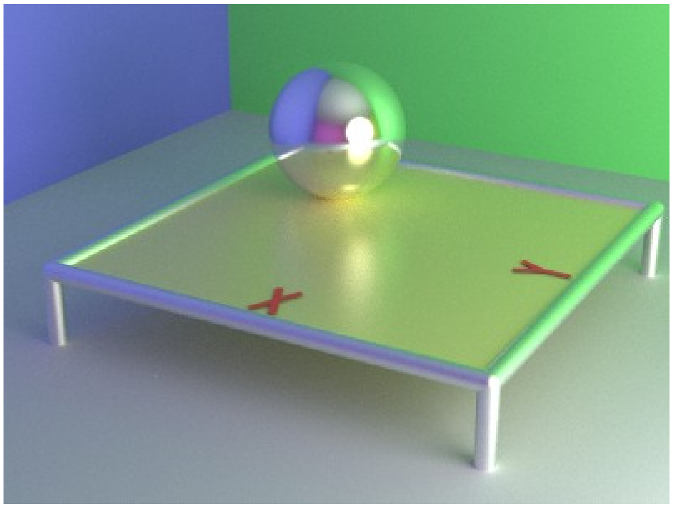
CS500 Project 3



Synopsis

Enhance the very minimal diffuse-only BRDF from Project 2 to include the full micro-facet BRDF in the lighting equation:

$$L_o(\omega_o) = L_i(\omega_i)\,(N\cdot\omega_i)\,BRDF(\omega_i,\omega_o)$$

where

$$BRDF(\omega_i,\omega_o) = \frac{K_d}{\pi} + \frac{D(m)\,G(\omega_i,\omega_o,m)\,F(\omega_i\cdot m)}{4|\omega_i\cdot N||\omega_o\cdot N|},$$

thereby allowing the modeling of surfaces over a broad range of roughness and shininess values.

Note: $m=(\omega_o+\omega_i)/\|\omega_o+\omega_i\|$ is the half-vector.

Instructions

Very few changes need to be made in the path-tracing algorithm of the previous project. Most of the changes which do occur are in the three functions that provide the interface to the BRDF: SampleBrdf, PdfBrdf, and EvaBrdf. See the next page for details.

Changes to basic algorithm

The calculations in the three BRDF functions must be provided with a new parameter, the outgoing light direction ω_o, in addition to the incoming light direction ω_i, and the normal N. Their signatures will now be:

- $\omega_i = \text{SampleBrdf}(\omega_o, N)$
- $\text{PdfBrdf}(\omega_o, N, \omega_i)$
- $\text{EvalScattering}(\omega_o, N, \omega_i)$

The vector ω_o starts as the negative of the input ray's direction, and is stepped forward to $-\omega_i$ at the bottom of the loop:

```
TracePath(Ray ray):
    ωo = −ray.direction
    while ...
        // Explicit
            ...
        // Extend
            ...
        // Implicit
            ...
        // Step forward
        P ← Q
        ωo = −ωi
```

Warning about NaN's and Inf's

If an illegal floating point value ever gets into the image, no amount of arithmetic afterward will ever remove it. Prevent this by using **isnan** and **isinf** to test for illegal values.

```
for pass=1 to forever:
    #pragma for OpenMP
    for each y scanline:
        for each x pixel:
            C = TracePath( <ray from eye through pixel> )
            if all components of C are legal:
                image[x,y] += C
    occasionally:
        write image[x,y]/pass
```

The full algorithm as it now stands

```
TracePath(Ray ray):
        C = (0,0,0) // Accumulated light
        W = (1,1,1) // Accumulated weight
        // Initial ray
        P = Trace ray into the scene  // Intersection points must record: object, distance, normal ...
        N = P's normal
        if P indicates no intersection:  return (0,0,0)
        if P is a light: return Radiance(P) // Light objects must provide a radiance method

        ωo = −ray.direction
        while random() <= RussianRoulette: // 0.8 is a good value for RussianRoulette
            // Explicit light connection
            L = SampleLight()        // Randomly choose a light and a point on that light.
            p = PdfLight(L)/GeometryFactor(P,L) // Probability of L, converted to angular measure
            ωi = direction from P  toward L
            I = Trace ray from P toward L // Sometimes called a shadow-ray
            if p>0 and I exists and is the chosen point on the chosen light:
                f = EvalScattering(ωo, N, ωi)
                C += W * f/p * EvalRadiance(L)

            // Extend path
            ωi = SampleBrdf(ωo, N)  // Choose a sample direction from P
            Q = Trace ray from P in direction ωi into the scene
            if Q is non-existent: break
            f = EvalScattering(ωo, N, ωi)
            p = PdfBrdf(ωo, N, ωi) * RussianRoulette
            if p < ϵ : break  // Avoid division by zero or nearly zero: ϵ=10^−6
            W *= f/p

            // Implicit light connection
            if Q is a light:
                C += W * Radiance(Q)
                break

            // Step forward
            P ← Q
            N ← P's normal
            ωo ← −ωi
        return C
```

Changes to the BRDF:

The interaction of a ray with a surface can now result in either a diffuse interaction, or a reflection interaction, and the the path-tracing loop must now choose (with a known probability) between the two: As a reasonable start, choose the probabilities of diffuse sampling and reflection sampling as

$$p_d = \|K_d\|/s\ ,\quad p_r = \|K_s\|/s\ \text{, respectively,}\quad \text{where}\ \ s = \|K_d\| + \|K_s\|\ .$$

Experiment with other choices if you wish. Just make sure the two probabilities are non-negative and sum to one.

Sampling a BRDF: $\text{SampleBrdf}(\omega_o, N)$

Choose **diffuse** or **reflective** via a random number ξ :

if $\xi < p_d$: *choice*=**diffuse**

otherwise: *choice*=**reflection**

If *choice*=**diffuse**, sample a micro-facet around N via two random numbers ξ_1, ξ_2 :

Return $\omega_i = \mathit{SampleLobe}(N\ ,\ \sqrt{\xi_1}\ ,\ 2\pi\xi_2)$

If *choice*=**reflection,** sample a micro-facet normal in a lobe around N and reflect:

$m = \mathit{SampleLobe}(N\ ,\ \cos\theta_m\ ,\ 2\pi\xi_2)$

return $\omega_i = 2\,|\omega_o \cdot m|\, m - \omega_o$

where $\cos\theta_m$ depends on the $D()$ function used.

Phong BRDF: $\cos\theta_m = \xi_1^{\frac{1}{\alpha+1}}$

GGX BRDF: $\cos\theta_m = \cos\left(\tan^{-1}\left(\frac{\alpha_g\sqrt{\xi_1}}{\sqrt{1-\xi_1}}\right)\right)$

Beckman BRDF: $\cos\theta_m = \cos\left(\tan^{-1}\left(\sqrt{-\alpha_b^2\log(1-\xi_1)}\right)\right)$

PDF of the sample: $\text{PdfBrdf}(\omega_o, N\ , \omega_i)$

Given p_d and p_r as the probabilities of *choice* made in SampleBRDF, and P_d and P_r as the probabilities of the vector ω_i being calculated for the diffuse and reflection directions respectively, return the combined probability:

The **diffuse** probability calculation:

$P_d = |\omega_i \cdot N|/\pi$

The (specular) **reflection** probability calculation:

$$P_r = D(m)|m \cdot N|\ \frac{1}{4|\omega_i \cdot m|}\ \text{ where }\ m = (\omega_o + \omega_i).\mathit{normalized}()\ .$$

Return $p_d\, P_d + p_r\, P_r$

Evaluate a micro-facet BRDF scattering: $\text{EvalScattering}(\omega_o, N\ , \omega_i)$

The BRDF is now the sum of the diffuse and specular terms

Diffuse: $E_d = K_d/\pi$,

Specular: $E_r = \dfrac{D(m)\ G(\omega_i, \omega_o, m)\ F(\omega_i \cdot m)}{4|\omega_i \cdot N||\omega_o \cdot N|}$ where $m = (\omega_o + \omega_i).\mathit{normalized}()$

Return $|N \cdot \omega_i|\ \ (E_d + E_r)$

Microfacet Models for Refraction through Rough Surfaces

https://faculty.digipen.edu/~gherron/references/References/BRDF/EGSR07-btdf.pdf

All microfacet BRDFs have this general form:

$$\frac{K_d}{\pi} + \frac{D(m)\, G(\omega_i, \omega_o, m)\, F(\omega_i \cdot m)}{4|\omega_i \cdot N||\omega_o \cdot N|}$$

Where

- $m = (\omega_o + \omega_i)/||\omega_o + \omega_i||$ is the half-vector.
- K_d is the diffuse reflection (albedo) of the surface
- K_s is the specular reflection in the $\omega_i = \omega_o = N$ direction,
- α_p, α_b, α_g are the surface roughness values for **Phong**, **Beckman** and **GGX** respectively.

The paper lists three common versions for **D** and **G: Phong**, **Beckman** and **GGX**.

The easiest is **Phong**, but recently, graphics has been trending toward **GGX**.

Characteristic factor:

The so called characteristic function in the **D** and **G** factors below is defined as

$$\chi^+(d) = \begin{cases} 1 & \text{if } d > 0 \\ 0 & \text{if } d \le 0 \end{cases}$$

and implemented with a simple "if" statement.

F factor

F is the Fresnel (reflection) is usually approximated by Schlick as

$$F(d) = K_s + (1 - K_s)(1 - |d|)^5$$

where K_s is the specular reflection color at $L = V = N = H$.

The exact formulation (if you are interested) is

$$F(L, H) = \frac{1}{2} \frac{(g-c)^2}{(g+c)^2} \left(1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right)$$

where

$$g = \sqrt{\frac{\eta_t^2}{\eta_i^2} - 1 + c^2}$$

and

$$c = |L \cdot H|$$

and η_i and η_t are indices of refraction of the two materials.

D factor

D is the micro-facet distribution.

In the following, $\tan\theta_m = \sqrt{(1.0-(m\cdot N)^2)}\,/\,(m\cdot N)$ and $\tan\theta_v = \sqrt{(1.0-(v\cdot N)^2)}\,/\,(v\cdot N)$.

Phong: $D_p(m) = \chi^+(m\cdot N)\ \frac{\alpha_p+2}{2\pi}(m\cdot N)^{\alpha_p}$

(α_p : $1..\infty$; increasing means smoother surface)

Beckman: $D_b(m) = \chi^+(m\cdot N)\ \frac{1}{\pi\,\alpha_b^2 (N\cdot m)^4}\ e^{\frac{-\tan^2\theta_m}{\alpha_b^2}}$

(α_b : $0..1$; increasing means rougher surface)

similar to Phong for smooth surfaces using $\alpha_p = 2\alpha_b^{-2} - 2$

GGX: $D_g(m) = \chi^+(m\cdot N)\ \frac{\alpha_g^2}{\pi (N\cdot m)^4 (\alpha_g^2+\tan^2\theta_m)^2}$

G factor:

The G term should be calculated via the smith method:

$$G(\omega_i, \omega_o, m) = G_1(\omega_i, m)\ G_1(\omega_o, m)$$

where $G_1(\ldots)$:

Beckman:

$$G_1(v, m) = \chi^+\left(\frac{v\cdot m}{v\cdot N}\right) \begin{cases} \dfrac{3.535a + 2.181a^2}{1.0 + 2.276a + 2.577a^2} & \text{if } a<1.6 \\ 1 & \text{otherwise} \end{cases}$$

where $a = 1/(\alpha_b \tan\theta_v)$, and $\tan\theta_v$ is defined above.

Phong:

Same G_1 as **Beckman**, but with $a = (\sqrt{\alpha_p/2+1})\,/\,\tan\theta_v$.

GGX:

$$G_1(v, m) = \chi^+\left(\frac{v\cdot m}{v\cdot N}\right) \frac{2}{1 + \sqrt{1+\alpha_g^2 \tan^2\theta_v}}$$

Beware round off errors in calculating the G_1 function:

- The value of $(v\cdot N)$ may round up to greater than 1.0 (mathematically it shouldn't, but computationally it sometimes does). If so, return $G_1(\ldots)=1.0$.
- The calculation of $\tan\theta_v$ may be zero. If so, don't divide by it, instead return $G_1(\ldots)=1.0$.