UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

Physically-based rendering of human skin

Author:
Roger Hernando Buch

Supervisors:
Antonio Chica
Pere-Pau Vázquez

A thesis submitted in fulfilment of the requirements
for the degree of Master in Innovation and Research in Informatics

in the

Facultat d'Informàtica de Barcelona
Department of Computer Science

October 15, 2015

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Facultat d'Informàtica de Barcelona

Department of Computer Science

Master in Innovation and Research in Informatics

Physically-based rendering of human skin

by Roger Hernando Buch

The plausible rendering of human skin is a challenging but crucial problem for photo-realistic graphics, due to the inherent complexity of the interaction between the light and the different layers of the skin.

This thesis explores a set of screen space physically-based subsurface scattering algorithms in order to improve the rendering of scanned human faces. More specifically, three subsurface scattering simulation methods and one forward scattering simulation method are introduced and discussed. Then, we present some extensions to those methods, in order to tackle the artifacts those methods produce. Additionally, we propose an strategy to modulate the subsurface scattering effect over the face using the local mesh curvature.

Furthermore, we implement an application which integrates the whole subsurface scattering rendering pipeline along with some Physically Based Rendering strategies to produce high quality renderings. Moreover, the application will be a test-bed to evaluate the implemented methods both in therms of performance and perceptual appealing.

We show that combining the subsurface scattering methods along with the extensions we propose in order to tackle the artifacts introduced by the screen space methods, improves the rendering quality of human faces, and also eliminates the artifacts introduced by the screen-space methods.

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Facultat d'Informàtica de Barcelona

Department of Computer Science

Master in Innovation and Research in Informatics

Physically-based rendering of human skin

by Roger Hernando Buch

El renderitzat realista de pell humana es un problema desafiant en el camp dels gràfics foto-realistes, degut a la complexa interacció inherent entre la llum, i les diverses capes de la pell.

Aquesta tesi, explora un conjunt d'algorismes que intenten imitar el fenomen del subsurface scattering en espai de pantalla, per tal de millorar el renderitzat de caps humans escanejats. Específicament, es presenten tres mètodes per simular l'efecte de subsurface scattering, i un per simular l'efecte de forward scattering. A continuació presentem algunes extensions per a aquests mètodes, amb l'objectiu de reduir els possibles errors visuals que aquests metodes produeixen. Addicionalment, es proposa un conjunt d'estratègies per modular l'efecte del subsurface scattering utilitzant la curvatura de l'objecte computada en espai de pantalla.

A més a més, implementem una aplicació que integra el pipeline per simular el subsurface scattering, a més a més d'utilitzar algunes estratègies de Physically Based Rendering per produir renders d'alta qualitat. Addicionalment, l' aplicació serà utilitzada com a banc de proves per avaluar el rendiment dels mètodes implementats.

En aquest treball, mostrem que combinant els mètodes per simular el subsurface scattering, i les extensions que proposem, podem obtindre renders de cares humanes d'alta qualitat, i eliminant els problemes que presenten els metodes de subsurfac escattering en espai de pantalla.

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Facultat d'Informàtica de Barcelona

Department of Computer Science

Master in Innovation and Research in Informatics

Physically-based rendering of human skin

by Roger Hernando Buch

El renderizado realista de piel humana es un problema desafiante en el campo de los gráficos foto-realistas, debido a la compleja interacción inherente entre la luz, y las diversas capas de la piel.

Esta tesis, explora un conjunto de algoritmos que intentan imitar el fenómeno del subsurface scattering en espacio de pantalla, a fin de mejorar el renderizado de cabezas humanas escaneadas. Específicamente, se presentan tres métodos para simular el efecto de subsurface scattering, y uno para simular el efecto de forward scattering. A continuación presentamos algún extensiones para estos métodos, con el objetivo de reducir los posibles errores visuales que estos producen. Adicionalmente, se propone un conjunto de estrategias para modular el efecto del subsurface scattering utilizando la curvatura del objeto computada en espacio de pantalla.

Además, implementamos una aplicación que integra el pipeline para simular el subsurface scattering, además de utilizar algunas estrategias de Physically Based Rendering para producir renders de alta calidad. Adicionalmente, la aplicación será utilizada como banco de pruebas para evaluar el rendimiento de los métodos implementados.

En este trabajo, mostramos que combinando los métodos para simular el subsurface scattering, y las extensiones que proponemos, podemos obtener renders de caras humanas de alta calidad, y eliminando los problemas que presentan los metodos de subsurface scattering en espacio de pantalla.

Acknowledgements

I would like to thank my supervisors, Antonio Chica and Pere-Pau Vázquez, for their exemplary guidance, support and encouragement throughout the elaboration of these work.

A special thanks to all the peers I have met during this two years and a half. This Master would not have been the same without them. Finally, I thank all my family and girlfriend for their constant and unconditional support.

Contents

List of Figures

Chapter 1

Introduction

Skin is probably one of the most difficult materials to reproduce in computer graphics. Indeed, the plausible rendering of human skin is a challenging but crucial problem for photo-realistic graphics, owing to the fact that the light enters at one point, scatters within it, and then exists at some another point. Such effects are poorly approximated by traditional surface shading models, which are usually described in terms of the Bidirectional Reflectance Distribution Function (*BRDF*) [Nic+92].

There are several factors that distinguish skin from other materials and put it in a very special category. The first one is the complexity of the skin itself, because the skin is made up of multiple layers (Figure 1.2), which are composed of different types of cellular level elements [INN]. Hence, they scatter light according their own composition. The second factor is a perceptual one. Human perception is highly specialized for perception of skin, and especially sensitive to facial appearances and expressions. Slightly errors in its simulation would be picked easier and for example if its shading is a bit off, it would look waxy, dead, or just awkward for our perception.

The light reflected and scattered through a translucent object such as the skin is described in terms of the Bidirectional Surface Scattering Reflectance Distribution Function (*BSSRDF*) [Nic+92] which is depicted in Figure 1.1. This effect softens the lighting on the skin, making it less geometrically dependent, because the light is absorbed and scattered in its way through the different layers of the skin. This softens the appearance of the skin imperfections, in such a way that non-lit areas in the pores and imperfections are filled with light coming from well-lit adjacent zones. Furthermore the skin is also affected by the forward scattering effect, which takes into account the light that travels through thin slabs like ears and nostrils, according to the transmittance factor of the skin, giving a reddish colour as we can see in the Figure 1.3.

Taking into account that human perception of skin is very accurate, in any rendered scene where a human-like character with visible skin appears, the accurate or believable simulation of the subsurface scattering is very important to make the scene convincing. For that reason, in this master thesis we want to explore and compare several methods to physically-based render translucent materials (human skin) in real time. A

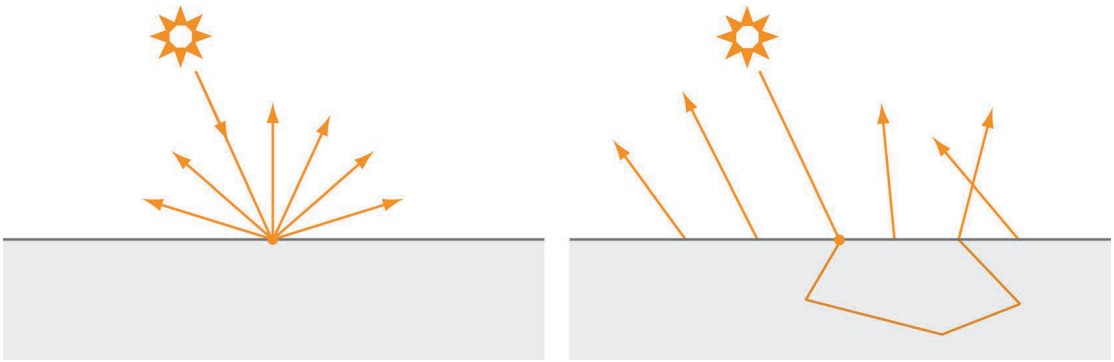

FIGURE 1.1: In an opaque object, light reflects in the same point where it hits the surface (left); in a translucent object, light scatters inside the object before exiting the surface (right)

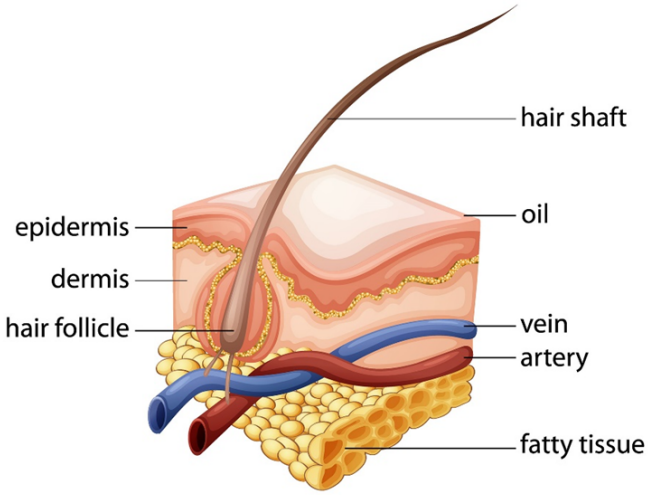

FIGURE 1.2: Figure depicting the different layers of teh skin

second objective is to propose and test several techniques intended to improve the render quality of raw scanned acquired human faces, that is, without manual prepocessing by an artist.

Raw scanned acquired models are 3d photographs of an object, which just include color and geometry information, and sometimes a normal map depicting the fine details of the skin (e.g. pores, wrinkles). This means that the models do not come with a mask which indicates what parts of the model are skin or not, thus causing some artifacts that we will try to solve, such as halos and blurring between skin and non-skin zones.

FIGURE 1.3: Lighted hand photograph showing the subsurface and forward scattering effect present on human skin

1.1 Objective

The aim of this thesis is to explore the real-time physically-based rendering of human skin, in particular to improve the rendering of scanned human faces. In order to do, so we defined the following sub-objectives:

- Implement three state-of-the-art methods in order to simulate the subsurface scattering effect [JG10; JJG15; JJ] and one to simulate the forward scattering [Jim+10]. The subsurface scattering simulation methods are physically based and what they try is to simulate the light diffusion through the skin in screen space. The forward scattering method obtains the thickness of the object and uses it to simulate the light transmitted through it. It is important to note that the simulation of subsurface scattering is done in screen space because we want it to run in real-time.
- Propose some extensions to the subsurface simulation algorithms. Some of the extensions are inspired by the literature, whereas the others are meant to solve some artifacts that the former methods present with our models.
- Implement an application which integrates the subsurface scattering methods and also some Physically Based Rendering (*PBR*) strategies in order to be able to generate high quality renders. Moreover, the application will be a test-bed to evaluate the performance of the implemented methods.
- Compare and test the implemented methods, and also evaluate their performance.

As a preliminary example, Figure 1.4 shows a comparison between two renderings with and without subsurface scattering done in our application. The skin of the subsurface scattering simulated image looks smoother than the other, and has a more natural appearance, in contrast to the skin of the image without subsurface scattering

FIGURE 1.4: The difference between render which simulate (left) and do not simulate (right) the subsurface scattering effect is noticeable because the latter one seems more unnatural.

1.2 Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews different approaches to render human skin that appear in recent literature. Chapter 3 introduces and explains the studied methods. Moreover, in Section 3.3 we propose several extensions to the subsurface scattering methods. Later the methods will be experimentally evaluated in Chapter 5. Chapter 4 presents the application we implemented in order to generate the renderings shown in this thesis, and evaluate the methods presented. Finally, Chapter 6 concludes this dissertation.

Chapter 2

Related Work

2.1 Introduction

Subsurface skin rendering/simulation techniques according to their temporal cost, can be initially classified into off-line or on-line rendering techniques. Off-line techniques are used, for instance, in movies, or in applications which need to compute accurately and in a photorealistic way the skin appearance and do not require interactive manipulation. Such techniques involve the accurate simulation of light rays going through the skin simulating their scattering effects, which is a very demanding process in terms of computational time, especially if solved for a high number of ray bounces.

In contrast, on-line techniques are useful for real-time environments such as video games, which need real-time interaction and manipulation. The main challenge of such techniques is to compute an approximation of the complex subsurface scattering effects, which should be good enough to be perceptually plausible, but at the same time fast enough to allow for real-time rendering. Furthermore, they should be easy to implement so that they integrate well with existing pipelines (e.g. rendering engines).

FIGURE 2.1: Comparison between an off-line technique such such as the one used in the render of Gollum from The Lord of the Rings movie (left), and an on-line one as in Metal Gear Solid V: The Phantom Pain game (right)

2.2 Off-line techniques

The scattering simulation inside translucent materials dates back to the radiative transfer equation [Cha60]. Off-line techniques compute the *BSSRDF* accurately, although the full multiple scattering simulation within a *BSSRDF* might be computationally prohibitive. A *BSSRDF* is an 8D function (Equation 3.1) that describes the light transport from one point to another for a given illumination and viewing direction. Monte Carlo simulation (ray tracing) is often the tool of choice to solve the light transport problem.

Jensen et al. [Jen+01], and later as a follow up article [JB02], used the complete *BSSRDF* along with a diffusion approximation to model subsurface scattering. The main idea behind this paper is to decouple the incident illumination from the evaluation of the *BSSRDF* by using a two-pass approach. In the first pass they compute the irradiance at selected points on the surface, and in the second pass the diffusion approximation is calculated using the dipole diffusion approximation from a pre-computed irradiance samples. The dipole diffusion approximation assumes that the material is homogeneous and semi-infinite, which is not the case of the human skin. This approach is substantially faster than directly sampling the *BSSRDF* since it only evaluates the incident illumination once at a given surface location. Figure 2.2 shows an example of their results using human faces.

FIGURE 2.2: Figure taken from [JB02], Face rendered using the *BRDF* approach (left) and rendered using the *BSSRDF* proposed by Jensen et al. [Jen+01; JB02] (right).

Later, Donner and Jensen [DJ05] extended the dipole model into a multipole one, which allows the modeling of multi-layered translucent materials, such as skin. They present a multipole diffusion approximation for light scattering in thin slabs, which generalizes to an arbitrary number of layers. This way, it enables the composition of arbitrary multi-layered materials with different optical parameters for each layer (i.e. roughness and refraction indices). This method is both accurate and efficient, and can be easily integrated into ray-tracing simulation methods using the dipole diffusion approximation to compute the scattering effects.

Following the previous work, the same authors introduced a photon diffusion technique to combine photon tracing and the diffusion approximation [DJ08]. This combination makes it possible to efficiently render highly scattering translucent materials while accounting for internal blockers, complex geometry, translucent inter-scattering, and transmission and refraction of light at the boundary causing internal caustics. Instead of sampling lighting at the surface as the previous techniques, this technique performs a photon tracing step to distribute photons in the material and store them volumetrically at the first scattering interaction with the material. Then, the radiant emittance at points on the material surface is computed by hierarchically integrating the diffusion of the light from photons.

More recently, D'Eon and Irving [DI11] presented a new *BSSRDF* for rendering images of translucent materials. Previous diffusion *BSSRDF*s are limited by the accuracy of classical diffusion theory. However, they introduce a modified diffusion theory which is more accurate for highly absorbing materials near the point of illumination. This new diffusion solution separates single and multiple scattering terms. Moreover, the authors derive an extended-source solution to the multi-layer searchlight problem by quantizing the diffusion Green's function obtaining a quantized-diffusion (QD) model. This can be done because the contribution from many depth sources at once arises from the separability of Gaussian functions. This allows the application of the QD multipole model to material layers several orders of magnitude thinner than previously possible and creates accurate results under high-frequency illumination.

Finally, Habel, Christensen and Jarosz [HCJ13] published the photon beam diffusion method, an efficient numerical method for accurately rendering translucent materials. Their approach interprets incident light as a continuous beam of photons inside the material. They leverage the improved diffusion model [DI11], but propose an efficient and numerically stable Monte Carlo integration scheme that gives equivalent results using only 3–5 samples instead of 20–60 Gaussians. This method can account for finite and multi-layer materials, and additionally supports directional incident effects at surfaces. Besides, their numerical approach allows to extend the accuracy and capabilities of the diffusion model and even combine it efficiently with more general Monte Carlo rendering algorithms.

Unfortunatelly, those methods are not suited for real-time because they require more than a few milliseconds to be computed; therefore, not being able to produce frame rates higher than 32 fps. Moreover, such methods are intended to be used with Monte Carlo rendering algorithms (e.g. ray tracing, photon mapping), which definitely are not able to produce high quality noiseless results in real time.

2.3 On-line techniques

On-line techniques are mainly based on, or try to improve, the approach presented in [BL03], which approximates subsurface scattering by blurring a 2D diffuse irradiance texture using a gaussian filter. While it is efficient and maps well to the GPU, it neglects the more subtle details of subsurface scattering.

The previous idea is extended by D'Eon and Luebke [dL08] to develop a high-quality real-time skin shader. The key idea of this paper is to approximate the multipole diffusion profiles of thin homogeneous slabs [DJ05] of a multi-layered translucent material such as the human skin, as a linear combination of carefully chosen gaussian basis functions, in order to use them to blur the irradiance signal in texture space. Since the gaussian convolution is separable, this allows transforming the expensive 2D convolutions into a cheaper set of 1D convolutions. This representation greatly accelerates the computation of multi-layer profiles and enables improved algorithms for texture-space diffusion and global scattering. In order to compute the light transmitted through thin parts of the object, the technique presented in [DS03] is used.

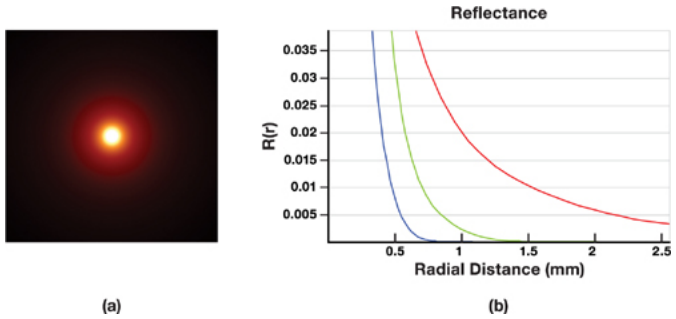

FIGURE 2.3: A visualization of the skin diffusion profile. Figure taken from [dL08].

Although the previously mentioned techniques are based on blurring the irradiance signal in texture space providing real-time performance, they scale poorly with the number of translucent objects in the scene, since the subsurface-scattering simulation needs to be performed on a per-object basis. To overcome this issue, Jimenez and Gutierrez [JG10] proposed to translate the simulation from texture to screen space. The diffuse irradiance of all objects is blurred once as a preprocessing step employing the sum-of-Gaussians formulation, thereby limiting subsurface scattering computations to the visible parts of the objects. Although the algorithm is faster due to the fact that it works on screen space, the algorithm has less information to work with, as opposed to algorithms that work in 3D or texture space. Therefore, the screen-space algorithm

loses irradiance in all points of the surface not seen from the camera, since only the visible pixels are rendered. Thus, the method cannot calculate the transmittance of light through thin parts of an object.

Moreover, due to this screen space lack of information, the method produces artifacts such as thin halos near the silhouette of the surface. Mikkelsen [Mik10] showed that the surface convolution by a Gaussian function can be weighted with a cross bilateral filter (*CBF*) over an image containing the edges from the observer point of view, thus solveing these silhouette errors.

The aforementioned method also fails to simulate the light transmitted through high-curvature features because of the lack of lighting information behind the objects. For this reason, its authors extended the method in [Jim+10] to simulate the transmittance of light through the skin. Basically, the authors proposed an approximation to reconstruct the irradiance on the back of an object. This, in turn, is used to approximate the transmittance based on the multipole theory [DJ05] theory. Such technique requires standard shadow maps as input, which eases its integration with rendering pipelines, also reducing the memory usage compared to previous work techniques which take transmittance into account.

In [SKP09], Shah et al. proposed a method that computes the *BSSRDF* using the dipole diffusion model. The main idea of this work is to employ the dipole diffusion model with a splatting approach to evaluate the integral over the surface area in an image-space framework, in order to compute the illumination due to multiple scattering. The main contribution of this paper is to take sample points on the surface visible from the light source, and splat the scattering contribution to all points visible to the viewer within the effective scattering range from each point. Finally, each point on the rendered surface receives the scattering contribution from all points that have an influence on it.

One of the most recent contributions [JJG15] proposes two real-time models to generate separable approximations of diffuse reflectance profiles to simulate subsurface scattering. It just uses two 1D convolutions, reducing both execution time and memory consumption, while delivering results comparable to techniques with higher cost. To approximate a 2D diffuse reflectance profile by a single separable kernel, the authors relax the requirement of the radial symmetry of the diffusion models. They also show how by combining importance sampling and jittering strategies such as the one used in [Hua+11], a small number of samples per pixel are enough in many cases of practical interest. In order to compute the light transmitted through thin parts of the object, the approach presented in [Jim+10] is used.

Unlike the previous described methods, which are based on gathering the neighboring light in order to simulate the subsurface scattering effects, the authors of [PB10] presented an approach where the effects of scattered light are pre-integrated into a texture.

They define three regions of the mesh, in which the subsurface scattering it is important to render the object realistically: zones with high surface curvature, zones with small surface bumps and the zones which lay onto the shadow edges. To obtain the scattering that occurs due to the curvature of the surface and the shadow edges, a precomputed subsurface texture texture is used, and accessed with the surface local curvature and the shdowness level of the region. To take into account the subsurface scattering due the small surface bumps, they propose a strategy of diffuse normals in which they filter the mesh normal map with R/G/B skin profiles (Figure 2.3). The authors claim that this strategy allows to achieve the non-local effects of subsurface scattering using only locally stored information.

Finally, Chen et al. [CLP14] presented Pre-integrated Deferred Subsurface Scattering (*PDSS*), a technique that adapts pre-integrated skin scattering to screen space, making it suitable for use in a deferred lighting pipeline and increasing its visual quality. Surface curvature is calculated in real time by evaluating the curvature from the gradient of the world space normals in the G-Buffer, avoiding curvature calculation artifacts. PDSS has the advantages of being independent of the scene geometry and scaling well in the number of lights and the number of objects. The method presented in [PB10] is used to calculate the subsurface scattering, which uses the curvature and a shadowing factor to look up into a pre-baked scattering texture and also the diffused normals. The approach presented in [Jim+10] is used to compute the light transmitted through thin parts of the object.

Chapter 3

Methods

As our objective is to improve the rendering appearance of scanned human faces but the hard constraint here is to do it in real time, in order to simulate the subsurface scattering, we have chosen methods which operate in screen space because they are fast and the additional memory consumption is relatively low in comparison to other sate-of-the-art methods. Moreover, those methods scale very well with the number of translucent objects in the scene, which is not so important in our particular case when only one model is in the scene, but it is worth to mention to understand the attractiveness of the screen space methods.

However apart from being fast and efficient, simulating the subsurface scattering in screen space produces some artifacts, such as halos. Furthermore, since our models do not have any mask that indicates which areas are skin or not, the algorithms diffuse the light between skin and non-skin zones. Later in this chapter (Section 3.3) we propose some extensions to the screen space methods in order to solve or alleviate the aforementioned problems.

We have implemented three methods in order to simulate the subsurface scattering effect in screen-space, and one to simulate the forward scattering, which are:

- Screen space subsurface scattering [JG10].
- Separable pre-integrated subsurface scattering [JJG15].
- Separable artistic subsurface scattering [JJ].
- Real-Time realistic skin translucency [Jim+10].

Some of the techniques are simple adaptations of the shader source code that accompanies some papers, but in any case, the images shown throughout this section come from our implementation of the described techniques.

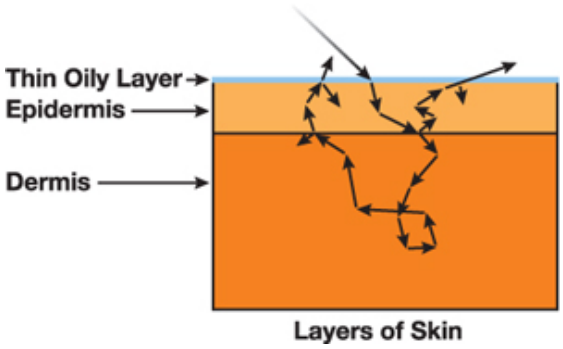

FIGURE 3.1: Image showing an example of light ray traveling trough the skin. Figure taken from [dL08].

3.1 Subsurface scattering

Subsurface scattering is a complex phenomena which describes how light enters an object, interacts with its different layers (Figure 3.1), and may exit at various points around the incident point or by transmitted through the object. This effect is described in terms of the *BSSRDF* S which relates the outgoing radiance $L_0(x_0, \overrightarrow{\omega_0})$ at a point x_0 to the radiant flux $\Phi_i(x_i, \overrightarrow{\omega_i})$ at the point x_i from the direction ω_i:

$$dL_0(x_0, \overrightarrow{\omega_0}) = S(x_i, \overrightarrow{\omega_i}; x_0, \overrightarrow{\omega_0}) d\Phi_i(x_i, \overrightarrow{\omega_i}) \tag{3.1}$$

The subsurface scattering effect can also be described using radially symmetric diffusion profiles. A diffusion profile is a function $R_d(x, y)$ that describes the light reflected around a normally incident pencil beam on the origin of a surface of an infinite half-space. For an homogeneous material, R_d is radially symmetric and can be characterized by a 1D diffusion profile $R_d(r)$, which describes how the light attenuates at each point as a function of the distance $r = ||(x, y)||$ from the incident point. To obtain such diffusion profiles, the authors of [Jen+01] use the diffusion theory to arrive at a diffusion equation:

$$D\nabla^2\phi(x) = \rho_\alpha(x) - Q_0(x) + 3D\overrightarrow{\nabla} \cdot \overrightarrow{Q}_1(x) \tag{3.2}$$

For an infinite media this equation has a simple solution, however for a finite media this equation has no analytical solution.

Applying a diffusion profile is simple. Consider a point $P(x, y)$ on the surface. We want to obtain the contribution of all the points around P. Part of the light arriving at such adjacent points will penetrate into the object and exit at P, with the specific attenuation given by the diffusion profile $R(r)$, expressed by:

$$M(x,y) = \int\int E(x',y')R_d(r')dx'dy' \tag{3.3}$$

being $M(x,y)$ the radiant exitance at point P, and $E(x,y)$ the irradiance around P. Equation 3.3 sums the contribution of each point around P, each of them weighted by the diffusion profile $R(r)$ according to its distance r to P. Threfore, it can be rewritten as a two-dimensional convolution:

$$M(x,y) = E(x,y) * R_d(r) \tag{3.4}$$

Carrying out the 2D convolution of Equation 3.4 is costly for real-time applications. However, if $R_d(r)$ can be approximated by a sequence of $2N$ $1D$ separable convolutions, A, represented as:

$$A(r) = \sum_{i=1}^{N} a_i(r) \tag{3.5}$$

where the approximation A is defined by $1D$ functions a_i.Due the radial symmetry of R_d the same functions a_i can be employed in both coordinate directions.

3.1.1 Screen Space sum of gaussinas subsurface scattering

From Equation 3.4, D'Eon and Luebke [dL08] observed that the skin diffusion profile (Figure 2.3) reassembles the aspect of a gaussian, so a sum of gaussian functions (Table 3.1) is suitable for approximation, being $R_d(r)$:

$$R_d(r) = \sum_{i=1}^{k} w_i G(v_i, r) \tag{3.6}$$

Variance	Color Weights		
	Red	Green	Blue
0.0064	0.233	0.455	0.69
0.0484	0.1	0.336	0.344
0.187	0.118	0.198	0
0.567	0.113	0.007	0.007
1.99	0.358	0.004	0
7.41	0.078	0	0

TABLE 3.1: Sum-of-gaussians parameters for a skin model depicted in [dL08].

Following the previous idea, Jimenez et al. [JG10] propose to perform this sum of gaussians approach in screen space instead of in texture space. The method requires the diffuse render, the linear depth of the scene, and the use of the stencil buffer to distinguish the zones which are skin or not. Next, it generates the different levels of gaussian blurring, and adds up all these levels using the weights of Table 3.1 in order to obtain the subsurface scattering contribution. Finally, it adds up the specular term to obtain the final render.

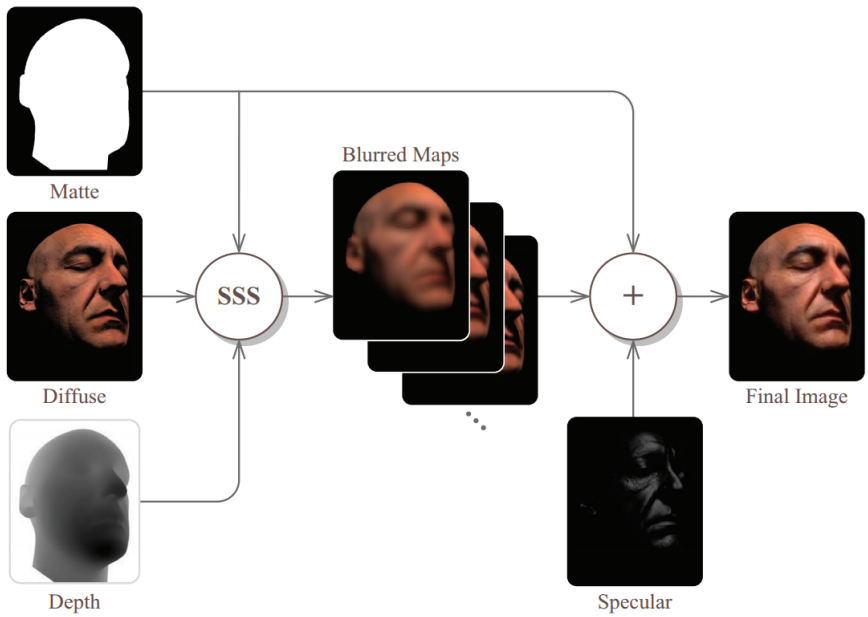

FIGURE 3.2: Overview of our screen space algorithm. Figure taken from [JG10].

It is worth noting that pixels located far from the camera will have narrower kernel sizes than pixels near the camera, so the width of the kernel is modified accordingly to

FIGURE 3.3: Comparison between two renders, the original render without subsurface scattering (left), and teh one using the scree space sum of gaussians technique (right).

the distance to the camera. Besides, a correction component is introduced to prevent scattering through neighboring pixels in screen space but farther away in the geometry. Figure 3.3 shows an example of our implementation of this algorithm, and Figure 3.4 shows the shader function we applied to compute one separable step of the gaussian blurring used to compute the subsurface scattering in screen space.

This way, the technique mimics the results of the method proposed by D'Eon and Luebke [dL08], at a fraction of its cost both in time and memory usage. What this method cannot reproduce or match from the previous method is to simulate the light transmitted through the thin slabs of skin. Therefore, this method must be used along with methods that simulate the forward scattering.

We implemented this approach by taking the example shader and adapting it to handle an arbitrary number of samples. Besides, we had to implement the generation of the gaussian kernel which is performed every time the number of samples changes.

```
uniform int ssss_n_samples;
uniform float offsets[100];
uniform float weights[100];

vec4 BlurSSSS(float sssWidth, float gauss_size, vec2 pixel_size, vec2 dir,
float correction, vec2 vUV, sampler2D color_texture, sampler2D depth_texture,
float fovy)
{
    // Fetch color of current pixel:
    vec4 colorM = texture(colorTex, texcoord).rgba;

    // Fetch linear depth of current pixel:
    float depthM = texture(depthTex, texcoord).r;

    // Calculate the sssWidth scale (1.0 for a unit plane sitting on the
    // projection window):
    float distanceToProjectionWindow = 1.0 / tan(0.5 * fovy);//fovy in rads
    float scale = distanceToProjectionWindow / depthM;

    // Calculate the final step to fetch the surrounding pixels:
    vec2 finalStep = gauss_size*sssWidth*scale*dir;

    vec3 colorBlurred = vec3(0);
    float weigths = 0;
    for (int i = 0; i < ssss_n_samples; i++)
    {
        //Fetch color and depth for current sample:
        vec2 despl = offsets[i] * finalStep;
        vec2 offset = vUV + despl;
        vec3 colorS = texture2D(color_texture, offset).rgb;
        float depth = texture2D(depth_texture, offset).r;

        //correction
        float s = min(correction * abs(depthM - depth), 1.0);
        colorS = mix(colorS, colorM, s);

        float weight = weights[i];
        colorBlurred += weight * colorS;
        weigths += weight;
    }
    return vec4(colorBlurred/vec3(weigths), 1);
}
```

FIGURE 3.4: Code used to compute one step of the separable blurring of the screen space subsurface scattering.

3.1.2 Separable subsurface scattering

The previous method shows that the skin diffusion profile can be approximated by a sum of separable gaussian functions. To be more precise six gaussian blurrings have to be applied to the diffuse render. In total, it performs twelve convolutions to an image. This is quite costly, and in order to alleviate this, the authors of [JJG15] propose a method to reduce it to just two separable convolutions.

Due to the non-separability of discretized representations of realistic diffusion profiles, it is not possible to fully reconstruct the effect of their convolution with 2D signals by a single separable kernel. It is, however, possible to completely reproduce a profile's behavior on a special class of signals: assuming that the irradiance is additively separable, $E(x, y) = E_1(x) + E_2(y)$. Then, the Equation 3.3 becomes:

$$M(x, y) = \int \int E(x', y') \frac{1}{||a_p||_1} a_p(r) dx' dy' \tag{3.7}$$

where a_p denotes the pre-integrated one dimensional approximation of R_d along a coordinate axis. Due to the radial symmetry of R_d, a_p is equal for each axis y or x, and $||R_d||_1 = ||a_p||_1$ by definition. Hence, the pre-integrated kernel A_p of the diffusion profile is defined as follows:

$$A_p(x, y) = \frac{1}{||R_p||_1} a_p(r) \tag{3.8}$$

The basic idea is to compute those kernels, and later apply them directly to simulate the subsurface scattering, just in the two separable convolutions.

In order to compute the kernels, the diffusion reflectance profiles are simulated using Monte Carlo modeling of light transport in multi-layered tissues (*MCML*) [Wan+92]. *MCML* simulates cylindrically symmetric tissue models, and outputs the diffuse reflectance profile as a 1D function. In order to be used in a render, this 1D function must be discretized. In general, those functions exhibit very uneven energy distribution, concentrating its energy near the origin, such that more samples are taken near the center, and few far away from it.

FIGURE 3.5: Comparison between two renders, the original render without subsurface scattering (left), and the one using the pre-integrated kernel subsurface scattering technique (right).

Figure 3.5 shows an example of our implementation of this algorithm, whereas Figure 3.6 shows the shader function we employed to compute one separable step of the blurring using the pre-integrated kernels described above.

Using the pre-integrated kernel computed exactly with a Monte Carlo simulation, this method produces high quality results, close to the ground truth of the simulation according to its authors.

We implemented this approach by taking the full computed kernels provided with the paper additional material. Then, we sample the kernels taking more samples near the center of the kernel and few far from it. Thus, obtaining the final kernel we will use for rendering. Finally, in order to use the kernel we obtain, we adapted the shader that we used at the previous method Figure 3.4 to handle a kernel with different weight per each color channel.

3.1.3 Artistic separable subsurface scattering

The previous method produces good results, but it is not practical in a production environment. Such environments require more flexible methods that could be easily tuned and tested, and usually the exact physical simulation it is not as important as the final appearance. Consequently, this is not the case of the previous method since each kernel needs to be simulated and discretized, this being a costly process and offering a limited

```
uniform int ssss_n_samples;
uniform vec4 kernel[100];

vec4 SSSBlur(vec2 texcoord, sampler2D colorTex, sampler2D depthTex,
float sssWidth,  vec2 dir, float fovy, bool follow_surf)
{
    // Fetch color of current pixel:
    vec4 colorM = texture(colorTex, texcoord).rgba;

    // Fetch linear depth of current pixel:
    float depthM = texture(depthTex, texcoord).r;

    // Calculate the sssWidth scale (1.0 for a unit plane sitting on the
    // projection window):
    float distanceToProjectionWindow = 1.0 / tan(0.5 * fovy);
    float scale = distanceToProjectionWindow / depthM;

    // Calculate the final step to fetch the surrounding pixels:
    vec2 finalStep = sssWidth * scale * dir;

    vec4 colorBlurred = vec4(0);
    vec3 weigths = vec3(0);

    for (int i = 0; i < ssss_n_samples; i++) {
        // Fetch color and depth for current sample:
        vec2 despl = kernel[i].a * finalStep;
        vec2 offset = texcoord + despl;
        vec4 colorS = texture2D(color_texture, offset).rgba;

        //correction
        float depth = texture(depthTex, offset).r;
        float s = min(correction * abs(depthM - depth), 1.0);
        colorS.rgb = mix(colorS.rgb, colorM.rgb, s);

        vec3 weight = kernel[i].rgb;

        // Accumulate:
        colorBlurred.rgb += weight * colorS.rgb;
        weigths += weight;
    }
    colorBlurred.rgb /= weigths;
    return colorBlurred;
}
```

FIGURE 3.6: Code used to compute on step of the separable blurring of the separable subsurface scattering.

artistic control over the final result. Consequently, Jimenez, Jarabo and Gutierrez [JJ] propose an artist friendly method to compute a diffusion kernel.

The artistic diffusion profile is formulated using the initial 1D sum of gaussians diffusion profile, but modified according to three parameters: *weight*, *strength* and *falloff*. *Weight* defines the global level of subsurface scattering (the filter width). *Strength* defines how much diffuse light penetrates the skin per channel and contributes to the subsurface scattering. Finally, *falloff* defines the amount of light travelling through the skin per channel. Therefore, the artistic friendly diffusion profile A_a is defined as follows:

$$A_a(x, y) = p\left[\frac{x * w}{0.001 + f}\right] * s + \delta(x) + (1 - t) \tag{3.9}$$

where w, s, and f are the parameters *weight*, *strength*, and *falloff*, respectively. $\delta(s)$ is a delta function which returns one if $x = 0$ and zero otherwise, and $p[x]$ is the initial 1D sum of gaussians diffusion profile of Equation 3.6.

We implemented this method firstly creating a function to build the kernel according to its parameters, and re-using the shader used by the previous method (Figure 3.6) but with the parameters, kernel, and number of samples defined accordingly. Figure 3.7 depicts an example of our implementation of this algorithm, using the baseline parameters to generate similar results to the skin diffusion profile, which are set as follows: $weight = 0.014$, $strength = (0.48, 0.40, 0, 28)$, and $falloff = (1, 0.37, 0.3)$. Figure 3.8 shows examples of subsurface scattering customization by modifying the *weight*, *strength* and *falloff* parameters.

Using this easily configurable kernel, this method produces a wide range of results according to its parameters. Moreover, the computation of the kernel is easy and fast, so this method allows to test and tune various sets of parameters according to an artistic criteria, making this method suitable for a production environment while producing high quality results.

FIGURE 3.7: Comparison between two renders, the original render without subsurface scattering (left), and the one using the separable artistic subsurface scattering technique (right).

FIGURE 3.8: Examples of editing the appearance of subsurface scattering: artistic kernel fitted to be similar to the original diffusion profile (left), $strength = (0.48, 0.40, 0.70)$ to increase the amount of blue light absorbed by the skin (centre), and $falloff = (0, 1, 0)$ to only allow green light to travel through the skin (right).

3.2 Forward Scattering Simulation

The aforementioned methods simulate the subsurface scattering over the visible surface of the skin shown by the screen, but do not simulate the light transmitted through thin geometry. Thus, in order to simulate the light coming from the light sources behind the observed geometry, we applied the method described in [Jim+10], because it produces nice results and it is widely used to simulate forward scattering effects.

The key fact of this method is to find a way to capture the distance that light travels inside the object(s), in other words, the thickness of the object from the light source to

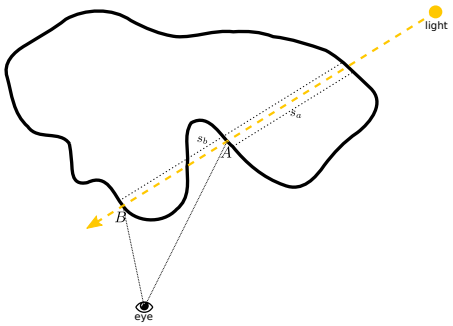

FIGURE 3.9: Light travels through the object. The thickness from the light source to the eye is needed to get the distance travelled by the light inside the object. S_b and S_a are the thickness associated with points B and A respectively.

the eye. Then, it employs a physically based function $\mathrm{T}(s)$ which relates the attenuation of the light to the object thickness.

One naive approach to obtain the thickness of the object would be to render the object from the light point of view, and subtract the depth of its front faces with the depth of its back faces. Then, using that thickness value to obtain The amount of light transmitted trough the object. However, that strategy would not capture the thickness of some points, as shown by the points A and B depicted in Figure 3.9. Moreover, with this strategy the full geometry must be rendered, thus not exploiting the face culling benefits, so a more clever strategy is required.

It is important to note that, if we are in a usual render pipeline which perform shadow mapping, we are already capturing this notion of how far the front faces of the object are from the light. Moreover, in the main render pass, where the visible parts of the objects are reprojected to the light space coordinates in order to found if these parts are occluded or not, we are implicitly obtaining information about how far the visible part is from the light. Then, by combining this information with the previous one, the object thickness s from the light to the eye is found.

Once s is found, $T(s)$ is used to get the light transmitted through the skin. This result is multiplied by the diffuse emission of the model, and then added to the diffuse reflectance of the main rendering, in order to feed the subsurface scattering algorithms with this information, and simulate the diffusion between the areas lighted from the forward scattering and its neighbours. Figure 3.10 shows an example of our implementation of this algorithm, and Figure 3.11 shows the shader function we used to compute the light transmitted through the skin.

$$T(s) = \sum_{i=1}^{k} w_i e^{-s^2/v_i} \tag{3.10}$$

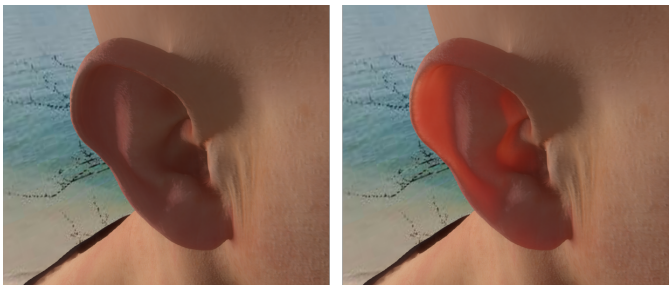

FIGURE 3.10: Comparison between ears, rendered without (left) and with (right) forward scattering.

$T(s)$ is defined in terms of the skin diffusion profile approximated by the sum of gaussians depicted in Table 3.1, being w_i its weights and v_i its variance.

We implemented this approach by taking the example shaders provided by the authors method, and adapting them to our framework.

```
vec3 transmittance(float translucency, float sss_width, vec3 world_position,
vec3 world_normal, vec3 light_vector, sampler2D shadow_map, mat4 light_V,0
mat4 ligh_BP, float light_far_plane)
{
    //scale factor
    float scale = 8.25 * (1.0 - translucency) / sss_width;

    //distance calculus
    vec4 shrinked_pos = vec4(world_position - 0.005 * world_normal, 1.0);
    vec4 shadow_pos = light_V*shrinked_pos;

    float d2 = (shadow_pos.z/shadow_pos.w)/-light_far_plane;
    shadow_pos = ligh_BP * shadow_pos;
    //linear depth stored in the shadow map texture
    float d1 = texture(shadow_map, shadow_pos.xy/shadow_pos.w).r;
    d1 *= light_far_plane;
    d2 *= light_far_plane;

    //distance between the point observed by the light and the point observed
    //by the camera
    float d = abs(d1 - d2)*scale;;
    float dd = -d * d;
    vec3 profile =  vec3(0.233, 0.455, 0.649) * exp(dd / 0.0064) +
                    vec3(0.1,   0.336, 0.344) * exp(dd / 0.0484) +
                    vec3(0.118, 0.198, 0.0)   * exp(dd / 0.187)  +
                    vec3(0.113, 0.007, 0.007) * exp(dd / 0.567)  +
                    vec3(0.358, 0.004, 0.0)   * exp(dd / 1.99)   +
                    vec3(0.078, 0.0,   0.0)   * exp(dd / 7.41);

        return profile * clamp(0.3 + dot(light_vector, -world_normal), 0.0, 1.0);
}
```

FIGURE 3.11: Code used to compute the light transmitted through the skin due to the forward scattering effect.

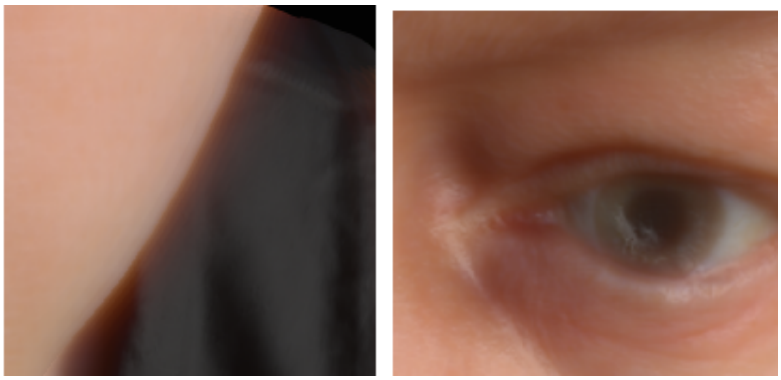

FIGURE 3.12: Figure depicting the problems we have detected using the screen space subsurface scattering algorithms. Halos (left) and incorrect diffusion (right).

3.3 Extensions

While testing the algorithms to simulate the subsurface scattering, we detected some problems (Figure 3.12). The screen space approaches produce halos between neighboring zones in image space but at different depth levels. Furthermore, as our scanned faces do not have a mask that indicates which areas are skin or not, the algorithms diffuse the light between skin and non-skin zones (e.g. eyes, hair).

Therefore, we propose some extensions to the methods in order to deal with the aforementioned problems, namely:

- Halo reduction: we propose an strategy extending the method proposed by Mikkelsen [Mik10], to effectively reduce the halos produced by the subsurface simulation techniques.
- Incorrect diffusion: in order to not diffuse light between skin and non-skin zones we propose to weight the kernels with a bilateral filter, to effectively improve the render quality.

In addition, we try to modulate the subsurface scattering according to the mesh local information (e.g. curvature), inspired by [PB10].

3.3.1 Reducing halos

The authors of the aforementioned subsurface scattering methods noticed the halos problems as well, and tried to tackle them with the correction factor (central image of Figure 3.16), which modulates the color of the samples which, although being near the central point of the diffusion profile in image space, are far away in the geometry, using the difference in depth between the central and the sampled points. Unfortunately, the

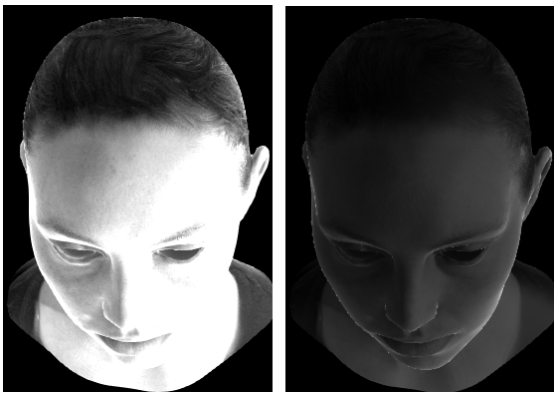

FIGURE 3.13: The image shows the unnormalized strategy proposed in [Mik10] (left) vs. our normalized strategy (right).

correction factors are not enough and Mikkelsen [Mik10] showed that using a cross bilateral filter (*CBF*) to weight the diffusion profile fixes the halos problem.

A *CBF*, works like a bilateral filter (Equation 3.13), but uses an auxiliary image to compute the weights instead of the image that is being filtered. *CBF* is characterized by the following equation:

$$CBF[I,E]_p = \frac{\sum_{q \in S} G_{\sigma_s} e^{-||p-q||} G_{\sigma_r} e^{-(E_p - E_q)} I_q}{\sum_{q \in S} G_{\sigma_s} e^{-||p-q||} G_{\sigma_r} e^{-(E_p - E_q)}} \tag{3.11}$$

where I is the original input image, E is the auxiliary image used to compute the difference of intensities, p are the coordinates of the current pixel to be filtered, S is the window centered in p, and G_{σ_r} and G_{σ_s} are the distance and color weighting factors, respectively.

The auxiliary image I is defined as an image that distinguishes between zones whose normal is perpendicular to the view direction and zones which are not. This creates an image of contours from the point of view of the observer, highlighting the edges between continuous areas in screen space but not in the geometry. That image is defined as follows:

$$I(p) = I(x(p)) * cos^3(\phi_i) \frac{||x(p)||^2}{cos(\phi_j)} \tag{3.12}$$

where $x(p)$ is the object point which is drawn in pixel p, ϕ_i is the angle between z-axis and the direction from the observer to the point $x(p)$, ϕ_j is the angle between the surface normal and the vector from $x(p)$ to the observer, and $I(x(p))$ the intensity of the pixel p.

FIGURE 3.14: The image shows the artifacts (black dots near teh edges) introduced by the *CBF* method when used directly with our shaders.

However, the effectively of this approach as is proposed depends on how far the model is from the projection plane, losing the power of detecting edges and therefore not removing the halos, as can be seen in Figure 3.13. In order to obtain the auxiliar image invariant to the distance, we always take the point $x(p)$ as if was always resting on the projection plane. Moreover, using this cross bilateral weighting directly within our shaders (Figures 3.6 and 3.4) produces some ugly artifacts as can be seen in Figure 3.14. Consequently, in order to fix those issues we had to modify our shaders to take into account that if the final color is black the original diffuse color must be used. In contrast to what Mikkelsen proposes, we use this technique along with the correction factors.

Figure 3.15 shows the auxiliary image used by the *CBF* with a highlighted area which is the same as used in the Figure 3.16, which shows the halos effect and its reduction using this method.

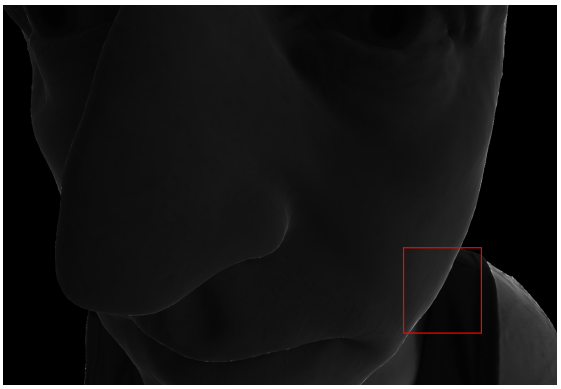

FIGURE 3.15: The auxiliar image used in the *CBF* weighting, in order to reduce the halos. The highlighted section corresponds to the region used in Figure 3.16.

FIGURE 3.16: Halos comparison: not using any method to correct them (left), using the correction factors (center), and using our modified *CBF* approach(right).

3.3.2 Reducing blurring between skin and non-skin zones

The scanned models are 3d photographs, and do not differentiate between the different parts of the captured reality. Therefore, the subsurface scattering simulation is equally applied to all elements of the object, whether they be skin or not, and due to the symmetry of the diffusion profile this produces the light being diffused from skin and non-skin zones (e.g. eyes, hair, cloths). The main difference between skin areas and non-skin areas is the color of such areas, being for example the eyes or hair of a completely different color compared to the surrounding skin.

In video games or production environments, an artist usually segments the areas which are or not skin, or each different part that composes the model comes directly as separated models. As our scanned models do not come with this information, we propose to use of a bilateral filter weighted by the color distance of neighboring pixels, in order to modulate the contribution of each sample. To take into account the human perception of colors, the color distance is performed in lab color space. Lab color space groups

FIGURE 3.17: Blurring between skin and non-skin zones: without simulating subsurface scattering (left), simulating subsurface scattering (centre), and using a bilateral filter to avoid blurring between skin and non-skin zones (right).

colors according to their perceptual similarity. Therefore, colors with less distance between them are going to be perceptually more similar, whereas colors with greater distance between them are not.

The bilateral filter we used is defined by the following equation:

$$BF[I]_p = \frac{\sum_{q \in S} G_{\sigma_s} e^{-||p-q||} G_{\sigma_s} e^{-(I_p - I_q)} I_p}{\sum_{q \in S} G_{\sigma_s} e^{-||p-q||} G_{\sigma_s} e^{-(c_p - c_q)}} \tag{3.13}$$

where I is the original input image, p are the coordinates of the current pixel to be filtered, S is the window centered in p, I_p and I_q are the lab color of the image I at pixel p and q respectively, and G_{σ_r} and G_{σ_s} are the distance and color weighting factors, respectively. As for the *CBF* method, we used G_{σ_r} as the diffusion kernel weight and G_{σ_s} is set to one.

Figure 3.17 shows the blurring of skin and non-skin zones, and how the bilateral filtering deals with the problem. It is not a perfect solution because it still blurs some high frequency details (i.e. thin hair), but it substantially improves the render quality.

3.3.3 Modulating the scattering effect

Inspired by the work of [PB10], which states that the scattering is more noticeable in higher curvature zones, we try to modulate our physically-based subsurface scattering with the curvature. Therefore, being the effect stronger at zones with higher curvature and weaker at zones with lower curvature. We also tried to modulate the forward scattering strength in the high curvature zones. Our contribution consists in trying to merge the physically-based subsurface scattering simulation with a non-physically based subsurface scattering technique, such as the curvature modulation.

The curvature is computed in screen space, using the normals of the neighboring pixels, to obtain the oriented gradient in each x and y axis, and finally obtain the curvature

FIGURE 3.18: Using the normal map stored normals to compute the screen space curvature results in a noisier curvature(left). Meanwhile, using the geometry normals reduces the noise (center), we smooth that curvature to feed the algorithm with an smooth curvature (right) free of high frequency discontinuities.

analyzing the magnitude of the variation of these axes. It is worth noting that, in order not to introduce high frequency discontinuities, the normals used to compute the curvature are the geometry normals and not the normal map normals (Figure 3.18). Besides, the curvature should be smoothed (i.e. mean blurring) to avoid such artifacts.

We have modulated the subsurface scattering effect with the curvature in three different ways (Figure 3.19):

- Increasing the subsurface scattering strength of a pixel according to its local curvature. Such strategy proved to be a poor option, because the screen space curvature is higher at the contours of the geometry, causing the subsurface scattering effect stronger on the edges. Therefore, increasing the filter size at the contours and making the halo artifacts more noticeable.
- Reducing the subsurface scattering effect at zones with lower curvature and remaining the same at zones with higher curvature. This strategy caused the zones with nearly zero curvature not to simulate the subsurface scattering at all, and breaking the high quality skin rendering.
- Reducing the subsurface scattering effect at zones with lower curvature up to a minimum and remaining the same at zones with higher curvature. This proved to be a good strategy because the subsurface scattering is simulated and strengthens the effect at the high curvature zones.

We have also tried to modulate the forward scattering strength according to the mesh local curvature, which proved to be a bad idea since it produces ugly artifacts (i.e. extremely bright translucency areas) at high curvature zones as shown in Figure 3.20.

FIGURE 3.19: Face rendered without the subsurface scattering effect (top left). Face rendered simulating the subsurface scattering but not modulating its effect (middle top). Face rendered using the blurred screen space curvature (top right) to modulate its subsurface scattering effect: increasing the subsurface scattering strength according to its local curvature (bottom left), reducing the subsurface scattering effect at zones with lower curvature (middle botom), and reducing the subsurface scattering effect at zones with lower curvature up to a minimum (bottom right).

FIGURE 3.20: When the forward scattering is modulated with the screen space curvature, it produces bright artifacts at high curvature areas (e.g. nostrils).

Chapter 4

Application

4.1 Description

In order to test all the methods, an OpenGL application has been developed, which allows to explore an scene, load different kinds of models, load textures and use them in the render phase to enhance the model appearance, use the previously mentioned methods to simulate and render human skin, and also to tune the algorithm parameters in an interactive fashion. Moreover, the application uses some physically-based-rendering techniques to improve the quality of the scene rendering, which will be further explained in the next sections. Figure 4.1 shows the main view of the developed application.

4.1.1 Overview

The application can be subdivided into the different rendering steps used to generate the final rendering, which are:

- Shadow mapping. At the shadow mapping step, is where the geometry is rendered from the light point of view to create the shadow maps, which are used later to compute the shadows and the forward scattering effect.
- Main rendering step. At main rendering step, the diffuse and specular scene colors are rendered, and the auxiliary *CBF* factor and the screen space curvature are computed.
- Subsurface scattering simulation step. The subsurface simulation step is the stage that uses the methods described in Section 3.1 to simulate the subsurface scattering effect.
- The speculars addition step. At the speculars addition step, the specular color is added to the subsurface scattering simulated rendering obtained from the previous rendering step.

FIGURE 4.1: Snapshot of the application interface.

- Tonemap step. At the tonemap step, the final rendering is composed, re-correcting the gamma of the scene object to be correctly shown by the screen, and also renders the background of the scene.

Figure 4.2 shows the abovementioned application rendering pipeline.

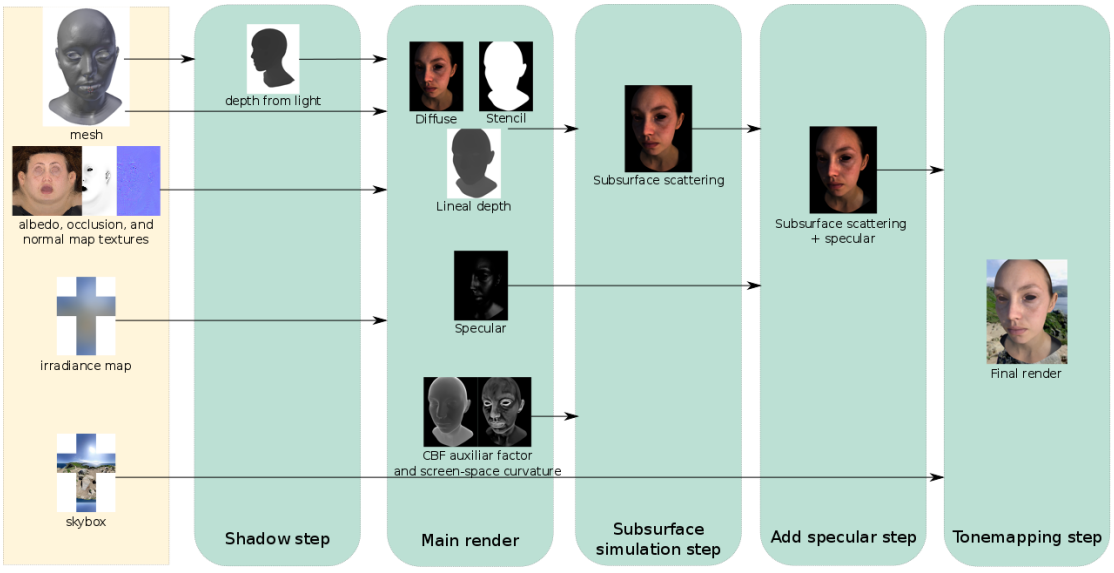

FIGURE 4.2: Application pipeline

4.1.2 Physically-based speculars

Phong specular model is well known and widely used in computer graphics, but using a more accurate physically based surface reflectance model will improve the image quality. The Phong model fails to capture increased specularity at grazing angles and it is not physically plausible, since it outputs more energy than it receives. The physically-based specular BRDF terms are typically based on microfacet theory [Hil+15], which assumes that the surface is composed of many microfacets, too small to be seen individually. The specular *BRDF* term is defined as follows:

$$C_{spec}(l,v) = \pi \frac{F(l,h)G(l,v,h)D(h)}{4(n\cdot l)(n\cdot v)} L_c(n\cdot l) \tag{4.1}$$

where l and v are the light and the viewing vector respectively, h is the half vector $h = \frac{l+v}{|l+v|}$, and L_c is the light color. D is the distribution of normals function which defines the concentration of microfacets that are oriented such that they could reflect light from l into v. G is the shadow-masking function, which defines the percentage of microfacets with h as their normal vector that are not shadowed or masked. F is the Fresnel therm which defines the fraction of light reflected from an optically flat surface. Owing to the fact that the full Fresnel equation is rather complex, and the parameters that it uses are not convenient for a production environment, the Fresnel term is approximated with the Slick's Fresnel equation [Sch94]:

$$F_{slick}(F_0,l,h) = F_0 + (1-F_0)(1-(l\cdot h))^5 \tag{4.2}$$

where F_0 is the characteristic specular reflectance of the material. In order to simulate the skin specular we set $F_0 = (0.029, 0.029, 0.029)$.

As a shadow-masking function $G(l,v,h)$ we use, as many games do, the so-called "implicit" masking function (Equation 4.3), which is equal to one when $l = n$ and $v = n$, and equal to zero for either glancing view angles or glancing light angles. That means that the probability of a microfacet being occluded by other microfacets increases with viewing angle, going up to 100% in the limit. Furthermore, it simplifies the specular equation, thus reducing the computation time.

$$G_{implicit}(l,v,h) = (n\cdot l)(n\cdot v) \tag{4.3}$$

Finally, as a distribution of normals function $D(h)$ we used the so-called "Phong NDF" (Equation 4.4), where the parameter α_r is the roughness parameter. High α_r values represent smooth surfaces, whereas low values rough ones. Roughness is not an artist-friendly parameter because it is arbitrarily huge (infinite for perfect mirrors) and is

FIGURE 4.3: Comparison between the two specular models, phong (left) and BRDF (right).

expressed in terms of the glossiness s (which has a range from zero to one), being $\alpha_r = \alpha_m a x^s$. Usually, α_{max} is set to 8192.

$$D_{phongt}(h) = \frac{\alpha_r + 2}{2\pi}(n \cdot h)^{\alpha_r} \tag{4.4}$$

Finally, our physically-based specular *BRDF* is defined as follows:

$$C_{spec}(l, v) = \frac{\alpha_r + 2}{8}(n \cdot h)^{\alpha_r} F_{slick}(F_0, l, h) L_c(n \cdot l) \tag{4.5}$$

4.1.3 Indirect illumination

In order to enhance the visualization, we use a skybox to depict the scene surrounding environment. A skybox is a cube map which reproduces a panoramic view of an environment that has been mapped to the inside of a cube.

Since the scene is surrounded by that environment, the interaction of the environment light and the scene must be taken into account. A good way to simulate the global light from the environment is to consider a cube map as a light source. This technique is called Image Based Lighting (*IBL*) [CON99] and it is often associated with *PBR* pipelines.

Given a point to be shaded with normal n, color C_p, and occlusion factor O_p, the amount of environment light received by the point coming from the environment is defined as follows:

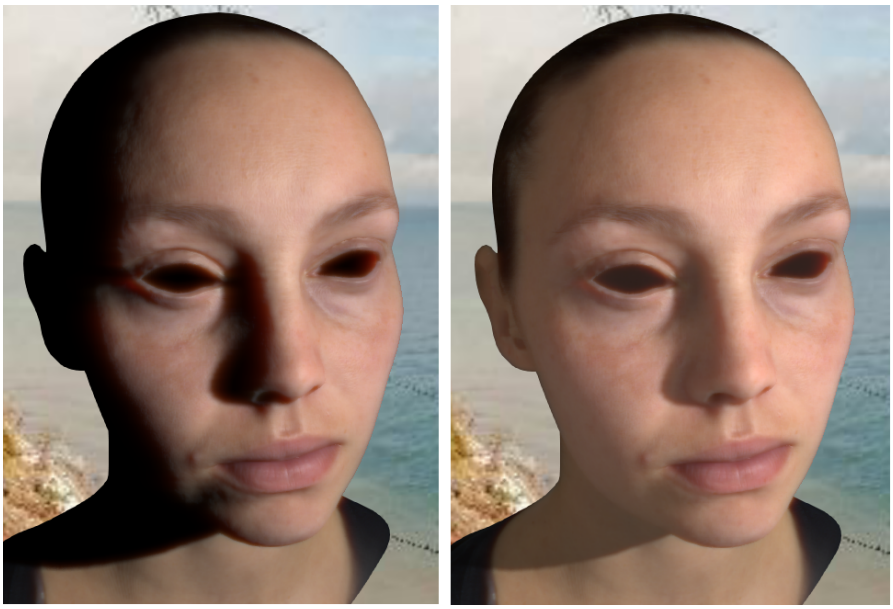

FIGURE 4.4: Comparison between two renders without (left) and with (right) indirect lighting.

$$EM_{diff}(n) = C_p AO_p \frac{\sum_{k\in\Omega} max(0, l_k \cdot n) L(l_k)}{\sum_{k\in\Omega} max(0, l_k \cdot n)} \tag{4.6}$$

where l_k is a direction over the hemisphere Ω centered at n , $L(l_k)$ is the cube map color at the direction l_k. Computing this equation in real time would be very costly and prohibitively because a lot of look-ups to the cube map would be needed.

From Equation 4.6 we can see that points with the same normal will have the same environment contribution. Hence, we can create a pre-filtered irradiance map such that each value maps the environment light at a given direction:

$$EM_{diff}(n) = \frac{\sum_{k\in\Omega} max(0, l_k \cdot n) L(l_k)}{\sum_{k\in\Omega} max(0, l_k \cdot n)} \tag{4.7}$$

Then, during the rendering, only one texture look-up to the irradiance map is needed.

The occlusion component O_p, which says how exposed each point in a scene is to ambient lighting, is given by the ambient occlusion factor. Particularly in our application, the ambient occlusion is not computed in real time, but baked into a texture for convenience instead. Anyway, this factor can also be computed in real time using a screen space approach.

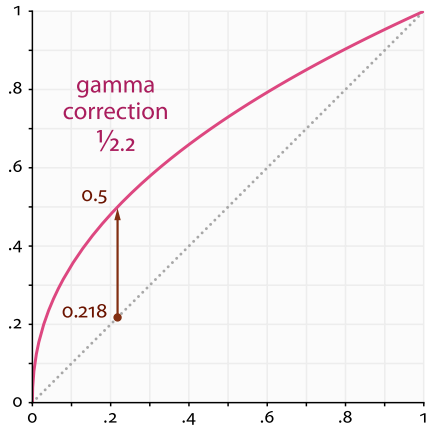

FIGURE 4.5: Example of gamma correction. The dashed line represents the linear intensities perceived by the camera, and the continuous line the gamma encoded intensities. Figure adapted from https://commons.wikimedia.org/wiki/File:GammaFunctionGraph.svg.

4.1.4 Importance of Linearity

It is important to note that, as we are working with 3d scanned models whose color is captured by a camera, the color is stored in a non-linear way. This is due to the fact that cameras do not perceive the light the same way humans do. With a digital camera, when twice the number of photons hit the sensor, it receives twice the signal (a "linear" relationship). Instead, we perceive twice the light as being only a fraction brighter (a "nonlinear" relationship). Compared to a camera, we are much more sensitive to changes in dark tones than we are to similar changes in bright tones.

Therefore, when a camera captures a photo and saves it, the information is "gamma encoded" such that the image is stored in a non linear way, redistributing the native camera tonal levels to ones more perceptually uniform. Consequently, devoting more bits to describe darker tones and fewer bit to describe brighter tones. Later, when the image is visualized on a monitor, a step of "gamma correction" is automatically performed by the monitor to convert the light stored in the image into light from the original scene.

When a pixel is read from a texture it pixel is "gamma encoded" and therefore non linear, but the rendering color operations assume linear color so we should feed the render algorithms with linear color values, and later "gamma encodeing" the final render to be shown correctly by the screen. If this steps of correcting the gamma, use the algorithm with the gamma corrected colors, and finally "gamma encode" the image are not performed, the result of the algorithms would be wrong (e.g. yellowish specular highlights), as shown in Figure 4.6.

FIGURE 4.6: Comparison between two renders without (left) and with (right) gamma taken into account.

Wen the cammera codifies the image in order to store it raises the color intensity to some power *gamma* (c^{γ}). The precise value of gamma which encodes an image is usually specified by a color profile that is embedded within the file, which is usually $\gamma = 1/2.2$. If no color profile is embedded, then a "standard" gamma of $1/2.2$ is assumed. This way when a color is read from an image to linearize its value a "gamma correction" of $\gamma = 2.2$ is performed. Later, a "gamma encoding" of $\gamma = 1/2.2$ is applied to render the final image on the screen.

Moreover, since we are internally working with linear color not checking if it's rage exceeds the typically RGB range (zero to one). HDR tone mapping strategies can be used to correct the range of the rendering from zero to one, such as linear, exponential, Reinhard[Rei+02], or filmic tone mapping.

Chapter 5

Evaluation

In this chapter we are going to evaluate and compare the different methods used, and also the overall performance of the application that integrates all the methods, and each of its non subsurface scattering simulation methods.

5.1 Performance

All test were performed using a viewport size of $1681x942$ and a model of 900096 triangles, in a computer with a Nvidia GeForce GTX 970M graphics card with 3Gb of memory, Intel Core i7-4710HQ CPU, 16Gb of RAM memory.

Each time sample actually is an average of hundreds of frames, in each frame the time elapsed in each of the application steps is measured, using the OpenGL *glFinish()* call to ensure that the graphics card has finished the operations associated with the task.

As we stated at Secction 4 the application can be subdivided into different rendering steps: shadow mapping, main render, subsurface simulation, add speculars, and tonemapping. In Table 5.1 we evaluate the time elapsed in each step, except the subsurface simulation step, for which each included method will be separately evaluated next. For all the renders shown in this thesis we also enabled FXAAx6 Nvidia graphics card antialiasing to improve the render quality.

Note that the execution time of every method described is dominated by the convolution computation, so their timings are roughly proportional to the number of convolutions they require. Therefore, the method using sum of six gaussians approach is

Application			
Shadow map	MainRender	AddSpecular	Tonemapp
0.447 ms	1.262 ms	0.148ms	0.977 ms

TABLE 5.1: Elapsed time of each non-subsurface scattering simulation application render steps.

View	Gausian sum	Artistic	Pre-int Kernel
Close	9.428 ms	1.731 ms	1.799 ms
Mid	2.01 ms	0.492 ms	0.487 ms
Far	0.676 ms	0.312ms	0.36 ms

TABLE 5.2: Elapsed time of each subsurface scattering simulation algorithm, with different points of view to fill the screen with more or less skin area. In this test the number of samples was set to 20.

be much slower than the methods using just two separable convolutions. As well as the methods presented are in screen space, the amount of skin filling the screen would affect the algorithms performance, as Table 5.2 shows. Moreover, the performance of subsurface simulation methods is also influenced by the number of samples of that such algorithms take. Increasing more rapidly in the sum of six gaussian method, as shown in Figure 5.1.

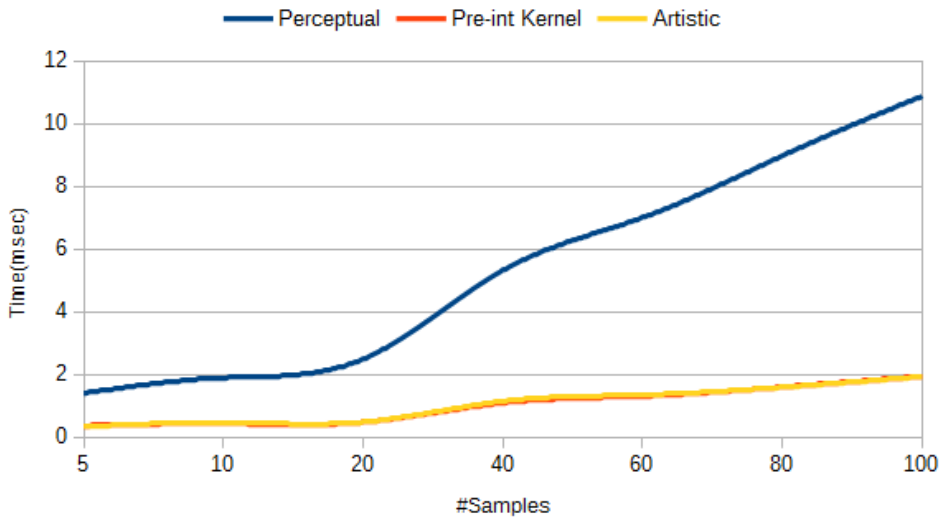

FIGURE 5.1: Graphic showing how the number of samples taken by the algorithms affects its performance.

The methods that perform two one-dimensional convolutions perform a total of twice samples, and the screen space sum of six gaussian method performs twelve times the samples, thus being much slower than the other methods, since it takes six times more samples than the others. The time each extension adds to the render is given by the number of samples that are added to the algorithm, being twice the samples in both curvature and *CBF*. From Figure 5.1 it can be seen that the overhead added by those extensions would be a real performance killer for the sum of six gaussians method, whereas the other two methods could assume the overhead and perform in real time.

5.2 Perceptual

Figure 5.2 shows the different subsurface scattering simulation methods used to render the same scene. We can observe that each of the presented methods enhances the rendering by softening the appearance compared to the render without scattering, as well as changing a little bit the color of the rendering according to the diffusion profile used.

Perceptually, in our opinion, the best results are obtained with the pre-integrated method combined with the *CBF* in order to remove halos, and the bilateral filter using the lab color space to eliminate the diffusion between skin and non-skin zones. The rendering with the least amount of artifacts that we have been able to achive is shown in Figure 1.4.

In order to correctly evaluate the methods, perceptually speaking a formal user study should be performed. Due to the lack of time a user study is left as future work. With a user study we could conclude which methods are more perceptually realistic and which are not, and how the Extensions we propose affect positively or negatively the final rendering.

The methods implemented in this thesis, even though meant to simulate human skin subsurface scattering effect, could also be used to simulate other translucent materials, such as marble or leaves as shown in Figure 5.3.

FIGURE 5.2: Methods comparison: original (top left), screen space sum of gaussians (top right), separable Artistic (bottom left), and pre-integrated kernel (bottom right).

FIGURE 5.3: The subsurface simulation methods used also can be used to simulate other translucent materials such as marble or leaves.

Chapter 6

Conclusions

6.1 Conclusions

The main contributions of this master thesis are:

- The development of an application that covers most of the state of the art screen space skin rendering methods.
- The implementation of some extensions to the current methods: halos reduction, incorrect diffusion between skin and non-skin zones, and modulate the subsurface scattering strength overt the mesh.
- The evaluation of the methods both in terms of quality and rendering speed.

This thesis had explored different algorithms of real-time physically-based rendering of human skin. Next, we have introduced the problematics associated and the importance of having a high quality rendering of human skin. Then, a review of the literature associated has been made. And after that different methods to render skin realistically have been presented. We have explained three screen space methods to simulate the subsurface scattering effect based on diffusion profiles, and one to simulate the forward scattering effect.

Later, we have proposed and explained an **extension** to improve the rendering quality, by modulating the subsurface scattering effect with the local curvature as is done in the literature. Moreover, we have also proposed two extensions to fix the artifacts which those methods present, such as halos and diffusion between skin and non-skin zones.

We have also implemented a full featured **application** that integrates the methods to simulate the skin rendering, and the extensions we propose to them. This application has been used as a test-bed system for the different algorithms in the literature, and the extensions we have proposed. It also includes other physically-based techniques in addition to the simulation of skin, in order to increase the realism and rendering quality. As a result, our application is capable of rendering high quality human scanned faces as shown in the extensive amount of examples shown in this memoir.

We have also **analyzed the performance** of the presented methods as well as the improvements we have proposed, in order to show the impact the methods have on the overall rendering time. We have shown that the method which performs a sum of six gaussians to simulate the subsurface scattering is the most costly method and depending on the number of samples and the skin elements on the scene could not be a viable option to real-time rendering of human skin. In terms of the perceptual validity of the methods, further analysis should be made (i.e. user study), but subjectively we think that the best method, perceptually speaking, is the separable pre-integrated kernel method coupled with the techniques we have presented to solve the artifacts it presents.

6.2 Future Work

As a future work, we would like to explore non-physically-based methods to render the skin such as [CLP14], and explore non-real-time physically-based methods such as [HCJ13], in order to further complete the study about physically-based skin rendering presented in this thesis.

In addition, in order to improve the re-usability of the forward scattering method it would be usefull, to include method to deduce its weights from the kernels used by the separable artistic and pre-integrated methods.

Finally, it would be interesting to explore on-the-fly or pre-process methods to segmentate the skin from the non-skin zones, and create a mask to effectively distinguish between zones where the subsurface scattering should be applied and zones which it should not.

Bibliography

[BL03] George Borshukov and J. P. Lewis. “Realistic Human Face Rendering for "The Matrix Reloaded"”. In: *ACM SIGGRAPH 2003 Sketches &Amp; Applications*. SIGGRAPH ’03. 2003.

[Cha60] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960. ISBN: 9780486605906. URL: http://books.google.es/books?id=CK3HDRwCT5YC.

[CLP14] Xi M. Chen, Timothy Lambert, and Eric Penner. “Pre-integrated Deferred Subsurface Scattering”. In: *ACM SIGGRAPH 2014 Posters*. SIGGRAPH ’14. 2014. ISBN: 978-1-4503-2958-3.

[CON99] Brian Cabral, Marc Olano, and Philip Nemec. “Reflection Space Image Based Rendering”. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 165–170. ISBN: 0-201-48560-5. DOI: 10.1145/311535.311553. URL: http://dx.doi.org/10.1145/311535.311553.

[DI11] Eugene D’Eon and Geoffrey Irving. “A Quantized-diffusion Model for Rendering Translucent Materials”. In: *ACM Trans. Graph.* 30.4 (July 2011), 56:1–56:14. ISSN: 0730-0301. DOI: 10.1145/2010324.1964951. URL: http://doi.acm.org/10.1145/2010324.1964951.

[DJ05] Craig Donner and Henrik Wann Jensen. “Light Diffusion in Multi-layered Translucent Materials”. In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 1032–1039. ISSN: 0730-0301. DOI: 10.1145/1073204.1073308. URL: http://doi.acm.org/10.1145/1073204.1073308.

[DJ08] Craig Donner and Henrik Wann Jensen. “Rendering Translucent Materials Using Photon Diffusion”. In: *ACM SIGGRAPH 2008 Classes*. SIGGRAPH ’08. 2008.

[dL08] Eugene d’Eon and David Luebke. “Advanced Techniques for Realistic Real-Time Skin Rendering”. In: *GPU Gems 3*. Ed. by Hubert Nguyen. Addison-Wesley, 2008, pp. 293–347.

[DS03] Carsten Dachsbacher and Marc Stamminger. “Translucent Shadow Maps”. In: *Proceedings of the 14th Eurographics Workshop on Rendering*. EGRW ’03. 2003. ISBN: 3-905673-03-7.

[HCJ13] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. "Photon Beam Diffusion: A Hybrid Monte Carlo Method for Subsurface Scattering". In: *Computer Graphics Forum (Proceedings of EGSR)* 32.4 (June 2013). DOI: 10.1111/cgf.12148.

[Hil+15] Stephen Hill et al. "Physically Based Shading in Theory and Practice". In: *ACM SIGGRAPH 2015 Courses*. SIGGRAPH '15. 2015. ISBN: 978-1-4503-3634-5.

[Hua+11] Jing Huang et al. "Separable Approximation of Ambient Occlusion". In: *Eurographics 2011 - Short papers*. 2011.

[INN] Takanori Igarashi, Ko Nishino, and Shree K. Nayar. *The Appearance of Human Skin: A Survey*.

[JB02] Henrik Wann Jensen and Juan Buhler. "A Rapid Hierarchical Rendering Technique for Translucent Materials". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 576–581. ISSN: 0730-0301. DOI: 10.1145/566654.566619. URL: http://doi.acm.org/10.1145/566654.566619.

[Jen+01] Henrik Wann Jensen et al. "A Practical Model for Subsurface Light Transport". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 511–518. ISBN: 1-58113-374-X. DOI: 10.1145/383259.383319. URL: http://doi.acm.org/10.1145/383259.383319.

[JG10] Jorge Jimenez and Diego Gutierrez. "GPU Pro: Advanced Rendering Techniques". In: ed. by Wolfgang Engel. AK Peters Ltd., 2010. Chap. Screen-Space Subsurface Scattering, pp. 335–351.

[Jim+10] Jorge Jimenez et al. "Real-Time Realistic Skin Translucency". In: *IEEE Computer Graphics and Applications* 30.4 (2010), pp. 32–41.

[JJ] Diego Gutierrez Jorge Jimenez Adrian Jarabo. *Separable Subsurface Scattering*. Tech. rep. Universidad de Zaragoza.

[JJG15] Adrian Jarabo Christian Freude Thomas Auzinger Xian-Chun Wu Javier von der Pahlen Michael Wimmer Jorge Jimenez Károly Zsolnai and Diego Gutierrez. "Separable Subsurface Scattering". In: *Computer Graphics Forum* (2015), n/a–n/a. ISSN: 1467-8659. DOI: 10.1111/cgf.12529. URL: http://dx.doi.org/10.1111/cgf.12529.

[Mik10] Morten S Mikkelsen. "Skin Rendering by Pseudo–Separable Cross Bilateral Filtering". In: *Naughty Dog Inc* (2010), p. 1.

[Nic+92] F. E. Nicodemus et al. "Radiometry". In: ed. by Lawrence B. Wolff, Steven A. Shafer, and Glenn Healey. USA: Jones and Bartlett Publishers, Inc., 1992. Chap. Geometrical Considerations and Nomenclature for Reflectance, pp. 94–145. ISBN: 0-86720-294-7. URL: http://dl.acm.org/citation.cfm?id=136913.136929.

[PB10] Eric Penner and George Borshukov. "GPU Pro 2: Advanced Rendering Techniques." In: ed. by Wolfgang Engel. AK Peters Ltd., 2010. Chap. Pre-Integrated Skin Shading, pp. 41–55.

[Rei+02] Erik Reinhard et al. "Photographic Tone Reproduction for Digital Images". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 267–276. ISSN: 0730-0301. DOI: 10.1145/566654.566575. URL: http://doi.acm.org/10.1145/566654.566575.

[Sch94] Christophe Schlick. "An Inexpensive BRDF Model for Physically-based Rendering". In: *Computer Graphics Forum* 13.3 (1994), pp. 233–246. ISSN: 1467-8659. DOI: 10.1111/1467-8659.1330233. URL: http://dx.doi.org/10.1111/1467-8659.1330233.

[SKP09] Musawir A. Shah, Jaakko Konttinen, and Sumanta Pattanaik. "Image-space Subsurface Scattering for Interactive Rendering of Deformable Translucent Objects". In: *IEEE Comput. Graph. Appl.* 29.1 (Jan. 2009), pp. 66–78. ISSN: 0272-1716. DOI: 10.1109/MCG.2009.11. URL: http://dx.doi.org/10.1109/MCG.2009.11.

[Wan+92] Lihong Wang et al. "Monte Carlo modeling of light transport in multi-layered tissues in standard C". In: *Cancer Center, University of Texas, Houston,Tex* (1992).