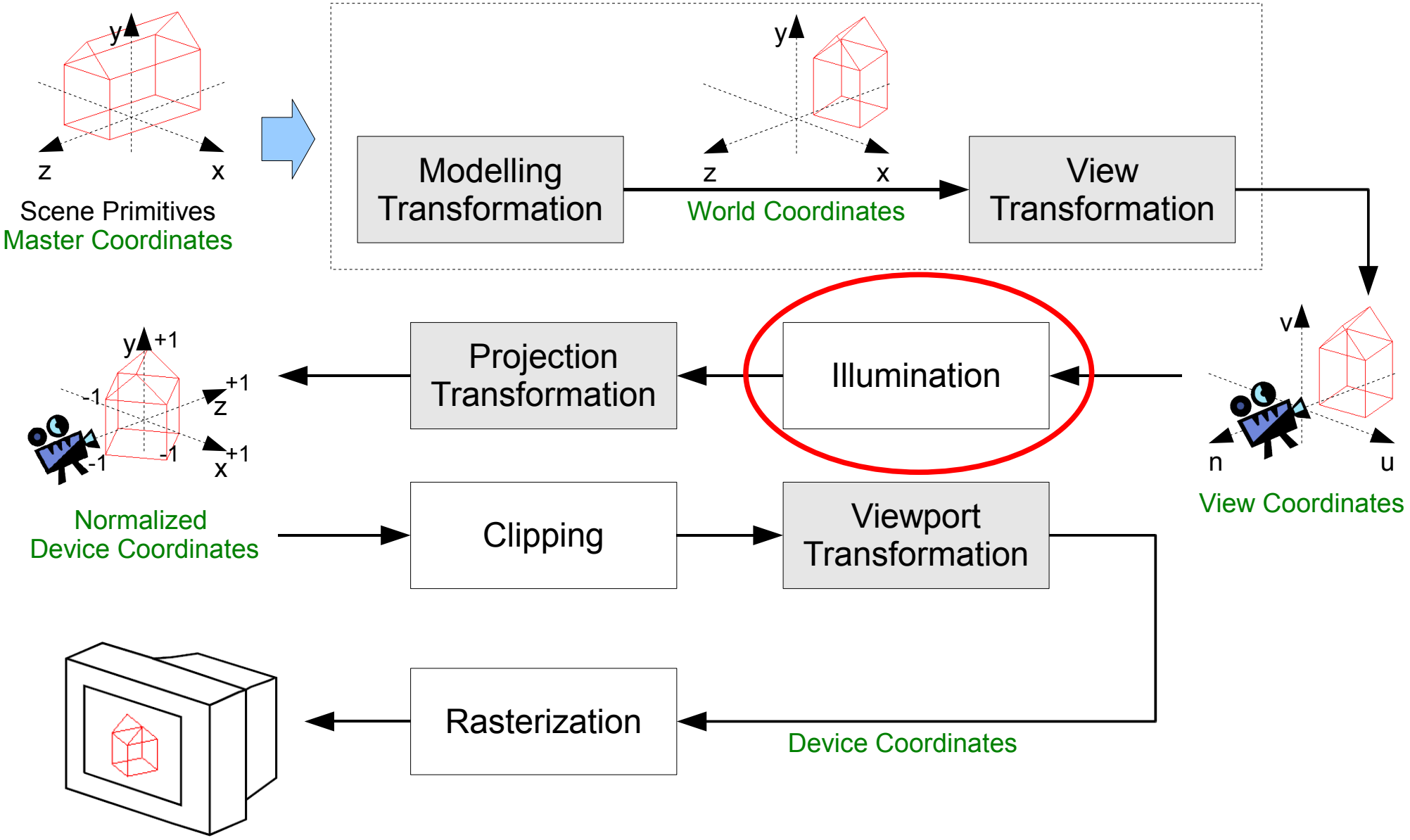
CompSci 372 – Tutorial

Part 8

Illumination

OpenGL Render Pipeline

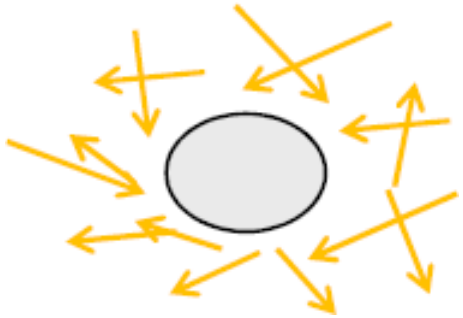


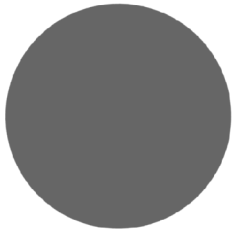
Illumination

- Sources
  - Point sources (e.g. lamp, headlight, spotlight)
  - Directional sources (e.g. sun)
- Events
  - Reflection (e.g. mirror)
  - Absorption (e.g. black cloth)
  - Transmission (e.g. glass, water)

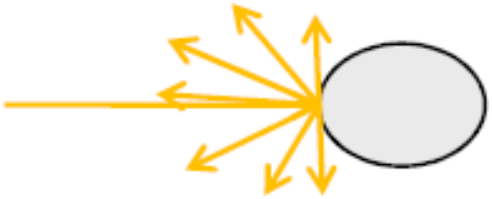
Reflection

- Ambient

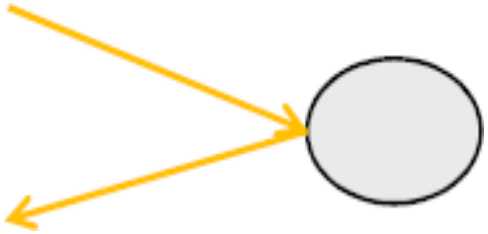


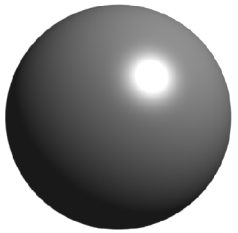


- Diffuse



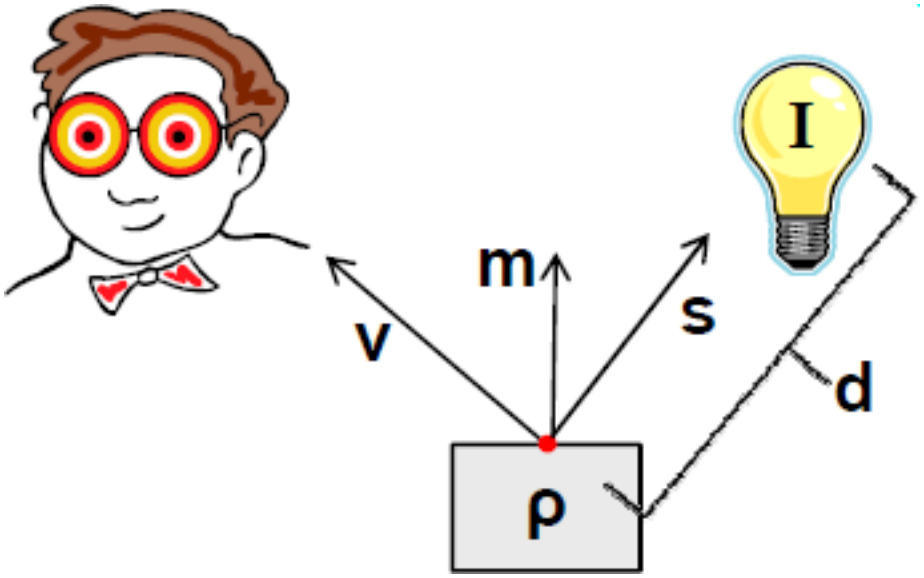
- Specular





Phong Illumination Model

- Parameters
  - Incident light $I_{a,d,s}$
  - Light distance d
  - Light direction $\boldsymbol{s}$
  - Surface normal $\boldsymbol{m}$
  - Viewer direction $\boldsymbol{v}$
  - Surface parameters ρ
    - $\rho_{a,c}$: Ambient, $\rho_{d,c}$: Diffuse, $\rho_{s,c}$: Specular
      $c \in$ [Red, Green, Blue]

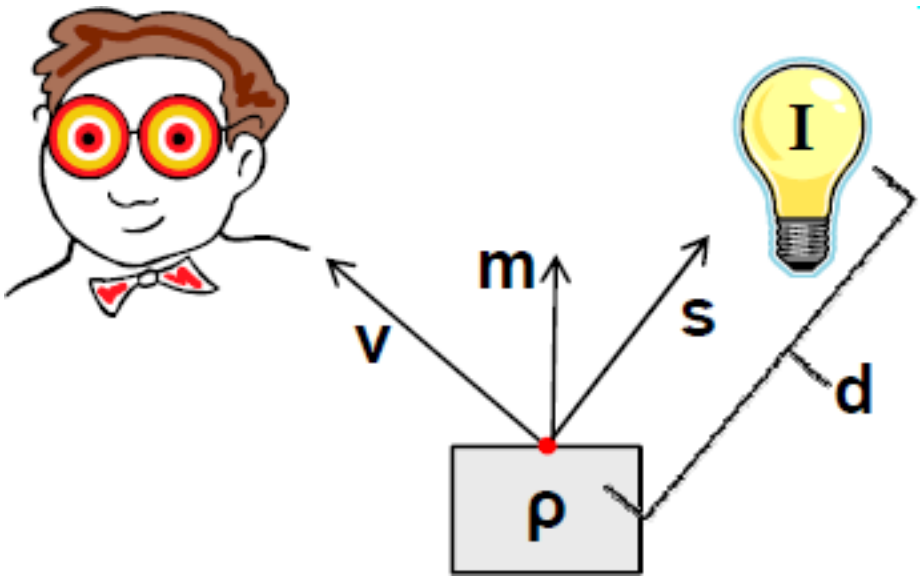


Phong Illumination Model

$$R_{r,c} = f\left(I_{r,c}, \rho_{r,c}, d, \boldsymbol{s}, \boldsymbol{m}, \boldsymbol{v}\right)$$

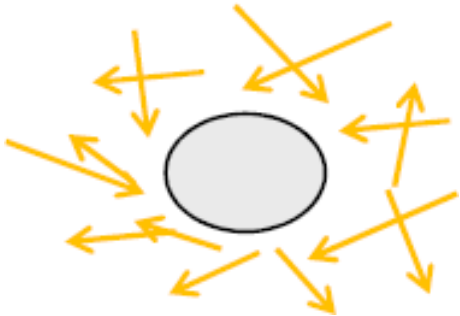
$$r \in [\, Ambient, Diffuse, Specular \,]$$

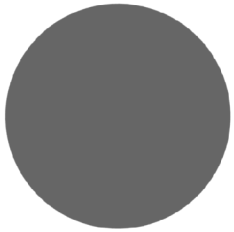
$$c \in [\, Red, Green, Blue \,]$$



Ambient Light

- d: no influence
- s: no influence
- m: no influence
- v: no influence

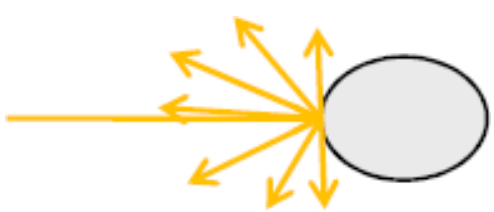




- $R_{a,c} = I_{a,c}\rho_{a,c}$

Diffuse Light

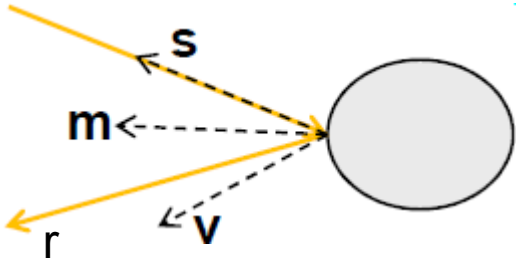
- d: Intensity "Hack"
- $\boldsymbol{s}$: Lambert's Law
- $\boldsymbol{m}$: Lambert's Law
- $\boldsymbol{v}$: no influence



$$R_{d,c} = I_{d,c}\,\rho_{d,c}\,\frac{\boldsymbol{s}\cdot\boldsymbol{m}}{|\boldsymbol{s}||\boldsymbol{m}|}\,\frac{1}{k_c + k_l d + k_q d^2}$$

Specular Light

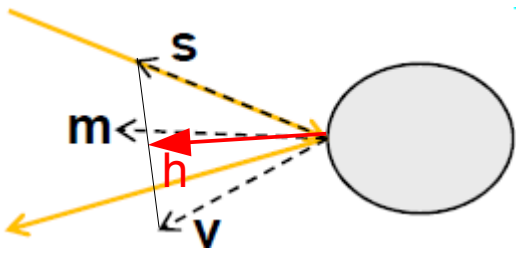
- d: Intensity "Hack"
- $\boldsymbol{s}$: Light source
- $\boldsymbol{m}$: Reflection normal
- $\boldsymbol{v}$: Highlight Intensity



$$R_{s,c} = I_{s,c}\rho_{s,c}\left(\frac{\boldsymbol{v}\cdot\boldsymbol{r}}{|\boldsymbol{v}||\boldsymbol{r}|}\right)^{\alpha}\frac{1}{k_c + k_l d + k_q d^2}$$

Specular Light

- d: Intensity "Hack"
- $\boldsymbol{s}$: Reflection source
- $\boldsymbol{m}$: Reflection normal
- $\boldsymbol{v}$: Viewpoint



Halfway-Vector $\boldsymbol{h} = \frac{\boldsymbol{s}+\boldsymbol{v}}{|\boldsymbol{s}+\boldsymbol{v}|}$

- $$R_{s,c} = I_{s,c}\rho_{s,c}\left(\frac{\boldsymbol{h}\cdot\boldsymbol{m}}{|\boldsymbol{h}||\boldsymbol{m}|}\right)^{\alpha}\frac{1}{k_c + k_l d + k_q d^2}$$

Phong Illumination Model

$$R_c = R_{a,c} + R_{d,c} + R_{s,c}$$

$$= I_{a,c}\rho_{a,c} + \left(I_{d,c}\rho_{d,c}\frac{\boldsymbol{s}\cdot\boldsymbol{m}}{|\boldsymbol{s}||\boldsymbol{m}|} + I_{s,c}\rho_{s,c}\left(\frac{\boldsymbol{h}\cdot\boldsymbol{m}}{|\boldsymbol{h}||\boldsymbol{m}|}\right)^{\alpha} \right) \frac{1}{k_c + k_l d + k_q d^2}$$

Phong Illumination Model

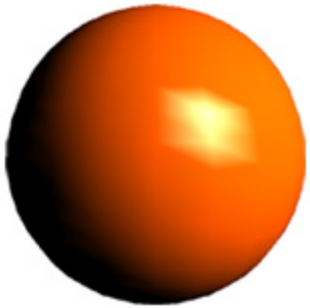
- Parameters
  - Incident light $I_{a,d,s\,/\,r,g,b}$
  - Light distance d
  - Light direction $\boldsymbol{s}$
  - Surface normal $\boldsymbol{m}$
  - Viewer direction $\boldsymbol{v}$
  - Surface parameters $\rho_{a,d,s\,/\,r,g,b}$
  - Shininess α
  - Light intensity hack k_c, k_l, k_s

Shading Algorithms

- Flat
  - per face

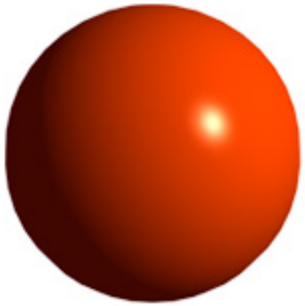
  - fast
  - Mach bands

- Gouraud
  - per vertex



  - fast
  - poor highlights

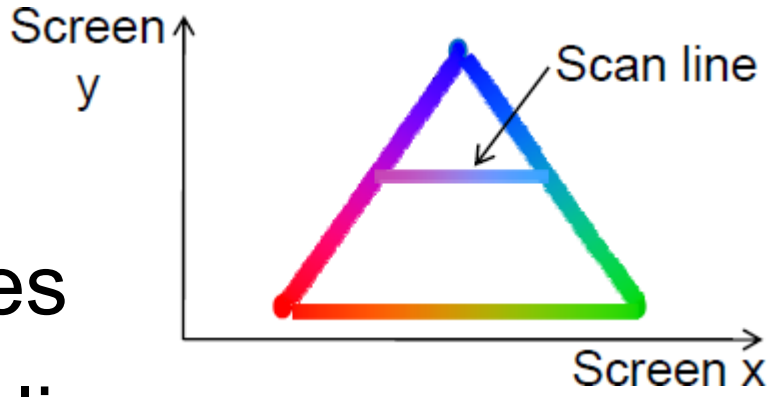
- Phong
  - per pixel

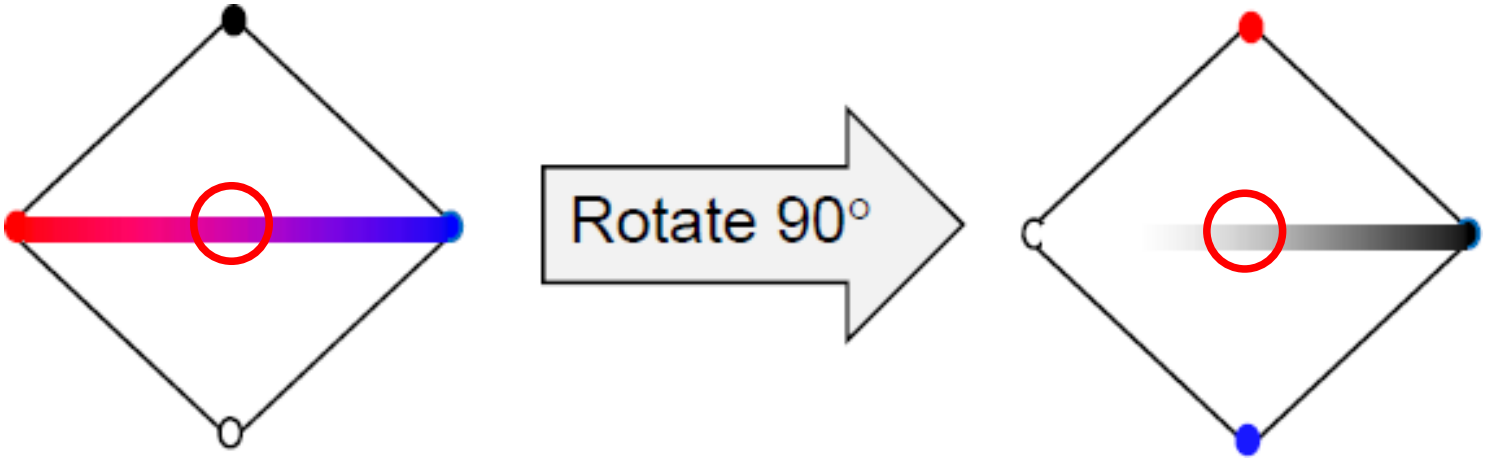


  - slow
  - good highlights

Gouraud Shading

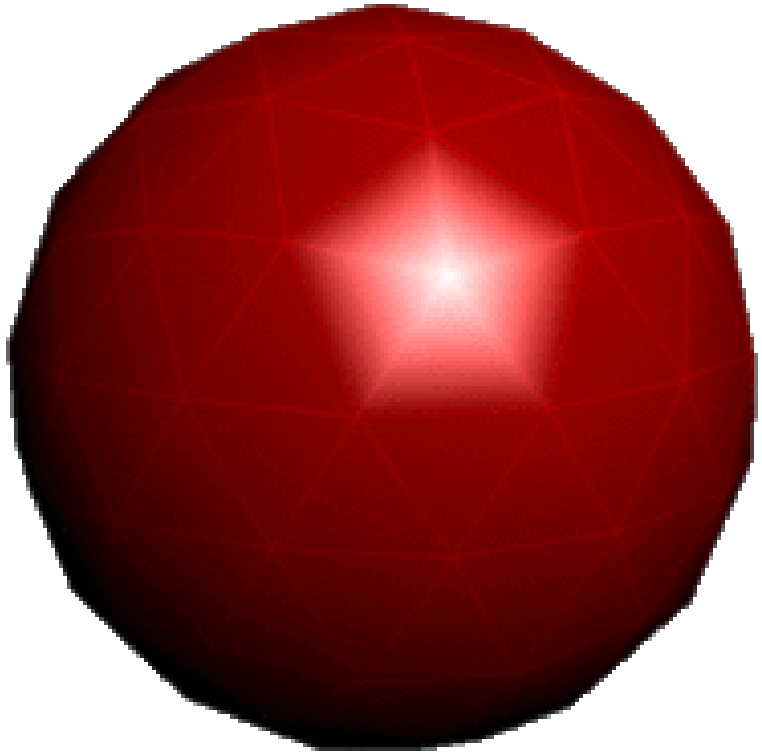
- Calculate colour for each vertex
- Interpolate colour along edges
- Interpolate colour along scanline
  - Not rotation invariant!

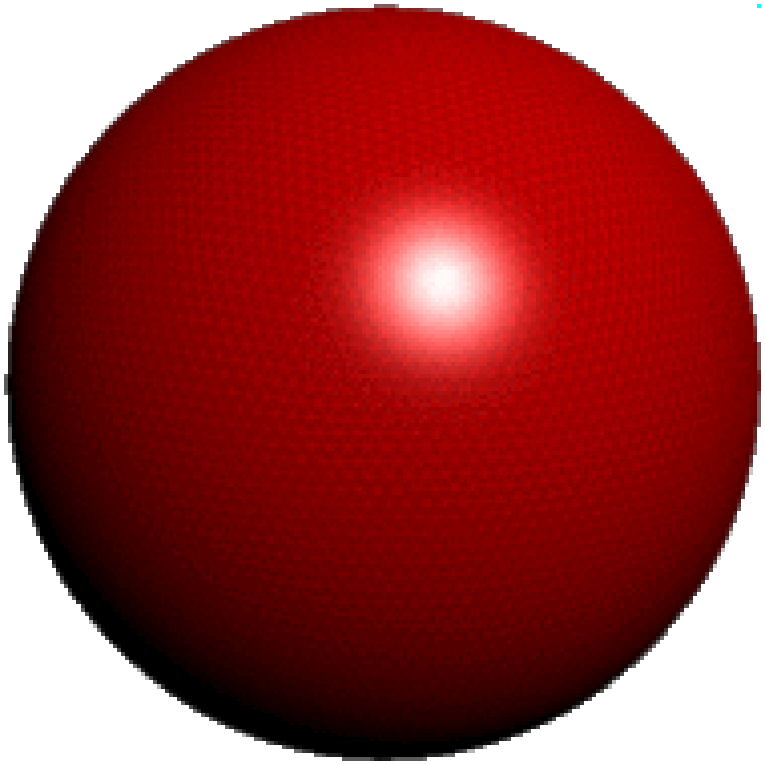




Gouraud Shading

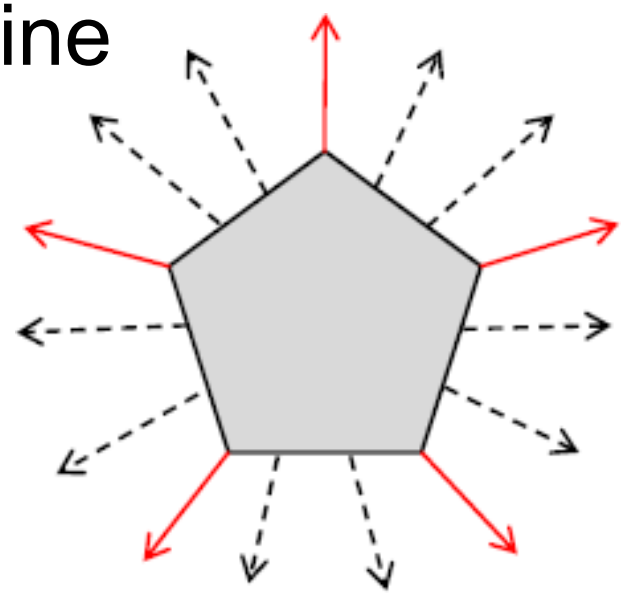
- Poor highlights
- Workaround: increase polycount (O RLY?)





Phong Shading

- Interpolate normals along scanline
  - Speed compromise
  - No high polycount necessary
  - But now per-pixel operation

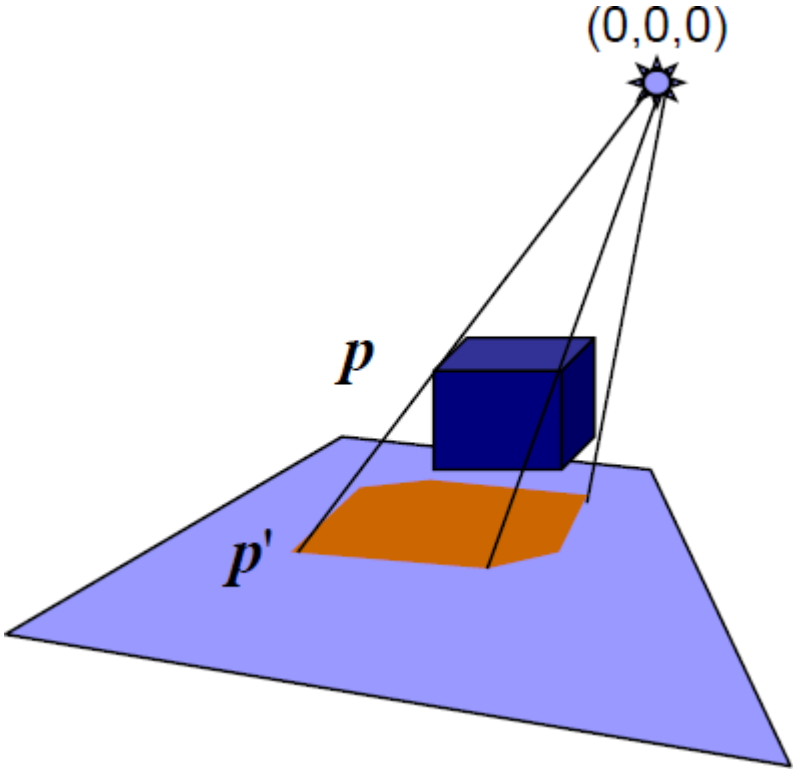


Shadows

- Shadow:
  - Visible point, not illuminated by light sources
  - Only affected by ambient light colour
- Methods:
  - Ground-plane-projection (fast, limited)
  - Shadow buffer (compromise)
  - Raytracing (slow, flexible)

Ground-Plane Projection

1. Draw the projection plane
2. Turn off depth testing
3. Turn off lighting, set shadow colour
4. Push MODELVIEW matrix
5. Shift origin to light source position
6. Set plane projection transformation
7. Draw all shadow-casting objects
8. Pop MODELVIEW matrix
9. Turn on lighting and depth testing
10. Draw all other scene objects



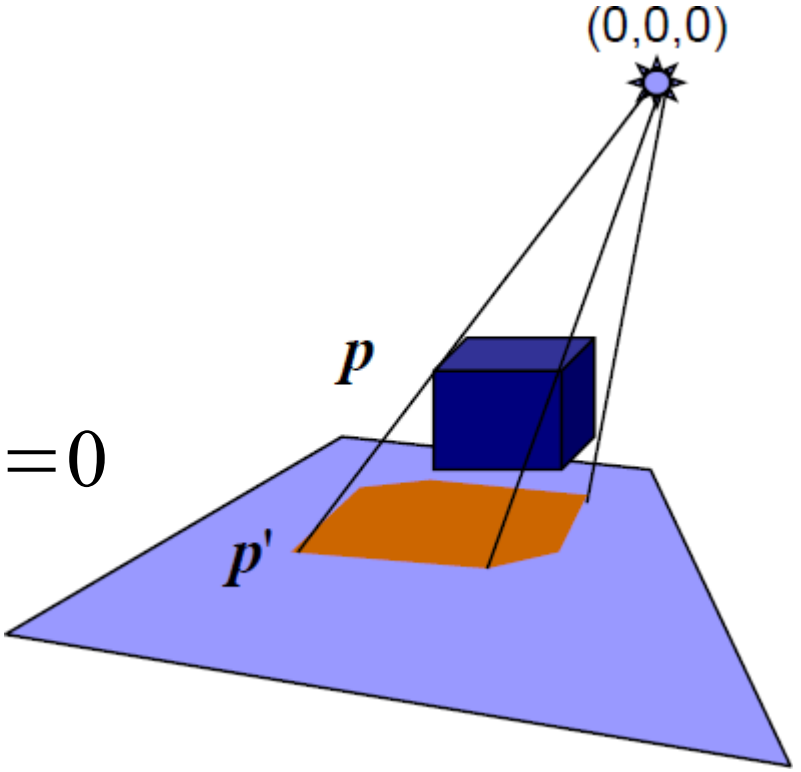
Ground-Plane Projection

- Light source at origin
- Light ray: $\boldsymbol{q}(t) = \boldsymbol{p}\,t$
- Plane: $ax + by + cz + d = 0$
- Shadow: $a\,p'_x + b\,p'_y + c\,p'_z + d = 0$

  $a\,t\,p_x + b\,t\,p_y + c\,t\,p_z + d = 0$
- Solve for t:

  $(a\,p_x + b\,p_y + c\,p_z)t + d = 0$

  $$t = \frac{-d}{a\,p_x + b\,p_y + c\,p_z}$$



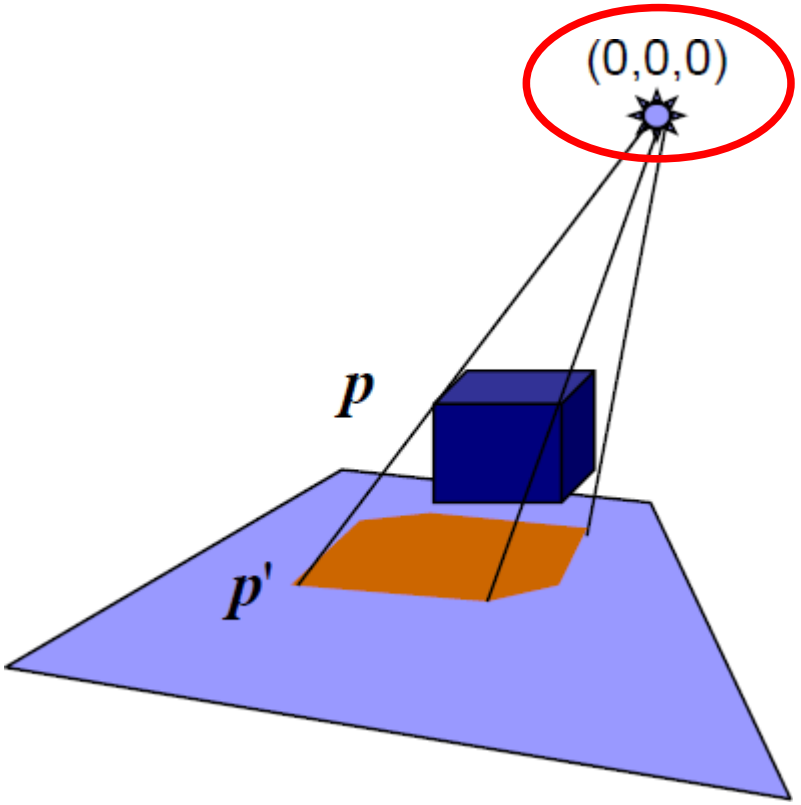
Ground-Plane Projection

- Insert t into $\boldsymbol{q}(t)$:

$$\boldsymbol{p}'\,t = \frac{-d}{a\,p_x + b\,p_y + c\,p_z}\,\boldsymbol{p}$$

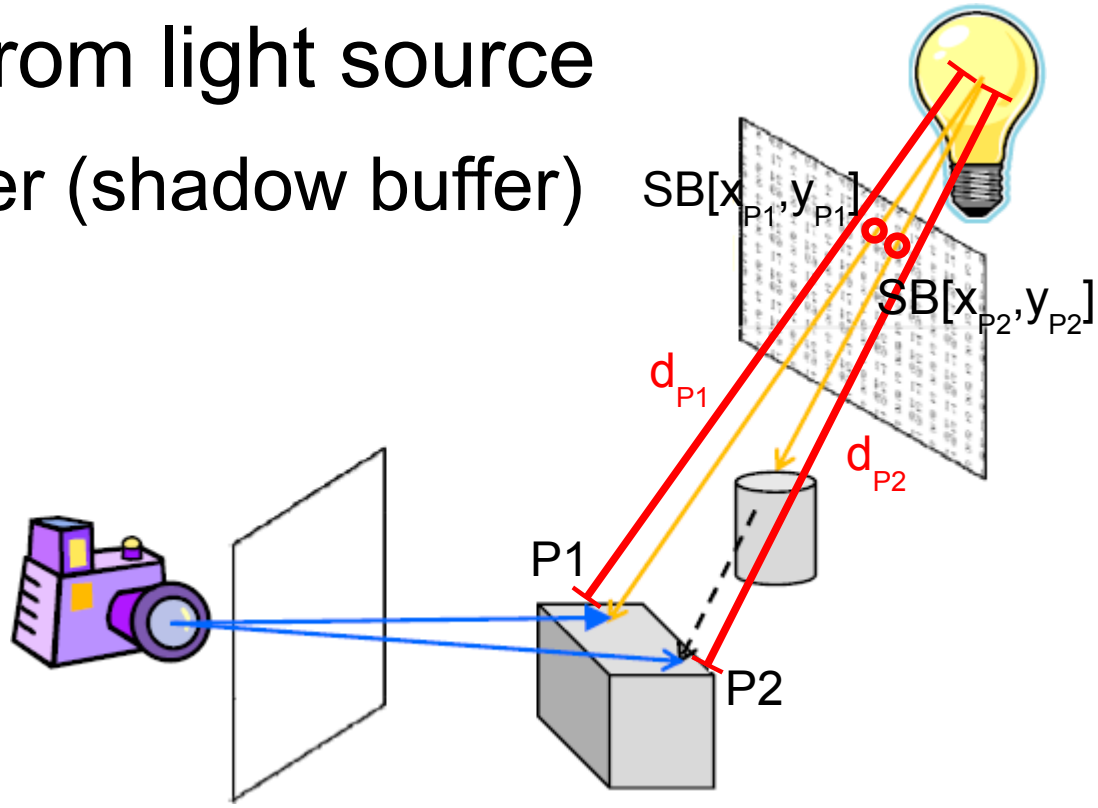
$$= \frac{-d\,\boldsymbol{p}}{a\,p_x + b\,p_y + c\,p_z}$$

$$\boldsymbol{p}' = \begin{pmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{pmatrix} \boldsymbol{p}$$



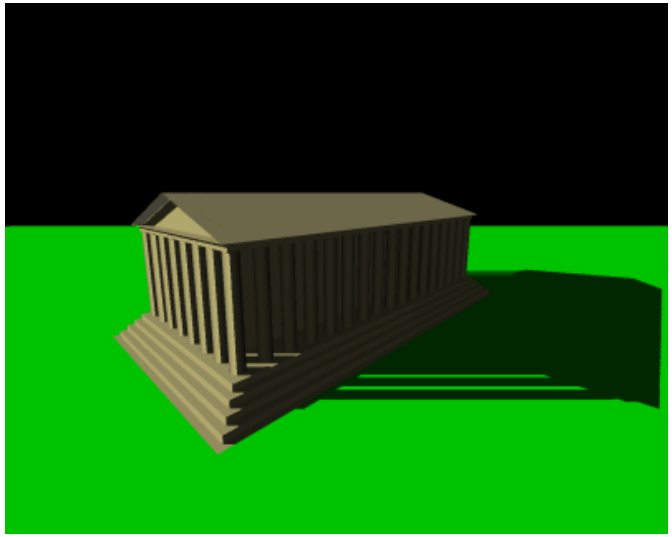
Shadow Buffer

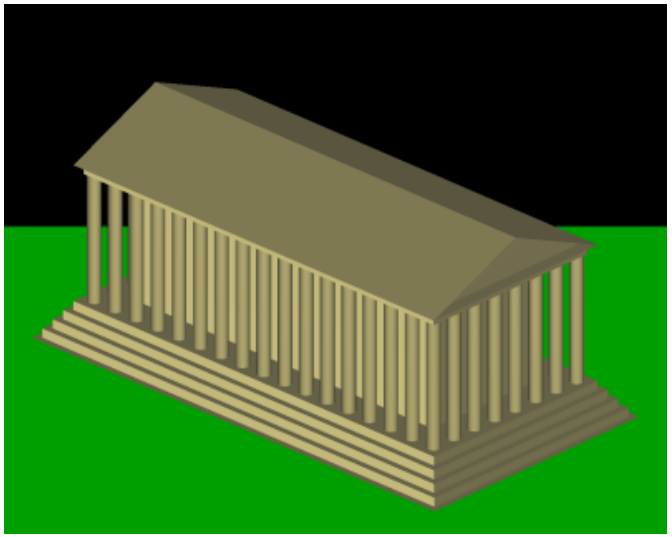
- Render scene seen from light source
  - Keep only depth buffer (shadow buffer)
- Illuminate scene
  - Get pseudodepth d_P
  - Get shadow buffer coordinate x_P, y_P
  - d_P > SB[x_P, y_P]
    - Yes: Shadow: Apply only ambient illumination
    - No: Light: Apply full illumination

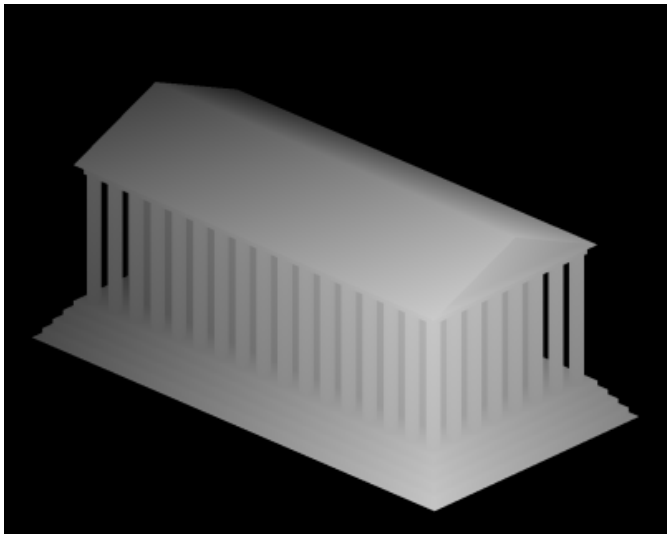


Shadow Buffer

- Example







Shadow Buffer

- Problem: Size