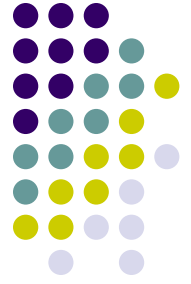


Computer Graphics
CS 543 – Lecture 6 (Part 3)
Projection (Part 2)

Prof Emmanuel Agu

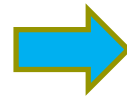
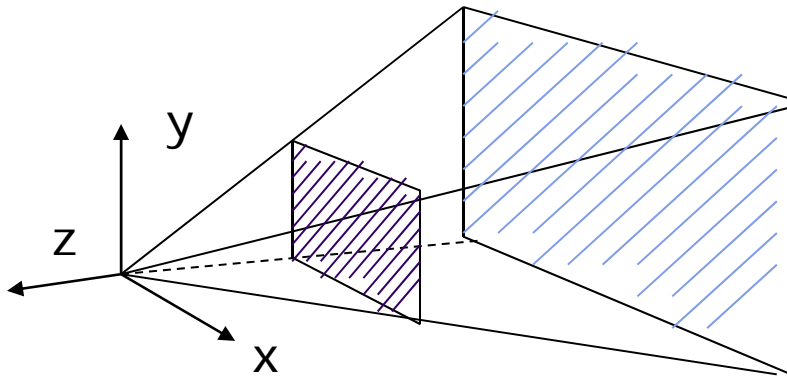
Computer Science Dept.
Worcester Polytechnic Institute (WPI)





Perspective Projection

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

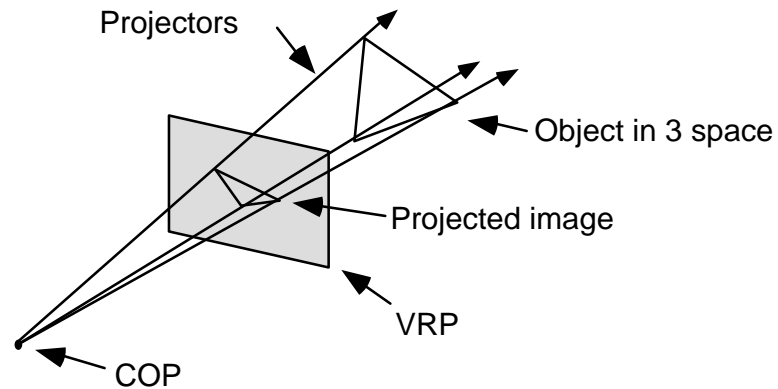
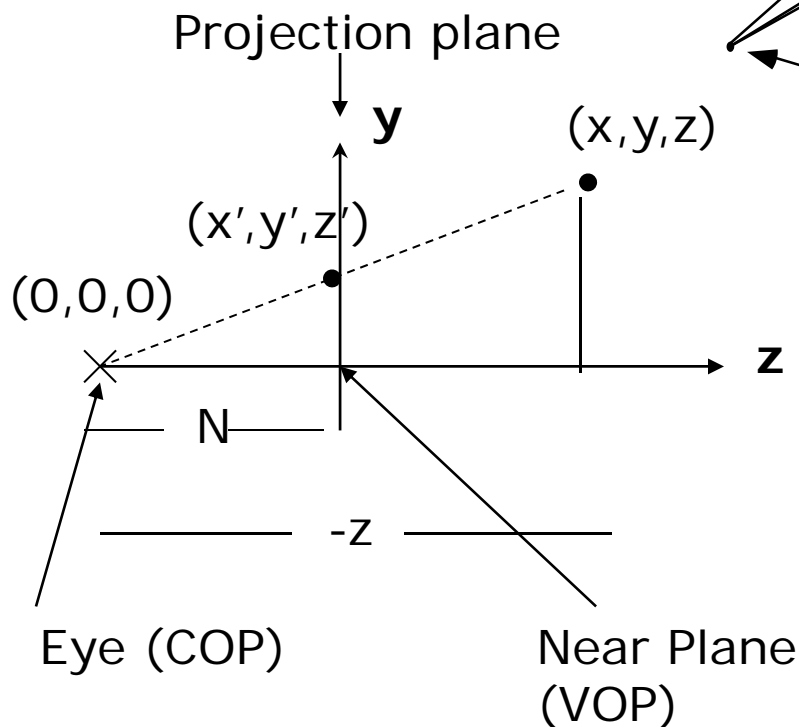


Perspective()
Frustum()



Perspective Projection: Classical

- Side view:



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:

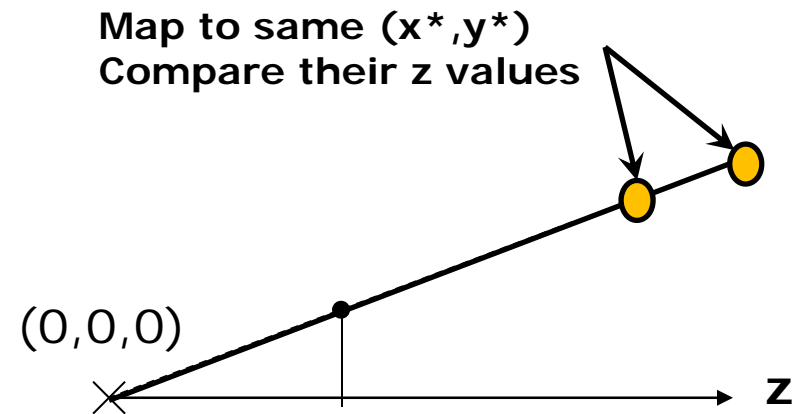
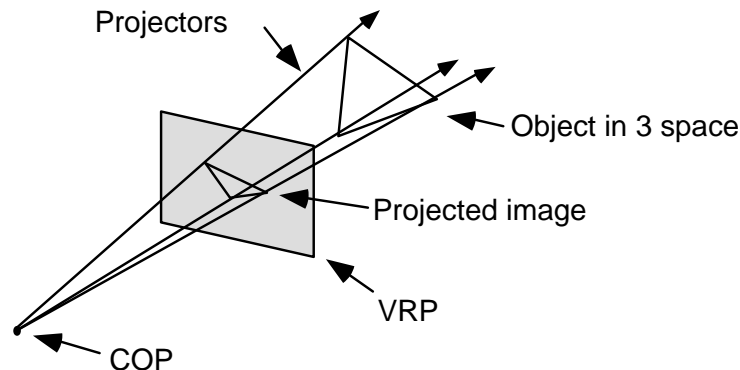
Q. Where on the viewplane does $P = (1, 0.5, -1.5)$ lie for a near plane at $N = 1$?

$$(x^*, y^*) = \left(x \frac{N}{-z}, y \frac{N}{-z} \right) = \left(1 \times \frac{1}{1.5}, 0.5 \times \frac{1}{1.5} \right) = (0.666, 0.333)$$



Pseudodepth

- Classical perspective projection projects (x,y) coordinates to (x^*, y^*) , drops z coordinates



- But we need z to find closest object (depth testing)
- Actual z distance of P from eye could be large and cumbersome

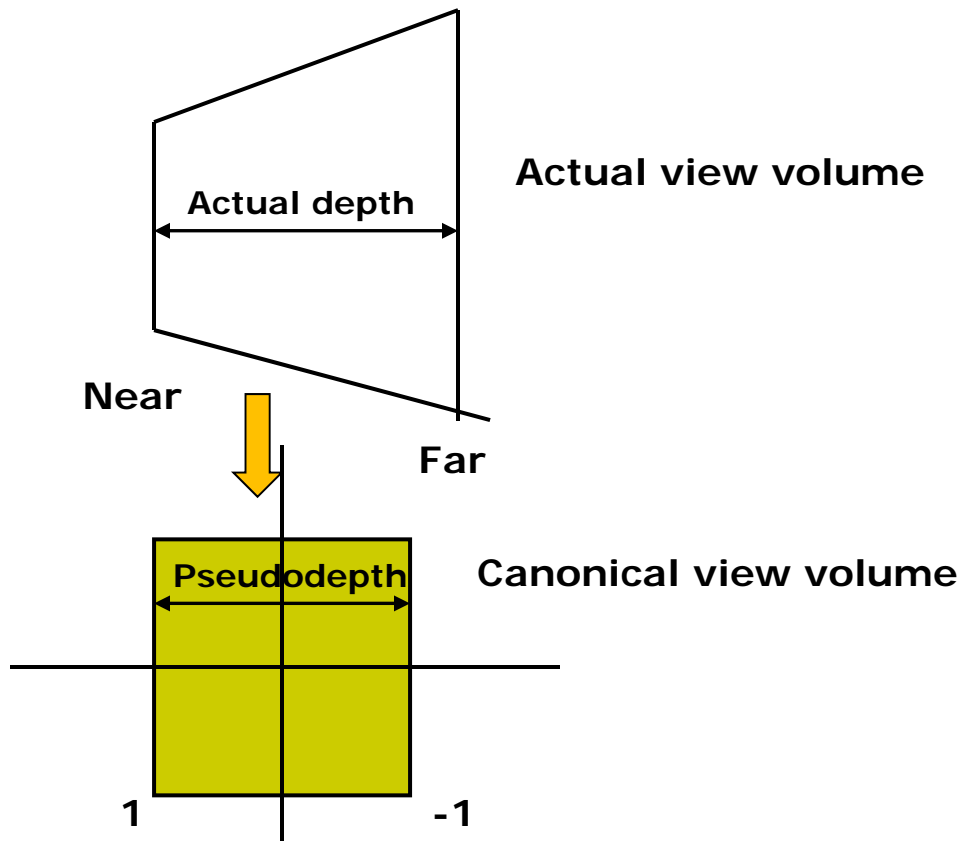
$$distance = \sqrt{(x^2 + y^2 + z^2)}$$

- Introduce **pseudodepth**: Transform actual z distance to a range of $z = [-1,1]$, which can be used for depth testing



Perspective Transformation

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



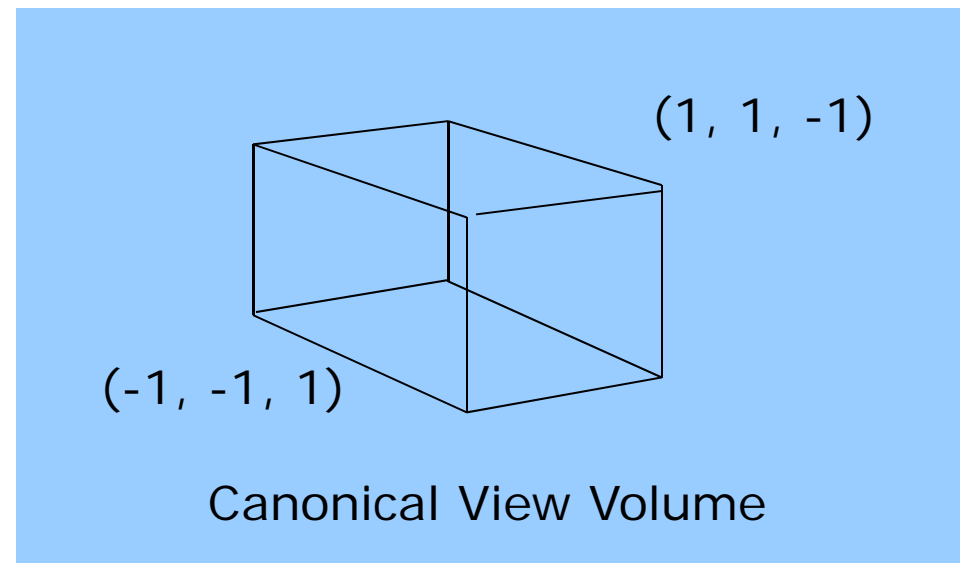
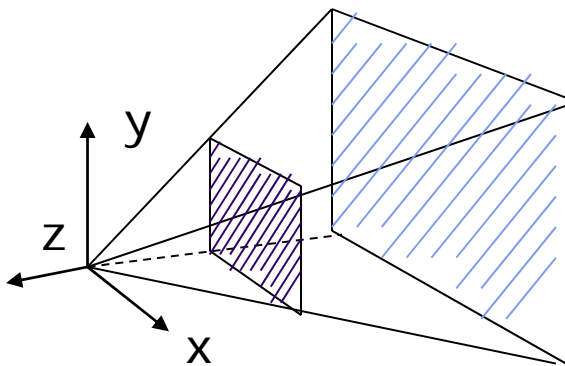
We want perspective Transformation and NOT classical projection!!

Set scaling z
Pseudodepth = $az + b$
Next solve for a and b

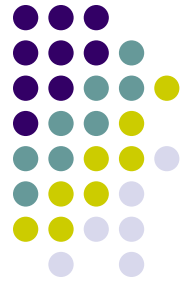


Perspective Transformation

- We want to transform viewing frustum volume into canonical view volume

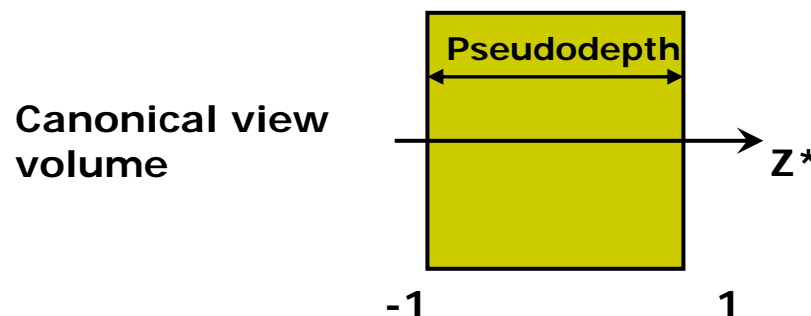
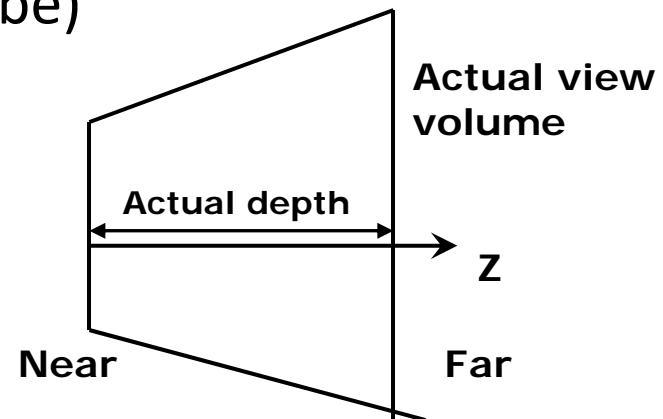


Perspective Transformation using Pseudodepth



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

- 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
 - $z^* = -1$ when $z = -N$(1)
 - $z^* = 1$ when $z = -F$(2)
- Set up simultaneous equations

$$-1 = \frac{-aN + b}{N} \Rightarrow -N = -aN + b \dots \dots (1)$$

$$1 = \frac{-aF + b}{F} \Rightarrow F = -aF + b \dots \dots (2)$$



Transformation of z : Solve for a and b

$$-N = -aN + b \dots \dots (1)$$

$$F = -aF + b \dots \dots (2)$$

- Multiply both sides of (1) by -1

$$N = aN - b \dots \dots (3)$$

- Add eqns (2) and (3)

$$F + N = aN - aF$$

$$\Rightarrow a = \frac{F + N}{N - F} = \frac{-(F + N)}{F - N} \dots \dots (4)$$

- Now put (4) back into (3)



Transformation of z : Solve for a and b

- Put solution for a back into eqn (3)

$$N = aN - b \dots \dots (3)$$

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

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

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

- So

$$a = \frac{-(F + N)}{F - N} \qquad b = \frac{-2FN}{F - N}$$



What does this mean?

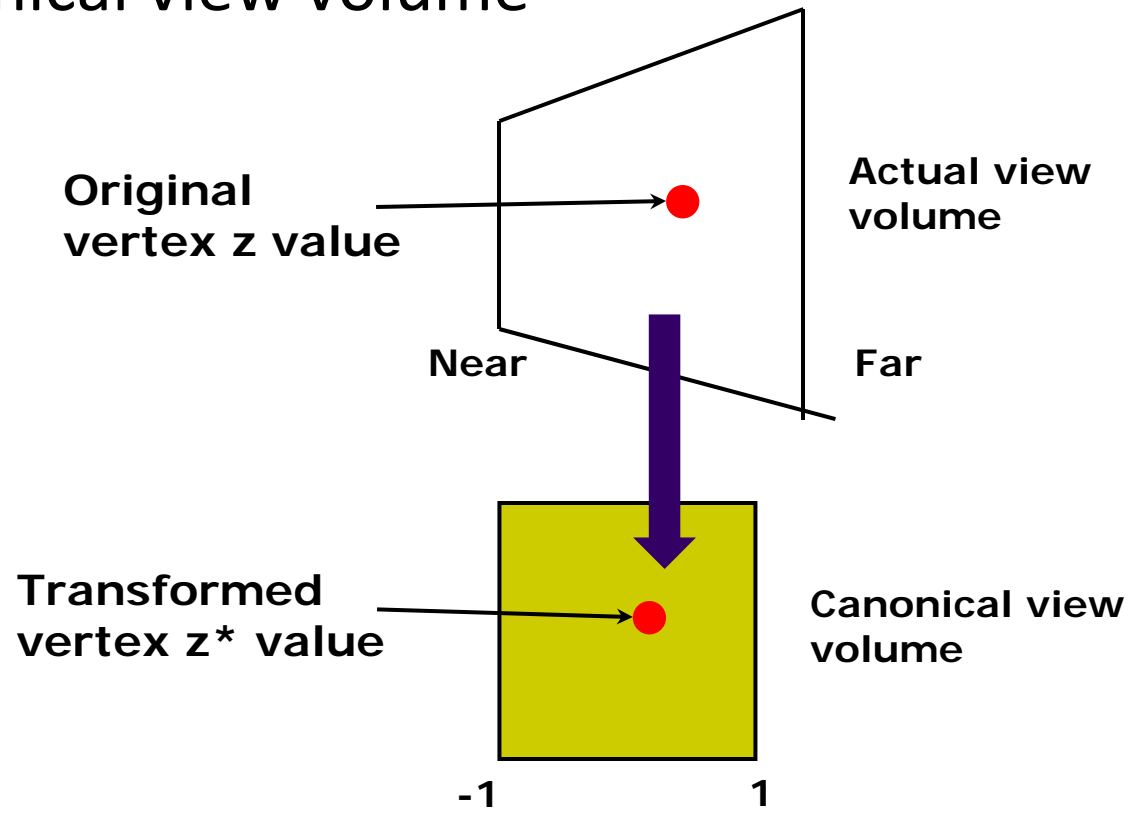
- Original point z in original view volume, transformed into z^* in canonical view volume

$$z^* = \frac{az + b}{-z}$$

- 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
 $P = (wP_x, wP_y, wP_z, w)$ (**Note:** w 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: $x^* = x \frac{N}{-z}$ $y^* = y \frac{N}{-z}$ $z^* = \frac{az + b}{-z}$

- In matrix form:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} = \begin{pmatrix} wNx \\ wNy \\ w(ax + b) \\ -wz \end{pmatrix} \Rightarrow \begin{pmatrix} x \frac{N}{-z} \\ y \frac{N}{-z} \\ \frac{az + b}{-z} \\ 1 \end{pmatrix}$$

Perspective
Transform Matrix

Original
vertex

Transformed
Vertex

Transformed Vertex
after dividing by 4th term



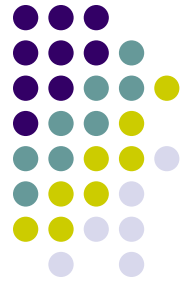
Perspective Projection Matrix

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

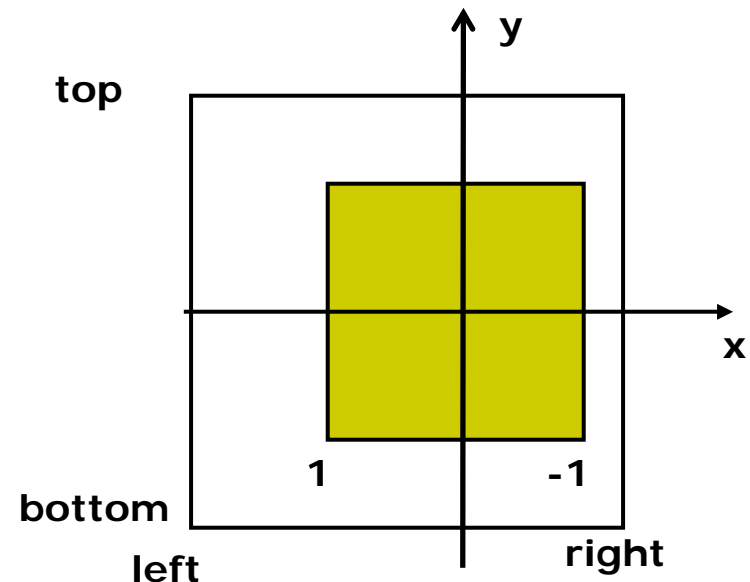
$$a = \frac{-(F+N)}{F-N} \quad b = \frac{-2FN}{F-N}$$

- In perspective transform matrix, already solved for a and b :
- So, we have transform matrix to transform \mathbf{z} values

Perspective Projection



- Not done yet!! Can now transform z!
- Also need to transform the $\mathbf{x} = (x_{\min}, x_{\max})$ and $\mathbf{y} = (y_{\min}, y_{\max})$ 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



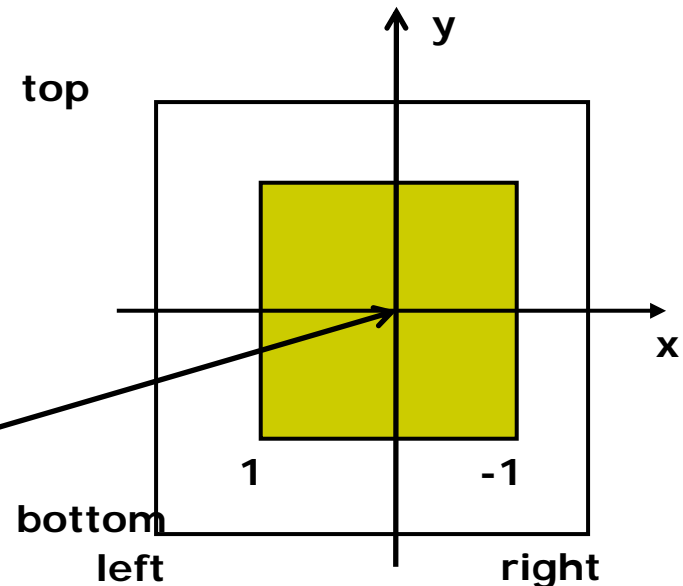
Perspective Projection



- Translate along x and y to line up center with origin of CVV
 - $-(right + left)/2$ in x
 - $-(top + bottom)/2$ in y
- Multiply by translation matrix:

$$\begin{pmatrix} 1 & 0 & 0 & -(right + left) / 2 \\ 0 & 1 & 0 & -(top + bottom) / 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Line up centers
Along x and y



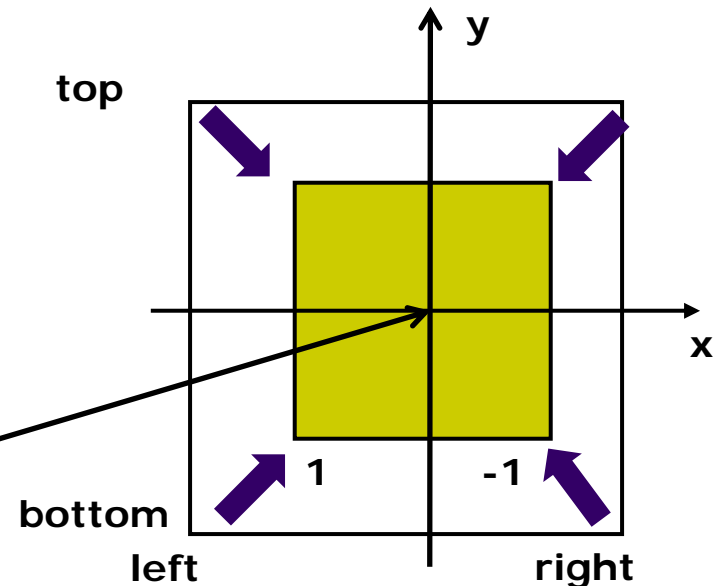
Perspective Projection



- To bring view volume size down to size of of CVV, scale by
 - $2/(\text{right} - \text{left})$ in x
 - $2/(\text{top} - \text{bottom})$ 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



$$\begin{pmatrix} \frac{2}{right - left} & 0 & 0 & 0 \\ 0 & \frac{2}{top - bottom} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -(right + left) / 2 \\ 0 & 1 & 0 & -(top + bottom) / 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

→

$$\begin{pmatrix} \frac{2N}{x_{max} - x_{min}} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2N}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & \frac{-(F + N)}{F - N} & \frac{-2FN}{F - N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

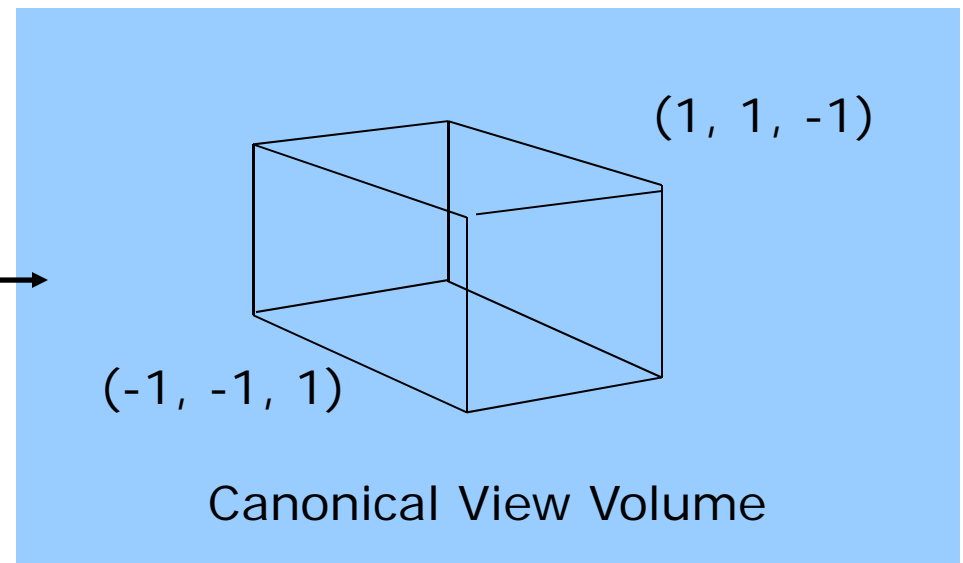
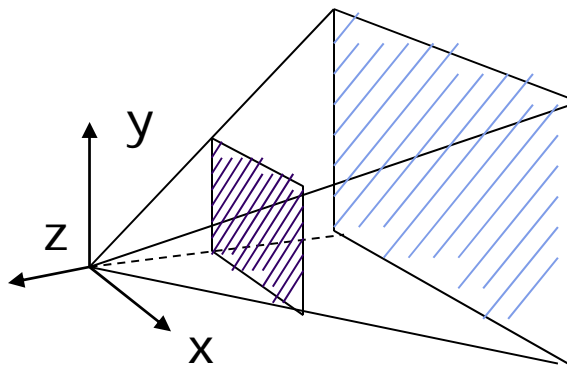
Final Perspective Transform Matrix

glFrustum(left, right, bottom, top, N, F) N = near plane, F = far plane



Perspective Transformation

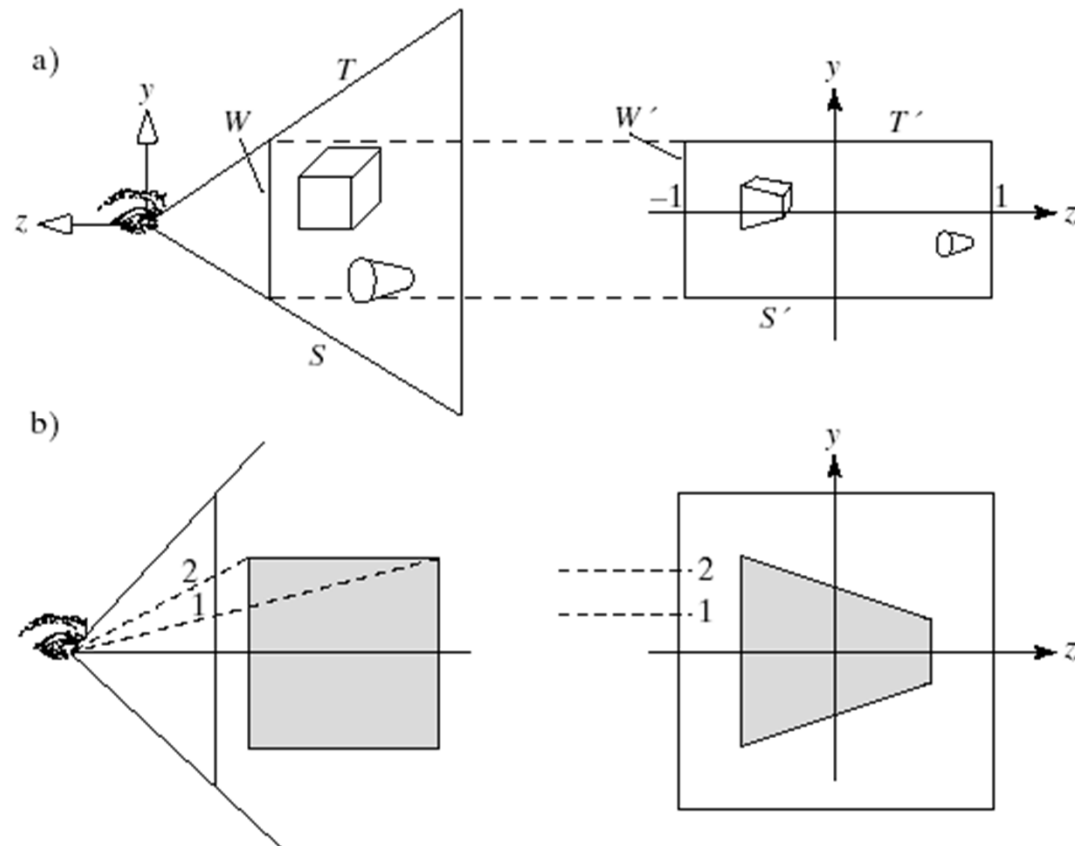
- After perspective transformation, viewing frustum volume is transformed into canonical view volume



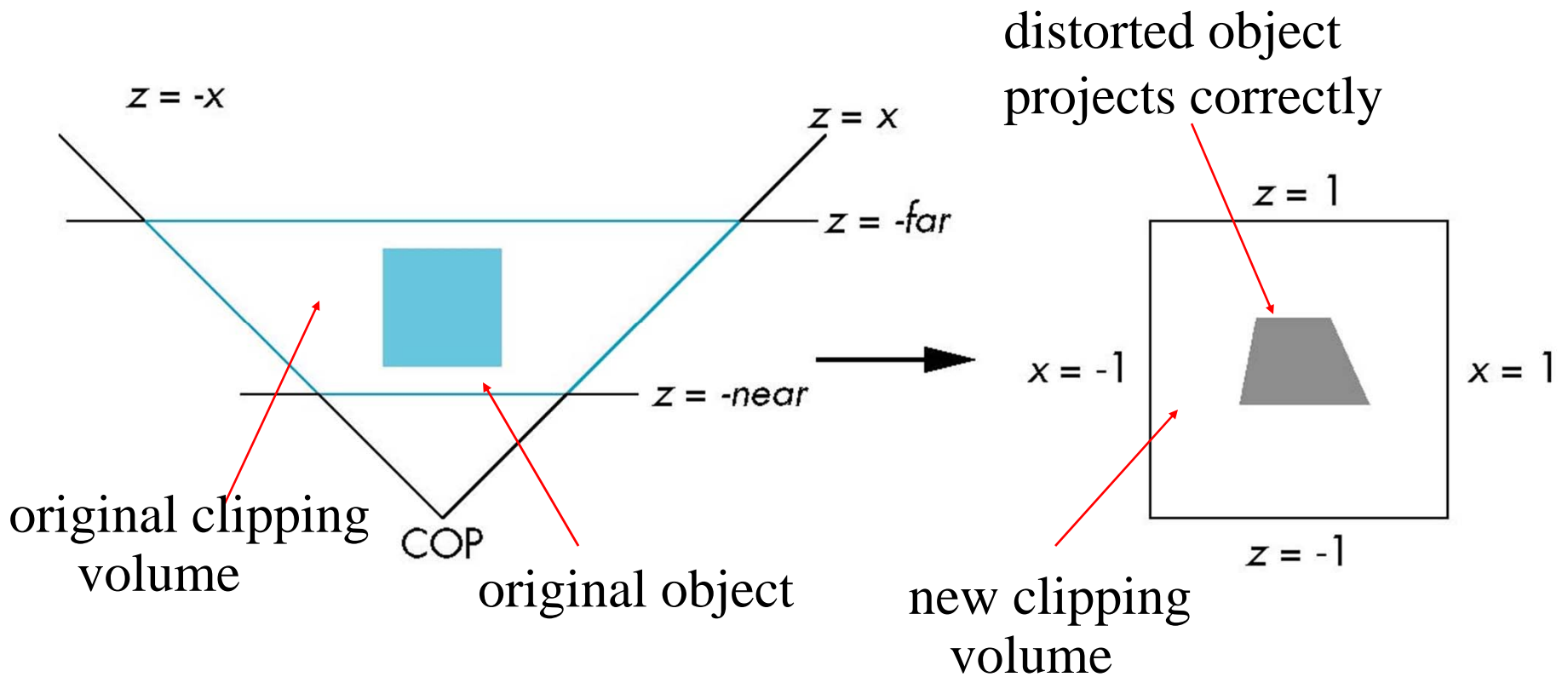
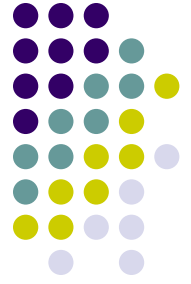
Geometric Nature of Perspective Transform



- 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

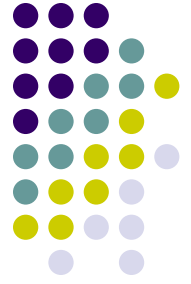




Implementation

- Set modelview and projection matrices in application program
- Pass matrices to shader

```
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

```
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

- Angel and Shreiner, Chapter 4
- Hill and Kelley, Computer Graphics using OpenGL, 3rd edition