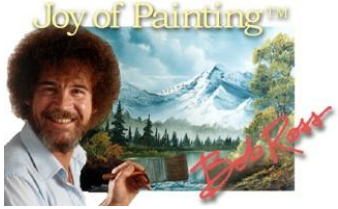
Visible Surface Determination

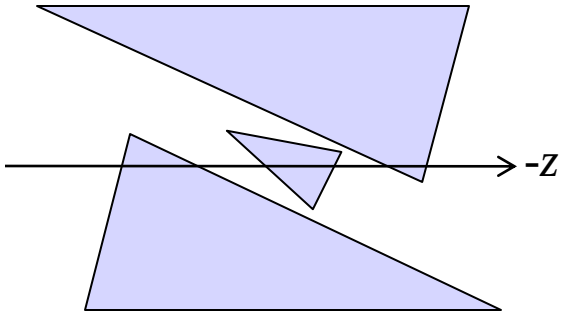
CS418 Computer Graphics

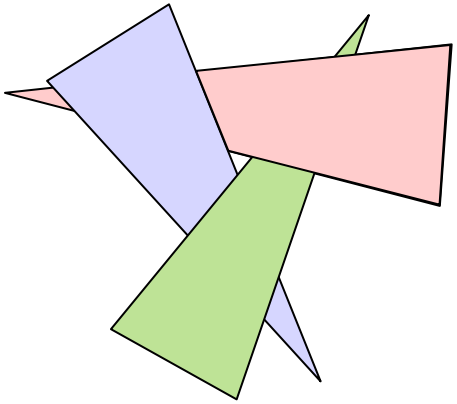
John C. Hart

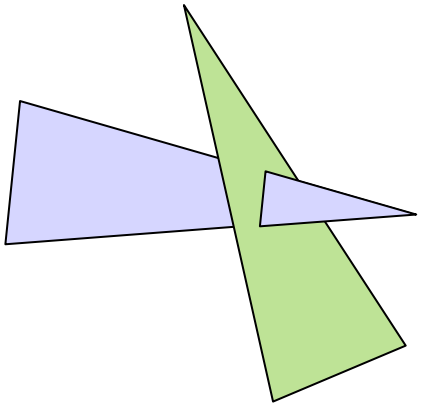
Painter's Algorithm



- Display polygons in back-to-front order
- Sort polygons by z-value
  - Which vertex?
  - $O(n \log n)$
- Problems…

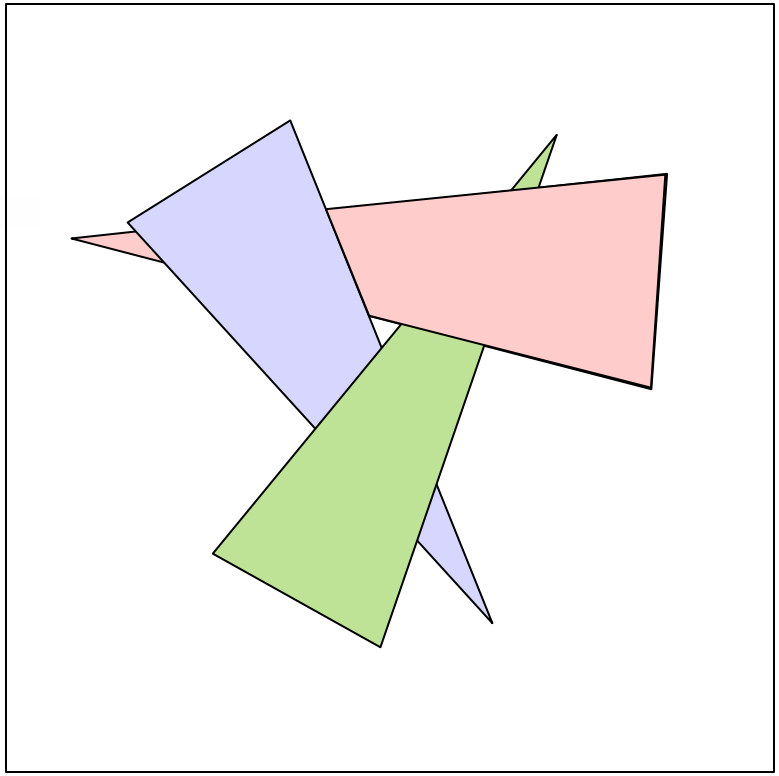






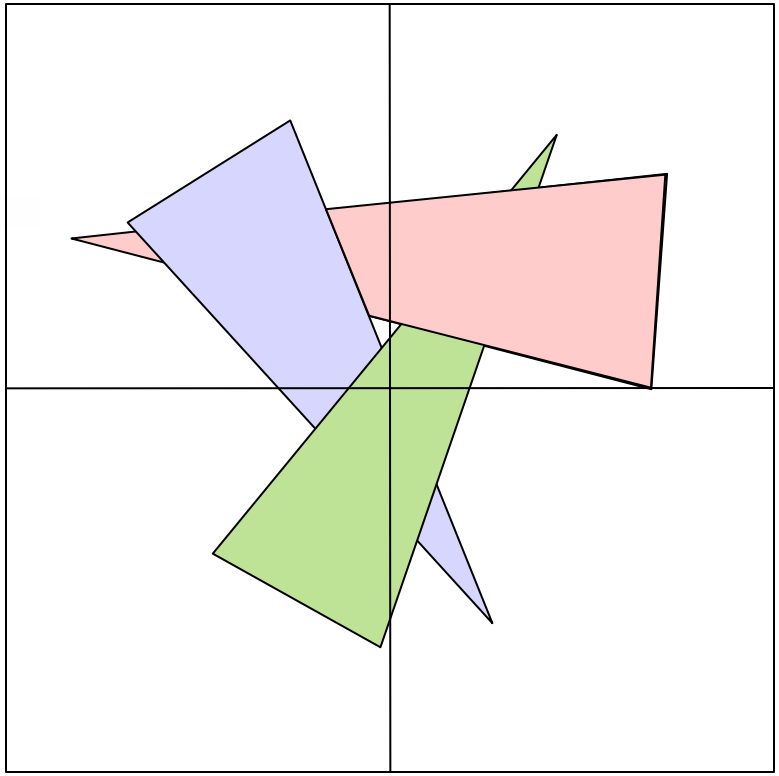
Quadtree Algorithm

- Sort polygons
- Subdivide screen until each region contains one or zero edges
- Invented by John Warnock in 1969



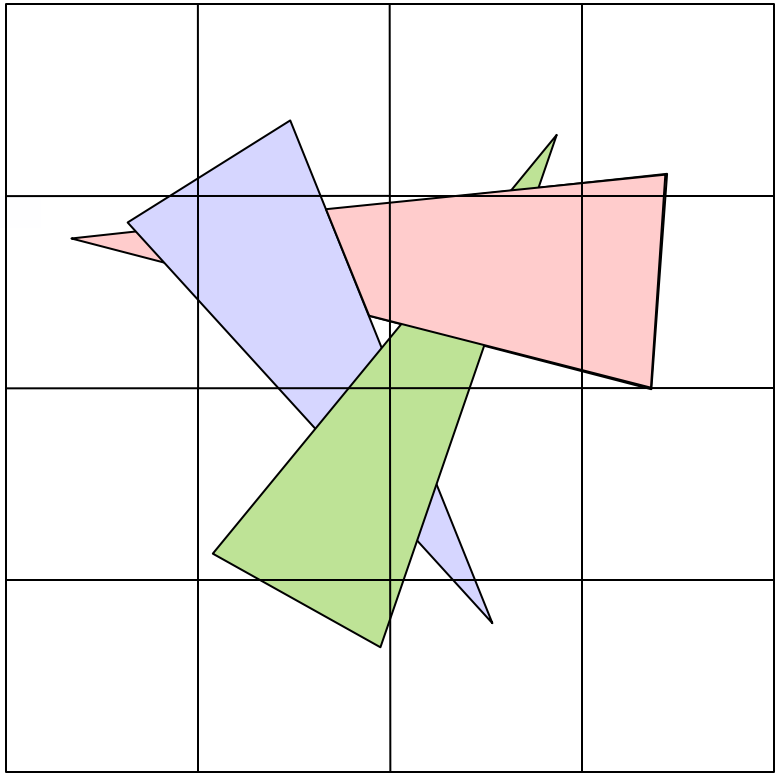
Quadtree Algorithm

- Sort polygons
- Subdivide screen until each region contains one or zero edges
- Invented by John Warnock in 1969



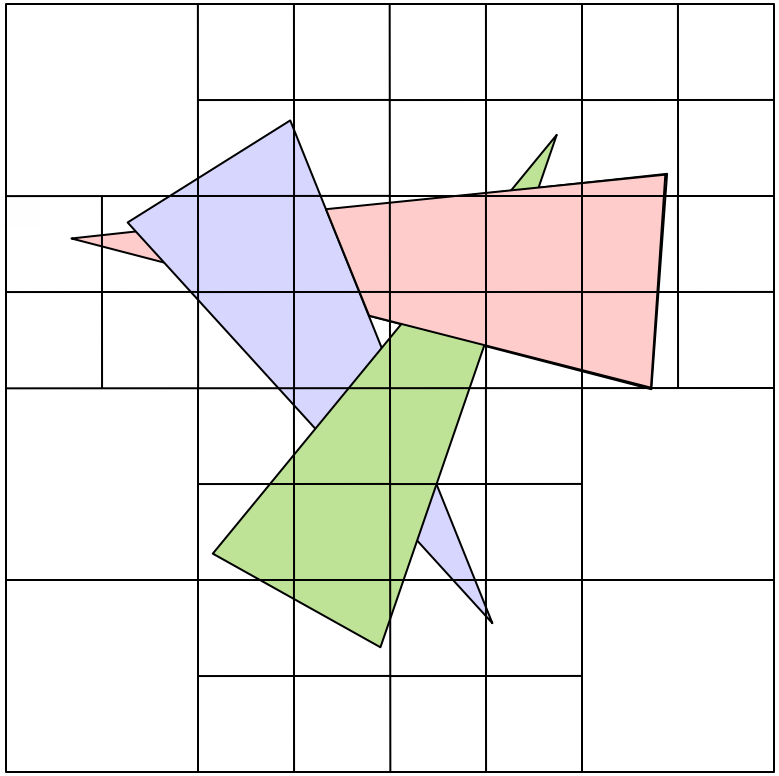
Quadtree Algorithm

- Sort polygons
- Subdivide screen until each region contains one or zero edges
- Invented by John Warnock in 1969



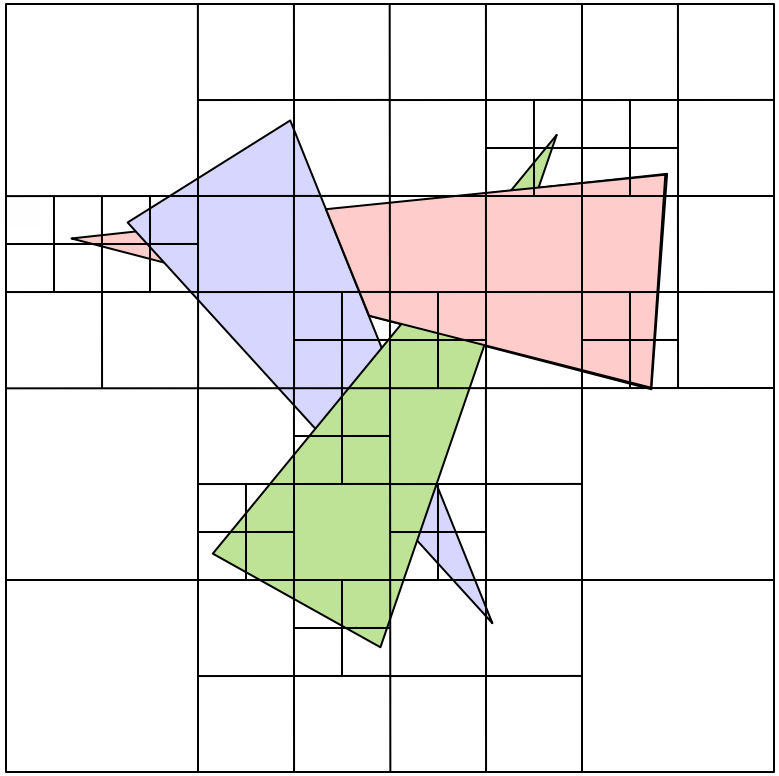
Quadtree Algorithm

- Sort polygons
- Subdivide screen until each region contains one or zero edges
- Invented by John Warnock in 1969



Quadtree Algorithm

- Sort polygons
- Subdivide screen until each region contains one or zero edges
- Invented by John Warnock in 1969



Z-Buffer

Key Observation: Each pixel displays color of only one triangle, ignores everything behind it

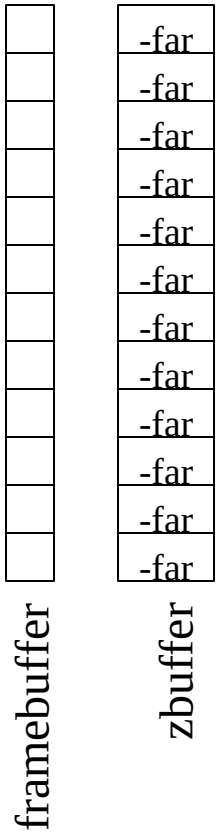
- Don’t need to sort triangles, just find for each pixel the closest triangle
- Z-buffer: one fixed or floating point value per pixel
- Algorithm:

For each rasterized fragment (x,y)

If z > zbuffer(x,y) then

framebuffer(x,y) = fragment color

zbuffer(x,y) = z



Z-Buffer

Key Observation: Each pixel displays color of only one triangle, ignores everything behind it

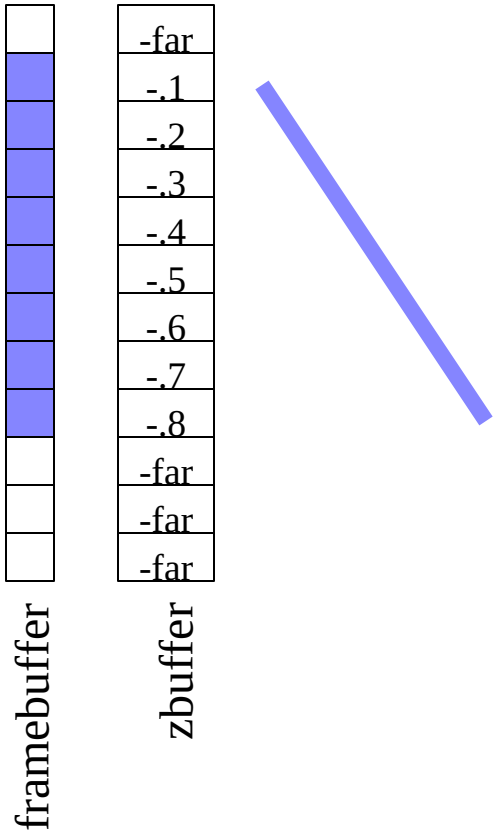
- Don’t need to sort triangles, just find for each pixel the closest triangle
- Z-buffer: one fixed or floating point value per pixel
- Algorithm:

For each rasterized fragment (x,y)

If z > zbuffer(x,y) then

framebuffer(x,y) = fragment color

zbuffer(x,y) = z



Z-Buffer

Key Observation: Each pixel displays color of only one triangle, ignores everything behind it

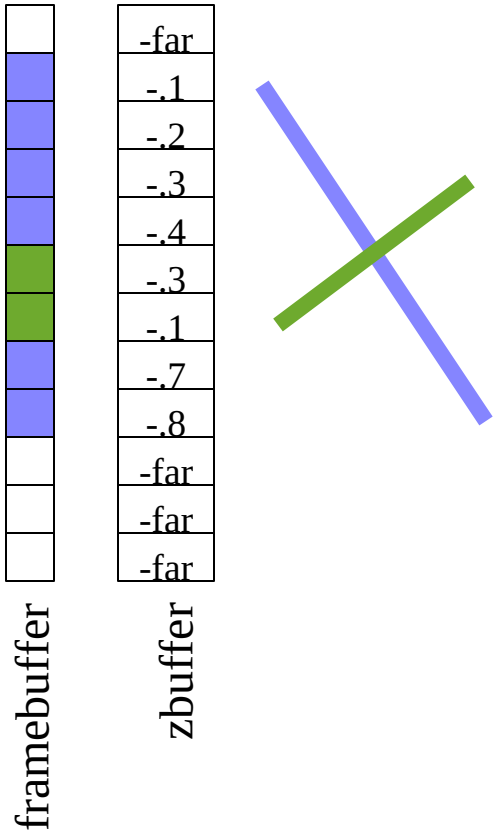
- Don't need to sort triangles, just find for each pixel the closest triangle
- Z-buffer: one fixed or floating point value per pixel
- Algorithm:

For each rasterized fragment (x,y)

If z > zbuffer(x,y) then

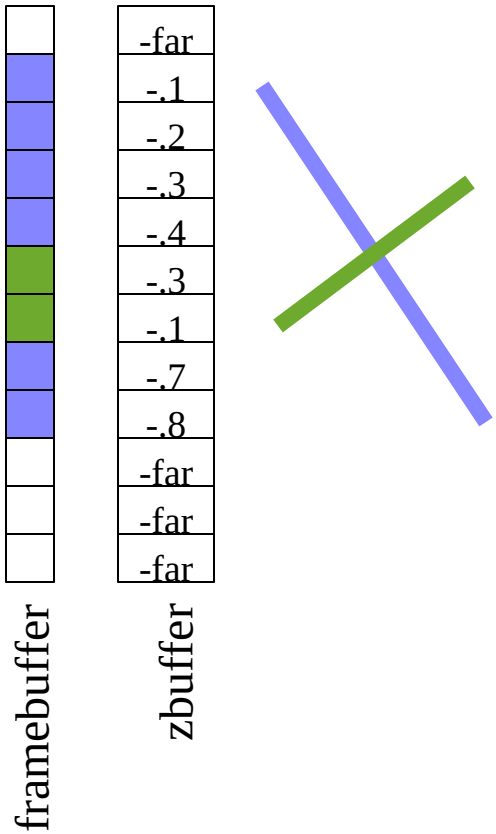
framebuffer(x,y) = fragment color

zbuffer(x,y) = z



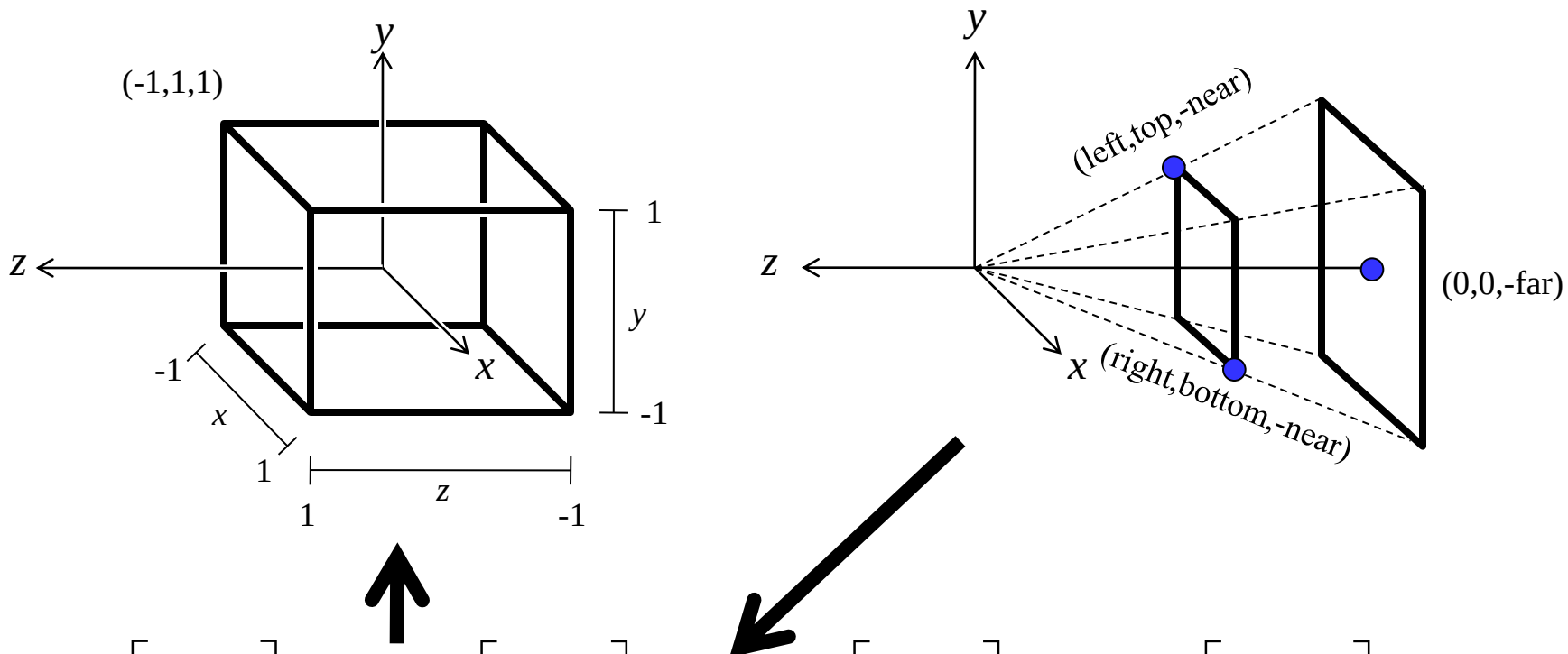
Z-Buffer

- Get fragment z-values by interpolating z-values at vertices during rasterization
- Perspective projection destroys z-values, setting them all to *–d*
- Need a perspective distortion that preserves at least the ordering of z-values



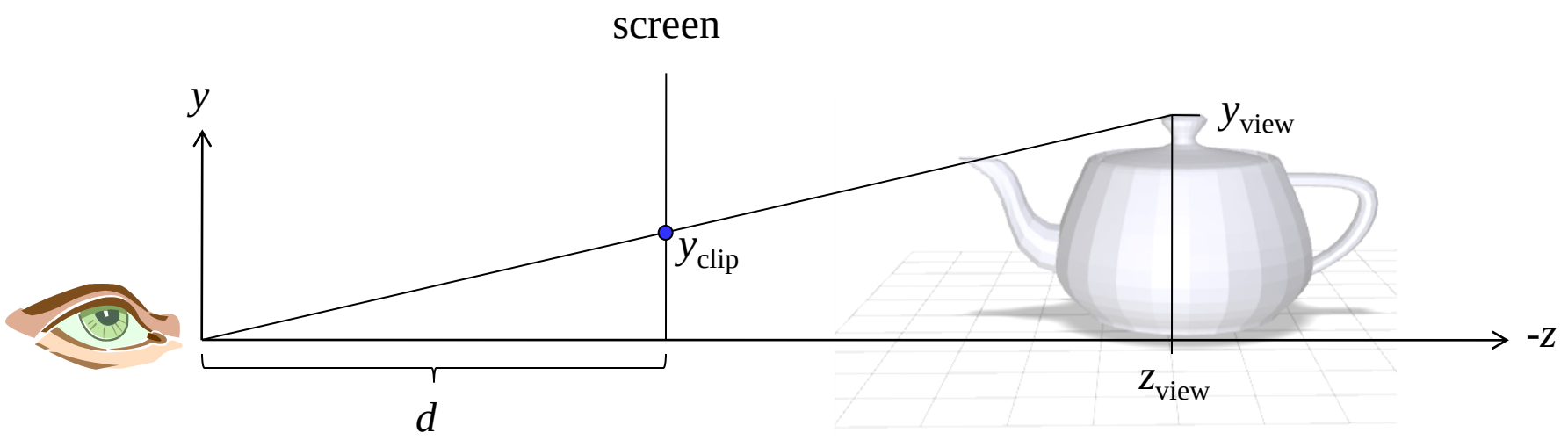
Normalized View Volume

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



Screen Coords | W2V | Clip Coords | Persp | Viewing Coords | View | World Coords | Model | Model Coords

Perspective Projection

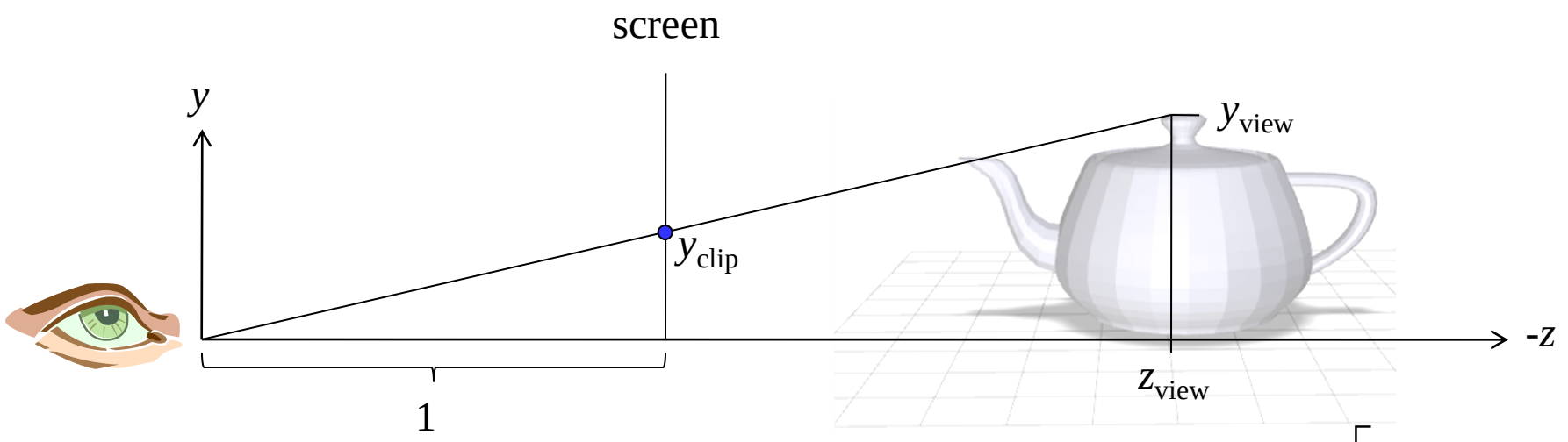


$$\frac{y_{clip}}{d} = \frac{y_{view}}{-z_{view}}$$

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

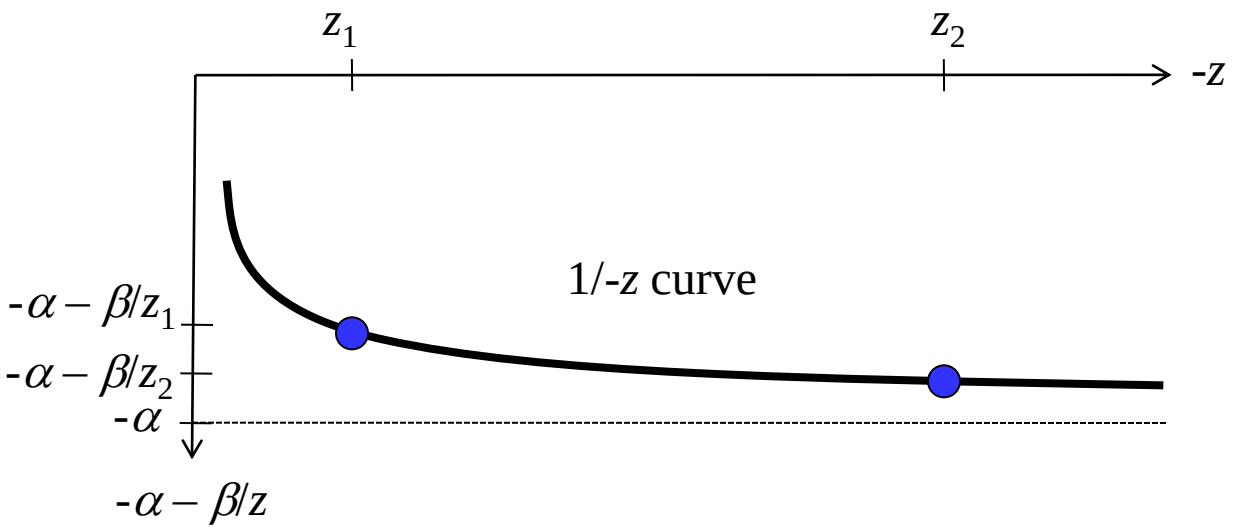
$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1/d & 0 \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \\ -z_{view}/d \end{bmatrix} \equiv \begin{bmatrix} \frac{x_{view}}{-z_{view}/d} \\ \frac{y_{view}}{-z_{view}/d}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

Perspective Distortion



$$y_{clip} = \frac{y_{view}}{-z_{view}} \Rightarrow \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \alpha & \beta \\ & & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{view} \\ y_{view} \\ \alpha z_{view} + \beta \\ -z_{view} \end{bmatrix} \equiv \begin{bmatrix} \frac{x_{view}}{-z_{view}} \\ \frac{y_{view}}{-z_{view}} \\ -\alpha - \frac{\beta}{z_{view}} \\ 1 \end{bmatrix}$$

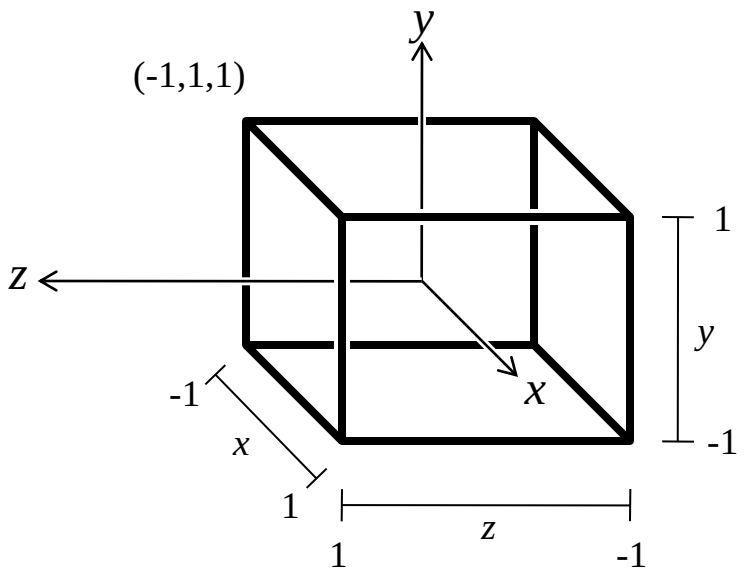
Distorted z-Values

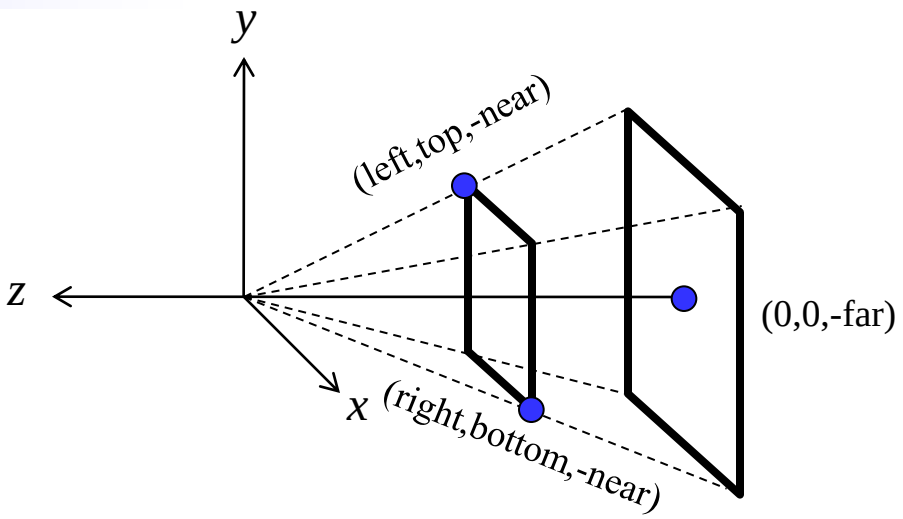


if $z_1 > z_2$ then
$-\alpha - \beta/z_1 > -\alpha - \beta/z_2$

$$y_{\text{clip}} = \frac{y_{\text{view}}}{-z_{\text{view}}} \Rightarrow \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \alpha & \beta \\ & & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{\text{view}} \\ y_{\text{view}} \\ z_{\text{view}} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{\text{view}} \\ y_{\text{view}} \\ \alpha z_{\text{view}} + \beta \\ -z_{\text{view}} \end{bmatrix} \equiv \begin{bmatrix} \frac{x_{\text{view}}}{-z_{\text{view}}} \\ \frac{y_{\text{view}}}{-z_{\text{view}}} \\ -\alpha - \frac{\beta}{z_{\text{view}}} \\ 1 \end{bmatrix}$$

Normalized Perspective Distortion

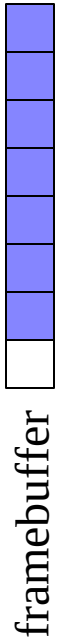




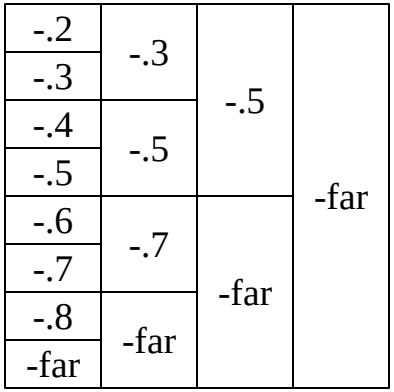
$$\begin{bmatrix} \frac{-2\times near}{right-left} & & \frac{right+left}{right-left} & \\ & \frac{-2\times near}{top-bottom} & \frac{top+bottom}{top-bottom} & \\ & & -\frac{far+near}{far-near} & \frac{2\times far\times near}{far-near} \\ & & -1 & 0 \end{bmatrix}$$

Hierarchical Z-Buffer

- Invented by Ned Green in 1994
- Creates a MIP-map of the z-buffer
  - z-value equal to farthest z-value of its four children
- Before rasterizing a triangle…
  - Check z-value of its nearest vertex against z-value of the smallest quadtree cell containing the triangle
  - If z-test fails, then the entire triangle is hidden and need not be rasterized
- Works best when displaying front-to-back



framebuffer



hierarchical zbuffer