Assignment 16 (BRDF Models)

BRDF = diffuse reflection + specular reflection

Diffuse reflection: main color of the object.

Specular Reflection: effect which shiny objects give when they reflect light in a particular angle.

In **scan-line rendering**, values of the BRDF are in the range $[0, 1]$. Since lights may change those values and make them higher than 1 we need to `clamp` the values at the end of the computation:

$$L\left(x, \omega_r\right) = \text{clamp}\left(\sum_l L(l, \vec{x}) f_r\left(x, \overrightarrow{lx}, \omega_r\right)\right)$$

$$\text{clamp}(y) = \begin{cases} 0 & y < 0 \\ y & y \in [0, 1] \\ 1 & y > 1 \end{cases}$$

HDR instead allows values outside of the $[0, 1]$ range, while mapping values outside of that interval into values inside of it via non-linear functions.

Diffuse reflection models

Lambert

According to the reflection law by Lambert, each point of an object hit by a ray of light, reflects it with uniform probability in all the directions.

The reflection is thus independent of the viewing angle and it corresponds to a constant BRDF: $f_r\left(x, \omega_i, \omega_r\right) = \rho_x$.

The quantity of light received by an object is however not constant: it depends on the angle between the ray of light and reflecting surface. Suppose we have the following parameters:

- n_x: unitary normal vector to the surface;
- d: direction of ray of light;
- α: angle between the two vectors.

The incidence of the incoming light is maximized when angle α is $0°$, and null if it is greater or equal than $90°$. In particular, Lambert has shown that the amount of light reflected is proportional to $\cos\alpha$, which can be computed as $\cos\alpha = n_x \cdot d$. In the assignment this becomes: `dot(L, N)`.

We can express the BRDF of the Lambert reflection for scan-line rendering with the following expression:

$$f_r\left(x, \overrightarrow{lx}, \omega_r\right) = f_{\text{diffuse}}\left(x, \overrightarrow{lx}\right) = \boldsymbol{m}_D \cdot \max\left(\overrightarrow{lx} \cdot \boldsymbol{n}_x, 0\right)$$

Which in shaders code becomes:

```
// vec3 C: color of the surface (RGB vector)
f = C * max(dot(L, N), 0.0);
```

Oren-Nayar

Idea: some materials are characterized by a phenomenon called retroreflection: they tend to reflect back in the direction of the light source. They are characterized by very rough surfaces, and they cannot be accurately simulated with the Lambert diffuse reflection model. The Oren-Nayar diffuse reflection model has been devised to more appropriately model such materials.

Typical real life materials that require this special technique are clay, dirt and some types of cloths:

It requires three vectors:

1. d: direction of the light;
2. n: normal vector;
3. ω_r: direction of the viewer.

The model is characterized by two parameters:

1. $m_D = (m_R, m_G, m_B)$: main color of the material;
2. $\sigma \epsilon \left[0, \frac{\pi}{2}\right]$: roughness of the material, higher values of σ produces rougher surfaces. If $\sigma = 0$ it converges to the Lambert diffusion.

The model, can be computed with the following formulas:

$$\theta_i = \cos^{-1}\left(\overrightarrow{lx} \cdot \boldsymbol{n}_x\right)$$
$$\theta_{\boldsymbol{r}} = \cos^{-1}\left(\omega_r \cdot \boldsymbol{n}_x\right)$$
$$\alpha = \max\left(\theta_{\boldsymbol{i}}, \theta_{\boldsymbol{r}}\right)$$
$$\beta = \min\left(\theta_{\boldsymbol{i}}, \theta_{\boldsymbol{r}}\right)$$
$$A = 1 - 0.5\frac{\sigma^2}{\sigma^2 + 0.33}$$
$$B = 0.45\frac{\sigma^2}{\sigma^2 + 0.09}$$
$$\boldsymbol{v_i} = \text{ normalize } \left(\overrightarrow{lx} - \left(\overrightarrow{lx} \cdot \boldsymbol{n}_x\right)\boldsymbol{n}_x\right)$$
$$\boldsymbol{v_r} = \text{ normalize } \left(\omega_r - \left(\omega_r \cdot \boldsymbol{n}_x\right)\boldsymbol{n}_x\right)$$
$$G = \max\left(0, \boldsymbol{v_i} \cdot \boldsymbol{v_r}\right)$$
$$L = m_D \cdot \text{clamp}\left(\overrightarrow{lx} \cdot \boldsymbol{n}_x\right)$$
$$f_{diffuse} = \left(x, \overrightarrow{lx}, \omega_r\right) = L(A + BG\sin\alpha\tan\beta)$$

Which in code become:

```
float thi = acos(dot(L, N));
float thr = acos(dot(V, N));
float alpha = max(thi, thr);
float beta = min(thi, thr);
float A = 1 - 0.5 * pow(sigma, 2) / (pow(sigma, 2) + 0.33);
float B = 0.45 * pow(sigma, 2) / (pow(sigma, 2) + 0.09);
vec3 vi = normalize(L - dot(L, N) * N);
vec3 vr = normalize(V - dot(V, N) * N);
float G = max(dot(vi, vr), 0.0);
vec3 L1 = C * clamp(dot(L, N), 0.0, 1.0);
return L1 * (A + B * G * sin(alpha) * tan(beta));
```

Toon (Diffuse)

Toon shading simplifies the output color range, using only discrete values according to a set of thresholds. In this way it achieves a cartoon-like rendering style. It can be used both for the diffuse and specular components of the BRDF.

The technique starts from a standard Lambert BRDF for the diffuse, and from a Phong or Blinn BRDF with $\gamma = 1$ for the specular components. Then it uses two colors (m_{D1}, m_{D2}) or (m_{S1}, m_{S2}) and a threshold (t_D or t_S) for determining which one to choose.

$$f_{\text{diffuse}}\left(x, \overrightarrow{lx}\right) = \begin{cases} \boldsymbol{m}_{D1} & \overrightarrow{lx} \cdot \boldsymbol{n}_x \geq t_D \\ \boldsymbol{m}_{Dn} & \overrightarrow{lx} \cdot \boldsymbol{n}_x < t_D \end{cases}$$

```
vec3 Toon_Diffuse_BRDF(vec3 L, vec3 N, vec3 V, vec3 C, vec3 Cd, float thr) {
    float thr2 = 0.07;
    vec3 Cd1 = 0.002f * C;

    if (dot(L, N) < thr2) {
        return Cd1;
    } else if(dot(L, N) < thr) {
        return Cd;
    } else {
        return C;
    }
}
```

Specular reflection models

Phong

In the Phong model, the specular reflection has the same angle as the incoming ray with respect to the normal vector, but it is oriented in the opposite direction, and it is positioned on the same plane as the light and the normal vectors.

The Phong model computes the intensity of the specular reflection from $\cos\alpha$: in this way the term is maximum if the specular direction is aligned with the observer, and zero when the angle is greater than 90°.

To create more contained highlight regions, the term $\cos\alpha$ is raised at a power γ. The greater is γ, the smaller is the highlight, and the more shiny the object appears to be.

Computation steps:

1. First we compute n', the projection of the light vector over the normal vector: $n' = n_x \cdot (d \cdot n_x)$;
2. $d' = n' - d$;

   $r = d + 2d'$, or with the notation of the rendering equation: $\boldsymbol{r}_{l,x} = 2\left(\overrightarrow{xl} \cdot \boldsymbol{n}_x\right)\boldsymbol{n}_x - \overrightarrow{xl}$;

   Where r is the reflected vector. In GLSL we can also do: `vec3 r = -reflect(L, N);`;
3. We can then compute the intensity of the specular reflection term as: $\mathrm{CoS}^{\gamma}\,\alpha = \mathrm{clamp}\,(\omega_r \cdot \boldsymbol{r})^{\gamma}$.

To summarize:

$$\boldsymbol{r}_{l,x} = 2\boldsymbol{n}_x \cdot \left(\overrightarrow{lx} \cdot \boldsymbol{n}_x\right) - \overrightarrow{lx}$$

$$f_{\text{specular}}\ (x, \vec{x}, \omega_r) = \boldsymbol{m}_S \cdot \mathrm{clamp}\,(\omega_r \cdot \boldsymbol{r}_{l,x})^{\gamma}$$

```
vec3 Phong_Specular_BRDF(vec3 L, vec3 N, vec3 V, vec3 C, float gamma)  {
    // reflection happens in the opposite direction
    // For this reason the minus sign is required
    vec3 r = - reflect(L, N);
    float intensity = pow(clamp(dot(V, r), 0.0, 1.0), gamma);
    return C * intensity;
}
```

Blinn

The Blinn reflection model is an alternative to the Phong shading model that uses the half vector h: a vector that is in the middle between ω_r and d.

$$\boldsymbol{h}_{l,x} = \frac{\overrightarrow{lx} + \omega_r}{\left|\overrightarrow{lx} + \omega_r\right|} = \text{ normalize } \left(\overrightarrow{lx} + \omega_r\right)$$

The formula when considering Blinn specular reflection becomes:

$$f_{\text{specular }}(x, \vec{x}, \omega_r) = \boldsymbol{m}_S \cdot \text{clamp}\left(\boldsymbol{n}_x \cdot \boldsymbol{h}_{l,x}\right)^{\gamma}$$

Toon (Specular)

$$\boldsymbol{r}_{l,x} = 2\boldsymbol{n}_x \cdot \left(\overrightarrow{lx} \cdot \boldsymbol{n}_x\right) - \overrightarrow{lx}$$

$$f_{specular}\left(x, \overrightarrow{lx}, \omega_r\right) = \begin{cases} \boldsymbol{m}_{S1} & \omega_r \cdot \boldsymbol{r}_{l,x} \geq t_S \\ \boldsymbol{m}_{S0} & \omega_r \cdot \boldsymbol{r}_{l.x} < t_S \end{cases}$$

```
vec3 Toon_Specular_BRDF(vec3 L, vec3 N, vec3 V, vec3 C, float thr)  {

    vec3 Ms0 =  C;
    vec3 Ms1 =  vec3(0, 0, 0);
    vec3 r = - reflect(L, N);
    if(dot(V, r) < thr) {
        return Ms1;
    } else {
        return Ms0;
    }
}
```