Realtime Rendering CS7GV3: Subsurface Scattering

Abstract

This project demonstrates the implementation of Subsurface Scattering on human skin. To enable a detailed render of human skin, the underlying rendering engine had to be developed with features like hierarchical mesh system which can be accessed using a Scene Graph and individual materials per mesh. This enabled the customization of different parts of the face, like setting material parameters for eyelashes, eyes and skin independently. The material shaders use state-of-the-art Physically Based Rendering techniques implemented in Unreal Engine 4 and Blender 2.8 based on Disney's Research on Physically Based Shading. For the diffuse part, it uses Disney's Principled BSDF model and for the specular component, the shader uses the GGX model used in Unreal Engine 4. The BSSRDF components used in the shader are the Dipole model and the Christensen-Burley model, designed at Disney.

Background

The BSSRDF model introduced in [1] simulates subsurface light transport in translucent materials. It enables efficient simulation of light effects that are not captured by a simple BRDF model. A general BRDF model assumes, light entering a surface at a particular point, leaves from the same point. This assumption holds true for metals, but fails for translucent materials, which show a significant transport below the surface. Using BRDF for materials that do not seem very translucent, generate a hard-looking images. Methods that factor in subsurface scattering can only capture the appearance of true translucent materials. The BSSRDF, denoted by ***S***, relates the out going radiance to the incidence flux. BRDF approximates BSSRDF with the assumption that light entering at some point leaves from the same point (**Figure 1**).

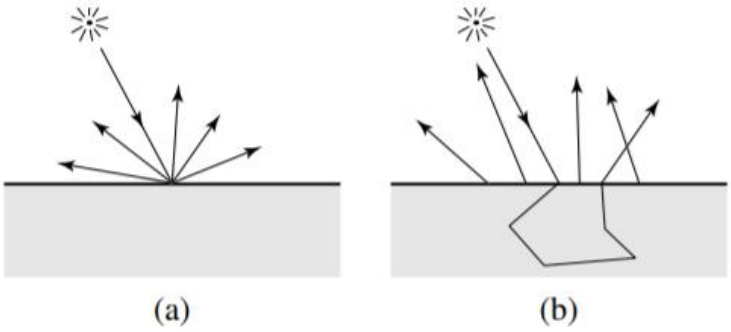

Figure 1: Scattering of light in (a) a BRDF, and (b) a BSSRDF.

Given BSSRDF, the outgoing radiance is computed by integrating the incident radiance over incoming directions and area.

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\, L_i(x_i, \vec{\omega}_i)\, (\vec{n} \cdot \vec{\omega}_i)\, d\omega_i dA(x_i).$$

Light propagating in participating medium is described by the volume rendering equation. For an infinitesimal beam entering a homogeneous medium, the incoming radiance will decrease exponentially with distance ***s***. This is called the reduced intensity.

$$L_{ri}(x_i + s\vec{\omega}_i, \vec{\omega}_i) = e^{-\sigma_t s} L_i(x_i, \vec{\omega}_i).$$

Here $\boldsymbol{\sigma_t}$ is the extinction coefficient, defined as $\boldsymbol{\sigma_t = \sigma_a + \sigma_s}$, where $\boldsymbol{\sigma_a}$ is the absorption coefficient and $\boldsymbol{\sigma_s}$ is the scatter coefficient which describe the properties of the medium in the volume rendering equation:

$$(\vec{\omega}\cdot\vec{\nabla})L(x,\vec{\omega}) = -\sigma_t L(x,\vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega},\vec{\omega}')L(x,\vec{\omega}')\,d\omega' + Q(x,\vec{\omega}).$$

The complete BSSRDF model is a summation of diffusion term and single scatter term, where the diffusion term is defined as:

$$S_d(x_i,\vec{\omega}_i;x_o,\vec{\omega}_o) = \frac{1}{\pi} F_t(\eta,\vec{\omega}_i) R_d(||x_i - x_o||) F_t(\eta,\vec{\omega}_o)$$

The $\boldsymbol{F_t}$ terms are the two Fresnel transmission terms for the incoming and outgoing radiance. The $\boldsymbol{R_d}$ term is the Reflectance or Diffusion Profile, here referring to the diffuse reflectance due to dipole source, which is computed as:

$$\begin{aligned} R_d(r) &= -D\frac{(\vec{n}\cdot\vec{\nabla}\phi(x_s))}{d\Phi_i} \\ &= \frac{\alpha'}{4\pi}\left[(\sigma_{tr}d_r + 1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr}d_v + 1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right]. \end{aligned}$$

The diffusion profile is formulated via a dipole approximation. The contribution of the incoming light ray is the one of two virtual sources. One positive real light source $\boldsymbol{Z_r}$ beneath the surface and one negative virtual light source is positioned above the surface at $\boldsymbol{Z_v = Z_r + 4AD}$ *and* $\boldsymbol{F_{dr}}$ is the Fresnel reflectance term, computed using η the IOR as:

$$F_{dr} = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta. \qquad D = \frac{1}{3\sigma_t'} \qquad A = \frac{1+F_{dr}}{1-F_{dr}}.$$

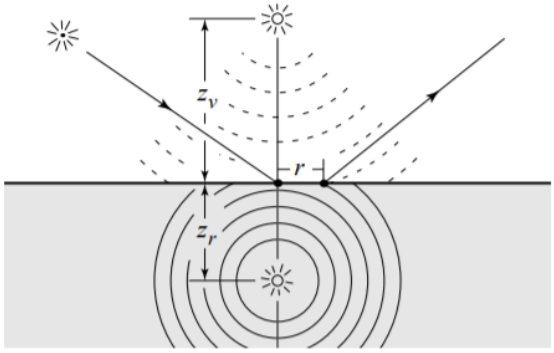

Figure 2: Dipole source for diffusion approximation

In (**Figure 2**) we can see the dipole source for diffusion profile approximation, where the incoming ray at a point X_i leaves from another point X_o the difference of which is the argument for the diffusion profile function ($|X_i - X_o|$).

The single scatter term is defined as $S^{(1)}$. The total outgoing radiance, L_o, due to single scattering is computed by integrating the incident radiance along the refracted outgoing ray.

$$L_o^{(1)}(x_o, \vec{\omega}_o) = \sigma_s(x_o) \int_{2\pi} F\, p(\vec{\omega}_i' \cdot \vec{\omega}_o') \int_0^{\infty} e^{-\sigma_{tc} s} L_i(x_i, \vec{\omega}_i)\, ds\, d\vec{\omega}_i$$

Here, $F = F_t(\eta, \vec{\omega}_o) F_t(\eta, \vec{\omega}_i)$ is the product of two Fresnel Transmission term and σ_{tc} is the combined extinction coefficient defined as: $\sigma_{tc} = \sigma_{tc}(x_o) + G\ \sigma_{tc}(x_i)$, where G is a geometry factor for flat surface defined as: $G = \frac{|\vec{n}_i \cdot \vec{\omega}_o'|}{|\vec{n}_i \cdot \vec{\omega}_i'|}$. The single scattering BSSRDF is defined implicitly by:

$$= \int_A \int_{2\pi} S^{(1)}(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\, L_i(x_i, \vec{\omega}_i)\, (\vec{n} \cdot \vec{\omega}_i)\, d\omega_i dA(x_i).$$

Where there is a change in variables as the first equation only integrates over the path length s along the refracted outgoing ray, when the two incoming and outgoing refracted rays intersect (**Figure3**), while in the second equation, defining the single scatter BSSRDF $S^{(1)}$ integrates over all incoming and outgoing rays.

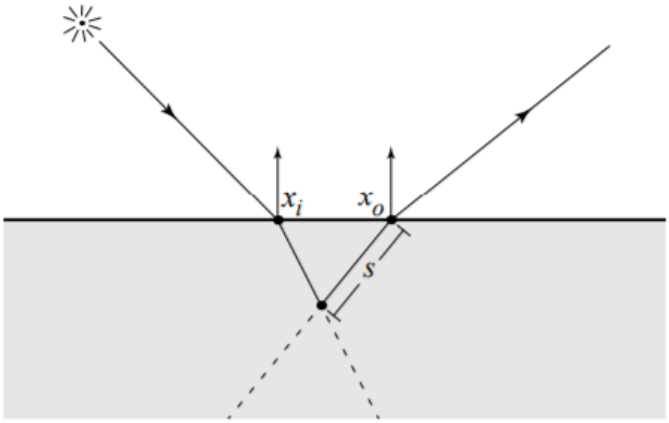

Figure 3: Single Scatter

Based on the strong foundation set by Jensen et al. many graphics researchers have developed various physically based subsurface scattering BSSRDF models. Similarly, another model introduced in [2] famously known as the Christensen -Burley Subsurface Scattering model is just as simple as the dipole model but matches Monte-Carlo reference very closely. Unlike the dipole model discussed above, this model has the single scattering part built in. At the time of working on this project, to my best knowledge this is the current state-of-the-art subsurface scattering technique used in Realtime Rendering. It is also implemented in Blender 2.8 and Walt Disney Animation Studio's Hyperion renderer.

Burley [2013; 2015] noted that the shape of the diffuse reflectance profile can be approximated quite well with a curve in the shape of a sum of two exponential functions divided by distance r:

$$R(r) = \frac{e^{-r/d} + e^{-r/(3d)}}{8\,\pi\,d\,r}$$

The ***d*** parameter shapes the height and width of the curve and is used as per artistic requirements, to be either physically accurate or exagerated. To determine the appropriate value of ***d*** corresponding to a physically meaningful quantity, volume mean free path length ***ℓ*** is chosen. If the relationship between ***d*** and ***ℓ*** is expressed with a scaling factor ***s***, then ***d*** = ***ℓ/s*** in the above equation gives:

$$R(r) = A\,s\frac{e^{-sr/\ell} + e^{-sr/(3\ell)}}{8\,\pi\,\ell\,r}$$

To determine ***s*** by curve fitting ***ℓ*** = 1 is taken since the shape of the Monte Carlo reference curve for a given ***A*** is independent of ***ℓ***. So for the curve fitting the following equation is used:

$$R_{\ell=1}(r) = A\,s\,\frac{e^{-sr} + e^{-sr/3}}{8\,\pi\,r}$$

In (**Figure 4**) we see the reflectance profiles ***R(r)*** computed for subsurface scattering after ideal diffuse surface transmission (cosine distribution) for surface albedos ***A*** between 0.1 and 0.9, computed with Monte Carlo simulation.

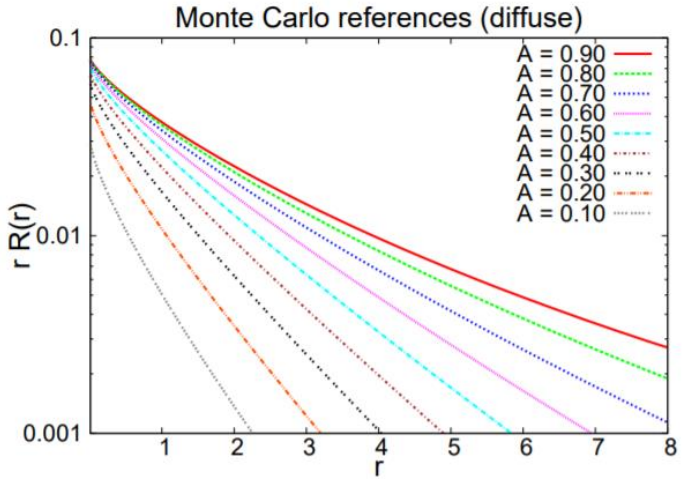

Figure 4: Reflectance Profile for Diffuse Transmission (Monte Carlo)

Manual curve fitting is used to find the expression for ***s*** that gives a good fit to the optimal values. For the following expressoin that gives the optimal ***s*** values we get only 3.9% average relative error with respect to the Monte Carlo references.

$$s = 1.9 - A + 3.5\,(A - 0.8)^2$$

Implementation

This project is implmented on a render engine I have built from scratch as a learning part of this module. I have closesly followed some of the principles in [3] for game engine architecture. As mentioned above, some critical implmentatin requirement to execute the demonstration of this project was a hierarchical Scene Graph with support for independent shader material manipulation. This was a a very difficult part to implment since it needed a lot of structural changes in the existing code, which can been seen in my GitHub repository that I maintain for this project.

Having successefully and efficently implmented the features requried with consistent frame rates, I moved on to the next part of the project, that of working on state-of-the-art PBR shader. As proposed in my presentation, I planned on implmenting the Principled BSDF shader introduced by Disney in their Siggraph course on Physically Based Shading. This model is used in Disney's Hyperion Renderer, also the latest version of Blender provides the Principled BSDF. I looked up on blender shader for the minimum shader parameters that I would need to implement to get a generic shader that can also render realistic human skin.

On completion of my experiments with blender, I narrowed down to seven shader parameters that I would require (**Figure 5**). Additionally, I added Occlusion maps into it along with some existing parameters, which complete my implementation of the Principled BSDF that can render human skin.

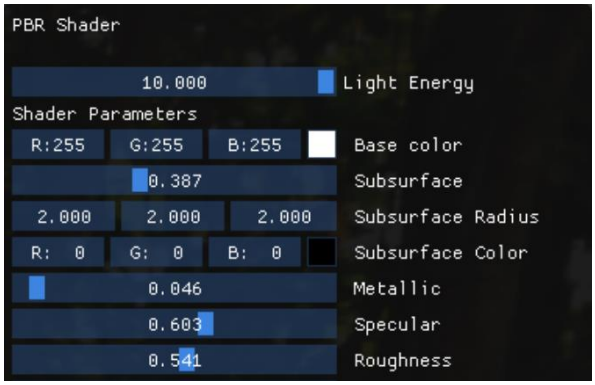

Figure 5: Implmented Shader Parameters for Principled BSDF

My implmentation also demonstrates the multi-scatter presented in [1], but does not account for the single scatter part, as a well described analytical sollution to single scatter was out of the scope of paper [1]. Since just the multi-scatter was not a very visually pleasing render, I implemented the model proposed in [2] which accounts for both single and multi scatter. Followed by the equations presented in [2], I looked up on how blender implments the same Christensen-Burely Subsurface Scattering. It was very challenging going through Blender's source code [4] and understanding its structure, but on figuring out few things, I found a customized implmentation of the model in C++. I implemented a similar model in GLSL.

In total, the final implementation has 3 variations of Subsurface scattering:

Diapole Model [1]:

This was the trickiest of the lot, since the equation used had multiple references to coeffiecient definition spread all across the paper.

The implmentation (**Code 1**) begins with computing three coefficients, σ'_t, which is the reduced extinction coefficient defined as $\sigma'_t = \sigma'_s + \sigma_a$, where σ'_s is the reduced scatter coefficient and the σ_a is the absorption coefficient. Following these coefficients, we compute α', which is the reduced albedo, defined as $\alpha' = \sigma'_s / \sigma'_t$. The values of the absorption coeefficent and the reduced scatter coefficient are given in a look up table in [1] (**Table 1**). All these computed values are plugged into the equations defined above for computing Diffuse Reflectance Profile.

```
vec3 dipole_bssrdf()
{
    vec3 r_scatter = vec3(0.74, 0.88, 1.01);
    vec3 absorption = vec3(0.032, 0.17, 0.4);
    float F = 0.444762840;
    vec3 r_extinction = r_scatter + absorption;
    vec3 r_albedo = r_scatter/r_extinction;
    vec3 effective_extinction = sqrt(3*absorption*r_extinction);
    vec3 D = 1.0/(3*r_extinction);
    vec3 dr = abs(1.0/r_extinction);
    float A = (1.0+F)/(1.0-F);
    vec3 dv = 4*D*A;
    vec3 zv = dr + dv;
    vec3 rp1 = ((effective_extinction*dr+1)*(exp(-effective_extinction*dr)/r_extinction*(dr*dr*dr)));
    vec3 rp2 = zv*((effective_extinction*dv+1)*(exp(-effective_extinction*dv)/r_extinction*(dv*dv*dv)));
    vec3 rp = (r_albedo/(4*PI))*(rp1+rp2);
    return rp;
    //vec3(0.44, 0.22, 0.1);
}
```

Code 1: Multi-Scatter Dipole Model.

Material	σ'_s [mm^{-1}]			σ_a [mm^{-1}]			Diffuse Reflectance			η
	R	G	B	R	G	B	R	G	B	
Apple	2.29	2.39	1.97	0.0030	0.0034	0.046	0.85	0.84	0.53	1.3
Chicken1	0.15	0.21	0.38	0.015	0.077	0.19	0.31	0.15	0.10	1.3
Chicken2	0.19	0.25	0.32	0.018	0.088	0.20	0.32	0.16	0.10	1.3
Cream	7.38	5.47	3.15	0.0002	0.0028	0.0163	0.98	0.90	0.73	1.3
Ketchup	0.18	0.07	0.03	0.061	0.97	1.45	0.16	0.01	0.00	1.3
Marble	2.19	2.62	3.00	0.0021	0.0041	0.0071	0.83	0.79	0.75	1.5
Potato	0.68	0.70	0.55	0.0024	0.0090	0.12	0.77	0.62	0.21	1.3
Skimmilk	0.70	1.22	1.90	0.0014	0.0025	0.0142	0.81	0.81	0.69	1.3
Skin1	0.74	0.88	1.01	0.032	0.17	0.48	0.44	0.22	0.13	1.3
Skin2	1.09	1.59	1.79	0.013	0.070	0.145	0.63	0.44	0.34	1.3
Spectralon	11.6	20.4	14.9	0.00	0.00	0.00	1.00	1.00	1.00	1.3
Wholemilk	2.55	3.21	3.77	0.0011	0.0024	0.014	0.91	0.88	0.76	1.3

Table 1: Measured parameters for some common materials.

The Christensen-Burley model [2]:

This is a very simple implementation (**Code 2**) of the equation presented above, where in the function takes in as argument the artistic parameter (radius / wavelength [RGB]) and the roughness.

```
vec3 principled_bssrdf(vec3 radius, float roughness)
{
    vec3 exp_r_3_d = exp(-roughness/ (3.0f * radius));
    vec3 exp_r_d = exp(-roughness/ radius);
    vec3 bssrdf = (exp_r_d + exp_r_3_d) / (8.0f * PI * radius * roughness);
    return bssrdf;
}
```

Code 2: Christensen-Burley Model Implementation [2]

The Christensen-Burley model Blender Source [4]:

This is a very similar implementation (**Code 3**), with the same function arguments, just with some customization to the equation as seen in blender source code. This has hardly any visual difference on the final render.

```
vec3 principled_bssrdf(vec3 radius, float roughness)
{
    vec3 Rm = BURLEY_TRUNCATE * radius;
    if (radius.x > Rm.x)
        return vec3(0.0f, radius.y, radius.z);
    if (radius.y > Rm.y)
        return vec3(radius.x, 0.0f, radius.z);
    if (radius.z > Rm.z)
        return vec3(radius.x, radius.y, 0.0f);
    vec3 exp_r_3_d = exp(-roughness/ (3.0f * radius));
    vec3 exp_r_d = exp_r_3_d * exp_r_3_d * exp_r_3_d;
    vec3 bssrdf = (exp_r_d + exp_r_3_d) / (4.0f * radius);
    return bssrdf * (1.0f/BURLEY_TRUNCATE_CDF);;
}
```

Code 3: Christensen-Burley Model Implementation [4]

Both the implmentation of Christensen-Burley Model uses a common function (**Code 4**) to find the optimal ***s*** value as discussed above.

```
vec3 burley_mfp_to_d(vec3 radius, vec3 albedo)
{
    vec3 l = 0.25f * (1.0/PI) * radius;
    vec3 s = 1.9f - albedo + 3.5f * ((albedo - 0.8f) * (albedo - 0.8f));
    return l/s;
}
```

Code 4: Optimal ***s*** value

Results

The final results have been rendered on my personal laptop with a Nvidia GTX 980m. All models tested run above 250 FPS, which is a limit my Nvidia card has while running OpenGL, not known to me why. But overall the performce is uncompromised with all implementations.

For the demonstration, primarily there are two models used [5][6].

Emily:

Result 1: No Subsurface Scattering

Result 2: Christensen-Burley Subsurface Scattering [2]

Result 3: Multi-Scatter Dipole Subsurface Scattering [1]

Infinite Realities Head Model:

Results with multiple parameters of Christensen-Burley Model.

Result 1: No Subsurface Scattering

Result 2: Christensen-Burley Subsurface Scattering [2]

Result 3: Christensen-Burley Subsurface Scattering [2]

Result 4: Christensen-Burley Subsurface Scattering [2]

Result 5: Christensen-Burley Subsurface Scattering [2]

Result 6: Multi-Scatter Dipole Subsurface Scattering [1]

Improvement and Limitation

The Dipole model implemented does not use a single scatter component in the BSSRDF model introduced in [1], since it was out of scope in the original paper. The paper only gives an equation without explaining it in depth, citing it to another paper for reference. In my knowledge, I understand the entire equation, except the numerator, which I believe is the phase function used in Radiative Transport Equation.

$$f_r^{(1)}(x, \vec{\omega}_i, \vec{\omega}_o) = \alpha F \frac{p(\vec{\omega}_i' \cdot \vec{\omega}_o')}{|\vec{n} \cdot \vec{\omega}_i'| + |\vec{n} \cdot \vec{\omega}_o'|}$$

Single Scatter Unexplained Equation in [1].

References

[1] Jensen, H. W., Marschner, S. R., Levoy, M., & Hanrahan, P. (2001). A practical model for subsurface light transport. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH 01. doi:10.1145/383259.383319

[2] Christensen, P. H. (2015). An approximate reflectance profile for efficient subsurface scattering. ACM SIGGRAPH 2015 Talks on - SIGGRAPH 15. doi:10.1145/2775280.2792555

[3] Gregory, J. (2019). Game engine architecture. Boca Raton ; London ; New York: CRC Press.

[4] Blender3D source code: https://developer.blender.org/diffusion/B/

[5] Emily Model: http://gl.ict.usc.edu/Research/digitalemily/

[6] Head Model: Infinite Realities http://ir-ltd.net/irs-digital-journey-prt2/