Computer Graphics (CS 4731)
Shadows

Joshua Cuneo

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

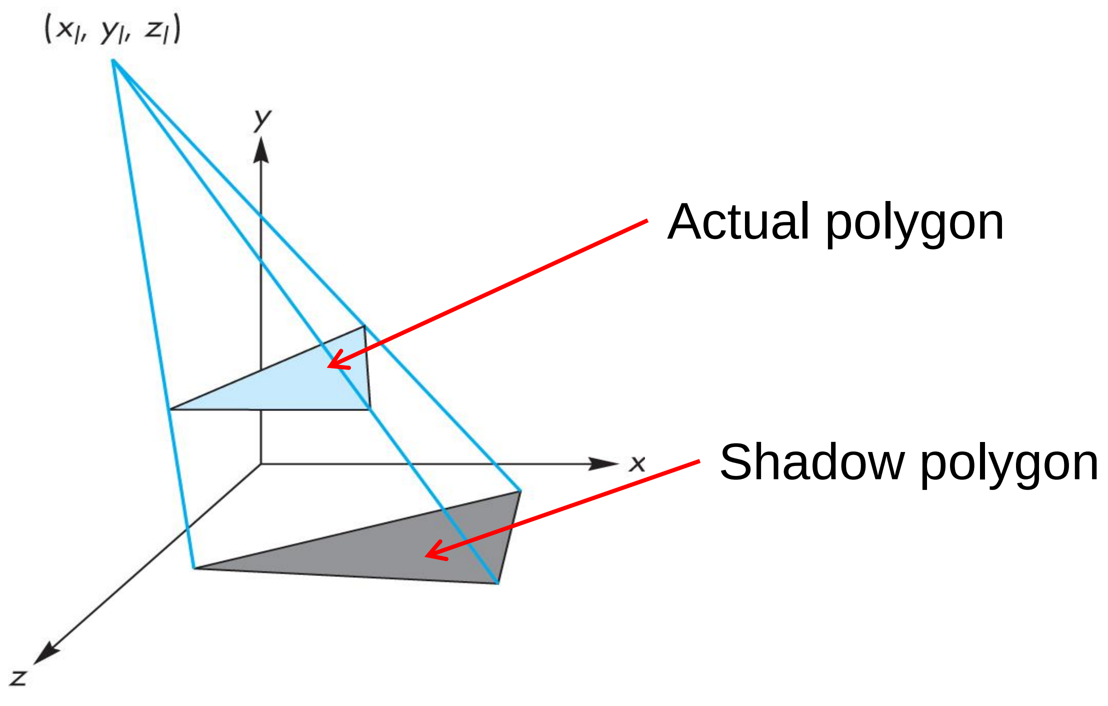
Slides Courtesy of Emmanuel Agu (WPI)

Introduction to Shadows

- Two popular shadow rendering methods:
  1. Shadows as texture (projection)
  2. Shadow buffer

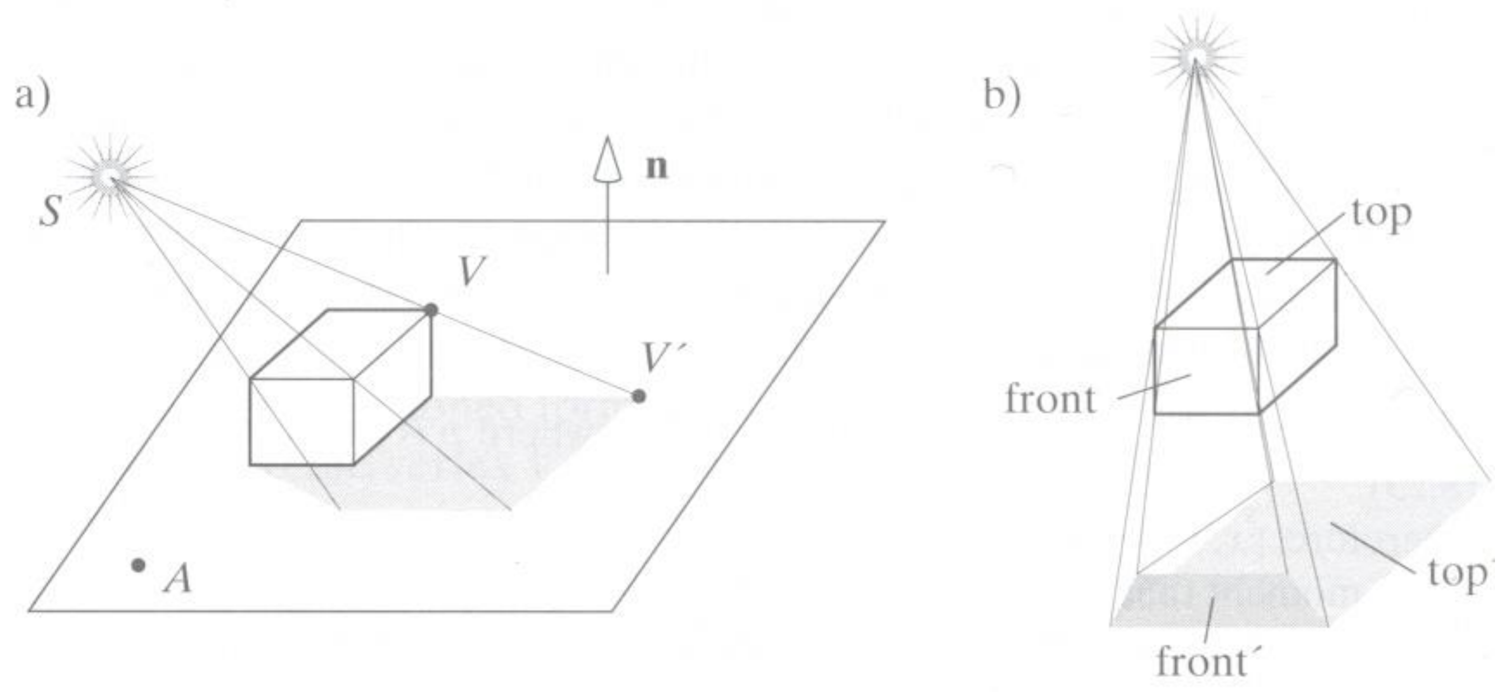
Projective Shadows

- Oldest method: Used in early flight simulators
- Projection of polygon is polygon called **shadow polygon**



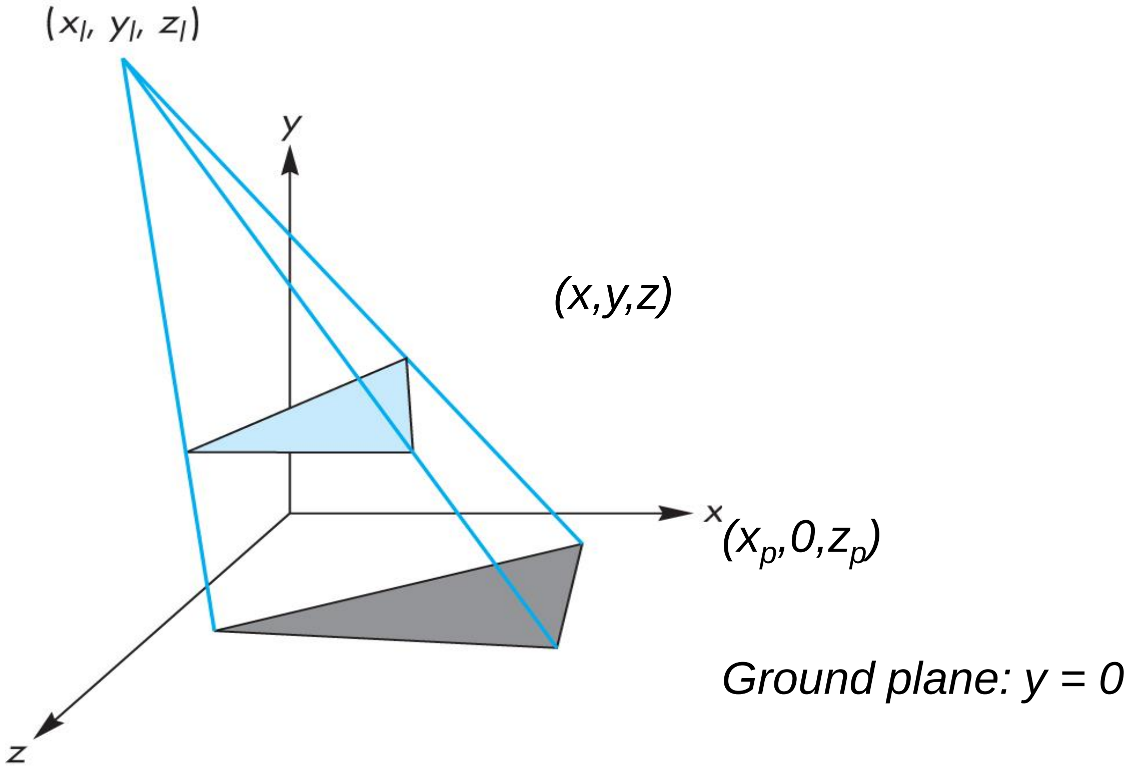
Projective Shadows

- Works for flat surfaces illuminated by point light
- For each face, project vertices **V** to find **V’** of shadow polygon



Projective Shadows for Polygon

1. If light is at (x_l, y_l, z_l)
2. Vertex at (x, y, z)
3. Would like to calculate shadow polygon vertex V projected onto ground at $(x_p, 0, z_p)$

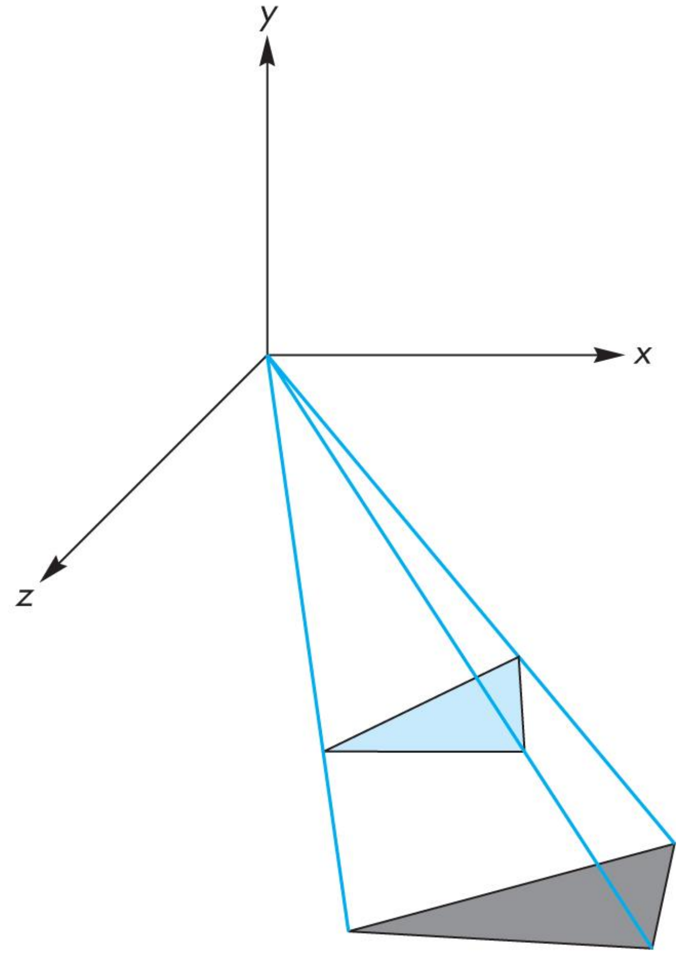


Projective Shadows for Polygon

- If we move original polygon so that light source is at origin
- Matrix *M* projects a vertex V to give

  its projection V' in shadow polygon

$$m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \dfrac{1}{-y_l} & 0 & 0 \end{bmatrix}$$



Building Shadow Projection Matrix

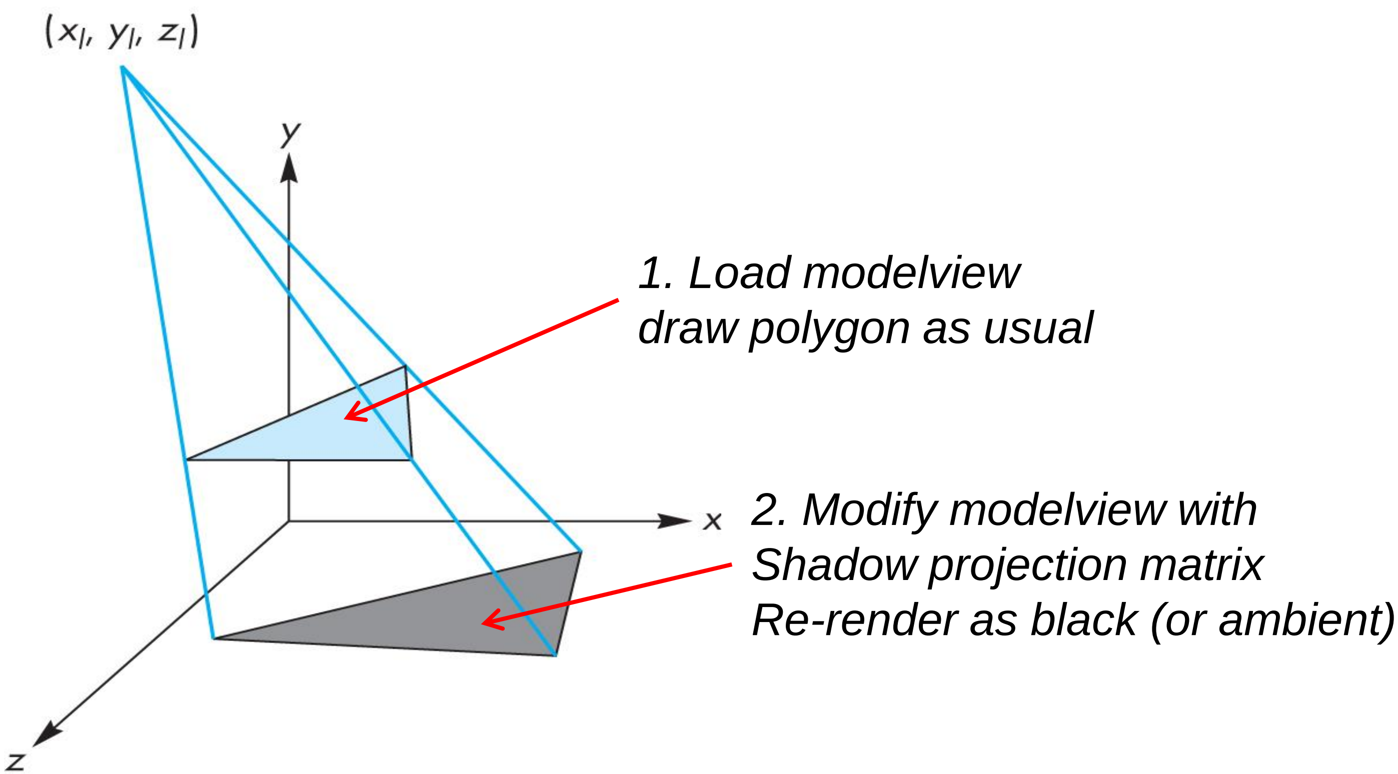
1. Translate source to origin with $T(-x_l, -y_l, -z_l)$
2. Perspective projection
3. Translate back by $T(x_l, y_l, z_l)$

$$M = \begin{bmatrix} 1 & 0 & 0 & x_l \\ 0 & 1 & 0 & y_l \\ 0 & 0 & 1 & z_l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \dfrac{1}{-y_l} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_l \\ 0 & 1 & 0 & -y_l \\ 0 & 0 & 1 & -z_l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

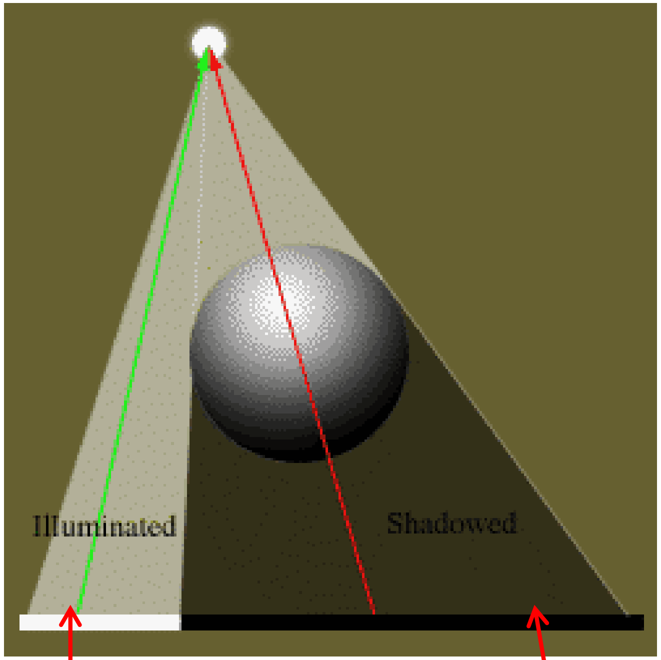
Final matrix that projects
Vertex V onto V' in shadow polygon

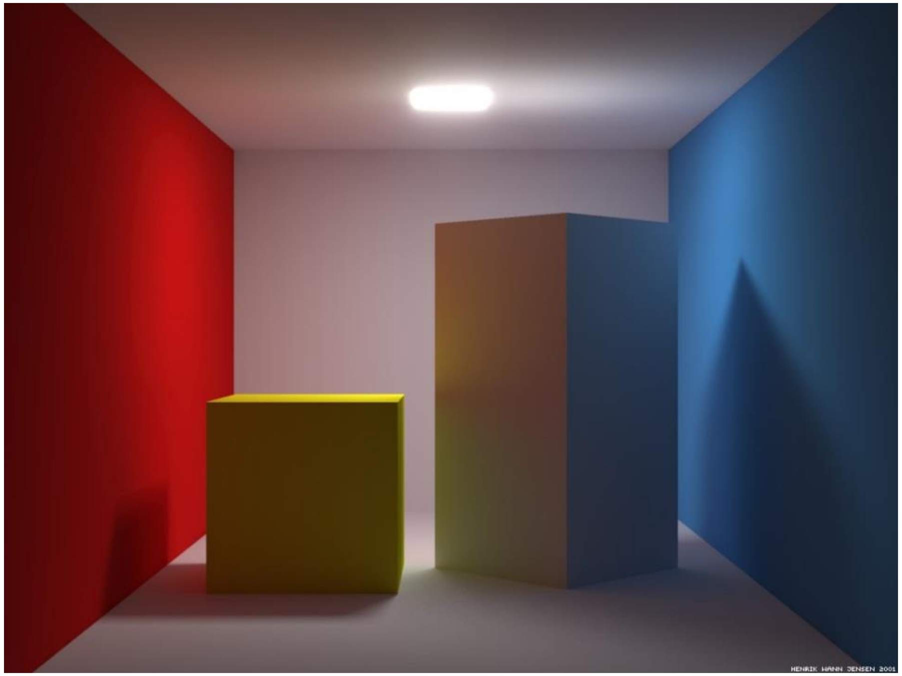
Implementing Shadows

- Render the polygon
- Then load shadow projection matrix, change color to black, re-render polygon



Coloring Shadows





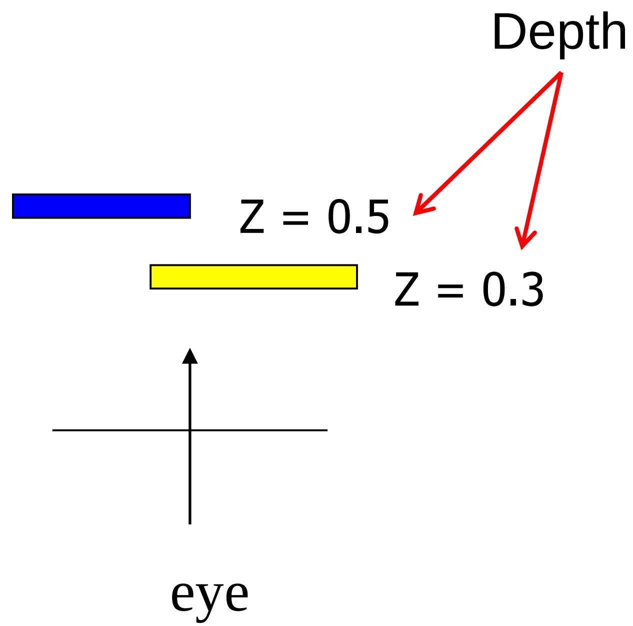
Use ambient + diffuse + specular components

Use just ambient component

Shadow Buffer (Z Buffer)

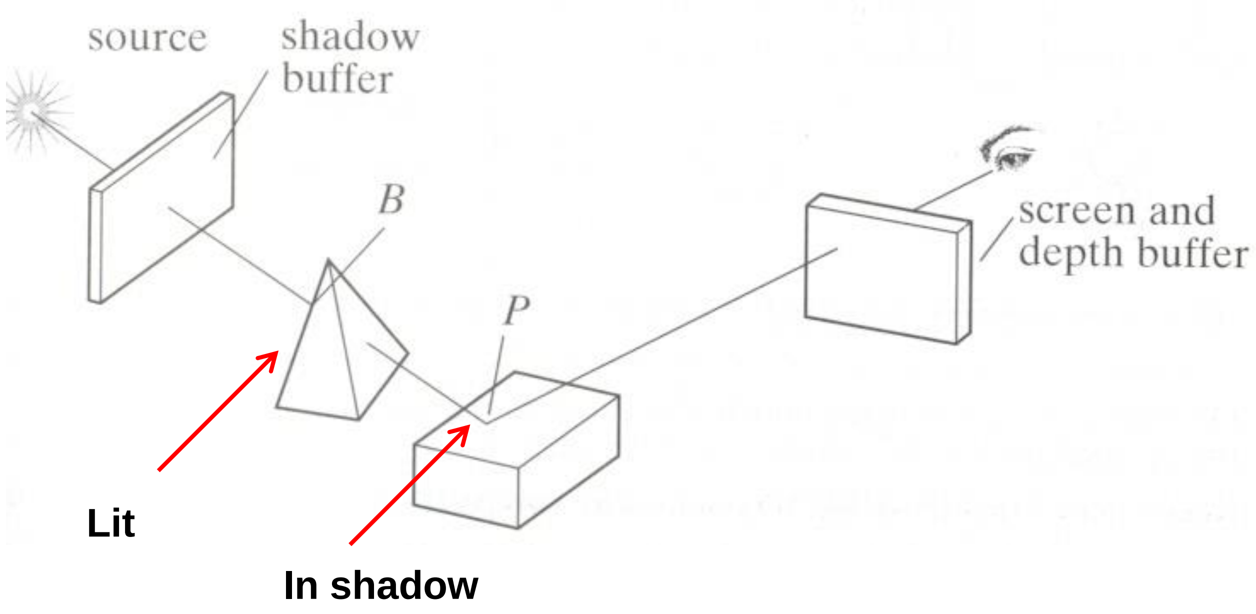
- **Depth:** While drawing objects, depth buffer stores distance of each polygon from viewer
- **Why?** If multiple polygons overlap a pixel, only closest one polygon is drawn

1.0	1.0	1.0	1.0
1.0	0.3	0.3	1.0
0.5	0.3	0.3	1.0
0.5	0.5	1.0	1.0



Shadow Buffer Theory

- Along each path from light
  - Only closest object is lit
  - Other objects on that path in shadow
- Shadow buffer stores closest object on each path

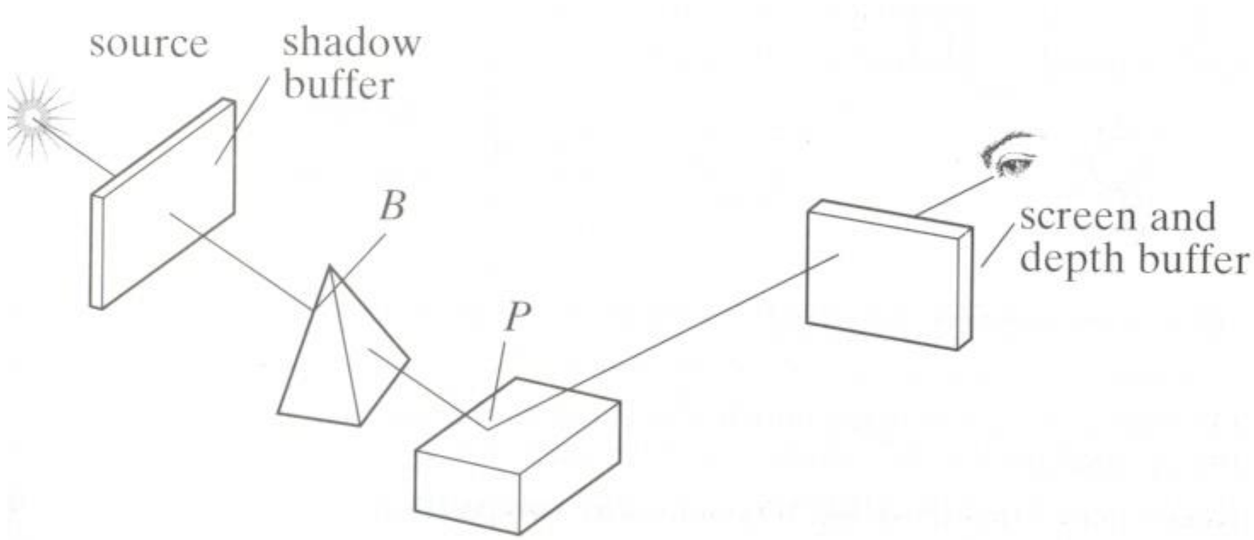


Shadow Buffer Approach

- Rendering in two stages:
  - Loading shadow buffer
  - Render the scene

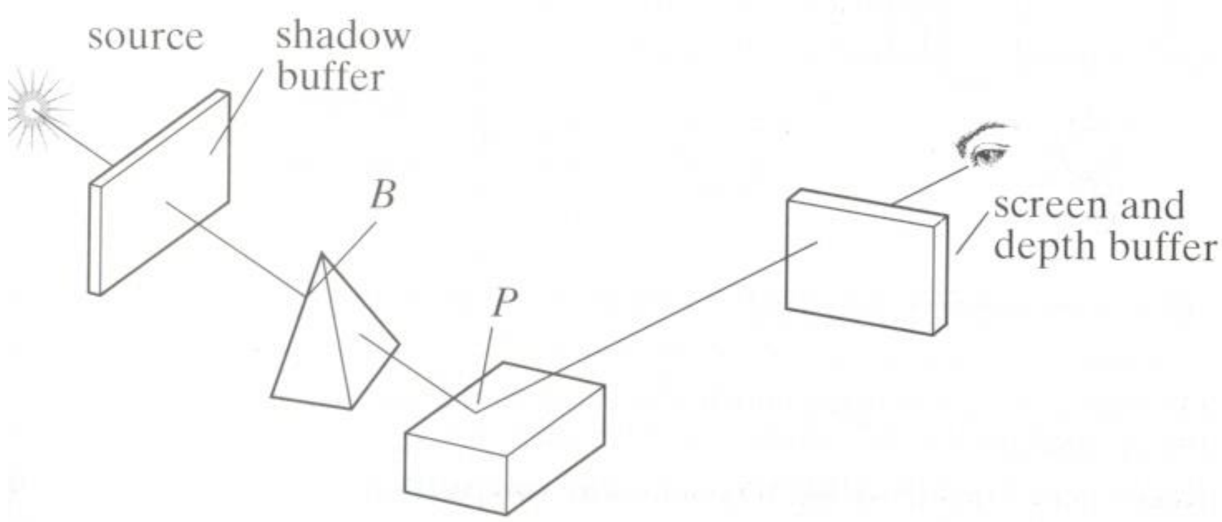
Loading Shadow Buffer

- Initialize each element to 1.0
- Position a camera at light source
- Rasterize each face in scene updating closest object
- Shadow buffer tracks smallest depth on each path



Loading Shadow Buffer

- Shadow buffer calculation is independent of eye position
- In animations, shadow buffer loaded once
- If eye moves, no need for recalculation
- If objects move, recalculation required



Shadow Buffer (Rendering Scene)

- Render scene using camera as usual
- While rendering a pixel find:
  - pseudo-depth D from light source to P
  - Index location [i][j] in shadow buffer, to be tested
  - Value d[i][j] stored in shadow buffer
- If d[i][j] < D (other object on this path closer to light)
  - point P is in shadow
  - lighting = ambient
- Otherwise, not in shadow
  - Lighting = amb + diffuse + specular