View-Perspective Projection
Default OpenGL View

- Eye at Origin
- Image plane perpendicular to negative Z
- View Up Vector coincident with Y
View Transformation

- **Eye at** $E=(x_0, y_0, z_0)$
  - Need not be $(0,0,0)$
- **LookAt** $(x_n, y_n, z_n)$ vector defines normal $N$ to the image plane
  - Need not be the Z axis
- **View Up** vector $V$ defines orientation of the view.
  - Need not be perpendicular to $N$
  - Need not be the Y axis
- **Transformation to default OpenGL View**

\[
T(-E).P = \begin{cases} 
    u'_z = N/|N| \\
    u'_x = u'_z \times (V/|V|) \\
    u'_y = u'_x \times u'_z 
\end{cases}
\]
View Transformation

- Eye at $E=(x_0, y_0, z_0)$
  - Need not be $(0,0,0)$
- LookAt $(x_n, y_n, z_n)$ vector defines normal $N$ to the image plane
  - Need not be the Z axis
- View Up vector $V$ defines orientation of the view.
  - Need not be perpendicular to $N$
  - Need not be the Y axis
- Transformation to default OpenGL View

$$R(N,V).T(-E).P$$
**gluLookAt**

- **gluLookAt**
  - Eye coordinate (E)
  - Look At vector – where normal meets the plane
    - Find N and n
  - View Up Vector (V)
- Generates this matrix and premultiplies with modelview matrix

\[
R(N,V).T(-E). P = P_M
\]
Perspective Projection
Perspective Projection

- Eye (E) : (0, 0, 0)
- View Up Vector (V) : (0, 1, 0)
- LookAt
  - Normal to the Image Plane (N) : (0,0,1)
  - Distance to the Image Plane : n
- View Direction
  - Mimics eye movement after head is fixed
Perspective Projection

\[
x_p / x = y_p / y = z_p / z
\]

\[
x_p = \frac{x}{n} \quad y_p = \frac{y}{n}
\]

(0,0,0) (0,0,z_p) (0,0,z)

E P

(0,0,0) (0,0,z)
Perspective Projection

$$\begin{align*}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & \frac{1}{n}
\end{align*}$$

$$\begin{align*}
x & \Rightarrow & x \\
y & \Rightarrow & y \\
z & \Rightarrow & z \\
z/n & \Rightarrow & \frac{z}{n} \\
1 & \Rightarrow & 1
\end{align*}$$
Perspective Projection

$$M(n) \cdot P_M = P_p$$
View Direction

\[ (0,0,z_p) \]

\[ (x_v, y_v, z_p) \]
Projection Matrix

- Make the view direction coincident with negative z-axis
- Shear matrix

\[
\begin{bmatrix}
1 & 0 & x_v/n & 0 \\
0 & 1 & y_v/n & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[\text{Sh}(x_v/n, y_v/n) = \]
Projection Matrix

- $x_v$ and $y_v$ are given in terms of center of a window
  - Extends in x direction from r to l
  - Extends in y direction from t to b

$$\text{Sh}(x_v/n, y_v/n) = \begin{bmatrix}
1 & 0 & x_v/n & 0 \\
0 & 1 & y_v/n & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$
View Direction

\[(0,0,z_p)\]

\[(x_v, y_v, z_p)\]
Projection Matrix

- $x_v$ and $y_v$ are given in terms of center of a window
  - Extends in x direction from $r$ to $l$
  - Extends in y direction from $t$ to $b$

$$\text{Sh}((r+l)/2n, (t+b)/2n) = \begin{bmatrix}
1 & 0 & r+l/2n & 0 \\
0 & 1 & t+b/2n & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$
Projection Matrix

- $x_v$ and $y_v$ are given in terms of center of a window
  - Extends in x direction from r to l
  - Extends in y direction from t to b

\[
M(n) \cdot Sh(r+l, t+b) \cdot P_M = P_p
\]
Projection Matrix

- Cannot determine the size of the framebuffer since it is dependent on r, l, t, b
  - Normalize the window to map [r, l] and [t, b] to [-1, +1]
- Scaling Matrix

\[
M(n). \text{Sc}
\begin{pmatrix}
2 & 2 \\
\frac{r-1}{r-l} & \frac{t-b}{2n}
\end{pmatrix}
\text{Sh}(r+l, t+b).P_M = P_p
\]
Projection Matrix

- With this transformation
  - x and y coordinates map between -1 to +1
  - But z maps to n
  - Since we are generating a 2D image with the image plane at depth n

\[ M(n) \cdot Sc\left(\frac{2}{r-l}, \frac{2}{t-b}\right) \cdot Sh\left(\frac{r+l}{2n}, \frac{t+b}{2n}\right) \cdot P_M = P_p \]
Problem with non-unique z

- Mathematically correct
- We would like to resolve occlusion using z
  - Option 1: Object space – render from back to front
    - Does not work for intersecting objects
  - Option 2: Screen space – resolve occlusion while rasterization
    - Need to maintain proper z for triangle for screen space z interpolation
    - Encode this information in the z after transformation
How to do this?

This is the correct perspective transform

\[
\begin{bmatrix}
  x \\ y \\ z \\ 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
  x_p \\ y_p \\ -n \\ 1
\end{bmatrix}
\]

We would like to retain the value of \( z \). We are only changing the value of \( z \), which is anyway not useful for 2D image generation using perspective projection.
Screen Space Interpolation

- Linear interpolation of $z$ in screen space must give the linear interpolation of points in object space

\[
\frac{X_t}{Z_t} = \frac{X_0 + t(X_1 - X_0)}{Z_0 + t(Z_1 - Z_0)} = s_0 + t(s_1 - s_0)
\]

This does not hold!
Screen Space Interpolation

- Linear interpolation of $z$ in screen space must give the linear interpolation of points in object space

\[ \frac{X_t}{Z_t} = \frac{X_0 + t(X_1 - X_0)}{Z_0 + t(Z_1 - Z_0)} = s_0 + u(s_1 - s_0) \]

\[ u = \frac{Z_1 t}{Z_0 (1-t) + t Z_1} \]
Screen Space Interpolation

- Correct interpolation
  - Reciprocal of $Z$
  - Interpolate in screen space
  - Take reciprocal again

\[
\frac{1}{Z_t} = \frac{1}{Z_0} (1-u) + \frac{1}{Z_1} u
\]
Transforming $z$ to $1/z$

Instead of this ...

we would like to store $1/z$ for interpolation purposes
Normalizing $1/z$

- **Unbounded** $-1/z$
  - Define far plane at *distance* $f$
- **Bound** $-1/n$ and $-1/f$ between $-1$ to $+1$
  - Three steps only on z coordinates
    - Translate the center between $-1/n$ and $-1/f$ to origin
      - $T(tz)$ where $tz = (1/n+1/f)/2$
    - Scale it to match $-1$ to $+1$
      - $S(sz)$ where $sz = 2/(1/n-1/f)$
- **Whole z transform**
  - $(1/z + tz)sz = 1/z(2nf/(f-n)) + (f+n)/(f-n)$
Complete Transformation

- $M$ and the $1/z$ normalization can be combined to one matrix $D(n,f)$

$$M(n) . Sc(2^{\frac{r-l}{2n}}, 2^{\frac{t-b}{2n}}) . Sh(r+1, t+b) . P_M = P_p$$
Complete Transformation

- `glFrustum(r, l, t, b, n, f)`

\[
D(n,f). \text{Sc}(2, 2). \text{Sh}(\frac{r+l}{2n}, \frac{t+b}{2n}). P_M = P_p
\]

\[
D(n,f) = \begin{bmatrix}
    n & 0 & 0 & 0 \\
    0 & n & 0 & 0 \\
    0 & 0 & f+n & 2nf \\
    f-n & f-n & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\]
gluPerspective

- Difference between gl and glu functions
- gluPerspective(vertical fov, aspect ratio, near, far)
  - Calls glfrustum
  - Near and far pass directly
  - \( t = n \tan(\text{v-fov}/2), \ b = -t \)
  - \( r = t \times \text{aspect ratio}, \ l = -r \)
Final Drawing

Transform all vertices;
Clear frame buffer;
Clear depth buffer;
for $i=1:n$ triangles
    for all pixels $(x_s,y_s)$ in the triangle
        pixelz = $1/z$ interpolated from vertex;
        if (pixelz < depthbuffer$[x_s][y_s]$)
            framebuffer$[x_s][y_s]$ = color interpolated from vertex attributes;
        endif;
    endfor;
endfor;
Perspective Projection
Perpendicular Parallel Projection

When eye is at infinity