Parallel Projection

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

glOrtho(left, right, bottom, top, near, far)
**Parallel Projection: Ortho**

- Parallel projection: 2 parts
  1. **Translation**: centers view volume at origin
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: \( \frac{2}{\text{right} - \text{left}}, \frac{2}{\text{top} - \text{bottom}}, \frac{2}{\text{far} - \text{near}} \)
- Scaling Matrix M2:

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

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

$$
\begin{bmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & 0 \\
0 & \frac{2}{\text{top} - \text{bottom}} & 0 & 0 \\
0 & 0 & \frac{2}{\text{far} - \text{near}} & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\times
\begin{bmatrix}
1 & 0 & 0 & - (\text{right} + \text{left}) / 2 \\
0 & 1 & 0 & - (\text{top} + \text{bottom}) / 2 \\
0 & 0 & 1 & - (\text{far} + \text{near}) / 2 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
$$

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

- Set \( z = 0 \)
- Equivalent to the homogeneous coordinate transformation
  \[
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}
  \]

\( M_{\text{orth}} = \)

- 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

Map to same \((x^*,y^*)\)  
Compare their \(z\) values

- But we need \(z\) to find closest object (depth testing)!!!
**Perspective Transformation**

- **Perspective transformation** maps actual z distance of perspective view volume to range $[-1 \text{ to } 1]$ (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

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

- 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
  - \( z^* = -1 \) when \( z = -N \) \ldots \ldots (1)
  - \( z^* = 1 \) when \( z = -F \) \ldots \ldots (2)

- Set up simultaneous equations

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

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

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

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

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

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

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

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

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

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

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

- 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
  \[ P = (Px,Py,Pz) \Rightarrow (Px,Py,Pz,1) \]
- Introduce arbitrary scaling factor, \( w \), so that
  \[ P = (wPx, wPy, wPz, w) \quad \text{(Note: } w \text{ is non-zero)} \]
- 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

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

- We have:

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

- In matrix form:

\[
\begin{bmatrix}
N & 0 & 0 & 0 & wNx \\
0 & N & 0 & 0 & wNy \\
0 & 0 & a & b & w(az + b) \\
0 & 0 & -1 & 0 & -wz \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
= \begin{bmatrix}
x \frac{N}{-z} \\
y \frac{N}{-z} \\
z \frac{az + b}{-z}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\begin{bmatrix}
wNx \\
wNy \\
w(az + b) \\
-wz
\end{bmatrix}
\]

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

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

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

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

- Scale by:
  - \(\frac{2}{\text{right} - \text{left}}\) in x
  - \(\frac{2}{\text{top} - \text{bottom}}\) 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:

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

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

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

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

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

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

- 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 = -near

z = -far

distorted object projects correctly

x = -1

new clipping volume

z = -1

z = 1

x = 1
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