Perspective Projection

- Projection – map the object from 3D space to 2D screen

```
Perspective()
Frustrum()
```
Perspective Projection: Classical

- Side view:

  Based on similar triangles:

  \[
  \frac{y'}{y} = \frac{-N}{z} \]

  \[
  y' = y \times \frac{-N}{z} \]
Perspective Projection: Classical

- So \((x^*, y^*)\) projection of point, \((x, y, z)\) unto near plane

  \[ (x^*, y^*) = \left( x \frac{N}{-z}, y \frac{N}{-z} \right) \]

- Numerical example:

  Q. Where on the viewplane does \(P = (1, 0.5, -1.5)\) lie for a near plane at \(N = 1\)?

  \[ (x^*, y^*) = \left( x \frac{N}{-z}, y \frac{N}{-z} \right) = \left( 1 \times \frac{1}{1.5}, 0.5 \times \frac{1}{1.5} \right) = (0.666, 0.333) \]
Pseudodepth

- Classical perspective projection projects \((x,y)\) coordinates to \((x^*, y^*)\), drops \(z\) coordinates

- But we need \(z\) to find closest object (depth testing)
- Actual \(z\) distance of \(P\) from eye could be large and cumbersome

\[ \text{distance} = \sqrt{x^2 + y^2 + z^2} \]

- Introduce pseudodepth: Transform actual \(z\) distance to a range of \(z = [-1,1]\), which can be used for depth testing

\[ \text{Map to same } (x^*, y^*) \]

\[ \text{Compare their } z \text{ values} \]

\[ \text{(0,0,0)} \]

\[ \text{VRP} \]

\[ \text{COP} \]

\[ \text{Object in 3 space} \]

\[ \text{Projected image} \]

\[ \text{Projectors} \]
Perspective Transformation

- **Perspective transformation** maps actual $z$ distance of perspective view volume to range of $-1$ to $1$ (Pseudodepth) for canonical view volume.

We want perspective transformation and NOT classical projection!!

Set scaling $z$

$\text{Pseudodepth} = az + b$

Next solve for $a$ and $b$
**Perspective Transformation**

- We want to transform viewing frustum volume into canonical view volume.
Perspective Transformation using Pseudodepth

\[(x^*, y^*, z^*) = \left( \frac{N}{x - z}, \frac{N}{y - z}, \frac{az + b}{-z} \right)\]

- Choose \(a, b\) so as \(z\) varies from \textbf{Near} to \textbf{Far}, pseudodepth varies from \(-1\) to \(1\) (canonical cube)

- Boundary conditions
  - \(z^* = -1\) when \(z = -N\)
  - \(z^* = 1\) when \(z = -F\)
Transformation of $z$: Solve for $a$ and $b$

- **Solving:**
  
  $$z^* = \frac{az + b}{-z}$$

- **Use boundary conditions**
  - $z^* = -1$ when $z = -N$........(1)
  - $z^* = 1$ when $z = -F$...........(2)

- **Set up simultaneous equations**

  $$-1 = \frac{-aN + b}{N} \Rightarrow -N = -aN + b........(1)$$

  $$1 = \frac{-aF + b}{F} \Rightarrow F = -aF + b........(2)$$
Transformation of z: Solve for a and b

\[-N = -aN + b \quad \ldots \ldots (1)\]
\[F = -aF + b \quad \ldots \ldots (2)\]

- Multiply both sides of (1) by -1
  \[N = aN - b \quad \ldots \ldots (3)\]

- Add eqns (2) and (3)
  \[F + N = aN - aF\]
  \[\Rightarrow a = \frac{F + N}{N - F} = \frac{-(F + N)}{F - N} \quad \ldots \ldots (4)\]

- Now put (4) back into (3)
Transformation of z: Solve for a and b

- Put solution for $a$ back into eqn (3)

\[ N = aN - b \ldots \ldots (3) \]

\[ \Rightarrow N = \frac{-N(F + N)}{F - N} - b \]

\[ \Rightarrow b = -N - \frac{-N(F + N)}{F - N} \]

\[ \Rightarrow b = \frac{-N(F - N) - N(F + N)}{F - N} = \frac{-NF - N^2 - NF + N^2}{F - N} = \frac{-2NF}{F - N} \]

- So

\[ a = \frac{-(F + N)}{F - N} \]

\[ b = \frac{-2FN}{F - N} \]
What does this mean?

- Original point $z$ in original view volume, transformed into $z^*$ in canonical view volume

$$z^* = \frac{az + b}{-z}$$

- where

$$a = \frac{-(F + N)}{F - N}$$
$$b = \frac{-2FN}{F - N}$$
Homogenous Coordinates

○ Want to express projection transform as 4x4 matrix
○ Previously, homogeneous coordinates of
  \[ P = (P_x, P_y, P_z) \Rightarrow (P_x, P_y, P_z, 1) \]
○ Introduce arbitrary scaling factor, \( w \), so that
  \[ P = (wP_x, wP_y, wP_z, w) \quad (\text{Note: } w \text{ is non-zero}) \]
○ For example, the point \( P = (2, 4, 6) \) can be expressed as
  ○ \((2, 4, 6, 1)\)
  ○ or \((4, 8, 12, 2)\) where \( w = 2 \)
  ○ or \((6, 12, 18, 3)\) where \( w = 3 \), or….
○ To convert from homogeneous back to ordinary coordinates, first divide all four terms by \( w \) and discard 4\textsuperscript{th} term.
Perspective Projection Matrix

- Recall Perspective Transform

\[ (x^*, y^*, z^*) = \left( \frac{x}{-z}, \frac{N}{-z}, \frac{az + b}{-z} \right) \]

- We have:

\[ x^* = x \frac{N}{-z} \quad y^* = y \frac{N}{-z} \quad z^* = \frac{az + b}{-z} \]

- In matrix form:

\[
\begin{pmatrix}
N & 0 & 0 & 0 & wX \\
0 & N & 0 & 0 & wY \\
0 & 0 & a & b & wz \\
0 & 0 & -1 & 0 & w
\end{pmatrix}
\begin{pmatrix}
wX \\
wY \\
wz \\
w
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
x \frac{N}{-z} \\
y \frac{N}{-z} \\
w(az + b) \\
-wz \\
-1
\end{pmatrix}
\]

Transformed Vertex after dividing by 4th term
**Perspective Projection Matrix**

\[
\begin{pmatrix}
N & 0 & 0 & 0 \\
0 & N & 0 & 0 \\
0 & 0 & a & b \\
0 & 0 & -1 & 0
\end{pmatrix}
\begin{pmatrix}
wP_x \\
wP_y \\
wP_z \\
w
\end{pmatrix}
= \begin{pmatrix}
wNP_x \\
wNP_y \\
w(aP_z + b) \\
-wP_z
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
x \\
y \\
N \\
-1
\end{pmatrix}
\]

\[
a = \frac{-(F + N)}{F - N}
\quad b = \frac{-2FN}{F - N}
\]

- In perspective transform matrix, already solved for \(a\) and \(b\):
- So, we have transform matrix to transform \(z\) values
Perspective Projection

- Not done yet!! Can now transform z!
- Also need to transform the \( x = (\text{left}, \text{right}) \) and \( y = (\text{bottom}, \text{top}) \) ranges of viewing frustum to \([-1, 1]\)
- Similar to glOrtho, we need to translate and scale previous matrix along x and y to get final projection transform matrix
  - we translate by
    - \(-(\text{right} + \text{left})/2\) in x
    - \-(\text{top} + \text{bottom})/2\) in y
  - Scale by:
    - \(2/(\text{right} − \text{left})\) in x
    - \(2/(\text{top} − \text{bottom})\) in y
Perspective Projection

- Translate along x and y to line up center with origin of CVV
  - -(right + left)/2 in x
  - -(top + bottom)/2 in y

- Multiply by translation matrix:

\[
\begin{pmatrix}
1 & 0 & 0 & - (\text{right} + \text{left}) / 2 \\
0 & 1 & 0 & - (\text{top} + \text{bottom}) / 2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Perspective Projection

- To bring view volume size down to size of of CVV, scale by
  - \( \frac{2}{\text{right} - \text{left}} \) in \( x \)
  - \( \frac{2}{\text{top} - \text{bottom}} \) in \( y \)

- Multiply by scale matrix:

\[
\begin{pmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & 0 \\
0 & \frac{2}{\text{top} - \text{bottom}} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

Scale size down along \( x \) and \( y \)
Perspective Projection Matrix

\[
\begin{pmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & 0 \\
0 & \frac{2}{\text{top} - \text{bottom}} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\times
\begin{pmatrix}
1 & 0 & 0 & -\left(\text{right} + \text{left}\right)/2 \\
0 & 1 & 0 & -\left(\text{top} + \text{bottom}\right)/2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\times
\begin{pmatrix}
N & 0 & 0 & 0 \\
0 & N & 0 & 0 \\
0 & 0 & a & b \\
0 & 0 & -1 & 0
\end{pmatrix}
\]

\[
\begin{pmatrix}
\frac{2N}{x_{\text{max}} - x_{\text{min}}} & 0 & \frac{\text{right} + \text{left}}{\text{right} - \text{left}} & 0 \\
0 & \frac{2N}{\text{top} - \text{bottom}} & \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} & 0 \\
0 & 0 & \frac{-\left(F + N\right)}{F - N} & -2FN \\
0 & 0 & \frac{F - N}{-1} & F - N
\end{pmatrix}
\]

\text{glFrustum(left, right, bottom, top, N, F)} \quad \text{N = near plane, F = far plane}
After perspective transformation, viewing frustum volume is transformed into canonical view volume.
Geometric Nature of Perspective Transform

a) Lines through eye map into lines parallel to z axis after transform
b) Lines perpendicular to z axis map to lines perp to z axis after transform
Normalization Transformation

Original clipping volume

Original object

COP

Z = -x

Z = -far

Z = -near

Z = x

Distorted object projects correctly

X = -1

Z = 1

X = 1

New clipping volume
Implementation

- Set modelview and projection matrices in application program
- Pass matrices to shader

```c
void display( ){
    ......  
    model_view = LookAt(eye, at, up);
    projection = Ortho(left, right, bottom, top, near, far);

    // pass model_view and projection matrices to shader
    glUniformMatrix4fv(matrix_loc, 1, GL_TRUE, model_view);
    glUniformMatrix4fv(projection_loc, 1, GL_TRUE, projection);
    ......  
}
```

4x4 matrices
Variable names in shader
Implementation

- And the corresponding shader

```glsl
in vec4 vPosition;
in vec4 vColor;
Out vec4 color;
uniform mat4 model_view;
Uniform mat4 projection;

void main( )
{
    gl_Position = projection*model_view*vPosition;
    color = vColor;
}
```
References

- Interactive Computer Graphics (6\textsuperscript{th} edition), Angel and Shreiner
- Computer Graphics using OpenGL (3\textsuperscript{rd} edition), Hill and Kelley