Illumination Models and Surface Rendering

Eric C. McCreath

School of Computer Science
The Australian National University
ACT 0200 Australia

ericm@cs.anu.edu.au
Overview

- Light Sources
- Ambient, Diffuse, and Specular Lighting
- Phong Reflection Model
- Surface Rendering - Phong, Gouraud
- Limitations and some more advanced approaches
The illumination model or lighting model is used to determine the
colour at points on an object's surface.

Surface-rendering methods use a lighting model (combined with
information about the scene) to generate the pixel colours that make
up the scene.
Introduction

• Photorealism requires:
  – accurate representation of the surfaces (and their properties) that make up scene, and
  – accurate modeling of light and its interaction within the scene.

• Both of these are complex, can use large amount of memory and are computationally intensive. Hence, simplification is required to reduce programming complexity, memory footprint, and computational overhead.
Light Sources

• Light sources contribute to the lighting effects and can be modeled in a number of ways including:
  - ambient
  - point light source
  - infinitely distant light source
  - light surfaces
• Light bounces (reflects) around a scene. So from any point in the scene there is background light moving in all directions. A simple way of modeling this is called ambient lighting.

• One could measure the ambient light in a scene by taking the average amount of light at different wavelengths traveling in all directions from every point in the scene. This average light can then be used in modeling the scene.

• For the entire scene the only information that needs to be stored is the RGB intensity.

• This provide a very simple way of modeling the multitude of reflections that take place in a scene. However it will not capture the variance (directional and positional) of light moving around a scene.
Point Light Sources

- If a scene includes a point from which light is coming, such as a light globe, then such lighting can be simply modeled as a point. There may be a number of such light sources in the scene.

- For each light source one would store: a position, RGB intensity, and also attenuation parameters.

- Physics tells us light from a point source attenuate proportional to $1/d^2$, where $d$ is the distance from the source to the surface. However this often creates scenes which have objects close to the light source very bright and object away from the light source gaining very little light. So a function that enables more control over attenuation is often used:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$
A point light source may be simply augmented to be a directional spot light source. This can give a “spot-light” effect. And can be simply done with an angular intensity attenuation function (e.g., powers of cos) along with information about the direction the light is pointing.
Infinitely Distant Light Sources

- Some light source may be “distant” from the scene and as such the light will basically travel in parallel within the scene and the attenuation will not be noticeable, these light source can be modeled as a infinitely distant light source.

- This will involve storing a light direction and the RGB intensity of the light source.
Ambient Light

- Ambient lighting can be applied to the diffuse material properties of a surface's material.
- Ambient lighting produces flat-looking images.
- Colour of a surface does not change with different viewing angles.

\[ I = I_p (\cos \theta) dA d\omega \]

\[ I = I_p \frac{(\cos \theta) dA d\omega}{(\cos \theta) dA_o d\omega_o} \]
• To get lighting working in OpenGL you need to first turn lighting on:

```c
gl.glEnable(GL2.GL_LIGHTING);
```

• Then you can set up and enable a light:

```c
float ac[] = { 0.2f, 0.2f, 0.2f, 1.0f };
gl.glLightfv(GL2.GL_LIGHT1, GL2.GL_AMBIENT, ac, 0);
gl.glEnable(GL2.GL_LIGHT1);
```

• The material needs to be given a colour:

```c
float df[] = { 1.0f, 0.2f, 0.0f, 0.0f };
gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE, df, 0);
```
Diffuse lighting is good for approximating surfaces where light is reflected in all directions (like the mat paint on a rooms walls). These are known as Lambertian surfaces. The viewing direction makes no difference on the intensity of the observed surface, however, the direction the angle light sources hits the surface does make a difference.

- The intensity is proportional to \( \cos \theta \) of the angle the light hits the surface.

```c
float f[] = { 0.0f, 0.0f, 5.0f, 1.0f };  float dc[] = { 1.0f, 1.0f, 1.0f, 1.0f };  gl.glLightfv(GL2.GL_LIGHT1, GL2.GL_DIFFUSE, dc, 0);  gl.glLightfv(GL2.GL_LIGHT1, GL2.GL_POSITION, f, 0);  ```

Setting up the diffuse lighting in OpenGL. (diffuse material set in the previous slide)
Specular Lighting

- Specular lighting helps model shiny surfaces.

Setting up the light:

```cpp
float dc[] = { 1.0f, 1.0f, 1.0f, 1.0f };
gl.glLightfv(GL2.GL_LIGHT1, GL2.GL_SPECULAR, dc, 0);
```

Setting up the material:

```cpp
float sf[] = { 1.0f, 1.0f, 1.0f, 0.0f };
gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR, sf, 0);
gl.glMaterialf(GL2.GL_FRONT_AND_BACK, GL2.GL_SHININESS, 10.0f); // 1 - 128
```
• To get lighting working in OpenGL you need to first turn lighting on:

```c
gl.glEnable(GL2.GL_LIGHTING);
```

• Then you can set up and enable a light:

```c
float ac[] = { 0.2f, 0.2f, 0.2f, 1.0f };
gl.glLightfv(GL2.GL_LIGHT1, GL2.GL_AMBIENT, ac, 0);
gl.glEnable(GL2.GL_LIGHT1);
```

• The material needs to be given a colour:

```c
float df[] = { 1.0f, 0.2f, 0.0f, 0.0f };
gl.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE, df, 0);
```
The Phong reflection model combines: ambient, diffuse, and spectral lighting effects.

\[ I_p = k_a i_a + \sum_{\text{lights}} \left( k_d (L.N) i_d + k_s (R.V) ^ \alpha i_s \right) \]

Note the old fixed pipeline graphics cards use cos of the angle between L+V and N rather than R and V. This is referred to as the Blinn-Phong approach.
Normals

- Normals undergo a transformation that is based on the model view matrix. These normals are important for the lighting calculations.

- If $M$ is the upper left $3 \times 3$ matrix of the model view matrix then the transformation matrix on the normals is $(M^{-1})^T$. We can derive this by asserting that the new normal must have the same dot product between any vector and the original normal and that of the transformed vector and the new transformed normal. That is if $n$ is the original normal, $m$ is the new normal and $u$ is some vector we expect:

$$ m \cdot Mu = n \cdot u $$

- The matrix $(M^{-1})^T$ may not preserve the length of the normal vector, as the reflection model may expect unit length normals these vectors may require normalization. This may be done within a shader. Or in the old fixed pipeline approach this may be enabled by:

```glsl
gl.glEnable(GL2.GL_NORMALIZE);
```
Gouraud Shading

- In Gouraud shading colours are calculated at the vertices and then interpolated across the pixels of the polygon.

- Note, the surfaces norms can be different at different vertices on the polygon.

- Advantages:
  - fast,
  - better than flat shading (entire polygon has the same colour in flat shading).

- Limitations:
  - specular highlights are not properly rendered across the polygon,
  - often require lots of small polygons.
Phong Shading

- The surface norms are interpolated across the pixels of the polygon and than the Phong reflection model is used to calculate the colour at each pixel.

- Phong shading requires a lot more computation at each pixel (in comparison to Gouraud shading).

Spot light on a tiled surface, with a lot more tiles then the previous image. (should give an effect like phong shading)

With modern GPUs and their programmable shaders we can simply and efficiently implement Phong shading (and lots of other shading approach).
• The bidirectional reflectance distribution function (BRDF) characterizes the amount of light reflected on a surface as the lights incident direction and observation direction changes.

• The function is often denoted:

\[ f_r(\omega_r, \omega_i) \]

• where \( \omega_i \) is the unit length vector that points to the light, and \( \omega_r \) is the unit vector that points to the observer (with respect to the surface normal). The BRDF is a ratio of the reflected radiance to the incident irradiance and has units \( \text{sr}^{-1} \) (where sr is steradians or the solid angle).
Better Material Modeling

• As we model a surface of a material the diffuse and specular aspects can be separated and modeled using different approaches. The colour can be additively combined.

• This Blinn-Phong approach is a simple and a good starting point for modeling surface material, however, it is somewhat limited and does not model some surfaces well. (e.g. with metal surfaces the amount of reflection is dependent on the angle of the light source, or object like the moon or a tennis ball their intensity is more uniform rather than be Lambertion in terms of the diffuse aspects)

• One could empirically measure the BRDF of a material and then use this to model the material.

• There as also more complex aspects that may need to be considered such as: subsurface scattering (skin), bumpy material, scratched surfaces (stainless steel), wave properties (surface of a CD).
Fresnel Effects

- Surfaces, such as metal, water, glass, etc., have Fresnel effects at glazing angles. This is where the amount of light reflected increases (in a non-linear fashion) as the angle of the indicant light increases.

- So if you graphed the amount of light reflect at different angles you could use these graphs to better model materials.

- The simple Blinn-Phong approach does not capture this effect.

- To model this people will often move to a Cook-Torrance model.
Rough Surfaces

- The Lambertians model makes an assumption that the intensity of a surface is the same from all viewing angles. Although a good approximation, particularly for very smooth surfaces, this is generally not the case for the diffuse aspects of rough surfaces.

- So for rough surfaces, such as the moon or a tennis ball, the Oren-Nayar Model will produce more realistic images.

- Modern shaders provide great ways of including these more sophisticated surface models within real-time scenes.

https://en.wikipedia.org/wiki/Oren%E2%80%93Nayar_reflectance_model - Both images GPL
References

- Specular BRDF (Bidirectional Reflectance Distribution Function) - https://support.solidangle.com/pages/viewpage.action?pageId=5800819
- Chapter 4.4, Fundamentals of Computer Graphics, David J. Eck, 2010