Parallel Projection

- normalization ⇒ find 4x4 matrix to transform user-specified view volume to canonical view volume (cube)

\[ \text{glOrtho}(\text{left, right, bottom, top, near, far}) \]
Parallel Projection: Ortho

- Parallel projection: 2 parts
  1. **Translation**: centers view volume at origin

![Diagram showing parallel projection with translation example]
Parallel Projection: Ortho

2. **Scaling**: reduces user-selected cuboid to canonical cube (dimension 2, centered at origin)
Parallel Projection: Ortho

- Translation lines up midpoints: E.g. midpoint of $x = (\text{right} + \text{left})/2$
- Thus translation factors:
  $-(\text{right} + \text{left})/2, -(\text{top} + \text{bottom})/2, -(\text{far}+\text{near})/2$
- Translation matrix:

$$
\begin{pmatrix}
1 & 0 & 0 & -(\text{right} + \text{left})/2 \\
0 & 1 & 0 & -(\text{top} + \text{bottom})/2 \\
0 & 0 & 1 & -(\text{far} + \text{near})/2 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$
Parallel Projection: Ortho

- Scaling factor: ratio of ortho view volume to cube dimensions
- Scaling factors: $2/(\text{right} - \text{left}), \ 2/(\text{top} - \text{bottom}), \ 2/(\text{far} - \text{near})$
- Scaling Matrix $M_2$:

$$
\begin{pmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & 0 & 0 \\
0 & \frac{2}{\text{top} - \text{bottom}} & 0 & 0 & 0 \\
0 & 0 & \frac{2}{\text{far} - \text{near}} & 0 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$
Parallel Projection: Ortho

Concatenating **Translation** \( \times \) **Scaling**, we get Ortho Projection matrix

\[
P = ST = \begin{pmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & 0 & - \frac{\text{right} + \text{left}}{2} \\
0 & \frac{2}{\text{top} - \text{bottom}} & 0 & 0 & - \frac{\text{top} + \text{bottom}}{2} \\
0 & 0 & \frac{2}{\text{far} - \text{near}} & 0 & - \frac{\text{far} + \text{near}}{2} \\
0 & 0 & 0 & 1 & 1
\end{pmatrix}
\]
Final Ortho Projection

- Set $z = 0$
- Equivalent to the homogeneous coordinate transformation
  
  $M_{\text{orth}} = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}$

- Hence, general orthogonal projection in 4D is
  
  $P = M_{\text{orth}}ST$
Perspective Projection

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

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

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 \(N\) is given as:

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

- 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)!!!
Perspective Transformation

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

We want perspective Transformation and NOT classical projection!!

Set scaling $z$

$Pseudodepth = az + b$

Next solve for $a$ and $b$
We want to transform viewing frustum volume into canonical view volume.
Perspective Transformation using Pseudodepth

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

- Choose \(a, b\) so as \(z\) varies from Near to 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
  - \[ z^* = -1 \text{ when } z = -N \ldots (1) \]
  - \[ z^* = 1 \text{ when } z = -F \ldots (2) \]

- Set up simultaneous equations

  \[ -1 = \frac{-aN + b}{N} \Rightarrow -N = -aN + b \ldots (1) \]

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

\[-N = -aN + b \ldots \ldots \text{(1)}\]

\[F = -aF + b \ldots \ldots \text{(2)}\]

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

- Add eqns (2) and (3)
  \[F + N = aN - aF\]

  \[\Rightarrow a = \frac{F + N}{N - F} = \frac{- (F + N)}{F - N} \ldots \ldots \text{(4)}\]

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

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

$$N = aN - b \quad \ldots \ldots (3)$$

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

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

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

- So

$$a = \frac{-(F + N)}{F - N} \quad \quad \quad b = \frac{-2FN}{F - N}$$
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
  \[ P = (wP_x, wP_y, wP_z, w) \]  \( \text{Note: } w \text{ is non-zero} \)
- 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 4th term
Perspective Projection Matrix

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

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

- In matrix form:
  \[
  \begin{pmatrix}
  N & 0 & 0 & 0 \\
  0 & N & 0 & 0 \\
  0 & 0 & a & b \\
  0 & 0 & -1 & 0 \\
  \end{pmatrix}
  \begin{pmatrix}
  wx \\
  wy \\
  wz \\
  w \\
  \end{pmatrix}
  =
  \begin{pmatrix}
  wNx \\
  wNy \\
  w(az + b) \\
  -wz \\
  \end{pmatrix}
  \Rightarrow
  \begin{pmatrix}
  x \frac{N}{-z} \\
  y \frac{N}{-z} \\
  az + b \\
  wz \\
  \end{pmatrix}
  \]

  Perspective Transform Matrix  Original vertex  Transformed Vertex  Transformed Vertex after dividing by 4th term
Perspective Projection Matrix

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

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

- 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 \( x = (\text{left, right}) \) and \( y = (\text{bottom, top}) \) 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
  - \(-\frac{(right + left)}{2}\) in x
  - \(-\frac{(top + bottom)}{2}\) in y

- Multiply by translation matrix:

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

Line up centers along x and y
**Perspective Projection**

- To bring view volume size down to size 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

$$\text{glFrustum}(\text{left}, \text{right}, \text{bottom}, \text{top}, \text{N}, \text{F}) \quad \text{N = near plane, F = far plane}$$
Perspective Transformation

- After perspective transformation, viewing frustum volume is transformed into canonical view volume

Canonical View Volume

$(-1, -1, 1)$

$(1, 1, -1)$
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

COP

original object

z = -x

z = -far

z = -near

x = -1

x = 1

z = 1

z = -1

new clipping volume

distorted object projects correctly
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);
    ....
}
```
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