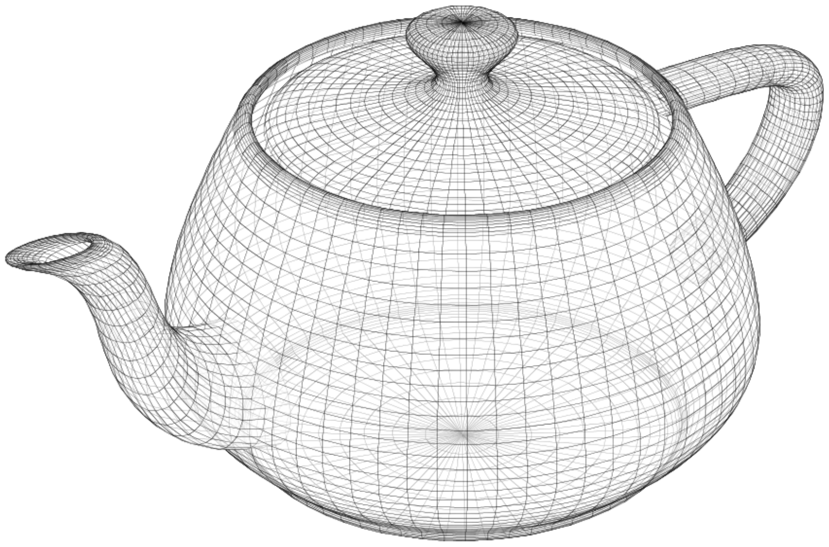


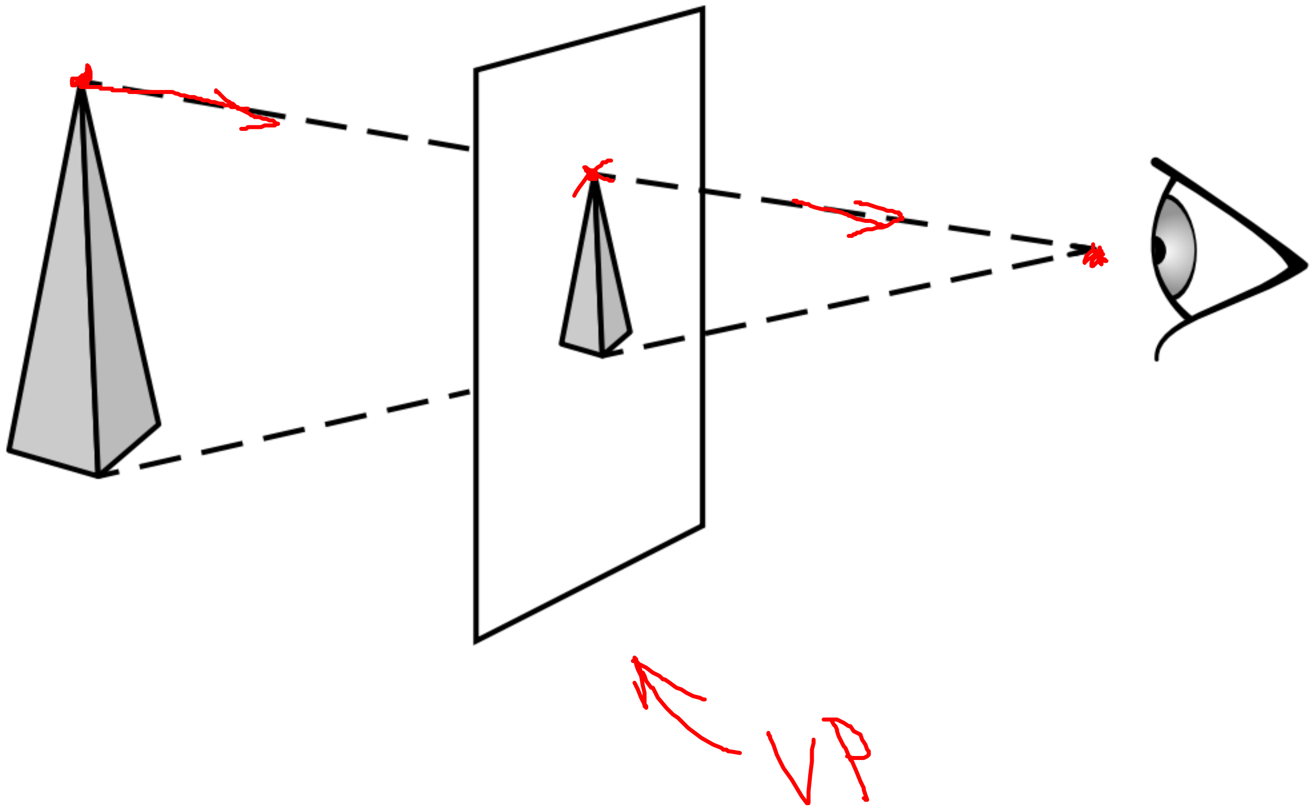
Perspective Projection in WebGL

CS 418: Interactive Computer Graphics

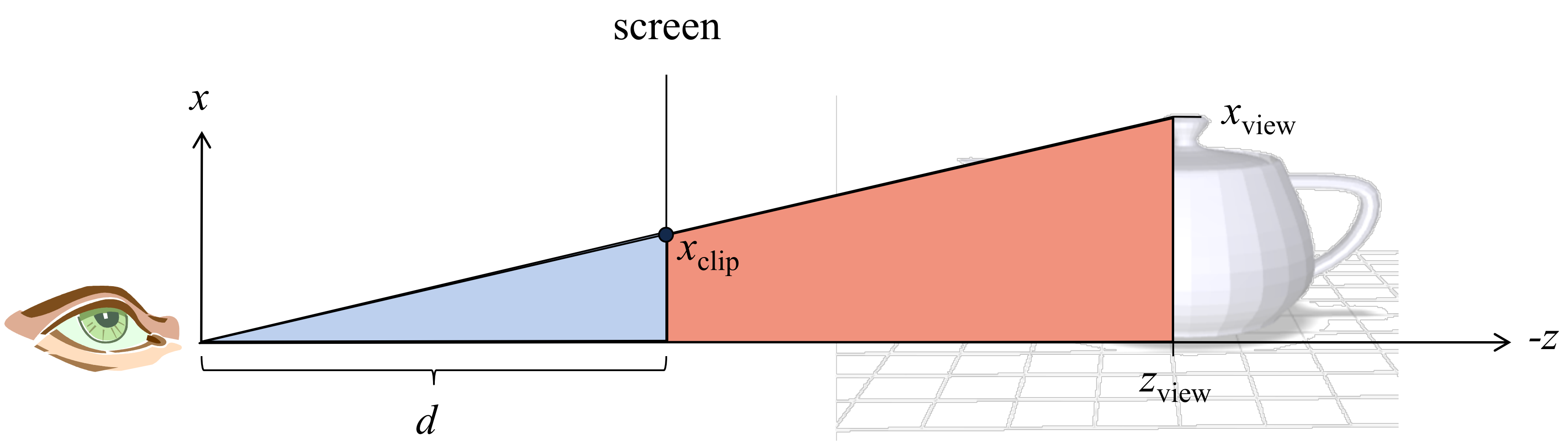
Professor Eric Shaffer

ILLINOIS

Perspective Projection



Perspective



$$x_{clip} = \frac{x_{view}}{-z_{view}/d}$$

$$y_{clip} = \frac{y_{view}}{-z_{view}/d}$$

$$z_{clip} = -d$$

Homogeneous Coordinates

- We can extend our use of homogeneous coordinates to handle projections
- Let the fourth homogeneous coordinate be any non-zero value *w*
- To find the point it corresponds to:
  - multiply all four coordinates by 1/*w*
- When homogeneous coordinate is zero
  - Denotes a “point” at infinity
  - Represents a vector instead of a point
  - Not affected by translation

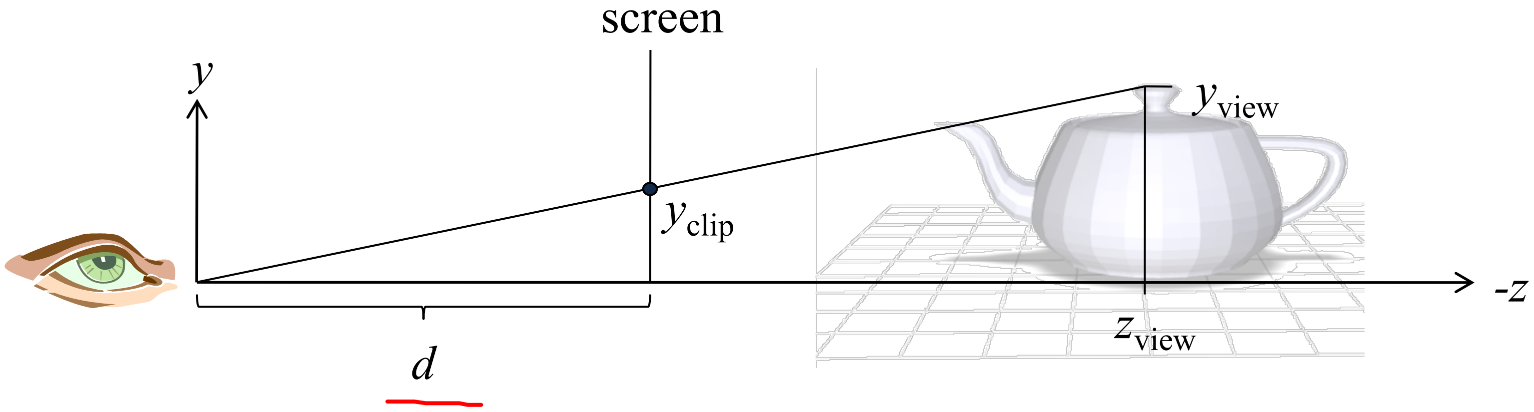
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

ILLINOIS

Perspective



By allowing w to change we represent more kinds of transformations

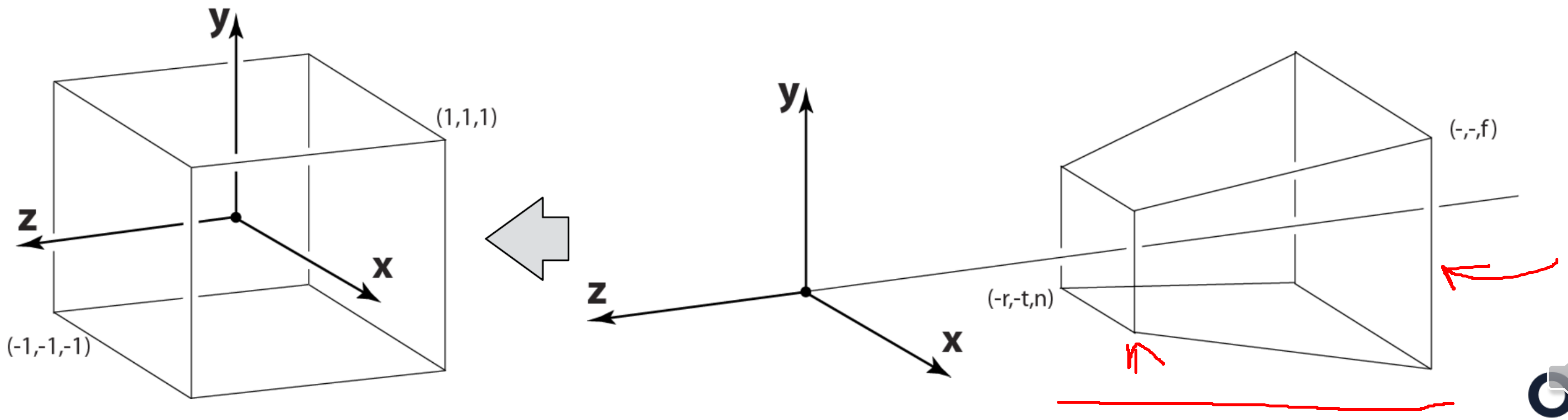
$$\frac{y_{\text{clip}}}{d} = \frac{y_{\text{view}}}{-z_{\text{view}}}$$

$$y_{\text{clip}} = \frac{y_{\text{view}}}{-z_{\text{view}}/d}$$

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1/d & 0 \end{bmatrix} \begin{bmatrix} x_{\text{view}} \\ y_{\text{view}} \\ z_{\text{view}} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{\text{view}} \\ y_{\text{view}} \\ z_{\text{view}} \\ -z_{\text{view}}/d \end{bmatrix} \equiv \begin{bmatrix} \dfrac{x_{\text{view}}}{-z_{\text{view}}/d} \\ \dfrac{y_{\text{view}}}{-z_{\text{view}}/d} \\ -d \\ 1 \end{bmatrix}$$

Perspective Projection in WebGL

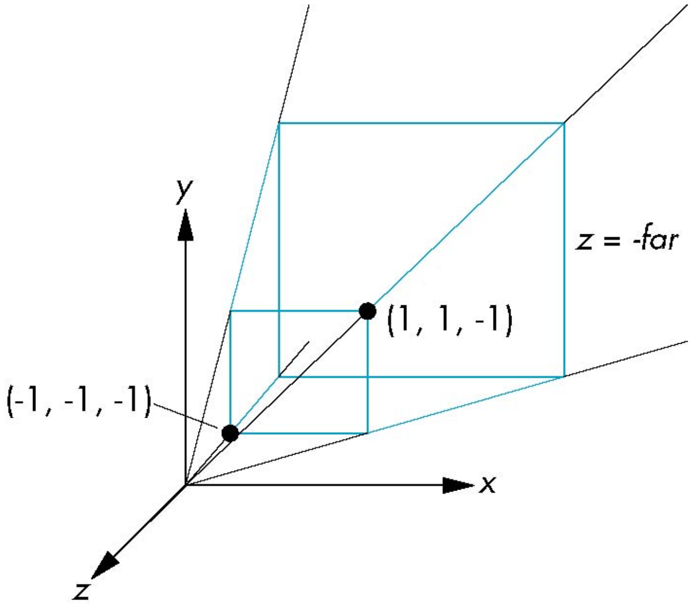
- Just like with the ortho matrix we will
  - Define a viewing volume
  - Map it into the WebGL view volume
- We turn a perspective projection into orthographic projection
- This allows us to use the built-in orthographic projection in WebGL



Perspective Normalization

Consider a simple perspective projection with

- the COP at the origin,
- the near clipping plane at z = -1, and
- a 90 degree field of view determined by the planes $x = \pm z$, $y = \pm z$



Perspective Matrices

Simple perspective projection matrix in homogeneous coordinates

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

This will generate the correct x' and y' and z' for perspective projection onto the z=-1 plane…but we actually want to map into the $[-1,1]^3$ WebGL view volume

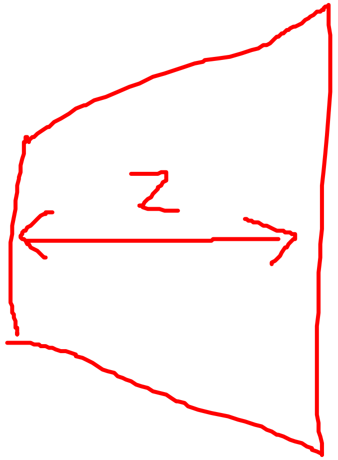
Note that this matrix is independent of the far clipping plane

Generalization

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

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

$$x' = -x/z$$
$$y' = -y/z$$
$$z' = -(\alpha + \beta/z)$$



Picking α and β

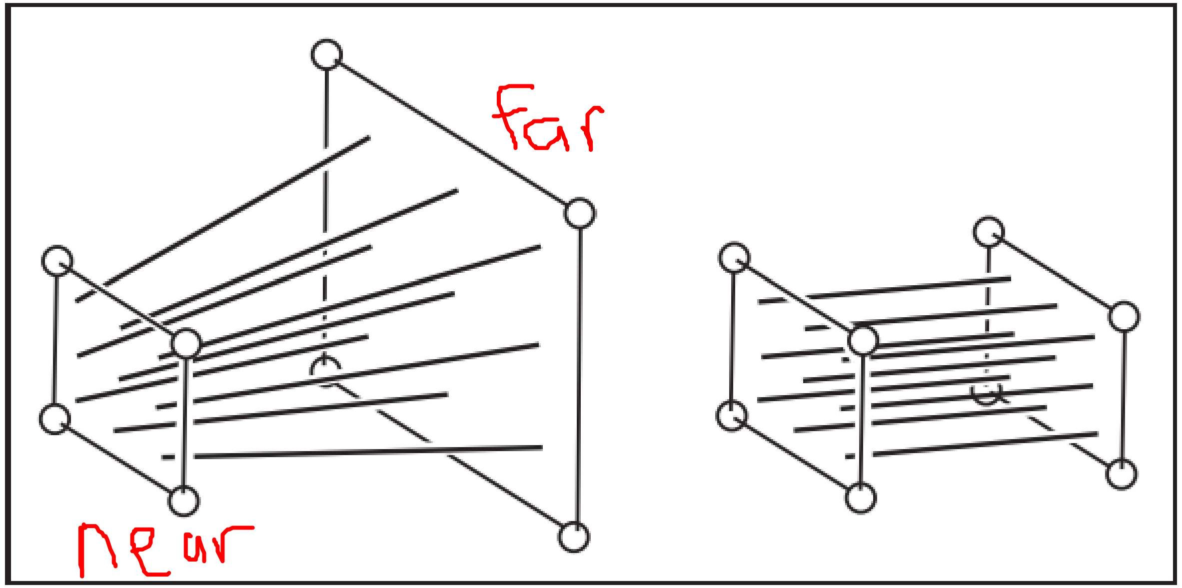
If we pick

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

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

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

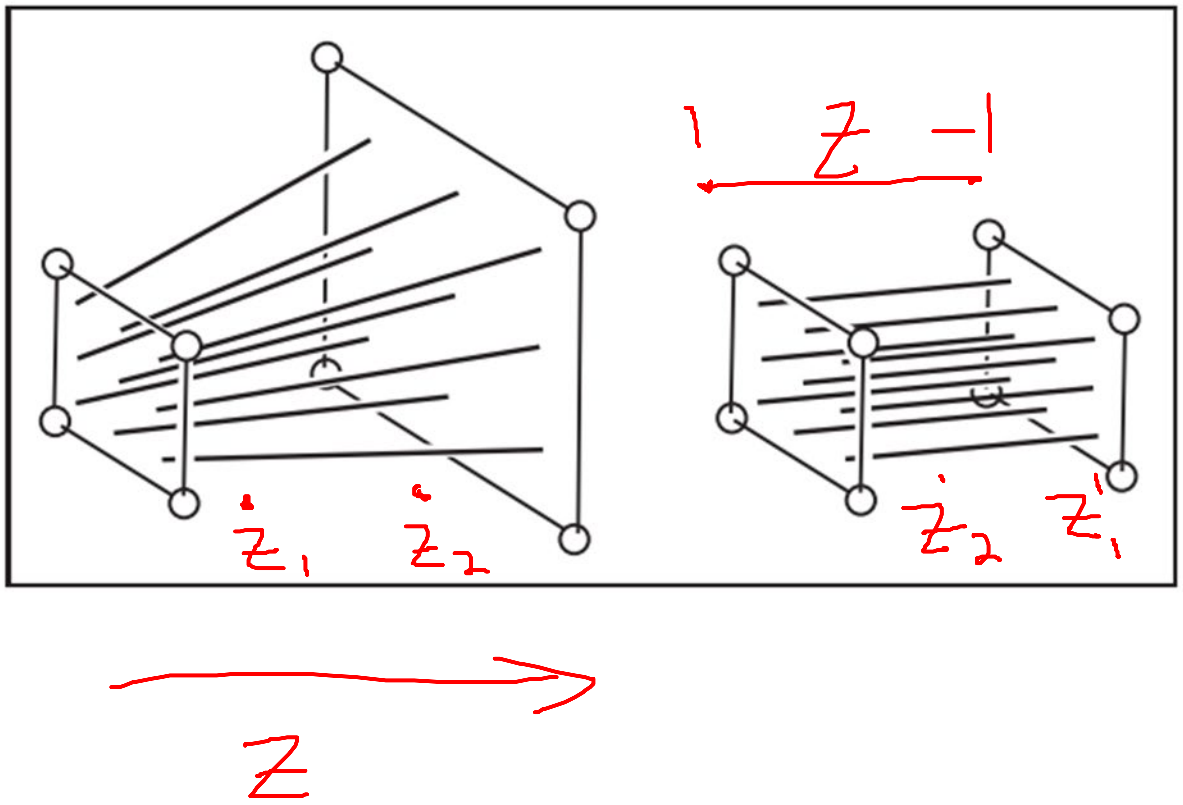
The new view volume is the default clipping volume



Projection and Hidden-Surface Removal

For points p_1 and p_2, if $z_1 > z_2$ in original clipping volume then for the transformed points $z_1' < z_2'$

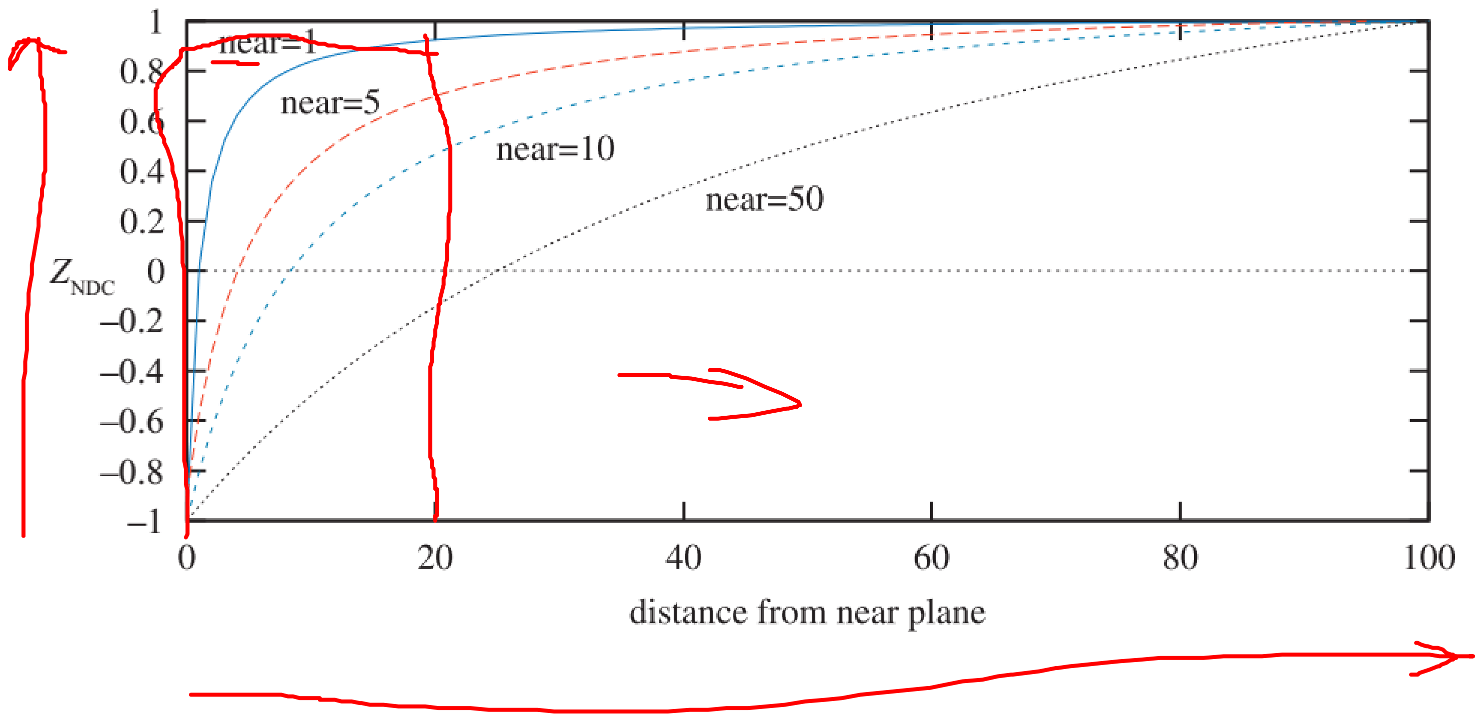
- We can use z' for hidden surface removal using depth comparison



Projection and Hidden-Surface Removal

The formula $z' = -(\alpha+\beta/z)$ → depths are distorted by the transformation

- This can cause numerical problems especially if the near distance is small



The effect of varying the distance of the near plane from the origin. The distance f −n is kept constant at 100. As the near plane becomes closer to the origin, points nearer the far plane use a smaller range of the normalized device coordinate (NDC) depth space. This has the effect of making the z-buffer less accurate at greater distances.

Real-Time Rendering, Fourth Edition (Page 100).

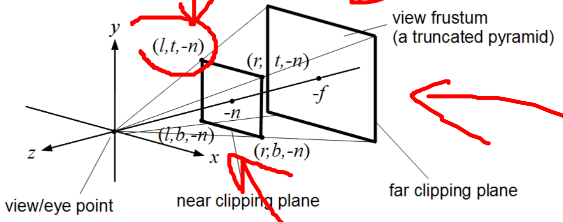
ILLINOIS

WebGL Perspective

- **`mat4.frustum`** allows for an asymmetric viewing frustum using left, right, bottom, top, near, far

z -1

near = 1



- **`mat4.perspective`** generates a symmetric frustum using fovy, aspect, near, far

  fovy → vertical viewing angle in radians

  aspect → aspect ratio of the viewport

  near → ***distance*** from center of projection to near clip plane

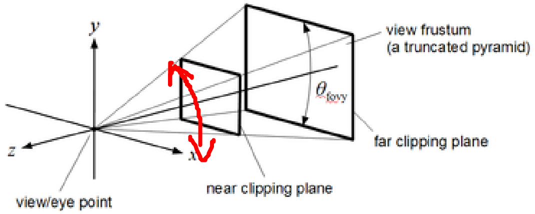
  far → ***distance*** from center of projection to far clip plane

  it computes a frustum with

  right=top x aspect

  top = near x tan(fovy/2)

  left = -right and bottom = -top



Perspective Matrices from glMatrix

frustum

$$P = \begin{bmatrix} \frac{2*near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2*near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2*far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

perspective

$$P = \begin{bmatrix} \frac{near}{right} & 0 & 0 & 0 \\ 0 & \frac{near}{top} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2*far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$