On the Midterm Exam

- **Monday, 10/17 in class**
- **Closed book and closed notes**
- **One-side and one page cheat sheet is allowed**
- **A calculator is allowed**
- **Covers the topics until the class on Wednesday, 10/12**

Take-home Quiz

It will be posted in dropbox Wednesday, Oct 12 at 5pm

Submit your solution through dropbox at 2am, Thursday, Oct13.

You can either type or take a picture of your handwritten answer. Please keep your original hard copy for reference.

Topics

Perspective Projection

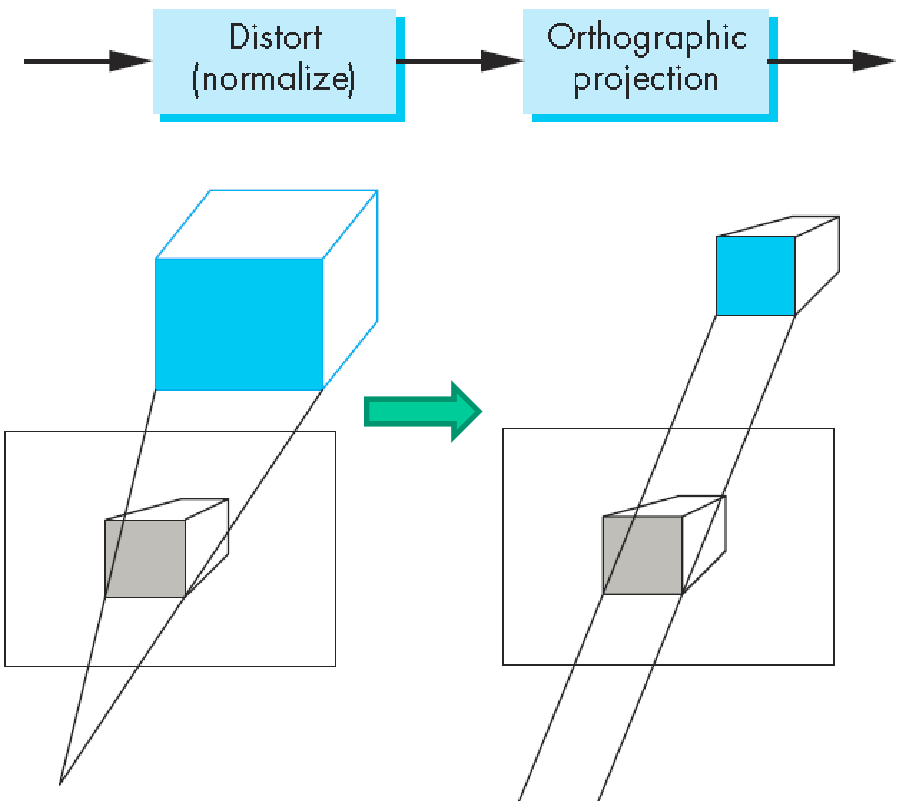
Review for Midterm

Projections and View Normalization

The default projection is **orthogonal (orthographic) projection**

Most graphics systems use ***view normalization***

- All other views are converted to the orthographic view by distorting the objects -- ***normalization***
- Allows use of the same pipeline for all views



E. Angel and D. Shreiner

Oblique Projections

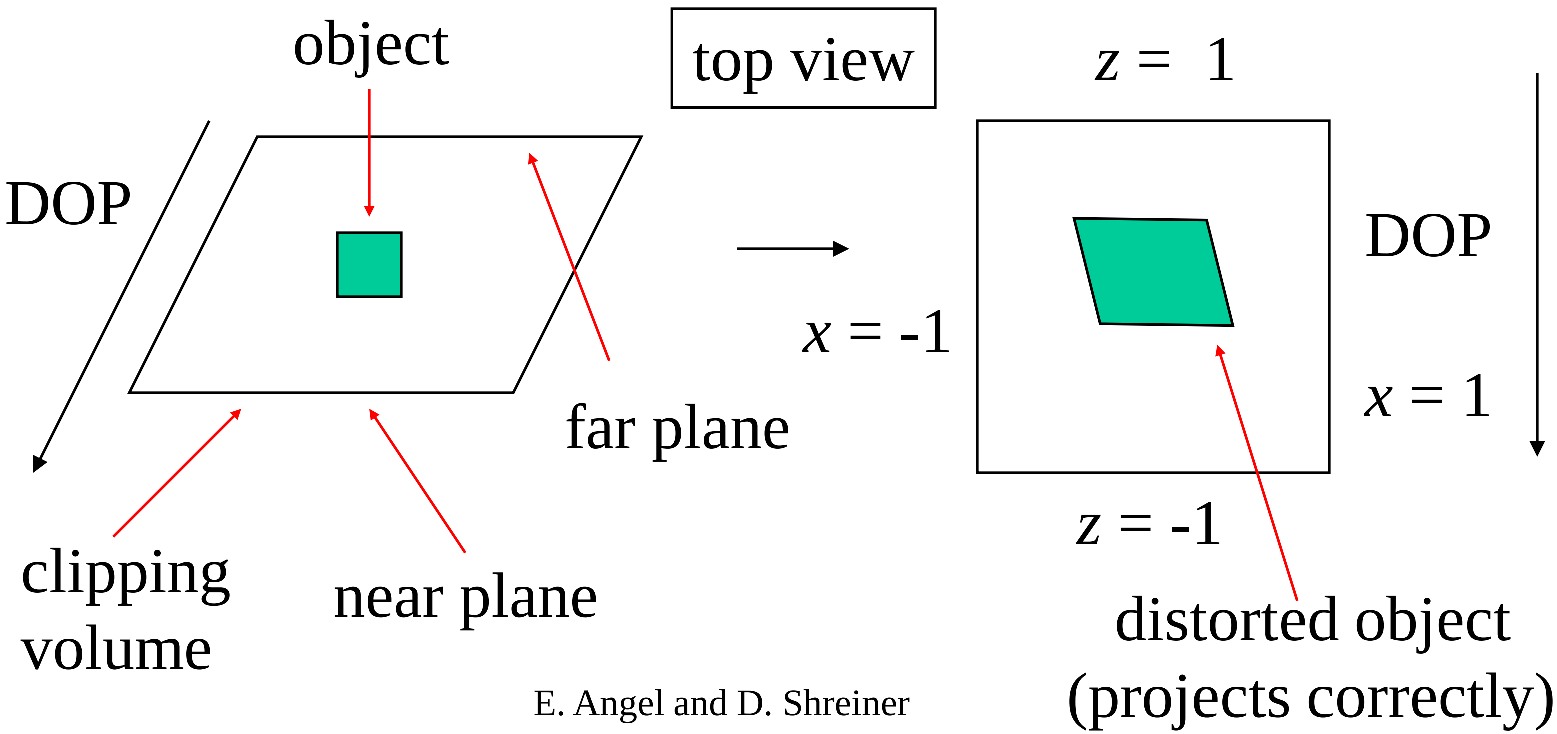
The OpenGL projection functions cannot produce general parallel projections – the oblique projection

Oblique Projection = Shear + Orthogonal Projection

$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\ \mathbf{STH}(\theta,\phi)$$

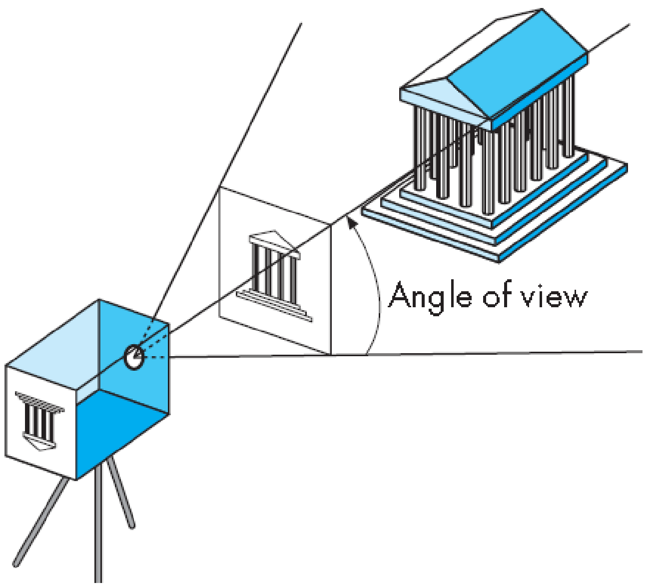
Effect on Clipping

The projection matrix P = STH transforms the original clipping volume to the default clipping volume

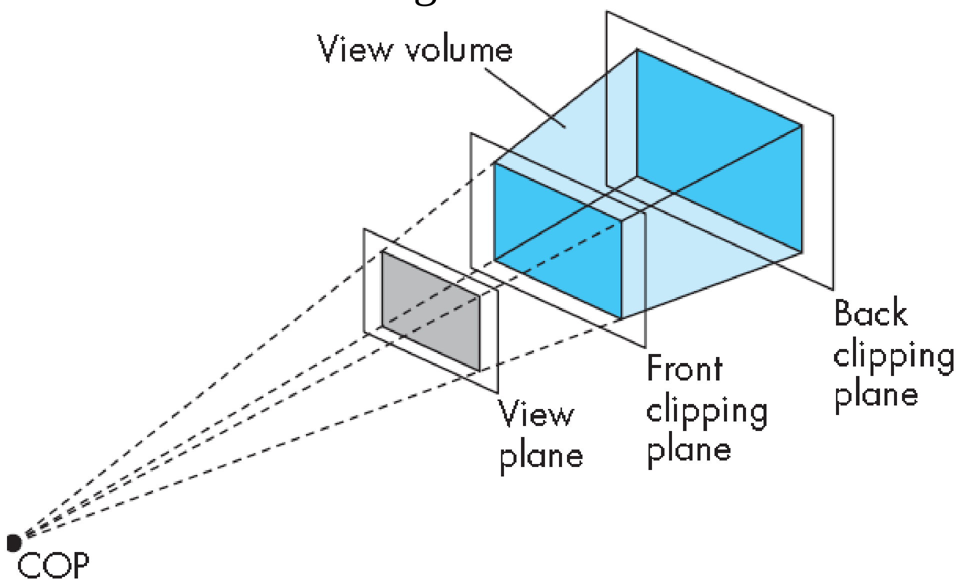


Perspective with OpenGL

View volume is determined by the angle of view (field of view)



Viewing frustum



Perspective Transformation

Perspective transformation is

- **Not linear**
- **Not affine**
- **Not reversible**

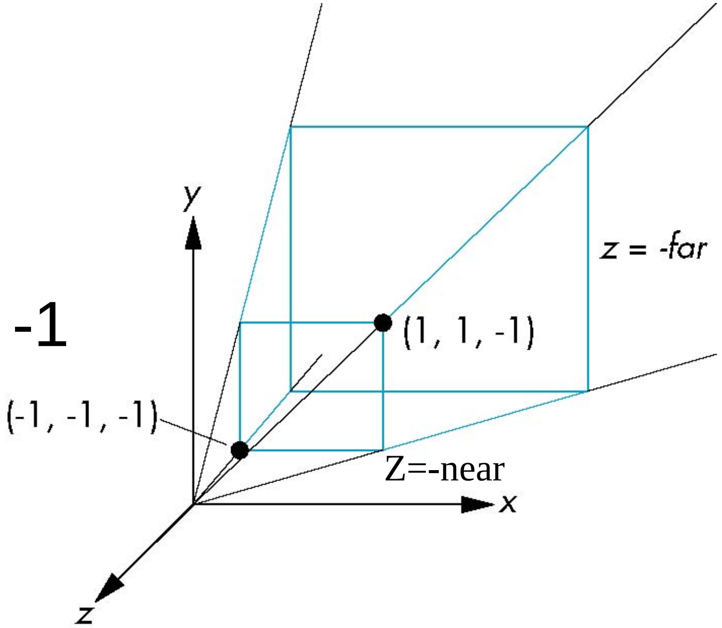
Simple Perspective with OpenGL

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 projection matrix is

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

where d = -1



E. Angel and D. Shreiner:

Perspective Projection and Normalization

The projection can be achieved by ***view normalization*** and an ***orthographic projection***

A point P=(x, y, z, 1) is project to a new point Q on the projection plane as

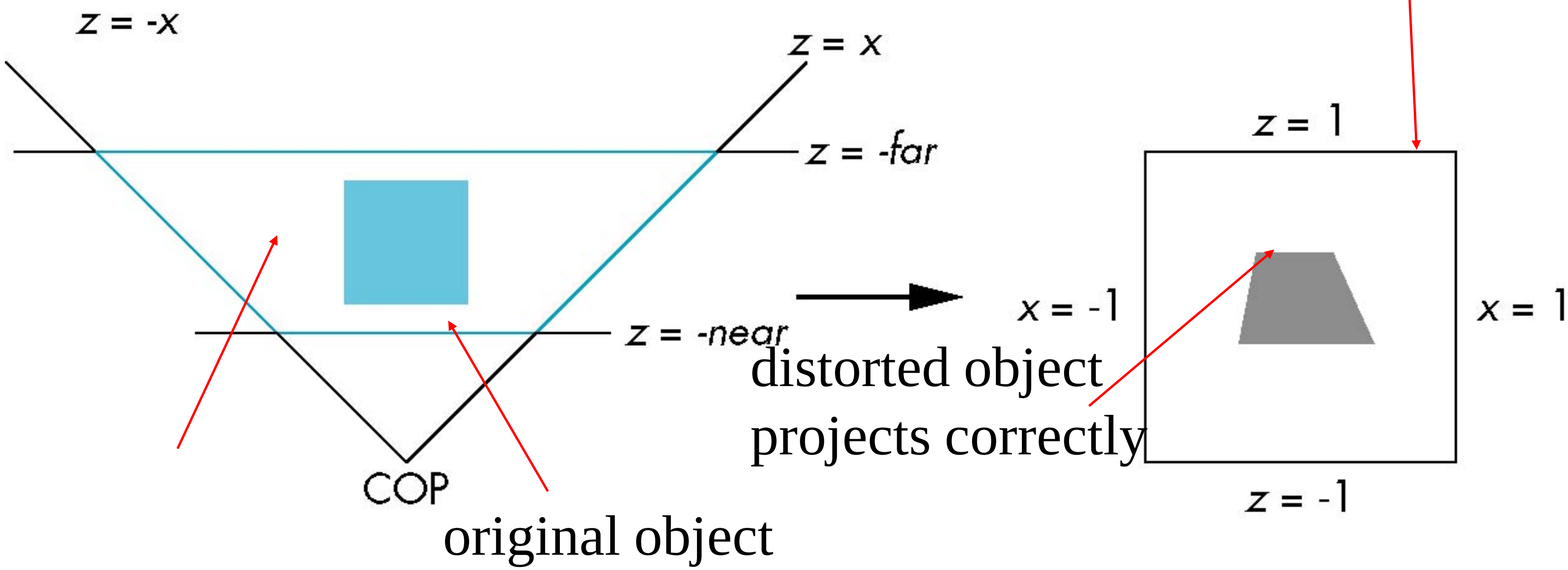
$$Q = \mathbf{M}_{\text{orth}}\mathbf{N}\mathrm{P}$$

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

Normalization Transformation

Original clipping volume

Normalized clipping volume



OpenGL Perspective

How do we handle the asymmetric frustum?

Convert the frustum to a symmetric one by performing a shear followed by a scaling to get the normalized perspective volume.

The final perspective matrix

$$\mathbf{M}_{\boldsymbol{p}} = \mathbf{NSH} = \begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{left+right}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{bottom+top}{top-bottom} & 0 \\ 0 & 0 & \frac{near+far}{near-far} & \frac{2near*far}{near-far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

A point P=(x, y, z, 1) is project to a new point Q on the projection plane as

$$Q = \mathbf{M}_{\mathrm{orth}}\mathbf{M}_{\boldsymbol{p}}\mathrm{P}$$

An Example

A camera located at $\mathbf{eye} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ is looking at the origin of the object frame $\mathbf{at} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ with the up vector defined as $\mathbf{up} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$. The frustum is defined by left = -1, right =2, bottom= -1, top = 2, near = 2, and far = 3.

For a point $P = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ in the object frame, where its projection Q on the projection plane z= -1?

An Example

$$Q = \mathbf{M}P$$

What types of transformation we need?

- **Model-view**
- **projection**

$$Q = \mathbf{M}_p \mathbf{M}_v P$$

An Example – Calculate Model View Matrix

How to build the model view matrix?

LookAt(eye, at, up)

Step 1: Calculate the normalized view plane normal

$$\mathbf{vpn} = \mathbf{at} - \mathbf{eye} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

$$\mathbf{n} = \frac{\mathbf{vpn}}{|\mathbf{vpn}|} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

Step 2: Calculate the other two vectors

$$\mathbf{u} = \frac{\mathbf{up} \times \mathbf{n}}{|\mathbf{up} \times \mathbf{n}|} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\mathbf{v} = \frac{\mathbf{n} \times \mathbf{u}}{|\mathbf{n} \times \mathbf{u}|} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

An Example – Calculate Model View Matrix

LookAt(eye, at, up)

Step 3: Construct the model view matrix

$$\mathbf{M}_v = \begin{bmatrix} -u_x & -u_y & -u_z & -\mathbf{u} \cdot \mathbf{vpn} \\ v_x & v_y & v_z & \mathbf{v} \cdot \mathbf{vpn} \\ -n_x & -n_y & -n_z & -\mathbf{n} \cdot \mathbf{vpn} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

An Example – Calculate Projection Matrix

How to build the projection matrix?

Frustum(left, right, bottom, top, near, far);

Note that this is the case of asymmetric frustum

$$\mathbf{M}_p = \mathbf{M}_{orth}\mathbf{NSH} =$$

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

$$= \begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{left+right}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{bottom+top}{top-bottom} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{\frac{4}{3}} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{4}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

An Example

$$Q = \mathbf{M}_p \mathbf{M}_v P = \begin{bmatrix} \frac{4}{3} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{4}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{13}{3} \\ \frac{5}{3} \\ 0 \\ -1 \end{bmatrix}$$

Hidden-Surface Removal: Z-buffer algorithm

The circle should be in front of the triangle

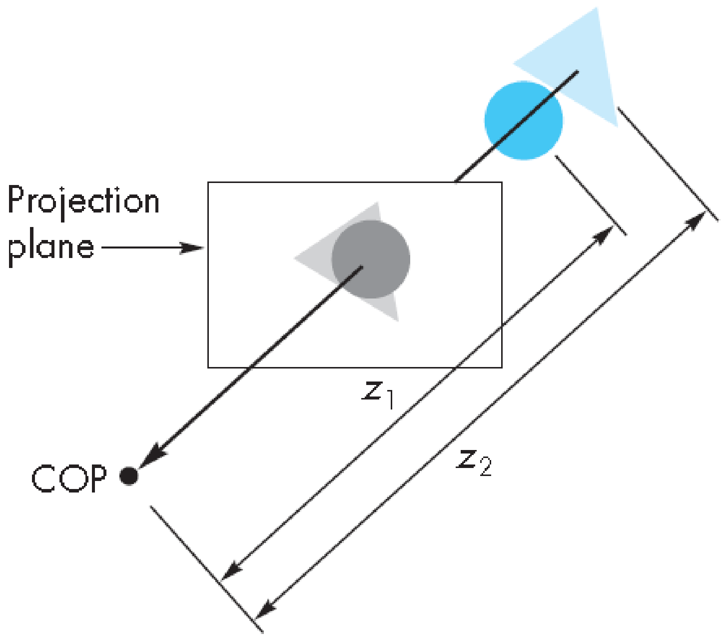
Problem:

The triangle may appear earlier in the pipeline

Need to determine which point is the closest

Z-buffer algorithm for hidden-surface removal

- Belongs to image-space algorithm
- Determines the relationship among points on each projector
- Works well in the pipeline



E. Angel and D. Shreiner

Hidden-Surface Removal: Z-buffer algorithm

Hidden surface removal works if we first apply the normalization transformation

Recall the perspective projection

$x'' = -x/z$

$y'' = -y/z$

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

⇒ Perspective projection is nonlinear.

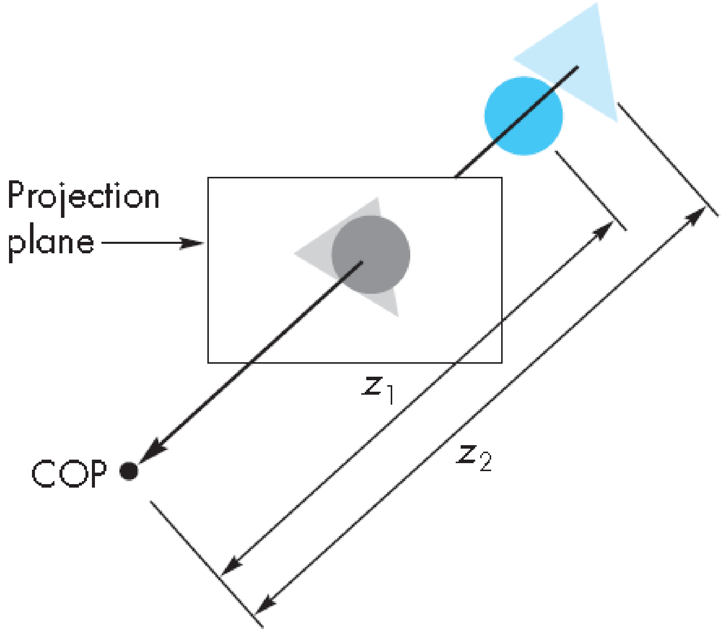
For $z_1 > z_2$, the projections $z_1'' > z_2''$

⇒ The order of depth is preserved.

Hidden-Surface Removal: Z-buffer algorithm

The color of the pixel in the color buffer is determined by the point closest to the viewer – with smaller depth

$$Q = \mathbf{NSH}P$$



E. Angel and D. Shreiner

Reading Assignments

Chapter 4.9 – 4.10, Angel & Shreiner

Chapter 5, Shreiner et al.