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)
- Normal to image plane is not Z, but arbitrary N
  - Normal meets image plane at (x_n, y_n, z_n)
- View Up V is not Y
  - Not perpendicular to N
- Transformation to default OpenGL View

\[
\begin{align*}
  &u'_z = \frac{N}{|N|} \\
  &u'_x = \left(\frac{V}{|V|}\right) \times u'_z \\
  &u'_y = u'_z \times u'_x
\end{align*}
\]
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

- 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

\[ \frac{x_p}{x} = \frac{y_p}{y} = \frac{z_p}{z} \]

\[ x_p = \frac{x}{\frac{Z}{\Pi}} \quad y_p = \frac{y}{\frac{Z}{\Pi}} \]
Aditi Majumder, CS 112 Slide 9

Perspective Projection

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
1
\end{pmatrix}
\begin{pmatrix}
x_p \\
y_p \\
z_p \\
1
\end{pmatrix}
\]

\[=\]

Aditi Majumder, CS 112 Slide 10

Perspective Projection

\[M(n).P_M = P_p\]
**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

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

\[\text{Sh}(x_v/n, y_v/n) = \begin{pmatrix}
1 & 0 & x_v/n & 0 \\
0 & 1 & y_v/n & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}\]
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 \(b\)

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

\[
\text{Sh}(\frac{x_v}{n}, \frac{y_v}{n}) = 
\]

View Direction
**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 $b$

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

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

\[
M(n) \cdot \text{Sc}(\frac{2}{r-l}, \frac{2}{t-b}) \cdot \text{Sh}(r+l, t+b) \cdot \text{PM} = \text{P}_p
\]

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

\[
M(n) \cdot \text{Sc}(\frac{2}{r-l}, \frac{2}{t-b}) \cdot \text{Sh}(r+l, t+b) \cdot \text{PM} = \text{P}_p
\]
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?

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

This is the correct perspective transform

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

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

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

$$
u = \frac{Z_1 t}{Z_0(1-t) + tZ_1}
$$

Aditi Majumder, CS 112
Screen Space Interpolation

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

\[
\frac{1}{Z_t} = \frac{1}{Z_o} (1-u) + \frac{1}{Z_1} u
\]

Transforming z to 1/z

Instead of this ...

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

we would like to store 1/z for interpolation purposes

\[
\begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
\begin{bmatrix}
 x_p \\
 y_p \\
 -1/z \\
 1
\end{bmatrix}
\]
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 $D(n,f)$

\[
M(n).Sc(2,2).Sh(r+1, t+b).P_M = P_p
\]

Aditi Majumder, CS 112
Complete Transformation

- \textbf{glFrustum}(r, l, t, b, n, f)

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

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

- \textbf{gluPerspective}

- Difference between \textit{gl} and \textit{glu} functions
- \textbf{gluPerspective}(vertical fov, aspect ratio, near, far)
  - Calls \textbf{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

Oblique Parallel Projection

When eye is at infinity