CS 418: Interactive Computer Graphics

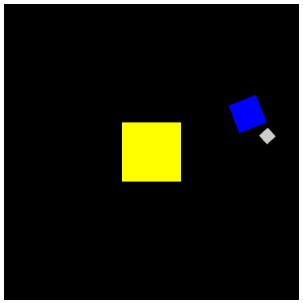
Projection

Eric Shaffer

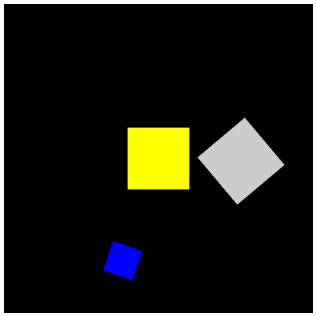
Based on John Hart's CS 418 Slides

Hierarchical Solar System Model

Without push/pop,
You can't move the Earth scale to before you draw the moon:
R_eT_e $R_mT_mS_m$ drawMoon



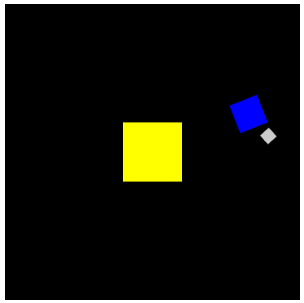
R_eT_e S_e $R_mT_mS_m$ drawMoon



The S_e will change both the T_m and S_m

Hiearchical Solar System Model

You can move the Earth scale to before you draw the moon:
R_eT_e Push S_e drawEarth Pop $R_mT_mS_m$ drawMoon

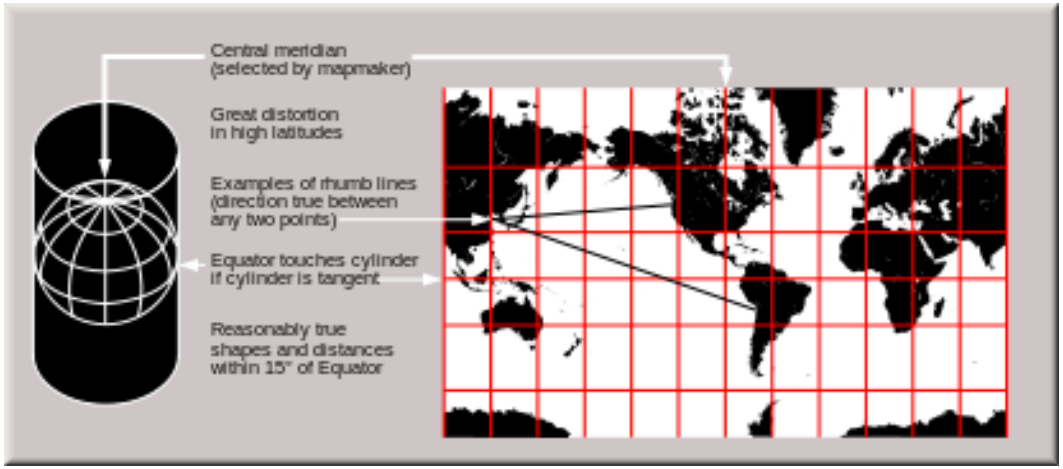


Projections

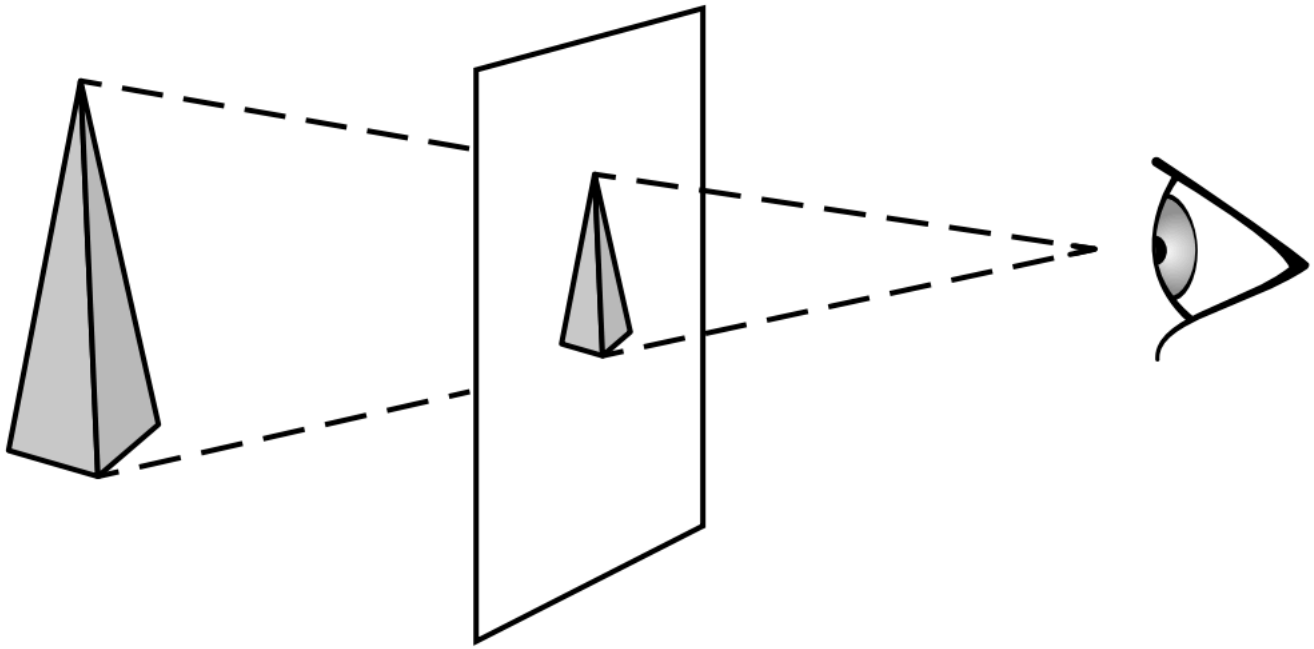
- In computer graphics,
  eventually we need to move from 3D space to 2D space
  - More accurately, from four-dimensional homogenous coordinates to three dimensional homogenous coordinates
- A ***projection*** is a transformation that maps from a high-dimensional space into a lower-dimensional space.
- We will look at some common projections
- …and then we will discuss projection within WebGL

Planar Geometric Projections

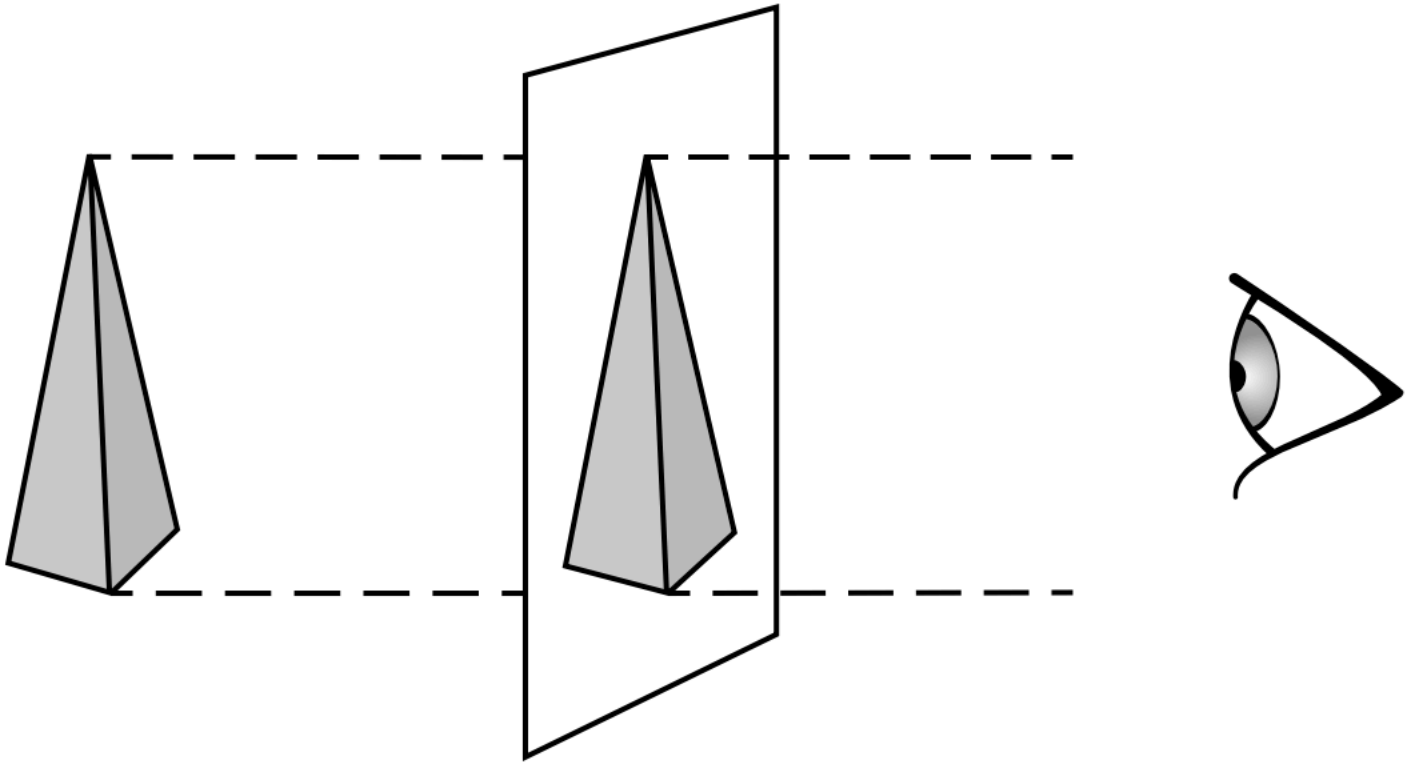
- Standard projections project onto a plane
- Projectors are lines that either
  - converge at a center of projection
  - are parallel
- Such projections preserve lines
  - but not necessarily angles
- Non-planar projections are used in map construction



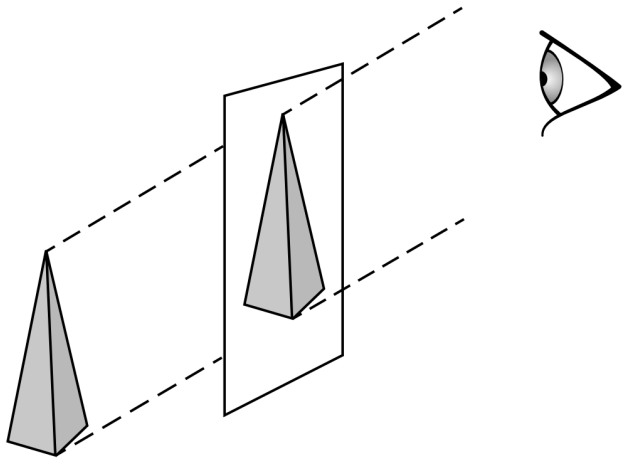
Perspective Projection



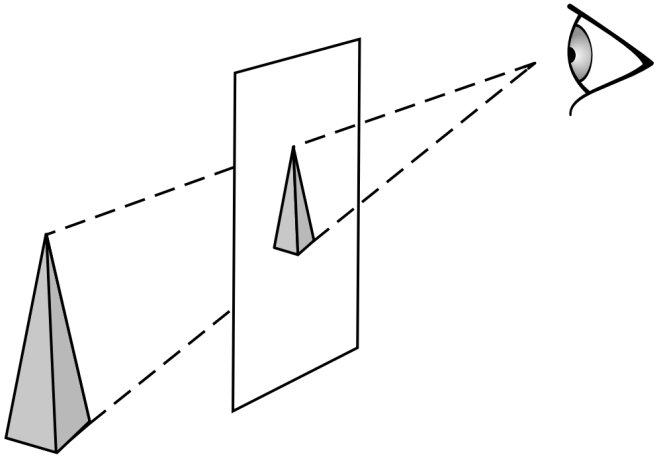
Orthographic Projection



Oblique Projections



Oblique parallel projection



Oblique perspective projection

Linear projections can be categorized

- By whether the projectors are parallel
- By whether the projectors are orthogonal to the view plane

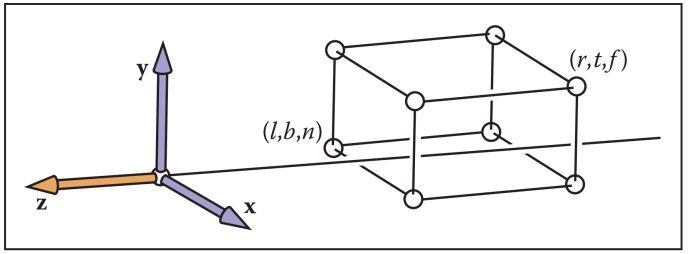
Definition to Know: Foreshortening

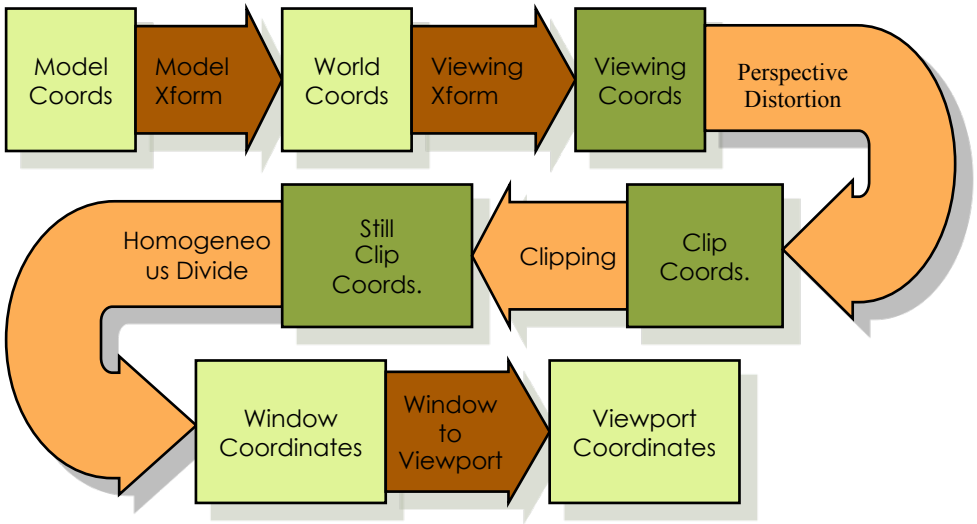
Foreshortening is the visual effect or optical illusion that causes an object or distance to appear shorter than it actually is because it is angled toward the viewer.

i.e. projections
squash
receding
surfaces

Andrea Mantegna
The Lamentation over the Dead Christ

WebGL Secrets





- WebGL only performs an orthogonal projection
  - Everything is projected to the z=0 plane in the normalized view volume
  - But you can distort your geometry to achieve a perspective projection
- The projection occurs when the geometry is in clip space (NDC)
  - and after homogeneous divide
- Even then, depth information is kept around to do hidden surface removal
  - What form is this "depth information"?

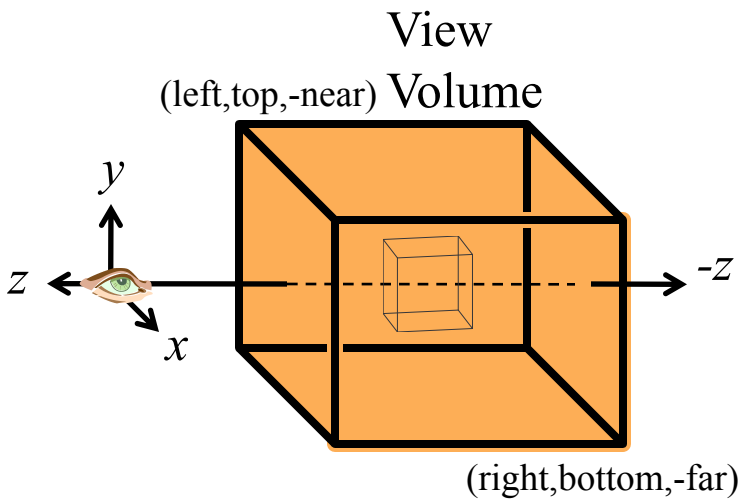
Orthographic Projection

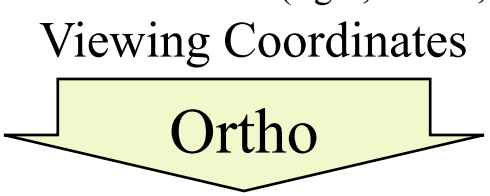
- Foreshortens
- No change in size by depth
- Classic Orthographic Projection matrix simply zeros the z- coordinate

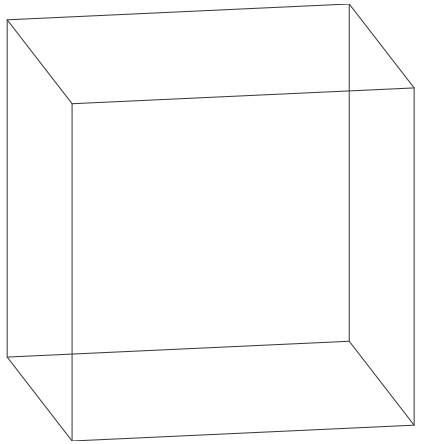
$$\begin{bmatrix} \text{W2V} \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \text{View} \end{bmatrix} \begin{bmatrix} \text{Model} \end{bmatrix}$$

- mat4.ortho(out,left,right,bottom,top,near,far)

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



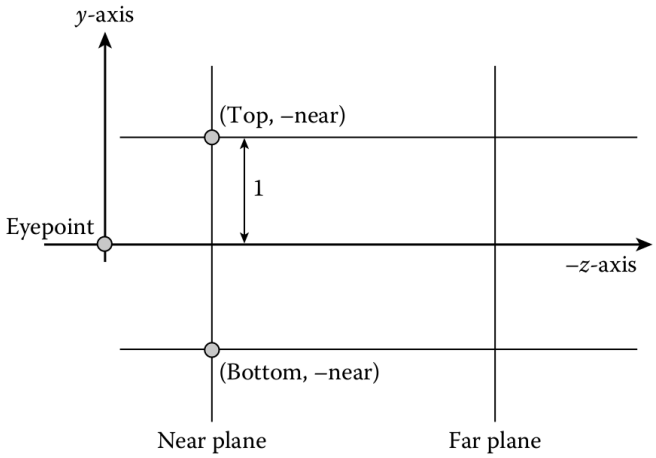




GLMatrix ortho matrix

`ortho(left,right,bottom,top,near,far)`

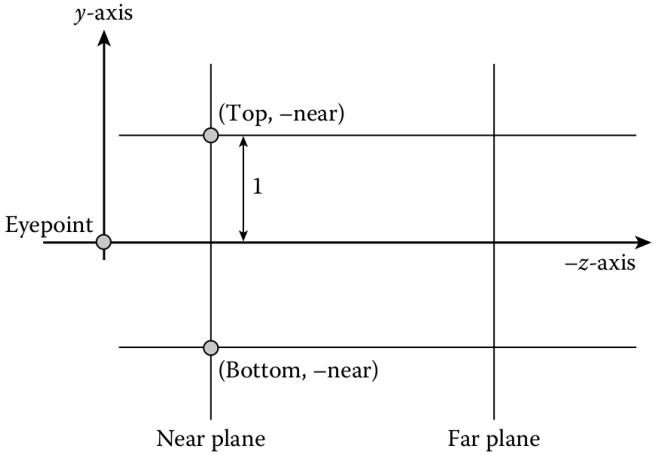
- **near** and **far** are distances measured from camera
  - where is the camera?
- l,r,b,t are coordinates of the bounding planes
- what does the matrix do?



GLMatrix ortho matrix

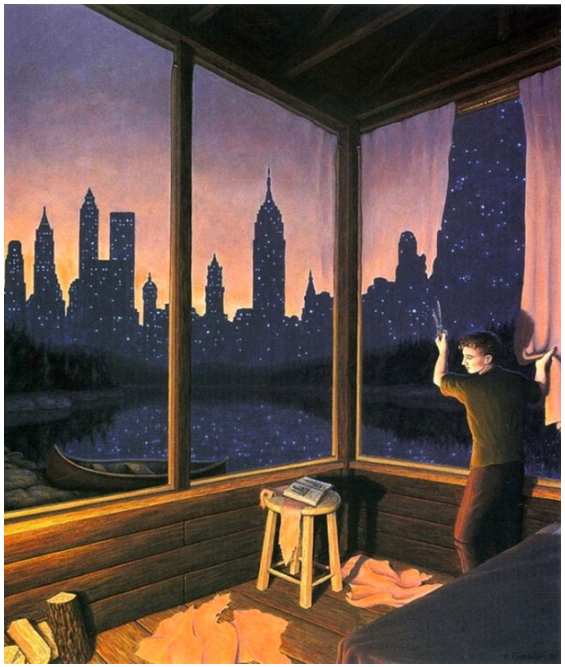
`ortho(left,right,bottom,top,near,far)`

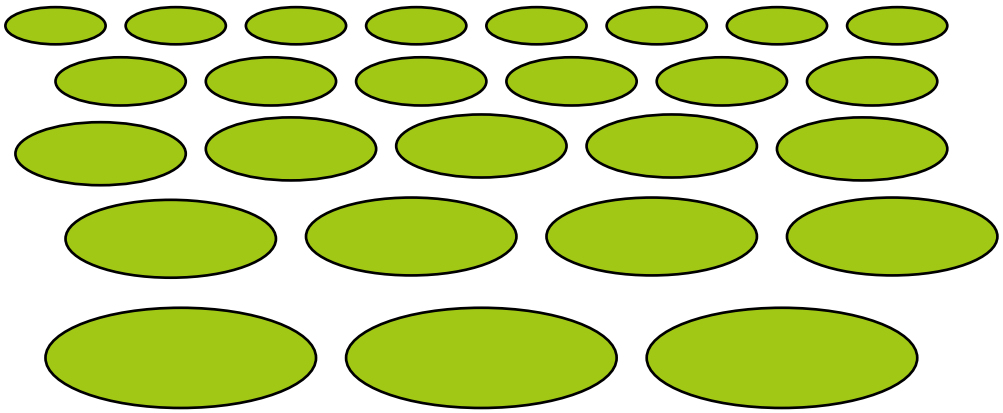
- **near** and **far** are distances measured from camera
  - where is the camera?
    At the origin due to the view transformation
- l,r,b,t are coordinates of the bounding planes
- what does the matrix do?
  It scales and translates the specified box to fit into the NDC view volume….it decides what you see and what gets clipped



Perspective

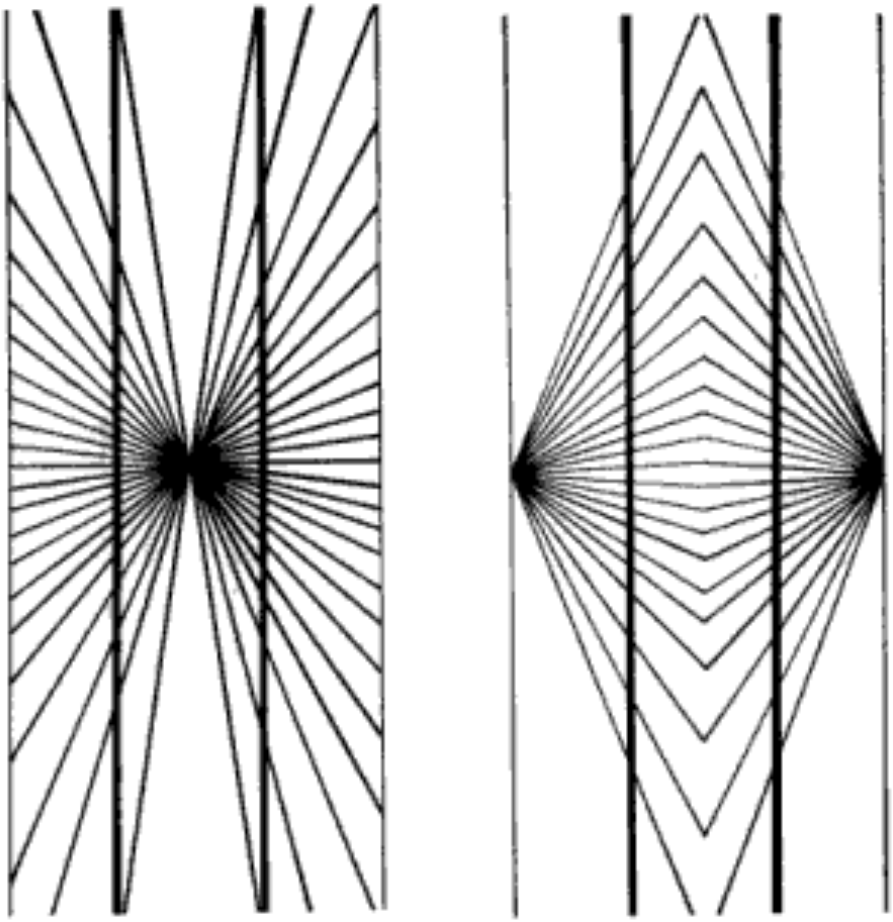
- Brain depends on shape constancy
  - Real world objects do not resize
  - Change in size due to depth
- Closer objects larger
- Farther objects smaller

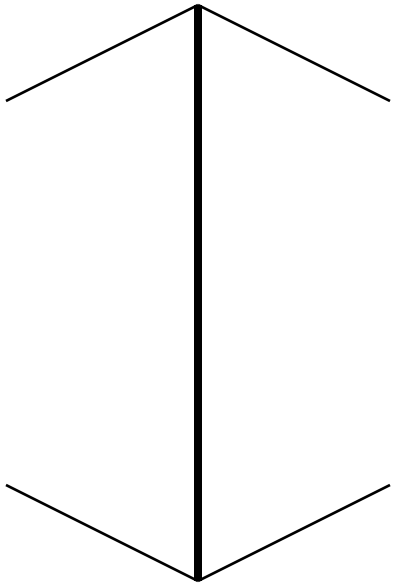


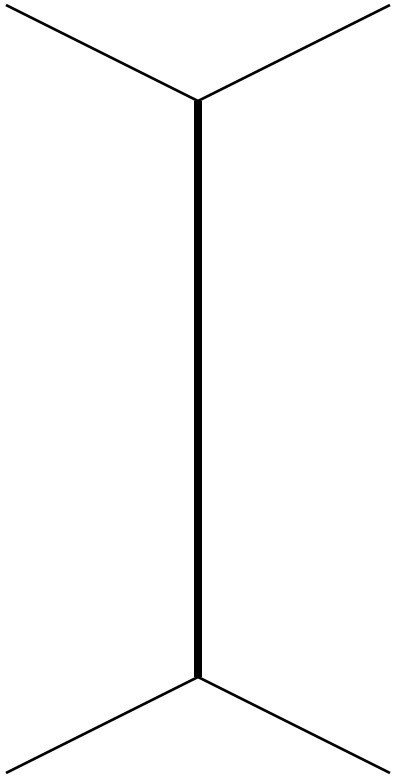


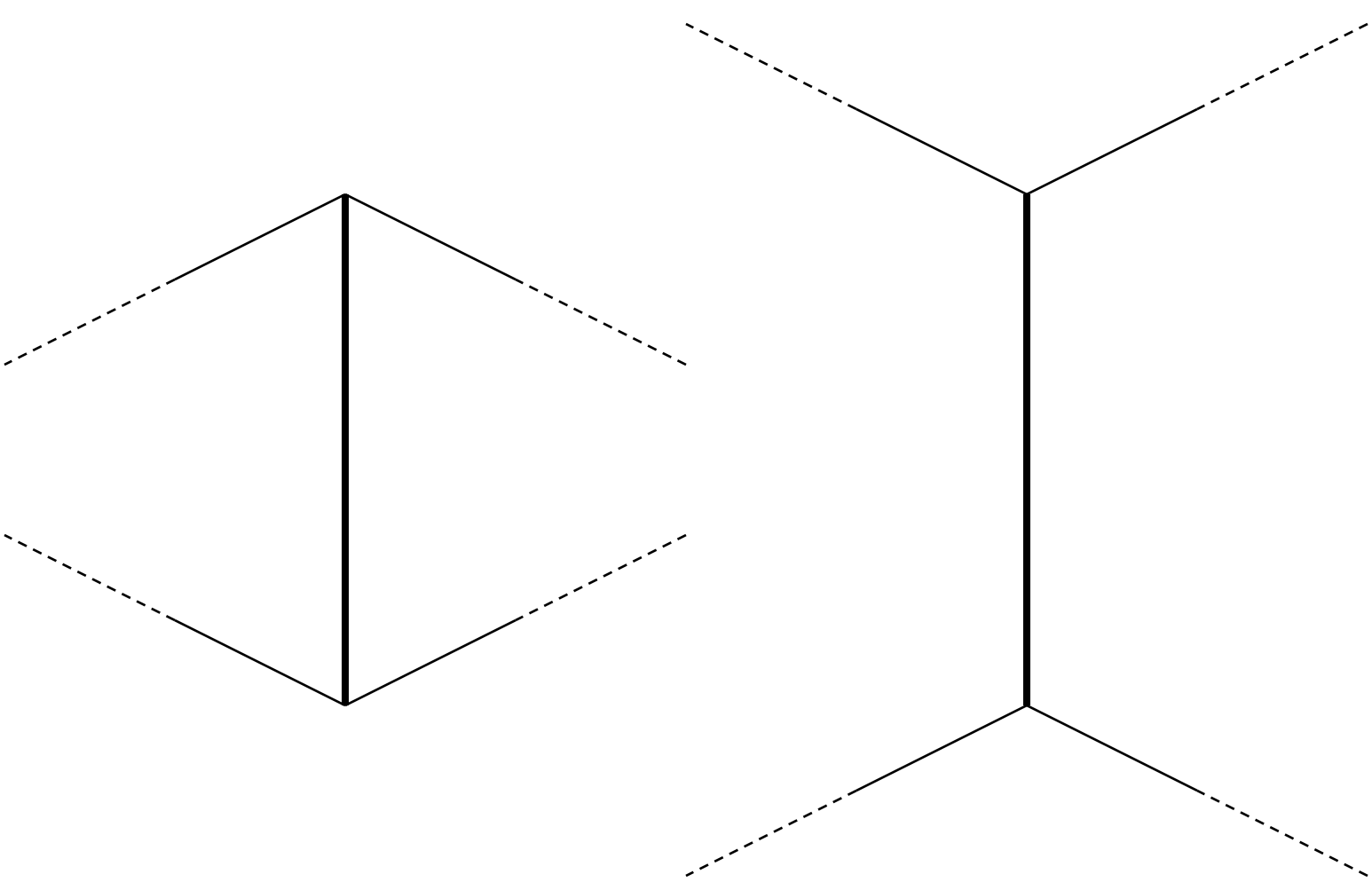
Ames Distorting Room

Hering Illusion





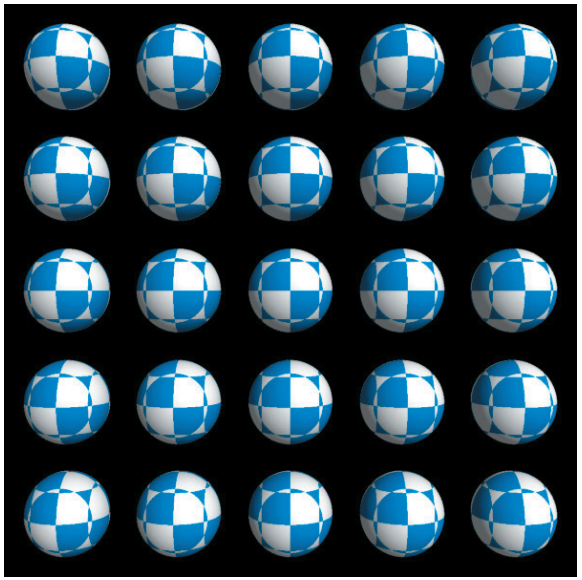


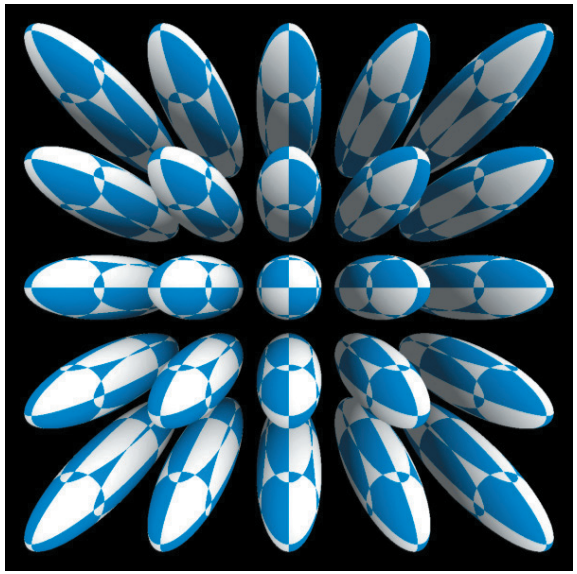


Perspective Projection

- Objects further from viewer are projected smaller than the same sized objects closer to the viewer (*diminution*)
  - Looks realistic
- Equal distances along a line are not projected into equal distances (*nonuniform foreshortening*)
- Angles preserved only in planes parallel to the projection plane
- More difficult to construct by hand than parallel projections

Perspective Distortion

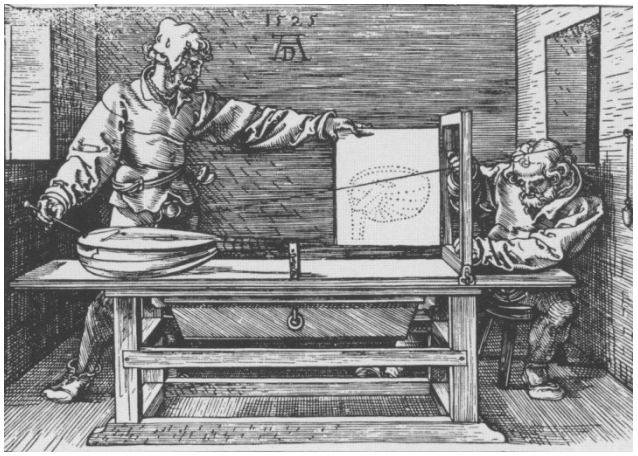




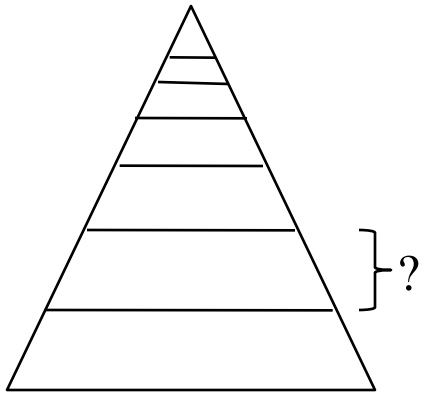
Simple Perspective Projection

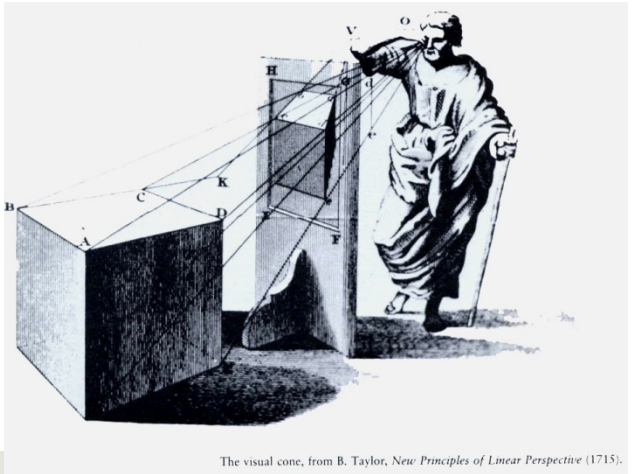
Albrecht Durer woodcut c. 1525,
swiped from Marc Levoy's CS48N notes c. 2007

- Brain depends on shape constancy
  - Real world objects do not resize
  - Change in size due to depth
- Closer objects larger
- Farther objects smaller
- How large, how small?



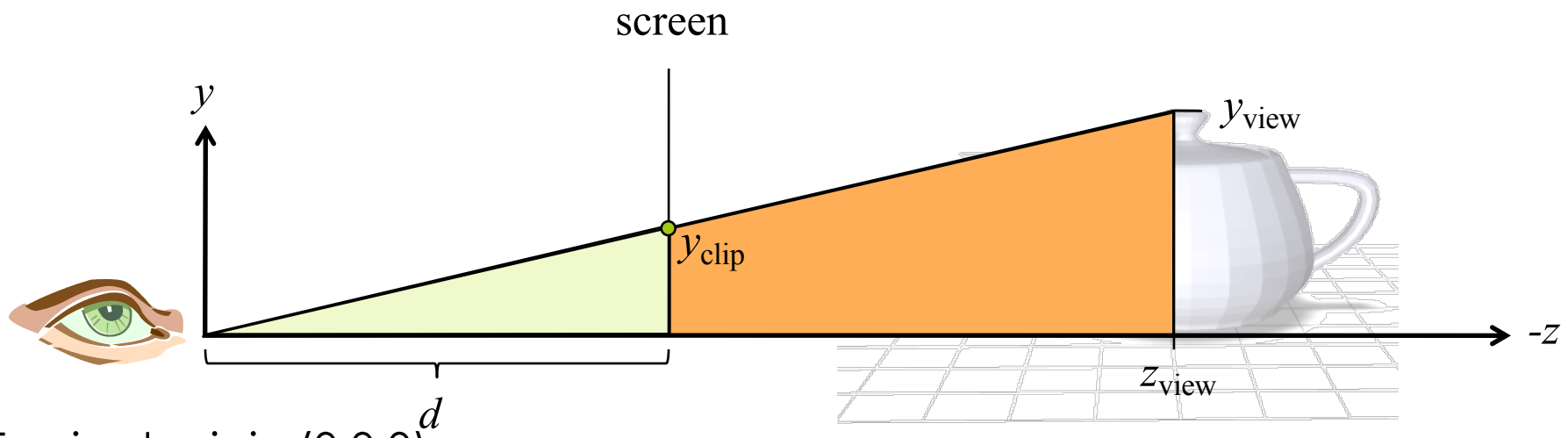
More Durer, swiped from Fredo Durand's Art of Depiction





The visual cone, from B. Taylor, *New Principles of Linear Perspective* (1715).

Simple Perspective



Eye is at origin (0,0,0)
Screen is distance d from the eye.
Looking down negative z-axis.

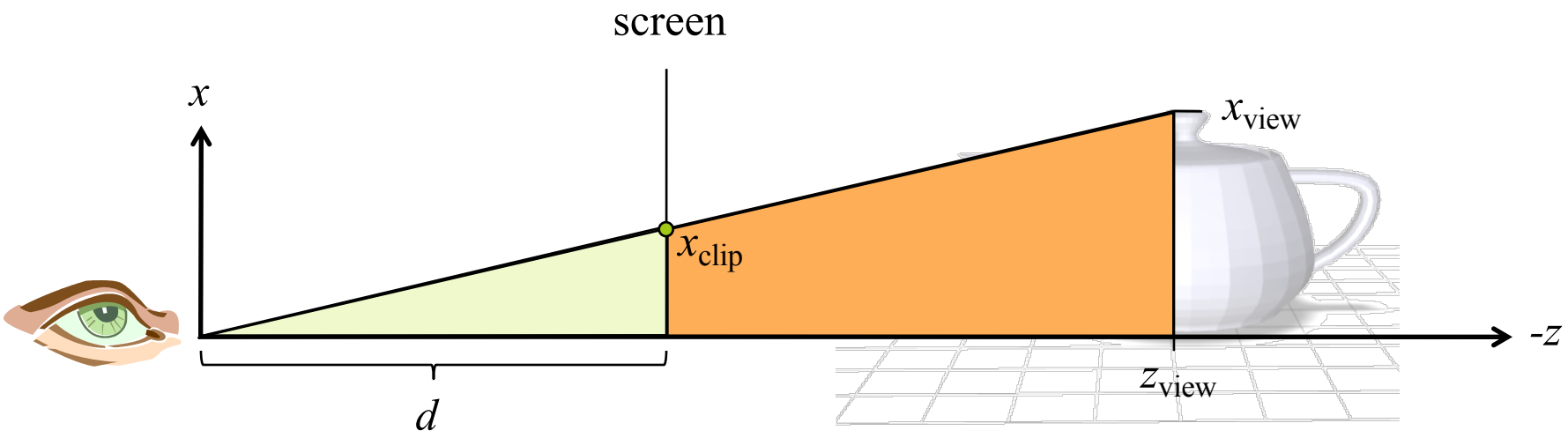
$$\frac{y_{clip}}{d} = \frac{y_{view}}{-z_{view}}$$

The two triangles are *similar*
(two angles are obviously congruent)

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

This means corresponding sides
are in the same proportions

Simple Perspective

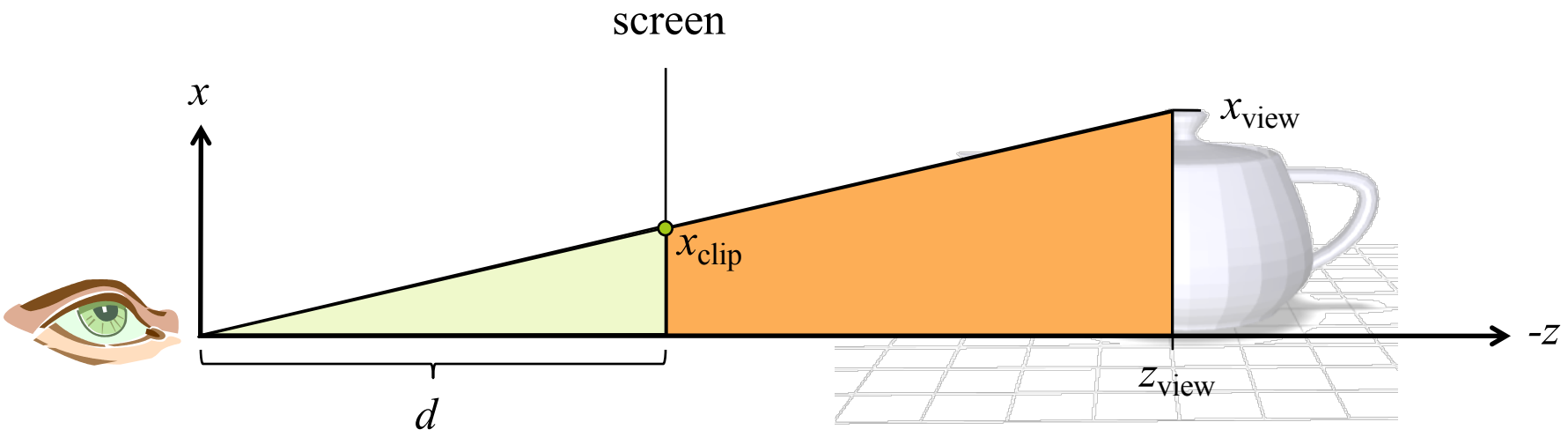


Same process derives the projection for the x coordinate.

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

What is z_{clip}?

Simple Perspective



1. This transformation is not invertible
2. It does preserve lines (except when?)
3. It is not an affine transformation
   (it does not preserve ratios of distances
4. For graphics we will want use a variant that preserves relative distances for hidden surface removal…we'll see that later

Projections Using Homogeneous Coordinates

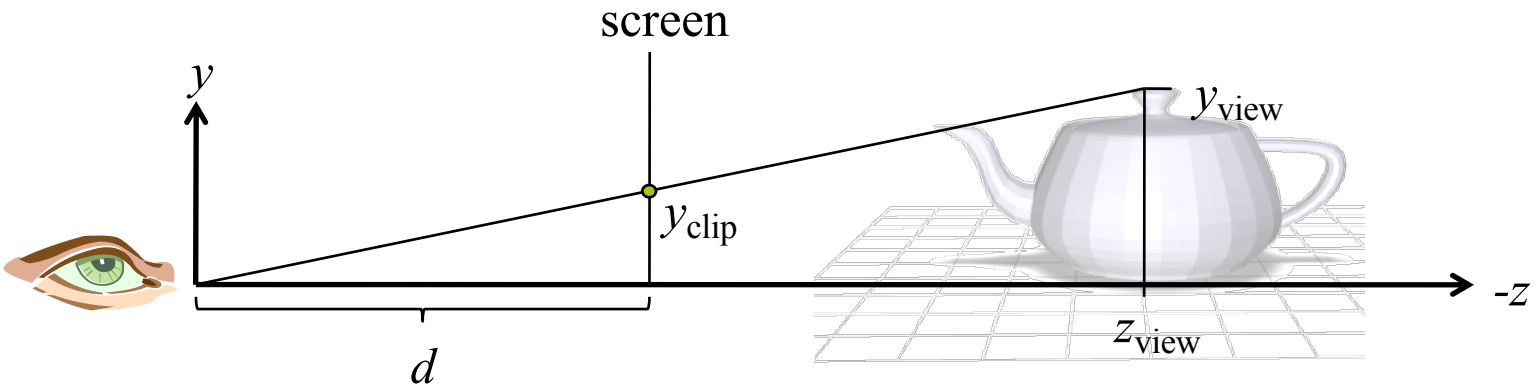
- We can extend our use of homogenous coordinates to handle projections
- Let the fourth homogeneous coordinate be any non-zero value w
- To find the point it corresponds to:
  - multiply all four coordinates by 1/w
- When homogeneous coordinate is zero
  - Denotes a "point" at infinity
  - Represents a vector instead of a point
  - Not affected by translation
- A point in 3D corresponds to what in the 4D homogenous space?

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

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

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

Simple Perspective



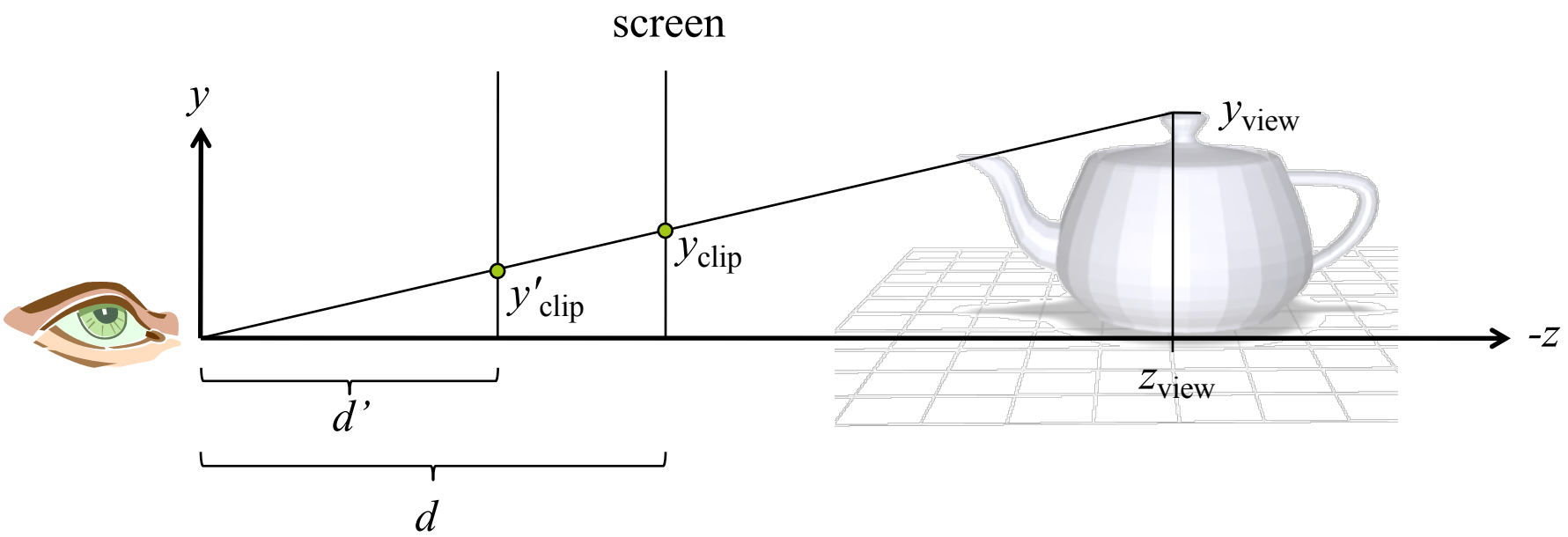
By allowing w to change we represent more kinds of transformations

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

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

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

Parameter d

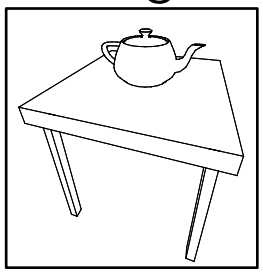


$y_{clip} = d\ y_{view}/(-z_{view})$

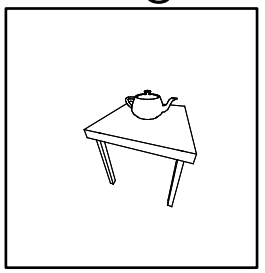
$y'_{clip} = d'\ y_{view}/(-z_{view}) = (d'/d)\ y_{clip}$

Changing parameter d just changes scale of projection

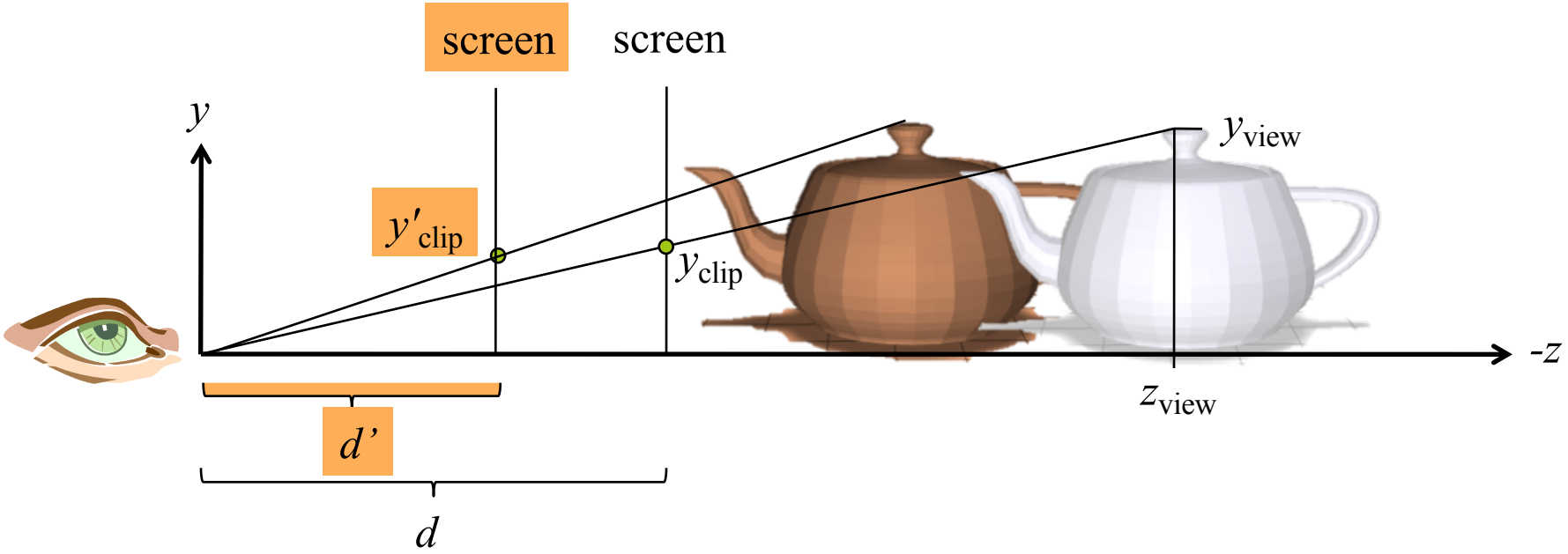
Using d



Using d'



Parameter d



To change degree of perspective distortion, need to change distance from eye to scene,

…by moving scene closer or farther to eye,

… along z axis in viewing coordinates

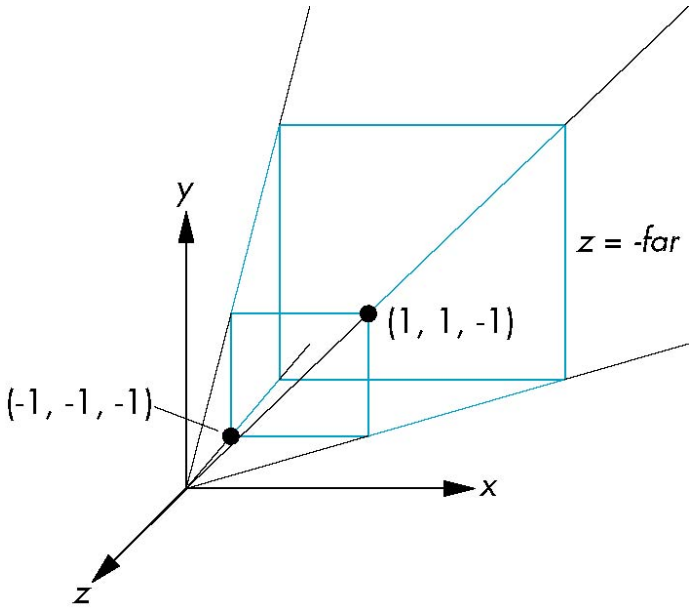
Perspective Projection in WebGL

- Just like with parallel viewing we will
  - Define a viewing volume
  - Construct a ***normalization transformation***
- We turn a perspective projection into orthogonal projection
- This allows us to use the built-in orthogonal projection in WebGL

Perspective Normalization

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

$$x = \pm z,\ y = \pm z$$



Perspective Matrices

Simple projection matrix in homogeneous coordinates

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

Note that this matrix is independent of the far clipping plane

Generalization

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

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

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

Generalization

- If we apply an orthogonal projection after N

- Get the correct x and y coordinates after homogenous divide
  - The same as produced by the simple perspective projection

$$x'' = -x/z$$
$$y'' = -y/z$$
$$z'' = 0$$

Picking α and β

If we pick

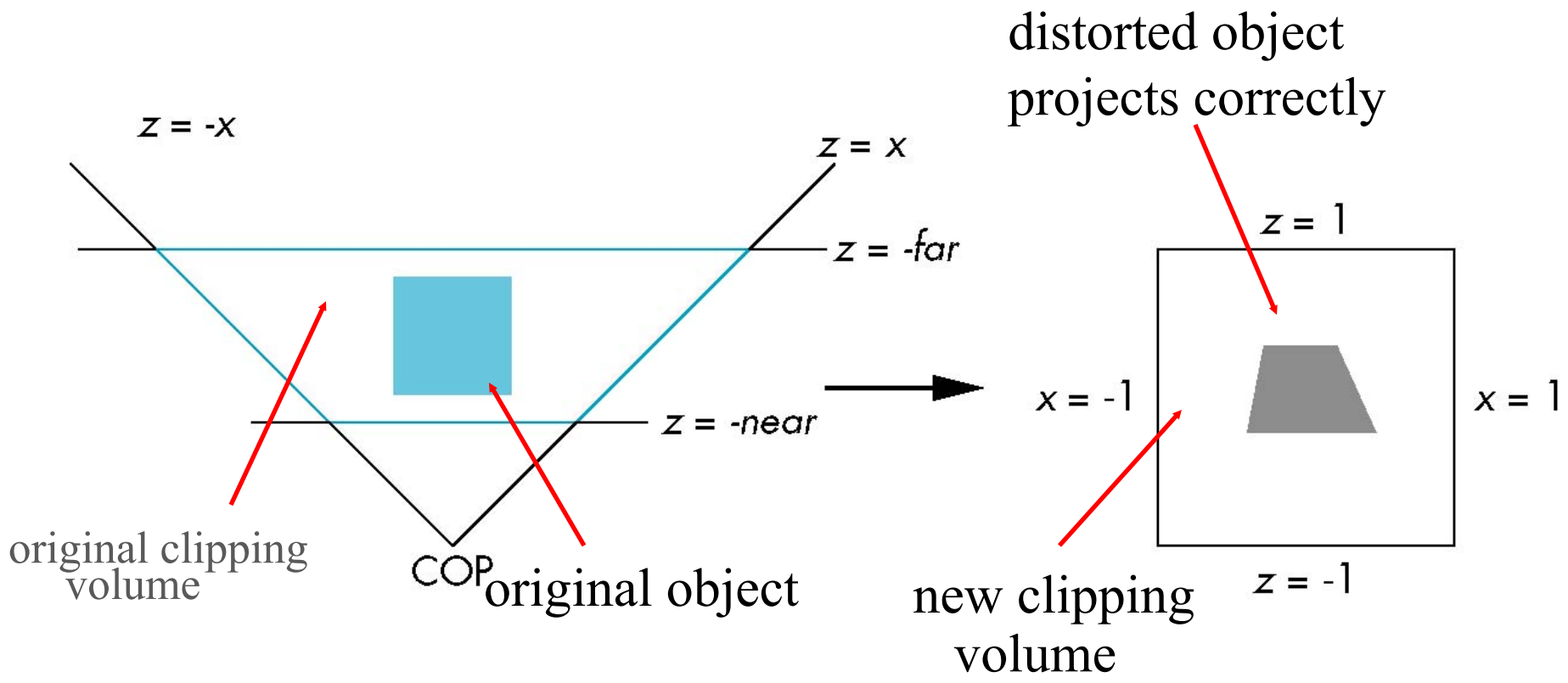
$$\alpha = \frac{\text{near} + \text{far}}{\text{far} - \text{near}}$$

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

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

The new clipping volume is the default clipping volume

Normalization Transformation

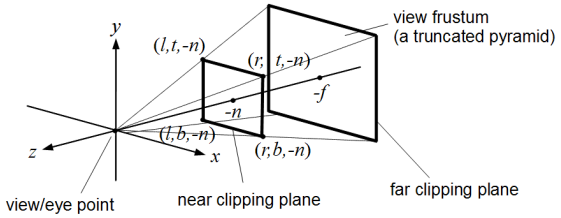


Normalization and Hidden-Surface Removal

- For two points p_1 and p_2, if $z_1 > z_2$ in the original clipping volume then the order for the transformed points $z_1' > z_2'$
- Thus hidden surface removal using depth comparison works if we first apply the normalization transformation
- However, the formula $z'' = -(\alpha+\beta/z)$ implies that the distances are distorted by the normalization which can cause numerical problems especially if the near distance is small

WebGL Perspective

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



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

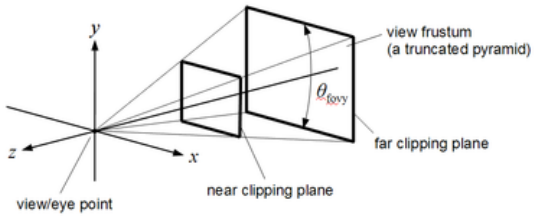
fovy → vertical viewing angle in radians

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)
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

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