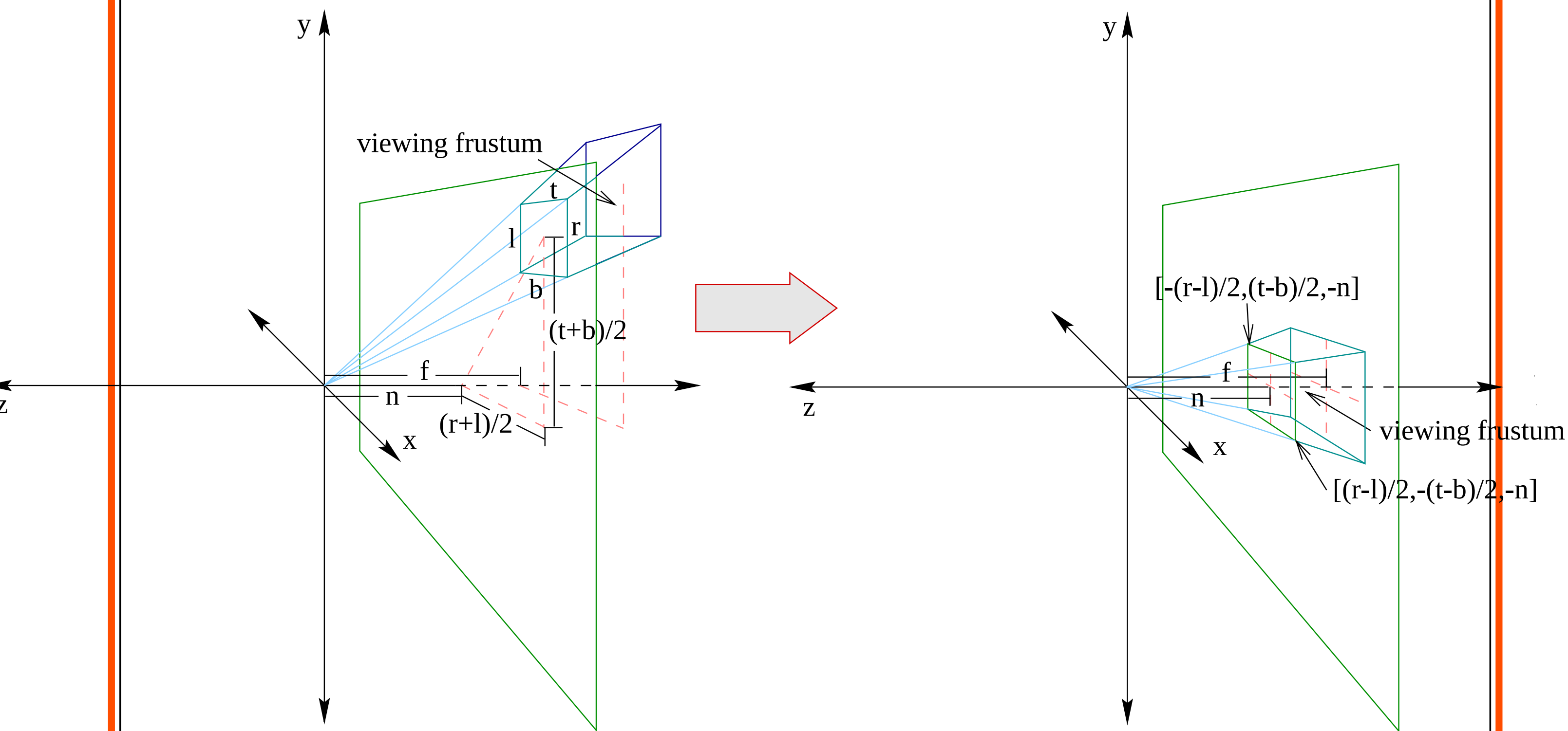
- The visible volume in world space is known as the *viewing pyramid* or *frustum*.
- Specify with the call **glFrustum**(l, r, b, t, n, f)
- In OpenGL, the window is in the *near* plane
- l and r are u-coordinates of left and right window boundaries in the near plane
- b and t are v-coordinates of bottom and top window boundaries in the near plane
- n and f are *positive distances* from the eye along the viewing ray to the near and far planes
- Maps the left and right clipping planes to $x = -1$ and $x = 1$
- Maps the bottom and top clipping planes to $y = -1$ and $y = 1$
- Maps the near and far clipping planes to $z = -1$ and $z = 1$



Shearing the Window to the z axis

- After applying the modelview matrix, we are looking down the $-z$ axis.
- We need to move the ray from the origin through the window center onto the $-z$ axis.
- Rotation won't do since the window wouldn't be orthogonal to the z axis.
- Translation won't do since we need to keep the eye at the origin.
- We need differential translation as a function of z, i.e. shear.
- When $z = -n$, δx should be $-\frac{r+l}{2n}$ and δy should be $-\frac{t+b}{2n}$, so we get

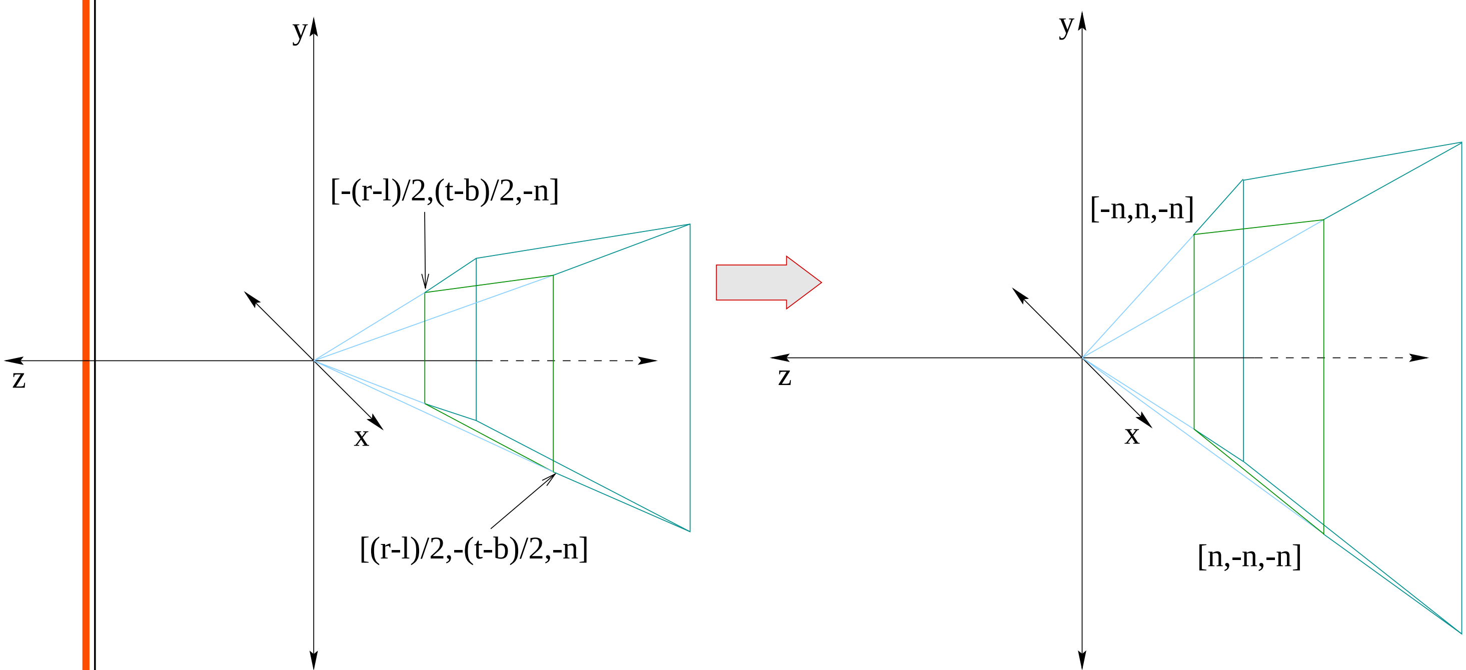
$$
\begin{aligned}
x' &= x + \frac{r+l}{2n} z \\
y' &= y + \frac{t+b}{2n} z \\
z' &= z
\end{aligned}
$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{r+l}{2n} & 0 \\ 0 & 1 & \frac{t+b}{2n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Adjusting the Clipping Boundaries

- For ease of clipping, we want the oblique clipping planes to have equations $x = \pm z$ and $y = \pm z$.
- This will make the window square, with boundaries $l = b = -n$ and $r = t = n$.
- This requires a scale to make the window this size.



Thus the mapping is

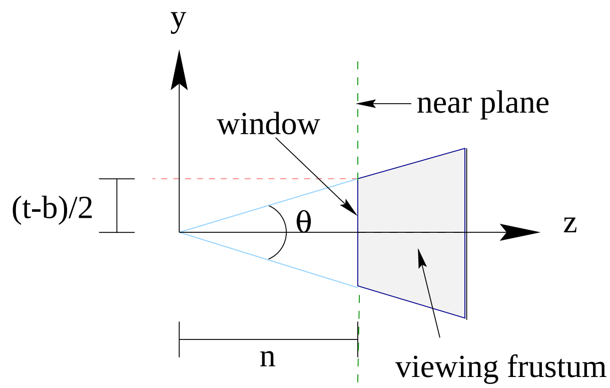
$$
\begin{aligned}
x' &= \frac{2nx}{r-l} \\
y' &= \frac{2ny}{r-l} \\
z' &= z
\end{aligned}
$$

or in matrix form:

$$
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

Field of View Frustum Scaling

- After the frustum is centered on the $-z$ axis:



- Note that $\frac{n}{t-b} = \cot\left(\frac{\theta}{2}\right)$
- This gives the y mapping $y'' = y' \cot\left(\frac{\theta}{2}\right)$
- Since the window need not be square, we can define the x mapping using the aspect ratio $aspect = \frac{\Delta x}{\Delta y} = \frac{(r-l)}{(t-b)}$
- Then x maps as $x'' = x' \frac{\cot\left(\frac{\theta}{2}\right)}{aspect}$

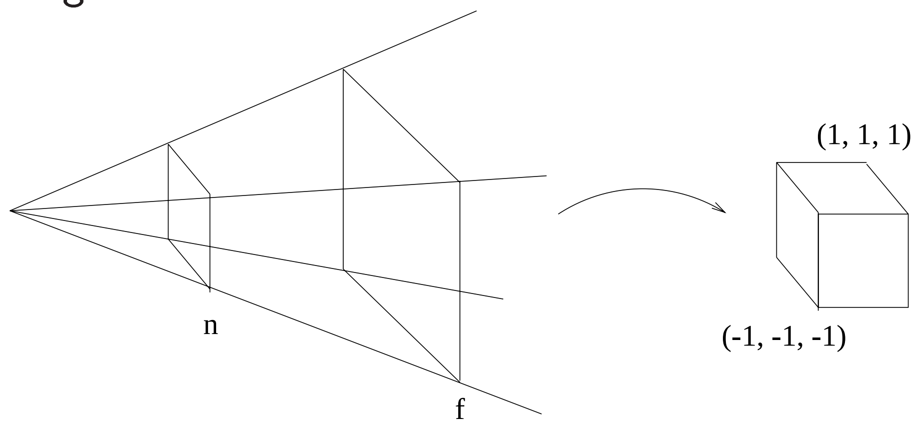
- This gives us the alternative scaling formulation:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\cot\left(\frac{\theta}{2}\right)}{aspect} & 0 & 0 & 0 \\ 0 & \cot\left(\frac{\theta}{2}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- This is used by **gluPerspective**$(\theta, aspect, n, f)$

Perspective Mapping

- Recall that we want to map the frustum to a 2x2x2 cube centered at the origin.



- OpenGL looks down $-z$ rather than z.
- Note that when you specify n and f, they are given as *positive* distances down $z = -1$.
- First we map the bounding planes $x = \pm z$ and $y = \pm z$ to the planes $x = \pm 1$ and $y = \pm 1$.
- This can be done by mapping x to $\frac{x}{-z}$ and y to $\frac{y}{-z}$.
- If we set $z' = -1$, this is equivalent to projecting onto the $z = -1$ plane.
- However, we want to derive a map for z that preserves lines and depth information.
- To map x to $\frac{x}{-z}$ and y to $\frac{y}{-z}$ -
- First use a matrix to map to homogeneous coordinates, then project back to 3 space by dividing (normalizing).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & c \\ 0 & 0 & b & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ az + c \\ bz + d \end{bmatrix}$$

$$\equiv \begin{bmatrix} \frac{x}{bz+d} \\ \frac{y}{bz+d} \\ \frac{az+c}{bz+d} \\ 1 \end{bmatrix}$$

- Now we solve for a, b, c and d such that $z \in [n, f]$ maps to $z' \in [-1, 1]$.
- To map x to $\frac{x}{-z}$,

$$\frac{x}{bz + d} = \frac{x}{-z} \Rightarrow d = 0 \text{ and } b = -1$$

- Thus

$$\frac{az + c}{bz + d} \text{ becomes } \frac{az + c}{-z}$$

- Since the near plane is at $z = -n$ and the far plane at $z = -f$, our constraints on the near and far clipping planes (e.g., that they map to -1 and 1) give us

$$
\begin{aligned}
-1 = \frac{-an + c}{n} \quad &\Rightarrow \quad c = -n + an \\
1 = \frac{-af - n + an}{f} \quad &\Rightarrow \quad (f + n) = a(n - f) \\
&\Rightarrow \quad a = \frac{f + n}{n - f} \\
&\Rightarrow \quad a = \frac{-(f + n)}{f - n} \\
&\Rightarrow \quad c = -n + \frac{-(f + n)n}{f - n} \\
&\qquad = \frac{-n(f - n) - n(f + n)}{f - n} \\
&\qquad = \frac{-2fn}{f - n}
\end{aligned}
$$

This gives us

$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}
\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
=
\begin{bmatrix} x \\ y \\ \frac{-z(f+n)-2fn}{f-n} \\ -z \end{bmatrix}
$$

- After normalizing we get

$$\left[\ \frac{x}{-z}\ ,\ \frac{y}{-z}\ ,\ \frac{-z(f+n)-2fn}{-z(f-n)}\ ,1\right]^T$$

- If we multiply this matrix in with the geometric transforms, the only additional work is the divide.
- After normalization we are in *left-handed* 3-dimensional *Normalized Device Coordinates*

Complete OpenGL Perspective Matrix

- Combining the three steps given above, the complete OpenGL perspective matrix is

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & \frac{r+l}{2n} & 0 \\ 0 & 1 & \frac{t+b}{2n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Using **gluPerspective** the matrix becomes

$$\begin{bmatrix} \frac{\cot(\theta/2)}{aspect} & 0 & 0 & 0 \\ 0 & \cot(\theta/2) & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Why Map Z

- 3D $\mapsto$ 2D projections map all z to same value.
- Need z to determine occlusion, so a 3D to 2D projective transformation doesn't work.
- Further, we want 3D lines to map to 3D lines (this is useful in hidden surface removal).
- The mapping $(x, y, z, 1) \mapsto (xn/z, yn/z, n, 1)$ maps lines to lines, but loses all depth information.
- We could use

$$(x, y, z, 1) \mapsto (xn/z, yn/z, z, 1)$$

  Thus, if we map the endpoints of a line segment, these end points will have the same relative depths after this mapping.
  BUT: It fails to map lines to lines
- The map

$$(x, y, z, 1) \mapsto \left(\frac{xn}{z}, \frac{yn}{z}, \frac{zf + zn - 2fn}{z(f - n)}, 1 \right)$$

  does map lines to lines, *and* it preserves depth information.

Mapping Z

- It's clear how x and y map. How about z?

$$z \mapsto \frac{zf + zn - 2fn}{z(f-n)} = P(z)$$

- We know $P(f) = 1$ and $(P(n) = -1$. What maps to 0?

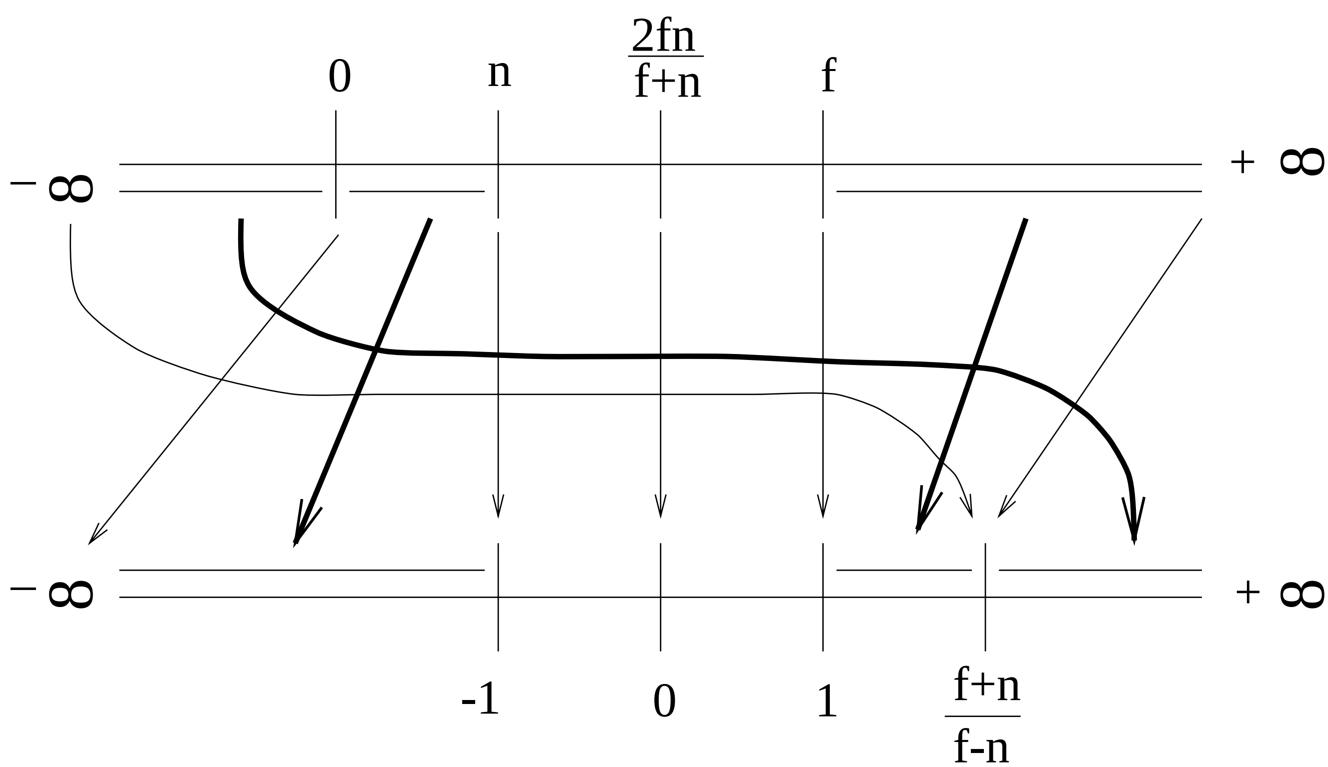
$$\begin{aligned} & P(z) && = 0 \\ \Rightarrow \quad & \frac{zf + zn - 2fn}{z(f-n)} && = 0 \\ \Rightarrow \quad & z && = \frac{2fn}{f+n} \end{aligned}$$

Note that $f^2 + 2f > 2fn/(f+n) > fn + n^2$ so

$$f > \frac{2fn}{f+n} > n$$

- What happens as map z to 0 or to infinity?

$$
\begin{aligned}
\lim_{z \to 0^+} P(z) &= \frac{-2fn}{z(f-n)} \\
&= -\infty \\
\lim_{z \to 0^-} P(z) &= \frac{-2fn}{z(f-n)} \\
&= +\infty \\
\lim_{z \to +\infty} P(z) &= \frac{z(f+n)}{z(f-n)} \\
&= \frac{f+n}{f-n} \\
\lim_{z \to -\infty} P(z) &= \frac{z(f+n)}{z(f-n)} \\
&= \frac{f+n}{f-n}
\end{aligned}
$$


0
n
2fn / f+n
f
− ∞
+ ∞
− ∞
+ ∞
-1
0
1
f+n / f-n

- What happens if we vary f and n?

$$\begin{aligned}\lim_{f \to n} P(z) &= \frac{z(f+n) - 2fn}{z(f-n)} \\ &= \frac{(2zn - 2n^2)}{z \cdot 0}\end{aligned}$$

which is not surprising, since we're trying to map a single point to a line segment.

$$\begin{aligned}\lim_{f \to \infty} P(z) &= \frac{zf - 2fn}{zf} \\ &= \frac{z - 2n}{z}\end{aligned}$$

- But note that this means we are mapping an infinite region to [0,1] and we will effectively get a far plane due to floating point precision,

$$\begin{aligned}\lim_{n \to 0} P(z) &= \frac{zf}{zf} \\ &= 1\end{aligned}$$

i.e., the entire map becomes constant (again, we are mapping a point to an interval).

- Consider what happens as f and n move away from each other.
  - We are interested in the size of the regions $[n, 2fn/(f+n)]$ and $[2fn/(f+n), f]$.
  - When f is large compared to n, we have

$$\frac{2fn}{f+n} \doteq 2n$$

So

$$\frac{2fn}{f+n} - n \doteq n$$

and

$$f - \frac{2fn}{f+n} \doteq f - 2n$$

But both intervals are mapped to a regions of size 1.
  - Thus, as we move the clipping planes away from one another, the far interval is compressed more than the near one. With floating point arithmetic, this means we'll lose precision.
  - In the extreme case, think about what happens as we move f to infinity: we compress an infinite region to an finite one.
  - Therefore, we try to place our clipping planes as close to one another as we can.

Clipping in Homogeneous Space

Projection: linear transformations then normalize

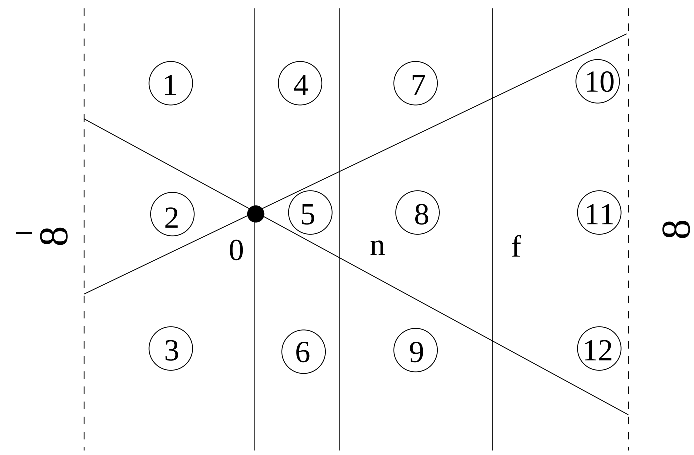
- Linear transformation

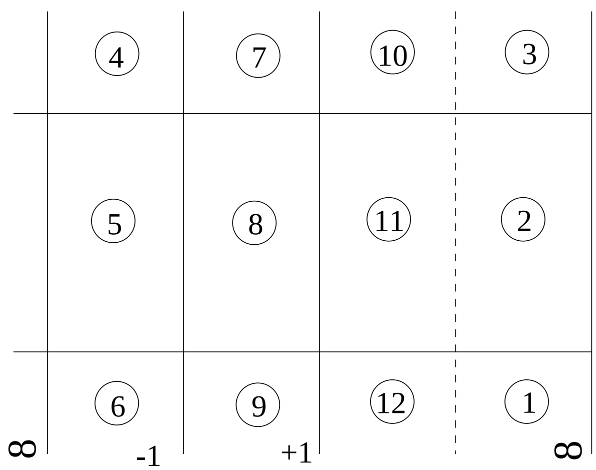
$$\begin{bmatrix} nr & 0 & 0 & 0 \\ 0 & ns & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \\ \bar{w} \end{bmatrix}$$

- Normalization

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \\ \bar{w} \end{bmatrix} = \begin{bmatrix} \bar{x}/\bar{w} \\ \bar{y}/\bar{w} \\ \bar{z}/\bar{w} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Region Mapping





Clipping not good after normalization:

- Ambiguity after normalization

$$-1 \leq \frac{\bar{x}, \bar{y}, \bar{z}}{\bar{w}} \leq +1$$

  - Numerator can be positive or negative
  - Denominator can be positive or negative
- Normalization expended on points that are subsequently clipped.

Clipping in homogeneous coordinates:

- Compare unnormalized coordinate against $\bar{w}$

$$-|\bar{w}| \leq \bar{x}, \bar{y}, \bar{z} \leq +|\bar{w}|$$