Computer Graphics

Viewing Transformations and Projection

Based on slides by Dianna Xu, Bryn Mawr College
Parallel Projection Clipping View Volume

- View Volume determined by the direction of projection and the window
Parallel Projection View Volume

- View Volume is now a parallelepiped.
The Synthetic Camera

- Translated via \textit{CP} changes.
- Rotated via \textit{UP} changes.
- Redirected via \textit{View Plane Normal} changes (e.g. panning).
- Zoom via changes in \textit{View Distance}
3D Viewing Pipeline

- Modeling transformation
- Viewing transformation
- Clipping transformation
- Clip
- Projection (homogeneous division)
- Image transformation
- NDC to physical device coordinates

3D WORLD

3D World

3D eye

3D clip

3D clip

3D clip

3D NDC

3D NDC

2D SCREEN

“Standard View Volume”
Transform World Coordinates to Eye Coordinates

Approximate steps:

• Put eye (center of projection) at (0, 0, 0).
• Make X point to right.
• Make Y point up.
• Make Z point forward (away from eye in depth).
• (This is now a left-handed coordinate system!)
World to Eye Transformation

START

View direction

Eye = center of projection
World to Eye Transformation
Translate eye to \((0, 0, 0)\)
World to Eye Transformation
Align view direction with +Z
World to Eye Transformation
Align VUP direction with +Y
World to Eye Transformation
Scale to LH coordinate system
3D Viewing Pipeline

Modeling transformation

Viewing transformation

Clipping transformation

Clip

Projection (homogeneous division)

Image transformation

NDC to physical device coordinates

2D SCREEN

3D WORLD

3D World

3D eye

3D clip

3D clip

3D clip

3D NDC

3D NDC

“Standard View Volume”
On to the Clipping Transformation

• It remains to do the transformations that put these coordinates into the clipping coordinate system
• We have to shear it to get it upright
Notice that the view pyramid is not a right pyramid. We must make it so with the shear transformation.
Scaling to Standard View Volume

\[ \begin{bmatrix} \frac{UMAX - UMIN}{2} \\ \frac{VMAX - VMIN}{2} \end{bmatrix} \cdot \begin{bmatrix} VIEWD - PR_N \\ 1 \end{bmatrix} \]

- \( Z_C = \text{VIEWD-PR}_N \)
- \( Z_C = \text{BACK-PR}_N \)
- \( Z_C = \text{FRONT-PR}_N \)
The Standard View Volume for Perspective Case

plane $Z_C = Y_C$

$(1, 1, 1)^T$

plane $Z_C = X_C$

back: $Z_C = 1$

$Z_C = \text{FRONT-PR}_N$

$\frac{1}{\text{BACK-PR}_N}$

$(1, -1, 1)^T$
Scaling to Standard View
Volume: Parallel
The Standard View Volume for Parallel: The Unit Cube $[0, 1]^3$
3D Viewing Pipeline

Modeling transformation

Viewing transformation

Clipping transformation

Perspective Transformation

Clip

Projection (homogeneous division)

Image transformation

NDC to physical device coordinates

“Standard View Volume”
The default projection in the eye (camera) frame is orthogonal.

For points within the default view volume:

\[ x_p = x \]
\[ y_p = y \]
\[ z_p = 0 \]
Normalization

- Most graphics systems use view normalization
  - All other views are converted to the default view by transformations that determine the projection matrix
  - Allows use of the same pipeline for all views
Homogeneous Coordinate Representation

\[
\begin{align*}
x_p &= x \\
y_p &= y \\
z_p &= 0 \\
w_p &= 1
\end{align*}
\]

default orthographic projection

\[
p_p = Mp
\]

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

In practice, we can let \( M = I \) and set the \( z \) term to zero later
Simple Perspective

- Center of projection at the origin
- Projection plane $z = d, d < 0$
Perspective Equations

Consider top and side views

\[ x_p = \frac{x}{z/d} \]
\[ y_p = \frac{y}{z/d} \]
\[ z_p = d \]
Normalize Homogeneous Coordinates (Perspective Only)

\[
x' = \frac{x}{w}
\]
\[
y' = \frac{y}{w}
\]
\[
z' = \frac{z}{w}
\]
provided \( w \neq 0 \)

Returns \( x' \) and \( y' \) in range \([-1, 1]\)

\( z' \) in range \([0, 1]\)
Homogeneous Coordinate Form

Consider \( q =Mp \) where

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0 
\end{bmatrix}
\]

\[
q = \begin{bmatrix}
x \\
y \\
z \\
z/d 
\end{bmatrix} \Rightarrow \quad p = \begin{bmatrix}
x \\
y \\
z \\
1 
\end{bmatrix}
\]
Perspective Division

- However $w \neq 1$, so we must divide by $w$ to return from homogeneous coordinates.
- This *perspective division* yields

\[
\begin{align*}
    x_p &= \frac{x}{z/d} \\
    y_p &= \frac{y}{z/d} \\
    z_p &= d
\end{align*}
\]

the desired perspective equations.
- We will consider the corresponding clipping volume with the OpenGL functions.
OpenGL Orthogonal Viewing

\[ \text{glOrtho}(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}, \text{near}, \text{far}) \]

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

near and far measured from camera
OpenGL Perspective

\texttt{glFrustum(xmin,xmax,ymin,ymax,near,far)}
Using Field of View

- With `glFrustum` it is often difficult to get the desired view
- `gluPerspective(fovy, aspect, near, far)` often provides a better interface

![Diagram showing field of view](image)

- **aspect** = \( \frac{w}{h} \)
- Front plane
Normalization

• Rather than derive a different projection matrix for each type of projection, we can convert all projections to orthogonal projections with the default view volume.
• This strategy allows us to use standard transformations in the pipeline and makes for efficient clipping.
Pipeline View

1. **modelview transformation**
2. **projection transformation**
3. **perspective division**
   - 4D $\rightarrow$ 3D
4. **clipping**
   - against default cube
5. **projection**
   - 3D $\rightarrow$ 2D

The process begins with a modelview transformation, followed by a projection transformation. The perspective division then reduces the 4D space to 3D, followed by clipping against a default cube. Finally, the projection step transforms the 3D space into a 2D representation.
Notes

• We stay in 4D homogeneous coordinates through both the modelview and projection transformations
  – Both these transformations are nonsingular
  – Default to identity matrices (orthogonal view)
• Normalization lets us clip against simple cube regardless of type of projection
• Delay final projection until end
  – Important for hidden-surface removal to retain depth information as long as possible
**Orthogonal Normalization**

```latex
\texttt{glOrtho(left,right,bottom,top,near,far)}
```

Normalization \(\Rightarrow\) find transformation to convert specified clipping volume to default

![Diagram](image)
Orthogonal Matrix

• Two steps
  – Move center to origin
    \[ T(-\frac{\text{right}+\text{left}}{2}, -\frac{\text{top}+\text{bottom}}{2}, \frac{\text{near}+\text{far}}{2}) \]
  – Scale to have sides of length 2
    \[ S(\frac{2}{\text{right}-\text{left}}, \frac{2}{\text{top}-\text{bottom}}, \frac{2}{\text{near}-\text{far}}) \]

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

\[
P = M_{\text{orth}}ST
\]
Oblique Projections

- The OpenGL projection functions cannot produce general parallel projections such as

- However if we look at the example of the cube it appears that the cube has been sheared

- Oblique Projection = Shear + Orthogonal Projection
General Shear

Diagram showing top and side views of a 3D object with clipping planes.
Shear Matrix

\( xy \) shear (\( z \) values unchanged)

\[
H(\theta, \phi) = \begin{bmatrix}
1 & 0 & -\cot \theta & 0 \\
0 & 1 & -\cot \phi & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Projection matrix

\[
P = M_{\text{orth}} H(\theta, \phi)
\]

General case:

\[
P = M_{\text{orth}} STH(\theta, \phi)
\]
Equivalency
Effect on Clipping

• The projection matrix $P = STH$ transforms the original clipping volume to the default clipping volume.

![Diagram showing the effect of the projection matrix on the clipping volume.](image-url)
Consider a simple perspective with the COP at the origin, the near clipping plane at $z = -1$, and a 90 degree field of view determined by the planes $x = \pm z, y = \pm z$. 

\begin{align*}
x &= \pm z, \\
y &= \pm z
\end{align*}
Perspective Matrices

Simple projection matrix in homogeneous coordinates

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

Note that this matrix is independent of the far clipping plane
Generalization

\[ N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix} \]

after perspective division, the point \((x, y, z, 1)\) goes to

\[
\begin{align*}
    x'' &= -\frac{x}{z} \\
    y'' &= -\frac{y}{z} \\
    z'' &= -(\alpha + \frac{\beta}{z})
\end{align*}
\]

which projects orthogonally to the desired point regardless of \(\alpha\) and \(\beta\)
Picking $\alpha$ and $\beta$

If we pick

$$\alpha = \frac{\text{near} + \text{far}}{\text{near} - \text{far}}$$

$$\beta = \frac{2\text{near} \times \text{far}}{\text{far} - \text{near}}$$

the near plane is mapped to $z = -1$
the far plane is mapped to $z = 1$
and the sides are mapped to $x = \pm 1, y = \pm 1$

Hence the new clipping volume is the default clipping volume
Normalization Transformation

- Original clipping volume
- COP
- Original object
- New clipping volume
- Distorted object projects correctly

Equations:
- \( z = -x \)
- \( z = x \)
- \( z = -\text{far} \)
- \( z = -\text{near} \)
- \( x = -1 \)
- \( x = 1 \)
- \( z = 1 \)
- \( z = -1 \)
OpenGL Perspective

- `glFrustum` allows for an asymmetric viewing frustum (although `gluPerspective` does not)
OpenGL Perspective Matrix

- The normalization in \texttt{glFrustum} requires an initial shear to form a right viewing pyramid, followed by a scaling to get the normalized perspective volume. Finally, the perspective matrix results in needing only a final orthogonal transformation.

\[
P = \text{NSH}
\]

our previously defined perspective matrix

shear and scale
OpenGL Perspective Matrix

- **H** (shear): skew the point \(((\text{left}+\text{right})/2, (\text{top}+\text{bottom})/2, -\text{near})\) to \((0, 0, -\text{near})\)
- **S** (scale): scale the sides to \(x = \pm z, \ y = \pm z\)
- **N** (normalization): get the far plan to \(z = -1\) and the near plane to \(z = 1\)

\[
P = \begin{bmatrix}
\frac{-2 \times \text{near}}{\text{right} - \text{left}} & 0 & \frac{\text{right} + \text{left}}{\text{right} - \text{left}} & 0 \\
0 & \frac{-2 \times \text{near}}{\text{top} - \text{bottom}} & 0 & \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} \\
0 & 0 & \frac{2 \times \text{far} \times \text{near}}{\text{near} - \text{far}} & \frac{2 \times \text{far} \times \text{near}}{\text{far} - \text{near}} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]
Why do we do it this way?

- Normalization allows for a single pipeline for both perspective and orthogonal viewing.
- We keep in four dimensional homogeneous coordinates as long as possible to retain three-dimensional information needed for hidden-surface removal and shading.
- We simplify clipping.
A point \((x, y, z)\) is in the view volume if and only if it lies inside these 6 planes.
Recall the Standard View Volume:

\[
\begin{align*}
\text{back: } Z_C &= 1 \\
\text{plane } Z_C &= Y_C \\
(1, 1, 1)^T \\
\text{plane } Z_C &= X_C \\
(1, -1, 1)^T \\
\text{front-PR}_N/Z_C &= \text{BACK-PR}_N/Z_C \\
\end{align*}
\]
The Perspective Transformation (for the Perspective Case only)

• Now that we have a normalized perspective view volume we apply one more matrix to it in order to permit simple depth comparisons

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{1}{1-Zf_{ront}} & \frac{-Zf_{ront}}{1-Zf_{ront}} \\
0 & 0 & \frac{1}{1-Zf_{ront}} & 0
\end{bmatrix}
\]

• Where \( Z_{front} \) is the value of the front clipping z coordinate after viewing transformation --

\[
(Front - PR_N) / (Back - PR_N)
\]
What does $M$ do?

• Notice that $M$ does not affect the $x$ or $y$ coordinates.

• $M$ sets the homogeneous coordinate $w \leftarrow z$.

• $z$ is changed to lie in the range $[0, 1]$.

• Check: if $z = Z_{\text{front}}$ then new $z \leftarrow 0$; if $z = 1$ then new $z \leftarrow 1$. 
$M$ Changes Standard Pyramid to This...
$M$ Creates the Foreshortening Effect

- Thus $M$ makes the projectors parallel, allowing later depth comparisons:

\[ \text{becomes} \]

\[ \text{becomes} \]
The Perspective Transformation $M$

- Preserves relative depth.
- Preserves linearity ("straightness").
- Preserves planarity.
- Produces perspective foreshortening.
- Still permits clipping -- just use $W$ coordinate.