Subsurface Scattering and Spectral Rendering

Jacky Sun

Abstract

This is the final project report for CMU 15668 Physics Based Rendering. In this report, I will show my exploration of rendering subsurface scattering materials using BSSRDF and rendering spectral effects such as dispersion. I will discuss the methodologies used to implement both topics and show the resulting images. The implementation is based on the rendering infrastructure DIRT which I had been using throughout the semester.

1 Introduction

In a lot of the modern renderers, subsurface scattering (known in the game industry as the SSS) is used to simulate appearance of objects that has light scatteing inside, such as human skin, wax and other semi-translucent materials. Renders of such materials often produces more interesting and believing result than simple reflection and refraction surfaces. Therefore I decided to research into this topic and take a look at the mathematics and physics foundation behind subsurface scattering.

I also explored spectral rendering to produce dispersion effect which cannot be obtained through RGB color calculations. In order to simulate light transport in a more physically authentic way, I have to take into account the visible spectrum to human eye, ranging from 400 to 700 nm, and use a new color evaluating structure based on spectral power distributions.

To speed up the rendering process a little, I implemented naive cpu parallelism with the help of OpenMP. Throughout the project, I encountered various problems and new interesting topics, and I will discuss them in detail in later sections.

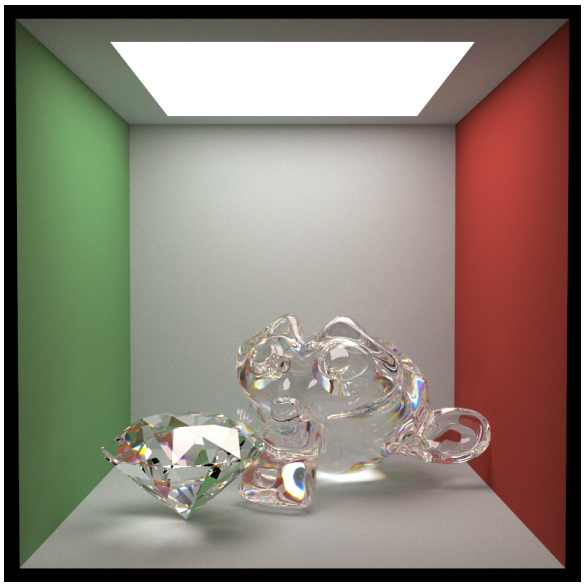

Figure 1: *Submission for the Rendering Competition*

2 Background (References)

While the BRDF models assumes that light enters and leaves a surface at the same location, [Jensen et al. 2001] introduced the method of BSSRDF to describe the light transport from a point on surface to another point on the same surface through a path below the surface. Such model is able to produce image that capture the appearance of certain translucent materials such as marble, skin, milk, etc.

[Evans and McCool 1999] introduced single wavelength sampling and stratified wavelength sampling techniques that is able to produce spectral effects like dispersion. A much more efficient sampling method called hero wavelength sampling [Wilkie et al. 2014] leverages multiple importance sampling and reduces the sampling cost when evaluating different paths. I did not get to implement HWS because I failed to find reference for wavelength dependent microfacet BTDFs.

3 Subsurface Scattering

We can think of the BSSRDF a more complex version of the BRDF, as it takes in an extra parameter indicating the surface point where incident light arrives at the surface. The rendering equation therefore becomes:

$$L_o(p_o, \omega_o) = L_e(p_o, \omega_o) + \int_A \int_\Omega L_i(p_o, o_\omega) * S(p_o, p_i, \omega_o, \omega_i) * \cos(\theta_i) d\omega_i dA$$

The integration domain is extended to the whole surface area of the geometry. A truly accurate BSSRDF model heavily relies on the exact shape of geometry it is used on, and could get over complex to implement. Therefore a lot of assumptions are needed to approximate result and simplify calculations.

3.1 Separable BSSRDF

Following PBRT and [Jensen et al. 2001], I used the method of separable BSSRDF to separate the evaluation of S into 3 parts:

$$S(p_o, p_i, \omega_o, \omega_i) \approx (1 - F_r(\cos(\theta_o)))S_p(p_o, p_i)S_\omega(\omega_i)$$

With three parts in this model, the first part is responsible for accounting for the energy loss during light leaving the surface from inside. $S_p(p_o, p_i)$is mainly responsible for approximating the falloff factor due to the distance between p_i and p_o. Like the first part, the last one is for evaluating light energy loss due to light coming in through the surface. We are using the diffusion approximation here, assuming that the scattering media is isotropic.

Just like how a BRDF model is implemented, the 2 main tasks for implementing a BSSRDF model is figuring out how to evaluate the function, and how to importance sample both a direction and a position.

3.2 Evaluating BSSRDF

The main work of evaluating the BSSRDF lies in the S_p term. To simplify things even more, $S_p(p_o, p_i)$ only considers the distance of the two input positions, instead of their actual positions on the geometry. We can then just set up a diffusion function to evaluate S_p. This is done by precomputing a BSSRDF Table during material instancing, with 100 albedo samples and 64 distance samples. The method here is the Photon Beam Diffusion Technique, which I have not been able to understand completely. I had to reuse a lot of PBRT's source code in the actual implementation.

3.3 Sampling BSSRDF

Importance sampling the BSSRDF involves two things: sampling an incoming direction and sampling a position where incoming light enters the SSS object. Sampling the direction is no different than sampling a BRDF. I'm using the existing microfacet model to sample ω_i after we get the sampled position.

Sampling the position is much more complex. We first do a uniform sample on a disk above the normal direction, and then from that point on the disk, we perform another ray-scene intersection check along the negative normal direction. However, this method is not robust as it assumes the geometry to be flat everywhere. In certain cases the projection ray could be almost perpendicular to the surface normal of the new intersection point. This is addressed by randomly selecting an axis along which the disk is placed. Following the implementation of PBRT, I arbitrarily set the probability of choosing the surface normal and its two tangent vectors to 0.5, 0.25 and 0.25. The pdf is a weighted average of the pdfs evaluated on all 3 axis. The projection ray is set to a maximum radius since the intersections outside that radius would contribute very little because of the S_p fall off. We then store all the intersections, and randomly choose an intersection to set as our sampled position. We also have to divide by $1/numberofintersections$ to account for the probability of choosing that intersection point.

The example below shows how the sss material displays a more waxy and soft appearance compared to a diffuse BRDF material.

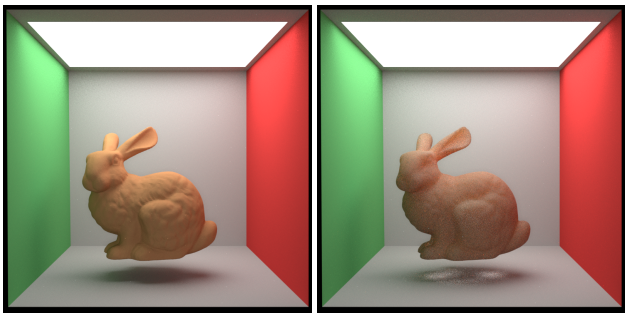

Figure 2: *Left: without sss Right: with sss (profile evaluated with skin albedo)*

3.4 Using measured albedos

Now I can use various different measured σ_a and σ_s values provided by [Jensen et al. 2001] to simulate different appearances of subject.

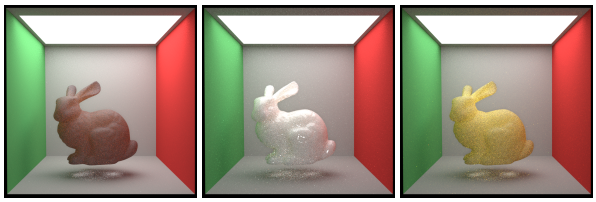

Figure 3: *From left to right: chicken, marble and potato*

4 Spectral Rendering

4.1 Representing colors with SPD

Instead of using RGB colors, we need a way to represent colors in the form of spectral power distributions(SPD). Following PBRT's implementation of the spectrum class, I now store radiant power across a vector of visible wavelengths. According to PBRT, intervals of 5 nm are more than enough to capture most of the spectral effects. For computaion efficiency, I'm using intervals of 15 nm. The spectrum calculations are very similar to vector calculations, as we are essentially just storing a much wider vector than Color3f. For existing RBG based materials, we need to upsample them into SPDs. The method involves storing the arbitrary SPDs of 7 different colors on the spectrum and remap each of their RGB contribution to a color onto the spectrum space. This also means now the BRDFs are wavelength dependent. However, it is necessary to convert the spectrum back to RGB for display. The CIE XYZ standard of conversion is used.

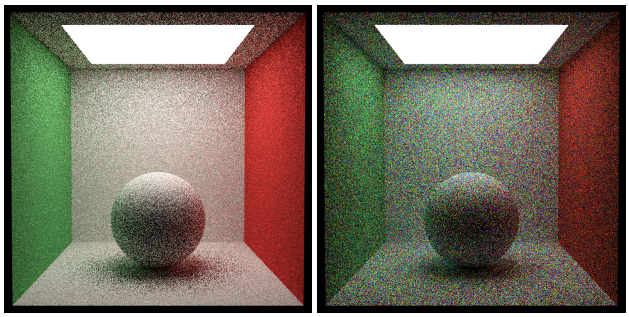

Figure 4: *Left: 1 spp with RGB. Right: 1 spp tracking spectrum*

4.2 Wavelength dependent refraction

The main cause of dispersion effect is the fact that index of refraction is different for light of different wavelengths. This means light rays of different wavelengths will travel differently along paths that contains refractions. Many work has been done trying to model the change of ior with respect to wavelengths, such as The Sellmeier Equation or the Cauchy's Equation. In my implementation I'm using an even more simpified version of Cauchy's Equation:

$$\eta(\lambda) = A + \frac{B}{\lambda^2}$$

where A is the original index of refraction, and B is a dispersion factor that can be tweaked. With this, I am now able to modify the sampling method for dielectric material to account for wavelengths associated with a ray.

4.3 Microfacet refraction

This is an extra feature that is implemented during the project. Since I have previously implemented this in another toy renderer it is straightforward to integrate into DIRT.

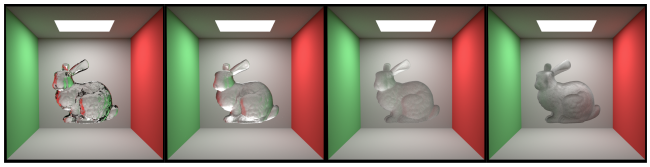

Figure 5: *Renderings of microfacet refractive materials. From left to right, the roughness is 0, 0.02, 0.2, 0.5*

4.4 Hero wavelength sampling

Hero wavelength sampling [Wilkie et al. 2014] is an efficient spectral sampling technique that can greatly reduce chromatic noises. It first picks a hero wavelength by uniformly sampling the spectrum,

and then fixes a certain number of other wavelength samples using a shifting function:

$$\lambda_i = (\lambda_H - \lambda_{min} + \frac{i}{c}(\lambda_{max} - \lambda_{min})) \mod (\lambda_{max} - \lambda_{min}) + \lambda_{min}$$

This distributes all the wavelength samples relatively evenly across the spectrum. Then, the path is generated only with the hero wavelength. Since every other wavelength need to be evaluated on this hero path, a spectral BSDF for rough material is needed. Unfortunately I am not able to find a source for such BSDF and did not include HWS in the implementation. The last part of HWS is setting each wavelength sample as the hero and combine each path's contribution with MIS weights.

4.5 Stratified wavelength sampling

The sampling method I ended up using is stratified wavelength sampling. Instead of choosing a wavelength based on a uniform random variable, I manually assign wavelengths to cover the whole visible spectrum. Then each path would only contribute to the final SPD on the same wavelength portion. This does mean that for paths without refractive surfaces the extra samples created are completely wasteful, making the rendering process really inefficient. Nevertheless dispersion effects can be easily observed in an acceptable amount of rendering time.

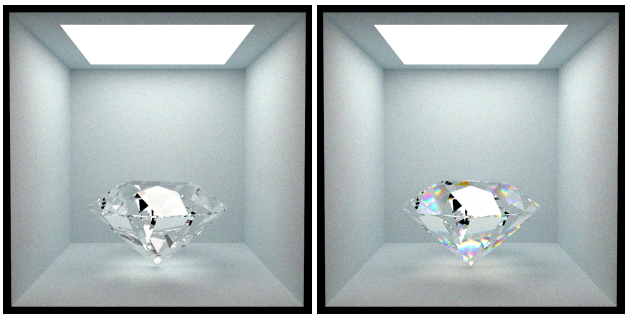

Figure 6: *Comparison of a diamond with and without Spectral Rendering. (Dispersion is exaggerated)*

5 Problems and Challenges

For subsurface scattering, the amount of related methodologies used in the implementation is quite out of my expectations. Since I'm following the exact implementations from pbrt, there is still parts where I do not completely understand, such as the technique to populate the bssrdf table. In the summer I plan to build my own renderer from scratch, and I hope to integrate my own version of subsurface scattering which I have more control over.

For spectral rendering, I wasn't able to find a reference to implement wavelength dependent BSDFs, so I could not implement hero wavelength sampling. Also, my current spectral renderer can only handle either specular or pure diffuse objects.

I did not have time to merge my two integrators together to be able to render sss material inside the spectral framework. I plan to address this in the summer by following a similar approach to PBRT. I will use spectrum as the color measurement throughout the system, and decide at compile time whether to use the sampled spectrum(for spectral rendering) or the RGB spectrum(just like Color3f).

References

EVANS, G. F., AND MCCOOL, M. D. 1999. Stratified wavelength clusters for efficient spectral monte carlo rendering. *International Code Council*.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*.

WALTER1, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. *Eurographics Symposium on Rendering*.

WILKIE, A., NAWAZ, S., DROSKE, M., WEIDLICH, A., AND HANIKA, J. 2014. Hero wavelength spectral sampling. *Computer Graphics Forum*.