Projections

- Mapping from d dimensional space to $d-1$ dimensional subspace
- Range of any projection $\mathcal{P} : R^3 \longrightarrow R^2$ 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

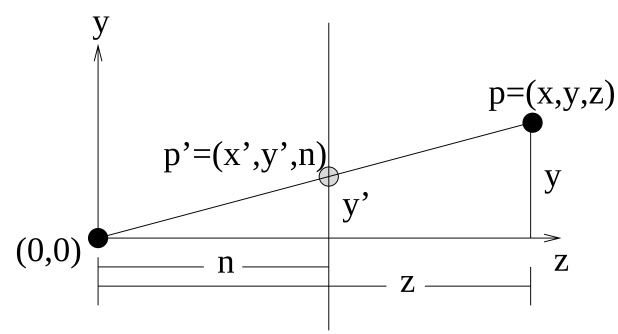
- All projection lines pass through the *center of projection* (eyepoint).
- Therefore also called *central projection*
- This is *not* affine, but rather a *projective transformation*.

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 $z = n$ projection plane
- Note that eye coordinates are left-handed

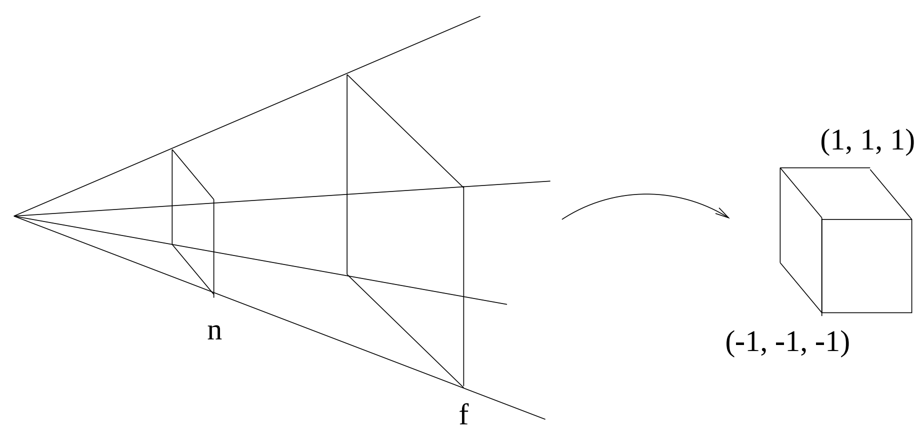


Projection plane, z = n

- Due to similar triangles $\mathcal{P}(\mathbf{p}) = (nx/z, ny/z, n)$
- For any other point $\mathbf{q} = (kx, ky, kz), k \neq 0$ on same projection line $\mathcal{P}(\mathbf{q}) = (nx/z, ny/z, n)$
- 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

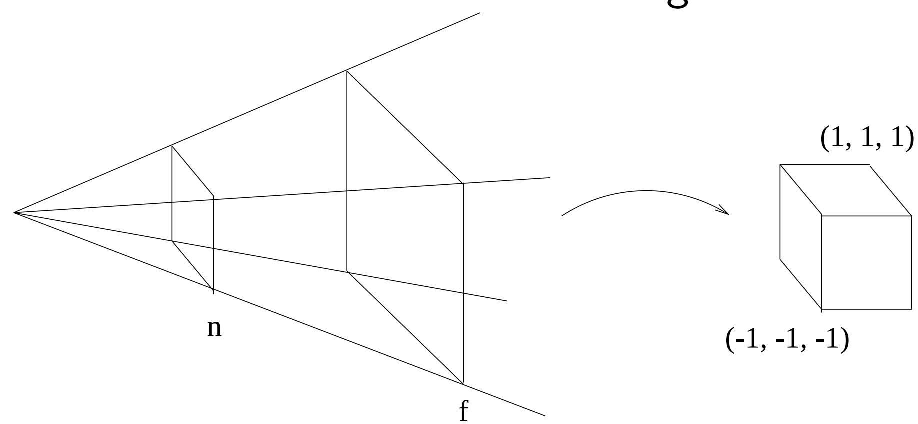
The OpenGL Viewing System

- The camera's visible volume in world space is known as the *viewing pyramid* or *frustum*.
- Specify perspective projection 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
- OpenGL looks down $-z$ rather than z.
- Specify orthographic projection with the call **glOrtho**(l, r, b, t, n, f). While similar parameters to glFrustum, the view volume is a right parallelopiped.



Mapping Perspective to Orthographic

- Map the frustum to a 2x2x2 cube centered at the origin.



-
- 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

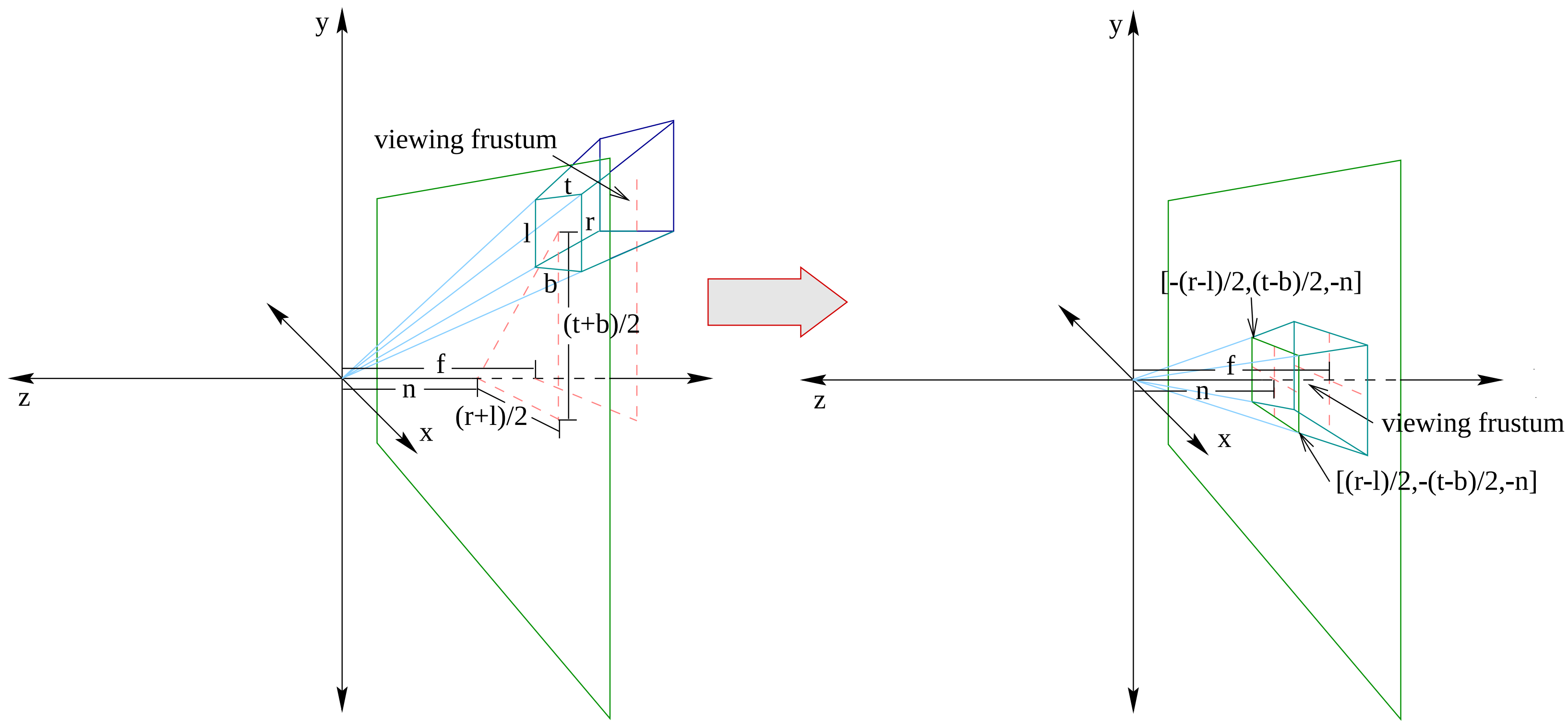
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \frac{-z(f+n)-2fn}{f-n} \\ -z \end{bmatrix}$$

- 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*

Mapping the Frustum to Canonical Position



- 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.

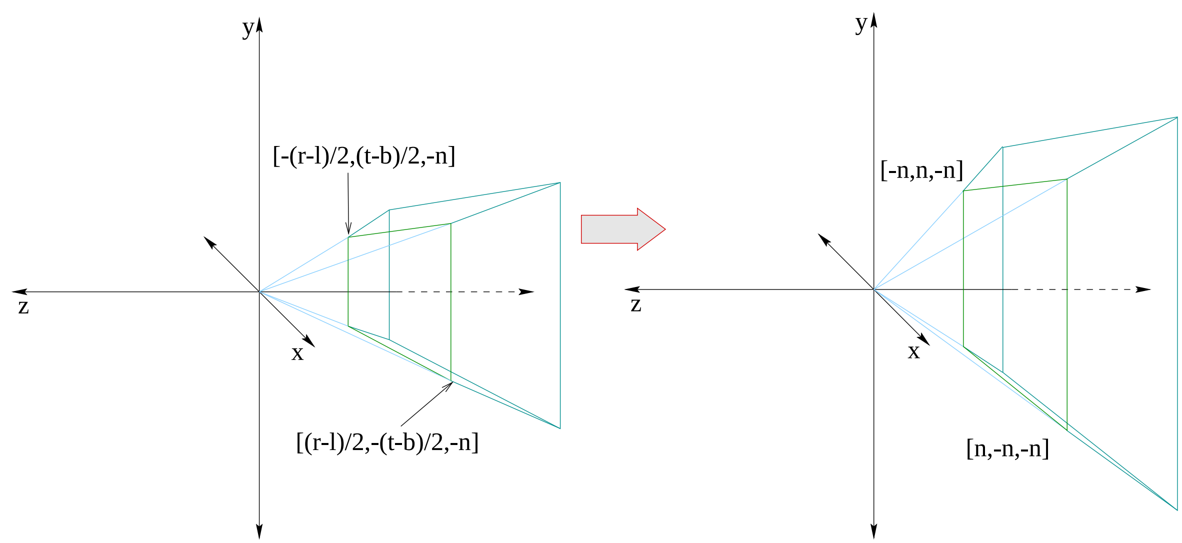
- When $z = -n$, δx should be $-\frac{r+l}{2n}$ and δy should be $-\frac{t+b}{2n}$, so we get

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

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

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:

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

Complete OpenGL Perspective Matrix

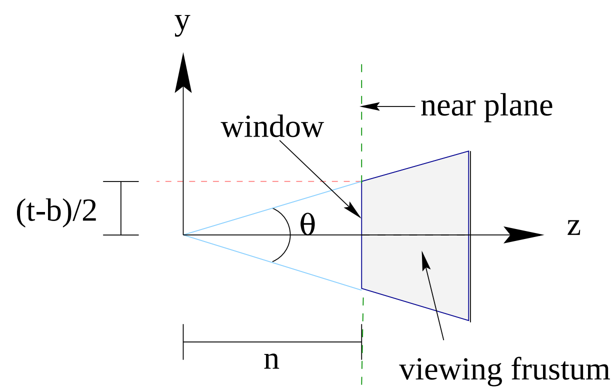
- Combining the three steps given above, the complete OpenGL perspective matrix is

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

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

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 rati
aspect $= \frac{\Delta x}{\Delta y} = \frac{(r-l)}{(t-b)}$

– Then x maps as $x'' = x' \frac{\cot\left(\frac{\theta}{2}\right)}{\text{aspect}}$

– This gives us the alternative scaling formulation:

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

– This is used by **gluPerspective**$(\theta, \text{aspect}, n, f)$

Reading Assignment and News

Chapter 5 pages 217 - 265, of Recommended Text.

Please also track the News section of the Course Web Pages for the most recent Announcements related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics23/cs354/)