CS500 Final Project: Subsurface Scattering

Markel Sevilla Pelayo

DigiPen Institute of Technology Europe

Index

1-Abstract 3
2-Introduction 3
3- Subsurface Scattering 3
3.1- Subsurface Scattering: Conceptually 3
3.2- Approaches to Subsurface Scattering 3
3.2.1-Directional Dipole for Subsurface Scattering in Translucent Materials 3
3.2.2-A Practical Model for Subsurface Light Transport 4
4-My Implementation: a BRDF approximation 7
4.1- Inserting it into the Ray Tracer 9
5-Demo and Outpts 10
5.1- Material 10
5.2- Outputs 10
6-PROS and CONS 12
6.1-PROS 12
6.1-CONS 12
7-Future work 12
7.1-DIPOLE Implementation 12
7.2-Scene integration 13
8-Conclussion 13

1-Abstract

This report explains my journey in the implementation of a variance in subsurface scattering in translucent materials, initially based on the research on *Directional Dipole for Subsurface Scattering in Translucent Materials* carried by the *Technical University of Denmark, Aarhus University* and *The Alexandra Institute,* and later focused on the research conducted by *Stanford University* on how to do *A Practical Model for Subsurface Light Transport.* I will explain the techniques applied in order to get the results that I achieved as well as the difficulties, improvements and different notes about my adventure deciphering and applying these approaches.

2-Introduction

After finishing our model of a Ray Tracer we have been able to render and accurately interpret various types of materials such as diffuse, metallic and dielectric. In this project I initially intended to add a new material that allowed subsurface scattering with configurable parameters. Following the previous approaches on how to do so I have implemented a personal approach that I will explain in the following sections.

3- Subsurface Scattering

3.1- Subsurface Scattering: Conceptually

Subsurface Scattering is a natural effect that semi-translucent objects tend to experience, in which an incident ray of light enters the surface of a material. While inside of the object this ray scatters, travelling a certain distance and exiting through different positions depending on the material properties. Achieving this effect in a ray tracing application can present a challenging obstacle that I tried to overcome by the research papers that I previously mentioned. Having said this, let’s dive into the first approach.

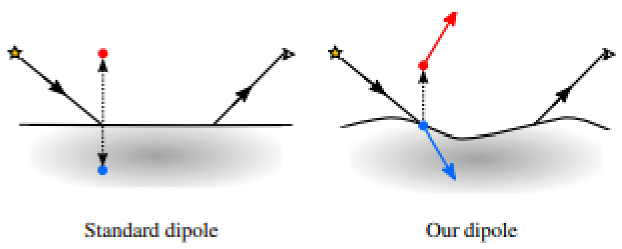
3.2- Approaches to Subsurface Scattering

3.2.1-Directional Dipole for Subsurface Scattering in Translucent Materials

3.2.1.1- Raw explanation of the paper

This paper by the combination of the *Technical University of Denmark, Aarhus University* and *The Alexandra Institute* presents an approach on how to model subsurface scattering with a separable BSSRDF approach adding an interesting twist on the boundary condition of the Surface area.

Opposed to the traditional dipole method, this group presents a case in which the assumption that the area is flat is loosened and assumes different geometries for a more realistic approach such that:



Apart from this feature they propose several changes in the equations of the separable BSSRDF approach, that after thorough math that I won't cover reach that the final equation lies as follows:

$$\boxed{S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) = S'_d(\boldsymbol{x}_i, \vec{\omega}_{12}; \boldsymbol{x}_o) - S'_d(\boldsymbol{x}_v, \vec{\omega}_v; \boldsymbol{x}_o)}$$

A key point in understanding why this equation differs from the next approach that I will explain is that after rendering both the single scattering and diffuse term they concluded that the first term contributed very little to the final output, and thus they used variations of the same diffuse term to model subsurface scattering.

3.2.1.2- Problems in application

My explanation of this method has been so concise because after digesting what the paper had to offer it proved to be too challenging for me to translate its dense math and physics into code. Although I grasped the concept of what the research was aimed at, I found myself incapable of replicating an acceptable demo with the approach that was presented, and thus I resigned from the direction that *Directional Dipole for Subsurface Scattering in Translucent Materials* presented, and jumped to the next approach.

3.2.2-A Practical Model for Subsurface Light Transport

3.2.2.1- Raw explanation of the paper

This paper presented by the *University of Stanford* introduces a simple model for subsurface scattering in translucent materials. It aims for a model based on a separable BSSRDF for correctly representing subsurface scattering in both anisotropic and isotropic materials in an infinite medium. Making use of the previously mentioned dipole method it presents a basis upon which I tried to create a subsurface light transport model. In this case, opposing to the previous approach, the presented dipole is a classical dipole instead of directional, and assumes a flat surface. For reference the basis of a dipole model lays in the fact that light is being cast from two sources of light. A real source that is at a certain distance z_r below the shaded point and a virtual source, that is at z_v above the point.

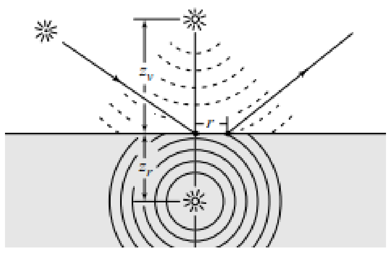


Figure 3: An incoming ray is transformed into a dipole source for the diffusion approximation.

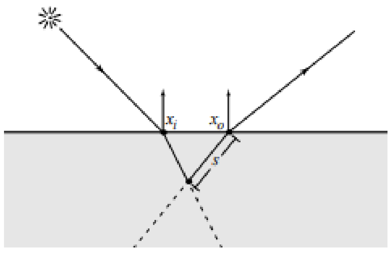


Figure 4: Single scattering occurs only when the refracted incoming and outgoing rays intersect, and is computed as an integral over path length s along the refracted outgoing ray.

After conducting some thorough math that I will further explain in the next sections the paper comes with the following equation that models the effect that we aim, separated as its name suggests, in the single scattering term $S^{(1)}$, and the diffuse term S_d:

$$S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) = S_d(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) + S^{(1)}(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)$$

3.2.2.2- Separable BSSRDF Diffuse Term

The diffuse term of the BSSRDF model presented by this research takes the form of the following equation:

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

After looking at S_d , there are some terms that we should be deciphering in order to correctly model it:

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

First, the diffuse reflectance term R_d, that takes as input a radius, in this case the distance from the entry point (x_i) and the exit point (x_o), with several coefficients.

The reduced extinction coefficient, based on the scattering term and absorption term of the material, and the mean cosine of the incident and out ray, (g = 0 for isotropic materials).

$$\sigma_t' = \sigma_s' + \sigma_a \quad \text{where} \quad \sigma_s' = \sigma_s(1-g) .$$

The effective extinction coefficient based on the previously mentioned.

$$\sigma_{tr} = \sqrt{3\sigma_a\sigma_t'}$$

The reduced albedo coefficient, computed as the reduced scattering over the extinction coefficient.

$$\alpha_{\text{red}} = \frac{\sigma_s}{\sigma_t}$$

And finally, the d_r and d_v terms that refer to the distances from the shaded point to the real source of light and the virtual source of light respectively.

Once having completed the diffuse reflectance term, the only remaining terms for the modelling of the diffuse equation are the Fresnel terms of both the incident and the outgoing rays. The Fresnel term is modelled as *1 – ReflectionCoefficient(w, n),* with *w* as the ray and *n* as the current index of refraction.

3.2.2.2- Separable BSSRDF Single Scattering Term

As the diffusion term the single scattering term can be expressed as a function of several terms:

$$L_o^{(1)}(x_o, \vec{\omega}_o) = \frac{\sigma_s(x_o) F p(\vec{\omega}_i \cdot \vec{\omega}_o)}{\sigma_{tc}} e^{-s_i' \sigma_t(x_i)} e^{-s_o' \sigma_t(x_o)} L_i(x_i, \vec{\omega}_i).$$

As we can see this equation is making use of many of the previously explained coefficients, with a small twist as it considers materials with different scattering and absorption along their texture, (thus computing the scattering and extinction coefficients of certain points). It also makes use of the incoming radiance, L_i, and the product of the incoming ray's and outgoing ray's Fresnel terms, *F.* Considering all of this the only three unknowns that remain are the function *p*, s_i and s_0. In this equation *p* refers to the phase function of the material we are representing. We are usually going to represent it as *p = 1/4pi* for isotropic materials. Regarding s_i and s_0 we will represent them as $s_0 = \log(\xi)/\sigma t(x_o)$ *(with* $\xi = (0,1]$*), and* s_i *:*

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

3.2.2.3- BRDF Approximation

Although not part of the separable BSSRDF the paper gives an interesting insight on how to model subsurface light transport as an approximation to a BRDF model. This is suitable since it only needs one source of light and gets rid of most of the complexity presented by the previous equations, as the diffusely reflected light is solely defined by the intrinsic parameters of the material:

$$f_r(x, \vec{\omega}_i, \vec{\omega}_o) = f_r^{(1)}(x, \vec{\omega}_i, \vec{\omega}_o) + F\frac{R_d}{\pi} .$$

Here we have two different terms that can be computed as follows:

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

F being the Fresnel term of the incoming ray and *p* being the phase function of the material.

Regarding our diffuse reflectance term:

$$R_d = 2\pi \int_0^\infty R_d(r)\, r\, dr = \frac{\alpha'}{2} \left(1 + e^{-\frac{4}{3}A\sqrt{3(1-\alpha')}}\right) e^{-\sqrt{3(1-\alpha')}} .$$

Our new expression's only unknown at this point would be the internal reflection parameter *A*:

$$A = \frac{1 + F_{dr}}{1 - F_{dr}}.$$

F_{dr}, in the same way is the rational approximation of the Fresnel formula, that solved by a numerical method would have the following shape(*n* being the I of Refraction):

$$F_{dr} = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta.$$

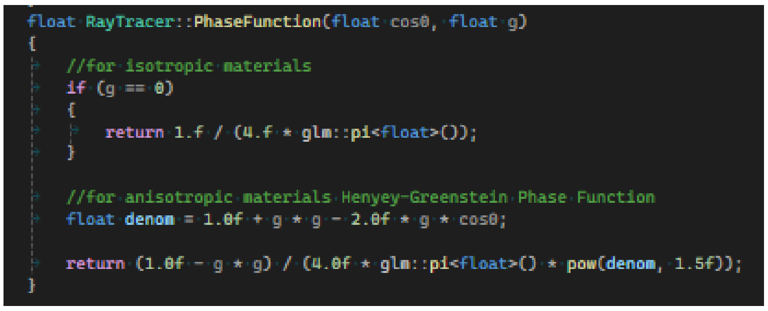
3.2.2.4- Problems in application

Although this approach proved to be much more realistic and understandable than the Directional Dipole method, I found myself between the devil and the deep blue sea again. During my research, I understood the theory behind the application and found that it was realistically attainable in paper. However, at the moment of translating it into code a similar problem as with the previous idea arose, as I could not find the way in which the dipole method worked. So, after unsuccessfully trying to correctly build a scene with a separable BSSRDF model I decided to opt for the BRDF approximation.

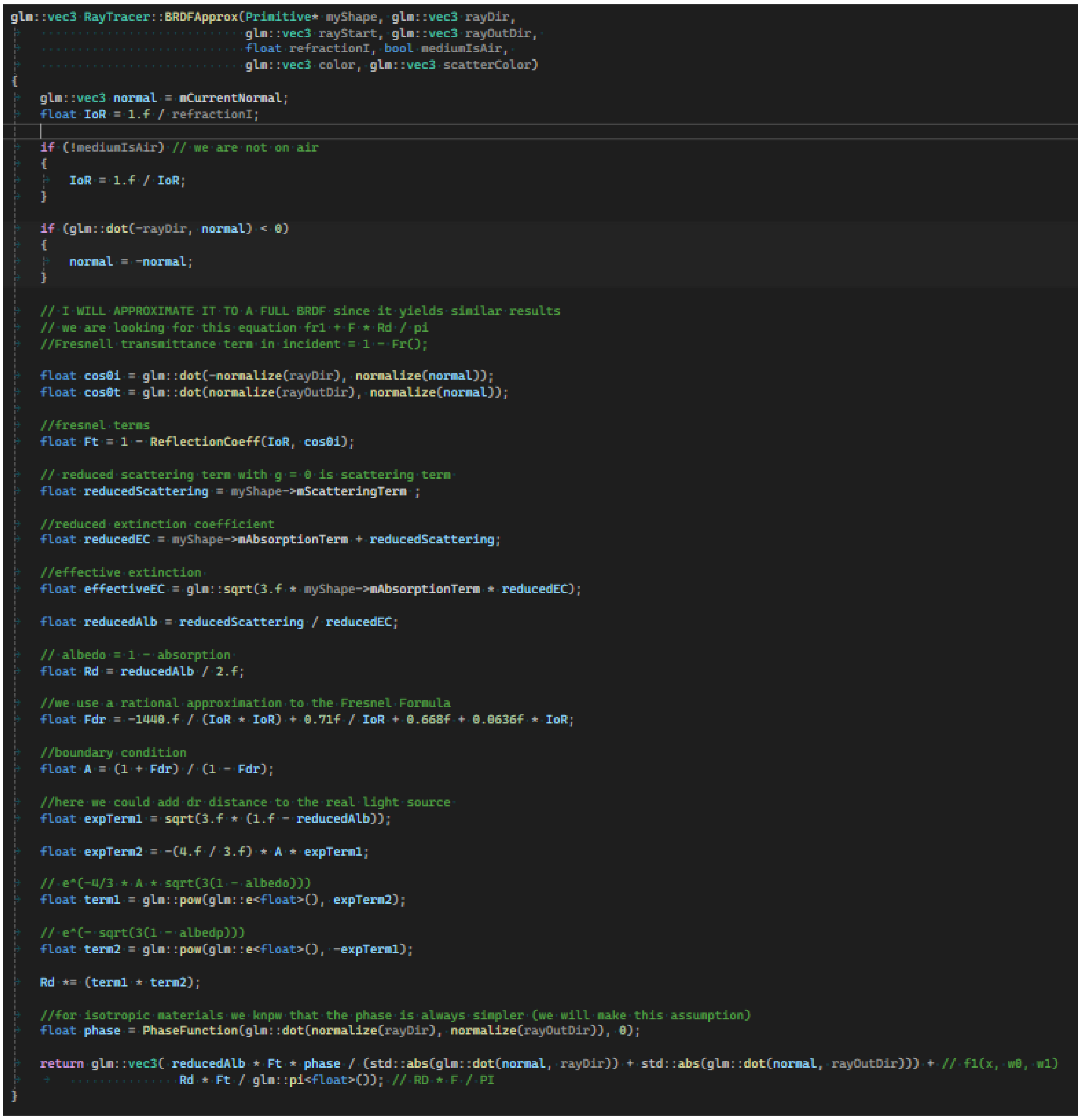
4-My Implementation: a BRDF approximation

As I have mentioned for this implementation, I based my work on the BRDF approximation for subsurface scattering presented by the University of Stanford. Following the explained mathematical explanation I was able to design a function that successfully represented the subsurface scattering equation presented in the previous chapter.

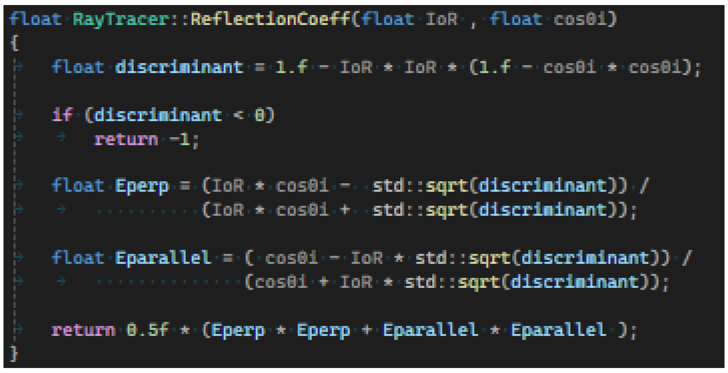
It is worth noting that for some reason even though the paper suggested dividing the phase function by the sum of the dot products between the normal and the refracted version of each ray, doing the same product with the original rays laid better results.



```cpp
float RayTracer::PhaseFunction(float cosθ, float g)
{
    //for isotropic materials
    if (g == 0)
    {
        return 1.f / (4.f * glm::pi<float>());
    }

    //for anisotropic materials Henyey-Greenstein Phase Function
    float denom = 1.0f + g * g - 2.0f * g * cosθ;

    return (1.0f - g * g) / (4.0f * glm::pi<float>() * pow(denom, 1.5f));
}
```



```cpp
glm::vec3 RayTracer::BRDFApprox(Primitive* myShape, glm::vec3 rayDir,
                                glm::vec3 rayStart, glm::vec3 rayOutDir,
                                float refractionI, bool mediumIsAir,
                                glm::vec3 color, glm::vec3 scatterColor)
{
    glm::vec3 normal = mCurrentNormal;
    float IoR = 1.f / refractionI;

    if (!mediumIsAir) // we are not on air
    {
        IoR = 1.f / IoR;
    }

    if (glm::dot(-rayDir, normal) < 0)
    {
        normal = -normal;
    }

    // I WILL APPROXIMATE IT TO A FULL BRDF since it yields similar results
    // we are looking for this equation fr1 + F * Rd / pi
    //Fresnell transmittance term in incident = 1 - Fr();

    float cosθi = glm::dot(-normalize(rayDir), normalize(normal));
    float cosθt = glm::dot(normalize(rayOutDir), normalize(normal));

    //fresnel terms
    float Ft = 1 - ReflectionCoeff(IoR, cosθi);

    // reduced scattering term with g = 0 is scattering term
    float reducedScattering = myShape->mScatteringTerm ;

    //reduced extinction coefficient
    float reducedEC = myShape->mAbsorptionTerm + reducedScattering;

    //effective extinction
    float effectiveEC = glm::sqrt(3.f * myShape->mAbsorptionTerm * reducedEC);

    float reducedAlb = reducedScattering / reducedEC;

    // albedo = 1 - absorption
    float Rd = reducedAlb / 2.f;

    //we use a rational approximation to the Fresnel Formula
    float Fdr = -1440.f / (IoR * IoR) + 0.71f / IoR + 0.668f + 0.0636f * IoR;

    //boundary condition
    float A = (1 + Fdr) / (1 - Fdr);

    //here we could add dr distance to the real light source
    float expTerm1 = sqrt(3.f * (1.f - reducedAlb));

    float expTerm2 = -(4.f / 3.f) * A * expTerm1;

    // e^(-4/3 * A * sqrt(3(1 - albedo)))
    float term1 = glm::pow(glm::e<float>(), expTerm2);

    // e^(- sqrt(3(1 - albedp)))
    float term2 = glm::pow(glm::e<float>(), -expTerm1);

    Rd *= (term1 * term2);

    //for isotropic materials we knpw that the phase is always simpler (we will make this assumption)
    float phase = PhaseFunction(glm::dot(normalize(rayDir), normalize(rayOutDir)), 0);

    return glm::vec3( reducedAlb * Ft * phase / (std::abs(glm::dot(normal, rayDir)) + std::abs(glm::dot(normal, rayOutDir))) + // f1(x, w0, w1)
                    Rd * Ft / glm::pi<float>()); // RD * F / PI
}
```



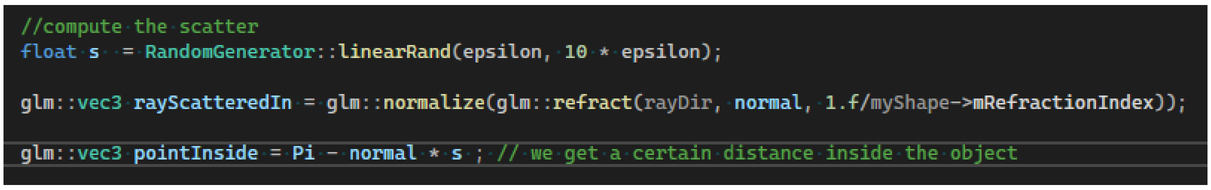
```
float RayTracer::ReflectionCoeff(float IoR , float cos0i)
{
    float discriminant = 1.f - IoR * IoR * (1.f - cos0i * cos0i);

    if (discriminant < 0)
        return -1;

    float Eperp = (IoR * cos0i -  std::sqrt(discriminant)) /
                  (IoR * cos0i +  std::sqrt(discriminant));

    float Eparallel = ( cos0i - IoR * std::sqrt(discriminant)) /
                      (cos0i + IoR * std::sqrt(discriminant));

    return 0.5f * (Eperp * Eperp + Eparallel * Eparallel );
}
```

4.1- Inserting it into the Ray Tracer

Once having coded the BRDF approximation all that was left was to create a way in which my rays and my material could interact with each other.

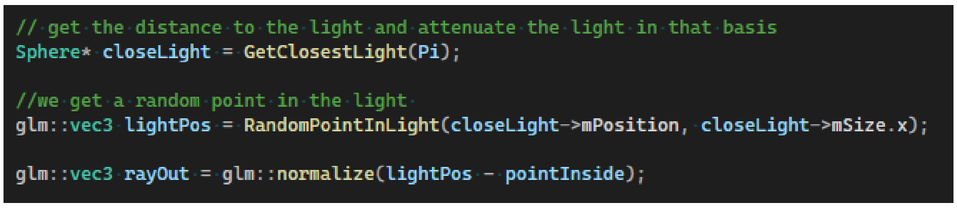
This was done in the material sampling side of the ray tracer, where if the material of the objects is a SUBSURFACE material, I sample it in a certain way.

For this kind of material, once the ray hits the surface, I make it enter the object a random amount of distance in the direction of the normal at the shaded point:



```
//compute the scatter
float s  = RandomGenerator::linearRand(epsilon, 10 * epsilon);

glm::vec3 rayScatteredIn = glm::normalize(glm::refract(rayDir, normal, 1.f/myShape->mRefractionIndex));

glm::vec3 pointInside = Pi - normal * s ; // we get a certain distance inside the object
```

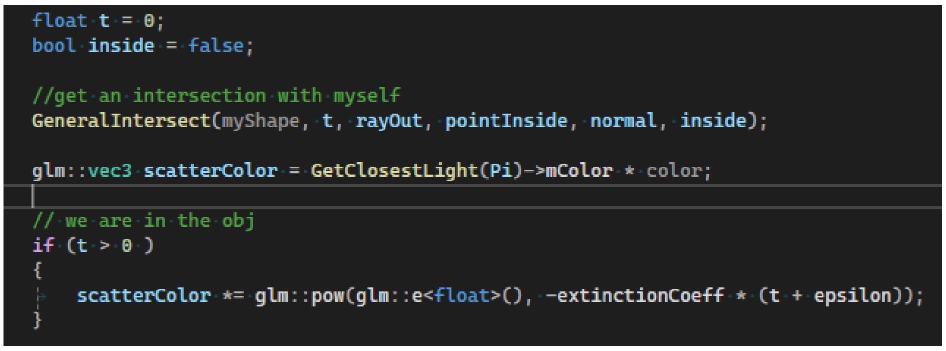
Once inside the object, since my objective is to model subsurface scattering in an infinite medium (void in this case), I bias the outgoing ray so it transmits in the direction of the light. To do so, I take a random point of the closest light to the point and create the outgoing ray.



```
// get the distance to the light and attenuate the light in that basis
Sphere* closeLight = GetClosestLight(Pi);

//we get a random point in the light
glm::vec3 lightPos = RandomPointInLight(closeLight->mPosition, closeLight->mSize.x);

glm::vec3 rayOut = glm::normalize(lightPos - pointInside);
```

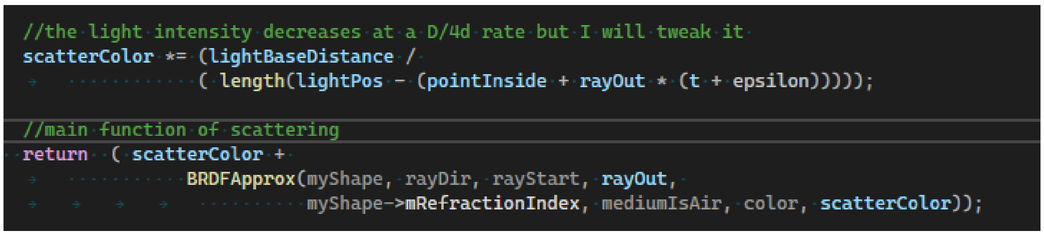
After this I intersect the outgoing ray with the object itself to determine the exit point in the direction of the light. In this way I can compute the distance the ray has traveled inside the object and attenuate the light according to the *reduced intensity* formula:

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



```
float t = 0;
bool inside = false;

//get an intersection with myself
GeneralIntersect(myShape, t, rayOut, pointInside, normal, inside);

glm::vec3 scatterColor = GetClosestLight(Pi)->mColor * color;

// we are in the obj
if (t > 0 )
{
    scatterColor *= glm::pow(glm::e<float>(), -extinctionCoeff * (t + epsilon));
}
```

Finally, I compute the attenuation of the distance from the light to the closest point in the mesh and add the color to my BRDF approximation equation.



```
//the light intensity decreases at a D/4d rate but I will tweak it
scatterColor *= (lightBaseDistance /
                ( length(lightPos - (pointInside + rayOut * (t + epsilon)))));

//main function of scattering
return  ( scatterColor +
               BRDFApprox(myShape, rayDir, rayStart, rayOut,
                         myShape->mRefractionIndex, mediumIsAir, color, scatterColor));
```

5-Demo and Outpts

This demo adds a new material to the framework:

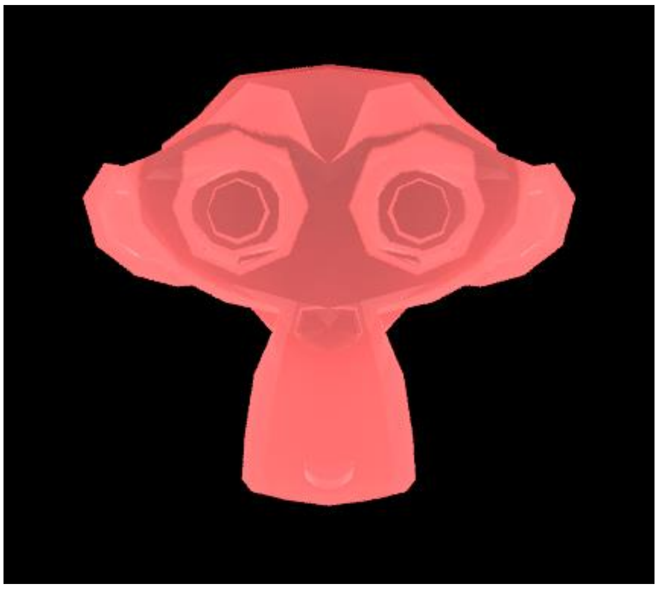
SUBSURFACE < DIFFUSE > < ABSORPTION > < INDEX OF REFRACTION > < SCATTERING >

5.1- Material

- < DIFFUSE > The color of the object
- < ABSORPTION > The absorption term of the object, the bigger it is the more light is absorbed
- < INDEX OF REFRACTION > The index of refraction of the material
- < SCATTERING > Single scattering term, the bigger it is the more the light will scatter

5.2- Outputs

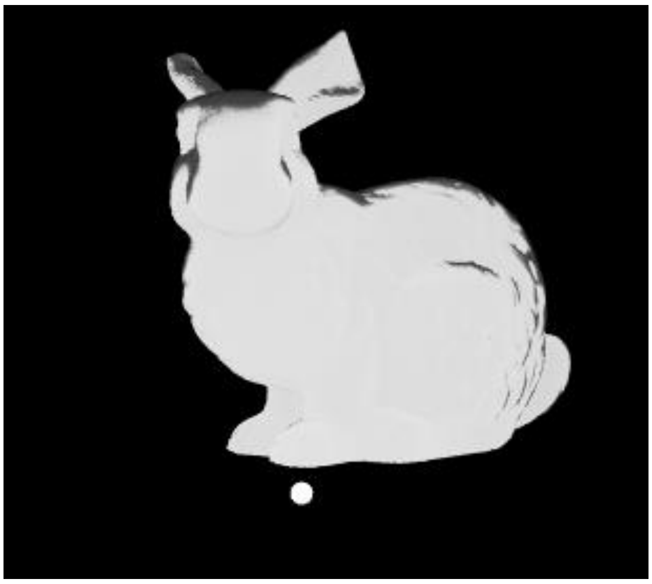
These are some of the outputs that I got to contrast with each other:



<ABSORPTION> 0.01

<IOR> 1.1

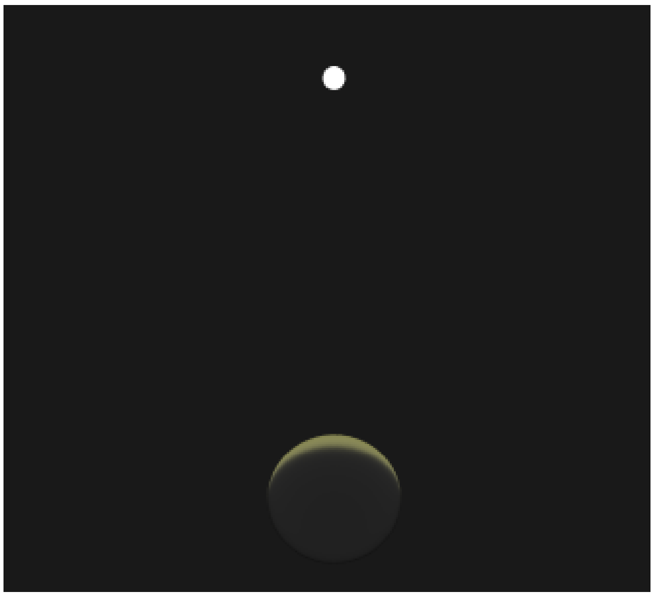
<SCATTERING> 1.2



<ABSORPTION> 0.001

<IOR> 1.8

<SCATTERING> 10



<ABSORPTION> 15

<IOR> 1.7

<SCATTERING> 15

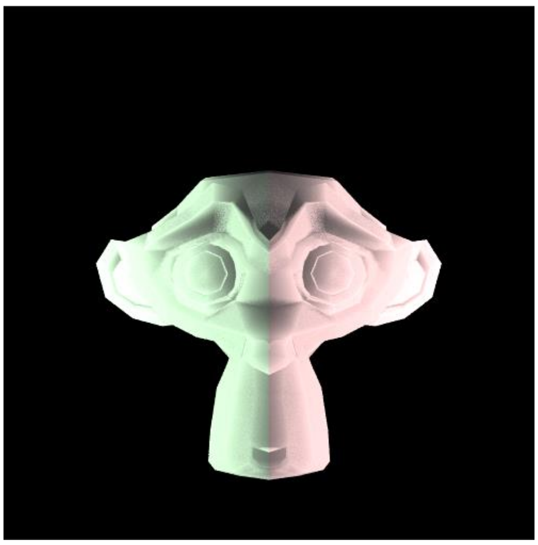
6-PROS and CONS

6.1-PROS

- A very cheap and simple to compute approximation
- Can add more than one light source and get quite fancy results
- The materials are easily parametrizable
- Since I am biasing the rays we don't have to wait to converge to get a cool image
- Quite intuitive approach physically speaking

6.1-CONS

- It's an approximation and not the actual dipole, meaning that the results are not the best
- It is not integrated with more objects
- Everything is biased and very little randomness which limits the outputs
- Although multiple lights work different attenuations are not completely blended
- Not naturally accurate



7-Future work

7.1-DIPOLE Implementation

The most obvious improvement that I could think of is the actual implementation of the dipole subsurface scattering model. Although I am quite happy with the results that I can achieve with the BRDF approximation the initial idea was to implement a separable BSSRDF and a complete subsurface scattering model, which in this case has been impossible.

With this approach we could have even better results and much more accurate representations of real materials.

7.2-Scene integration

Right now due to the biasing that I add in the scene it is impossible to add other objects. From the beginning of the project my goal was to render objects in the void as they did in the papers, however when working towards this goal I have limited other options that I didn't consider before. By integrating the subsurface scattering within the Ray tracer in a more stochastic way I could render whole scenes with different semi-translucent materials and create more intricated situations. This would also add naturality to the outputs and probably help to create more real life accurate materials.

8-Conclussion

Although I have not been able to do an actual implementation of a separable BSSRDF model I have learnt a lot about how light scattering works. My implementation might be rudimentary and limited, but it yields good results within its domain and permits lots of different interesting outputs that actually resemble subsurface light transport. Obviously, it is a shame that I have not been able to translate everything that I have learned into code, but I am sure that in the future (when I have a bit of time) I will be able to apply this knowledge and finish what I have started.

In conclusion, subsurface scattering and BSSRDF models are very deep and intricate subjects. The theory behind them is just a glance of their actual complexity, which lies more in digesting and actually translating the information into a real application. In retrospective, although I have not fully accomplished my goal, I have learnt a lot about the subject and I have experienced what is like dealing with it. I have been able to accomplish decent results and I have expanded my knowledge, for which I am very satisfied with the experience.