Lighting model

Local lighting model

Describes interaction of the light with the surface. Almost never truly based on physics: perception plays a greater role.

Visible light: electromagnetic waves, with wavelengths 400nm (violet) - 700nm (red); intensity can vary over many orders of magnitude.

Computer model: only three “frequencies”: RGB, intensity varies over a small range, typically only 255 discrete values/color.

Physics vs. graphics

Computer graphics terms are somewhat confusing and disagree with physics:

Graphics: color of an object; physics: reflection spectrum (i.e. fraction of light of each frequency that gets reflected).

Graphics: intensity or color of a light ray; physics: radiance distribution (measured in watts/steradian/meter²/meter)

Graphics: intensity or color of a point light source; physics: intensity spectrum of a point light source (almost in agreement!) measured in watts/steradian/meter.

Illumination model

Two main components:

- light source characteristics
  - position
  - intensity for each freq. (color)
    - often, different intensity can be specified for different colors
  - directional distribution

- surface properties
  - reflectance for each freq. (color)
  - different reflectance can be specified for diffuse and specular light

Illumination components

In the model commonly used in graphics applications, there are several components

- diffuse reflection: intensity does not depend on the direction to the viewer
- specular: simulates reflective surfaces and specular highlights depending on the direction to the viewer
- ambient: a crude approximation to the illumination created by the light diffusely reflected from surfaces
**Diffuse component**

Diffuse surfaces are surfaces following the Lambert's law: the energy of the light reflected from a surface in a direction \( D \) is proportional to the cosine of the angle between the normal and \( D \). As intensity (radiance) is proportional to the energy times cross section of the ray, it does not depend on the view direction, but is proportional to the cosine of the angle between the normal and the direction to the light.

\[
L_{\text{diff}} = k_{\text{diff}}(L,N) \cos \theta
\]

**Specular component**

Specular component approximates behavior of shiny surfaces. If a surface is an ideal mirror, the light from a source reaches the eye bouncing of a fixed point of the surface, only if the direction to the light coincides with the reflected direction to the eye:

\[
L = R
\]

**Specular reflection**

For non-ideal reflectors, the reflection of the light is still the brightest when \( L = R \) but decays, rather than disappears, as the angle between \( L \) and \( R \) increases. One way to achieve this effect is to use cosine of the angle to scale the reflected intensity:

\[
L_{\text{spec}} = k_{\text{spec}}(R,L)^p
\]

**Metal vs. plastic**

Natural look of metallic surfaces is difficult to simulate, but the first approximation is obtained using proper highlight color.

For plastic objects, highlights are close in color to the color of the light. For metals, to the color of the surface. Assuming white lights, for plastic set \( k_{\text{spec}} = [c,c,c] \), where \( c \) is a constant, for more metallic look set \( k_{\text{spec}} = k_{\text{diff}} \).

**Ambient component**

Not all light illuminating a surface comes from light sources, or reflections of light sources in ideal mirrors; however, the light diffusely reflected from other surfaces is difficult to take into account, especially for real-time rendering. It is approximated by the ambient component: a constant is added to all objects. To have more control over ambient contribution, surfaces can be assigned ambient reflectivity.

\[
L_{\text{amb}} = k_{\text{amb}} I_{\text{amb}}
\]
**Complete equation**

\[ I_{total} = k_{amb} + \sum_{all\,lights} l_i \left( k_{diff}(L_i, N) + k_{spec}(L_i, R)^p \right) \]

- \( I_{total} \): intensity of total light
- \( k_{amb} \): ambient light coefficient
- \( k_{diff}, k_{spec} \): diffuse and specular coefficients
- \( L_i \): direction of \( i\)-th light
- \( R \): direction of reflectivity

If we are ray tracing for rendering, in summation only visible lights are present, and there are two additional terms: contribution from the reflected ray and transmitted ray. If we are using Z-buffering, then all active light sources are regarded as visible.

**Attenuation**

In real life, radiance reaching us from a light source decreases with distance as \( 1/r^2 \) (the stars are much less bright than the sun). However, due to the nature of approximations used in graphics, the inverse-square law typically results in pictures that are too dark; the fix is to allow the programmer to control how fast the decay is. \( I_i \) in the formulas is replaced not by \( I_i/r^2 \), as physics suggest, but by

\[ I_i = a + b r + br^2 \]

and the most common choice of constants is \( a = 1, b = c = 0 \), that is, no attenuation!

**Snell’s law**

If a surface separates two media with different refraction indices (e.g. air and water) the light rays change direction when they go through.

Snell’s law: the refracted ray is stays in the plane spanned by the normal and the direction of the original ray. The angles between the normal and the rays are related by

\[ n_1 \sin \theta_1 = n_2 \sin \theta_2 \]

where \( n_1 \) and \( n_2 \) are refraction indices.

\[ \theta_1 \] normal

\[ \theta_2 \]

**Reflected ray direction**

\[ T = \omega N + \left( n(V_i) - n(V_i) \cdot N \right) N \]

**Ray tracing**

For each pixel shoot a ray \( R \) from camera;

\[ pixel = TraceRay(R) \]

The recursive ray tracing procedure:

- \( RGBvalue \) \( TraceRay(\text{Ray } R) \)
- shoot rays to all light sources;
- for all visible sources, compute RGB values \( r_s \);
- shoot reflected ray \( R_{refl} \):
  \( r_{refl} = TraceRay(R_{refl}) \)
- shoot refracted ray \( R_{trans} \):
  \( r_{trans} = TraceRay(R_{trans}) \)
- compute resulting RGB value from \( r_s, r_{refl}, r_{trans} \) using the lighting model.

**Image processing**
How computer images work?

Continuous real image

Digitization (e.g. scanning)

square array of numbers (abstract pixels)

each physical pixel covers an area

display (physical pixels)
The eye blurs pixels into continuous image

perceived image

What can go wrong?

Pipeline: sample - process - reconstruct
All kinds of artifacts can appear
- jaggies
- alias patterns
- moire patterns
- temporal aliasing (wheels going wrong way)

Question: how do we avoid all this?

Aliasing

Slightly different frequency

What can go wrong?

Lower frequency appears

Shrinking

Naïve 1.5x shrinking: drop 2 out of 3

What do we get?

But we want (impossible)
Frequency analysis

The key to fighting aliasing is to avoid frequencies we cannot represent.

**Big question:**
When can we reconstruct a continuous signal from samples?

![Graph showing original, samples, and reconstructed signals.]

Need enough samples, to be more precise, sampling frequency should be twice the frequency of the wave.

---

**Frequency analysis**

- decompose everything into waves
- mathematically convenient to use complex waves

\[ A \cos(\omega t + \phi) \]

Amplitude \( A \), frequency \( \omega \), phase \( \phi \)

\[ Ae^{i(\omega t + \phi)} = A \cos(\omega t + \phi) + i \sin(\omega t + \phi) \]

---

**Fourier transform**

Decomposes into freq. components

\[ \mathcal{F}[x](\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt \]

Inverse transform reconstructs the function

\[ \mathcal{F}^{-1}[X] = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega \]

---

**Examples of FT pairs**

- \( \mathcal{F}(\delta(x)) = \text{constant or wave} \)
- \( \mathcal{F}(\text{constant or wave}) = \delta(x) \)

**Box sinc function**

\[ \delta(x) \]

\[ \text{sinc}(x) \]

*For all \( x \neq 0 \), \( \delta(x) = 0 \)

\[ \forall \epsilon > 0, \int_{-\epsilon}^{\epsilon} \delta(x) dx = 1 \]
Examples of FT pairs

| Gaussian | Gaussian |

Comb function

$$comb = \sum_{n=1}^{\infty} \delta(t - nT)$$

FT of a comb is also a comb

Properties

<table>
<thead>
<tr>
<th>Name of the property</th>
<th>Signal Transformation</th>
<th>Fourier Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linearity</td>
<td>( x(t) = y(t) )</td>
<td>( \mathcal{F}{x(t)} = \mathcal{F}{y(t)} )</td>
</tr>
<tr>
<td>Time shifting (Section 3)</td>
<td>( x(t - a) )</td>
<td>( e^{-i2\pi a \omega} \mathcal{F}{x(\omega)} )</td>
</tr>
<tr>
<td>Zooming (Section 4.)</td>
<td>( x(\omega) )</td>
<td>( \mathcal{F}{x(t)}</td>
</tr>
<tr>
<td>Differentiation</td>
<td>( x(t) )</td>
<td>( i\omega \mathcal{F}{x(t)} )</td>
</tr>
<tr>
<td>Convolution (Section 2.3)</td>
<td>( x(t) * y(t) )</td>
<td>( \mathcal{F}{x(t) * y(t)} = \mathcal{F}{x(\omega)} \mathcal{F}{y(\omega)} )</td>
</tr>
</tbody>
</table>

Shannon theorem

A function can be reconstructed from samples if

- it is bandlimited (Fourier transform is zero for large frequencies)
- sampling frequency is at least twice the max. frequency of the function

Filtering

solution to aliasing problems: get rid of high frequencies we cannot reconstruct
In freq. Domain: multiply by a box what does it mean in the spatial domain?

Convolution

multiplication in the frequency domain is convolution in the spatial domain

\[ U(\omega) = X(\omega)Y(\omega) \]

\[ u(t) = \int_{-\infty}^{t} x(s)y(t-s)ds \]