Computer Graphics

Lighting and Shading

Illumination in Computer Graphics

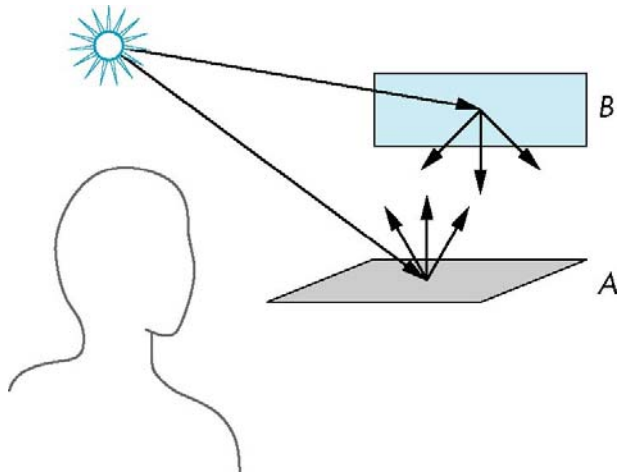
- In computer graphics the interaction between the lights and surfaces are modeled to find the color and brightness of a point on a surface.
- An illumination model considers
  - Object properties (color, transparency, ..)
  - Light properties (intensity, color, ..)
  - Model of the interaction between lights and objects

Light and Material Interaction

- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation

Scattering

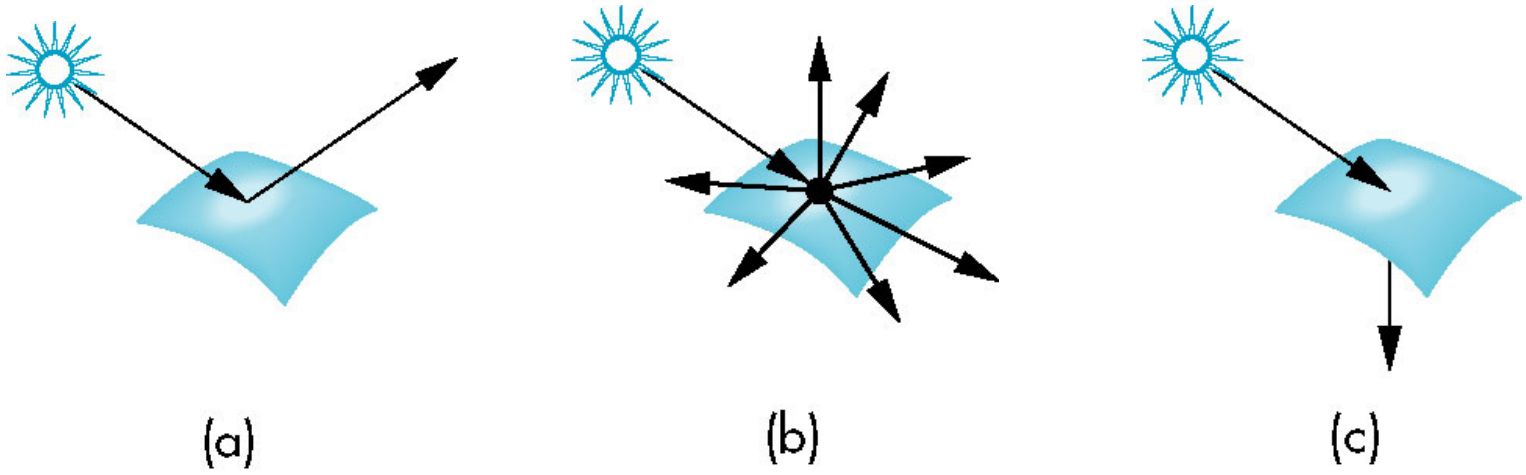
- Light strikes an object, A, some of it get absorbed and some of it is scattered.
- The scattered light hits some other object, B, some get absorbed and some of it is scattered
- This process is repeated
- The infinite scattering and absorption of light cannot be solved in general



Light & Material

- The amount reflected determines the color and brightness of the object
  - A surface appears green under white light because the green component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Light Reflection



specular diffuse translucent

Simple Light Sources

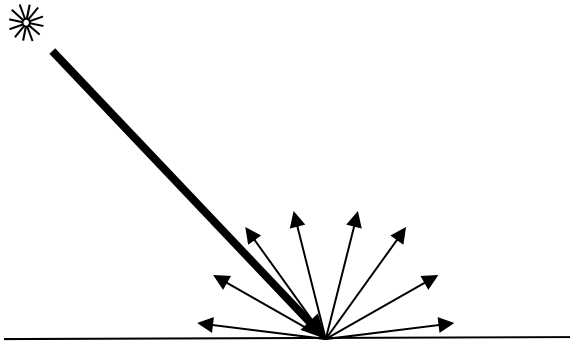
- Point light:
  - Specified by position and intensity
- Directional light
  - Specified by direction and intensity
- Spot lights
  - Specified by position, direction of center line, falloff exponent, and intensity

Ambient

- Independent of the location of viewer, light, and object
- Simplest illumination model.
- Global ambient illumination in the scene, I_a
- Ambient reflection coefficient of the object, k_a.
- $I = I_a.k_a$
- No physical basis

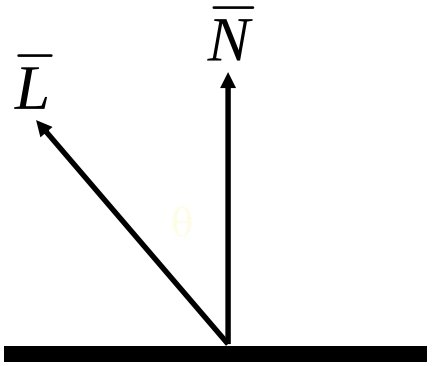
Diffuse Reflection.

- Reflect light with equal intensity in all directions.
- Brightness depends on the angle between the surface normal and the vector to the light source.



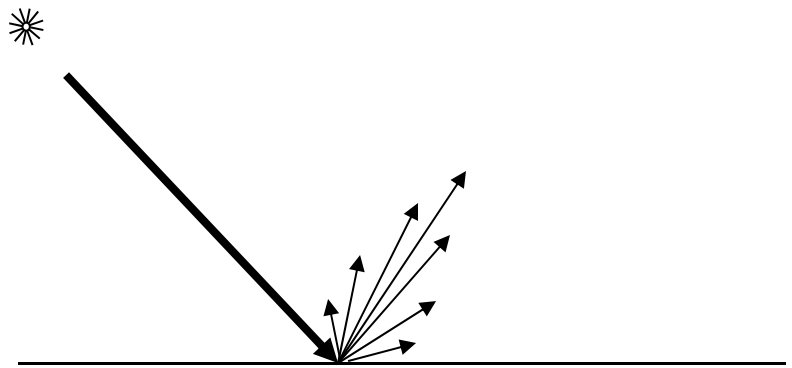
$$I = I_p k_d \cos\theta$$

$$I = I_p k_d (\overline{N}.\overline{L})$$

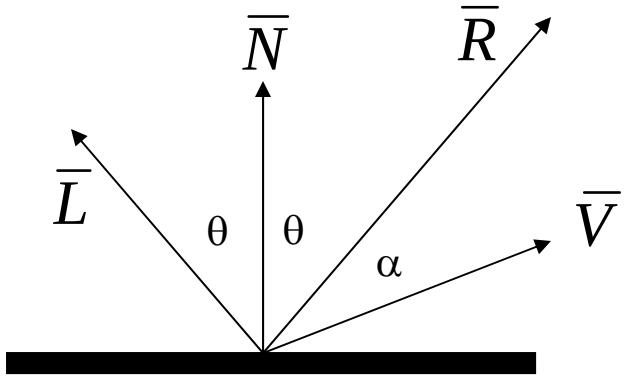


Specular

- Does not reflect light equally in all direction
- Favors the direction of ideal reflection
- Causes surfaces to appear shiny



Phong Specular Model.



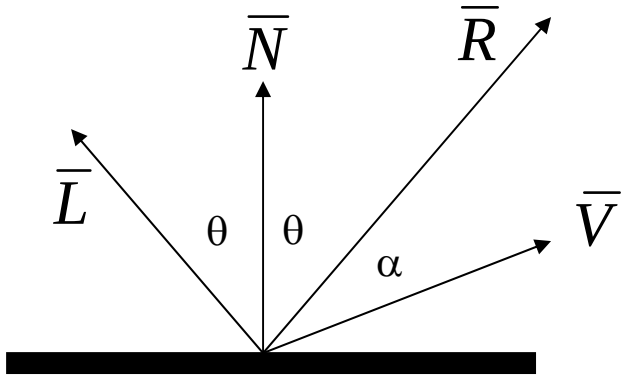
$$I = I_p k_s \cos^n \alpha$$

$$I = I_p k_s \left(\overline{R}.\overline{V}\right)^n$$

Use four vectors

- To light source L
- To viewer V
- Normal N
- Perfect reflector R

Ambient, Diffuse, and Specular

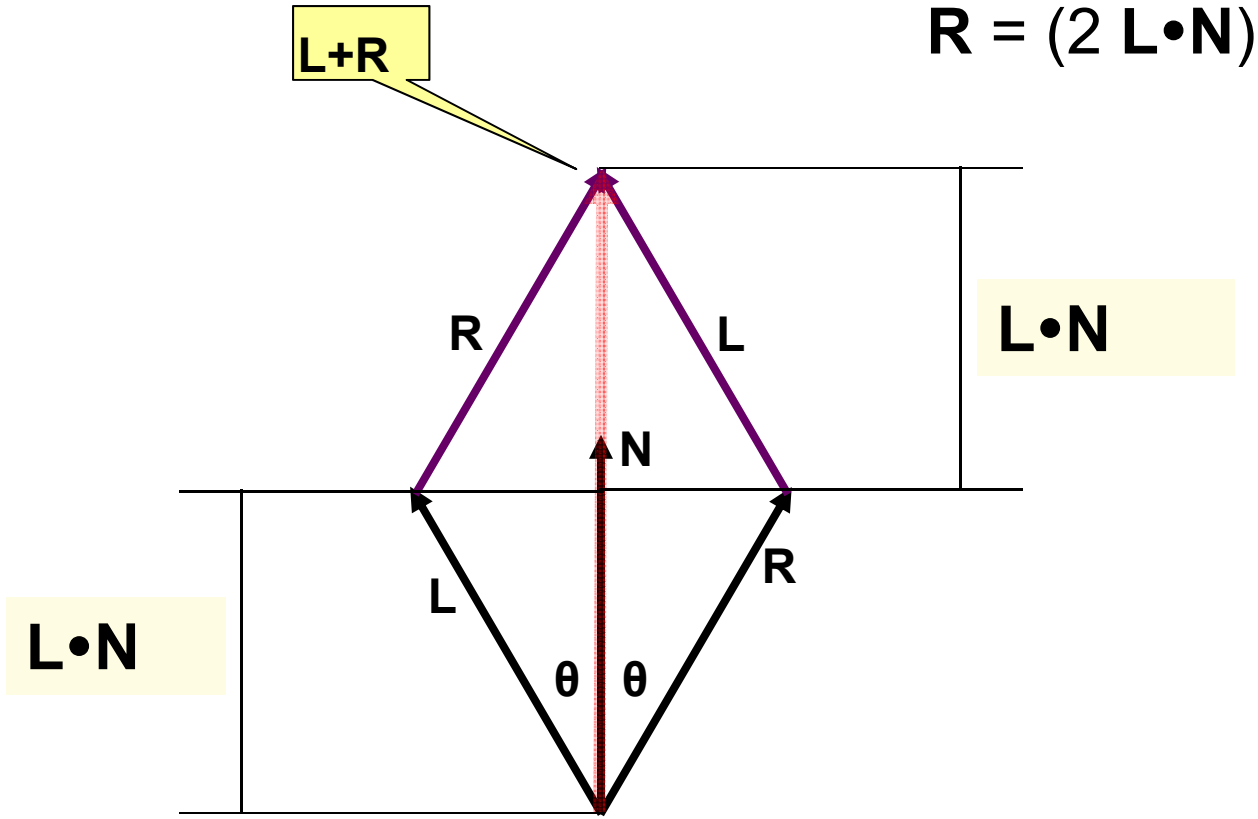


$$I = I_a k_a + I_p [k_d \cos\theta + k_s \cos^n \alpha]$$

Computing **R** (Reflection Vector)

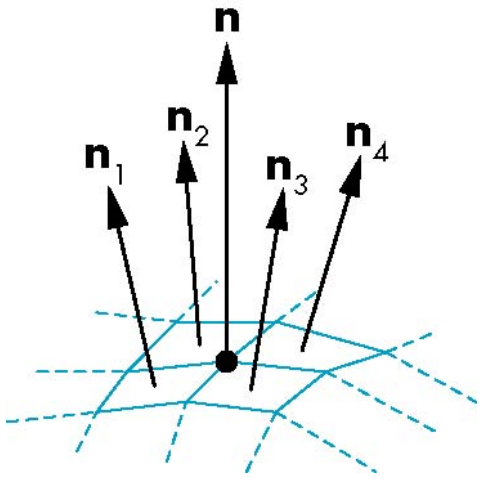
$$\mathbf{L} + \mathbf{R} = (2\ \mathbf{L}\bullet\mathbf{N})\ \mathbf{N}$$

$$\mathbf{R} = (2\ \mathbf{L}\bullet\mathbf{N})\ \mathbf{N} - \mathbf{L}$$



Gouraud and Phong Shading

- Gouraud Shading:
  - Find average normal at each vertex, i.e. average the normals for all the polygons that share a vertex to calculate normal at that vertex.
  - Calculate intensity at each vertex.
  - Interpolate vertex intensitiess across each polygon.
- Phong shading:
  - Find average normal at each vertex, i.e. average the normals for all the polygons that share a vertex to calculate normal at that vertex..
  - Interpolate normals across edges.
  - Interpolate normals across polygons
  - Calculate intensities at each point of a polygon.



OpenGL Shading

OpenGL shading functions.

- Flat.
- Smooth.
- Gouraud.

Steps in OpenGL shading

1. Enable shading and select model.
2. Specify normals.
3. Specify material properties.
4. Specify lights.

Shading Polygons

Shading calculations are done for each vertex

Vertex shades are interpolated across the polygon:

`glShadeModel(GL_SMOOTH)`

`GL_SMOOTH` is the default value

`glShadeModel(GL_FLAT)`

Color at the first vertex will determine the shade of the entire polygon.

OpenGL Normals

In OpenGL the normal vector is part of the state.

```
glNormal3f()
glNormal3fv()
```

Normal vectors should have unit length
Length of vectors can be affected by transformations.
Use glEnable(GL_NORMALIZE) for auto-normalization (but it has performance penalty).

Enable Lighting

- Lighting calculations are enabled by:
  - **`glEnable(GL_LIGHTING)`**
  - Once lighting is enabled, glColor() is ignored.

- Enable each light source individually:
  - **`glEnable(GL_LIGHTi)`** i=0,1….10

Lighting Model

`glLightModel`$_i$`(parameter, GL_TRUE)`

- **`GL_LIGHT_MODEL_LOCAL_VIEWER`**

  specifies how specular reflection angles are computed. If 0 (or 0.0), specular reflection angles take the view direction to be parallel to and in the direction of the -z axis, regardless of the location of the vertex in eye coordinates. Otherwise, specular reflections are computed from the origin of the eye coordinate system. The default is 0.

- **`GL_LIGHT_MODEL_TWO_SIDED`**

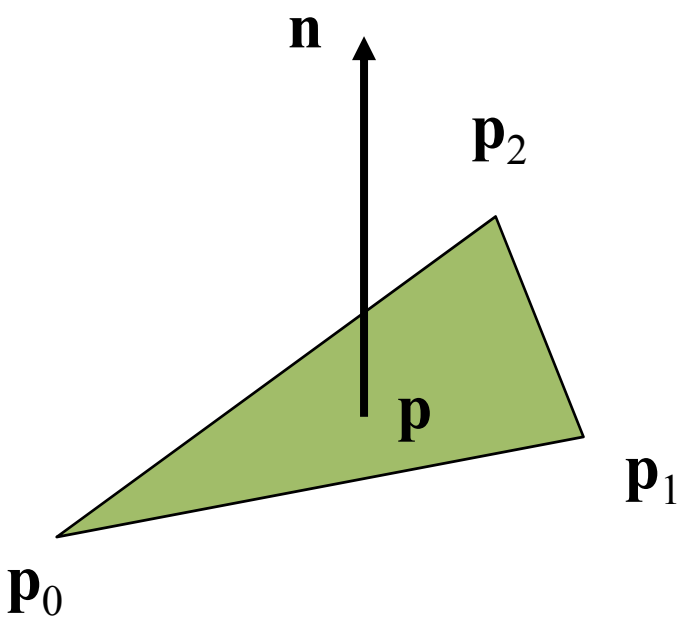
  specifies whether one-sided or two-sided lighting calculations are done for polygons. It has no effect on the lighting calculations for points, lines, or bitmaps. If param is 0 (or 0.0), one-sided lighting is specified, and only the front material parameters are used in the lighting equation. Otherwise, two-sided lighting is specified. In this case, vertices of back-facing polygons are lighted using the back material parameters, and have their normals reversed before the lighting equation is evaluated. Vertices of front-facing polygons are always lighted using the front material parameters, with no change to their normals. The default is 0.

Calculating Normals

For a triangle

$$\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)$$

normalize $\mathbf{n} \leftarrow \mathbf{n}/|\mathbf{n}|$



Be careful about direction of normals (use right-hand rule)

Lighting

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x*, *y*, *z*, *w*) position of light
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x*, *y*, *z*) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cutoff angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

Lighting

- `glEnable(GL_LIGHTING)`
- `glEnable(GL_LIGHT0)`
- `glLightv(GL_LIGHT0, GL_POSITION, light0_pos)`
- `glLightv(GL_LIGHT0, GL_AMBIENT, ambient0)`
- `glLightv(GL_LIGHT0, GL_DIFFUSE, diffuse0)`
- `glLightv(GL_LIGHT0, GL_SPECULAR, specular0)`
- The colors are specified in RGBA.
- The position is given in homogeneous coordinates:
  - If w =1.0, specifying a finite location.
  - If w =0.0, specifying a parallel source with the given direction vector.

Attenuation

- `glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a)`
- `glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b)`
- `glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c)`

$$f_{att} = \frac{1}{a + bd + cd^2}$$

- Default values are:
- a=1.0 (constant terms)
- b=0 (linear term)
- c=0 (quadratic terms).

Material

- `glMaterialf(GL_FRONT, GL_AMBIENT, ambient)`
- `glMaterialf(GL_FRONT, GL_DIFFUSE, diffuse)`
- `glMaterialf(GL_FRONT, GL_SPECULAR, specular)`
- `glMaterialf(GL_FRONT, GL_SHININESS, shine)`
- `glMaterialf(GL_FRONT, GL_EMISSION, emission)`

- Material properties are specified as RGBA values.
- The A value can be used to make the surface translucent.
- The default is that all surfaces are opaque regardless of A.