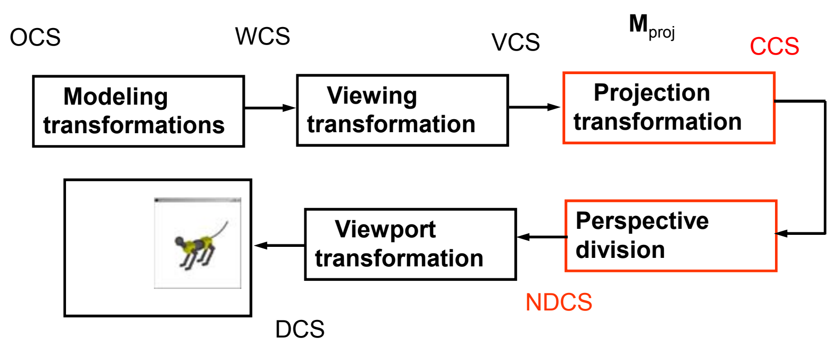
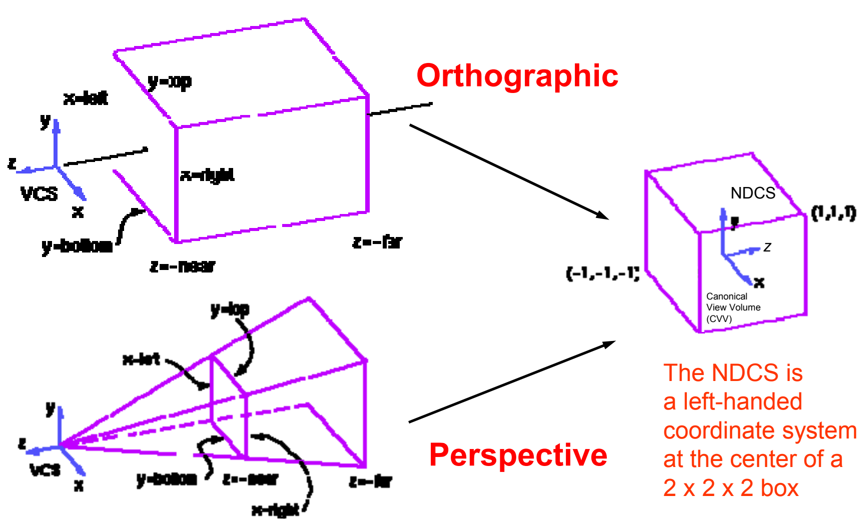
Graphics Pipeline



View Volumes

We display only the geometry in a view volume

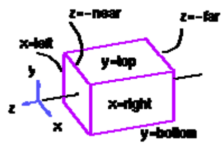


Derivation of the Orthographic Projection Matrix

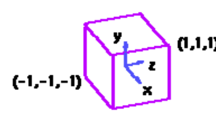
Another coordinate system transformation

- Scale 2x2x2 cube to the rectangular cuboid and flip z then translate appropriately

left: $x = l$	right: $x = r$
bottom: $y = b$	top: $y = t$
near: $z = -n$	far: $z = -f$



VCS



NDCS

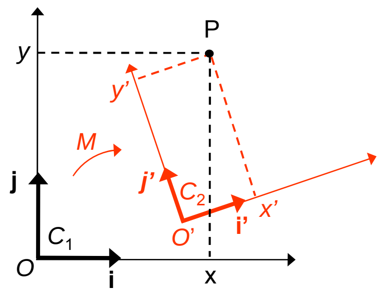
$$\mathbf{M}_O = \underbrace{\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Scaling}^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & +\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Translation}^{-1}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(note negation to flip z axis)

Explanation of the Above

Reminder: P in C_1 vs P in C_2

$C_1 \underset{T}{\mapsto} C_2$

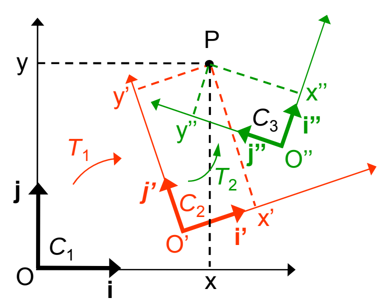


$$P_{C_1} = \mathrm{M} P_{C_2}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Reminder: Successive Coordinate System Transformations

$C_1 \underset{T_1}{\mapsto} C_2 \underset{T_2}{\mapsto} C_3$

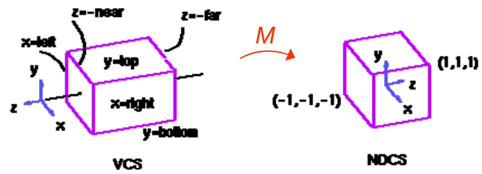


Working backwards:

$$P_{C_2} = \mathrm{M}_2 P_{C_3} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

$$P_{C_1} = \mathrm{M}_1 P_{C_2} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M}_1 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_1 \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

We Need $P_{NDCS} = M^{-1}P_{VCS}$



Obtaining M:

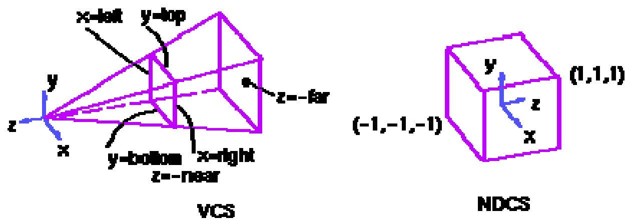
1. *Translate VCS to center of cuboid*
2. *Flip z axis and Scale cuboid to a 2 x 2 x 2 cube*

$\mathbf{M} = \mathbf{M}_T\mathbf{M}_S \quad \Rightarrow \quad \mathbf{M}^{-1} = \mathbf{M}_S^{-1}\,\mathbf{M}_T^{-1} = \mathbf{M}_O$

$$\mathbf{M}_O = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & +\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

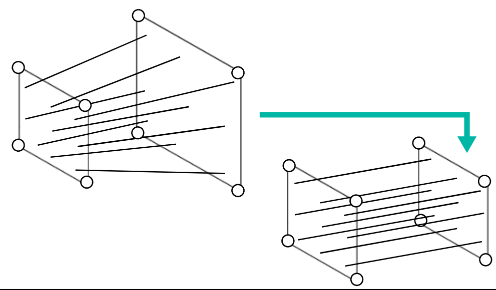
End of Explanation

Derivation of the Perspective Transformation Matrix



Maps any line through the origin (eye) to a line parallel to the z axis

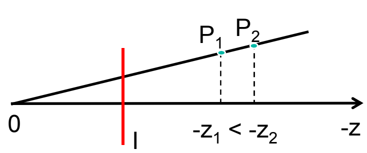
- *Without moving the point on the line at $z = -n$*
- *Leaves points on the $z = -f$ plane, while "squishing" them in x and y*

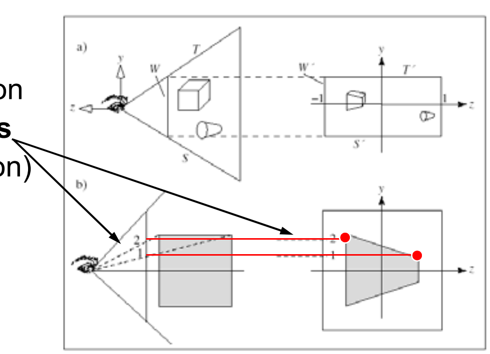


Derivation of the Perspective Transformation Matrix

This transformation warps the view volume and the objects in it

- Eye becomes a point at infinity, and the projection rays become **parallel lines** (i.e., orthographic projection)
- We also want to keep z
  - *Pseudodepth*





The Perspective Transformation Matrix

$$\mathbf{M}_P \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} nP_x \\ nP_y \\ P_z(n+f)+nf \\ -P_z \end{bmatrix} \xrightarrow[\div(h=-P_z)]{\text{homogenize}} \begin{bmatrix} -\frac{P_x n}{P_z} \\ -\frac{P_y n}{P_z} \\ -n-f-\frac{nf}{P_z} \\ 1 \end{bmatrix} = \begin{bmatrix} P'_x \\ P'_y \\ P'_z \\ 1 \end{bmatrix}$$

Therefore:

$$\mathbf{M}_P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & nf \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Note: $P'_z = \begin{cases} -n, & \text{when } P_z = -n \\ -f, & \text{when } P_z = -f \end{cases}$

The Projection Matrix

As defined by OpenGL

$$\mathbf{M}_{\text{proj}} = \mathbf{M}_O \mathbf{M}_P$$

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

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

Nonlinearity of Perspective Transformation

Railroad tracks:

z = -infinity to +infinity

Left track: x = -1, y = -1

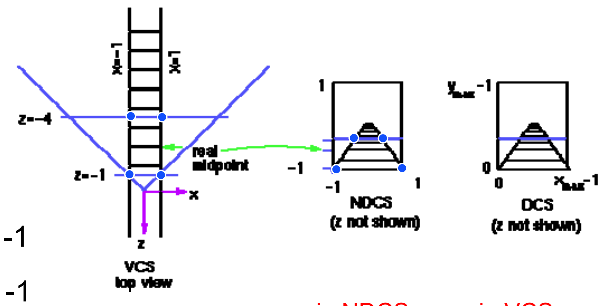
Right track: x = +1, y = -1

View volume:

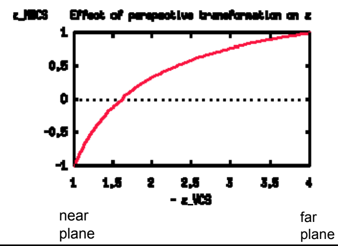
Left = -1, Right = +1

Bottom = -1, Top = +1

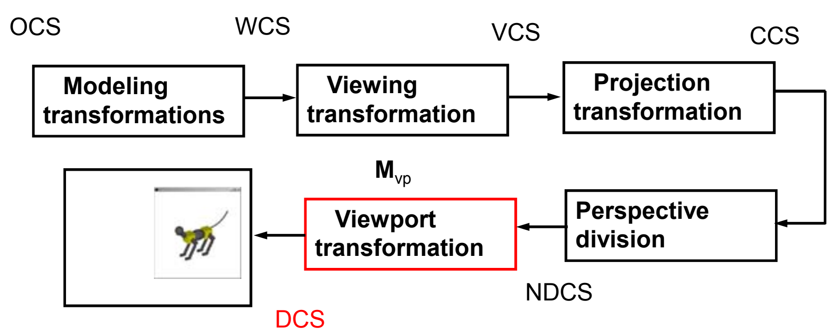
Near = +1, Far = +4



z in NDCS vs –z in VCS



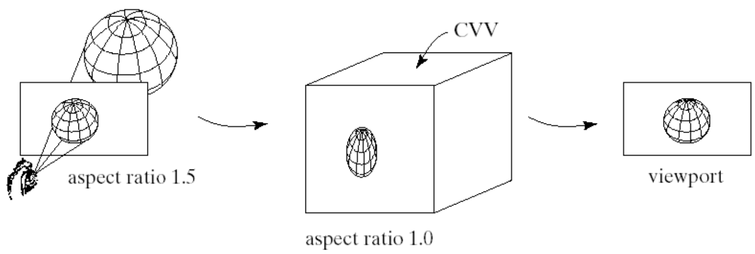
Viewport Transformation



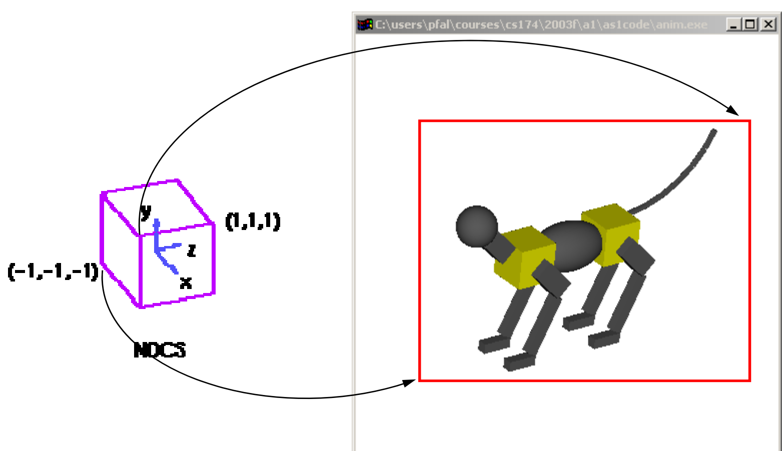
Why Viewports?

Undo the distortion of the projection transformation

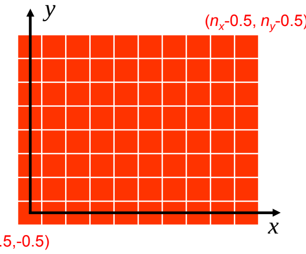
Map to pixel coordinates on screen



Viewport



Viewport Matrix

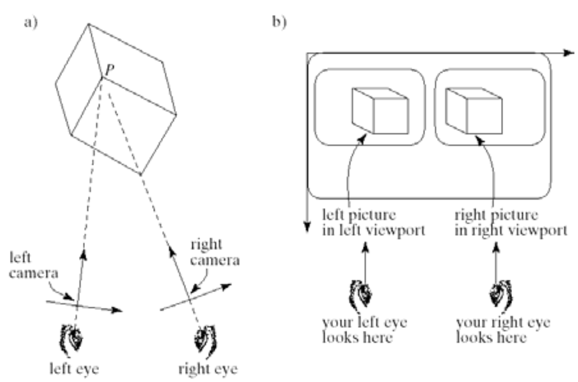


- Canonical image plane: [-1.0, -1.0] x [1.0, 1.0]
- Leave *z* coordinates unchanged
- Transform x,y coordinates to a rectangular viewport of size n_x x n_y pixels
  assume square pixels of size 1.0x1.0, from (0,0) at lower left;
  thus, viewport is [-0.5, n_x-0.5] x [-0.5, n_y-0.5]

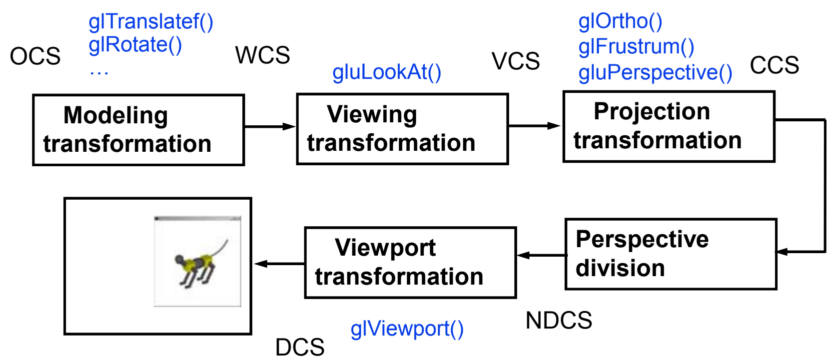
$$\mathbf{M}_{VP} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \frac{n_x-1}{2} \\ 0 & 1 & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Translation}} \underbrace{\begin{bmatrix} \frac{n_x}{2} & 0 & 0 & 0 \\ 0 & \frac{n_y}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Scaling}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What would change if *y* were increasing downward?

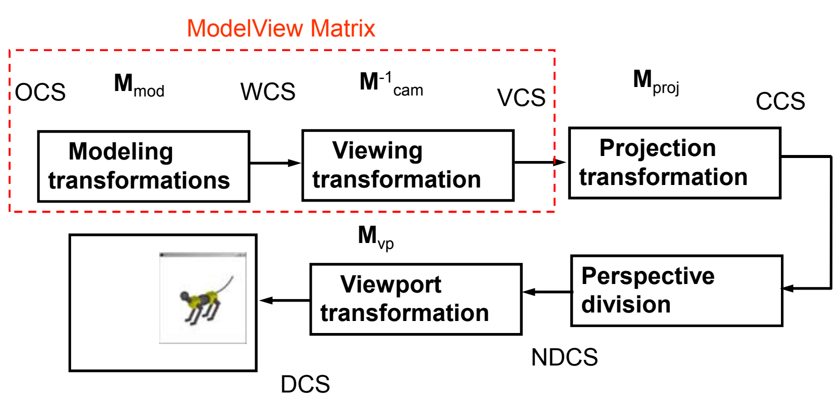
Stereo Views



OpenGL Functions in the Pipeline

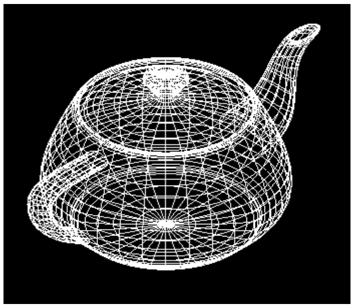


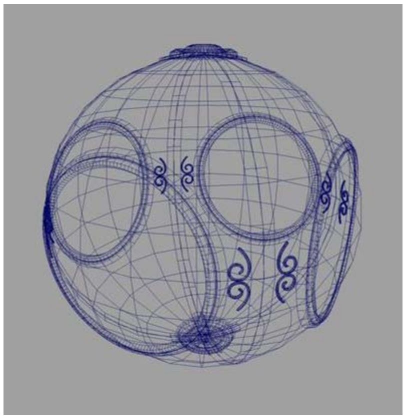
Transformations in the Pipeline



Wireframe Displays

No hidden line/surface removal





A Wireframe Rendering Algorithm

Compute $\mathbf{M}_{mod}$

Compute $\mathbf{M}^{-1}{}_{cam}$

Compute $\mathbf{M}_{modelview} = \mathbf{M}^{-1}{}_{cam}\ \mathbf{M}_{mod}$

Compute $\mathbf{M}_{O}$

Compute $\mathbf{M}_{P}$ // disregard $\mathbf{M}_P$ here and below for orthographic-only case

Compute $\mathbf{M}_{proj} = \mathbf{M}_{O}\ \mathbf{M}_{P}$

Compute $\mathbf{M}_{vp}$

Compute $\mathbf{M} = \mathbf{M}_{vp}\ \mathbf{M}_{proj}\ \mathbf{M}_{modelview}$

for each line segment *i* between points P_i and Q_i **do**

  $P = \mathbf{M}P_i$; $Q = \mathbf{M}Q_i$

  drawline(P_x/w_P, P_y/w_P, Q_x/w_Q, Q_y/w_Q) // w_P, w_Q are the 4th coords of P, Q

end for

More Complex Wireframe Displays

No hidden line/surface removal