CS 428: Fall 2009

Introduction to Computer Graphics

Polygonal rendering: illumination

Polygon shading

- Non-global illumination
  - No shadows, refraction, inter-object reflection…

- Describing light
  - Units – don't worry for now, just use ratio

$$\frac{\text{light exiting surface towards viewer}}{\text{light incident on surface from lights}}$$

Polygon shading

- Describing light
  - Units – don't worry for now, just use ratio

$$\frac{\text{light exiting surface towards viewer}}{\text{light incident on surface from lights}}$$

  - Depends on
    - Physical material/surface properties
    - Geometric relation between lights, surface and viewer
    - Color and intensity of lights in the scene

  - Hard to define these properties precisely

Bidirectional reflection distribution function (**BRDF**)

- Describes reflection of light
- Spectral reflection factor
- Ratio of reflected radiance L to incident irradiance E

$$\rho(\lambda,\phi_r,\theta_r,\phi_i,\theta_i)=\frac{L_{\lambda,r}(\lambda,\phi_r,\theta_r)}{E_{\lambda,i}(\lambda,\phi_i,\theta_i)}=\frac{L_{\lambda,r}(\lambda,\phi_r,\theta_r)}{\int L_{\lambda,i}(\lambda,\phi_i,\theta_i)\cos(\theta_i)d\Omega_i}$$

- Incident irradiance: Index i
- Reflected radiance: Index r

Bidirectional reflection distribution function (**BRDF**)

1. Reciprocity
   - ρ_λ does not change, when switching incident and reflected direction
2. ρ_λ is generally anisotropic
   - Rotation about the surface normal changes ρ_λ
   - Typical examples are cloth or brushed metal
3. Superposition
   - Light from various directions can be linearly added
   - Integrating over all incident directions leads to

$$L_{\lambda,r} = \int_{\Omega_i} \rho L_{\lambda,i} \cos(\theta_i) d\Omega_i$$

Bidirectional reflection distribution function (**BRDF**)

- Reflection factor is always positive
- In CG we use the reflection ratio r
  - Applied to luminance/brightness
  - Dimensionless

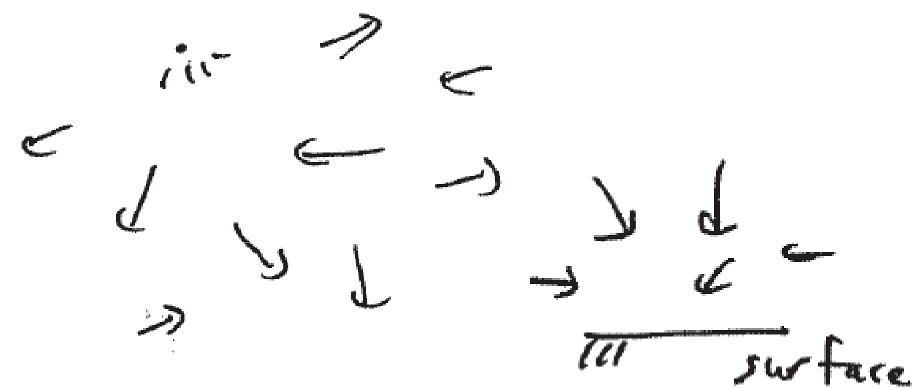
$$\frac{\text{light exiting surface towards viewer}}{\text{light incident on surface from lights}}$$

Illumination models

- Not physics-based
  - rather an approximation which is more computationally tractable
- **Ambient** reflection
- **Diffuse** reflection
- **Specular** reflection
- All use a **point** light source
  - (**x**,**y**,**z**) + Intensity (I_r, I_g, I_b)

Ambient reflection

- Light scattered in scene – uniformly

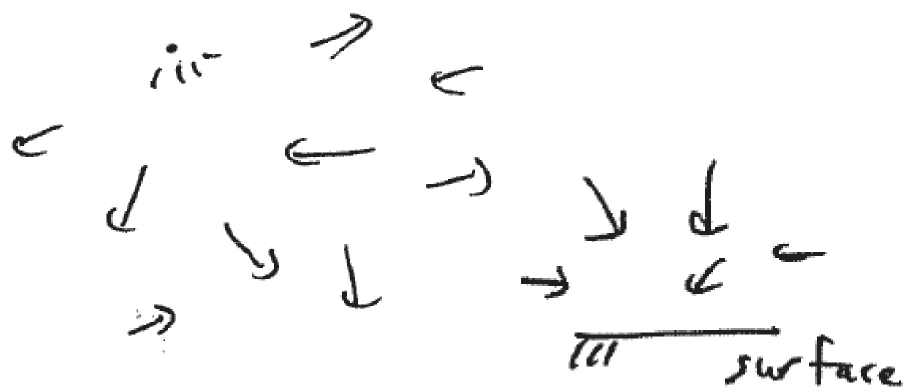


$$I_{ambient} = L_a k_a$$

- L_a: Intensity of ambient light
- k_a: fraction of ambient light reflected $\in [0,1]$

Ambient reflection

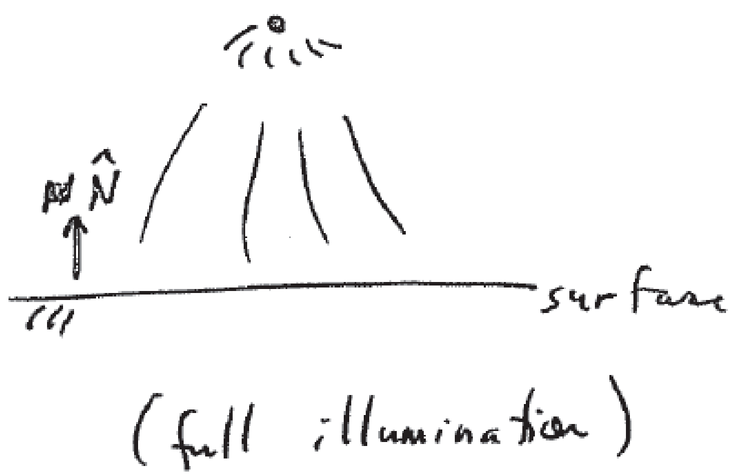
- Light scattered in scene – uniformly

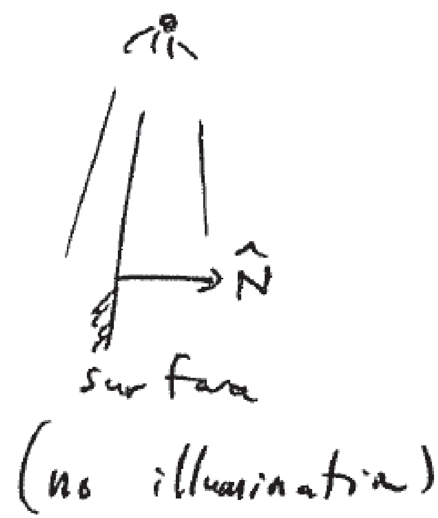


- Independent of light, viewer + surface position
- *Hack* to get some global illumination effects
- Without this term, images have too much contrast

Diffuse (Lambertian) reflection

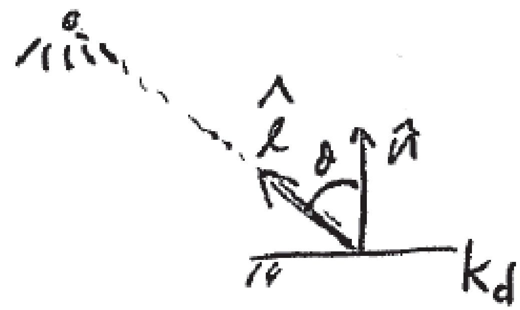
- Typical of dull, matte surfaces → rough
- Independent of viewer position
- Dependent on light position





Diffuse (Lambertian) reflection

- Lamberts cosine law



$$I_{diffuse} = L_d \, k_d \cos\theta = L_d \, k_d (\hat{n} \cdot \hat{\ell})$$

L_d: intensity of point light source

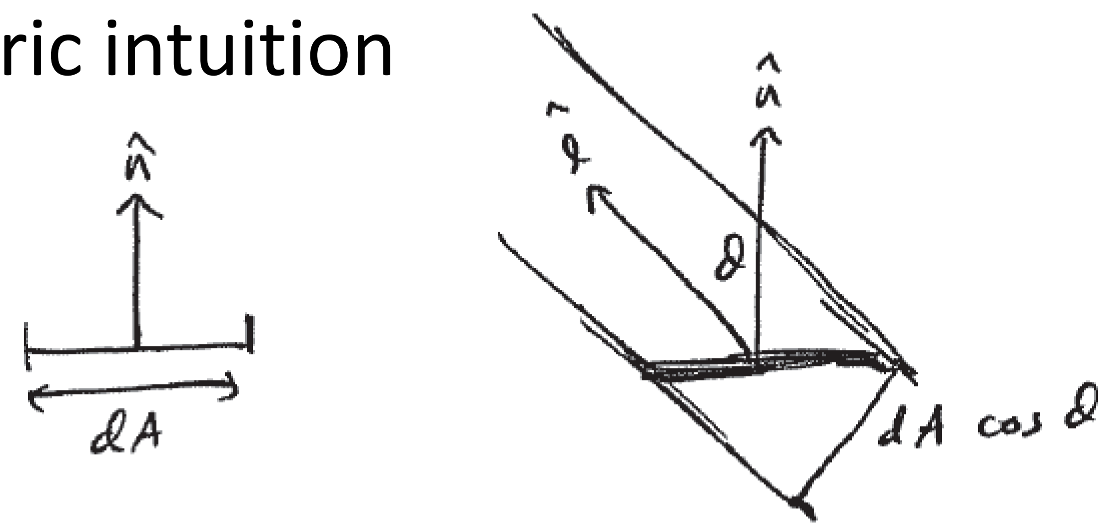
k_d: diffuse reflection coeff of surface $\in [0, 1]$

Diffuse (Lambertian) reflection

- Lamberts cosine law

$L_d = k_d \cdot \max(0, \cos\theta)$ for when surface faces the other way

- Geometric intuition



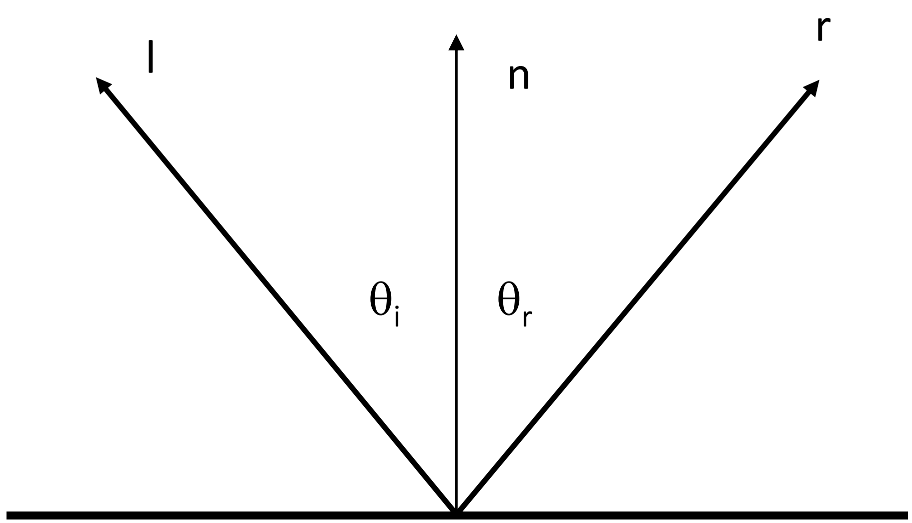
Ideal reflection

- Mirror reflection by law of reflection
  - The incident and reflected ray form the same angle with the surface normal
  - The incident and reflected ray and surface normal all lie in the same plane
  - In polar coordinates: $\theta_r = \theta_i$ and $\phi_r = \phi_i + \pi$
  - For view ray $\mathbf{l}$ and (normalized) normal $\mathbf{n}$

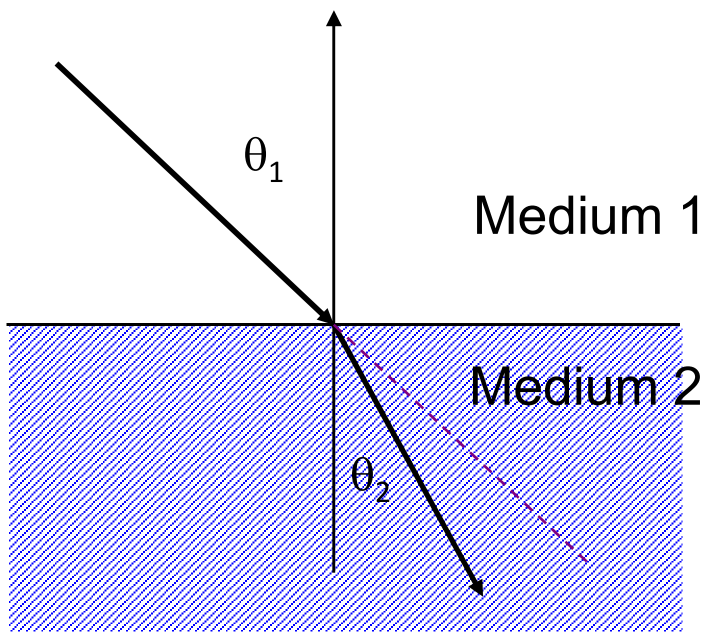
$$\mathbf{r} = -\mathbf{s} + 2\,(\mathbf{s} \cdot \mathbf{n})\,\mathbf{n}$$

Ideal reflection

Geometry of
Reflection law



Geometry of
refraction law



Ideal reflection

Law of refraction

- The incident and refracted ray and surface normal all lie in the same plane
- Sine of the incident angle has a constant ratio to the sine of the refraction angle
  - This ratio is dependent on the nature of the participating media

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \Leftrightarrow \frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1} = const.$$

  - n_1 and n_2 are the indices of refraction
    - Defined as the ratio of light speed in vacuum to light speed in the participating media

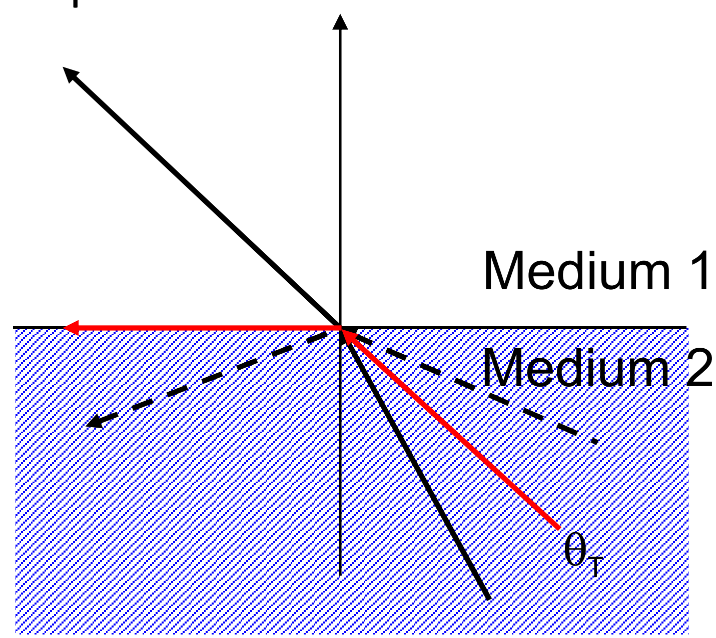
Ideal reflection

Total reflection

- Transition from optically dense to less dense material $n_2 < n_1$
  - Rays refracted away from the surface normal
  - There exists an incident angle θ_T with refraction angle of 90º

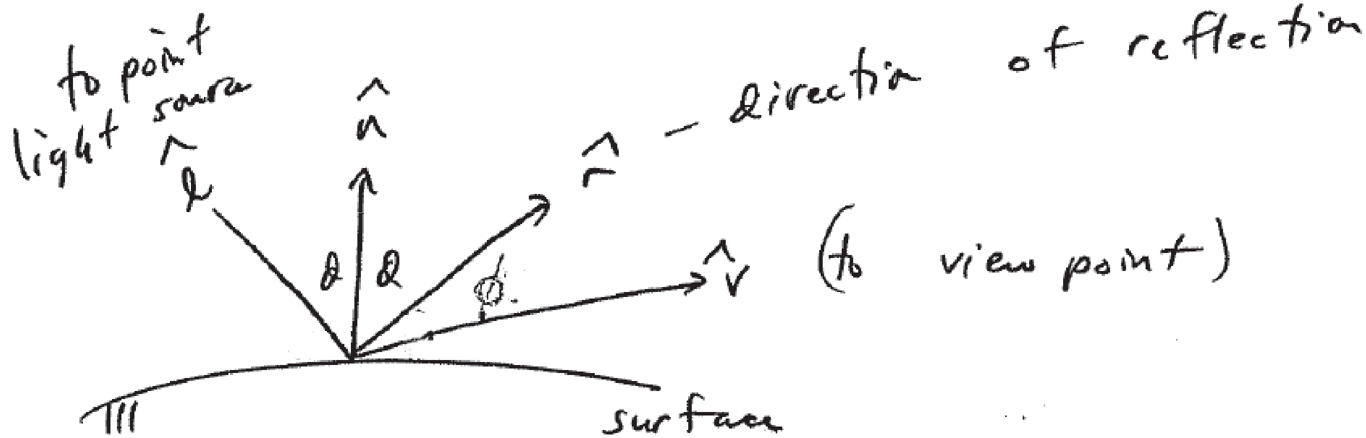
$$\sin\theta_T = \frac{n_2}{n_1}.$$

- Once θ_T is exceeded
  - All light reflected on the boundary between media
  - Total reflection



Specular reflection

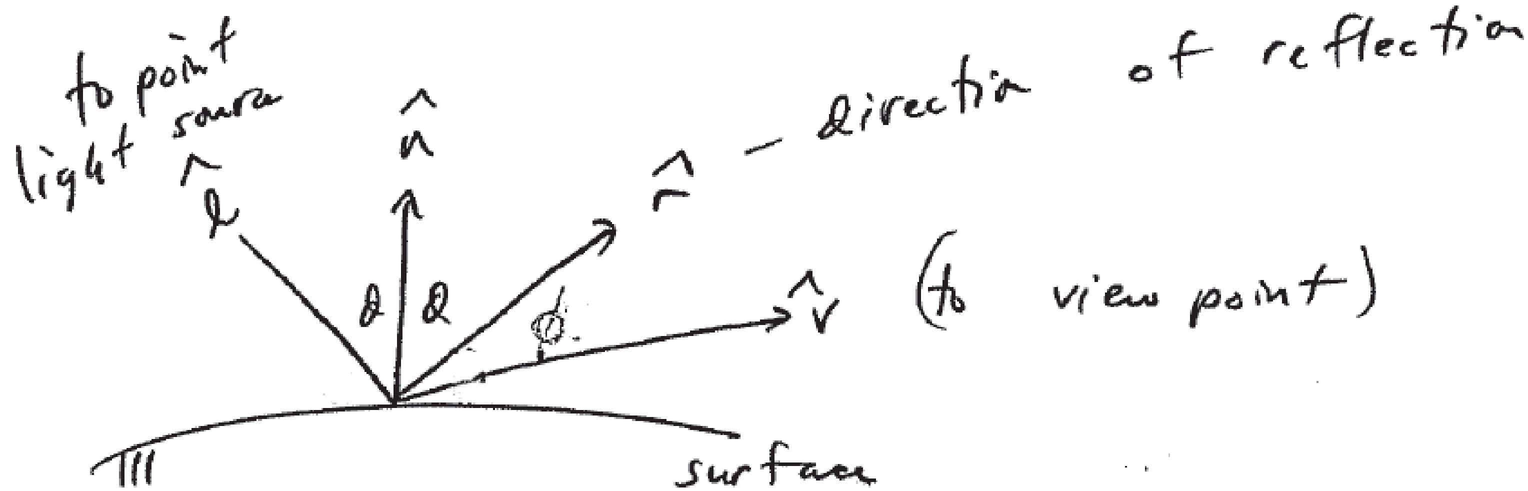
- Directed reflection from shiny surfaces



- Resulting color is a combination of surface color + light color

Specular reflection

- Directed reflection from shiny surfaces

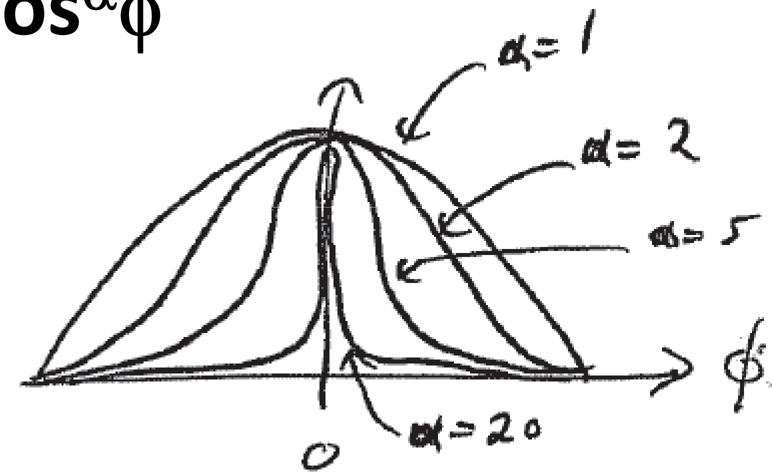


- More reflection as ϕ goes to 0

Specular reflection

Phong reflection

- More reflection as ϕ goes to 0
  - Not just $\cos\phi$ → use $\mathbf{cos}^{\alpha}\phi$
  - As α increases surface looks shinier
  - α is surface property



$$= L_s \, k_s \cos^{\alpha}\phi$$

$$I_{specular} = L_s \, k_s (\hat{r} \cdot \hat{v})^{\alpha}$$

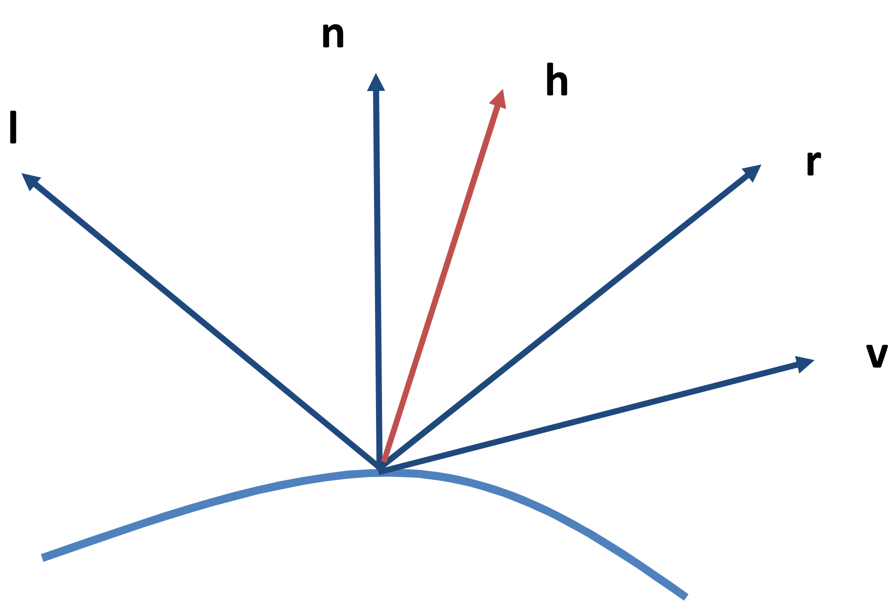
k_s: spec reflec coeff $\in [0,1)$

Specular reflection

Blinn-Phong reflection

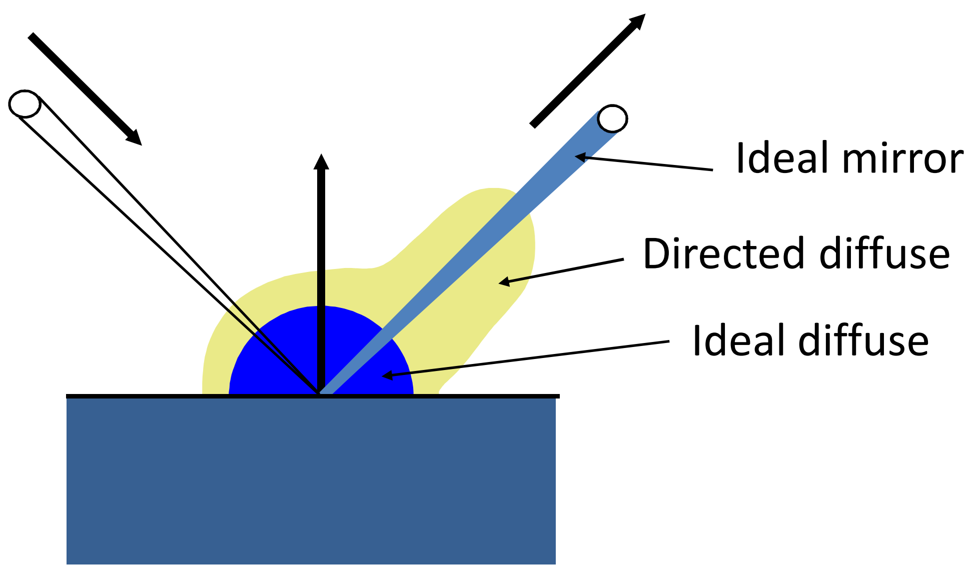
- Use halfway vector instead
  - Somwhat more efficient (less operations)
  - Used in OpenGL

$$h = \frac{l+v}{|l+v|} \rightarrow (n \cdot h) \text{ instead of } (r \cdot v)$$



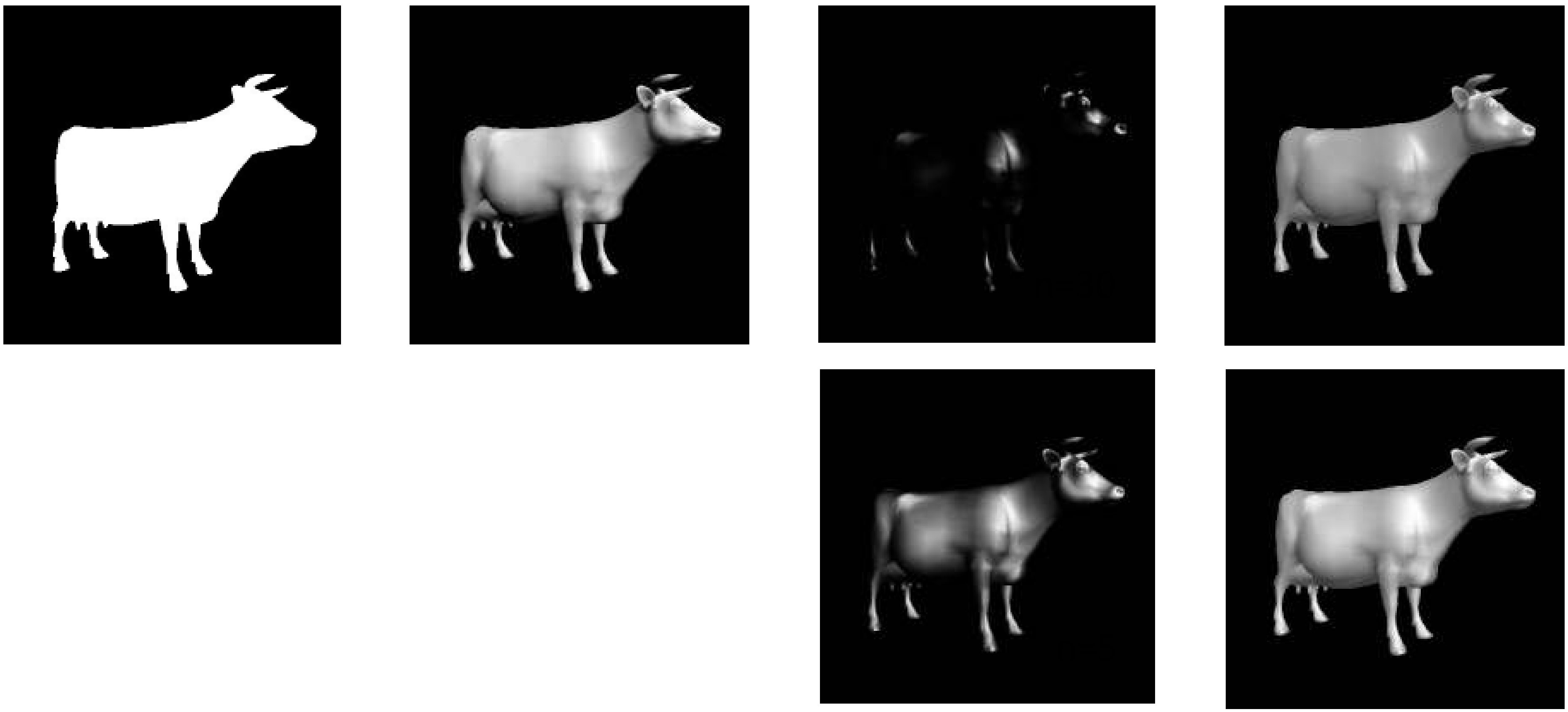
Directed diffuse reflection

- Ideal reflectors (Lambert or mirror) seldom
- Heuristic to model the real BRDF
- Combination of ambient, diffuse and specular
  - Should add to 1 (careful when selecting coeffs!)



Combination

ambient diffuse specular all



OpenGL details

- Colored lights and surfaces
- Also, light colors for each of the types of lighting, and each light source

OpenGL details

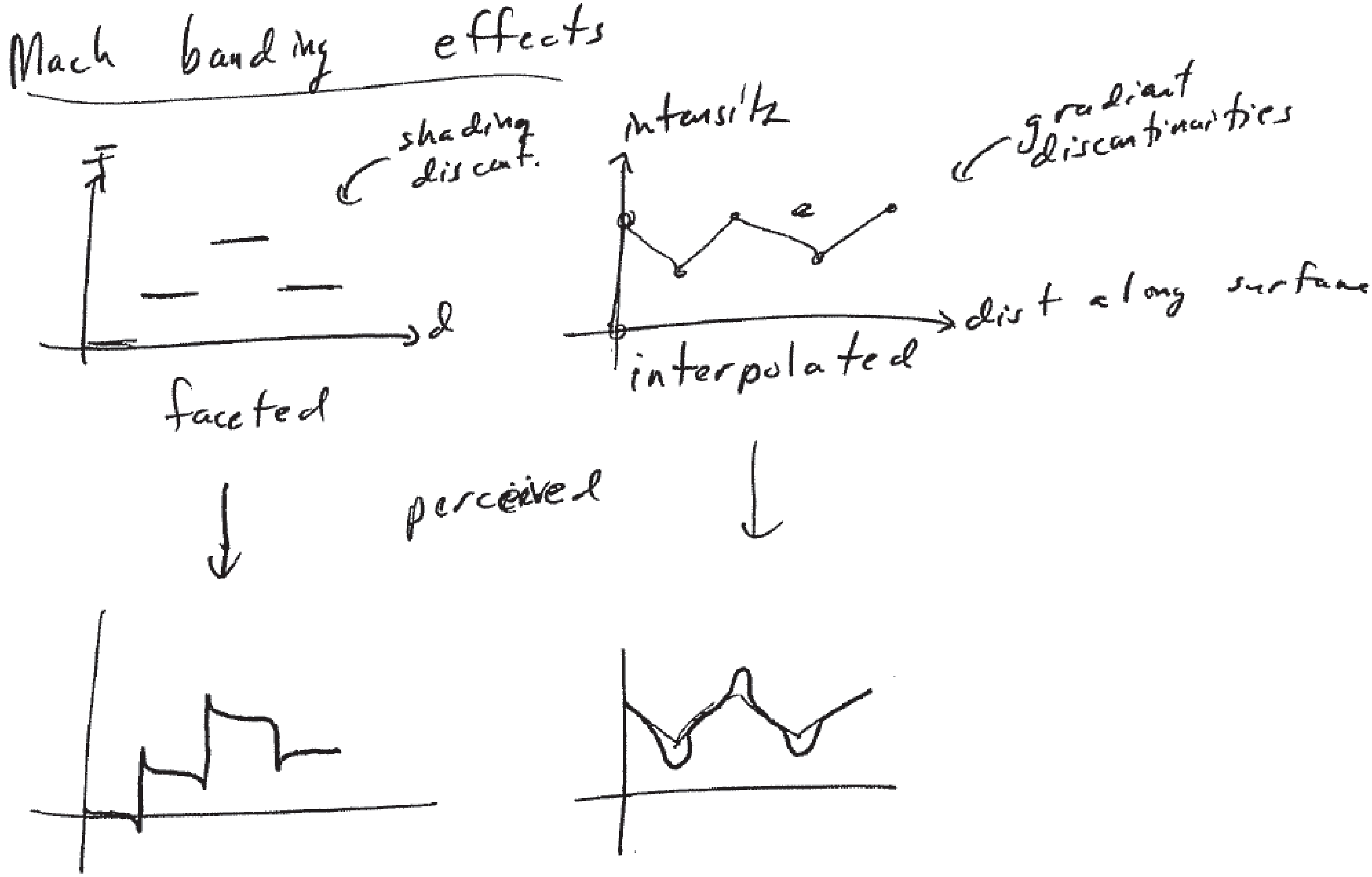
```
// light and material
float mat_ambient[] = { 0.5f, 0.5f, 0.5f, 1.0f };
float mat_specular[] = { 0.6f, 0.6f, 0.6f, 1.0f };
float mat_shininess[] = { 3.0f };
float model_ambient[] = { 0.3f, 0.3f, 0.3f };
float light_position[] = { 5.0f, 5.0f, 5.0f, 0.0f };
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, model_ambient);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

Polygon mesh shading

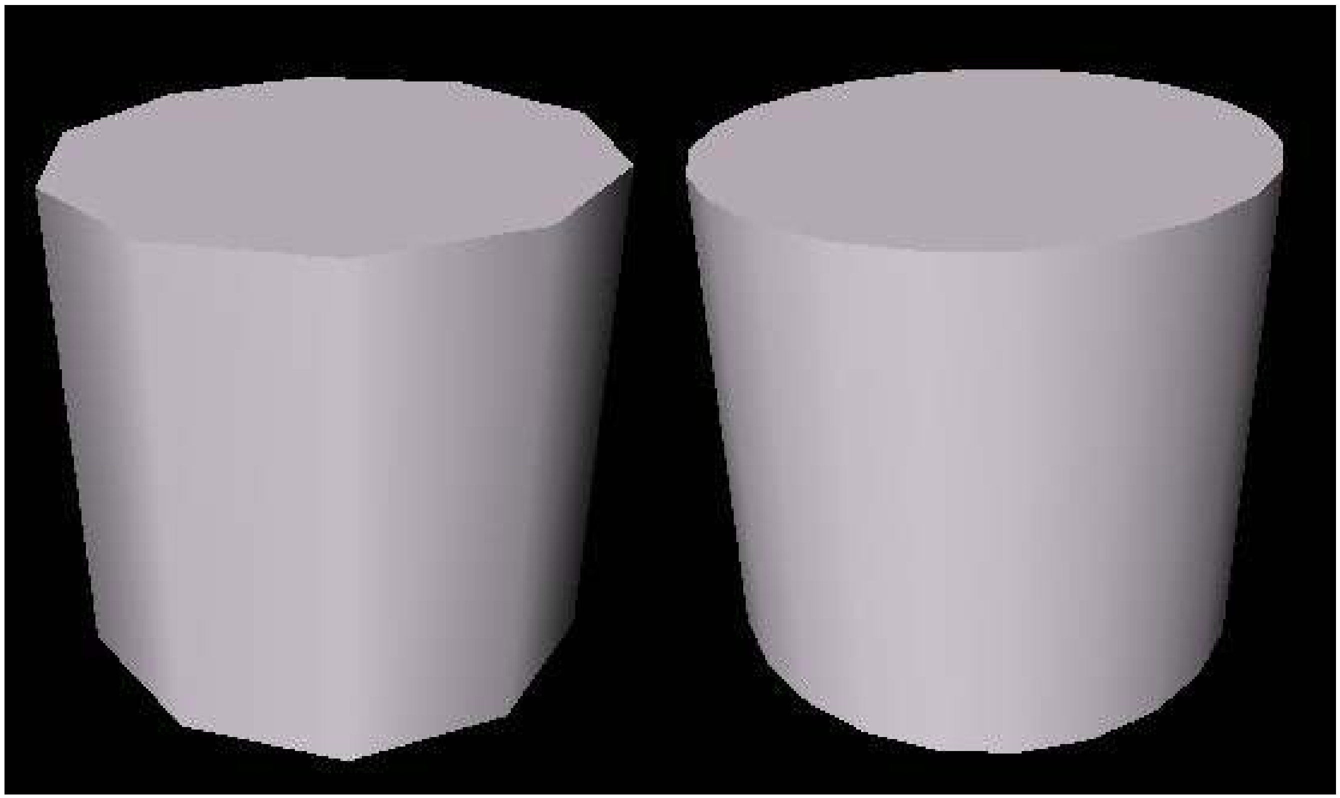
- Each polygon independent, shaded separately

- Three ways to do this
  - **Constant** – faceted. Single color per polygon
  - **Gouraud** – intensity interpolation
  - **Phong** – surface normal interpolation

Polygon mesh shading



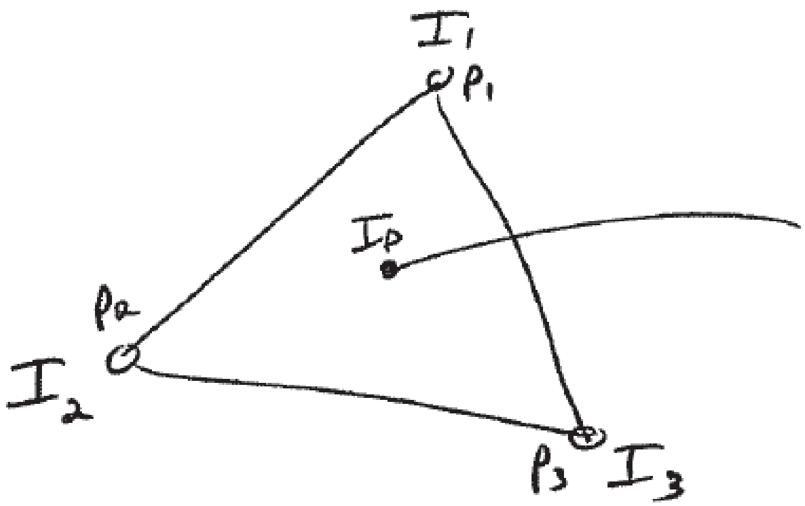
Gouraud Shading Mach bands



Gets better with more polygons ⟶

Gouraud shading

- Barycentric interpolation of illumination



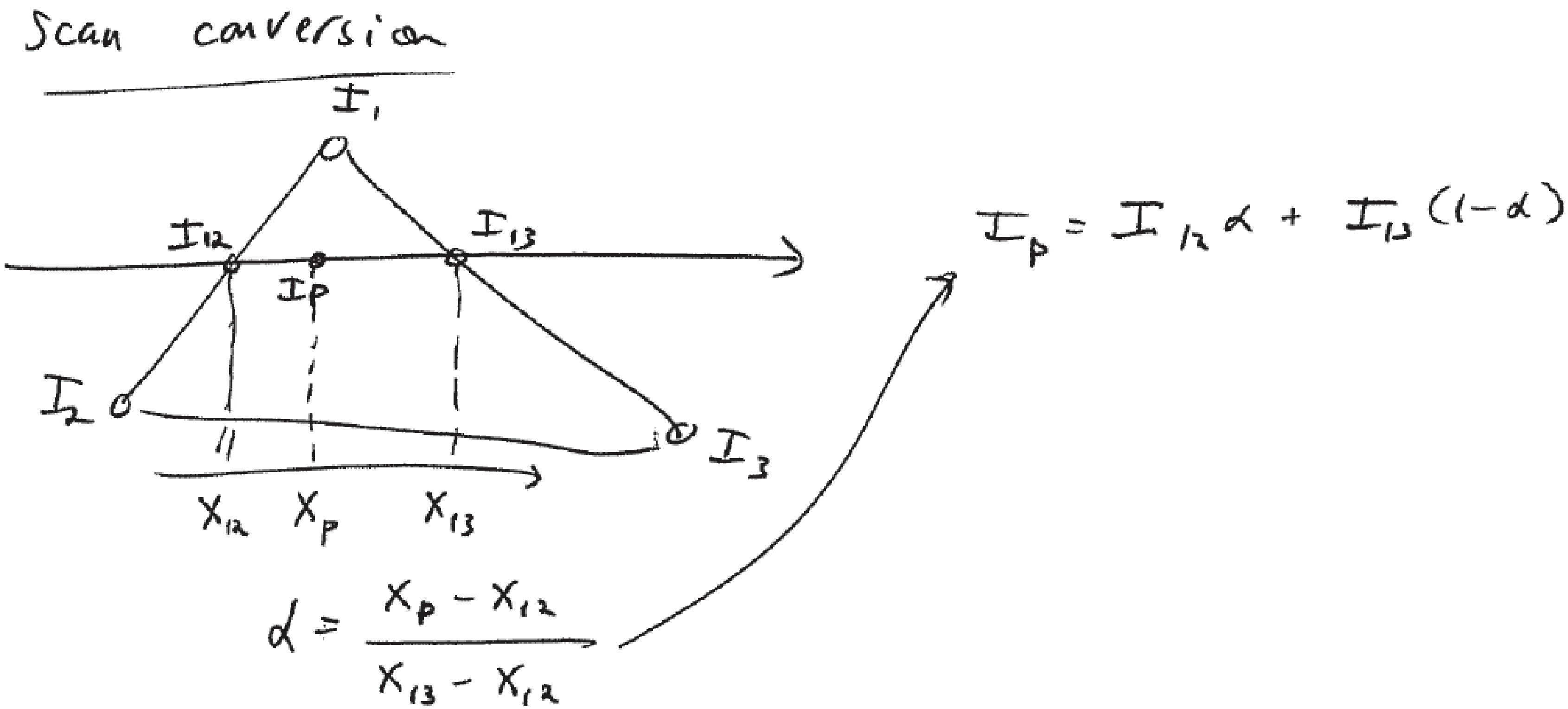
$$p = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3$$

$$I_p = \alpha_1 I_1 + \alpha_2 I_2 + \alpha_3 I_3$$

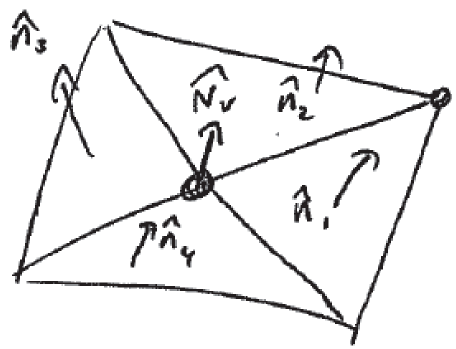
$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

Gouraud shading

- Barycentric interpolation of illumination

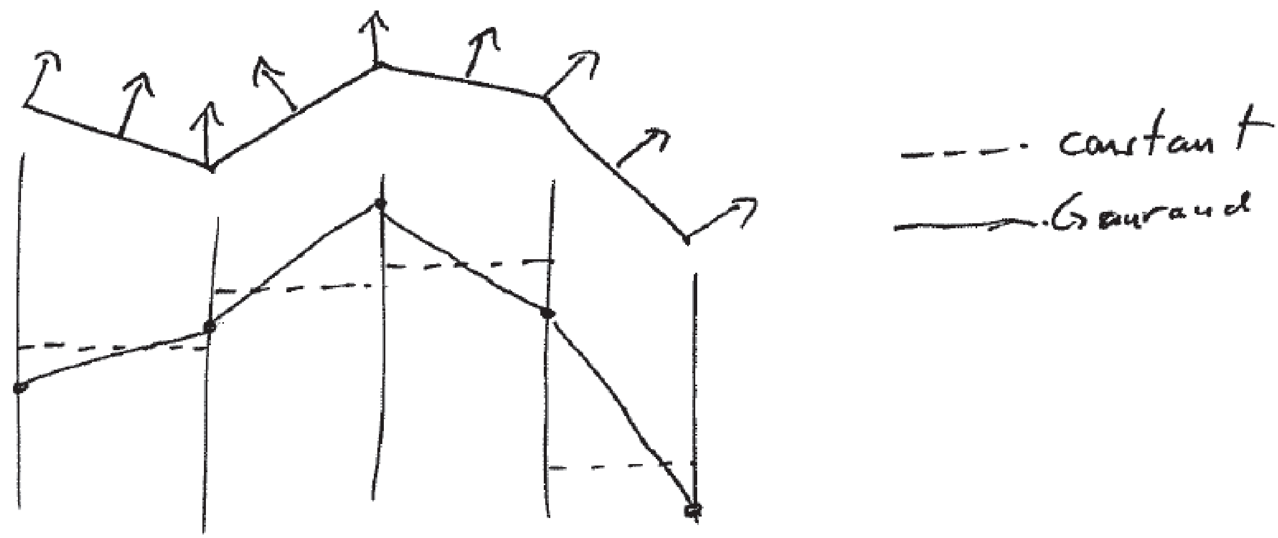


Gouraud shading



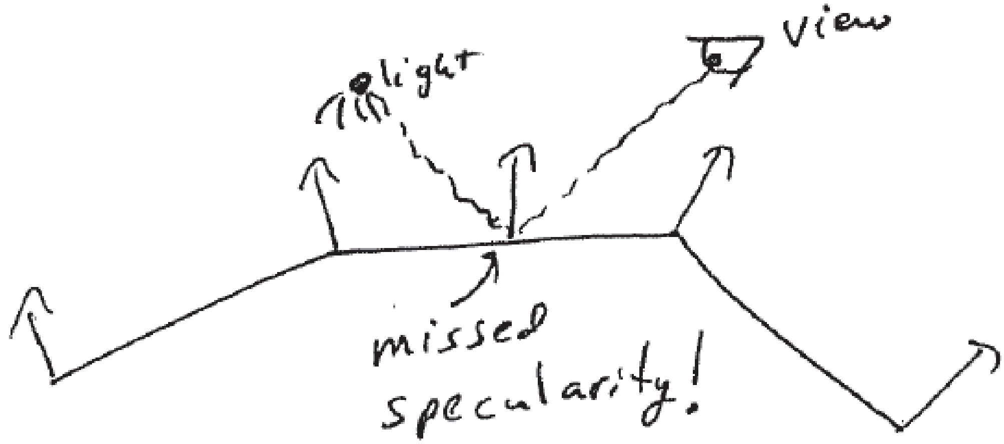
$$\hat{n}_v = \frac{\hat{n}_1 + \hat{n}_2 + \hat{n}_3 + \hat{n}_4}{\| \quad " \quad \|}$$

side view

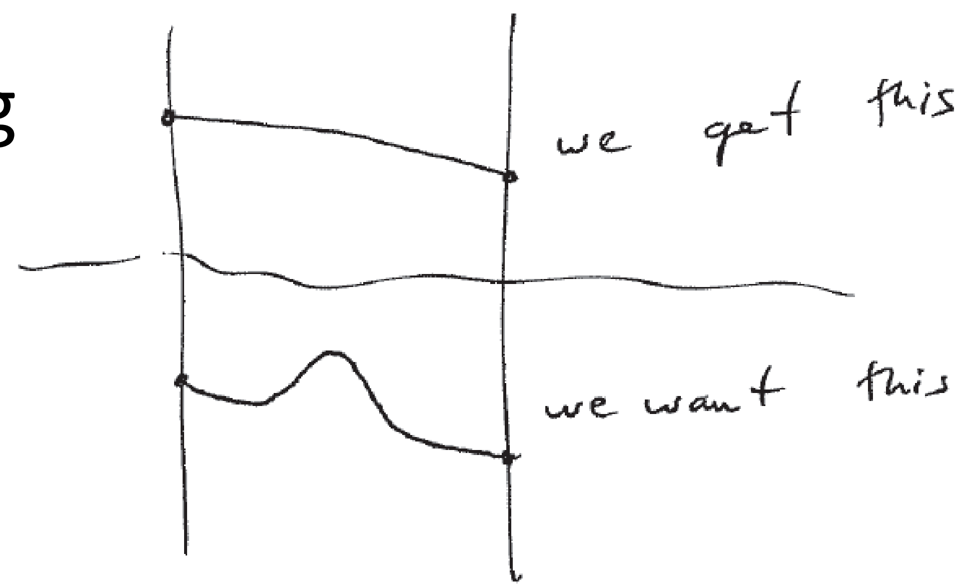


Gouraud shading

- Problems

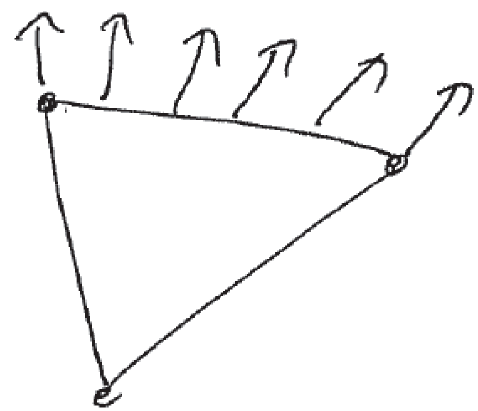


- Motion aliasing makes this worse!



Phong shading

- Interpolate normals linearly at each pixel
  - Lighting computation at each pixel



  - Looks much better
  - More expensive
  - Only works in graphics hardware (GLSL etc.)