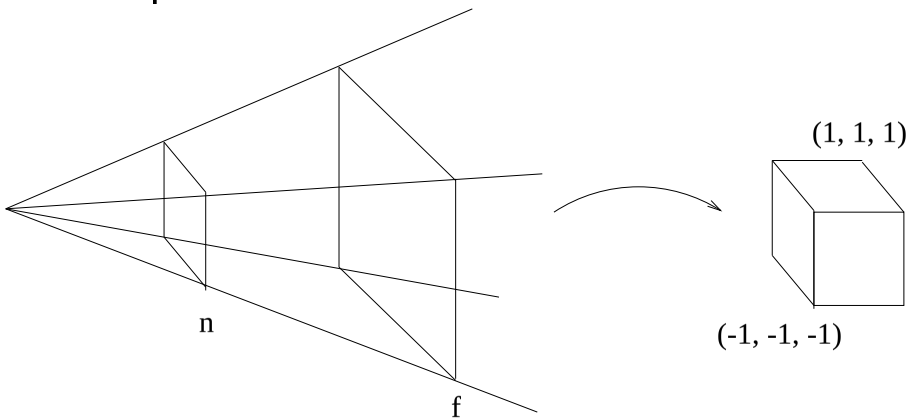
Perspective Mapping

- 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

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

Picking and 3D Selection

- *Pick:* Select an object by positioning mouse over it and clicking
- *Question:* How do we decide what was picked?
  - We could do the work ourselves:
    - Map selection point to a ray
    - Intersect with all objects in scene
  - Let OpenGL/graphics hardware do the work
- *Idea:* Draw entire scene, and "pick" anything drawn near the cursor
  - Only "draw" in a small viewport near the cursor
  - Just do clipping, no shading or rasterization
  - Need a method of identifying "hits"
  - OpenGL uses a *name stack* managed by `glInitNames()`, `glLoadName()`, `glPushName()`, and `glPopName()`
  - "Names" are short integers
  - When hit occurs, copy entire contents of stack to output buffer
- *Example:*

```
glSelectBuffer(size, buffer);           /* initialize */
glRenderMode(GL_SELECT);
glInitNames();
glGetIntegerv(GL_VIEWPORT, viewport);  /* set up pick view */
glMatrixMode(GL_PROJECTION);
glPushMatrix();
glIdentity();
gluPickMatrix(x, y, w, h, viewport);
glMatrixMode(GL_MODELVIEW);

ViewMatrix();
glLoadName(1);
Draw1();
glLoadName(2);
Draw2();

glMatrixMode(GL_PROJECTION);
glPopMatrix();
glMatrixMode(GL_MODELVIEW);
hits = glRenderMode(GL_RENDER);
```

- What you get back:
  - If you click on Item 1 only:
    ```
    hits = 1,
    buffer = 1, min(z1), max(z1), 1.
    ```
  - If you click on Item 2 only:
    ```
    hits = 1,
    buffer = 1, min(z2), max(z2), 2.
    ```
  - If you click over both Item 1 and Item 2:
    ```
    hits = 1,
    buffer = 1, min(z1), max(z1), 1, 1, min(z2), max(z2), 2.
    ```

- More complex example:

```
/* initialization stuff goes here */
glPushName(1);
    Draw1();                    /* stack: 1 */
    glPushName(1);
        Draw1_1();              /* stack: 1 1 */
        glPushName(1);
            Draw1_1_1();        /* stack: 1 1 1 */
        glPopName();
        glPushName(2);
            Draw1_1_2();        /* stack: 1 1 2 */
        glPopName();
    glPopName();
    glPushName(2);
        Draw1_2();              /* stack: 1 2 */
    glPopName();
glPopName();
glPushName(2);
    Draw2();                    /* stack: 2 */
```

```
glPopName();
/* wrap-up stuff here */
```

- What you get back:
  - If you click on Item 1:
    `hits = 1,`
    `buffer = 1, min(z1), max(z1), 1.`
  - If you click on Items 1:1:1 and 1:2:
    `hits = 2,`
    `buffer = 3, min(z111), max(z111), 1, 1, 1, 2, min(z12),`
    `max(z12), 1, 2.`
  - If you click on Items 1:1:2, 1:2, and 2:
    `hits = 3,`
    `buffer = 3, min(z112), max(z112), 1, 1, 2, 2, min(z12),`
    `max(z12), 1, 2, 1, min(z2), max(z2), 2.`
- In general, if h is the number hits, the following is returned.
  - `hits` $= h$.
  - h *hit records*, each with four parts:
    1. The number of items q on the name stack at the time of the hit (1 int).
    2. The minimum z value among the primitives hit (1 int).
    3. The maximum z value among the primitives hit (1 int).
    4. The contents of the hit stack, deepest element first (q ints).

- Important Details:
  - Make sure that projection matrix is saved with a `glPushMatrix()` and restored with a `glPopMatrix()`.
  - `glRenderMode(GL_RENDER)` returns negative if buffer not big enough.
  - When a hit occurs, a flag is set.
  - Entry to name stack only made at next `gl*Name(s)` or `glRenderMode` call. So, each draw block can only generate at most one hit.