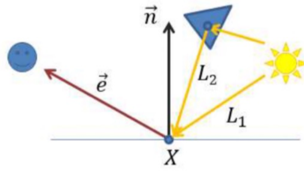
Lighting

Illumination Models: When lighting a point in a surface, typically a pixel, we must consider the point being lit and the light sources. When we consider specific elements we use a <u>local model</u> with direct illumination. In this model light travels from the light sources and when it hits a surface a part is absorbed, some reflected and in case of transparent or translucent objects, part is transmitted. The transmitted and reflected light keeps traveling until it hits another object. Each time a surface is hit part of the energy is absorbed, so the energy keeps decreasing with every bounce until it does not contribute to the final shading. This makes every object to be considered as a light source. Furthermore, objects may occlude each other (shadows may exist). When we consider this broader notion of light source as well as occlusions, the illumination model is <u>global</u>.

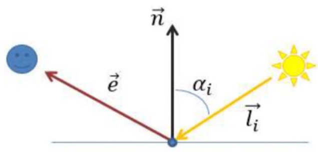


- <u>Local lighting</u>: computed based soley on the light sources, and the point being lit. It only takes into account direct illumination
- <u>Global lighting</u>: computation takes account indirect illumination, light that is reflected or transmitted from other sources, and occlusions.

Using Global illumination we can add effects such as shadows, caustics, color bleeding, reflections and refractions, amongst others. This carries a penalty when compared with local lighting, taking longer to render the final image.

Equation: (local) the intensity perceived when looking at a lit object is a function of the surface's properties, the light it receives and the camera position. The values for this function can be measured with special hardware and/or encoded in a function called a Bi-directional Reflectance Distribution Function (BRDF).

- there are materials that reflect light in a uniform way, regardless of the outgoing direction. The intensity of the reflected light is only a function of the intensity of the incoming light (I), the angle of the light makes with the normal vector (perpendicular to the surface) and a constant term that determines the diffuse color of the object (K_d).



$$\boxed{I = \textstyle\sum_i K_d \times I_i \times \cos(\alpha_i)}$$

This provides a local illumination solution to diffuse materials and is based on Lambert's law, which states that the intensity reflected by a purely diffuse material is proportional to the cosine of the angle between the surface normal and the incoming light direction. Only angles below 90° are considered. For lights below the surface the result is 0. The cosine computation $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos(\alpha)$

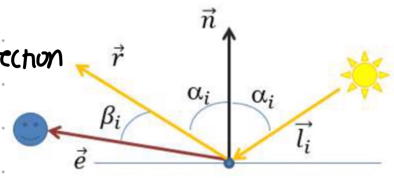
Here, if both vectors are normalized, the dot product becomes an efficient way to compute the cosine of the angle between them.

- Points on the surface facing away from the light will not be lit, producing black as the final color. In a global model these could receive indirect lighting. To simulate this effect, an ambient term is added. This should add a small amount of light to every point on the surface, regardless of lighting.

The new equation has 2 terms for the diffuse and ambient components and 2 new variables K_a and I_a. The first relates to the ambient term of the material itself whereas the second relates to light i contribution to the ambient illumination.

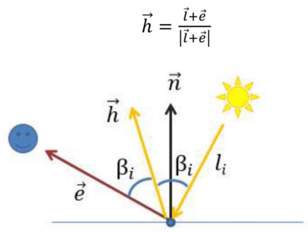
$$\boxed{I = \sum_i (K_a \times I_{a_i} + K_d \times I_i \times \cos(\alpha))}$$

- It was also introduced a term to represent specular lights. This term is dependent not only on the incoming light's direction, but also on the viewer's position. It has its maximum in the direction $\vec{r}$ that is the light direction reflection vector regarding the surface normal $\vec{n}$.



The intensity of the light reflected on the viewer's direction (w) is computed as the intensity of the incoming light weighted with the cosine β raised to a term called shininess. The light equation becomes:

$$\boxed{I = \sum_i (K_a \times I_{a_i} + K_d \times I_i \times \cos(\alpha_i) + K_s \times I_i \times \cos(\beta_i)^{shininess})}$$



- The equation above implies the computation of the reflection vector $\vec{r}$. We can use the half-vector instead of the reflection vector. The half vector is a vector that is halfway between the incoming's light direction ($\vec{l}$) and the viewer's direction ($\vec{e}$). It can easily be computed as the normalized sum of $-\vec{l}$ and $\vec{e}$. The formula becomes:

$$\boxed{I = \sum_i (K_a \times I_{a_i} + K_d \times I_i \times \cos(\alpha_i) + K_s \times I_i \times \cos(\theta_i)^{shininess})}$$

- Combining the 3 components:

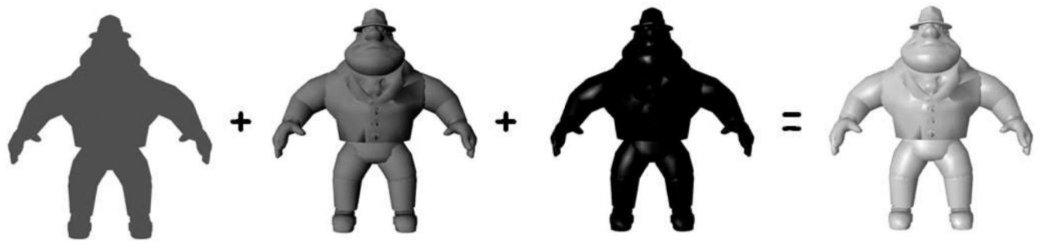


Figure 3.7 Ambient + diffuse + specular = final result

Normals: Considering the 3 vertices of a triangle, P_1, P_2 and P_3 we can build 2 vectors $\vec{v_1} = P_2 - P_1$ and $\vec{v_2} = P_3 - P_1$. The cross product of these 2 products provides a vector which is perpendicular to both, being perpendicular to the triangle. This normal vector should be normalized to enable the efficient computation of the cosines.

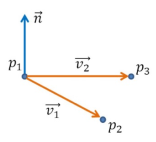


Figure 4.1 – Computing the normal vector

$$\boxed{\vec{n} = \frac{\vec{v_1} \times \vec{v_2}}{|\vec{v_1} \times \vec{v_2}|}}$$

Flat shading: The light equation is run only once per triangle and all its pixels are assigned the same color. The light direction is computed considering a particular point on the triangle, for instance one of its vertices or its center. A single normal per triangle is required. Due to its nature, triangles with different orientations can be clearly distinguished in the final rendered image.

The shading model only makes sense if:

- the light is infinitely distant, such that light rays for every point inside the triangle are parallel, hence arrive with the same direction, implying that the cosine term is the same for every point
- the viewer is also infinitely distant, such that the specular highlight computation provides the same results for every point inside a triangle.

- the triangle is actually an accurate representation of the object

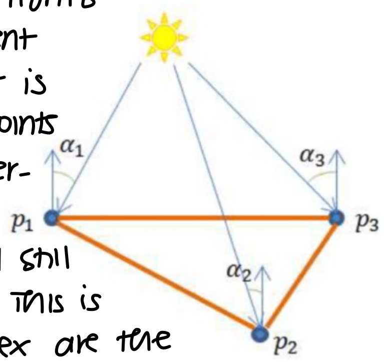
To get a smoother look we could use smaller triangles however this approach has 2 disadvantages:

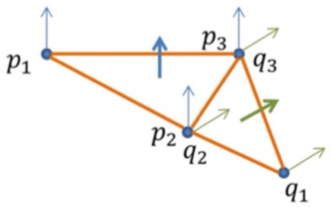
- the higher number of triangles can have a negative impact on performance
- the faceted look doesn't disappear unless we have a triangle per pixel, otherwise the issue can be enhanced due to the Mach band effect.

Figure 4.3 – Mach band effect

This happens because our system accentuates changes in brightness.

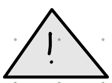
Interpolating shading: the light's intensity should be computed per vertex. The insensity, or color of the points inside the triangle is obtained using interpolation. This addresses the first 2 assumptions of the flat shading model. Since each vertex has a different incoming light's direction. Hence, the intensity will be different for each one when considering a light that is not infinitely distant. Since the color for points inside the triangle is obtained through interpolation, this creates a gradient inside the triangle. However, the rendered image will still show a discontinuity between triangles. This is because the normals used at each vertex are the triangle's normal, and the 2 triangles will each have a different normal at common vertices.





This figure shows 2 triangles $P(p_1, p_2, p_3)$ and $Q(q_1, q_2, q_3)$ sharing an edge. The shared vertices are $p_3 = q_3$ and $p_2 = q_2$. However, the normal is different for each triangle.

Gouraud shading: If we could have at each vertex the normal that matches the normal of the surface we are trying to approximate, as opposed to the triangle's normal the illumination would look as if we had a smoothed curve.
This suggests the usage of blue arrows as normals for each vertex. This implies that, in a triangle, each vertex can have a different normal. On the other hand, vertices shared amongst triangles will have the same normal vector.
If the surface we are approximating is based on a know equation, we can compute the normal analytically. This is not necessarily true for all models. It this case the solution requires that we initially compute the normals of all triangles. Then for each vertex we compute the normal as the average of the normals of all triangles that share the vertex.
Once the normals are available, the intensities are computed at each vertex, and for each point inside the triangle these intensities are interpolated.



There are some issues with this model. For instance, when a light does not reach any of the vertices but reaches the center, the vertices will have 0 intensity but the center will also be black because the interpolation is with 0 values. The interpolation also doesn't take into account the curvature of the surface. The specular lights are also an issue as they are high frequenly in nature.

Phong shading: instead of computing the intensity for each vertex and interpolate these values per pixel, we should interpolate the normal itself for each pixel, and then compute the lighting equation on every pixel, with the interpolated normals.

- for each vertex we compute a normal
- for each pixel interpolate the normal and compute the lighting equation using interpolated normal

SUMMARY

Flat model

Per vertex:

- Single **normal per triangle**
- Computes lighting equation **once per triangle**

Per Pixel:

- Applies the **same color** to all pixels inside the triangle

Interpolation

Per vertex:

- Single **normal per triangle**
- Computes lighting equation **once per vertex**

Per Pixel:

- **Interpolates colors** for pixels inside the triangle

Gouraud

Per vertex:

- **Per vertex normal** (approximation of the underlying surface's normal)
- Computes lighting equation **once per vertex**

Per Pixel:

- **Interpolates colors** for pixels inside the triangle

Phong

Per vertex:

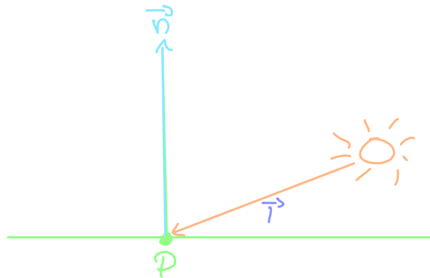
- **Per vertex normal** (approximation of the underlying surface's normal)

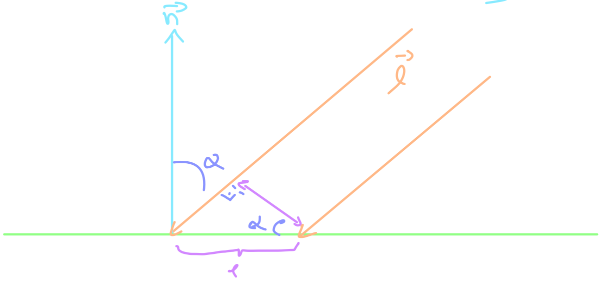
Per Pixel:

- **Interpolates normals** for pixels inside the triangle
- Computes **lighting equation per pixel** with the interpolated normal

Reflexão difusa

⇒ intensidade refletida
↳ direção da luz
↳ intensidade da luz
↳ orientação da superfície em relação à direção da luz





$d = \cos(\alpha)$

Lambert

$$I_d = L_d \cdot K_d \cdot \cos(\alpha)$$

- I_d → intensidade iluminação difusa
- L_d ↳ intensidade da luz difusa incidente
- K_d → cor difusa do objeto
- $\cos(\alpha)$ ↳ ângulo entre a luz e o vetor normal → $\cos(\alpha) = \vec{n} \cdot \vec{l}$ ↳ normalizados

Atenuação

$$I_{final} = f_{at} \times I_d + I_a$$

- I_{final} → intensidade final
- f_{at} ↳ fator de atenuação → $f_{at} = \frac{1}{d^2}$ ↳ distância do objeto
- I_d → intensidade difusa
- I_a → intensidade ambiente

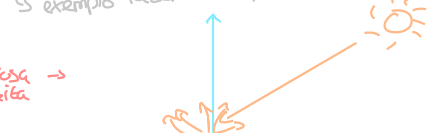
Luz ambiente

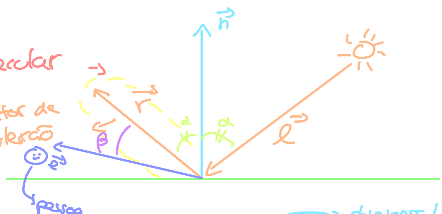
↳ não tem em consideração a relação espacial entre os objetos e a fonte de luz
↳ Todos os objetos são iluminados de forma uniforme independente da sua posição ou orientação
↳ A própria componente de luz não tem posição nem orientação

$$I = \underbrace{I_d \cdot K_d \cdot \cos(\alpha)}_{\text{difusa} \to I_d} + \underbrace{L_a\, K_a}_{\text{ambiente} \to I_a}$$

Espetacular

↳ exemplo laser no espelho



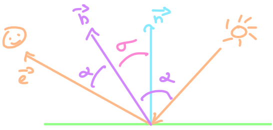


$$I_s = L_s \cdot K_s \cdot \cos(\beta)^S$$

- I_s → iluminação especular
- L_s ↳ luz especular
- K_s → cor especular
- S → shininess / brilho, determina a dimensão da mancha brilhante

$$I = \underbrace{L_a\, K_a}_{I_a} + \underbrace{f_{at}}_{\text{fator de atenuação}} \big(\underbrace{L_d\, K_d \cdot \cos(\alpha)}_{I_d}\big) + \underbrace{L_s\, K_s \cdot \cos(\beta)^S}_{I_s}$$

Blinn - Phong



$$h = \frac{\vec{e} + \vec{\ell}}{|\vec{e} + \vec{\ell}|}$$

$$I_s = L_s\, k_s \cdot \cos(\sigma)^s$$

Modelos de iluminação