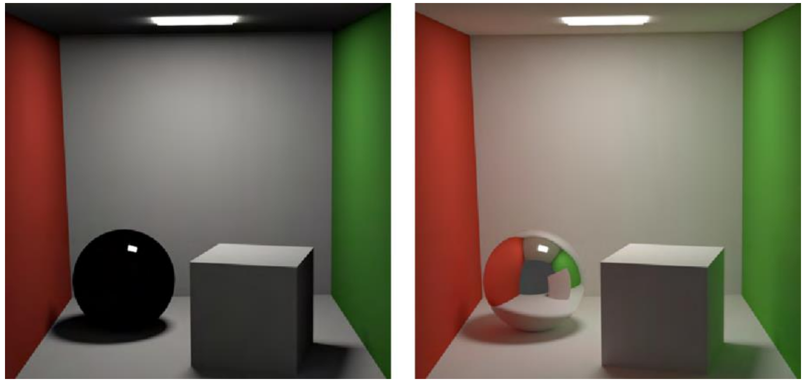
07 Lighting

Computer Graphics

Local (Direct) vs Global (Indirect) Illumination

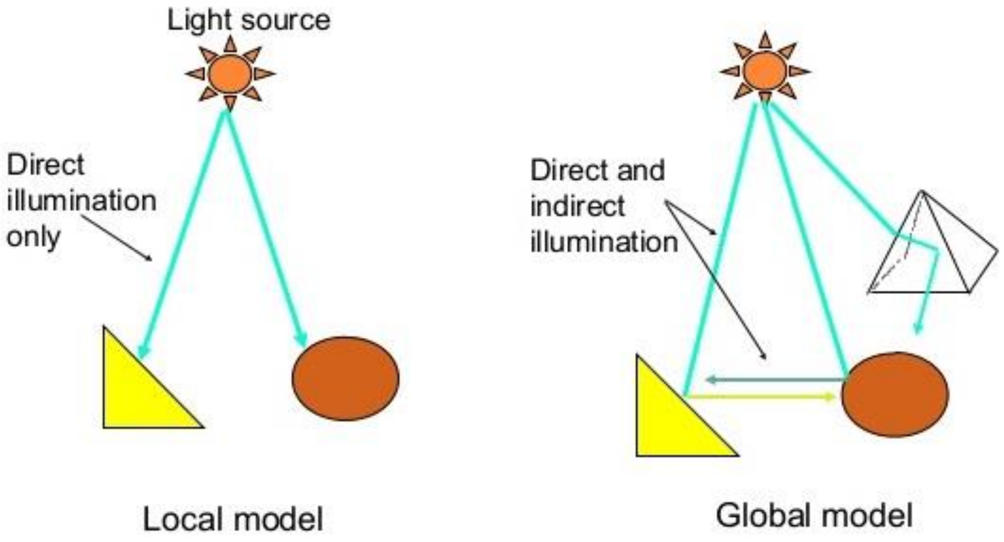
- Local Illumination
  - Direct illumination of surfaces by light sources
  - e.g. Phong and Cook/Torrence illumination

- Global (Indirect) Illumination
  - Local model + Light reflected from other surfaces to the current surface
  - More physically correct, more realistic images
  - Computationally expensive



Local Global

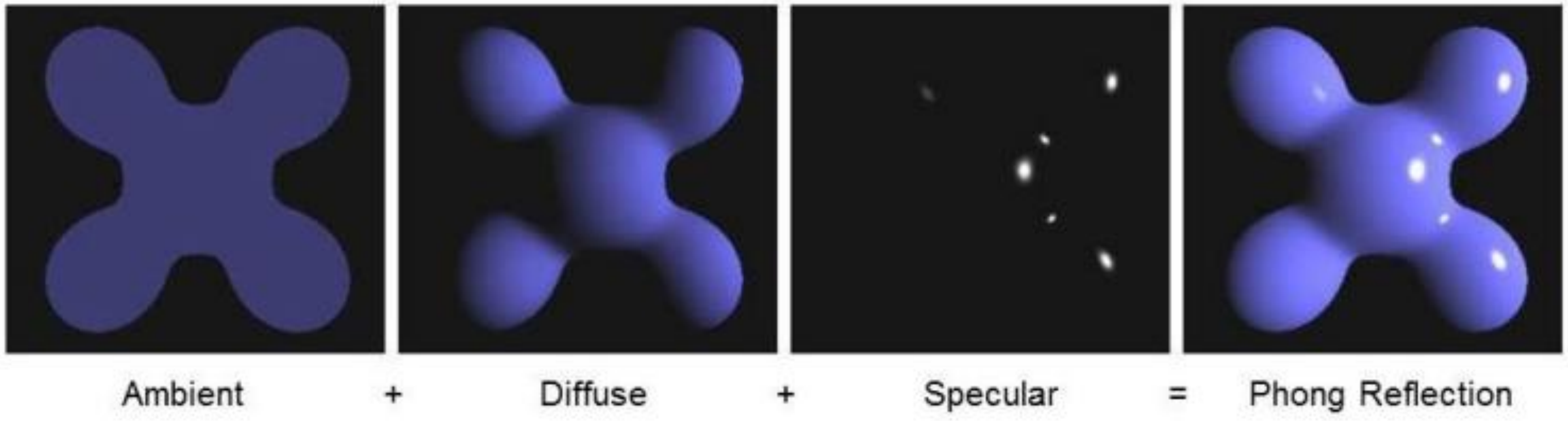
https://www.semanticscholar.org/paper/Comparing-a-Clipmap-to-a-Sparse-Voxel-Octree-for-Arneb%C3%A4ck/53a3f2a5f629745643a42658aebd9519e0fbf290/figure/0



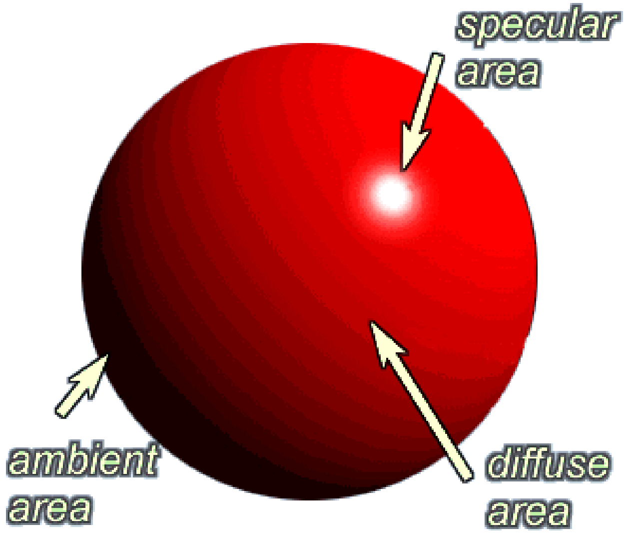
https://www.slideshare.net/michaeljamesheron/08-raytracing-and-radiosity

Phong Reflection Model

- Note
  - Phong Reflection Model ≠ Phong shading algorithm
- Reflection
  - Color of an object = Reflection of incoming lights
- Reflected light of object = Ambient reflection + Diffuse reflection + Specular reflection



https://www.chai3d.org/download/doc/html/chapter16-lighting.html



http://www.erimez.com/misc/Softimage/tutorials/si_help/introduction/si_uk_matter_intro.htm

Three Components in Reflection

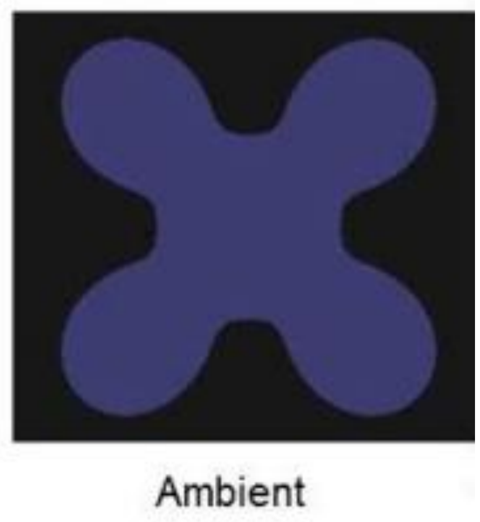
$$I = k_a I_a + k_d I_d + k_s I_s$$

$$\begin{aligned} k_a I_a &: \quad \text{ambient component} \\ k_d I_d &: \quad \text{diffuse component} \\ k_s I_s &: \quad \text{specular component} \end{aligned}$$

- k_a, k_d, k_s: Constants for controlling the proportions of three components: ambient, diffuse, and specular.

Ambient Reflection

- Ambient reflection: I_a
  - Constant, default color (without apparent lighting)
  - If no ambient reflection, we can see nothing



https://www.chai3d.org/download/doc/html/chapter16-lighting.html

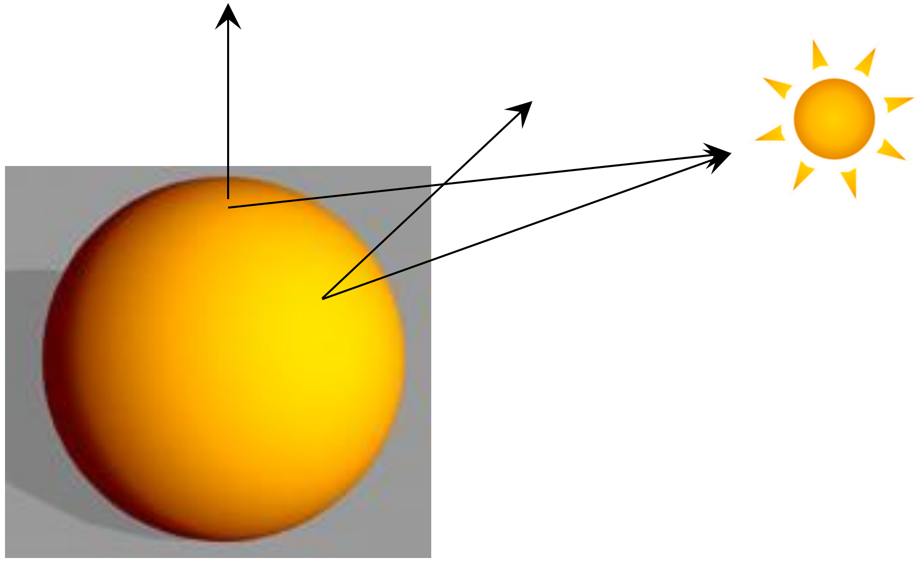
Rendering Example

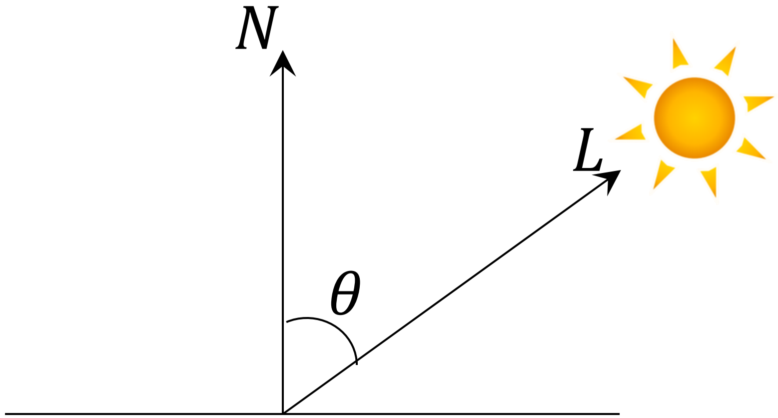
Only Ambient

https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

Diffuse Component

- $I_d = I_i \cos\theta$
  - I_i: intensity of the incident light
  - θ: angle between surface normal (N) and light source direction (L)
- $I_d = I_i \, (L \cdot N)$
  - If both L and N are unit vectors





Rendering Example

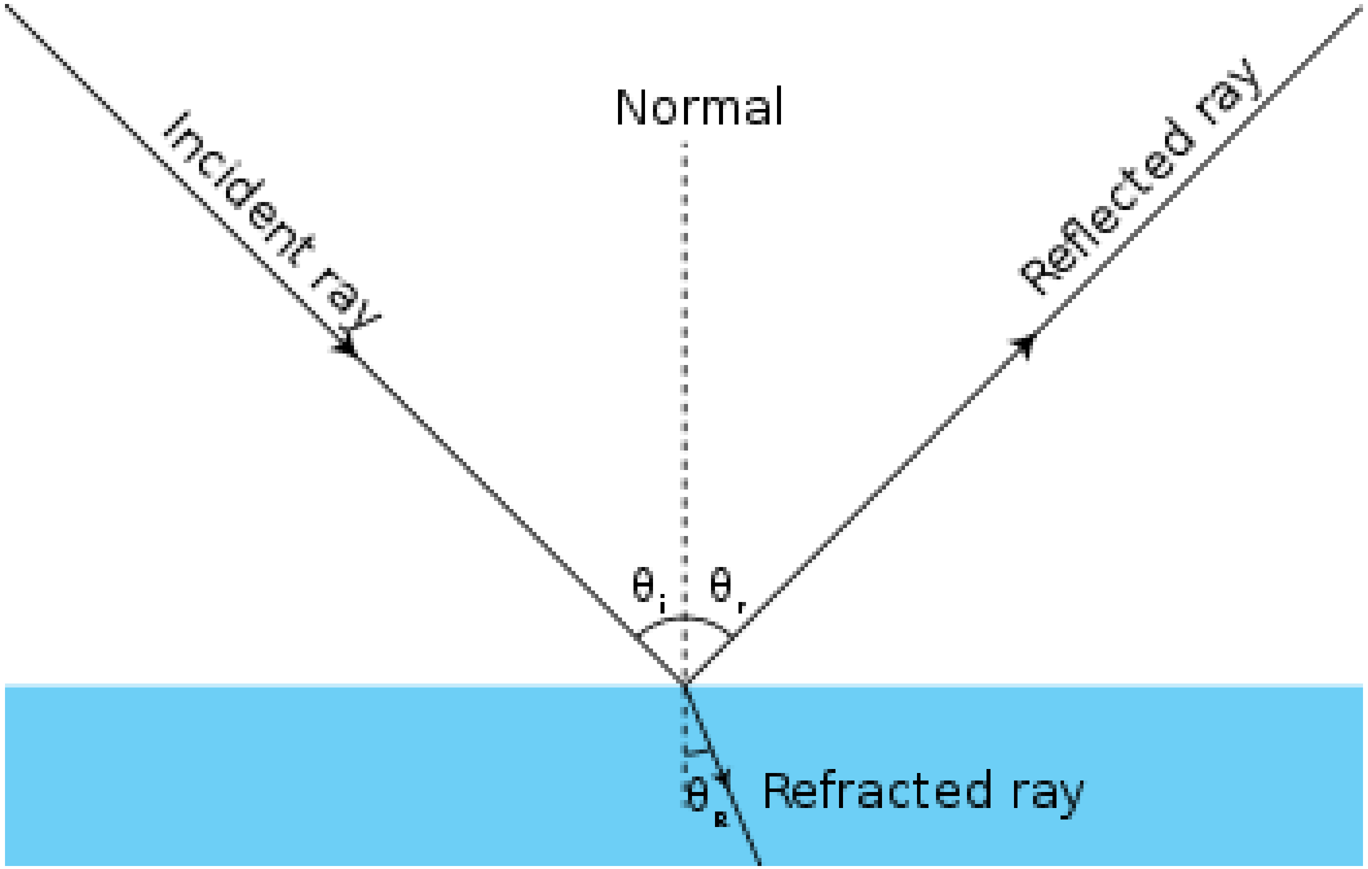
Only Ambient

https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

Ambient + Diffuse

https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

Incident and Reflected Light



https://en.wikipedia.org/wiki/Ray_(optics)

Specular Component

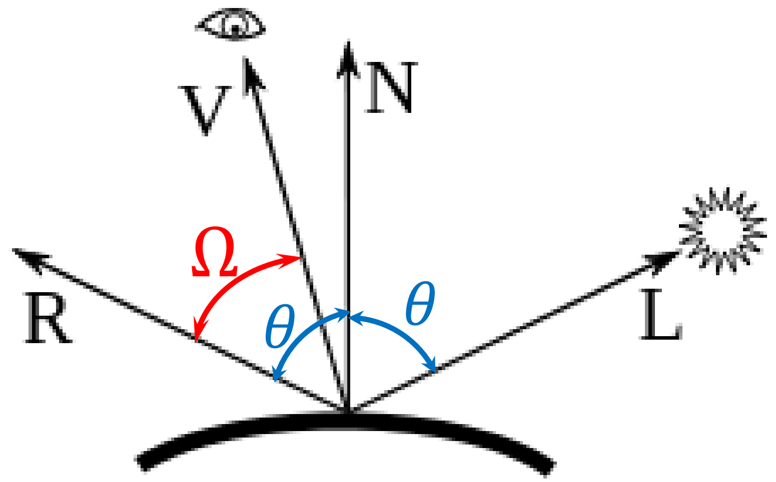
$$
\begin{aligned}
I_s &= I_i \cos^n \Omega \\
&= I_i (R \cdot V)^n
\end{aligned}
$$

V : viewing direction (unit vector)

R : reflection direction (unit vector)

Ω : angle between V and R

n : index of degree of imperfection of surface



https://commons.wikimedia.org/wiki/File:Blinn_Vectors.svg

Rendering Example

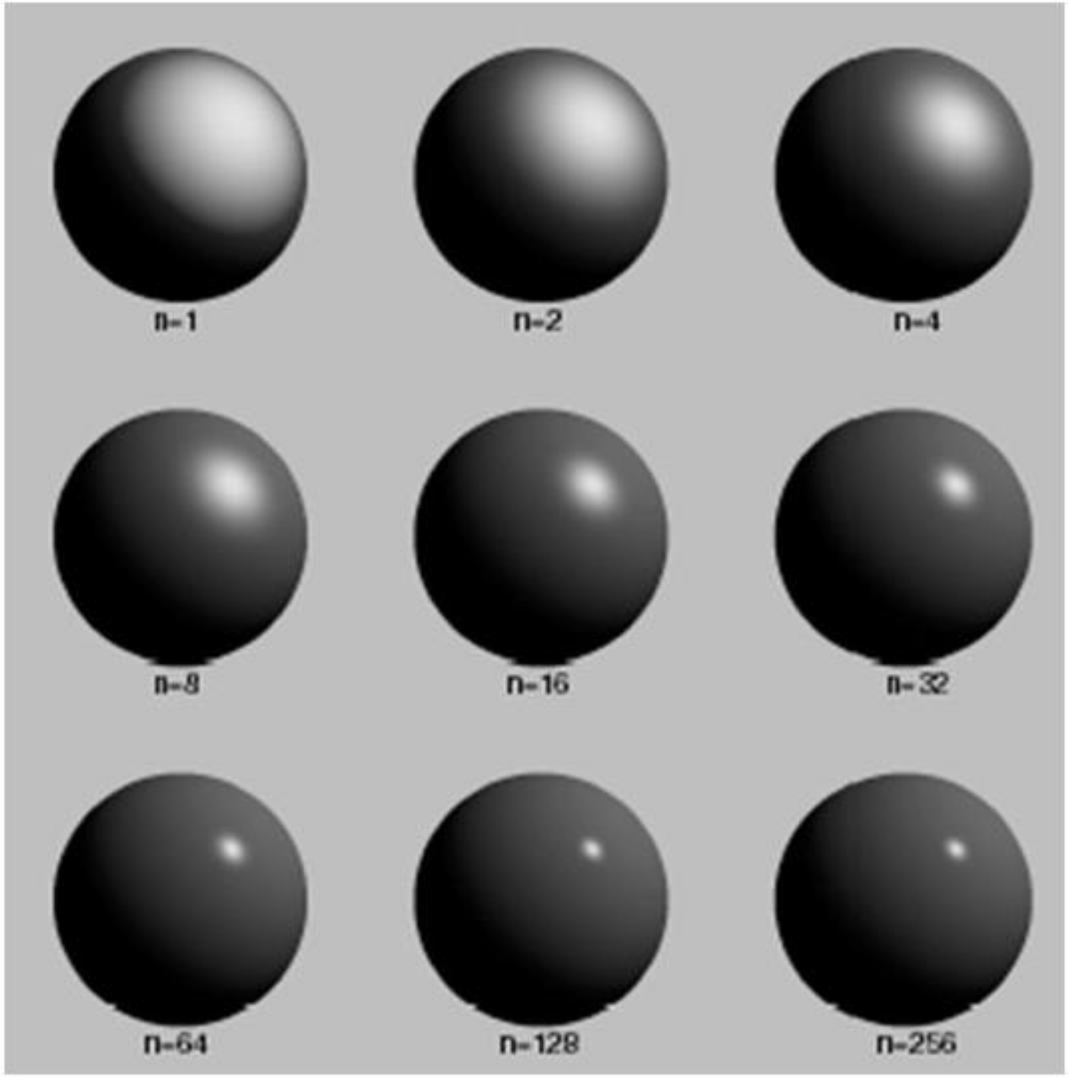
Ambient + Diffuse + Specular

https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

Ambient + Diffuse

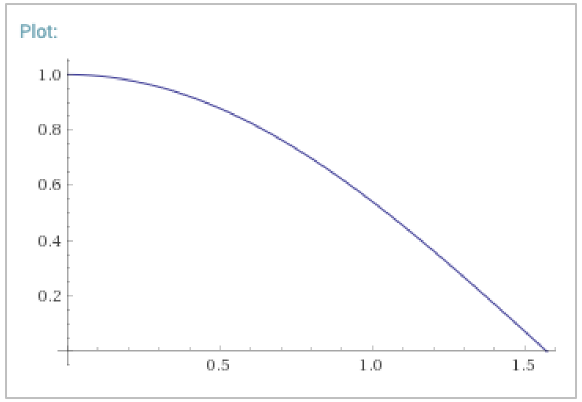
https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

Effect of n in Specular Reflection

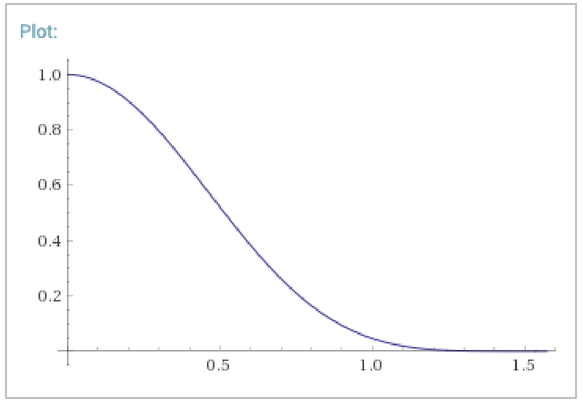


https://slideplayer.com/slide/7698973/

$\cos\theta$

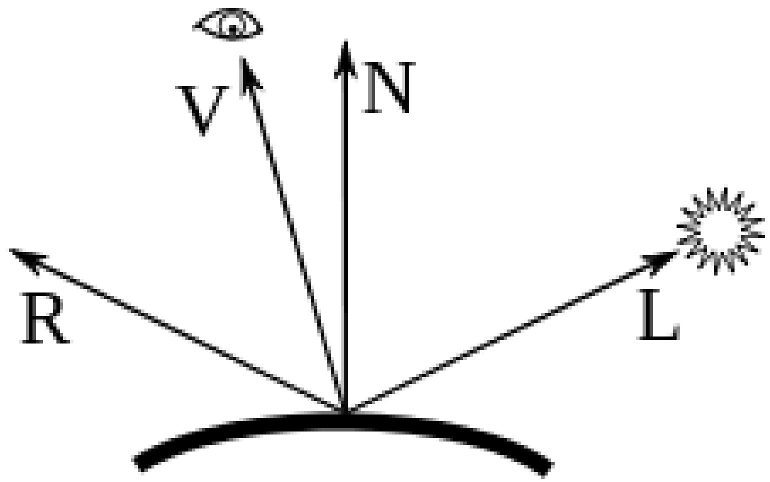


$\cos^5\theta$



Phong Reflection Model (Final)

$$I = k_a I_a + I_i(k_d(L \cdot N) + ks(R \cdot V)^n)$$



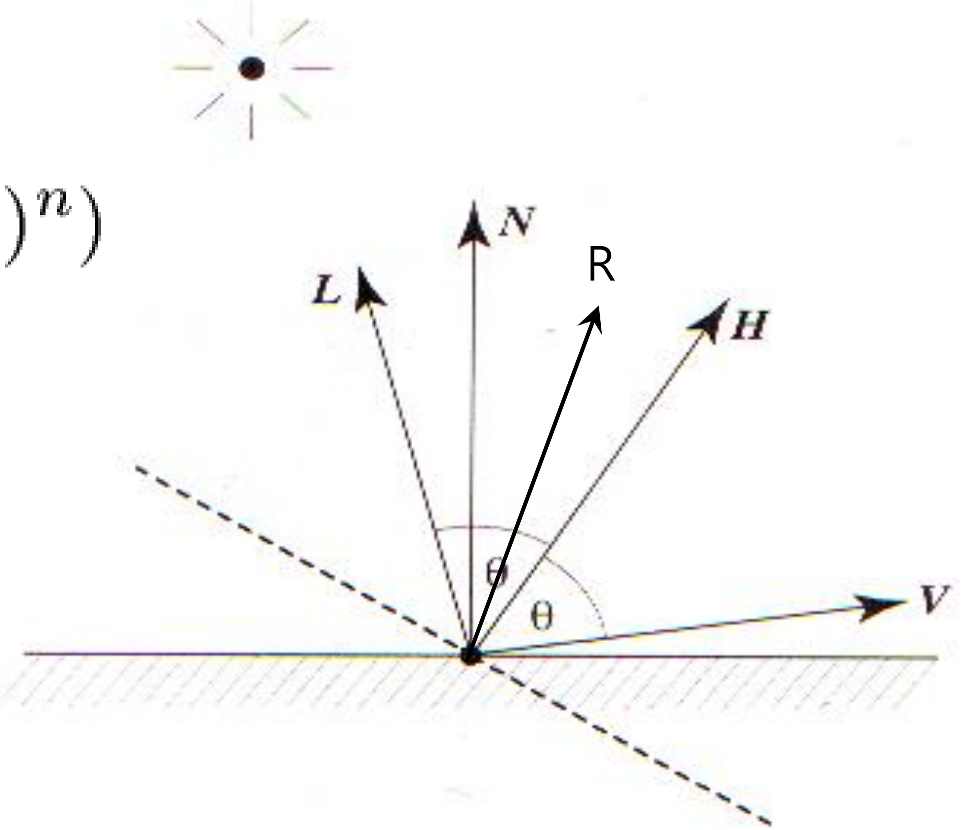
https://commons.wikimedia.org/wiki/File:Blinn_Vectors.svg

Simplification

$$I = k_a I_a + I_i(k_d(L \cdot N) + ks(R \cdot V)^n)$$

- Let $H = (L + V) / 2$.
- We can use $(N \bullet H)$ instead of $(R \bullet V)$ because it takes too much time to compute exact R.
- Now, the approximation version:

$$I = k_a I_a + I_i(k_d(L \cdot N) + ks(N \cdot H)^n)$$



Observation: If L goes left, then:
- R goes right so theta decreases
- H goes left so angle between N and H decreases

Reflection Model - Components

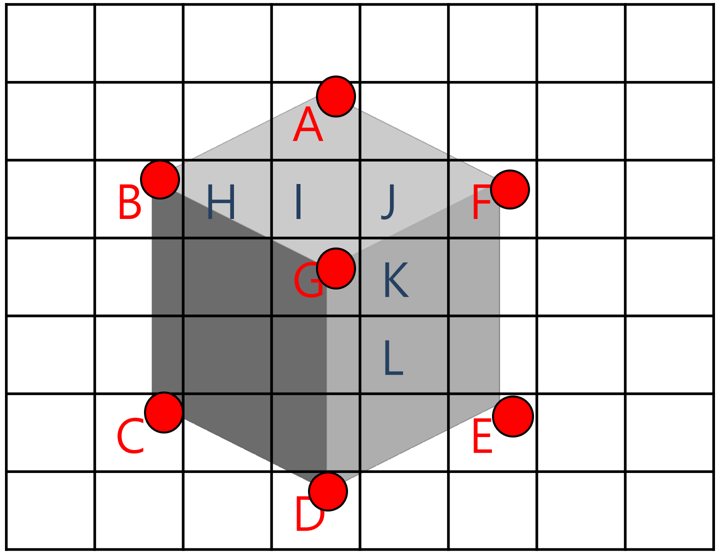
- R, G, B components

$$I_r = k_{ar}I_{ar} + I_{ir}(k_{dr}(L \cdot N) + k_{sr}(N \cdot H)^n)$$
$$I_g = k_{ag}I_{ag} + I_{ig}\left(k_{dg}(L \cdot N) + k_{sg}(N \cdot H)^n\right)$$
$$I_b = k_{ab}I_{ab} + I_{ib}(k_{db}(L \cdot N) + k_{sb}(N \cdot H)^n)$$

where
(I_r, I_g, I_b): final color for (r, g, b) components
$\left(k_{ar}, k_{ag}, k_{ab}\right)$: ambient constants for (r, g, b) components
$\left(k_{dr}, k_{dg}, k_{db}\right)$: diffuse constants for (r, g, b) components
$\left(k_{sr}, k_{sg}, k_{sb}\right)$: specular constants for (r, g, b) components
(I_{ar}, I_{ag}, I_{ab}): ambient light (color) for (r, g, b) components
(I_{ir}, I_{ig}, I_{ib}): incident light (color) for (r, g, b) components

Interpolative Shading Techniques

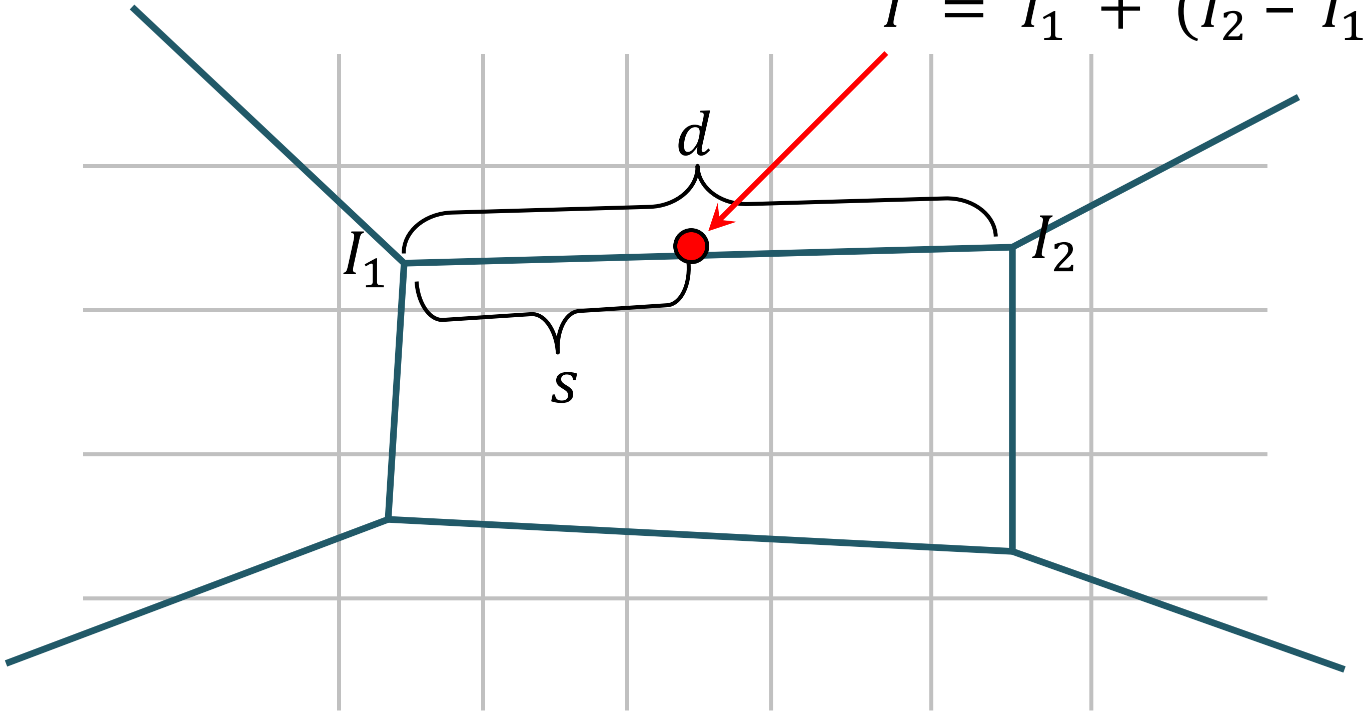
- Technique for computing intermediate pixels in polygon rendering with lighting
  - pixels including vertices: A, B, C, D, E, F, G
    - intensity can be computed using vertex coordinates and vertex normals
  - intermediate pixels (ex. H, I, J, K, L, ...)
    - How do we compute the intensities?
- Two alternatives
  - Gourad shading
  - Phong shading (≠ Phong illumination model)



Gouraud Shading

- Intensity is computed only at each vertex with given vertex coordinates and normal. ex) I_1 and I_2 in the figure
- Intensity of intermediate pixels: interpolated from the intensities of the pixels of vertices

$$I = I_1 + (I_2 - I_1) * (s/d)$$

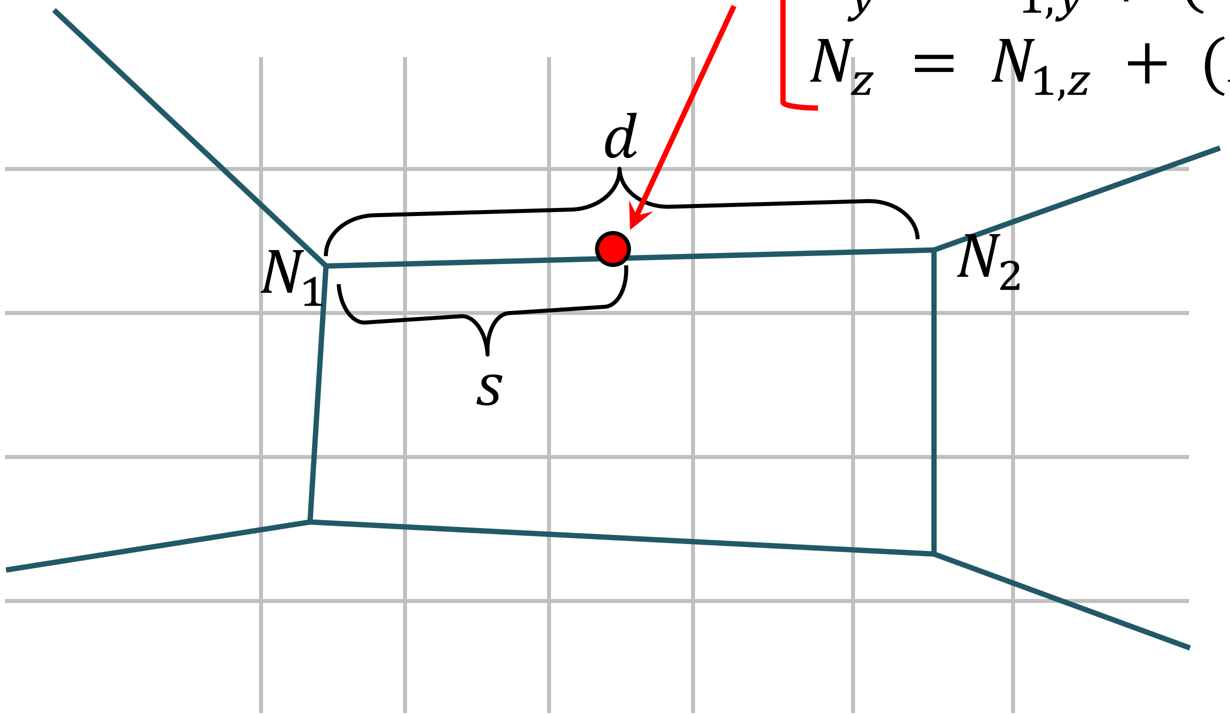


Phong Shading

- Interpolate the normal vectors
- Compute the intensity at every pixel

$$\boxed{I = k_a I_a + I_i(k_d(L \cdot N) + ks(R \cdot V)^n)}$$

$$\begin{cases} N_x = N_{1,x} + (N_{2,x} - N_{1,x}) * (s/d) \\ N_y = N_{1,y} + (N_{2,y} - N_{1,y}) * (s/d) \\ N_z = N_{1,z} + (N_{2,z} - N_{1,z}) * (s/d) \end{cases}$$



Gouraud v.s. Phong Shading

Gouraud

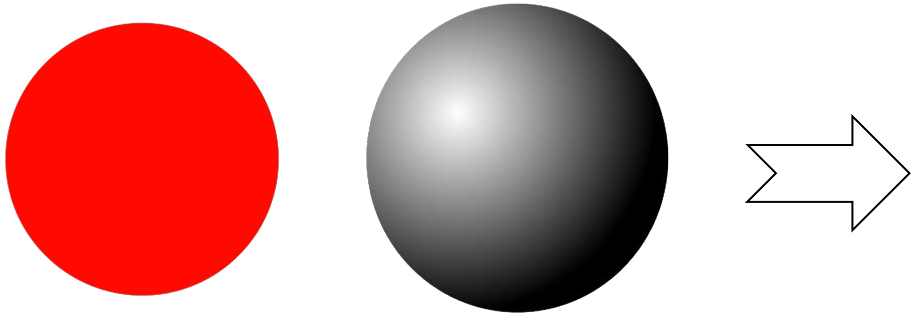
https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

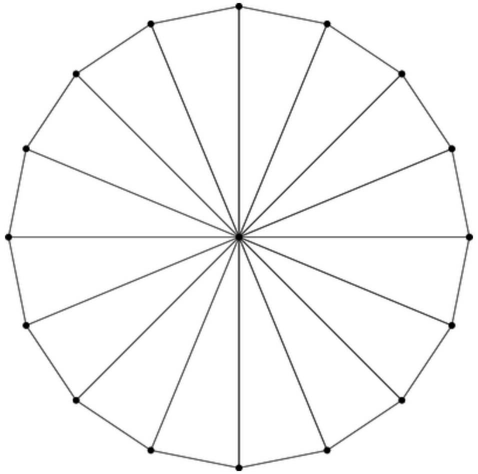
Phong

https://www.slideserve.com/chars/illumination-and-shading-powerpoint-ppt-presentation

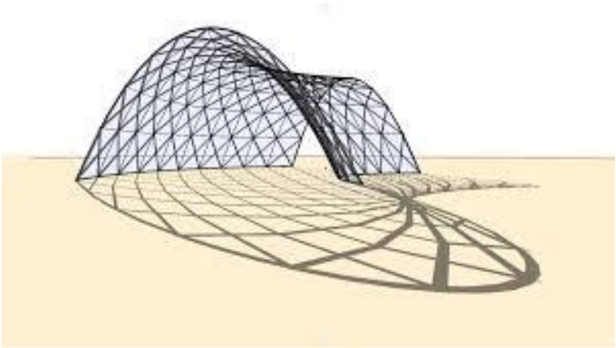
Smooth v.s Flat Shading - 1

- No curved primitives in OpenGL such as circle, sphere, and free-form surface
  - They should be rendered only with lines and triangles

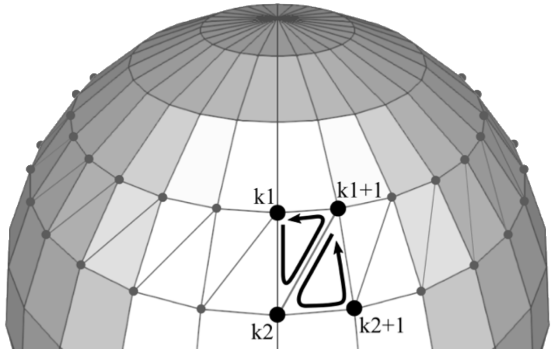




https://nullprogram.com/blog/2014/06/01/



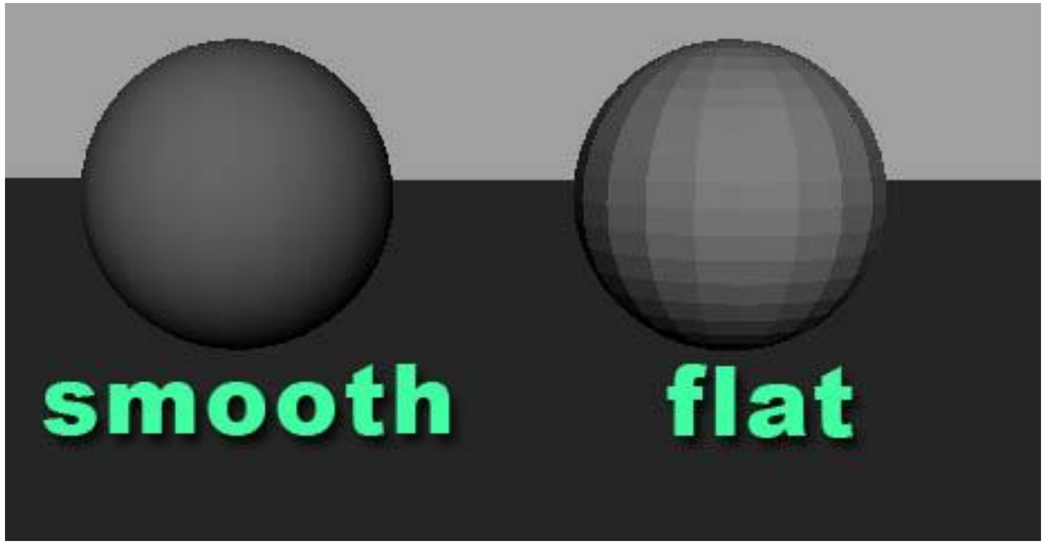
https://horstsondermann.com/grasshopper-archicad-freeform-surface-grid/



http://www.songho.ca/opengl/gl_sphere.html

Smooth v.s Flat Shading - 2

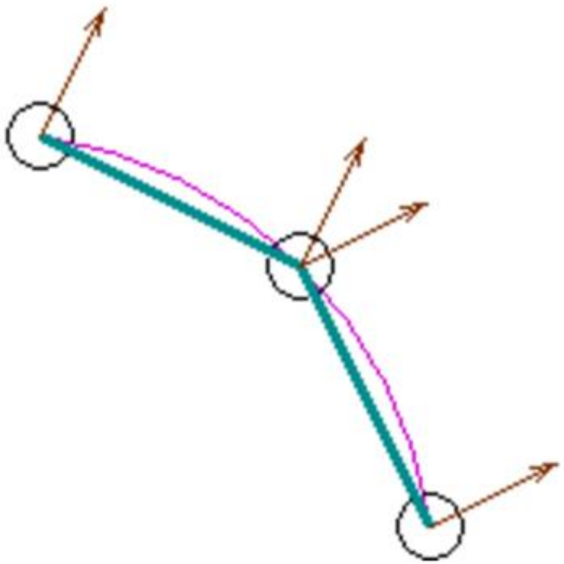
- Use the same primitives (triangles) but rendered differently
- Smooth shading
  - Render the curved objects with smoothly interpolated normal vectors
- Flat shading
  - Render the curved objects with discrete normal vectors

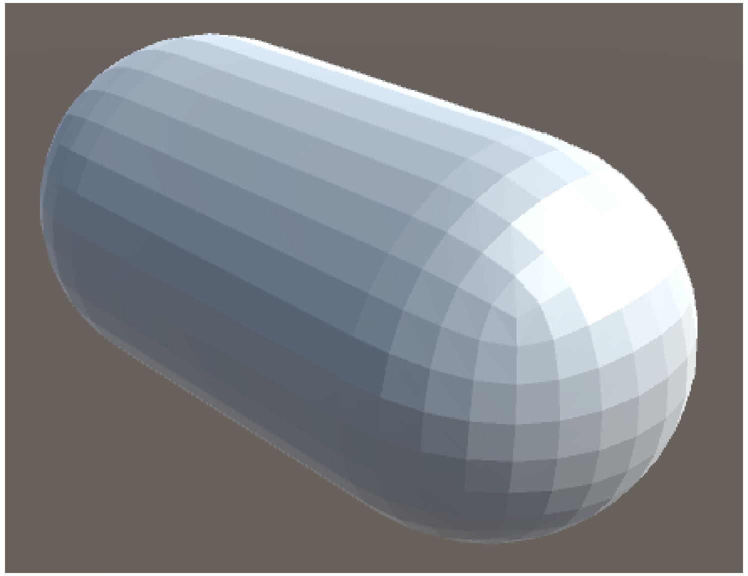


https://stackoverflow.com/questions/34731942/how-to-do-flat-shaded-polygons-geometry-in-scenekit-i-e-not-smooth

Flat Shading

- No vertex normal but only face normal for mesh object
- At each vertex, the normal is the same as the face normal
- Common vertices have different normal for each polygon

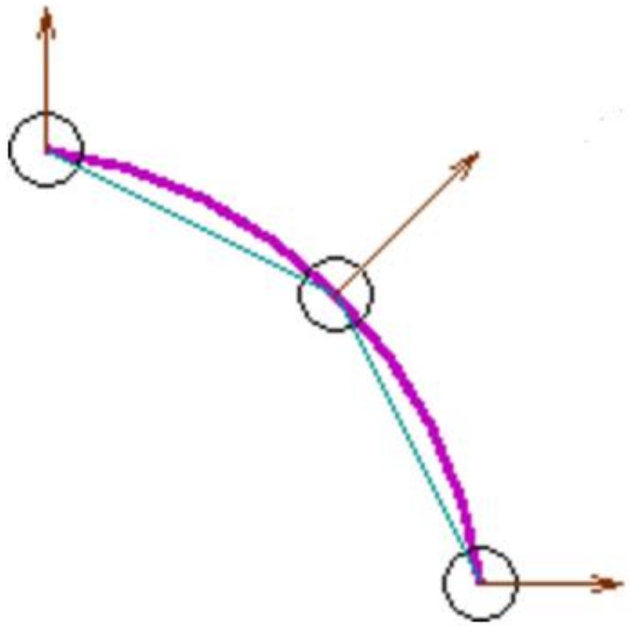


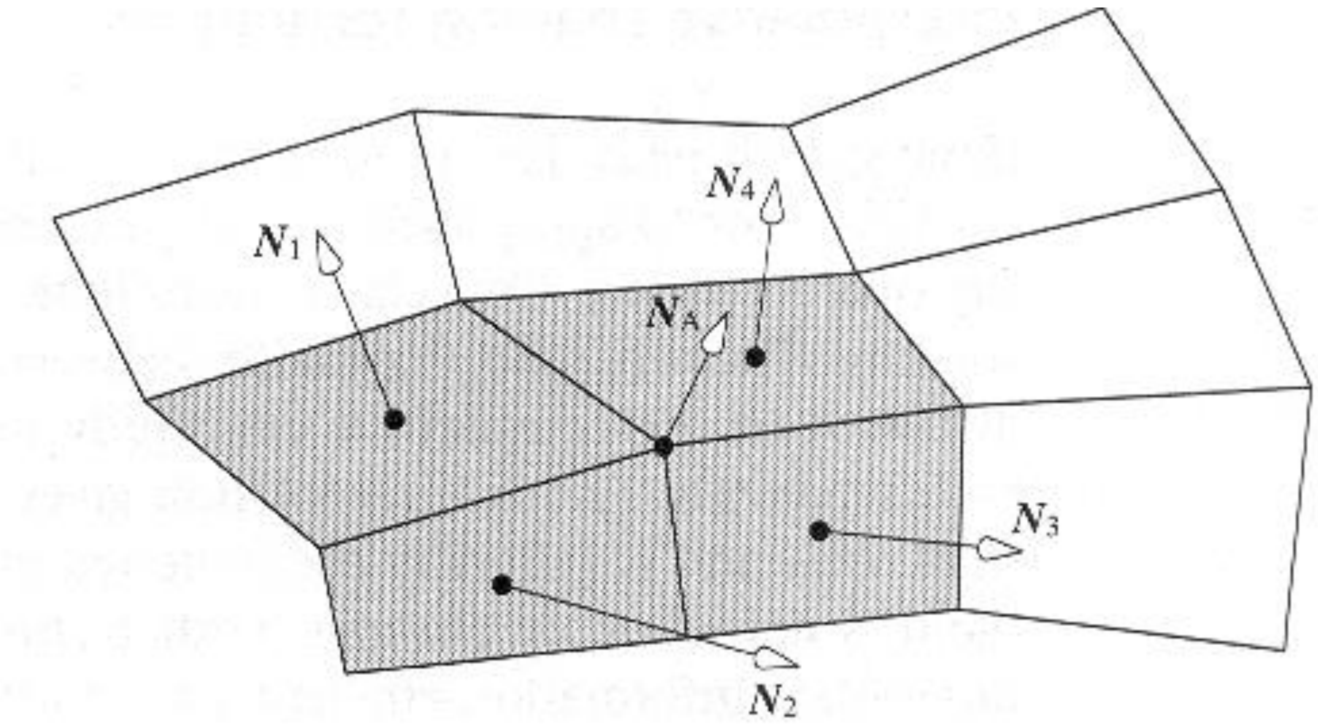


https://catlikecoding.com/unity/tutorials/advanced-rendering/flat-and-wireframe-shading/

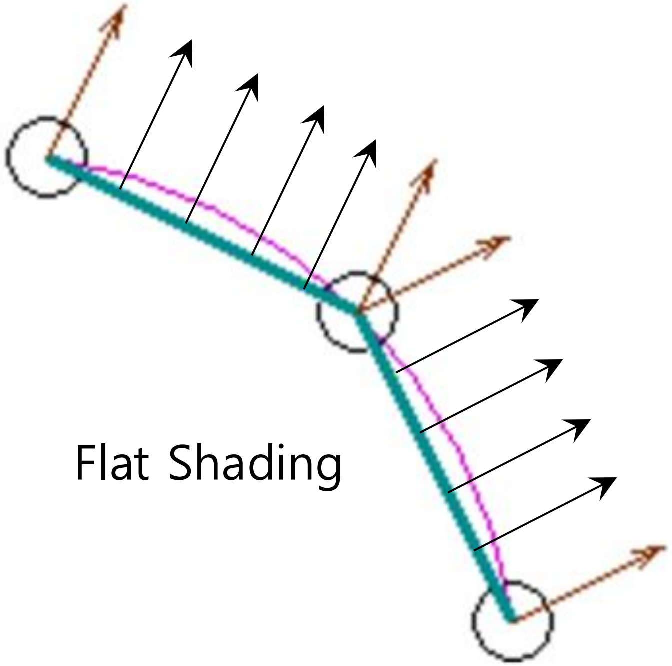
Smooth Shading

- Normal vector: computed at each vertex
  - Vertex normal: average of all incident face's (unit) normal vectors
  - Ex) $N_A = (N_1 + N_2 + N_3 + N_4)/4$

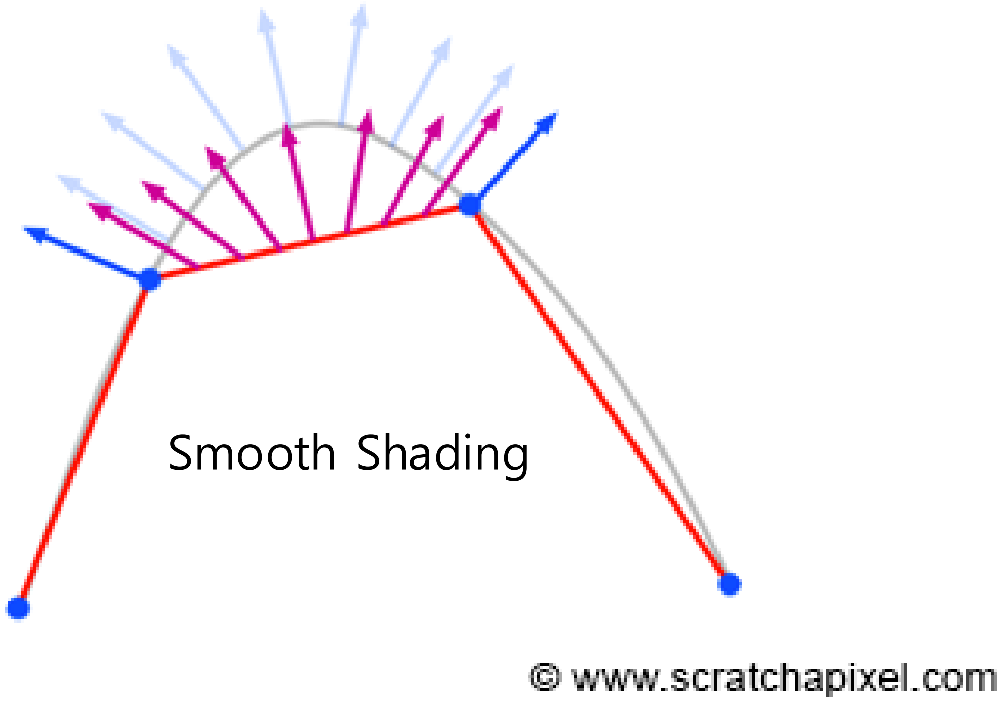




Interpolating Normal for Flat and Smooth Shading



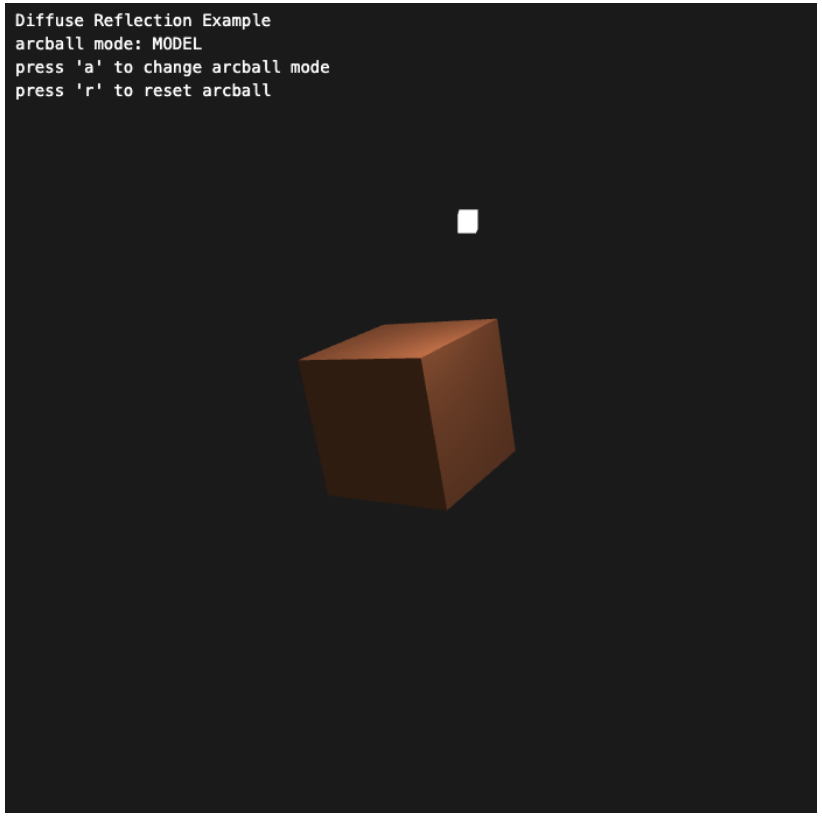
Parallel normals in a face



Interpolated normals in a face

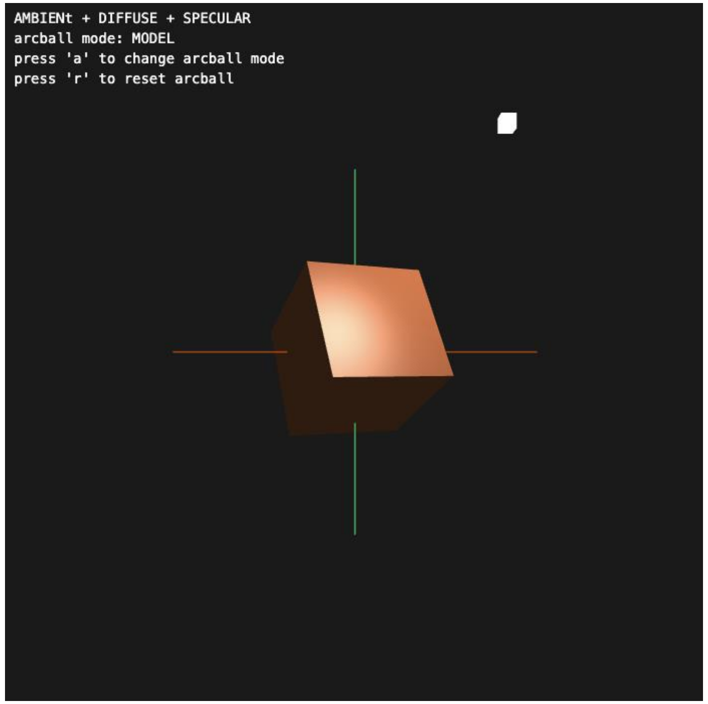
Flat and Smooth shadings can be implemented both with Gouraud and Phong Shading

Program 15_LightDiffuse



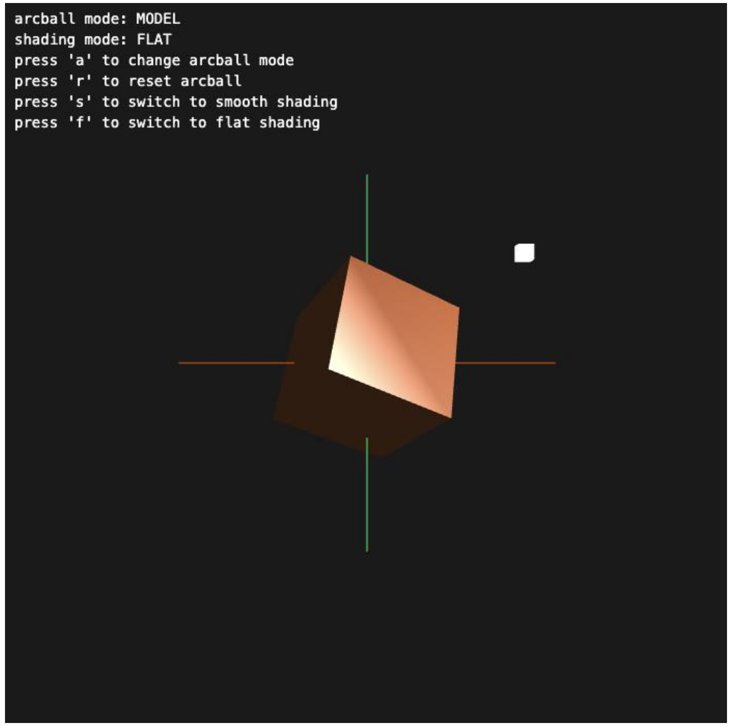
- Keyboard
  - r: reset the arc ball to initial state
  - a: toggle switch arc ball mode
- Mouse
  - left button: arc ball dragging

Program 16_LightSpecular



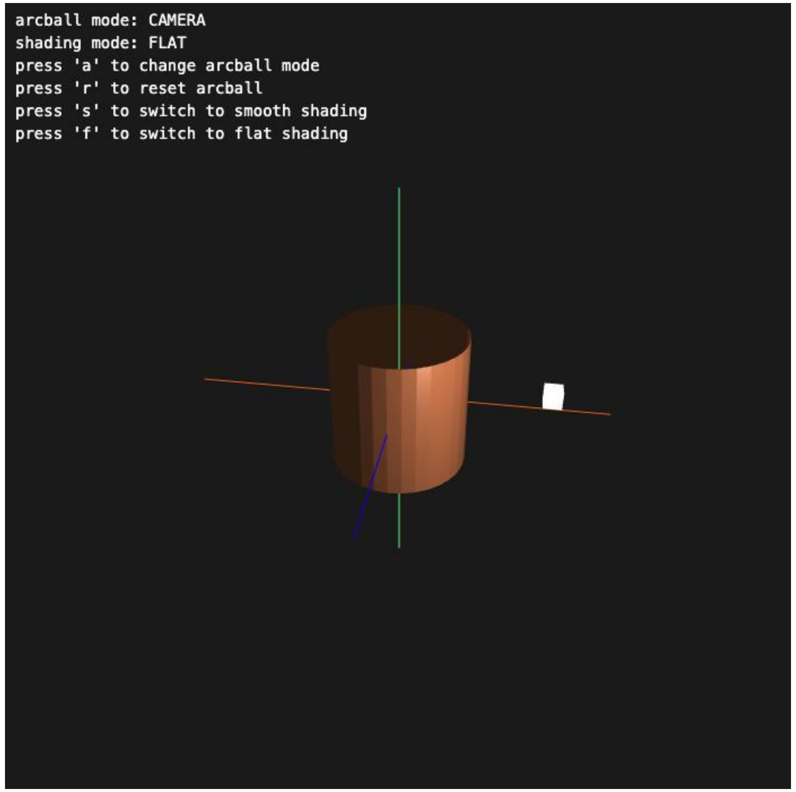
- Keyboard
  - r: reset the arc ball to initial state
  - a: toggle switch arc ball mode
- Mouse
  - left button: arc ball dragging

Program 17_GouraudShading



- Keyboard
  - r: reset the arc ball to initial state
  - a: toggle switch arc ball mode
  - s: smooth shading
  - f: flat shading
- Mouse
  - left button: arc ball dragging

Program 18_SmoothShading



- Keyboard
  - r: reset the arc ball to initial state
  - a: toggle switch arc ball mode
  - s: smooth shading
  - f: flat shading
- Mouse
  - left button: arc ball dragging