レイトレーシングによるコンピュータグラフィクス入門
- 物体の陰影の計算 -

金森 由博*

2014 年 4 月 23 日

1　課題

シーン定義ファイル diffuse.cfg および phong.cfg を読み込んで、課題のページの参照画像のような結果が得られるよう、DiffuseMaterial.h および PhongMaterial.h というファイルの calcReflectedLight 関数を実装しよう。

2　物体の陰影の計算

3 次元の形状モデルをより立体的に表現するためには、光源、視点、形状モデルの位置関係や、物体表面の材質などから、表面の明るさを計算する必要がある。そしてその明るさに応じて、物体の色を決める。このように物体表面の明るさ (陰影; shade) を計算することをシェーディング (shading) と呼ぶ[*1]。ここではシェーディングのための簡易な計算モデルとして、次のものを扱う。

拡散反射 (diffuse reflection)　石膏など、光沢がない物体表面での反射。
鏡面反射 (specular reflection)　プラスチック、金属などの光沢を生む反射。
環境光 (ambient illumination)　光が周囲の物体表面で反射を繰り返し、環境全体に、ある一定の明るさを生じる効果の近似。

シェーディングの際は、これらのモデルを組み合わせることで、物体表面の明るさを計算する。なお、上記のうち環境光は、物体の明るさに単純に一定の値を加算することで表現される[*2]。拡散反射と鏡面反射については、「光がある方向から入射してきたら、どの方向へどのくらいの強さで光が反射されるか？」という反射の割合を表す、**双方向反射率分布関数** (Bidirectional Reflectance Distribution Function; BRDF) で表現される。以下、拡散反射と鏡面反射の計算モデルについて説明する。

* kanamori@cs.tsukuba.ac.jp

[*1] 物体が光を遮ることで他の物体に落とす影を計算することは、シャドウイング (shadowing) といい、シェーディングとは区別する。

[*2] 光が空間中を行き交う様子を厳密に計算すれば、環境光は不要であり、より写実的な画像生成が可能である。このような計算を行う方法を**大域照明** (global illumination) と呼ぶ。それに対して局所的な光の振る舞いだけを扱う方法は**局所照明** (local illumination) と呼ばれる。この実験では局所照明のみを対象とする。

2.1 拡散反射

石膏や紙など、光沢のない物体表面は、すべての方向にほぼ一様な強さで光が反射するとみなせる。このような面を拡散反射面 (拡散面) と呼ぶ。特に、すべての方向に全く一様な強さで光が反射するとした場合は「その面は完全拡散面である」という。拡散反射した光は、見る方向 (面と視線のなす角度) に無関係に同じ明るさにになる。

光が拡散面に入射するとき、反射光の強さは、入射角の余弦に比例することが知られている。これを Lambert の余弦則という。Lambert の余弦則に基づき、拡散面での反射光の強さ $\mathbf{I}_d$ は次の式で計算できる (図 1)。

$$\mathbf{I}_d = \mathbf{I}_{in}\,\mathbf{k}_d\,\cos\alpha \tag{1}$$

ここで、$\mathbf{I}_{in}$ は入射光の強さ、α は面の単位法線ベクトル $\mathbf{N}$ と光の入射方向を表す単位ベクトル $\mathbf{L}$ のなす角であり、$\mathbf{k}_d$ はその面の**拡散反射率** (値の範囲は 0 から 1 の間) である。$\cos\alpha$ は $\mathbf{N}$ と $-\mathbf{L}$ の内積、つまり $-(\mathbf{N}\cdot\mathbf{L})$ である。なお、$\mathbf{N}$ と $-\mathbf{L}$ のなす角によっては $\cos\alpha$ の値が負になることがある。これは「面の裏側から光が届いている」という本来ありえない状態を表している。このような状態を除外するために、(1) 式の $\cos\alpha$ を次のように補正する。

$$\mathbf{I}_d = \mathbf{I}_{in}\,\mathbf{k}_d\,\max\{\cos\alpha, 0\} \tag{2}$$

ここで $\max\{a, b\}$ は a, b のうち大きい方を返す関数である。なお、計算上は、$\mathbf{I}_{in}$, $\mathbf{k}_d$ は RGB 成分それぞれ異なる値を持っているものとし、RGB 別々に計算する。つまり、$\mathbf{I}_d = (r_d, g_d, b_d)^T$, $\mathbf{I}_{in} = (r_{in}, g_{in}, b_{in})^T$, $\mathbf{k}_d = (r_k, g_k, b_k)^T$ であるとすると (列ベクトルだと考えて転置の記号 T をつけている)、

$$r_d = r_{in}\,r_k\,\max\{\cos\alpha, 0\} \tag{3}$$
$$g_d = g_{in}\,g_k\,\max\{\cos\alpha, 0\} \tag{4}$$
$$b_d = b_{in}\,b_k\,\max\{\cos\alpha, 0\} \tag{5}$$

のように計算する。これ以降の反射の計算も同様である。

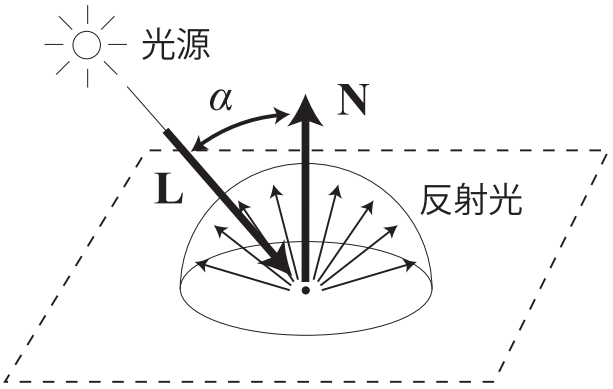

図 1　拡散反射.

2.2 鏡面反射

プラスチックや金属などでできた光沢のある面は、面の一部が光を強く反射し、ハイライトを生じる。ハイライトのできる場所や明るさは、拡散反射の場合と異なり、見る方向によって異なる。このような反射のことを**鏡面反射**と呼ぶ。特に、鏡のように入射光と反射光が面の法線となす角が等しい反射を完全鏡面反射 (perfect specular reflection) と呼び、それ以外の面 (表面に細かい凹凸がある面) で起こる反射を光沢反射 (glossy reflection) と呼ぶことがある。完全鏡面反射の場合は、光が強く反射されるのは入射角と反射角が一致する方向のみである。一方、光沢反射の場合、入射角と反射角が一致する方向から少しそれても表面の凹凸のために近傍からの光が反射されるが、入射角と等しい角からの角度のずれが大きくなるに従って、反射光の強さは急激に減衰する。

鏡面反射、特に光沢反射をモデル化したものとして、この実験では Phong のモデルを扱う (図 2)。入射角と反射角が等しくなるような反射方向 $\mathbf{R}$ と、視点の方向 $\mathbf{V}$ とのなす角を γ とすると、Phong のモデルでは、ずれ角 γ が大きくなるに従って反射光が減衰する現象を $(\cos\gamma)^n$ で近似する (n は自然数)。式 (1) の $\mathbf{k}_d$ に相当する反射率、つまり**鏡面反射率**は、この $(\cos\gamma)^n$ と、入射角 α の関数 $W(\alpha)$ とで表される。

$$\mathbf{I}_s = \mathbf{I}_{in}\, W(\alpha)\, \cos^n\gamma \tag{6}$$

ここで n の値を大きくすると、反射光の分布が狭まりシャープなハイライトが得られる (鏡に近くなる)。n を小さくするとハイライトがぼんやりする。実用上は、入射角 α とは無関係に $W(\alpha)$ を一定値 (材質ごとに異なる定数) として扱うことが多い。この実験でも同様に、$W(\alpha)$ を材質ごとの定数 $\mathbf{k}_s$ で置き換え、次のように計算する。

$$\mathbf{I}_s = \mathbf{I}_{in}\, \mathbf{k}_s\, \cos^n\gamma \tag{7}$$

なお、この式でも $\mathbf{I}_{in}$, $\mathbf{k}_s$ は RGB 成分それぞれ異なる値を持っているものとし、RGB 別々に計算する。また、$\cos\gamma$ が負になるような場合は除外する。

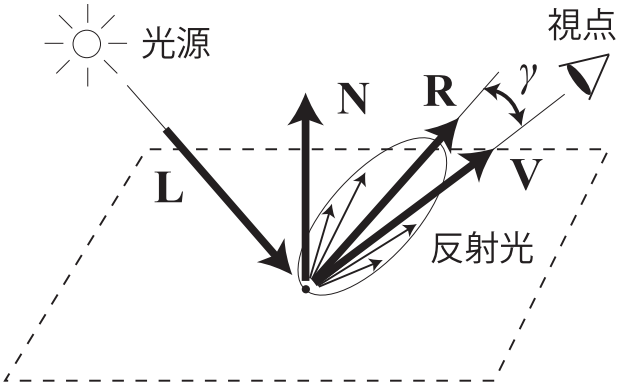

図 2 鏡面反射 (Phong モデル).

Phong モデルは簡易に計算でき、プラスチックなどの質感をうまく表現できる。鏡面反射を表現する他のモデルとして、金属の質感をうまく扱える Blinn モデル、Cook-Torrance モデルなどがある。

2.3 物体の表面の色

ここまで述べた、拡散反射 (式 (1))、鏡面反射 (式 (7))、環境光を組み合わせて、この実験では物体表面の色 $\mathbf{I}$ を次のように計算する。

$$\mathbf{I} = \mathbf{I}_d + \mathbf{I}_s + \mathbf{I}_a \tag{8}$$

ここで、$\mathbf{I}_a$ は環境光を表す定数であり、この値も物体ごとに決めることにする。また、やはり $\mathbf{I}_a$ も RGB ごとに異なる値を持つものとし、$\mathbf{I}$ の計算も RGB 別々に行う。

3 プログラムについて

この実験で使うプログラムでは、物体の材質を表す `Material` というクラスが定義されており、これを継承したクラスとして、

`DiffuseMaterial` 拡散反射
`PhongMaterial` 光沢反射 (Phong のモデル)
`PerfectSpecularMaterial` 完全鏡面反射 (今回は使わない)
`SpecularRefractionMaterial` 屈折 (今回は使わない)
`PseudoNormalColorMaterial` 法線ベクトルを擬似カラーで表示 (今回は使わない)
`TexturedMaterial` テクスチャ付き光沢反射 (今回は使わない)

の 6 つが用意されている。実験では、これらのクラスの中の、まだ実装されていない部分を埋めることになる。今回は `DiffuseMaterial` および `PhongMaterial` のみを実装する。

`Material` クラスには、

Material.h
```
vec3 calcReflectedLight(const HitRecord &record,
 const vec3 &incomingDir, const vec3 &outgoingDir)
```

という関数が定義されている。`Material` クラスを継承したクラスのソースコード内で、この関数の中身を実装すればよい。ここで `record` は、レイと物体との交点における情報を保持している。面の単位法線ベクトルを得るには `record.normal` にアクセスすればよい。`incomingDir` は入射光の方向を表す単位ベクトル (向きは光源から物体表面に向かう向き)、そして `outgoingDir` は反射光の方向を表す単位ベクトル (向きは物体表面から離れる向き) である。

`Material` クラスの中には

Material.h
```
vec3 ambient;
```

という変数があり、これが物体ごとの環境光 $\mathbf{I}_a$(式 (8)) を表している。ただし `ambient` の値は `calcReflectedLight` 関数の中では足す必要はない[*3]。また、式 (2) および式 (7) における $\mathbf{I}_{in}$ を

[*3] `calcReflectedLight` 関数の中で足さない代わりに、`RecursiveRayTracer::traceRec` 関数の末尾の `color += material->getAmbientCoeff();` という行で計算している。

calcReflectedLight 関数内で掛ける必要もない[*4]。DiffuseMaterial クラスには

DiffuseMaterial.h

```
vec3 kd;  // 拡散反射係数
```

が定義されている。物体の材質として DiffuseMaterial を選んだ場合は、式 (8) のうち、$\mathbf{I}_s$ の項を除いた計算が可能である。また、PhongMaterial クラスには

PhongMaterial.h

```
Real shininess; // cos^n γ の n
vec3 ks; // 鏡面反射係数
vec3 kd; // 拡散反射係数
```

が定義されているので、式 (8) をそのまま表現できる。これらの変数を使って、calcReflectedLight 関数をそれぞれ実装する。なお、Phong モデルを実装するために必要なベクトル $\mathbf{R}$(図 2 参照) は、ベクトル・マトリクスライブラリの中の vec3 クラスに定義された、reflect 関数で計算できる。

my_algebra.h

```
vec3 i; // 光の入射ベクトル
vec3 r = reflect(i, n); // n は単位法線ベクトル
```

また、余力があれば、Blinn モデルや Cook-Torrance モデルなどの反射モデルも、自分で調べて実装してみてほしい。

[*4] 同じく RecursiveRayTracer::traceRec 関数の中の光源に関するループの中で、mult(tmpColor, tmpColor, lightInfo.color); という行で光源の色 $\mathbf{I}_{in}$ が lightInfo.color として掛けられている。