Real-Time Rendering of Translucent Materials in a Deferred Render

Sam Oates

Computer Games Programming, Teesside University

**The rendering of translucent materials is a complicated procedure to produce accurate results, with many techniques not suitable for real-time applications and or implementations. This report aims to cover and explore a plausible implementation of translucent materials and sub-surface scattering in a real-time implementation. Mainly focusing on the highest possible performance of the application, rather than the flexibility of the solution.**

## Introduction

The world is made from many different types of translucent materials, from leaves and paper to skin and milk. Translucency occurs when light enters a given object and scatters based upon the physical properties of the objects material. The technical name for this physical phenomenon is subsurface scattering.

## Existing Techniques

Multiple implementations have been used in the past to try best reproduce subsurface scattering accurately and efficiently. The bidirectional reflection distribution function (BRDF) introduced by Nicodemus, considers the case where light striking a surface point gets reflected at the same point. Since, Henrik Wann Jensen proposed the use of the more generic bidirectional surface scattering distribution function (BSSRDF).

Current graphical hardware does not meet the standard required for highly computational implementations. Thus, for a real-time implementation approximations must be made. A simple approximation of light scattering comes in the form of wrap lighting, which is a somewhat crude approximation of the Oren-Nayar lighting model. However, these results are not accurate.

Jorge Jimenez proposed that to develop a practical model, the algorithm should be calculated in screen space rather than the traditional texture space. This therefore reduces the complications caused by translucently not being a post-process. However, calculations within screen space are hindered by not attaining the same amount of information found within traditional 3d and texture space algorithms.

## The bidirectional surface scattering distribution function versus the bidirectional reflectance distribution function

Henrik Wann Jensen is commonly attributed to the advances in using BSSRDF in the field of translucency and its taking of the lead in the field, over the traditionally used bidirectional reflectance distribution function (BRDF).

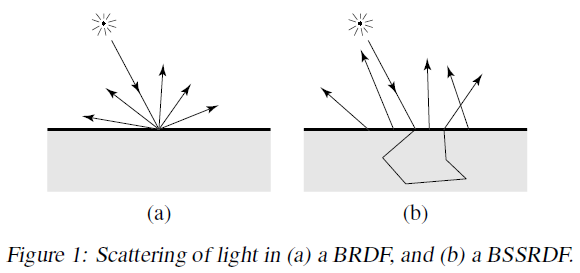
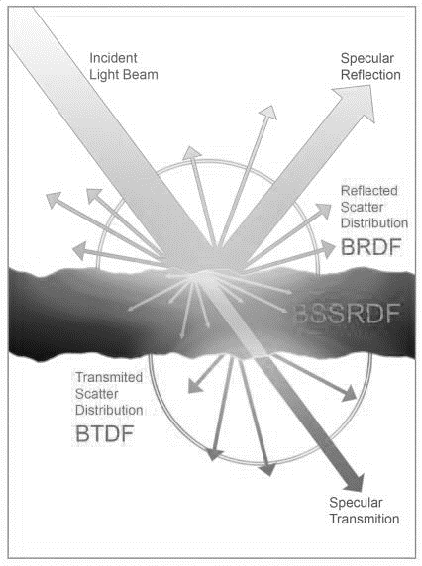
The main difference between BRDF and BSSRDF is that, BSSRDF can describe the light between and two given rays that hit the surface, whereas BRDF assumes that light hitting a surface enters and leaves at the same position (Figure 1).

Figure 1: Light scattering (a) BRDF and (b) BSSRDF

As one would presume the bidirectional surface scattering distribution function is much more accurate to real life, however in a real-time simulation it is usually more if not too expensive compared with the simpler bidirectional reflectance distribution function. However, this does not mean all elements of BSSRDFs should be discarded.

So in conclusion, current hardware doesn’t have enough spare resources to focus on BSSRDF implementations of translucency. It is possible to implement BSSRDF, but with a large expense which could be used elsewhere in the rendering pipeline to greater effect.

A combination of multiple distributions functions allows the best of both worlds (Figure 2). [More about how they are combined, also wtf is BTDF]

## Resources we can’t afford

Some implementations use additional depth map textures and or the use of texture space blurring, both of which require additional resources. Depth map textures will use additional memory (of which we are already using a lot of due to the differed render implementation) and the use of texture blurs will require significantly more computational power, again something we don’t have spare.

As with the majority of computer graphics, tricking the user where needed is the best option for both performance and quality. To do such; light that travels inside a given object must be influenced by the changing thickness of the object. Along with this it show some light and view-dependent attenuation.

The advantage with making the translucency method relatively cheap is that we can use it more around the scene not limiting the scene, along with not having different render qualities between cut scenes and real-time game play.

## Implementation of fast real-time translucency

To start with, we will look at basic shapes (spheres, cubes, etc…), the reason for this being is we can take advantage of the radial diffusion properties of lights. We can use distance attenuated regular diffuse lighting, combined with the distance attenuated dot product of the view vector and an inverted light vector, we can simulate basic light transport for basic shapes.

Figure 2: The combination of multiple distribution functions

However, for more complicated object we must take the amount of time light exists within the object (commonly calculated by evaluating the thickness of the object). This can be calculated via two main methods.

Option one, the use of additional depth maps, however as previously stated we wish to avoid this as the additional memory is too great of an overhead.

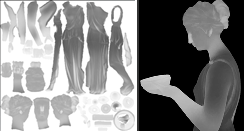
Option two is too pre-compute the depth in a local variation map. The map contains approximations for the thickness of the object, dark values represent thick opaque sections of the model, where as bright values represent thin translucent parts of the model (Figure 3).

Figure 3: An example of a local variation map

The creation of these maps can be done manually, however since the usage of this translucency method is such that in theory it can be used throughout a large scene, manual creation would be a long tedious process.

An alternative to manual creation of variation maps is to use a normal inverted calculation of ambient occlusion. This can be performed by most modelling and or render software. This works as ambient occlusion calculates how much light reaches a surface point, since the normals of the model are inverted this gives us the lighting information for inside the shape, giving a good representation of the thickness of the model (Figure 3).

We use this information to help improve the calculation of light transport, where results lay in-between real subsurface scattering and distance-based attenuation. This allows us to use a single distance value for distance based attenuation.

## Applying the subsurface scattering using our data