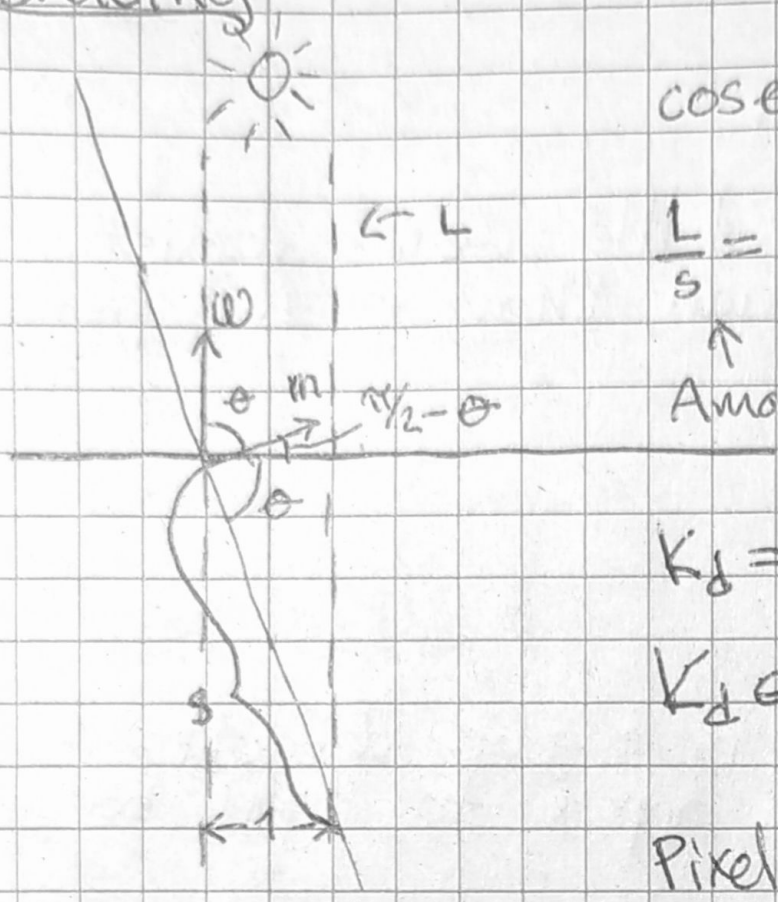
8 - Lights & shading

Shading



$\cos\theta = \frac{1}{s}$

Light gets darker as we tilt because it's spread over wider area

$\frac{L}{s} = L\underbrace{\cos\theta}_{\text{Geometry term}}$

↑ Amount of light hitting s

K_d = diffuse coefficient = material property

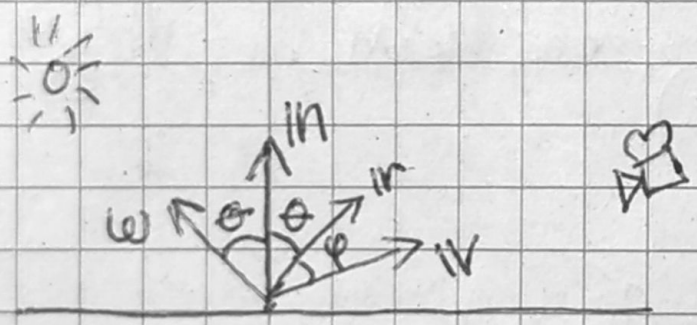
$K_d \cos\theta = K_d(\underset{\uparrow}{n} \cdot \underset{\uparrow}{\omega})$

if unit vectors

Pixel color = $C = I \cos\theta K_d$, I = intensity

We call this Lambertian (diffuse) material

Specular reflections (Phong)



K_s = specular reflection color

α = specular shininess

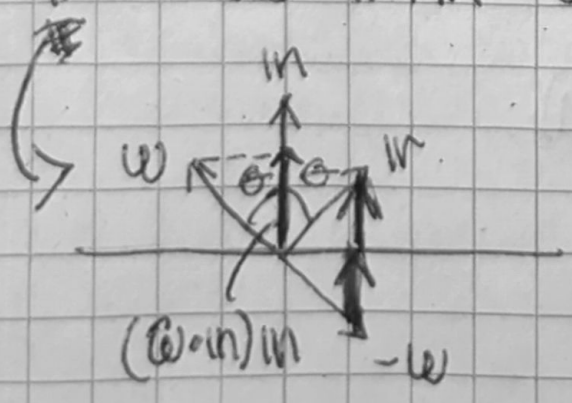
$I K_s (\cos\varphi)^{\alpha} = I K_s (v \cdot r)^{\alpha}$

Phong material model $\Rightarrow C = I(\max(0, \cos\theta) K_d + K_s(\max(0, \cos\varphi)^{\alpha})$

↑ Because we don't want negative

= diffuse + specular

$r = 2(\omega \cdot n)n - \omega$

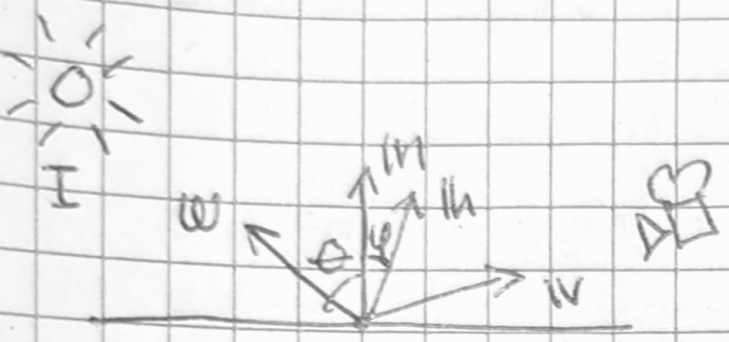


Blinn material model

$$C = I(\cos\theta K_d + K_s(\cos\varphi)^\alpha)$$

$$lh = \frac{\omega + v}{|\omega + v|} \qquad \cos\varphi = n \cdot lh$$

A little more realistic than Phong so it's prefered.



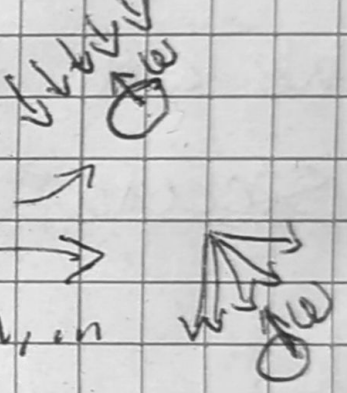
Ambient light

$C = \ldots + I_a K_a$, $K_a = K_d$ is a good default

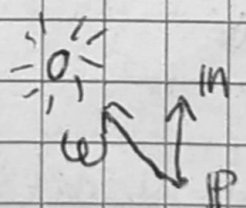
Ambient light is a good approx. to light bouncing of walls etc.

Light types

- Directional light: Intensity, direction
- Point/spot light: Intensity, position
- Area light: rectangle, disk, sphere, mesh, ...
- Sky light (environment light)



Shading transformations



Model space → (Model) → World space → (ω)View → View space

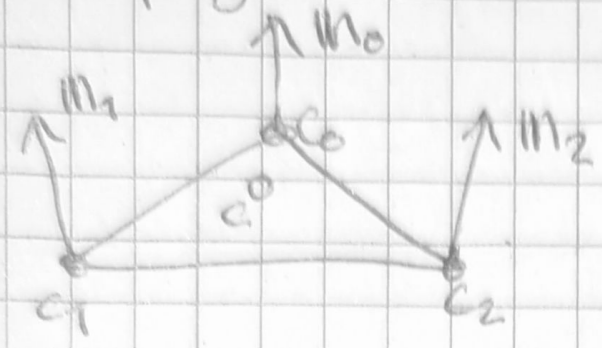
Model-view (p, n)

$p' = M p$ ← positions

$n' = (M_{3\times3}^{-1})^T n$ ← normals

Gouraud shading

Legacy OpenGL used Gouraud shading in a non-programmable pipeline.



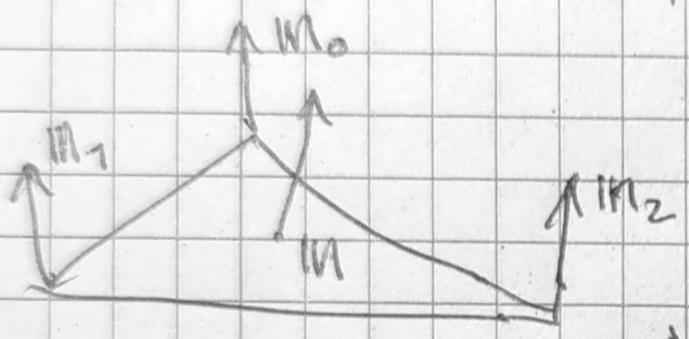
$$C = \alpha C_0 + \beta C_1 + \gamma C_2$$

Barycentric coordinates

Interpolates the color values between each vertex

Phong shading

Almost like Gouraud, but interpolates the normals.



$$n = \frac{\alpha n_0 + \beta n_1 + \gamma n_2}{|\alpha n_0 + \beta n_1 + \gamma n_2|}$$

Needs to be normalized

Overview (for the project)

- On the CPU
  - Compute matrices
    - Model-view
    - Model-view-projection
    - Model-view for normals
- Vertex shader
  - Transform position with model-view
  - Transform position with model-view-projection
  - Transform normal with model-view for normal
- Fragment shader
  - Compute lightning and shading