Computer Graphics

-Local Shading and Illumination-

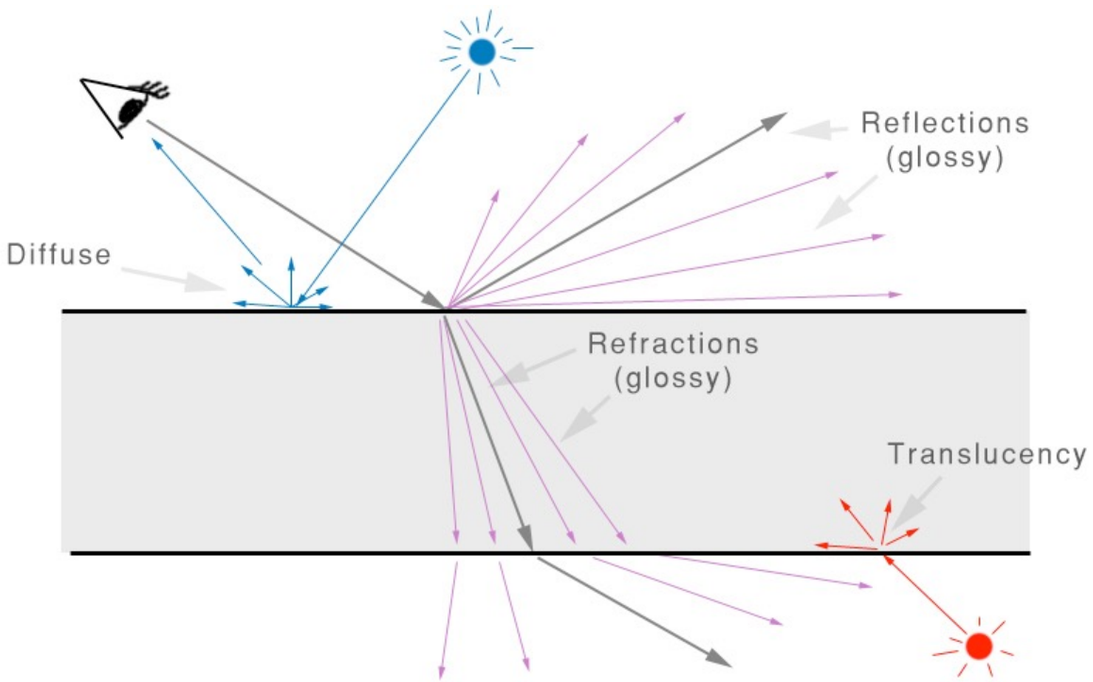
Oliver Bimber

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	01.03.2016	13:45-15:15	HS 18	Introduction and Course Overview	Conference
C2	15.03.2016	13:45-15:15	HS 18	Transformations and Projections	Easter Break
C3	05.04.2016	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	12.04.2016	13:45-15:15	HS 18	Local Shading and Illumination	
C5	19.04.2016	13:45-15:15	HS 18	Texture Mapping Basics	
C6	26.4.2016	13:45-15:15	HS 18	Advanced Texture Mapping & Graphics Pipelines	
C7	03.05.2016	13:45-15:15	HS 18	Intermediate Exam	
C8	09.05.2016	17:15-18:45	HS 18	Global Illumination I: Raytracing	
C9	10.05.2016	13:45-15:15	HS 18	Global Illumination II: Radiosity	Conference / Holiday
C10	31.05.2016	13:45-15:15	HS 18	Volume Rendering	
C11	07.06.2016	13:45-15:15	HS 18	Scientific Data Visualization	
C12	14.06.2016	13:45-15:15	HS 18	Curves and Surfaces	
C13	21.06.2016	13:45-15:15	HS 18	Basics of Animation	
C14	28.06.2016	13:45-15:15	HS 18	Final Exam	
C15	04.10.2016	13:45-15:15	TBA	Retry Exam	

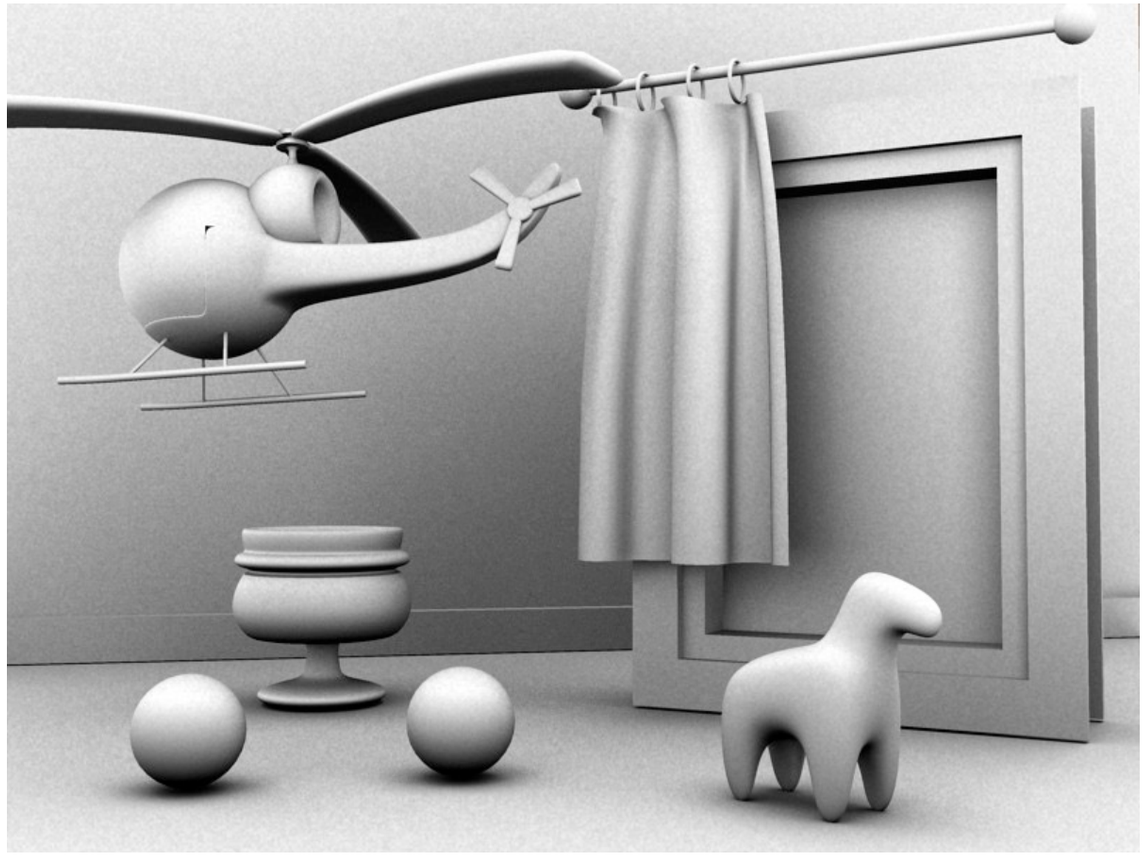
Surface Shading

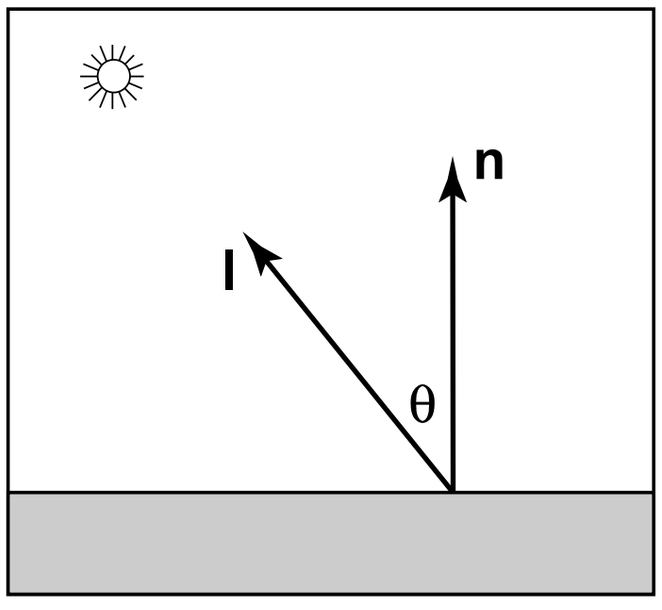
- So far, transformed and projected objects can be rendered, hidden surfaces can be removed, and visible surfaces can be colored
- How about simulating light sources?
- In general, this is called shading, or surface shading
- Different lighting/illumination models exist
  - Local/direct lighting: each surface point is shaded only based on the direct influence of the light sources
  - Global/indirect lighting: considers also indirect effects, such as surface inter-reflections, etc.



Lambertian Shading Model

- Many objects in real world have a matte surface appearance
- They can be considered to behave „Lambertian“ (after Johann Heinrich Lambert)
- Lambert‘s law states that the shading (c) of a surface point is proportional to the cosine of the angle between surface normal (n) and direction of the light source (l)
- This is assumption is only valid if the light source is relatively far away from the surface (optimally at infinity, since only in this case all light rays have the same direction)
- If a light source is infinitely far away and can be described only by its direction, this is called a directional light
- If a light source has a position, it is normally a point light
- A point light with restricted solid angle is called spot light

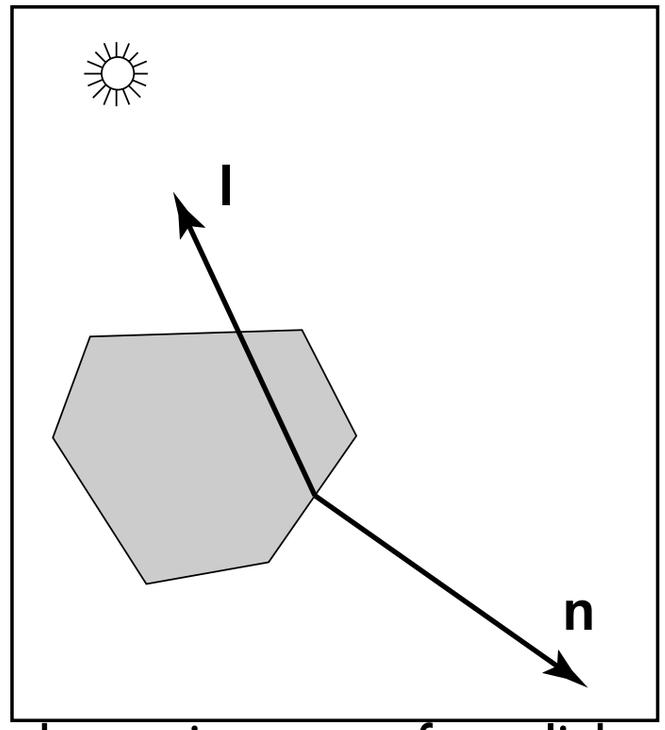




$$c \propto cos(\theta) = |n \cdot l|$$

Lambertian Shading Model

- Surfaces that point away from light source should not receive light
- Thus, either the angle (+/- 90deg) or the dot product of l and n (must be positive) should be checked
- Modulation of light is linear
  - The color of the surface material (c_r) and the color of the light source (c_l) should be considered through additional multiplications (e.g., in RGB)
  - Multiple (N) light sources (i) can be considered through summation (clipping must be handled if HDR is not possible)



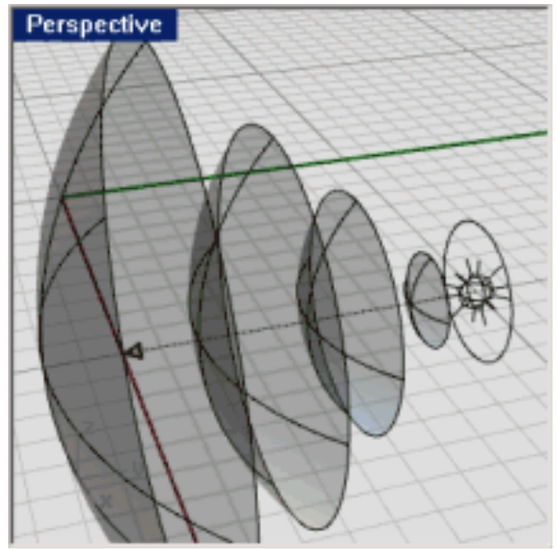
surface that points away from light source
(nl is negative, c=0)

$$c \propto c_r c_l \left| n \cdot l \right|$$

$$c \propto c_r \sum_{i}^{N} c_{l_i} \left| n \cdot l_i \right|$$

Note on Lambertian Model

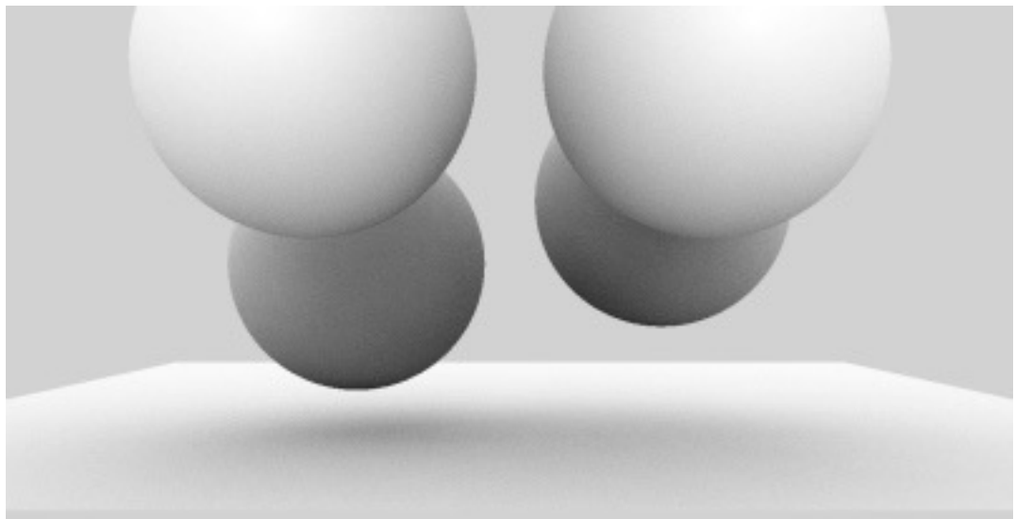
- In reality, shading on diffuse surfaces does not only depend on the angle between normal and light direction
- Light intensity also drops off proportional to the distance (r) between light source and surface in a quadratic manor
- The same amount of light energy that is emitted within a specific solid angle is distributed over a larger surface when being further away from the light source
- Hence, the amount of light energy that hits a single unit area at a distance is actually smaller
- This is called square-distance attenuation
- In CG, people don‘t like to use it, because it causes quick shading drop-offs (which don‘t look good)



$$c \propto \frac{cos(\theta)}{r^2} = \frac{|n \cdot l|}{r^2}$$

Ambient Component

- One problem with this shading model is, that surfaces that point away from light sources appear black (c=0)
- A solution to address this, is to introduce an ambient shading term (c_a)
- This is just a constant color being added to avoid that c=0
- It approximates the constant environment light that results from inter-reflections and scattering at other surfaces
- How would you approximate this quickly?

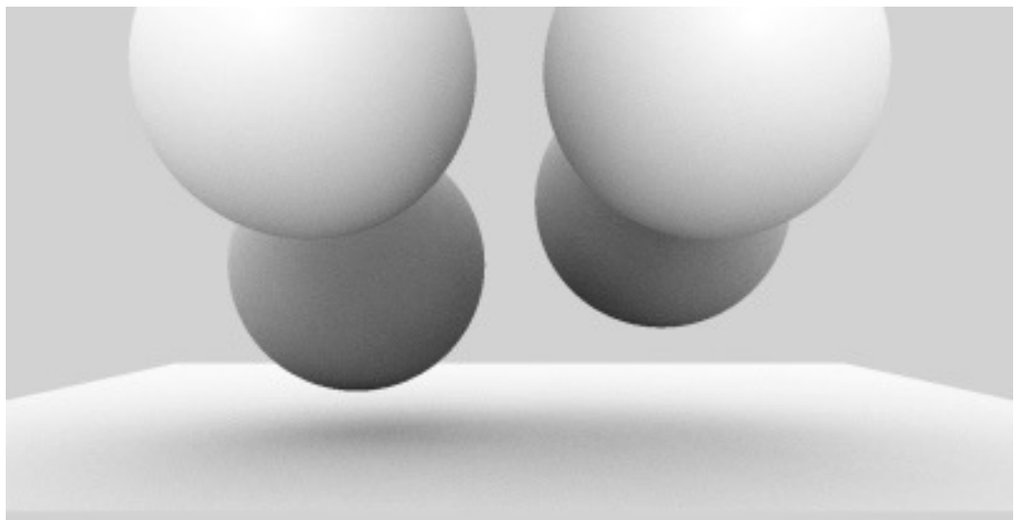


with ambient term

$$c \propto c_r\left(c_a + \sum_i^N c_{l_i} \left|n \cdot l_i\right|\right)$$

Ambient Component

- One problem with this shading model is, that surfaces that point away from light sources appear black (c=0)
- A solution to address this, is to introduce an ambient shading term (c_a)
- This is just a constant color being added to avoid that c=0
- It approximates the constant environment light that results from inter-reflections and scattering at other surfaces
- How would you approximate this quickly?
- Intuitively, you can derive it by computing the average color of all surfaces

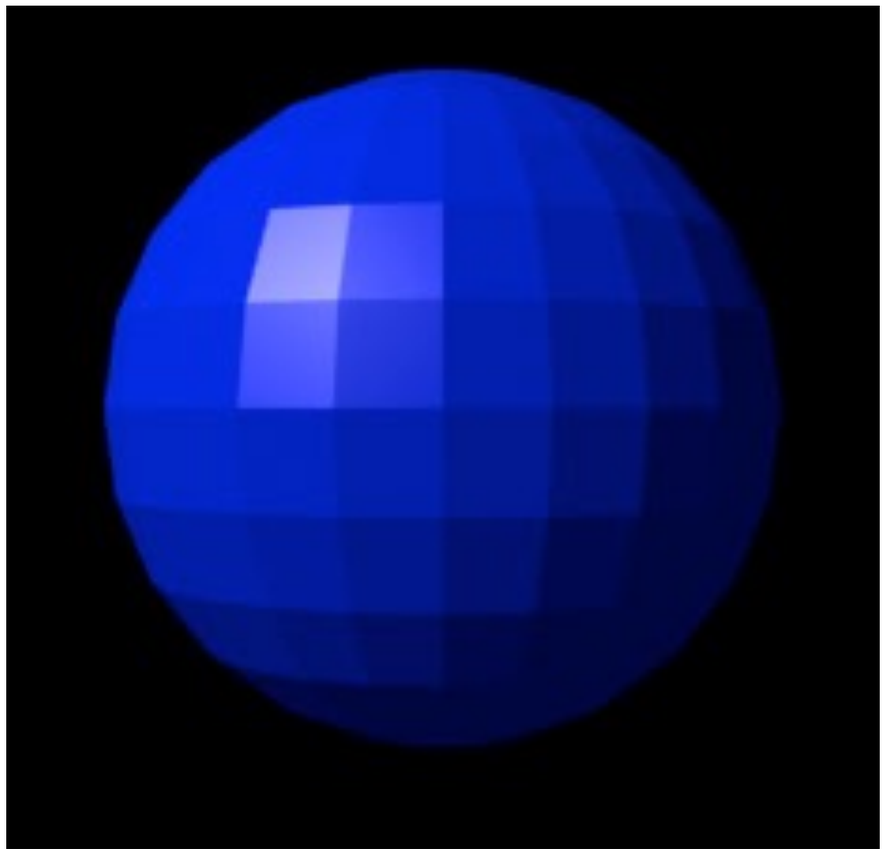


with ambient term

$$c \propto c_r \left(c_a + \sum_i^N c_{l_i} \left| n \cdot l_i \right| \right)$$

Flat Shading

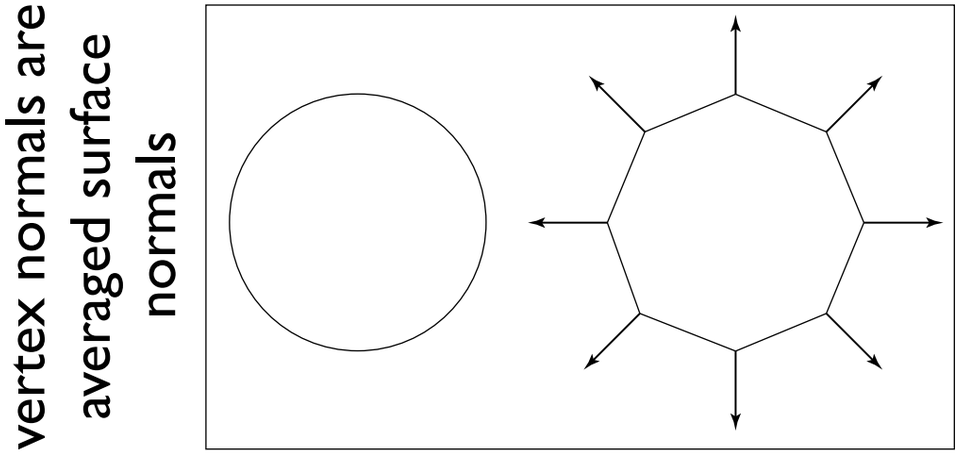
- To what do we apply our shading equation - surface (i.e., triangles), vertices, pixels?
- Computing c for each triangle (using its normal) and applying it to every pixel during rasterization is called flat shading
- A lot of triangles are needed to make the surface appear smooth
- A high geometric resolution will also lead to a loss in performance
- Can we do better?



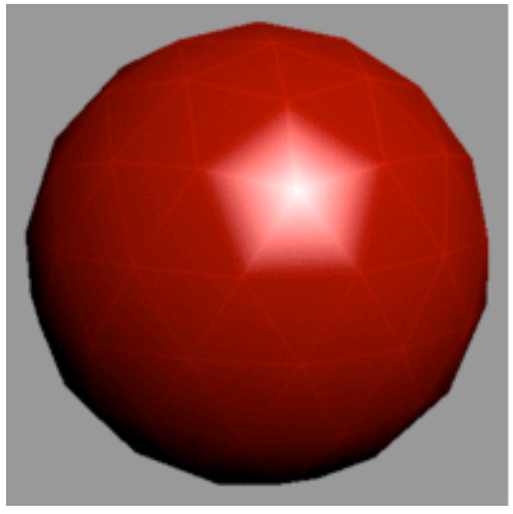
flat shading

Gouraud Shading

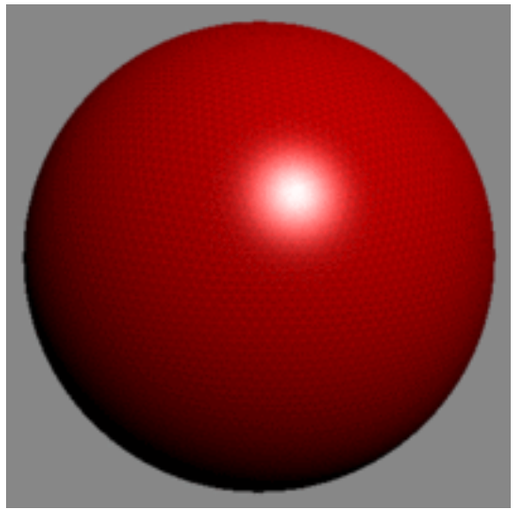
- Named after Henri Gouraud
- Use vertex normal (average of attached surface normals) for computing c for each vertex
- Interpolate (barycentric) vertex colors (as discussed in depth-handling and rasterization class) together with other parameters, such as depth (z) values
- The result is smoother than with flat shading, but still not good
- Increasing the geometric resolution gives better results - but again on the cost of performance
- Can we do better?



vertex normals are averaged surface normals



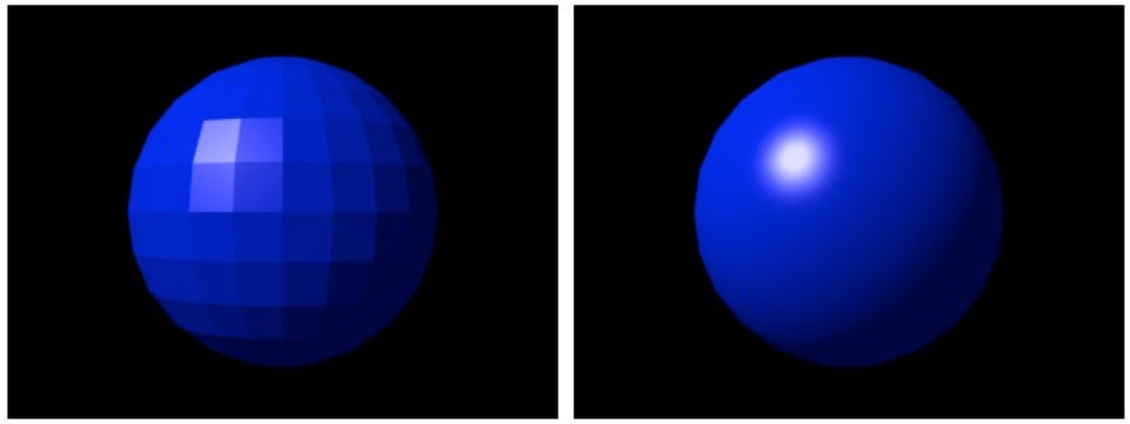
Gouraud shading with low geometry resolution



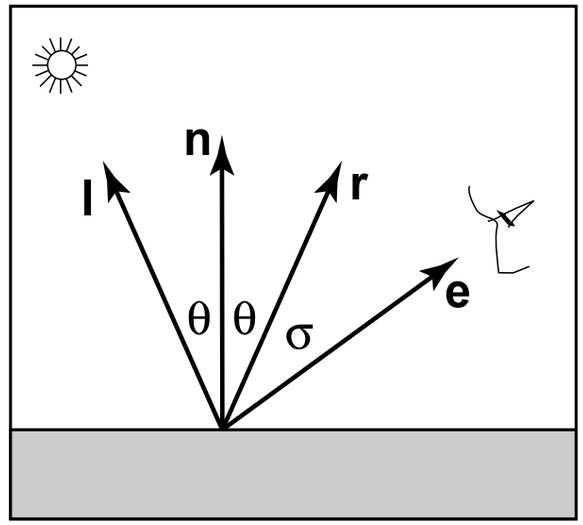
Gouraud shading with high geometry resolution

Phong Shading

- Named after Bui Tuong Phong
- So far, we only get diffuse reflections - but how about specular reflections for a smoother appearance?
- The Phong model approximates specular highlights on the surface by considering also the direction vector to the viewer (e) and the natural direction of the reflection (r)
- Thus, c is bright when e=r and falls of gradually if sigma becomes larger
- Yet, r*e can be negative (which must be handled)



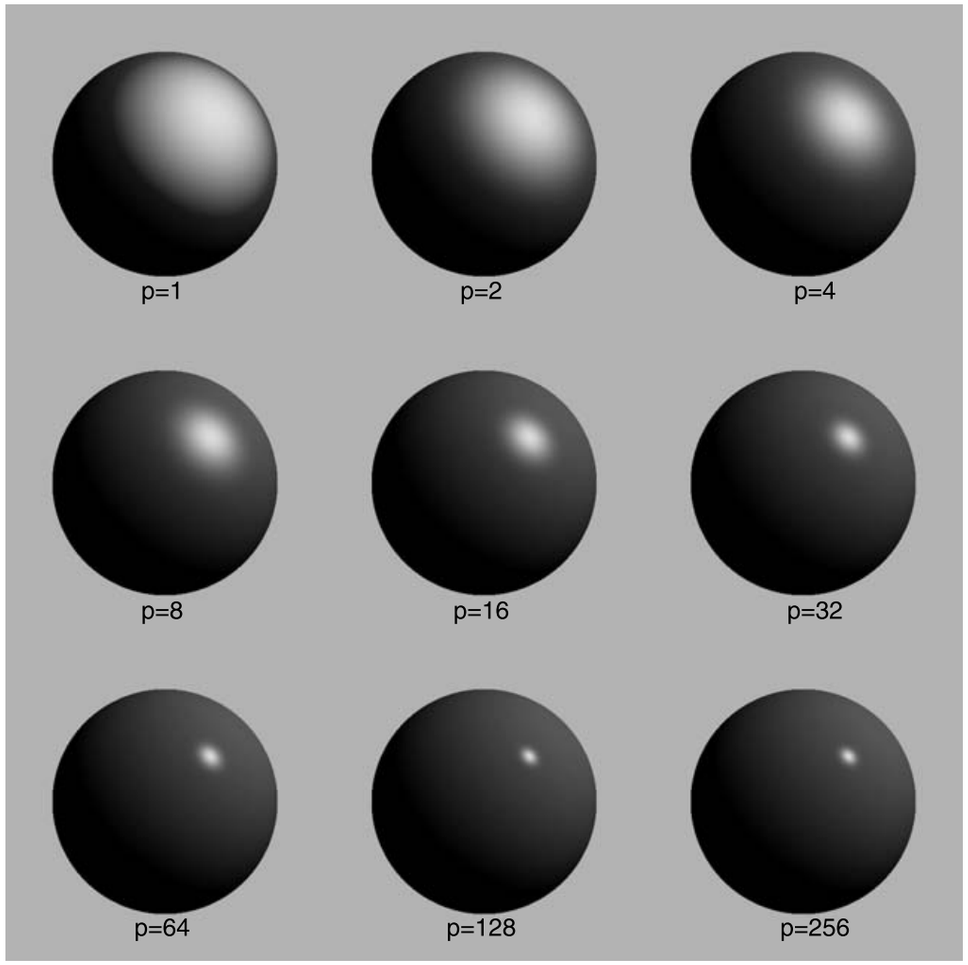
FLAT SHADING PHONG SHADING



$$c \propto c_r c_l (r \cdot e) = c_r c_l (cos(\sigma))$$

Phong Shading

- However, the specular highlights are unrealistically wide if the equation is applied as described before
- This can be handled by introducing the Phong exponent (p)
- Small p values result in wide specular highlights (e.g., for rough surfaces), and large p values result in small highlights (e.g., for smooth surfaces)
- But r*e must not be negative in this case!

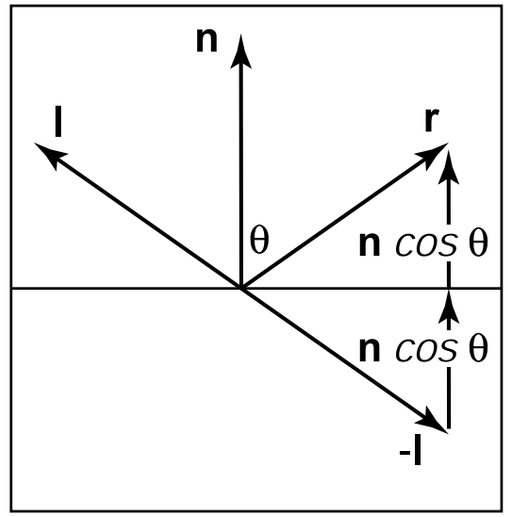


shadings with different Phong exponents

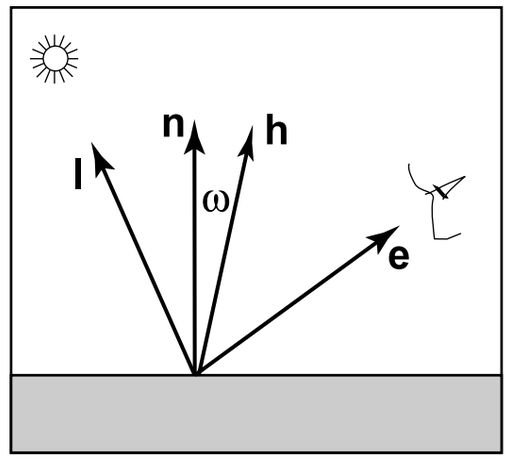
$$c \propto c_r c_l (r \cdot e)^p$$

Phong Shading

- How do we compute the vector r from given vectors l and n?
- There is a heuristic approximation to this, that does not require to check negative values (it is therefore faster)
- Instead of computing the reflection vector r, we compute the half-vector h (halfway between l and e)
- Highlights occur when h is near n
- The advantage is, that h*n is always positive (but sqrt and div still have to be computed)



$$r = -l + 2(l \cdot n)n$$



$$h = \frac{e + l}{\|e + l\|}$$

$$c \propto c_r c_l (h \cdot n)^p = c_r c_l cos(\omega)^p$$

Phong Vertex Interpolation

- If c is computed for vertices and then interpolated during rasterization, artifacts can still appear if the geometry resolution is too low
- If too high, the performance drops
- Can we still do better?

$$\alpha = (1 - \beta - \gamma)$$

$$p = \alpha p_0 + \beta p_1 + \gamma p_2$$

interpolation with barycentric coordinates for vertex positions

$$c = \alpha c_0 + \beta c_1 + \gamma c_2$$

interpolation with barycentric coordinates for vertex colors

Phong Normal Interpolation

- If c is computed for vertices and then interpolated during rasterization, artifacts can still appear if the geometry resolution is too low
- If too high, the performance drops
- Can we still do better?
- Solution: instead of interpolating colors, we interpolate normal vectors during rasterization and apply a shading model (in this case Phong shading) at each rastered pixel
- This is then called Phong normal interpolation (the standard for most CG systems)

$$\alpha = (1 - \beta - \gamma)$$

$$p = \alpha p_0 + \beta p_1 + \gamma p_2$$

interpolation with barycentric coordinates for vertex positions

$$c = \alpha c_0 + \beta c_1 + \gamma c_2$$

interpolation with barycentric coordinates for vertex colors

$$n = \alpha n_0 + \beta n_1 + \gamma n_2$$

interpolation with barycentric coordinates for vertex normals

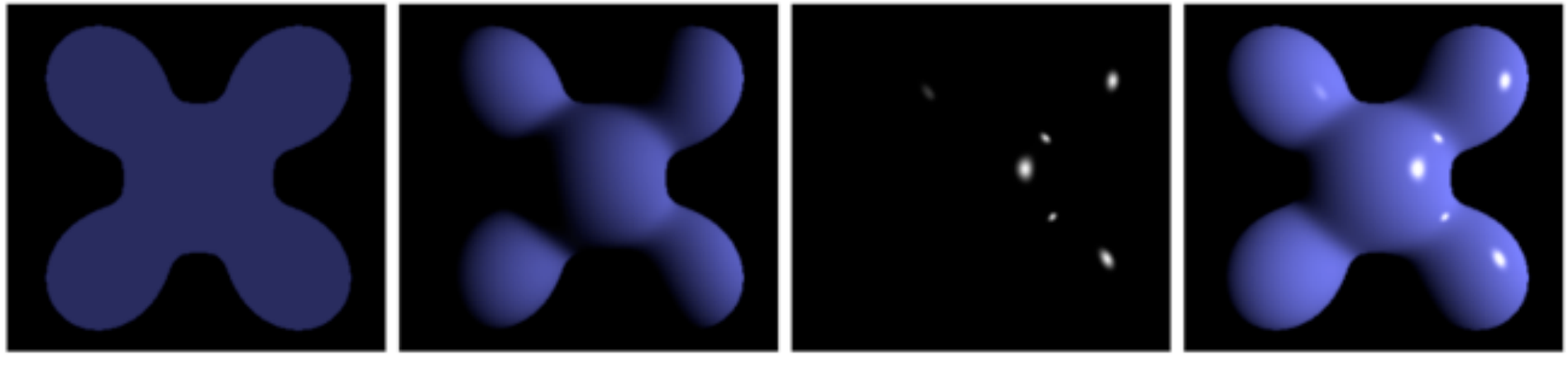
Putting It All Together

- Common real-time CG systems apply three components for local shading: ambient (c_a), diffuse (c_d) and specular (c_s,p)
- They are normally defined as material parameters of the surface (together with the surface reflectance c_r)
- Then we have multiple (N) light sources (i) with individual colors (c_l)
- And we know that modulation of light is always linear

surface reflectance light color

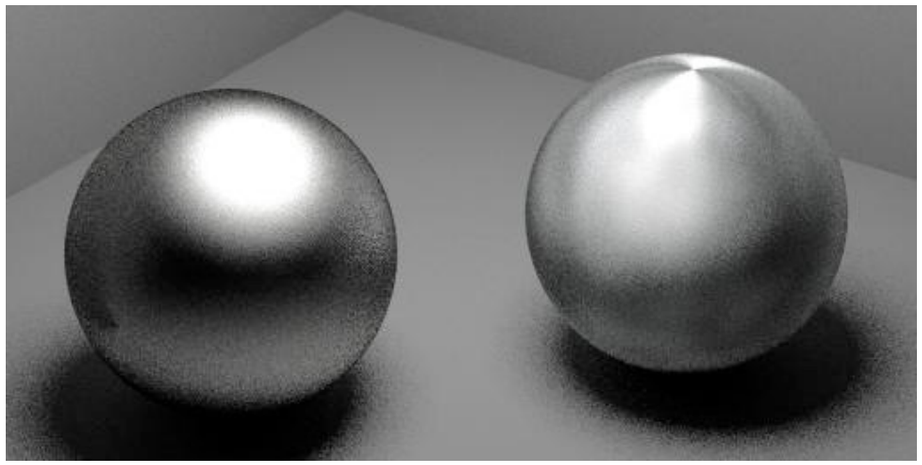
$$c = c_r(c_a + \sum_{i}^{N} c_{l_i}(c_d \left|n \cdot l_i\right| + c_s(h_i \cdot n)^p))$$

ambient diffuse specular



More Complex Materials

- Most materials reflect light in a much more complex way
- Think about polished metal or valvet
- Isotropic material reflect light at least symmetrically around their normal

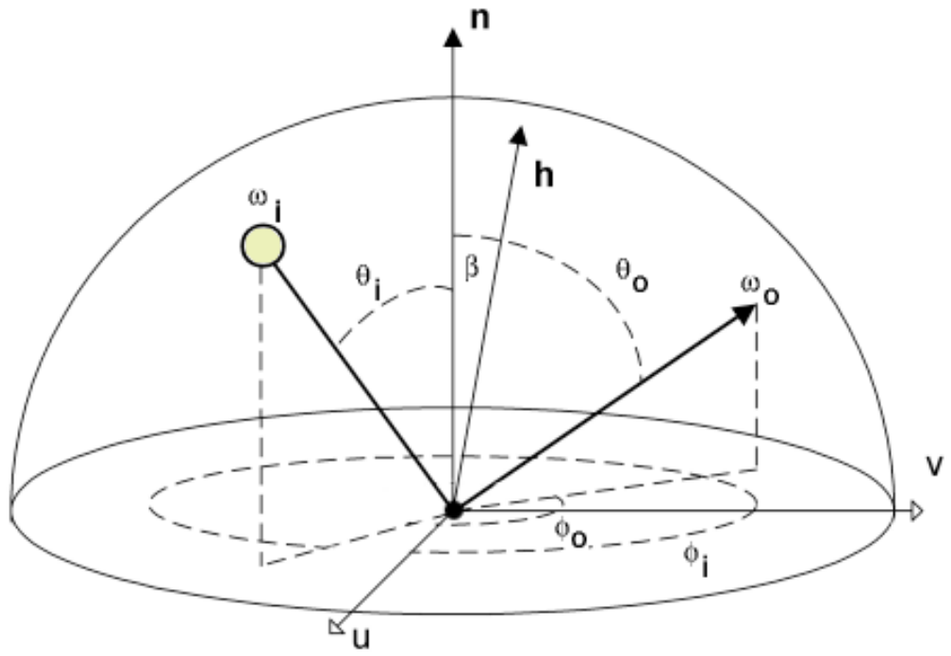


isotropic anisotropic

- Anisotropic materials can reflect light differently with respect to the light direction and viewing direction
- So far, we have described materials only with an RGB,a,p vector, and computed the reflected light with a very simple approximation
- Can we do any better?

BRDF Theory

- BRDF stands for bidirectional reflectance distribution function
- It describes material properties
- More specifically, it describes how light is reflected towards a given outgoing direction (o) with respect to a given input direction (i) and its wavelength (RGB colors)
- It is a 4D function, correlating incoming and outgoing light over two elevation and azimut angles
- The result is a unitless value that is relative to the amount of energy being reflected
- It does not describe how materials physically interact or modulate light

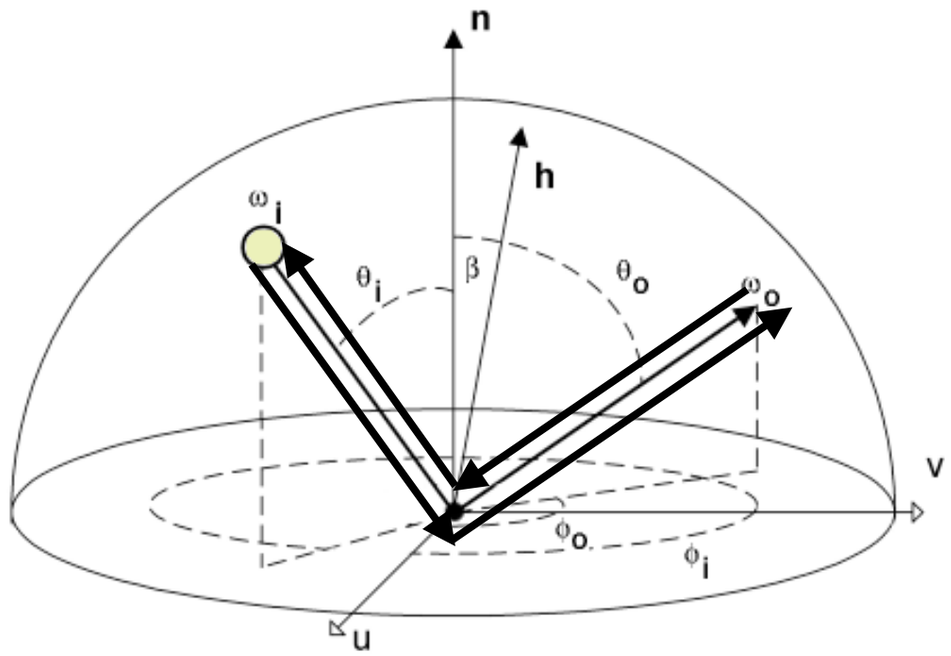


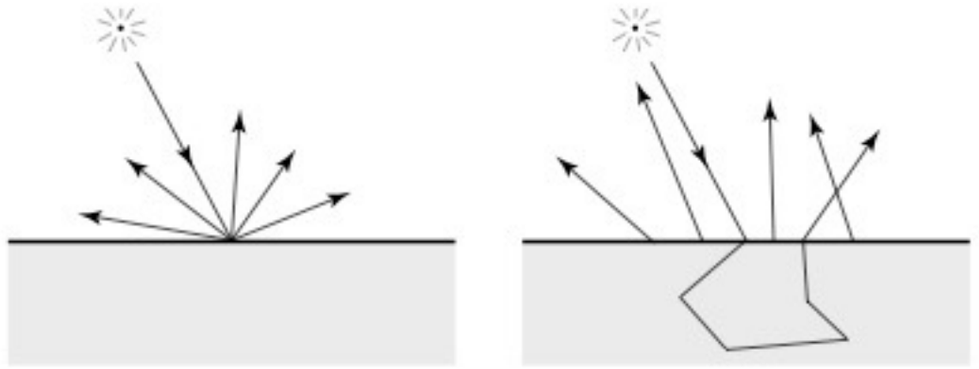
BRDF

$$f(\theta_o,\phi_o,\theta_i,\phi_i) = \frac{L(\theta_o,\phi_o)}{L(\theta_i,\phi_i)\cos(\theta_i)d\theta_i d\phi_i}$$

BRDF Theory

- BRDFs have a few important properties
  - input and output directions are dual (Helmholtz reciprocity)
  - BRDFs must always be normalized (total amount of outgoing energy must always be less or equal to the amount of incoming energy)
  - It describes how light is reflected at one surface point (light is reflected at the same point where incoming light arrives)
  - It does not describe subsurface scattering (bidirectional surface scattering reflectance distribution function, BSSRDF, supports this)
  - BRDFs and BSSRDFs do not describe transmission of light (BTDF do this, T stands for transmittance), BRDF +BTDF for both directions = BSDF (S stands for scattering)





BRDF BSSRDF

Reflectance Equation

- Given a BRDF f and an incoming radiance distribution, the reflectance distribution function determines the outgoing radiance for a given viewing direction

$$L(\theta_o,\phi_o) = \int_0^{2\pi}\int_0^{\pi/2} f(\theta_o,\phi_o,\theta_i,\phi_i)L(\theta_i,\phi_i)\cos(\theta_i)d\theta_i d\phi_i$$

- For a surface point, the incoming radiance has to be integrated over a hemisphere
- The equation is evaluated separately for all color channels
- For a single point light source, it can be simplified

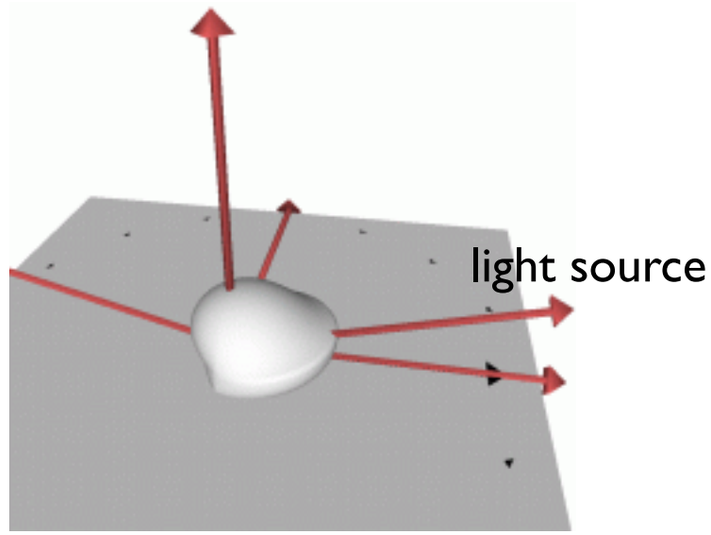
$$L(\theta_o,\phi_o) = f(\theta_o,\phi_o,\theta_i,\phi_i)L(\theta_i,\phi_i)\cos(\theta_i)$$

- and simplified in vector form

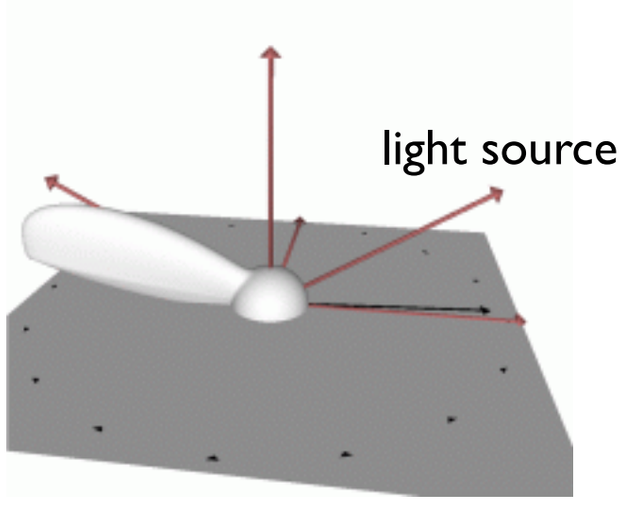
$$L(\omega_o) = f(\omega_o,\omega_i)L(\omega_i)(n\cdot\omega_i)$$

Reflectance Lobes

- One way to understand and visualize BRDFs is to hold the incoming direction constant and display the distribution of energy in all output directions
- These visualizations are called reflectance lobes
- The spherical part around the point describes the diffuse component
- The ellipsoid part describes the specular component
- Reflectance lobes can also be modeled do describe surface reflectance



example for diffuse surface



example for specular surface

Different Light Types

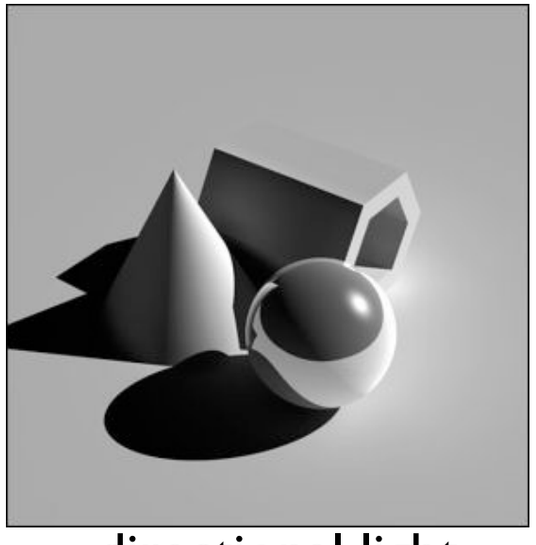
- We talked only about directional lights (only direction is known), point lights (infinitely small light sources with positions, radiating in all directions), and spot lights (point lights with limited solid angle)

- But off course, other light source types are used in CG, such as spherical, area and environment light

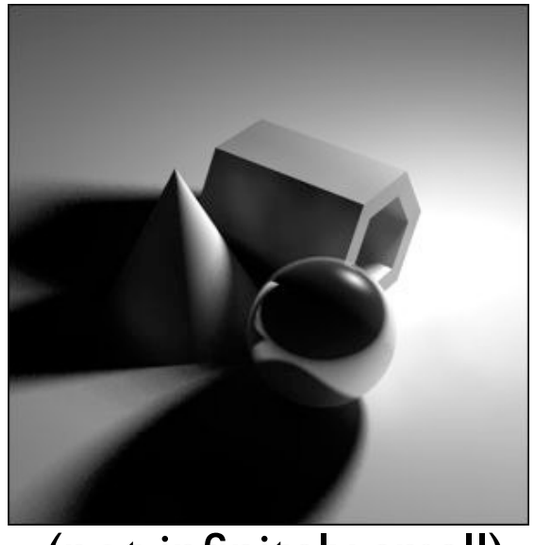
- The larger the light source area, the softer the rendering (shadows and shadings)

- By the way - we did not talk about shadows yet (this will be a topic next time)

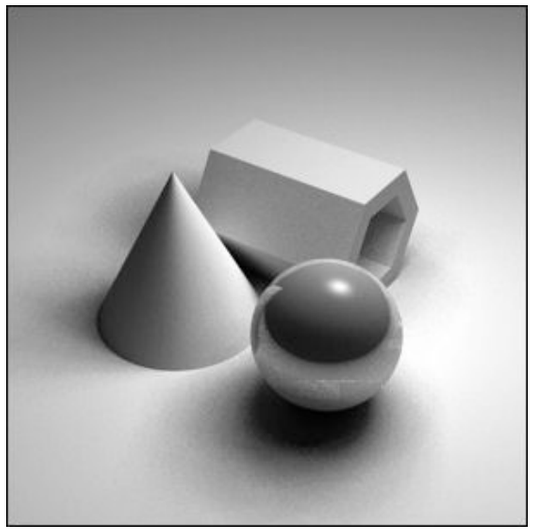
- There are other forms of shading that consider other parameters than just light sources



directional light



(not infinitely small) spherical light



area light

environment light

Non-Photorealistic Rendering

- The shading model that we discussed so far, aims at imitating real-world appearance (although it is by far not physically correct)

- Other shading techniques exist that try to communicate information visually with less clutter - and are therefore easy to read

- Examples are shading techniques that aim at simulating drawings, toons, paintings or sketches as by artists

- This is also referred to as non-photorealistic rendering (NPR)

Global Illumination Outlook

- We did talk about local shading (local illumination) - i.e., how we compute a pixel‘s color only based on the direct effect of light sources
- We did not talk about how more complicated shading effects are created, such as texture, shadows, reflection
  - This will be covered in the next two class
- We also did not talk about how more complex global shading illumination effects are created, such as inter-reflections, caustics, refraction, etc.
  - This will be covered in the two classes afterwards

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	01.03.2016	13:45-15:15	HS 18	Introduction and Course Overview	Conference
C2	15.03.2016	13:45-15:15	HS 18	Transformations and Projections	Easter Break
C3	05.04.2016	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	12.04.2016	13:45-15:15	HS 18	Local Shading and Illumination	
C5	19.04.2016	13:45-15:15	HS 18	Texture Mapping Basics	
C6	26.4.2016	13:45-15:15	HS 18	Advanced Texture Mapping & Graphics Pipelines	
C7	03.05.2016	13:45-15:15	HS 18	Intermediate Exam	
C8	09.05.2016	17:15-18:45	HS 18	Global Illumination I: Raytracing	
C9	10.05.2016	13:45-15:15	HS 18	Global Illumination II: Radiosity	Conference / Holiday
C10	31.05.2016	13:45-15:15	HS 18	Volume Rendering	
C11	07.06.2016	13:45-15:15	HS 18	Scientific Data Visualization	
C12	14.06.2016	13:45-15:15	HS 18	Curves and Surfaces	
C13	21.06.2016	13:45-15:15	HS 18	Basics of Animation	
C14	28.06.2016	13:45-15:15	HS 18	Final Exam	
C15	04.10.2016	13:45-15:15	TBA	Retry Exam	

Thank You!