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, via the means of comparing and discussing current techniques used, and said techniques origins.

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.

Background

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 [9] in 1977, considers the case where light striking a surface point gets reflected at the same point.

One of the most influential people in the field of subsurface scattering is Henrik Wann Jensen, whose paper *A Practical Model for Subsurface Light Transport* [1] proposed that subsurface scattering in translucent materials should take into account both single scattering of light rays and multiple scattering. Jensen proposed the use of a bidirectional subsurface scattering distribution function (BSSRDF), rather than the traditional BRDF.

However, although more accurate, BSSRDF requires more sampling and is more expensive than BRDF.

Jensen and Buhler then presented a faster approach to subsurface scattering in their paper *A Rapid Hierarchical Rendering Technique for Translucent Materials* [2] in 2002. Their approach uses a two-pass rendering technique and leans towards use with a forward rendering system rather than a deferred rendering system.

The two-pass approach consists of the following. The first pass where we compute the irradiance at selected points on the objects surface, and the second pass where we evaluate a diffusion approximation using the pre-computed irradiance samples. Jensen and Buhler's approach is substantially faster than sampling directly from the BSSRDF as it only has to evaluate the incident illumination once at a given surface location.

However, although Jensen and Buhler's approach is faster than previous translucent material implementations, it is far from real-time, requiring a 5 minute average render time.

Moving towards an interactive real-time solution Hendrik Lensch presented a paper for rendering translucent objects where the view point and illumination could be modified in real-time. The solution proposed in Lensch's *Interactive Rendering of Translucent Objects* [4] in 2002 produced results of on average 5 frames per second when rendering a model of 8,500 vertices, which in real-time terms is far from the desired frame rate.

Advancing more on Jensen's work with bidirectional subsurface scattering Craig Donner extended upon the previous theory on diffusion approximation in his Ph.D. thesis *Towards Realistic Image Synthesis of Scattering Materials* [5] in 2006. This theory used a number of multi-pole sources rather than a dipole source to calculate reflectance and transmittance.

Bidirectional Reflection Distribution Function

Four categories can be used to divide surfaces which reflect light without surface scattering; perfect specular, glossy specular, diffuse and retro-reflective surfaces.

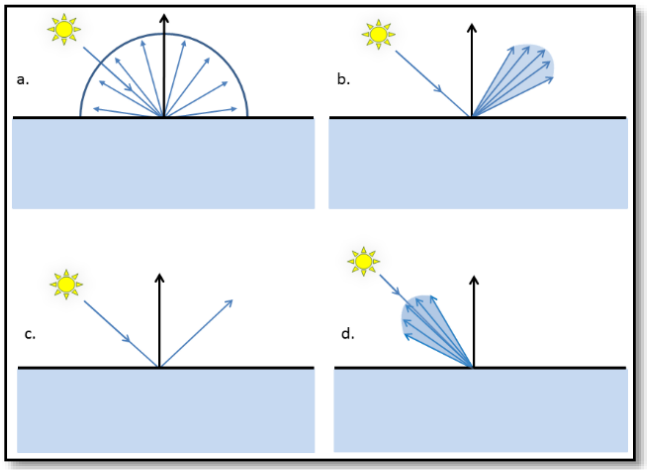

***Figure 1**: The four basic reflection types: a) diffuse; b) glossy specular; c) perfectly specular; d) retro-reflection.*

The BRDF function describes how light is reflected off a given opaque surface. The function is multi-dimensional, dependent on the material properties of the shaded point and the point of incidence, and the geometry of the scene.

Bidirectional subsurface scattering reflection distribution function

The bidirectional subsurface scattering reflection distribution function (commonly referred to as BSSRDF) is a function which describes the subsurface light transport in translucent materials as the ration between the differential radiance at a given incidence point xi and the differential irradiance exit point xo seen in *Figure 2*.

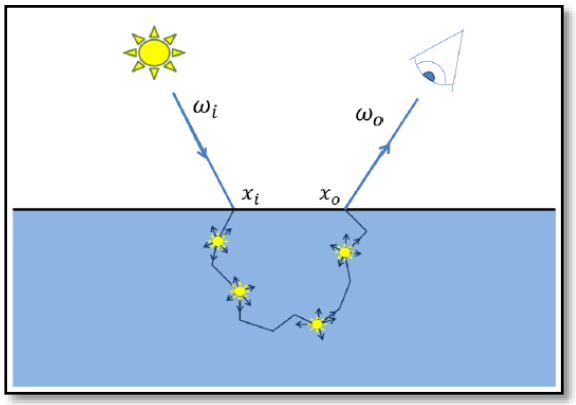

***Figure 2**: Subsurface Scattering*

The (analytical) equation that describes BSSRDF is as follows;

$$S(xo, wo, xi, wi) = \frac{\delta Lo(xo, wo)}{\delta \sigma(xi, wi)}$$

However, should $xi \equiv xo$ (the incidence point and the exit point be the same), the BSSRDF can be reduced to using BRDF.

Using Shadow Maps for Translucency

Translucent Shadow Maps [3] by Dachsbacher and Stamminger in 2003 proposes to extend the use of the binary shadow map lookup to the use of a shadow map filter which implements sub-surface scattering. A pixel in the translucent shadow map stores the

irradiance entering the object as well as storing the traditional depth (and therefore the 3D space of the sample) and the surface normal, seen in *Figure 3*.

***Figure 3**: Contents of a translucent shadow map. Left: Irradiance, Center: Depth, Right: Surface Normal*

As with Jensen and Buhler's solution this process requires multiple render passes. In the first pass the translucent shadow map data is computed. Two render targets are used; in the first render target the amount of light penetrating the materials boundary (in the RGB component) is stored. The second render target contains the distance from the visible surfaces to the light source (the light space depth-buffer) which parallels with a standard shadow map. Along with this the two dimensional projection of the surface normal in regards to the orthogonal lights direction is stored as colour. As visible surfaces in the translucent shadow map are facing the light, we can use the stored information to calculate the surface normal.

In the second render pass, the local and global response for all surface points in the camera view are evaluated. Proceeded by the point's coordinate being transformed into light space to calculate its texture coordinates in the previously created translucent shadow map as well as its distance to the light. A filter is then applied with different pre-computed weights to integrate the contribution from all sample points in the translucent shadow map. All these computations are performed in vertex and fragment programs respectively, as well as this solution relying on mip-map textures. Results from this solution averaged at 40 frames per second when rendering 5,100 vertices.

Using shadow maps is a liable solution for use with a deferred render, depending on hardware limitations as it does require extra resources in the gbuffer. As well as the potential hardware limitations, Dachsbacher and Stamminger's solution is restricted to directional light sources.

Texture space sampling and point splatting

Shah presented a different solution in their paper *Image-Space Subsurface Scattering for Interactive Rendering of Deformable Translucent Objects* [6] by the means of texture space sampling and point splatting.

Shah's solution uses two representations of the scene, one from the lights point of view the other from the cameras perspective. The scene is rendered twice to get the points at which light is incident and the points can be seen by the camera. The geometry visible by the light is then sampled for Irradiance and then stored in a texture, as well as screen aligned quads being positioned at the centre sample points. These quads are then used to shade the points visible to the camera, using the diffusion approximation originally proposed by Jensen [1] or Donner's [5] adaptation. Resulting in light transport toward the viewer being approximated from sampled irradiance.

Pre-computation of geometry thickness

Colin Barré-Brisebois [8] presented at GDC (Game Developers Conference) 2011, about how translucency and subsurface scattering is implemented in the deferred renderer Frostbite 2.

Barré-Brisebois proposed that similar to other translucency solutions, BSSRDFs are too expensive on current hardware, and

therefore should be used alongside the simpler BRDF, as seen in Figure 4.

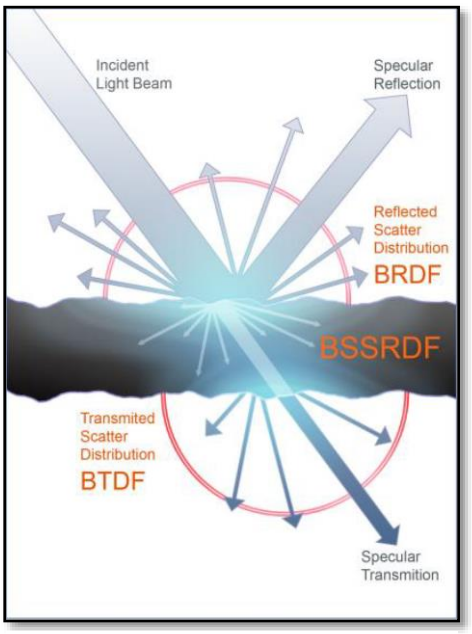

***Figure 4**: BSSRDF, BRDF and BTDF*

The solution by Barré-Brisebois wanted to create realistic looking translucency whilst avoiding the use of addition overheads (additional memory and or additional computational time) caused by additional depth maps and or texture-based blurs, commonly used in other real-time solutions.

The basics of the solution simply wanted to combine simpler techniques to approximate translucency in the scene. Two basic rules where suggested; Light passing through the translucent object should be influenced by the objects varying thickness, as well as some view and light dependent diffusion and attenuation. With said rules, translucency can be approximated creating a convincing render, as well as the implementation being cheap (resource wise) resulting in mass use of the technique.

The technique is required to know the thickness of a translucent object light should pass through. Rather than using a depth map to calculate this, Barré-Brisebois proposes that the depth should be pre-computed by external software (such as 3ds Max or Maya). By inverting the normals of a models geometry and rendering ambient occlusion, the texture created approximated how much light travelling inside the object would get occluded at the point it exits the object.

The resulting texture will show two types of areas: Lighter white areas of translucency and darker areas of opaqueness, seen in Figure 5.

***Figure 5**: An Example of a pre-computed local thickness map.*

The basics of computing the translucency can be described by the following equation:

$$LAttenuation(N.L + V.(-L))$$

The implementation itself contains four variants specified per light as well as the local thickness maps value at the sampled point, an overview of the code can be seen in Figure 7. The four variants are as follows;

- Scale – A scale applied to the light passing through the translucent object.
- Power – The Power of the direct translucency, breaking continuity.
- Distortion – Shifts the surface normal to break continuity.
- Ambient – The ambient value of the translucent object

```
half3 vLTLight = vLight + vNormal * fLTDistortion;
half fLTDot = pow(saturate(dot(vEye, -VLTLight)), iLTPower) * fLTScale;
half3 fLT = fLightAttenuation * (fLTDot + fLTAmbient) * fLTThickness;
outColour.rgb = cDiffuseAlbedo * cLightDiffuse * fLT;
```

***Figure 7**: Code snippet from the fragment program*

This solution is far from mathematically correct, but is convincing and flexible enough to suit for current graphical hardware, however this technique does not work with concavities nor animated objects.

As previously stated this technique is designed for a deferred rendering pipeline, using up one channel in a gbuffer (or three channels should the translucency also contain the colour of the translucent object, an example can be seen in Figure 6).

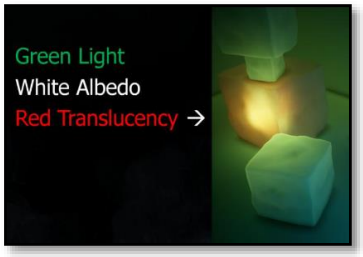

***Figure 6**: An example of a coloured local thickness map.*

Conclusion

As with most computer graphics the implementation of a feature lies heavily on its uses.

With my past experience working on a tool and engine system, the solution proposed by Barré-Brisebois, and used in the Frostbite engine seems like the best solution for a large scale engine and tool system, allowing artists to easily change the translucency and appearance of an object.

That being said, the implementation that I am likely to design will not have any pre-computation (due to not having a tools system in place), and should instead calculate translucency from data available within the scene. Tending me to believe that depth maps could be a global solution for all scenes, yet be limited by only working with directional lighting.

To maintain a high level of performance, whilst being flexible a solution would be to use the approximations for BSSRDF suggested by Barré-Brisebois but using a depth map to calculate the local thickness of the scene.

This technique would have the additional computational and resource overheads required when using additional depth maps, but has the flexibility to work with any scene without pre-computation of the geometry. This solution would result in not being able to adjust the local thickness maps manually, but the colour of the translucency could be approximated from the material applied to the translucent object.

References

[1] Jensen, H.W. (2001) 'A Practical Model for Subsurface Light Transport', SIGGRAPH.

[2] Jensen, H.W. and Buhler, J. (2002) 'A rapid hierarchical rendering technique for translucent materials', ACM Transactions on Graphics, 21(3).

[3] Dachsbacher, C. and Stamminger, M. (2003) 'Translucent Shadow Maps', Eurographics Symposium on Rendering.

[4] Lensch, H. (2002) 'Interactive Rendering of Translucent Objects', Pacific Graphics, 214-224.

[5] Donner, C. S. (2006) 'Towards Realistic Image Synthesis of Scattering Materials', PhD thesis, University of California, San Diego.

[6] Shah, M. A. (2009) 'Image-Space Subsurface Scattering for Interactive Rendering of Deformable Translucent Objects', IEEE Computer Graphics and Applications.

[7] Rohat, M. (2012) 'Real-time Rendering of Translucent Materials', Master's thesis, Technical University of Denmark.

[8] Barré-Brisebois, C. (2011) 'Approximating Translucency for a Fast, Cheap and Convincing Subsurface-Scattering Look'. [Shown at Games Developer Conference: San Francisco]

[9] Nicodemus, F. E. and Richmond, J. C. (1977) 'Geometric considerations and nomenclature for reflectance'. National Bureau of Standards (US), Monograph 161.