Computer Graphics Projection

Matthias Teschner
Homogeneous Coordinates - Summary

- $[x, y, z, w]^T$ with $w \neq 0$ are the homogeneous coordinates of the 3D position $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})^T$
- $[x, y, z, 0]^T$ is a point at infinity in the direction of $(x, y, z)^T$
- $[x, y, z, 0]^T$ is a vector in the direction of $(x, y, z)^T$
- $\begin{bmatrix} m_{00} & m_{01} & m_{02} & t_0 \\ m_{10} & m_{11} & m_{12} & t_1 \\ m_{20} & m_{21} & m_{22} & t_2 \\ p_0 & p_1 & p_2 & w \end{bmatrix}$ is a transformation that represents rotation, scale, shear, translation, projection
Outline

– Context
– Projections
– Projection transform
– Typical vertex transformations
Motivation

– 3D scene with a camera, its view volume and its projection

Orthographic projection  Perspective projection

[Song Ho Ahn]
Motivation

– Rendering generates planar views from 3D scenes
– 3D space is projected onto a 2D plane considering external and internal camera parameters
  – Position, orientation, focal length
– Projections can be represented with a matrix in homogeneous notation
Motivation

- Transformation matrix in homogeneous notation

\[
\begin{bmatrix}
m_{00} & m_{01} & m_{02} & t_0 \\
m_{10} & m_{11} & m_{12} & t_1 \\
m_{20} & m_{21} & m_{22} & t_2 \\
p_0 & p_1 & p_2 & w
\end{bmatrix}
\]

- $m_{ij}$ represent rotation, scale, shear
- $t_i$ represent translation
- $p_i$ are used in projections
- $w$ is the homogeneous component
Example

- Last matrix row can be used to realize divisions by a linear combination of multiples of $p_x, p_y, p_z, 1$

\[
p' = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
p_0 & p_1 & p_2 & w
\end{bmatrix} \begin{bmatrix}
p_x \\
p_y \\
p_z \\
1
\end{bmatrix} = \begin{bmatrix}
p_x \\
p_y \\
p_z \\
p_0 p_x + p_1 p_y + p_2 p_z + w
\end{bmatrix}
\]

~

\[
\begin{pmatrix}
p_x \\
\frac{p_0 p_x + p_1 p_y + p_2 p_z + w}{p_y} \\
\frac{p_0 p_x + p_1 p_y + p_2 p_z + w}{p_z} \\
\frac{p_0 p_x + p_1 p_y + p_2 p_z + w}{p_0 p_x + p_1 p_y + p_2 p_z + w}
\end{pmatrix}
\]
2D Illustration

\[ p' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -d & 0 \\ 1 & 0 & -d \end{bmatrix} \begin{bmatrix} wp_x \\ wp_y \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ -dwp_y \\ wp_x - wd \end{bmatrix} \approx \begin{bmatrix} 0 \\ -dwp_y/p_{x-d} \end{bmatrix} \]

\[ p' = \frac{\frac{p_y}{p_{x-d}}}{\frac{p_y}{-d}} \Rightarrow p'_y = -\frac{dp_y}{p_{x-d}} \]

\[ p_x = 0 \]

\[ v = [d, 0, 1]^T \]
Outline

– Context
– Projections
  – 2D
  – 3D
– Projection transform
– Typical vertex transformations
Setting

- A 2D projection from \( v \) onto \( l \) maps a point \( p \) onto \( p' \)
- \( p' \) is the intersection of the line through \( p \) and \( v \) with line \( l \)
- \( v \) is the viewpoint, center of perspectivity
- \( l \) is the viewline
- The line through \( p \) and \( v \) is a projector
- \( v \) is not on the line \( l, p \neq v \)
Classification

- If the homogeneous component of the viewpoint \( \mathbf{v} \) is not equal to zero, we have a perspective projection
  - Projectors are not parallel
- If \( \mathbf{v} \) is at infinity, we have a parallel projection
  - Projectors are parallel

\[
\mathbf{v} = [x, y, 1]^T
\]

\[
\mathbf{v} = [x, y, 0]^T
\]
Classification

- Location of viewpoint and orientation of the viewline determine the type of projection
  - Parallel (viewpoint at infinity, parallel projectors)
    - Orthographic (viewline orthogonal to the projectors)
    - Oblique (viewline not orthogonal to the projectors)
  - Perspective (non-parallel projectors)
    - One-point (viewline intersects one principal axis, i.e. viewline is parallel to a principal axis, one vanishing point)
    - Two-point (viewline intersects two principal axes, two vanishing points)
General Case

- A 2D projection is represented by a matrix in homogeneous notation

\[ M = vl^T - (l \cdot v)I_3 \]

\[ vl^T = \begin{bmatrix} v_xa & v_xb & v_xc \\ v_ya & v_yb & v_yc \\ v_wa & v_wb & v_wc \end{bmatrix} \]

\[ (l \cdot v)I_3 = (av_x + bv_y + cv_w) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ l = \{ax + by + c = 0\} = [a, b, c]^T \]

\[ p = [px, py, 1]^T \]

\[ r' = Mr \]

\[ s' = Ms \]
Example

\[ l = \{ x + 0y + 0 = 0 \} \]
\[ l = [1, 0, 0]^T \]
\[ p = [p_x, p_y, 1]^T \]
\[ p' = [p'_x, p'_y, 1]^T \]
\[ v = [d, 0, 1]^T \]

\[ M = \begin{bmatrix} d & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & -d \end{bmatrix} \]
\[ M = [1, 0, 0] - \left( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} d \\ 0 \\ 1 \end{bmatrix} \right) I_3 \]
\[ = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -d & 0 \\ 1 & 0 & -d \end{bmatrix} \]

\[ p' = Mp = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -d & 0 \\ 1 & 0 & -d \end{bmatrix} \begin{bmatrix} wp_x \\ wp_y \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ -dwp_y \\ wp_x - wd \end{bmatrix} = \begin{bmatrix} 0 \\ -dp_y/p_x - d \end{bmatrix} \sim \begin{bmatrix} 0 \\ -dp_y/p_x - d \end{bmatrix} \]
Discussion

- \textbf{Discussion}

- M and λM represent the same transformation \( \lambda M p = \lambda p' \)

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & -d & 0 \\
1 & 0 & -d
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
-\frac{1}{d} & 0 & 1
\end{bmatrix}
\]

are the same transformation

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & -d & 0 \\
1 & 0 & -d
\end{bmatrix}
\begin{bmatrix}
w p_x \\
w p_y \\
w
\end{bmatrix}
= \begin{bmatrix}
0 \\
-d w p_y \\
w p_x - d w
\end{bmatrix}
= \begin{bmatrix}
0 \\
-\frac{-d p_y}{p_x - d} \\
1
\end{bmatrix}
\sim \begin{bmatrix}
0 \\
-\frac{-d p_y}{p_x - d}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0 \\
-\frac{-p_y}{\frac{p_x}{d} - 1}
\end{bmatrix}
\sim \begin{bmatrix}
0 \\
\frac{-p_y}{\frac{p_x}{d} - 1}
\end{bmatrix}
= \begin{bmatrix}
0 \\
wp_y \\
-w \frac{p_x}{d} + w
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
-\frac{1}{d} & 0 & 1
\end{bmatrix}
\begin{bmatrix}
w p_x \\
w p_y \\
w
\end{bmatrix}
\]
Parallel Projection

– Moving $d$ to infinity results in parallel projection

$$\lim_{d \to \pm \infty} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{d} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

– $x$-component is mapped to zero
– $y$- and $w$-component are unchanged
Parallel Projection

\[ l = \{1x + 0y + 0 = 0\} \]
\[ l = [1, 0, 0]^T \]
\[ v = [-1, 0, 0]^T \]
\[ p' = [p'_x, p'_y, 1]^T \]
\[ p = [p_x, p_y, 1]^T \]

\[ M = vl^T - (l \cdot v)I_3 \]

\[ M = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} (1, 0, 0) - \left( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \right) I_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

X-component is mapped to zero.
Y-component is unchanged.
Discussion

− 2D transformation in homogeneous form

\[ M = \begin{pmatrix} m_{11} & m_{12} & t_1 \\ m_{21} & m_{22} & t_2 \\ p_1 & p_2 & w \end{pmatrix} \]

− \( p_1 \) and \( p_2 \) map the homogeneous component \( w \) of a point to a value \( w' \) that depends on \( x \) and \( y \)

− Therefore, the scaling of a point depends on \( x \) and / or \( y \)

− In perspective projections, this is generally employed to scale the \( x \)- and \( y \)-component with respect to \( z \), its distance to the viewer
Outline

− Context
− Projections
  − 2D
  − 3D
− Projection transform
− Typical vertex transformations
Setting

- A 3D projection from $v$ onto $l$ maps a point $p$ onto $p'$
- $p'$ is the intersection of the line through $p$ and $v$ with plane $n$
- $v$ is the viewpoint, center of perspectivity
- $n$ is the viewplane
- The line through $p$ and $v$ is a projector
- $v$ is not on the plane $n$, $p \neq v$

$n = \{ax + by + cz + d = 0\} = [a, b, c, d]^T$

$p = [p_x, p_y, p_z, 1]^T$
General Case

- A 3D projection is represented by a matrix in homogeneous notation

\[ M = vn^T - (n \cdot v)I_4 \]

\[ vn^T = \begin{bmatrix} v_x a & v_x b & v_x c & v_x d \\ v_y a & v_y b & v_y c & v_y d \\ v_z a & v_z b & v_z c & v_z d \\ v_w a & v_w b & v_w c & v_w d \end{bmatrix} \]

\[ (n \cdot v)I_4 = (av_x + bv_y + cv_z + dv_w) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ n = \{ax + by + cz + d = 0\} = [a, b, c, d]^T \]

\[ p = [p_x, p_y, p_z, 1]^T \]

\[ p' = Mp \]

\[ r' = Mr \]

\[ s' = Ms \]
Example

\[ n = \{0x + 0y + 1z + 0 = 0\} \]
\[ n = [0, 0, 1, 0]^T \]
\[ \mathbf{p} = [p_x, p_y, p_z, 1]^T \]
\[ \mathbf{p}' = [p'_x, p'_y, 0, 1]^T \]
\[ \mathbf{v} = [0, 0, d, 1]^T \]

\[
\begin{align*}
p'_x &= \frac{p_x}{p_z - d} \\
p'_y &= \frac{p_y}{p_z - d} \\
p'_z &= 0 \\
p' &= M \mathbf{p} = \\
&= \begin{bmatrix}
-d & 0 & 0 & 0 \\
0 & -d & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & -d \\
\end{bmatrix}
\begin{bmatrix}
w p_x \\
w p_y \\
w p_z \\
w \\
\end{bmatrix}
= \begin{bmatrix}
-w dp_x \\
-w dp_y \\
0 \\
-w p z \cdot dw \\
\end{bmatrix}
\sim \begin{bmatrix}
\frac{-dp_x}{p_z - d} \\
\frac{-dp_y}{p_z - d} \\
0 \\
1 \\
\end{bmatrix}
\end{align*}
\]

\[ M = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & -d \\
\end{bmatrix} \cdot (0, 0, 1, 0) - \begin{bmatrix}
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
\end{bmatrix} \cdot I_4 \]
Parallel Projection

\[ n = \{0x + 0y + 1z + 0 = 0\} \]
\[ n = [0, 0, 1, 0]^T \]
\[ v = [0, 0, -1, 0]^T \]
\[ p' = [p'_x, p'_y, 0, 1]^T \]
\[ p = [p_x, p_y, p_z, 1]^T \]

\[ M = vn^T - (n \cdot v)I_4 \]

\[ M = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} [0, 0, 1, 0] - \left( \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \right) I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

X- and y-component are unchanged. Z-component is mapped to zero.
Outline

– Context
– Projections
– Projection transform
  – Motivation
  – Perspective projection
  – Discussion
  – Orthographic projection
– Typical vertex transformations
Modelview Transform

Transformation from local into view space is realized with the modelview transform.
Objects: $V^1M_1, V^1M_2, V^1M_3$
Camera: $V^1V = I$
Projection Transform

Transformation from view space to clip space / NDC space is realized with the projection transform $P$. 

View frustum

View space / Camera space.

Canonical view volume

Clip space / NDC space.
Clip Space / NDC Space

- Allows simplified and unified implementations
  - Culling
  - Clipping
  - Visibility
    - Parallel ray casting
    - Depth test
  - Projection onto view plane / screen (viewport mapping)
Culling / Clipping / Visibility

Culling

Clipping

Visibility
Outline

– Context
– Projections
– Projection transform
  – Motivation
  – Perspective projection
  – Discussion
  – Orthographic projection
– Typical vertex transformations
Perspective Projection Transform

- Maps a view volume / pyramidal frustum to a canonical view volume
- The view volume is specified by its boundary
  - Left $l$, right $r$, bottom $b$, top $t$, near $n$, far $f$
- The canonical view volume is, e.g., a cube from $(-1,-1,-1)$ to $(1,1,1)$
Perspective Projection Transform

- Is applied to vertices
- Maps
  - The $x$-component of a projected point from (left, right) to (-1, 1)
  - The $y$-component of a projected point from (bottom, top) to (-1, 1)
  - The $z$-component of a point from (near, far) to (-1, 1)
- If a point in view space is inside / outside the view volume, it is inside / outside the canonical view volume
Derivation

\[
\begin{align*}
\frac{y_p}{n} &= \frac{y_v}{z_v} \Rightarrow y_p &= \frac{ny_v}{z_v} \\
x_p &= \frac{nx_v}{z_v}
\end{align*}
\]
Derivation

– From

\[
x_n = \frac{1}{z_v} \left( \frac{2n}{r-l} x_v - \frac{r+l}{r-l} z_v \right) \quad y_n = \frac{1}{z_v} \left( \frac{2n}{t-b} y_v - \frac{t+b}{t-b} z_v \right)
\]

we get

\[
\begin{bmatrix}
  x_c \\
  y_c \\
  z_c \\
  w_c
\end{bmatrix} =
\begin{bmatrix}
  \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\
  0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\
  . & . & . & . \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  x_v \\
  y_v \\
  z_v \\
  1
\end{bmatrix}
\]

Clip coordinates
(clip space)

with

\[
\begin{bmatrix}
  x_n \\
  y_n \\
  z_n \\
  1
\end{bmatrix} =
\begin{bmatrix}
  x_c/w_c \\
  y_c/w_c \\
  z_c/w_c \\
  w_c/w_c
\end{bmatrix}
\]

Normalized device coordinates
(NDC space)
Derivation

- $z_v$ is mapped from (near, far) or $(n, f)$ to (-1, 1)
- The transform does not depend on $x_v$ and $y_v$
- So, we have to solve for $A$ and $B$ in

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_vx_v \\ w_vy_v \\ w_vz_v \\ w_v \end{bmatrix}$$

$$z_n = \frac{z_c}{w_c} = \frac{Az_v + Bw_v}{z_v}$$
Derivation

- $z_v=n$ with $w_v=1$ is mapped to $z_n=-1$
- $z_v=f$ with $w_v=1$ is mapped to $z_n=1$

$$\Rightarrow A = \frac{f+n}{f-n} \quad \Rightarrow B = -\frac{2fn}{f-n}$$

- The complete projection matrix is

$$\begin{pmatrix}
\frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\
0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\
0 & 0 & 1 & 0
\end{pmatrix}$$
Perspective Projection Matrix

\[
P = \begin{bmatrix}
\frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & -\frac{r-l}{t+b} & 0 \\
0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
transforms the view volume, the pyramidal frustum to the canonical view volume

[Song Ho Ahn]
Outline

− Context
− Projections
− Projection transform
  − Motivation
  − Perspective projection
  − Discussion
  − Orthographic projection
− Typical vertex transformations
Symmetric Setting

– The matrix simplifies for $r=-l$ and $t=-b$

\[
\begin{align*}
  r + l &= 0 \\
  r - l &= 2r \\
  t + b &= 0 \\
  t - b &= 2t
\end{align*}
\]

\[P = \begin{bmatrix}
  \frac{n}{r} & 0 & 0 & 0 \\
  0 & \frac{n}{t} & 0 & 0 \\
  0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\
  0 & 0 & 1 & 0
\end{bmatrix}\]
Variants

- Projection matrices depend on coordinate systems and other settings
- E.g., OpenGL
  - Viewing along negative z-axis in view space
  - Negated values for $n$ and $f$

\[
P = \begin{bmatrix}
\frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & \frac{r+l}{t-b} & 0 \\
0 & 0 & \frac{f+n}{f-n} & -2fn \\
0 & 0 & -1 & 0
\end{bmatrix}
\]
Non-linear Mapping of Depth Values

\[ z_n = \frac{f+n}{f-n} - \frac{1}{z_v} \frac{2fn}{f-n} \]

- Near plane should not be too close to zero

\[ n = 9 \quad f = 10 \quad n = 1 \quad f = 10 \quad n = 0.1 \quad f = 10 \]
Non-linear Mapping of Depth Values

– Setting the far plane to infinity is not too critical

\[
P = \begin{bmatrix}
\frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & -\frac{r+l}{t+b} & 0 \\
0 & 0 & \frac{f+n}{t-b} & -2fn \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[f \to \infty\]

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

\[\Rightarrow z_n = 1 - \frac{2n}{2n}\]

\[n = 1 \quad f = \infty\]
Outline

− Context
− Projections
− Projection transform
  − Motivation
  − Perspective projection
  − Discussion
  − Orthographic projection
− Typical vertex transformations
Orthographic Projection

- View volume is a cuboid and specified by its boundary
  - Left \( l \), right \( r \), bottom \( b \), top \( t \), near \( n \), far \( f \)
- Canonical view volume is a cube from \((-1,-1,-1)\) to \((1,1,1)\)
Derivation

- All components of a point in view coordinates are linearly mapped to the range of (-1,1)

\[ x_n = \frac{2}{r-l} x_v - \frac{r+l}{r-l} \quad y_n = \frac{2}{t-b} y_v - \frac{t+b}{t-b} \quad z_n = \frac{2}{f-n} z_v - \frac{f+n}{f-n} \]

- Linear in \( x_v, y_v, z_v \)
- Combination of scale and translation
Orthographic Projection Matrix

– General form

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

– Simplified form for a symmetric view volume

\[\begin{align*}
  r + l &= 0 \\
  r - l &= 2r \\
  t + b &= 0 \\
  t - b &= 2t
\end{align*}\]

\[
\Rightarrow P = \begin{bmatrix}
\frac{1}{r} & 0 & 0 & 0 \\
0 & \frac{1}{t} & 0 & 0 \\
0 & 0 & \frac{2}{f-n} & -\frac{f+n}{f-n} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Outline

– Context
– Projections
– Projection transform
– Typical vertex transformations
Overview

\( V^{-1} M_i \)

Modelview transform depends on model \( i \).

\( P \)

Projection transform depends on camera parameters.

\[ PV^{-1} M_i \]
Coordinate Systems

<table>
<thead>
<tr>
<th>Transform Type</th>
<th>Source Space</th>
<th>Transformation</th>
<th>Target Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model transform:</td>
<td>Local space</td>
<td>→</td>
<td>Global space</td>
</tr>
<tr>
<td>View transform:</td>
<td>Local space</td>
<td>→</td>
<td>Global space</td>
</tr>
<tr>
<td>Inverse view transform:</td>
<td>Global space</td>
<td>→</td>
<td>View space</td>
</tr>
<tr>
<td>Modelview transform:</td>
<td>Local space</td>
<td>→</td>
<td>View space</td>
</tr>
<tr>
<td>Projection transform:</td>
<td>View space</td>
<td>→</td>
<td>Clip space</td>
</tr>
</tbody>
</table>
Camera Placement

\[ V = T_{\text{cam}} R_{\text{cam}} \]
Object Placement

Local space

Global space

$M_i$
View Transform

\[ V^{-1} = (T_{\text{cam}} R_{\text{cam}})^{-1} = R_{\text{cam}}^{-1} T_{\text{cam}}^{-1} = R_{\text{cam}}^{T} T_{\text{cam}}^{-1} \]
Projection Transform

View space

Clip space

P
Vertex Transforms - Summary

Transformations are applied to vertices. Internal and external camera parameters are encoded in the matrices for view and projection transform.

\[ P R_{\text{cam}}^T T_{\text{cam}}^{-1} M_i \]
References