Subsurface Scattering

Max Curran

Abstract

1 Introduction

Jensen et al. [2001] proposed a method for rendering the effects of subsurface scattering by using the Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF). Subsurface scattering is caused when light scatters one to multiple times in a semi-translucent object, and is very important in a variety of different objects, such as milk or skin. Without this effect, rendered objects have a rough and hard computer generated feel to them.

This effect is simulated in a raytracer by casting samples over the surface of the object and into the object. Samples are split into approximating the effects from one scattering event and multiple scattering events. Single scattering events are directly approximated by tracing random paths of single scattering events and computed the amount of radiance that goes along the path. Multiple scattering events are approximated using the diffusion approximation. This approximation allows for any two rays on the surface to be related.

This method differs from previous methods because it approximates the BSSRDF for subsurface scattering. Previous methods focused on approximations of the Bidirectional Reflectance Distribution Function (BRDF). The BSSRDF describes the amount of light transfered between any two rays. The BRDF is a specialization of the BSSRDF and requires that the two rays must share an endpoint. This requirement enforces the assumption that all light entering a point leaves that same point. This assumption is not true in the physical world, and therefore computers cannot create physically accurate scenes by using only the BRDF. The BSSRDF can describe more complex behavior of light, but its generality makes it more complex to compute. Jensen et al. [2001] only approximate the BSSRDF in cases of subsurface scattering, thereby avoiding the costly computation of the BSSRDF everywhere.

2 Project

For this project I have implemented a raytracer that approximates both single and multiple scattering using the methods proposed by Jensen et al. [2001]. The raytracer also supports soft shadowing, anti-aliasing, and reflection. These features were included in order to improve the overall quality of the rendered image. The raytracer is accompanied by a previewer which displays a rasterized version of the scene. This allows for the user to move the camera and select a good viewpoint to perform raytracing from. There is also a visualization tool, which allows for rays used to calculate one pixel to be viewed along with the rasterized scene. There are also four types of primitives- sphere, cube, cylinder, and cone- which can be added to a scene along with a mesh made of quadrilaterals. These are the primary features of the raytracer I implemented for my final project. Below is an image rendered using the raytracer.

2.1 Soft Shadowing

One of the basic features of the raytracer is soft shadowing. Soft shadowing is caused by area light sources, and occluders. With point light sources any point in the scene is either lit or occluded. This creates hard boundaries which the lighting has a discontinuity. This hard discontinuity is unrealistic as point light sources are not physically possible. All light sources in the physically world must possess an area. This area causes every point in the scene to be lit by some percentage of the light source, depending on what portion of the light source is not blocked by occluders.

For point light sources, a raytracer would require only one ray to determine if a point in the scene is occluded or not. This would be done by casting a ray from the point towards the point light source, if the ray intersects any other objects before it hits the light, then it is lit. Otherwise, the point is occluded. For area light sources, casting only one ray per point to determine lighting effects causes the area light source to essentially become a point light source. In order to accurately render the effects of soft shadowing multiple samples must be cast in order to calculate the percentage of the light source that is visible at to a point. This still can lead to errors since a finite number of locations are being tested. Ideally the percent of the area that is visible would be measured, but by using a finite number of samples, only the percentage of samples that are visible is calculated.

In order to make the percent of samples that are visible as close to the percent of area that is visible, we need to choose samples carefully. One method could be to cast samples to a set grid of points on the light source. This ensures that the entire light is sampled. However, this method just approximates the area light source to a set of point light sources arranged in a grid. Also casting samples towards any predetermined set of points on the light source will approximate the area light source to a point light source. This will create numerous hard boundaries, which is not good. With an extremely large number of samples, it is possible to achieve correct results with this method, but this would require a lot of computation.

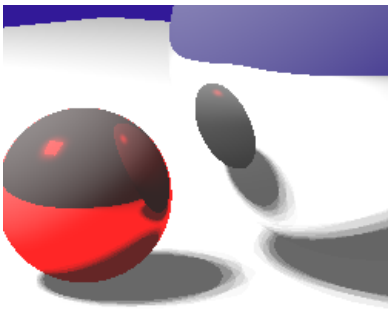

Figure 1: *Grid sampling of shadows*

The solution to this problem is to use some random process in determining where to cast samples. One method would be to cast to a random point on the light source. In this process it is possible that for adjacent pixels the sample distribution could be drastically different. This is unlikely, but with a larger number of pixels, this happening just once is possible. In order to decrease the likelihood of this happening more samples can be randomly drawn. Another method, called jittered sampling, would be to break the light into a grid and choose one random sample in each subsection of the light. This ensures that the distribution of samples cannot be as drastically different for any two random selections.

The raytracer uses jittered sampling in order to render the effects of soft shadowing. This method is still not perfect, and more samples

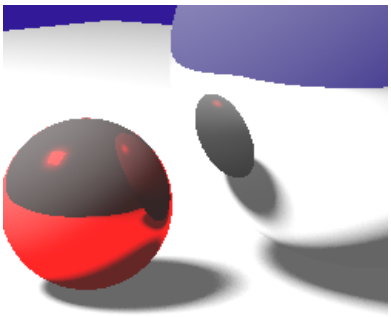

Figure 2: *Jittered sampling of shadows*

are needed in order to achieve smoother results. This of course requires more computation time, but the number of samples don't need to be as high to achieve similar results with the other methods

2.2 Anti-Aliasing

Another basic feature included in the raytracer is anti-aliasing by using distributed raytracing. This feature is very similar to sampling over an area light source, but here the area that is sampled is each pixel. In a physical camera, each pixel is determined by collecting light that hits a sensor. This sensor has an area which it collects light over. The color of each pixel is determined by averaging the light that hits the sensor in different locations.

The naïve method for determining the color of each pixel in a raytracer would be to cast one ray from the focal point through the center of each pixel. This is not accurate, and multiple samples should be taken and averaged. As with sampling an area light source, taking jittered sampling achieves better results than random sampling or breaking each pixel into a grid. Using jittered sampling, pixels on the boundary of two different objects are a weighted average of the objects.

2.3 Reflection

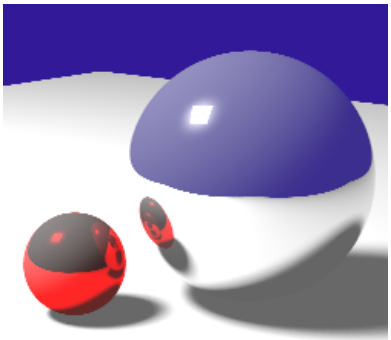

Figure 3: *Example of many levels of reflection*

When light interacts with certain materials, it reflects at the point of intersection and continues, but with a modified direction. For specular reflection the angle between the incoming light and the normal at the point of intersection is equal to the angle of the outgoing light and the normal. At this point of intersection not all of light is reflected, but a proportion based on the properties of the object.

In raytracing this is easy to simulate by recursively tracing a ray to calculate the contribution from the reflection. This recursion has no guarantee of stopping if the ray tracing constantly hits reflective surfaces. Assuming that all reflective surfaces do not add energy to the reflected light, then each subsequent recursive step has less impact on the final result. Therefore, the stopping the recursion after only a few number of steps ensuring halting and does not affect the quality of image by that much.

2.4 Previewer

In addition to rendering the scene using raytracing, my project allows the user to view a rasterized version of the scene. This preview solely for the purpose of choosing an appropriate angle and insuring that intersection tests are working properly. The previewer allows the user to move the camera around the scene in real-time, and then begin raytracing once the desired view is found. This is useful since, once raytracing begins the user cannot move the camera without having to restart the process. The previewer can also serve as a check on intersection tests, since the two scenes should have similar geometry. So if the two scenes seem drastically different in terms of where objects are, then something must have went wrong with the intersection test.

2.5 Raytree Visualizer

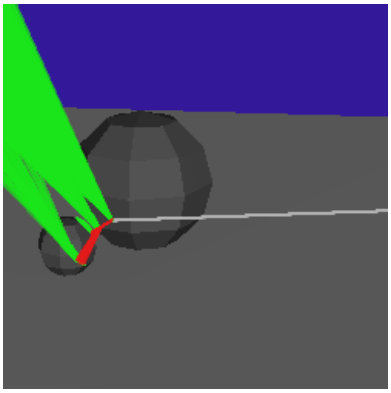

Figure 4: *Raytree visualizer. Red segments are reflected rays, white are primary rays, and green are shadow rays.*

Another included debugging tool included in the project is a raytree visualizer. The raytree is the tree of rays created to determine the color of one pixel. This visualizer allows the user to show the rays needed to determine one pixel. This visualizer is integrated with both the previewer and the raytracer. The raytracer must add rays and the distance they travel every time it casts a ray. Although the tree is created by using the raytracer it must be visualized in the previewer. This is because the user must be able to see the tree from multiple angles to get all the information it is conveying. This is not possible in the raytracer, because it cannot change views in real time. The previewer must also display rays inside objects and behind objects, but it must be clear that the ray is behind or inside an object. To do this, the previewer uses OpenGl's blending functions.

The raytree visualizer is important for debugging both the raytracer and the primitives, and is good for showing what rays are created for each feature. It is much easier to see where rays are not behaving properly by looking at the rays directly as opposed to the raytraced

scene. It can also show with interactions between the rays and primitives. Tracing rays for one pixel is much quicker than for the entire scene, so it also takes much quicker to ensure that the mechanics of the raytracer are working properly.

2.6 Primitives

With raytracing it is possible to find the exact intersections between geometric primitives. This creates better quality scenes than by including mesh approximations of primitives. For objects like spheres or cones, it allows for smooth normal and therefore shading. For scattering it means that the objects can be treated as one.

For this project, I included four primitives: cubes, spheres, cones, and cylinders. Although the cube could be represented using a collection of quad faces, making it a primitive made it easier to deal with it for scattering. The other three were included to allow for raytracing to be done on the continuous versions of the objects.

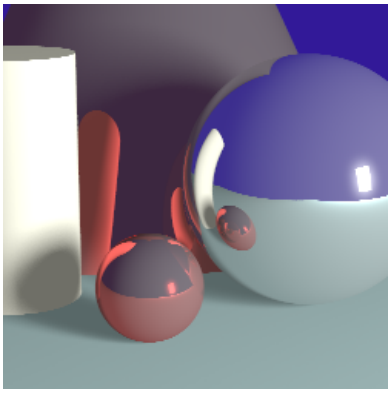

Figure 5: *Scene with different primitives*

For each primitive, three main features needed to be implemented. The first was an intersection test between a ray and the object. This is done by considering the ray as a function of t and using the implicit form of each object. For the cone and cylinder, the end caps also had to considered. The second feature was to create a rasterized version of primitive. This is needed to allow for the previewer to correctly display the primitives, since it cannot use their continuous definitions. The final feature needed for all primitives was getting a random point on the surface. This functionality is needed for computing multiple scattering and will described in detail later on.

3 Subsurface Scattering

Subsurface scattering occurs in many objects, and is essential to creating realistic renderings of certain objects. Without this feature, a glass of milk looks like a glass of white paint. This is because the light transported within milk plays a major role in its overall appearance. Without any subsurface scattering, the milk looks thick and has dark shadows on sides away from light. The color is also very white. Because of subsurface scattering, real milk has slight variations of color, and the side away from light is still lit.

Jensen et al. [2001] proposed a method for rendering the effects of subsurface scattering by using the BSSRDF. This differed from previous methods which used the BRDF to approximate subsurface scattering. In order to use th BRDF to approximate the effects of subsurface scattering, random walks needed to be taken in the object to simulate the transport of light between points. This is because the BRDF requires that light leaving a point must enter

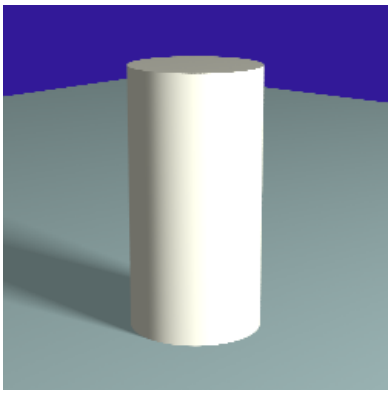

Figure 6: *Cylinder of milk with no scattering*

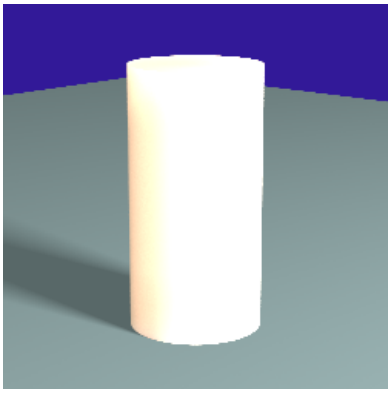

Figure 7: *Cylinder of milk with subsurface scattering*

that same point. The BSSRDF can describe light between any two points. The actual relationship between these rays must be defined for specific purposed, but it stills possesses the potential for more accurate raytracing. For subsurface scattering there exists approximations that can relate any two rays with endpoints on the same object. This is what allows for raytracing to render the effects of subsurface scattering using the BSSRDF.

The approximation makes assumptions about the properties about the medium and does not work for all types of scattering. The characteristics of the material in terms of subsurface scattering are described by two constants σ_a and σ_s. The first describes how likely the material is to absorb light, and the other describes how likely the material is to scatter light. Materials can have different values of these constants for different wavelengths of light, so they are usually expressed as 3 dimensional vectors, where each dimension the constant for red, green and blue light. The approximation assumes that $\sigma_s \gg \sigma_a$. This expresses the assumption that as light scatters in the medium its phase function tends towards isotropic. This means that each scattering event causes the distribution of light to become evenly distributed over the hemisphere.

This assumption means that the approximation is only good for a certain class of materials. The materials that meet the condition that $\sigma_s \gg \sigma_a$ are those that seem opaque but a large part of their appearance is due to subsurface scattering. They include milk, meat, and skin. These are the type of materials that can be accurately rendered using this method. For other types of materials, this approximation does not provide accurate results.

The approximation to the BSSRDF can calculate the contributions

of multiple scattering, but does not account for the special case of single scattering. Single scattering occurs when light scatters exactly once in the object. This means that incoming and outgoing rays must intersect. This is a special case that is not handled by the approximation to the BSSRDF. To calculate the contributions of single scattering, the BRDF is used, and the contributions are calculated by tracing the path of light through the object.

3.1 Theory

The contributions of subsurface scattering are partly calculated using the BSSRDF. This function relates the outgoing radiance at a point in a certain direction to the incoming radiance at any other point in any other direction:

$$dL_0(x_0, \vec{\omega}_0) = S(x_i, \vec{\omega}_i; x_0, \vec{\omega}_0) d\Phi_i(x_i, \vec{\omega}_i)$$

In this equation the BSSRDF is the function S. It relates the two radiances, but does not have an exact definition. The value of the function is determined by many factors, such as the geometry of the scene, and the characteristics of the objects involved. This is also true for the BRDF; however, it is possible to define it for specific situations. This is how the BRDF and BSSRDF are solved, by defining them for specific situations to mimic certain behaviors of light. To find the value of the radiance leaving a point in a given direction, the above equation is integrated over incoming directions and area:

$$L_0(x_0, \vec{\omega}_0) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i; x_0, \vec{\omega}_0) L_i(x_i, \vec{\omega}_i (\vec{n} \cdot \vec{\omega}_i) d\omega_i dA(x_i)$$

This describes the behavior of all light, and if the function S were to be defined correctly this could be used to create photo-realistic images. However, functions can be defined to simulate specific behaviors of light. The behavior of light in any participating media can be described by the radiative transport equation:

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

where σ_t is the extinction coefficient which is the sum of σ_a and σ_s, p is the normalized phase function, and Q is volume source distribution. Q is define as follows:

$$Q(x, \vec{\omega}) = \sigma_s \int_{4\pi} p(\vec{\omega}', \vec{\omega}) L_{ri}(x, \vec{\omega}' d\omega'$$

where L_{ri} is the reduced intensity. As light travels through a medium it gets absorbed or scattered, so it is as follows:

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

This describes how lightly will exponentially decrease as it travels further through the medium. These definitions allow us to accurately describe how light interacts with a participating media. If we then integrate the radiative transport equation over all directions at a point, we get:

$$\vec{\nabla} \cdot \vec{E} = -\sigma_a \phi(x) + Q_0(x) \tag{1}$$

In this equation ϕ is the radiant fluence, and $\vec{E}$ is the vector irradiance, which are defined as follows:

$$\phi(x) = \int_{4\pi} L(x, \vec{\omega}) d\omega$$
$$\vec{E}(x) = \int_{4\pi} L(x, \vec{\omega}) \vec{\omega} d\omega$$

In equation 1, Q_0 is the integral of Q over all directions. Solving this equation in general is not possible, but we can use the diffusion approximation to calculate the relationship of light that scatters within an object where $\sigma_s \gg \sigma_a$. In this case we can approximate the outgoing radiance by a two term expansion:

$$L(x, \vec{\omega}) = \frac{1}{4\pi} \phi(x) + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(x)$$

We can take the approximation to the outgoing radiance and substitute it into the radiative transport equation and integrate over $\vec{\omega}$ to get:

$$\vec{\nabla} \phi(x) = -3\sigma_t' \vec{E}(x) + \vec{Q}_1(x) \tag{2}$$

where σ_t is the reduced extinction coefficient, and $\vec{Q}_1$ is the first order volumetric source term. The reduced extinction coefficient is affected by the degree to which the material is anisotropic. If the material is isotropic, then is equal to σ_t. The first order volumetric source is the integral of all directions over the hemisphere weighted by the volumetric source distribution for that direction. When there are no sources the $\vec{Q}_1$ vanishes from the equation and we can related the vector irradiance to the scalar fluence using the gradient:

$$\vec{E} = -D \vec{\nabla} \phi(x)$$

where D is $1/3\sigma_t'$ and is called the diffusion constant. This shows that there is energy flow from areas of high energy density to regions of low energy density.

We can then derive the diffusion equation by substituting equation 2 into equation 1:

$$D \nabla^2 \phi(x) = \sigma_a \phi(x) - Q_0(s) + 3D \vec{\nabla} \cdot \vec{Q}_1(x)$$

This equation only has solutions in very specific cases, and only a few of these solutions are usable for use in a raytracer. For an isotropic light source in an infinite medium the solution is:

$$\phi(x) = \frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r(x)}}{r(x)}$$

where Φ is the power of the light source, σ_{tr} is $\sqrt{3\sigma_a \sigma_t'}$, and $r(x)$ is the distance between the light and the point. This solution is never actually useful for creating images, since it only allows for a single point light source and for one object which must be infinitely large. For more interesting cases, certain boundary conditions must be met. The boundary condition is that the net inward diffuse flux is zero at each point on the surface. Using the appropriate boundary conditions it is possible to compute the BSSRDF for diffuse reflectance, R_d. R_d is equal to the radiant exitance divided by the incident flux:

$$R_d(r) = -D\frac{(\vec{n}\cdot\vec{\nabla}\phi)(x_s)}{d\Phi_i(x_i)}$$

where x_s is a point on the boundary between two media, and r is the distance between x_s and x_i.

This equation does not have a solution for finite objects. This means we cannot analytically compute the BSSRDF for a sphere or cone directly. However, we can create an approximation for subsurface scattering. Here we model the interaction as a semi-infinite plane. It is possible to find exact equations for this case, but they involve an infinite summation of Bessel functions, which cannot be used in a raytracer.

An approximation exists which creates a dipole to satisfy the boundary conditions. The dipole consists of two point light sources, a real positive point light source below the surface, and a virtual negative point light source. The real light source is placed at a distance z_r below the surface and the virtual light source a distance z_vThis method yields the following approximation for R_d:

$$R_d(r) = \frac{\alpha'}{4\pi}\left[(\sigma_{tr}d_r+1)\frac{e^{-\sigma_{tr}d_r}}{\sigma'_t d_r^3} + z_v(\sigma_{tr}d_v + 1e^{-\sigma_{tr}d_v}\sigma'_t d_v^3\right]$$

where α' is the reduced albedo, d_r is the distance between x_i and the real source, and d_v is the distance between x_i and the virtual light source. This approximation is then used to compute the BSSRDF for subsurface scattering. The approximation does not include the contributions for single scattering, so that must be computed separately. The key to using this approximation is placing the dipoles in such a way that they satisfy the boundary conditions, which will be talked about more later.

3.2 Single Scattering

The diffusion approximation does not account for the contributions from single scattering. This can greatly affect the overall look of the object, since not a lot of energy is scattered or absorbed for one scattering event. Instead of computing this by using the BSSRDF, it is computed by the BRDF. This means that the path for the single scattering must be traced and the amount of radiance must be computed for each segment.

In order to compute the single scattering contribution we can use the reduced radiance L_{ri} to compute the amount of radiance that goes through medium for each ray. We must also take into consideration the proportion of light that will exit the surface in the outgoing direction as opposed to reflect or scatter at the boundary.

In the continuous solution, we integrate the incoming radiance over the refracted outgoing ray using the following integral:

$$L_o^{(1)}(x_o,\vec{\omega}_o) = \sigma_s(x_o)\int_{2\pi} Fp(\vec{\omega}'_i,\vec{\omega}_o)\int_0^{\infty} e^{\sigma_{tc}s}L_i(x_i,\vec{\omega}_i)dsd\vec{\omega}_i$$

where F is the product of the Fresnel term for the incoming and outgoing directions, σ_{tc} is the combined coefficient term. It combines σ_t at x_o to σ_t at x_i, but weights σ_t at x_i by a term that describes the geometry. This integral allows for a solution to the BSSRDF by using the BRDF to compute the radiance transfered between the points.

In raytracing we must use the discrete version of this integral to find the contributions from single scattering. We do this by sampling along the outgoing direction for scattering points. We then cast a ray from the scatter point towards a light source and find the intersection with the object itself. We finally cast a ray from this point to the light source. We sample the distances into the object based on the probability that the ray will travel that far before scattering. This means our random distance is $s'_o = \log(v)/\sigma_t(x_o)$, where v is a random value between zero and one. This causes smaller distances to be sampled more often than those far away. We can then find the outgoing radiance for this randomly drawn distance as follows:

$$L_o^{(1)}(x_o,\vec{\omega}_o) = \frac{\sigma_s(x_o)Fp(\vec{\omega}_i\cdot\vec{\omega}_o)}{\sigma_{tc}}e^{-s'_i\sigma_t(x_i)}e^{-s'_o\sigma_t(x_o)}L_i(x_i,\vec{\omega}_i)$$

where s'_i is the distance between x_i and the scatter point. We then average this for a user specified number of samples to find the overall contribution of single scattering.

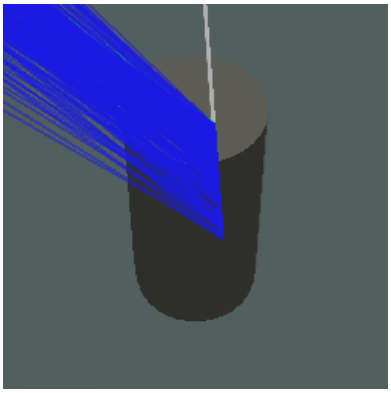

Figure 8: *Visualization of single scattering samples*

There are many details which need to be considered when computing this value. The light from x_i to the scatter point should refract at x_i; however, it is very costly to compute the correct direction to initial go to find x_i, such that the refracted ray goes towards a light. It is much easier to assume that light doesn't refract then approximate the refracted distance. We take the distance without refracting s_i and then use Snell's law to approximate the distance if we had refracted at x_i:

$$s'_i = s_i\frac{|\vec{\omega}_i\cdot\vec{n}_i|}{\sqrt{1-\left(\frac{1}{\eta}\right)^2(1=|\vec{\omega}_i\cdot\vec{n}(x_i)|^2}}$$

This allows us to approximate refracting the incoming radiance, which would be computational expensive to exactly determine.

The formulation also easily allows for differing properties of the object to be taken into account. It assumes that the properties are the same at the two surface points as they are between the surface point and the scatter point. This is not true in general, but calculating the properties in the volume would require a much more complex method. The current assumption is capable in accurately showing the effects of color bleeding. Due to the representation of material in my system, objects cannot have different properties on its surface. This means that my project cannot show the effects of color bleeding, but it would require only changes to how the properties of the material are represented. The calculation of the single scattering contribution can handle different properties of an object.

When choosing random distances it is possible to choose a distance which would result in the scatter point being outside the object, since the range of the distance is 0 to ∞. To ensure that the scatter point is in the object, before taking a sample a ray is cast from x_o in the outgoing direction. The intersection between this ray and the object is then found. The range of the random value is then clamped such that the range of distances is 0 to the distance the test ray traveled. Doing this ensures that all scatter points are in the object.

3.3 Multiple Scattering

Multiple scattering is solved by directly calculating the value of the BSSRDF. In order to do this efficiently random points are selected on the surface, and the contributions of these points are averaged together. Points that are further away contribute less to the overall appearance of the material, so importance sampling is used. Points are selected with a density of $\sigma_{tr}e^{-\sigma_{tr}d}$ where d is the distance to x_o.

Once we have our two points, x_o and x_i, we place the two point light sources in such a way that it satisfies the boundary conditions described earlier. A real positive light source is placed below the surface at a depth of $1/\sigma_t$. This is the mean free path of light in this material, or the average distance between scattering events. Then the virtual light source is placed above x_o in such a way that it satisfies the boundary conditions. This distance is based on the index of refraction of the material.

Once the dipole system is set up, rays are cast from the random point to light sources, and the incoming radiance is calculated. Then the diffusion approximation is solved and is used to determine the amount of radiance for this point. This has to be done for a large number of samples to produce good results.

This method runs into some trouble, such as how to efficiently choose random samples and what to do for arbitrary geometry. The approximation was derived for a semi-infinite plane. This means that space is divided by one plane between the medium and nothing. This does not tell us anything about an arbitrary scene. Consider a single cube, if x_o and x_i are on the same face of the cube, then the approximation to a semi-infinite plane will do alright. However, if we look at a pair of points that are placed on opposite sides of an edge of the cube, then we may place x_i extremely close to the real light source. This causes the BSSRDF to have an extremely high value which causes visual artifacts. This can cause white spots along creases of certain objects.

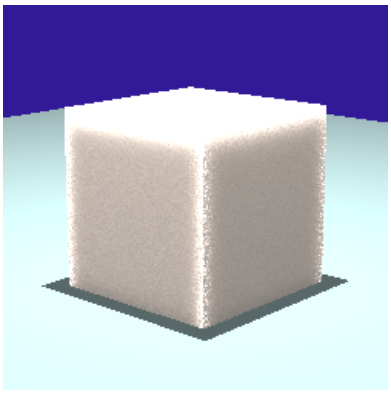

Figure 9: *Singularities occur on edge of box*

The solution to this is to make the minimum distance between x_i and either of the point light sources the distance between x_o and the light source. Jensen et al. [2001] found that this method worked in experiments they ran. It also seems to work for the tests I have run.

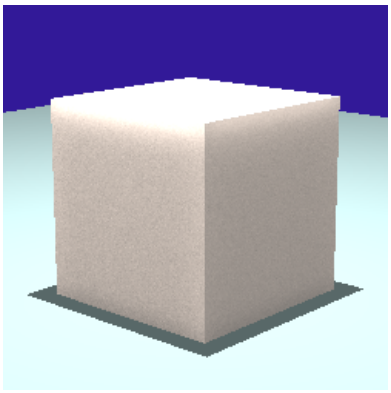

Figure 10: *Same cube using solution to singularity problem*

The other problem with this method is choosing a random sample. Ideally after choosing a distance for the sample, a sample could be chosen exactly at that distance. However, this would require computing the intersection between the primitive and a sphere. For primitives made up of different parts- cones, cylinders, and cubes- the intersection between one discrete part must be chosen with a probability related to the length of the curve made by the intersection of the part and the sphere. This process would be computationally expensive, so I used a different method for choosing a random point.

My raytracer has three different ways of computing a random point for use in multiple scattering. The first is the most basic and just chooses a random point on the surface. It does this by first choosing a section of the object by computing the surface area of each discrete section and then gives each section a probability related to its surface area. Then each discrete section has a routine to find a point on its surface uniformly at random. This process generates points over the entire object and does not depend on x_o. When using these points, the probability that it would be chosen using the ideal method must be factored into the calculations.

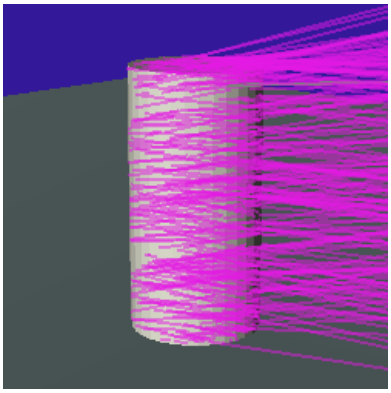

Figure 11: *Choosing samples for multiple scattering using the first method*

The second and third method build on the first. The second method finds a random point within a given distance to a point, and the third finds a random point between two supplied distances to a point. These two methods take advantage of the exponential fall off better, but the process for choosing the distances must be altered. The

chosen distance, d, must should an integral of the probability of choosing samples at distances between 0 and d. These methods can require numerous iterations to find a random point with the given properties. To ensure choosing a random sample does not take too long, modifications were made. For the second method, the random point must be within a given distance plus a constant to x_o. This ensures that it does not take too long to find a random point that has to be extremely close to x_o. For the the third method, after a given number of iterations, the second method is used given the larger of the two distances. Theses methods for choosing random points is not ideal, but does not require expensive calculations, which is important since a large number samples must be used to get noise free results.

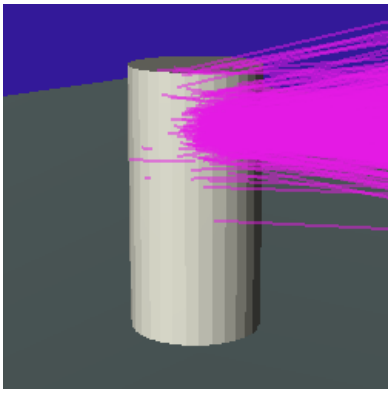

Figure 12: *Choosing samples for multiple scattering using the third method*

4 Results and Limitations

The resulting raytracer has the ability to approximate the effects of subsurface scattering in objects. This makes many rendered objects seem realistic. It allows for light to be transported to areas that would otherwise be in shadow. It also realistically renders the color of any light transported in this manner.

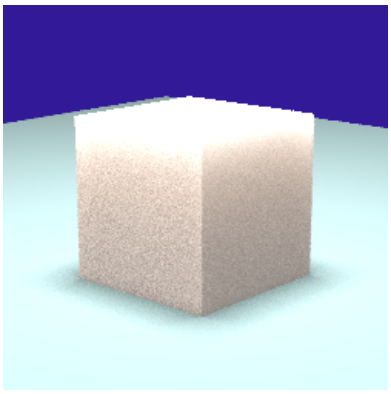

Figure 13: *Marble using a large number of samples for subsurface scattering*

The above scene is of a cube of marble. The constants to simulate the marble were abtained from Jesen et al. [2001]. Along with propsing the method to render subsurface transport, they collected data by measuring how light interacted with physical objects. They took a laser and a sample of a variety of materials and then measured the amount of light that was transported in a variety of directions. They then used this data to determine the two constants, σ_s and σ_a, that are used in calculting single and multiple scattering. Using these constants I was able to produce objects with properties of those of physcial objects. the marble cube below is lit directly from above, and without any subsurface scattering should have different colors for each face. However, due to subsurface scattering the top of the two vertical faces is brighter.

The speckling pattern of the marble was caused by noise, and is not supposed to be present. The noise could be recued by using more samples; however, this would require more computation time. The goal of the project was to be able to render of the effects of subsurface scattering, and not to do this extremely efficiently. The overall algorithm was aimed at the quality of the image in a reasonable amount of time, which was accomplished. Unfortunately due to time constraints high quality renderings for all material was not possible.

Another limititation of the raytracer is that the only advance effect that it can render is subsurface light transport. Below is a raytraced image of a cylinder of milk. Although the cylinder has the properties of milk, it does not look like milk since milk is a liquid and woul normally be in a glass. However, my raytracer does not have the ability to render materials like glass. Although refraction is involved in computing subsurface refraction, it cannot generalize to render translucent materials. This affects the overall quality images. Without any contextual clues, it is much harder to determine that the cylinder has the same properties of milk.

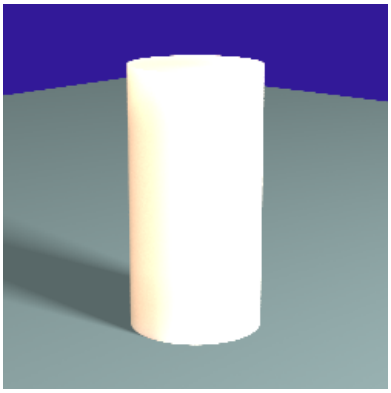

Figure 14: *Picture of a cylinder of milk*

5 Conclusion

Subsurface scattering is an extremely important effect required to created photo-realistic images of a variety of objects. It is caused when light enters an object scatters multiple times and then leaves. Without this effect, objects can have a hard computer generated feel. The contributions of this effect can be calculated by using the BSSRDF. Using this function instead of the BRDF gives betters results, since it approximates the contributions for all paths at once, instead of sampling a fwe random paths.

The result is a raytracer that can simulate this effect. It must separate single scattering because it is a special case where the two rays intersect. The resulting process requires a large number of samples to accurately compute the effect, so it is requires a long time to render one image. However, given enough time accurate images can be created.

References

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HAN-

RAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 511–518.