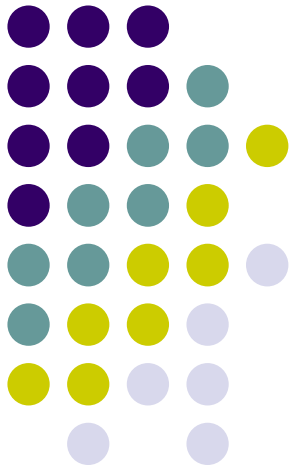
Computer Graphics
CS 543 – Lecture 8 (Part 3)
Shadows

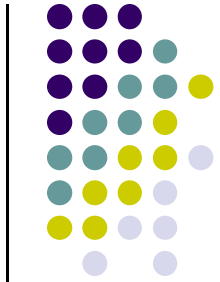
Prof Emmanuel Agu

Computer Science Dept.

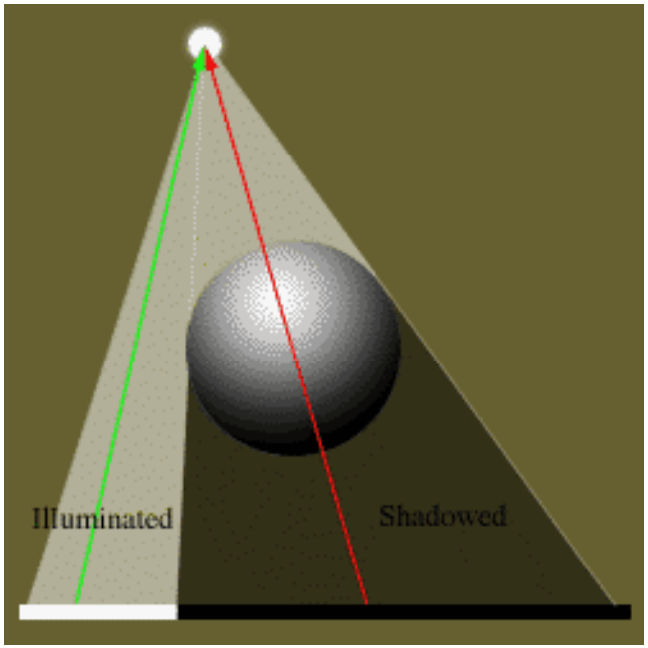
Worcester Polytechnic Institute (WPI)



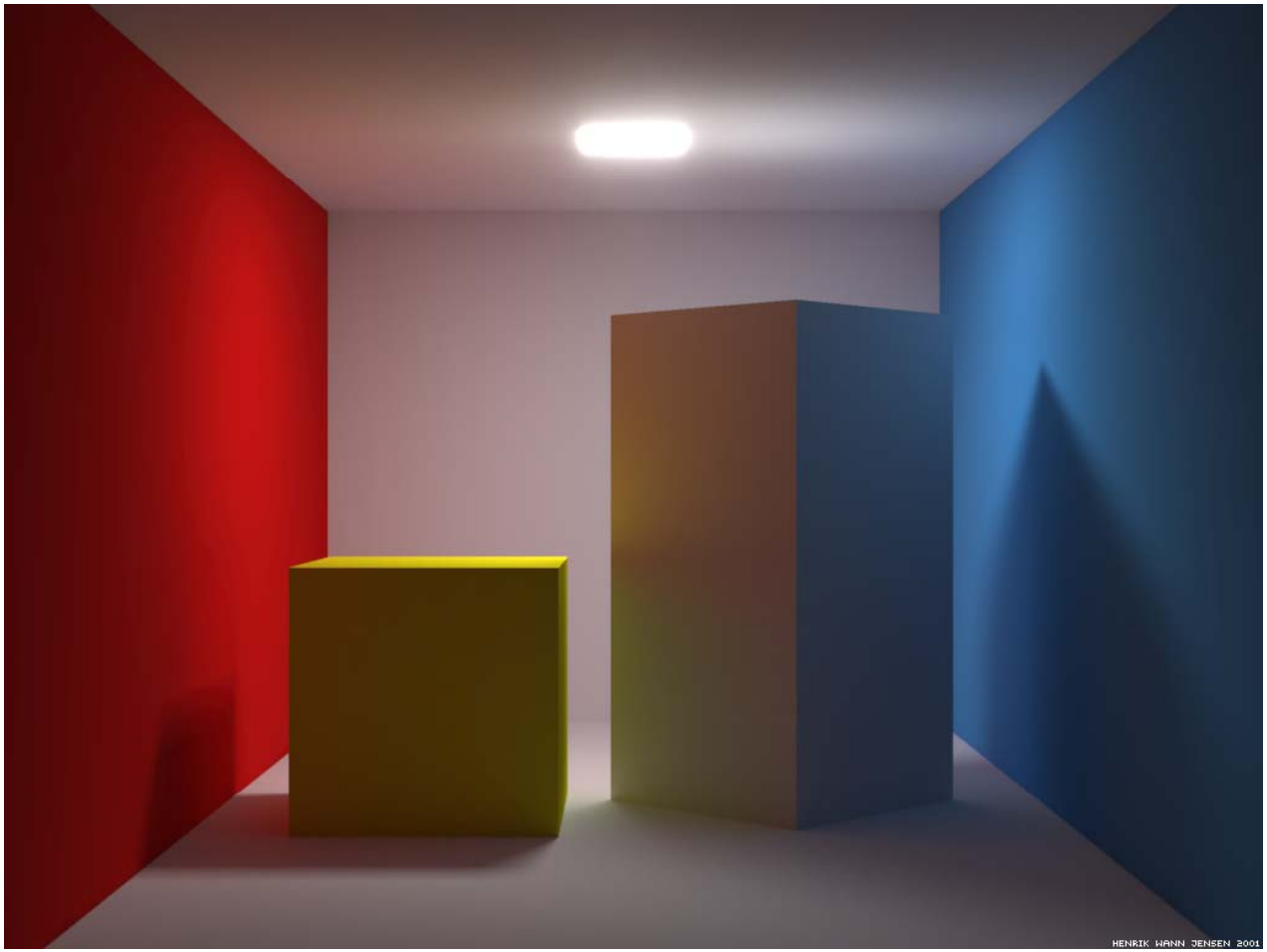
Introduction to Shadows



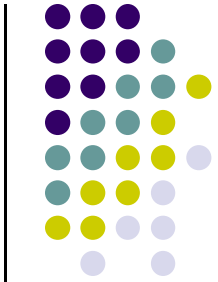
Basic idea:



Introduction to Shadows

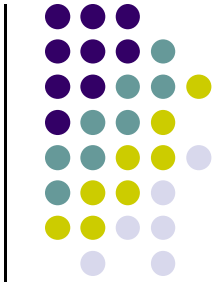


Introduction to Shadows



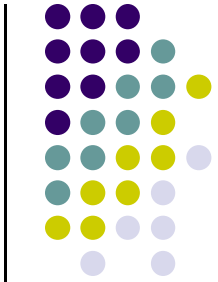
- Shadows make image more realistic
  - Important visual cues on relative positions of objects in scene
- Rendering shadows:
  - Points in shadow: use only ambient component
  - Points NOT in shadow: use all lighting components
  - Simple illumination models == simple shadows
- Two methods:
  - Shadows as texture (projection)
  - Shadow buffer
- Third method used in ray-tracing

Projective Shadows

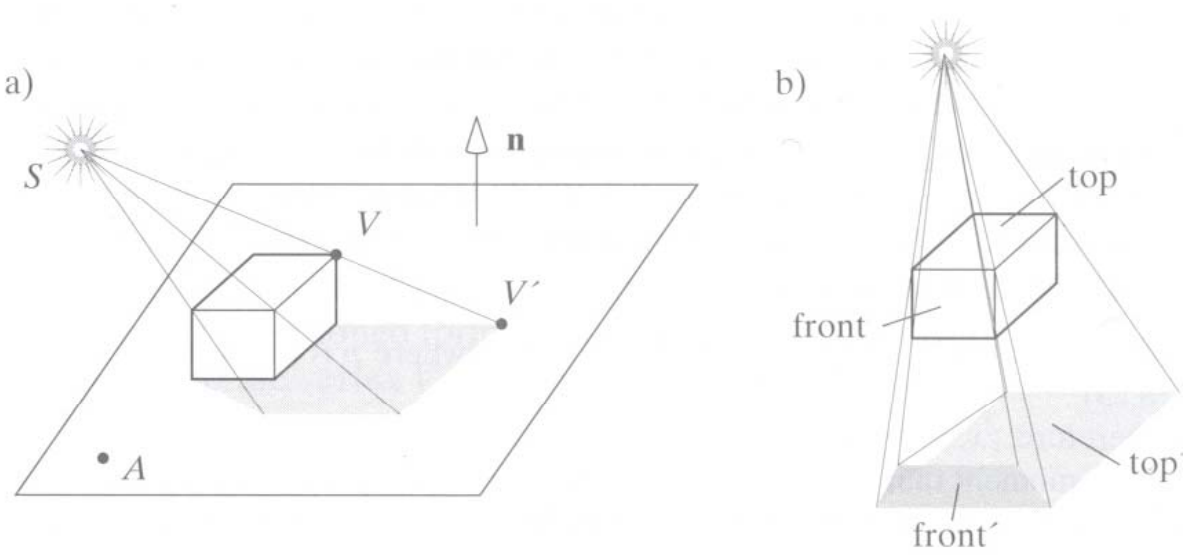


- Oldest methods
  - Used in flight simulators to provide visual clues
- Projection of a polygon is a polygon called a **shadow polygon**
- Given a point light source and a polygon, the vertices of the shadow polygon are the projections of the original polygon's vertices from a point source onto a surface

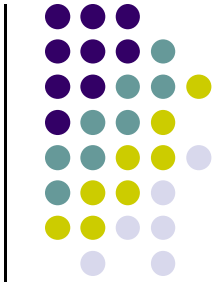
Projective Shadows



- Paint shadows as a texture
- Works for flat surfaces illuminated by point light source
- Problem: compute shape of shadow

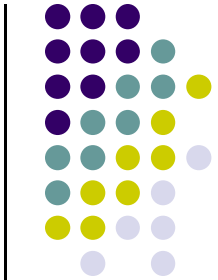


Projective Shadows

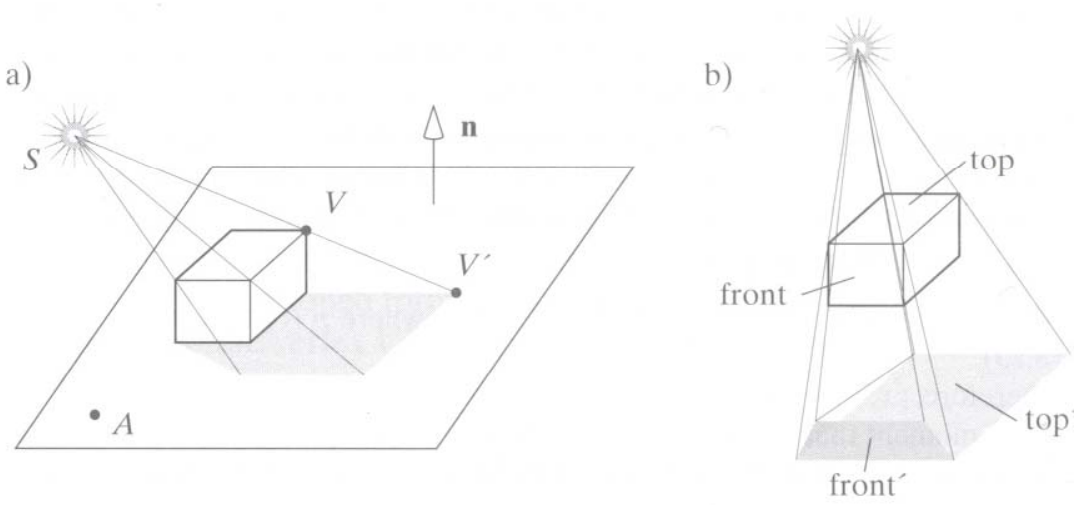


- Project light-object edges onto plane
- Want shadow of entire object
- Theory: union of projections of individual faces = projection of entire object
- Algorithm:
  - First, draw plane using specular-diffuse-ambient components
  - Then, draw shadow projections (face by face) using only ambient component

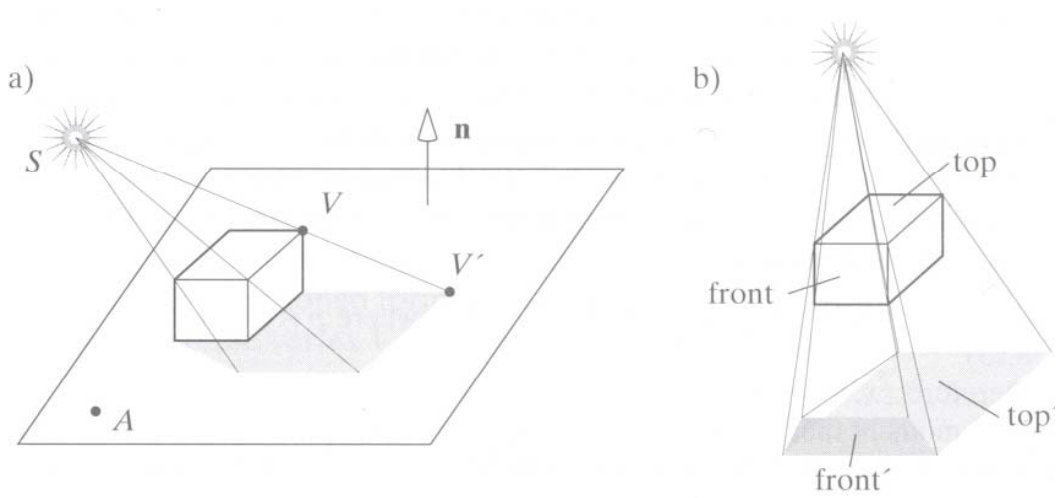
Projective Shadows



- **Problem:** find outline of shadow by calculating projections of object vertices onto plane
- Example: want to project vertex *V* to find *V'*
- Plane passes through point A and has normal, **n**



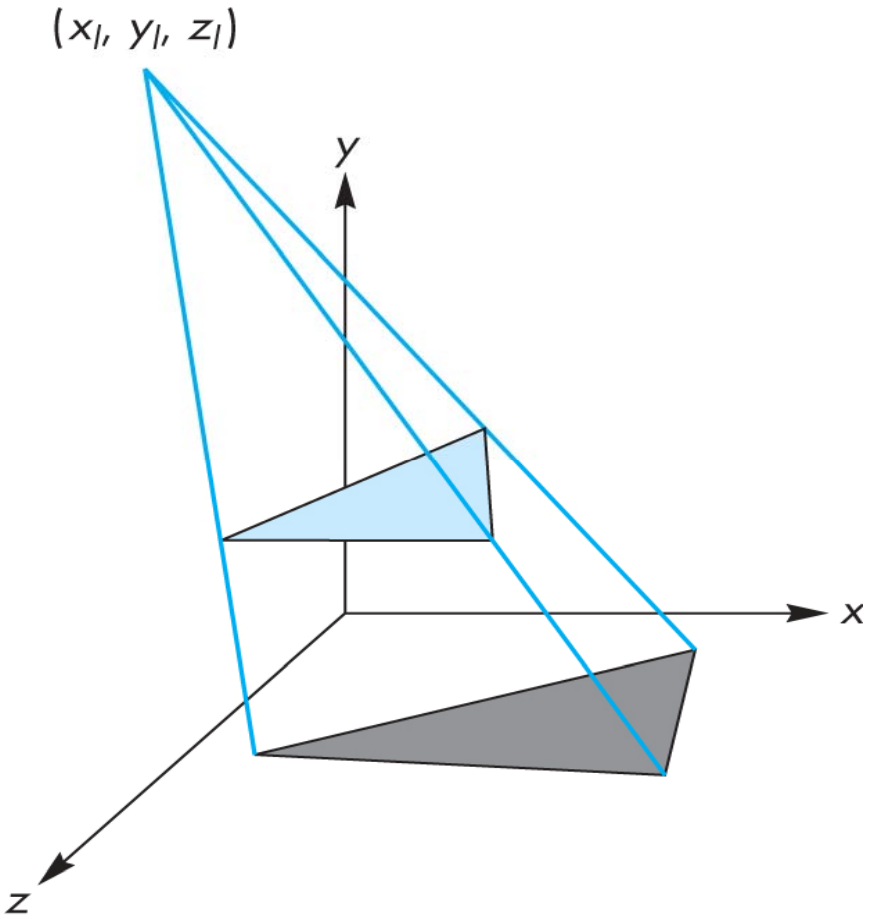
Projective Shadows



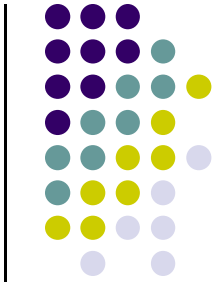
$$V' = S + (V - S)\frac{\mathbf{n} \cdot (A - S)}{\mathbf{n} \cdot (V - S)}$$

Note: can express projection in homogeneous coordinates and use matrices

Projective Shadows for Polygon



Computing Shadow Vertex

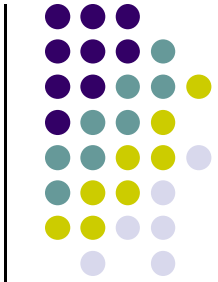


1. Source at (x_l, y_l, z_l)
2. Vertex at (x, y, z)
3. Consider simple case of shadow projected onto ground at $(x_p, 0, z_p)$
4. Translate source to origin with $T(-x_l, -y_l, -z_l)$
5. Perspective projection

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

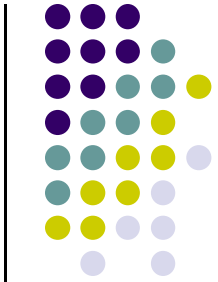
6. Translate back

Shadow Buffer Approach



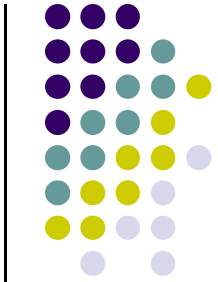
- Uses second depth buffer called shadow buffer
- Pros: not limited to plane surfaces
- Cons: needs lots of memory
- Theory:
  - Establish object-light path
  - Other objects in object-light path = object in shadow
  - Otherwise, not in shadow

Shadow Buffer Approach

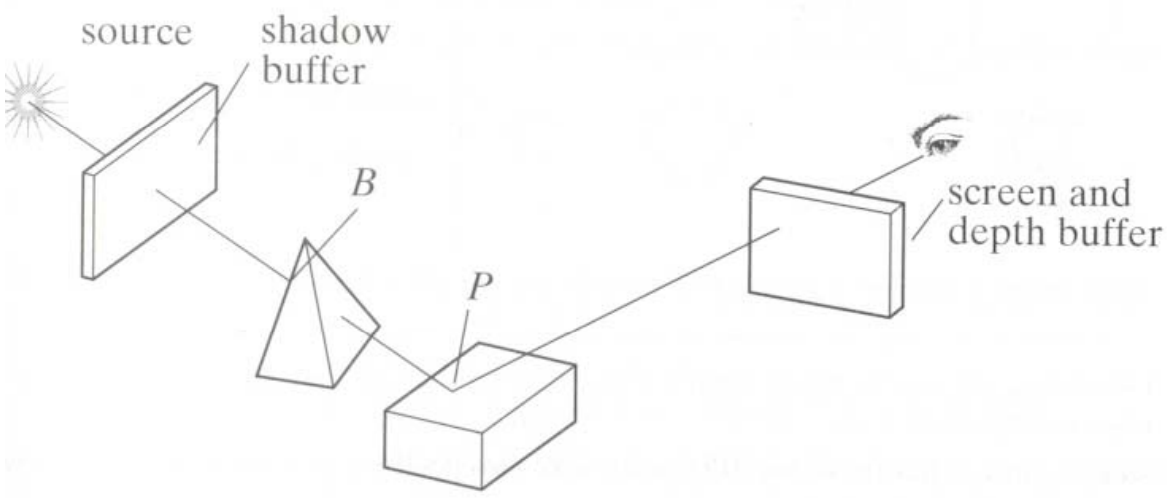


- Shadow buffer records object distances from light source
- Shadow buffer element = distance of closest object in a direction
- Rendering in two stages:
  - Loading shadow buffer
  - Rendering the scene

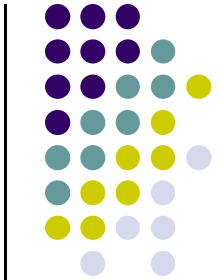
Loading Shadow Buffer



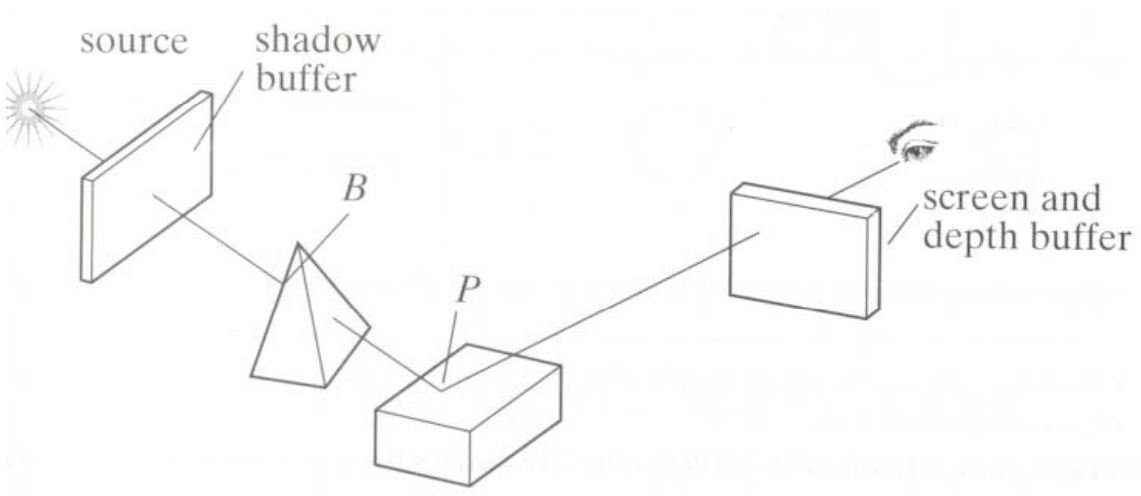
- Initialize each element to 1.0
- Position a camera at light source
- Rasterize each face in scene updating pseudo-depth
- Shadow buffer tracks smallest pseudo-depth so far



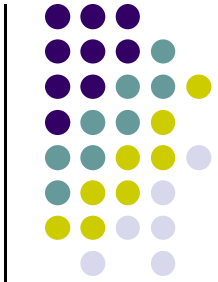
Loading Shadow Buffer



- 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

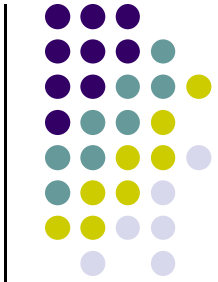


Shadow Buffer (Rendering Scene)

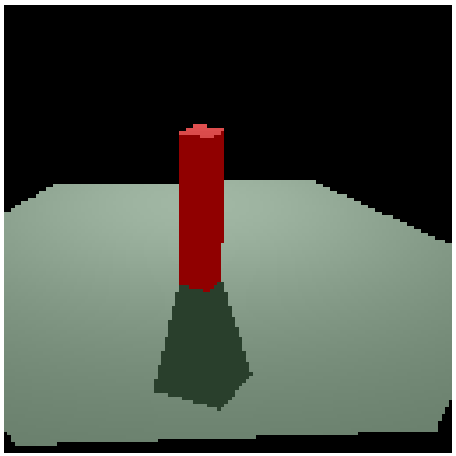


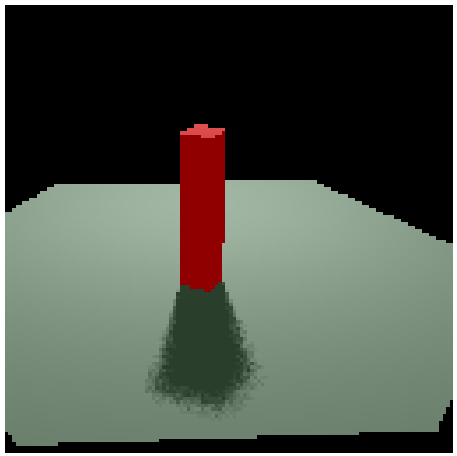
- Render scene using camera as usual
- While rendering a pixel find:
  - pseudo-depth D from light source to P
  - Index location [i][j] in shadow buffer, to be tested
  - Value d[i][j] stored in shadow buffer
- If d[i][j] < D (other object on this path closer to light)
  - point P is in shadow
  - set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

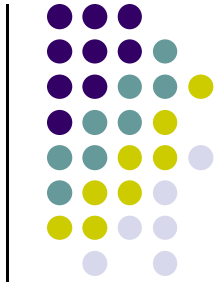
Other Issues



- Point light sources => simple but a little unrealistic
- Extended light sources => more realistic
- Shadow has two parts:
  - Umbra (Inner part) => no light
  - Penumbra (outer part) => some light







References

- Angel and Shreiner
- Hill and Kelley