Real-time Rendering of Translucency in Skin

by

David Whelan, B.Sc.

Thesis

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science

University of Dublin, Trinity College

September 2009

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

David Whelan

September 8, 2009

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

David Whelan

September 8, 2009

Acknowledgments

To Wen, for without her encouragement I would have never enrolled for this masters, nor submitted this thesis. To my family for the additional support and encouragement and all my classmates who made the year enjoyable and put in the hard work alongside me. It was worth it. I would also like to thank my supervisor, Veronica Sundstedt for her help and ideas, Diego Guiterrez and Jorge Jimenez for their previous work in this domain and ideas for my work.

David Whelan

University of Dublin, Trinity College
September 2009

Real-time Rendering of Translucency in Skin

David Whelan
University of Dublin, Trinity College, 2009

Supervisor: Veronica Sundstedt

Accurate rendering of human skin in real-time is becoming an important part of video games effects. Skin as a material is highly complex to render accurately. In real-time rendering, techniques are required to approximate the scattering behaviour of skin while maintaining a real-time frame rate. A by-product of subsurface scattering is translucency which is not catered for when rendering skin in video games.

We propose a skin shader that can render translucency in thin regions of skin simulating the translucency effect of subsurface scattering while leveraging previous work on screen space approximations of subsurface scattering. Our technique computes and renders translucency in screen-space which allows it to scale well, ensuring real-time performance even with multiple on-screen models. The shader is proposed as an extension of previous work by Jimenez *et al.* [24] which uses a screen space diffusion approximation of texture space diffusion as implemented by d'Eon *et al.* [10].

This work was leveraged to add translucency and demonstrate that this can be achieved at a relatively low computational cost. Optimisations are demonstrated that

can be applied to the algorithm to provide significant performance enhancements. We run a perceptual evaluation to fine tune the algorithm output as well as demonstrate a correlation with measured empirical evidence.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Real-time computer graphics researchers strive to model lighting systems in a manner that provides the most beneficial trade off between the accuracy of the model and the real-time performance. Complex lighting models involve the interaction of light within a material. The scattering of light needs to be considered to obtain a realistic final image. Scattering is a diffuse effect where light rays deviate from their expected path due to the effect of the medium in which they pass.

Scattering is considered as both a surface reflection effect and also as a sub-surface effect. Sub-surface scattering involves light rays penetrating the surface of the material, being scattered in multiple directions. Because the light is scattering it will exit the surface in multiple distinct locations. If the area is thin enough, some of the light can exit the material on the other side from which it entered contributing to the phenomena of translucency. Subsurface scattering is a common phenomenon, present to some extent in all non metallic materials.

Human skin is a particularly complex material to accurately model. There are two main reasons for this. First humans are conditioned to see human faces and have therefore evolved to notice flaws in computer graphics renderings. Secondly skin is composed of three primary layers, the epidermis, the dermis and the hypo-dermis. These layers can be further sub-divided into additional unique layers. Each of these layers contributes unique subsurface scattering properties that independently affect the look of skin.

Modeling of subsurface scattering is defined using the bidirectional surface scatter-

ing reflectance distribution function (BSSRDF), a mathematical construct to describe the relationship between light hitting a surface and light leaving that surface. The BSSRDF is a complex function that currently cannot be calculated in real-time. Various offline solutions exist within the domain of ray tracing for solving the BSSRDF.

1.1 Motivation

Only recently have techniques been introduced to allow a simulation of subsurface scattering in real-time. To achieve interactive frame rates, these techniques use various approximations of the complex calculations performed in the offline algorithms. Perhaps the most impressive is the work of d'Eon *et al.* [10] who independently considered algorithms for the simulation of scattering and the simulation of translucency in skin.

d'Eon uses an extension of translucent shadow maps to estimate scattering through thin regions. This provides a physically accurate estimation of the subsurface scattering lighting effect. While it runs well in isolation, it is computationally expensive when running within the scope of a real-time video game rendering engine. The use of translucent shadow maps also increases the memory requirements of the simulation, particularly as current generation graphics hardware on PCs and games consoles are quite constrained in terms of memory availability.

We investigate non-physically based heuristic approximations to simulate translucency in thin regions of skin. To avoid the overhead of shadow maps we attempt to work within screen-space rather than light space, irrespective of the number of light sources. Working in screen-space removes the restriction of the number of light sources we use, but it means that our algorithms cannot be physically accurate for all situations. These situations occur where the light source is not directly behind the object in relation to the camera. We investigate a number of techniques within the domain of screen-space to produce a perceptually believable image using the output of d'Eon *et al.*'s and ray traced images to compare our output with.

1.2 Goals

The primary goal of this dissertation is to implement an application and algorithm that can compute and render translucency in skin, using calculations made in the domain of screen space.

The secondary goal is to demonstrate that the result of the screen space implementation compares well with the physical based approach of d'Eon. Working within screen-space presents a set of challenges that need to be overcome. The algorithms can not match the fidelity of the physical algorithm in all situations. In these situations, solutions need to be found and evaluated.

A tertiary goal is to ensure that the implementation performs well and the overhead of running it is not prohibitive in the context of the general subsurface scattering algorithm. If this can be achieved, algorithm optimisations are investigated to extract the best performance, as measured by frame rate.

1.3 Dissertation Outline

This dissertation is laid out as follows:

Chapter 2 : Background and Related Work describes the problem domain specifically related to translucency in participating media and an overview of the main techniques that can be used to model this phenomenon.

Chapter 3 : Design presents a high level view of the design and requirements of the system and the implementation plan.

Chapter 4 : Implementation describes the basic implementation, the application framework to host the algorithm as well provides user manipulation functions and test scenarios. It also describes optimisations for the algorithm.

Chapter 5 : Evaluation describes the output of the algorithm with reference to three specific areas, perceptual evaluation, performance and comparisons with previous work.

Chapter 6 : Conclusions summarises the work and main results in the thesis and presents ideas for future work to improve the algorithm.

Chapter 2

Background and Related Work

Human skin as a material to be modeled by computer rendering systems is a complex surface to render in a perceptually believable manner. This is because skin exhibits a property known as subsurface scattering. Subsurface scattering is a loose description that describes light entering a material and randomly altering direction within the material, eventually exiting at a different surface point in a direction not necessarily related to the entry direction.

Subsurface scattering is present in all materials that exhibit translucency, such as milk, marble, jade and skin. Classification of these materials can be divided in two groups, homogeneous materials, those materials which exhibit uniform scattering and heterogeneous materials, those which exhibit non-uniform scattering. In the case of human skin it falls into the latter category, which increases the complexity of accurately modeling it. We can observe this phenomenon by taking a torchlight or similar focused light source close to our hand and observing the light diffusing through the skin. A glow can be observed through thin regions that do not contain any occluding regions such as bone.

Without addressing the subsurface scattering component in appropriate materials, renderings will fail to capture the softness and translucency that these materials exhibit.

This chapter describes a history of the major advancements in subsurface scattering rendering techniques, most of which have arrived in the last ten years. We divide this chapter into offline and real-time techniques. While this dissertation focuses on simulating translucency in real-time, physically accurate offline techniques for subsur-

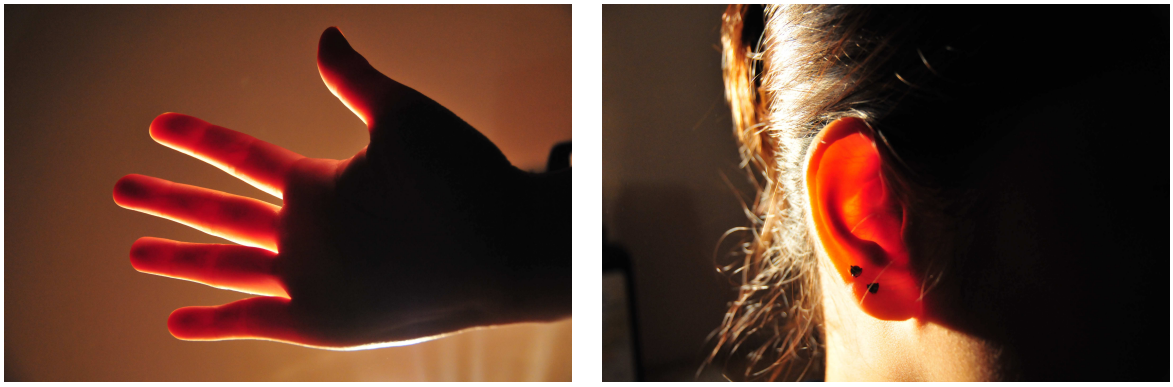

Figure 2.1: Photographs of subsurface scattering in a human hand (left) and a human ear (right). Both photographs are back lit with a halogen light source.

face scattering have been used in the film industry since the turn of the century. It is important to have an appreciation of these techniques before addressing the real-time algorithms.

2.1 The BRDF

Before subsurface scattering was simulated in computer graphics, a standard bidirectional reflectance distribution function (BRDF) was used to model light interaction with surfaces. The BRDF is a mathematical construct describing the ratio of reflectance along the output vector with respect to the irradiance of the incoming light vector. Typical examples of BRDF's that are used currently are the Blinn-Phong [2] specular model, the Gouraud diffuse model [17], and the Oren-Nayar diffuse model [30].

The BRDF treats assume that light in reflected from the surface of a material at the point at which it is struck. This restriction in the lighting model results in renderings using any one of these techniques tending to take on a hard course appearance.

As shown in [4] the BRDF function can be described as

$$BRDF_{\lambda}(\theta_i, \phi_i, \theta_o, \phi_o, u, v) \tag{2.1}$$

where λ is the wavelength of the light, θ_i and ϕ_i represent the incoming direction of light in spherical coordinates, similarly θ_o and ϕ_o represent the outgoing direction, with u and v representing the positional variance. Typically when using spherical

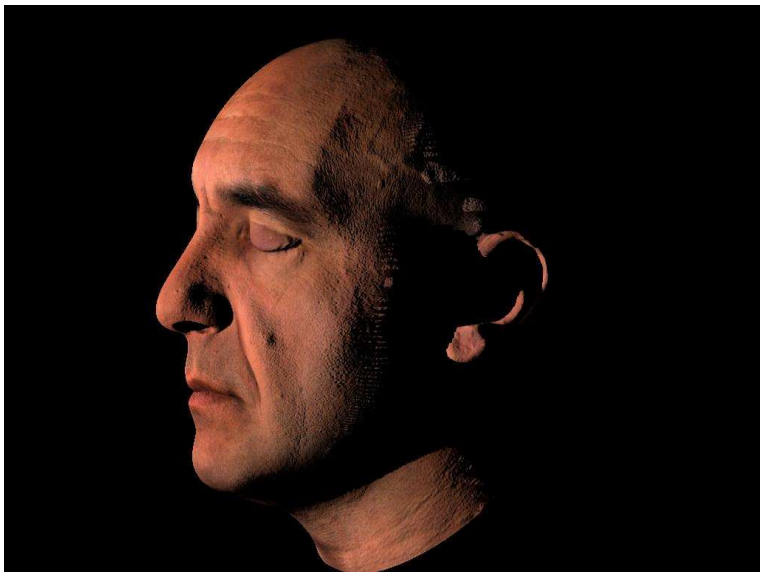

Figure 2.2: Render of a human head model with a diffuse BRDF (Lambertian).

coordinates a third component r is needed, representing distance from the origin of the point. This is ignored for this formulation. Burnaby [4] shows how this continuous form can be reduced to a discrete form using differential solid angles.

$$L_o = \sum_{in} BRDF(\theta_i, \phi_i, \theta_o, \phi_o) L_i cos\theta_i \tag{2.2}$$

2.2 The BSSRDF

The radiative transport equation [6] describes the light propagation within a material taking into account the scattering and absorption properties of the material. Mathematically it can be described as

$$(\vec{\omega}.\nabla)L(x,\vec{\omega}) = -\sigma_t L(x,\vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega},\vec{\omega}')L(x,\vec{\omega}')d\omega + Q(x,\vec{\omega}') \tag{2.3}$$

where σ_a is the absorption coefficient, σ_s is the scattering coefficient and $p(\vec{\omega},\vec{\omega}')$ is the phase function or the angular distribution of the light reflected from the surface based on the illumination direction.

This equation fully describes the propagation of light. However computational solutions are known only for simple case situations. One such solution is the bidirectional

surface scattering reflectance distribution function (BSSRDF).

The BSSRDF was introduced by Nicodemus [29] to describe the relationship between the incidence flux and outgoing radiance at a point on a surface primarily taking into account subsurface scattering within the material. The BSSRDF is a generalised form of the BRDF previously described in that the point of light incidence and radiance can be different. Mathematically it is described as

$$dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) d\Phi(x_i, \vec{\omega}_i) \tag{2.4}$$

where L_o is the outgoing radiance in the direction $\vec{\omega}_o$ at the point x_o and $\Phi_i(x_i, \vec{\omega}_i)$ is the incidence flux in the direction x_i at the point x_i. Resolving the outgoing radiance involves solving this using a double integral over both surface area A and incoming directions 2π resulting in

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) L_i(x, \vec{\omega})(\vec{n}.\vec{\omega}_i) d\omega_i dA(x_i) \tag{2.5}$$

It should also be noted that the BRDF is a special case of the BSSRDF, where the light enters and exits a surface at the same point

2.3 Challenges of rendering skin

In the real world very few materials behave according to a BRDF model. Most materials exhibit a degree of translucency such that some of the light hitting it is reflected, some is absorbed and some is refracted internally. It is these refractive photons that can be most critical to image quality.

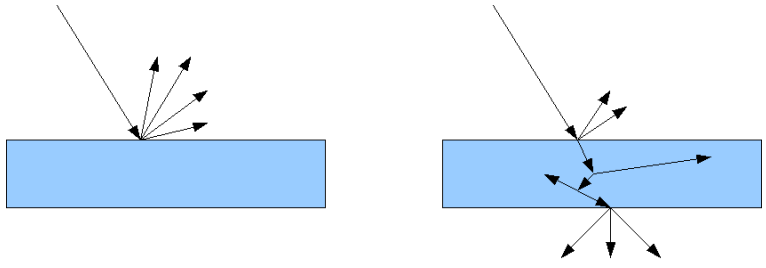

Figure 2.3: Reflection model of an opaque material (left) versus a transparent material (right).

To make matters more difficult, skin is not just a single layered material. It is composed of three major distinct layers: the epidermis, the dermis and the subdermis [18]. Each of these layers exhibits different scattering properties. To model this accurately each of these layers should be modeled independently. This would be considered a minimum model for accurate skin rendering.

Each layer consists of further sub-layers. The epidermis for instance can be classified into five further layers, the stratum basale (basal cell layer), the stratum spinosum (prickle cell layer), the stratum granulosum (granularcell layer), the stratum lucidum (clear layer) and the stratum corneum (horny cell layer) [18]. Each of these sub-layers has distinct and individual scattering properties.

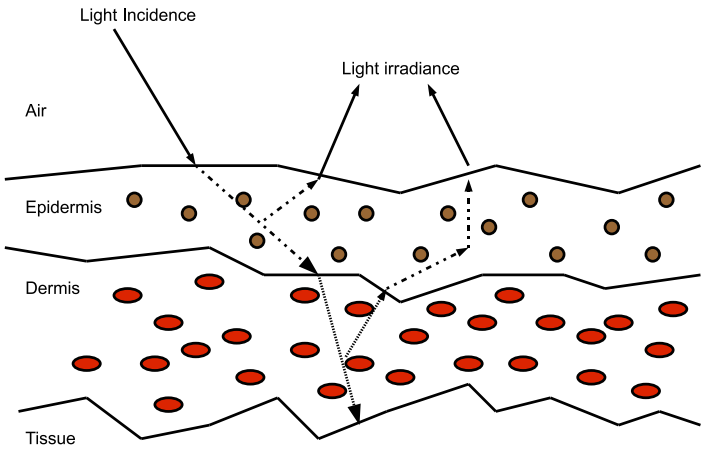

Figure 2.4: A simple two layer model of skin.

Given this, it is clear that the lighting model is a very complex process to calculate a result for. Indeed it which is not feasible, given the limitations of current available hardware.

2.4 Offline Techniques for Rendering the BSSRDF

In this section common methods for approximating the BSSRDF are introduced. Generally the goal of each presented algorithm has been to improve rendering performance over fidelity. Even though these are offline algorithms they are still constrained by

the amount of time required to generate the final image to have a useful application. Starting chronologically with photon mapping, the major innovations are introduced below.

2.4.1 Photon Mapping

Photon mapping was introduced as an extension to Monte Carlo ray tracing as a two-pass global illumination algorithm by Jensen [22] as a solution to the general rendering equation proposed by Kajiya [25]. While initially designed to address rendering issues such as soft shadows, caustic effects and colour bleeding, it can also be used for sub-surface scattering. Unlike Monte-Carlo ray tracing it dissociates the scene geometry from the global illumination solution. The first pass in this algorithm is called photon tracing, which produces a photon map as an output. Photons originate from a light source within the scene and their direction is described by the type of light source being used. The intensity of the light source reflects the quantity of photons to be emitted per time unit.

The emitted photons are then traced from the light source until they hit a surface in the scene. At this point, the algorithm decides, based on the surface material definition whether the photon should be reflected, refracted, or absorbed. These details are then stored (typically in a kd-tree). In the case where the photon is not absorbed, new photons from this point are generated and recursively solved until terminating conditions are met.

Once this first pass completes and a photo map is computed, the second rendering pass is performed. In this pass the scene is rendered using standard ray tracing methods. On a collision between the ray and a surface at a point P, the photon map is used to lookup nearby photons in a spherical neighborhood to point P. The radiance as calculated for a given photon is described in [36] as

$$I_p \times (d.N) \times diffusefactor \tag{2.6}$$

where I_p is the photon intensity and $(d.N)$ is the dot product of the photon incidence direction d and the normal at P. All photons in the spherical neighbourhood of P are summed and averaged. This value is the radiance value at P as calculated by the photon map. It is then added to the radiance value calculated by ray tracing to give a

final radiance value at P.

While not initially developed with a subsurface scattering solution in mind, it was extended to address this issue by Jensen *et al.* [20] and Dorsey *et al.* [13]. Jensen *et al.* introduced the concept of a volume photon map to support rendering participating media (objects in which light can pass through). Dorsey used the technique for rendering light scattering in stone. Pharr *et al.* [31] derived generalised scattering equations and showed a Monte Carlo integration solution to these equations.

2.4.2 A Dipole Approximation

Figure 2.5: A comparison of a face rendered using a BRDF versus Jensen *et al.* [21] BSSRDF model. Note the harshness when rendered using a BRDF (left) in comparison with the BSSRDF rendering(right).

The work of Jensen *et al.* [21] is considered seminal in the area of subsurface scattering and presents a simple yet effective model for simulation of light transport in translucent materials. In this paper scattering is described as the sum of a single scattering term and a diffuse approximation term (simulating the multiple scattering effect within the material). The diffuse approximation is based on an observation that light distribution in highly scattering materials becomes isotropic (uniform). The perceived effect of this observation is that the material is seen to become blurred.

Jensen calculated this using a dipole approximation based on previous work by Farrel *et al.* [16] which showed this to be as accurate as using the diffusion approximation.

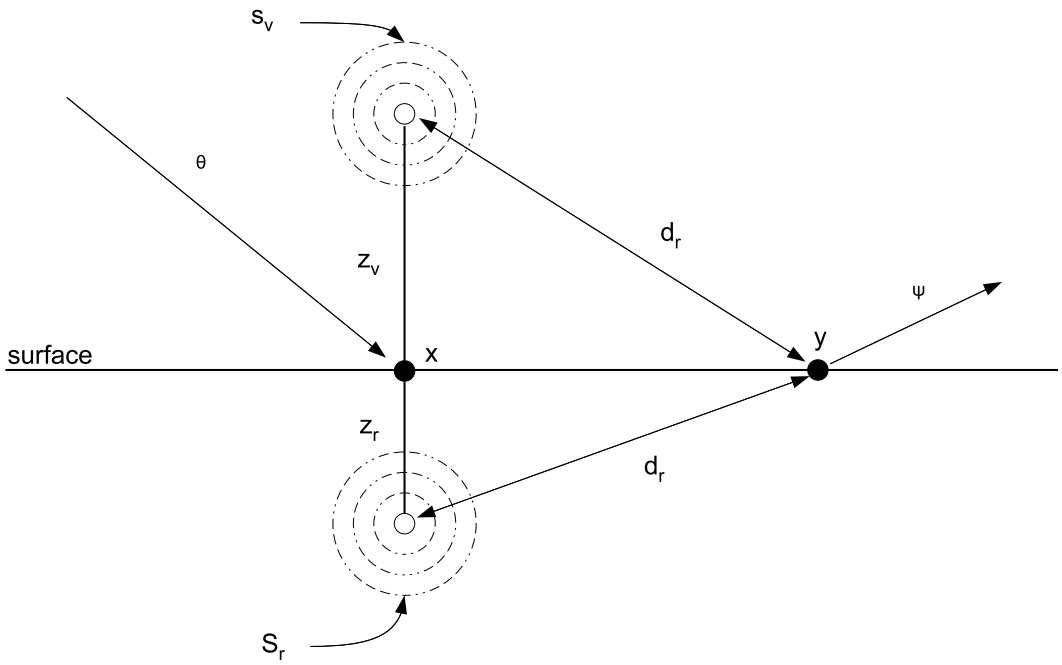

Figure 2.6: Dipole source approximation implementation. A pair of imaginary point sources s_r and s_v are placed above and below the the surface point x. The distances from the surface z_v and z_r at which the dipoles are placed are calculated from $\sigma\prime_s$ the reduced scattering coefficient and σ_a the absorption coefficient of the material (see [21] and [14] for more information).

Jensen also explains why skin cannot be rendered using a standard BRDF model. This is due to the fact that skin is both a highly scattering material and a very anisotropic material, resulting in a large number of internal scattering events for a given photon of light interacting with it.

The key observation of Jensen *et al.* was that in highly scattering materials, the light distribution becomes more anisotropic (uniform) and forms a solution which he defines as the diffusion approximation based on a dipole approximation constructed as in Figure 2.6.

Jensen *et al.* [19] later developed a new technique based on hierarchical sampling to approximate the BSSRDF implementation previously developed which would work with both ray tracers and scan-line renderers. While the BSSRDF solution provided

by Jensen *et al.* [21] was orders of magnitude faster than equivalent photon mapping approximations it is still computationally slow for highly scattering materials such as skin. This is due to the algorithm having to sample the incidence flux over a given surface area. As a material is defined to be more translucent, the computation of the BSSRDF term using a diffusion approximation requires additional neighbourhood sampling. For highly translucent materials such as skin this becomes very costly. It also only considers scattering from direct illumination and cannot easily handle indirect illumination.

Hierarchical sampling

Jensen and Buhler [19] developed a speedup technique to reduce the computation time of calculating the diffuse approximation using the dipole solution. This hierarchical sampling algorithm is a two pass algorithm. In the first pass the irradiance at predefined sample locations in the model mesh is generated.

In the second pass, using the sample data, the diffusion approximation could be inefficiently calculated by summing the contribution of all the irradiance samples for a given point and applying the formula described in [19] to calculate the radiance emitted at a specific point. This is a time consuming calculation, so a method of hierarchically storing the irradiance samples in an octree where each node contains a representation of the illumination contributed by all its children as well as the surface area consumed by all children is used. The tree is traversed from the root down until a terminating condition is met. Jensen uses the solid angle of the sample points contained within the voxel as the terminating condition.

As can be seen this algorithm produces a rendered image with almost indistinguishable fidelity to the dipole BSSRDF rendering. Jensen and Buhler reported a speedup factor of 153 based with this technique. They reported that based on the hardware at the time it still took seven seconds to render the sample image. Even using the hierarchical technique, real-time rendering speeds were still not possible.

2.4.3 Texture Space Diffusion

One of the most interesting techniques in relation to real-time rendering is the approach taken in the Matrix Reloaded film. Texture space diffusion was introduced by

Figure 2.7: A comparison of a translucent Utah teapot rendering using the 2001 BSSRDF algorithm (left) versus the hierarchical BSSRDF algorithm (right). The hierarchical rendering is practically indistinguishable from the previous BSSRDF rendering.

Borshukov and Lewis [3] as a technique for rendering faces of the main characters for computer generated scenes. This is a novel approach which can be considered as the forbearer to the real-time subsurface scattering techniques to follow. It renders the diffuse illumination of the human skin to a two dimensional texture, which is then diffused in texture space using a simple paramterised approximation. Relative to previous techniques, it is computationally inexpensive. However it has the limitation of not being able to capture translucency.

2.4.4 Multipole Diffusion Approximation

Donner and Jensen [11] presented a multipole technique to simulate scattering in multilayered materials. Prior to this, most work assumed semi-infinite homogeneous materials. Previous work based on a dipole approximation assumed that light photons entering a material are either absorbed or reflected back out of the material. In this paper Donner and Jensen show how multiple substrates with different scattering properties could be layered and rendered in a physically accurate manner. They demonstrated how this could be applied to skin rendering by using it in a three layer skin model, modeling the epidermis, the upper dermis, and the bloody dermis.

Donner and Jensen *et al.* [12] extended the previous multipole rendering of skin by defining a specific two layer multipole model of skin with a simple parameter set to control the behaviour of the model. They used four parameters (total melanin content, melanin type, hemoglobin content, and surface oiliness) directly related to the biology of skin to control the appearance of rendered skin rather than the more typical material

Figure 2.8: A rendering of a human face using the dipole technique (left) versus a three layer multipole model (right).

scattering coefficients. They implemented this using a Monte Carlo ray tracer.

2.5 Rendering the BSSRDF in Real-time

All the techniques presented thus far are computationally too complex to run in real-time at 30 frames per second. Recently techniques have been developed that provide accurate approximations for the offline algorithms. These techniques have rarely been implemented by the video game industry, primarily due to the fact they are are computationally expensive relative to many other effects that game engines must render from frame to frame. The most important real-time techniques are presented below.

2.5.1 Translucent Shadow Maps

A translucent shadow map (TSM) [9] is an extension of classical shadow mapping. It extends the concept of shadow mapping to store more information than the depth value, allowing an approximation of translucency to be calculated.

A traditional shadow map is a representation of the scene as viewed from the perspective of the light source. Rather than storing the colour value of the pixel, it stores a depth value representing the distance from the light source to that pixel. When rendering the scene, the position of each pixel in the scene is converted to light space and its distance from the light source is compared with the corresponding value contained in the shadow map. If this value is greater than that in the shadow map, it is presumed that there is a surface occluding this pixel and therefore the pixel must be in shadow.

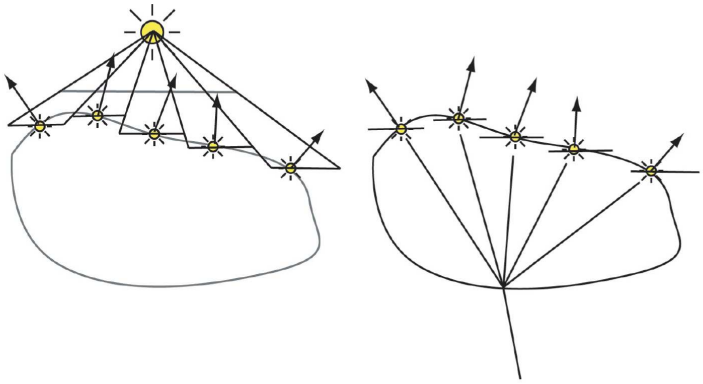

Figure 2.9: Calculating the transmitted light using a translucent shadow map. Irradiance samples are calculated and stored in the first pass (left). In a second pass, calculation of the outgoing radiance (right) at a surface point consists of sampling and combining the irradiance values on the lit side of the surface.

A TSM extends this concept by storing an irradiance value and a surface normal with each pixel as well as the depth value. An initial pass is executed to calculate and store these values. The irradiance impulse as stored in the map is given by

$$E(x_{in}) = F_t(n, w_{in})|N(x_{in}).w_{in}|I(w_{in})[9] \tag{2.7}$$

where n is the optical density of the material, $I(w_{in})$ is the input impulse, $N(x_{in})$

is the normal at the incidence point and F_t is the Fresnel term, which is approximated using a method proposed by Schlick [34]. In the second pass these values are used to calculate the transmitted radiance $B(x_{out})$, where x_{out} is the shadowed point to be rendered, according to. This is shown in Figure 2.9.

$$B(x_{out}) = \int_S E(x_{in}) R_d(x_{in}, x_{out}) dx_{in} \tag{2.8}$$

To solve the transmitted radiance, R_d needs to be calculated across the incidence surface. The solution to this integral is approximated by a filtering algorithm with predefined mip-mapped textures allowing real-time calculations to be performed. One disadvantage of this technique is that it fails for non-concave surfaces as the calculations are based on the depth stored within the shadow map which is assumed to be solid from the light facing surface to the shadowed surface.

2.5.2 A GPU Algorithm for Subsurface Scattering Simulation

Carr *et al.* [5] described a three pass GPU algorithm for approximating subsurface scattering which is an extension of earlier work done by Lensch *et al.* [26]. Lensch *et al.* noted a strong correlation between the radiosity formula and the earlier BSSRDF solution presented by Jensen *et al.* [21]. Carr *et al.* exploits this fact and uses the solution for the radiosity formula to approximate the solution to the BSSRDF.

This is done by taking a non deformable model and discretising it into a set of patches using a mip-mapped texture atlas, and applying their GPU algorithm to this patch set. The first pass calculates the illumination on each patch scaled by the Fresnel term, forming a radiosity map that is stored as a texture.

In the second pass, the radiosity map is converted into a scattered irradiance map (using previously precomputed values). To do this the algorithm solves the following equation which is a reformulation of the BSSRDF formula presented previously by Jensen *et al.*.

$$B_i = \sum_{j=1}^{n} F_{ij} E_j \tag{2.9}$$

where F_{ij} is the multiple diffuse throughput factor calculated as

$$F_{ij} = \frac{1}{A} \int_{x_i \in P_i} \int_{x_j \in P_j} R_d(x_j, x_i) dx_j dx_i \tag{2.10}$$

and $R_d(x_j, x_i$ is the diffuse subsurface scattering reflectance, A is the surface, and P_i and P_j are patches on the surface.

Finally the third pass renders the scene by adding the scattered irradiance map texture to the surface of the material before applying the lighting calculations.

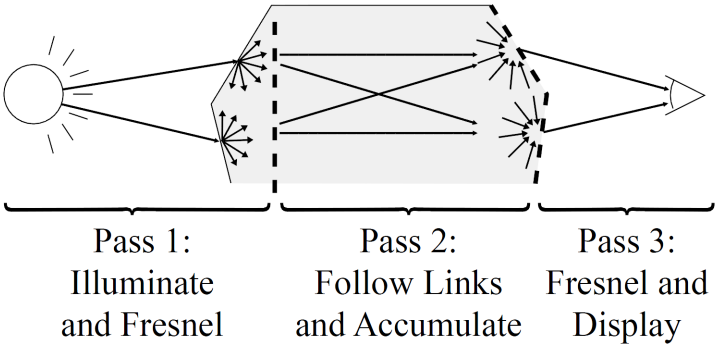

Figure 2.10: Carr *et als'* three pass method for simulating subsurface scattering on the GPU [5]

2.5.3 Gaussian Approximation of the Multipole

d'Eon *et al.* [10] describe an approach allowing real-time interactive frame rates while producing similar image fidelity as the offline mulitpole techniques previously described. Their approach implements the multipole approximation presented by Jensen *et al.* [21] using a linear combination of Gaussian basis functions as an approximation of the diffusion profile on each homogeneous substrate. Figure 2.11 shows that a four Gaussian sum can accurately approximate the dipole.

Calculating the sum of of Gaussians is extremely efficient primarily as they can be applied as two one-dimensional convolutions (blurs) in x and y. This calculation is fast enough to run at real-time frame rates. It is also the first real-time algorithm that requires no pre-computation step. This allows for animation and deformation of the scene geometry, something that was not possible with previous techniques which

required a pre-computation step. The implementation was done in texture space using the premise of texture space diffusion [3] to simulate the subsurface scattering and an extension of TSMs [9] to handle light transmittance in thin regions.

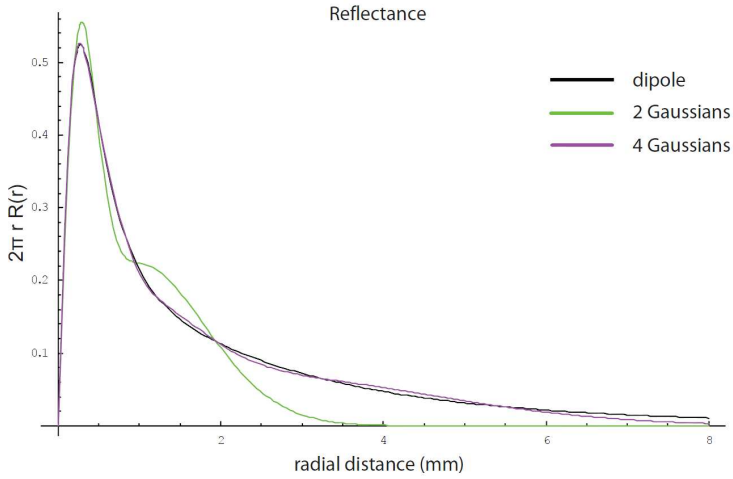

Figure 2.11: Courtesy of [10]. The dipole can be accurately be approximated by using the sum of four Gaussian terms. This resulting in indistinguishable rendering output to the dipole approximation. Two Gaussian terms generally cannot approximate the dipole accurately.

2.5.4 Simulating Translucency

While using a Gaussian approximation of the multipole will simulate the scattering effect, it will not simulate translucency. In order to introduce this to the simulation, d'Eon *et al.* combines the Gaussian approximation with a modified version of the TSM.

Modified TSMs were introduced to supplement texture space diffusion for thin regions where some light should pass through completely. Texture space diffusion alone can not handle these situations where light transmits though thin regions, such as the ears and the nose. This is because while the surface points are close in world space, their corresponding texture coordinates are further apart in texture space. When the diffusion approximation is applied in texture space for thin regions, it will not be able to take into account the nearby irradiance values in world space. To address these issues modified TSMs are used to supplement the algorithm. These differ from standard

translucent shadow maps by storing the depth value (z) and the (u, v) coordinates of the light facing surface. Recall that a standard translucent shadow map will store the depth values, the irradiance values, and the surface normal values. Using the texture coordinates (u, v), the algorithm will have access to the convolved (blurred) irradiance texture on the non-shadowed side as well as the thickness of the surface through which the light passes.

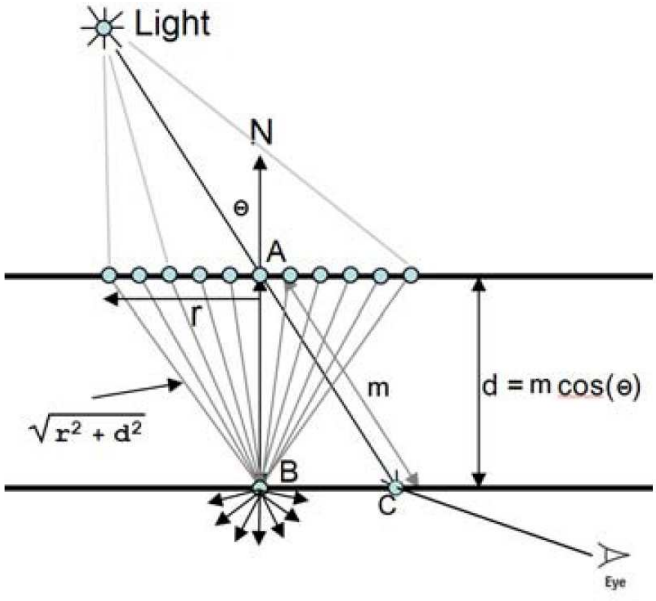

Figure 2.12: Calculation for the modified translucent shadow map as presented by d'Eon [10]

To calculate the transmitted scattered light at point C which is shadowed, shown in figure 2.12, the modified TSM us used to retrieve the depth m and the texture coordinates at point A on the lit side. C is estimated by taking weighted samples about A. In order to speed up computation and to simplify the code, it is preferable to sample at B. This is possible because when θ is small, C and B are close and for large values of θ the Fresnel term as as well as the dot product between the normal and light direction will reduce the amount of transmittance to C. d'Eon *et al.* defines the radiance R at the shadowed surface as

$$R(\sqrt{r^2 + d^2}) = \sum_{i=1}^{k} w_i G(v_i, \sqrt{r^2 + d^2}) = \sum_{i=1}^{k} w_i e^{-d^2/v_i} G(v_i, r) \tag{2.11}$$

where d is the thickness of the surface, corrected by the cosine value, G is the convolved Gaussian value on the non-shadowed side and $w_i e^{-d^2/v_i}$ is a weighting term based on the distance from A.

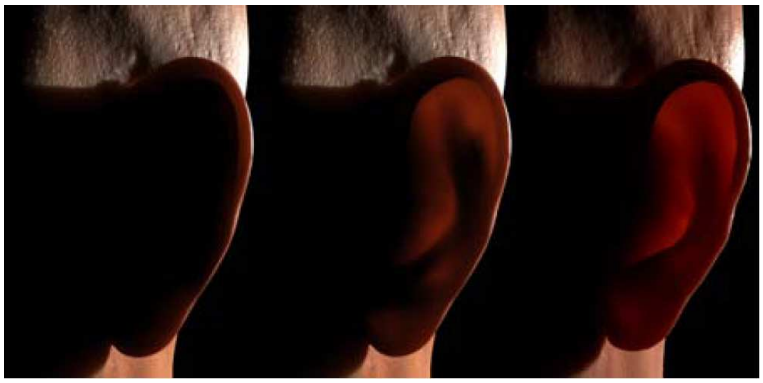

Figure 2.13: Rendering without considering translucency does not capture scattering in thin parts of the ear (left). d'Eon's modified translucent shadow map implementation (center) generates results which match the equivalent Monte Carlo rendering (right) well while running at interactive frame rates.

2.5.5 Screen space diffusion

In d'Eon's work, calculated irradiance is unwrapped to texture space before the blur is applied. The blurring performed in texture space is then wrapped back onto the model to produce the rendered approximation of subsurface scattering. Jimenez *et al.* [24] introduced a skin shader that 'translates the simulation of subsurface scattering from texture space to a screen-space diffusion approximation'. Texture space diffusion has some limitations that are addressed in screen space. These include a loss of GPU optimisations (backface culling and viewport clipping), the requirement to maintain an irradiance map per model and duplication of screen pixels in texture space amongst others.

Jimenez *et al.* demonstrate that the quality loss of working in screen space is not perceived by the human visual system. It shows that even with just a single head model being rendered, there is a performance increase over texture space diffusion. The performance decrease from rendering five heads using the screen space implementation is 13% compared to 65% using the texture space algorithm.

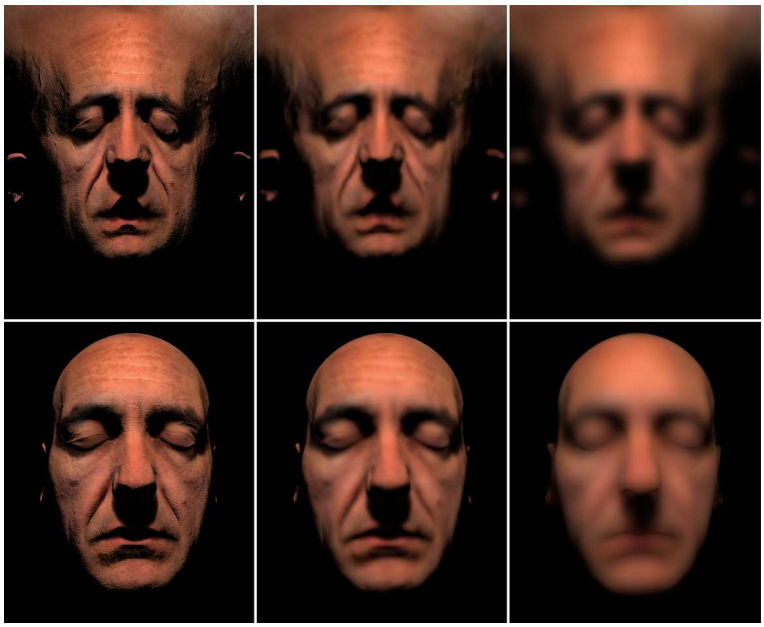

Figure 2.14: Comparison of d'Eon texture space implementation (top) compared with Jimenez screen space implementation (bottom). Courtesy of [24].

2.6 Subsurface Scattering in Entertainment

Recent advances in subsurface scattering algorithms has meant that the film and video game industry have been able to make use of them. The effect is more important for the film industry as the rendered images can coexist with real human actors. In the games industry real-time implementations are still in their infancy.

2.6.1 Film

In the last decade, subsurface scattering algorithms have been developed for use as CGI effects in the film industry. One of the earliest examples of this was the film "The Hulk" [33] which used subsurface scattering effects to render the Hulk's green skin. Pixar initially implemented subsurface scattering within its Renderman [32] engine. It has been successfully used in Finding Nemo [7] to render characters skin and rendering the coral reefs. Pixar then implemented a full skin scattering algorithm for The Incredibles and further extended it for Ratatouille by giving it non physically based parameters to allow artists more control of the output. The Lord of the Rings Trilogy used subsurface

scattering techniques when rendering the synthetic character Gollum.

2.6.2 Video Games

Up until very recently no video game engine provided for subsurface scattering effects. Earlier attempts modeled skin as a material similar to other materials in the games, resulting in a plastic look. The skin rendering work done by Borshukov [3] in rendering skin for the Matrix reloaded film was adapted by both ATI and Nvidia as the premise for real-time demonstrations. Beeson [1] faked subsurface scattering by modeling it as simple formulas based on the surface normal and the lighting or viewing vectors. In a demonstration by ATI [27] a Precomputed Radiance Transfer (PRT) function is used to simplify the BSSRDF portion of the rendering equation.

Recently the Crytek's CryEngine 2 [28] introduced subsurface scattering which is used to model transparency in ice and and other translucent materials within the game Crysis. It was also used in the rendering of human skin in a demonstration of the engine capabilities.

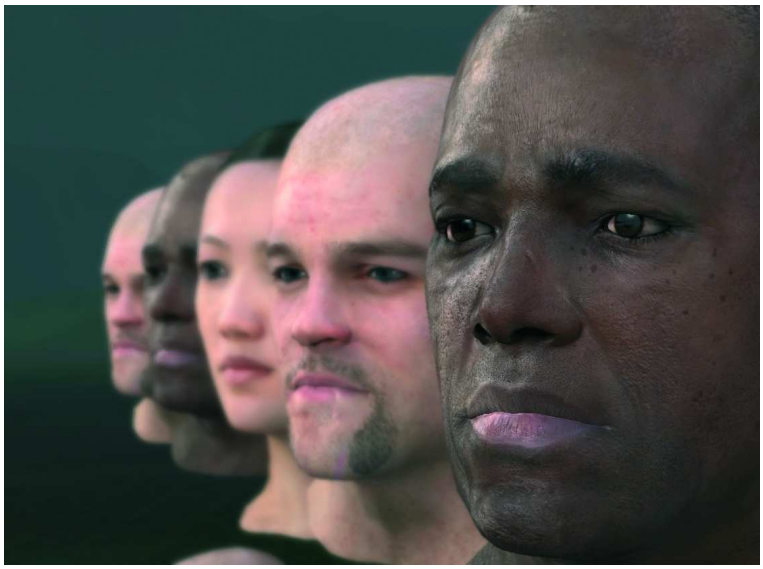

Figure 2.15: A sample of subsurface rendering using CryEngine2.

2.7 Summary

This chapter has given an overview of subsurface scattering and the major breakthroughs that have been made culminating in algorithms that can simulate subsurface scattering in real-time. Because of the complexity of approximating the BSSRDF, only recently have real-time demonstrations emerged. However this area is a growing area of research and as the video game industry strives for more realism, it is anticipated that more games will introduce subsurface scattering effects.

Chapter 3

Design

This chapter describes the design of the algorithm and proposed evaluative tests.

3.1 Plan

The plan for this dissertation was broken into three distinct areas as follows:

1. Investigate: Generating translucency without referring to the direction of light source is a new technique that has not been previously implemented. General investigation was required to determine the feasibility and drawbacks of this approach and to identify impediments to the process.

2. Build: The build phase was broken into an initial framework phase. Building the framework involved implementing the various components of the rendering application. The required components were identified as:

   (a) Camera and camera control system.

   (b) Lights and lighting control system.

   (c) Menu system allowing dynamic modification of shader parameter.

   (d) Debug viewer allowing intermediate steps in the rendering pipeline to be visualised.

   (e) XML based scene descriptions and the ability to save and load them.

   (f) Test framework to conduct the perception test through.

(g) Benchmarking framework to capture performance data.

3. Validate: Create tests to validate the fidelity and performance of the rendered image.

3.2 Requirements

1. The primary high level requirement of the screen-space forward scattering algorithm is to produce a rendered skin model with image fidelity that could be considered similar to the image fidelity generated by the physically based rendering algorithms. Previous work by Jimenez *et al.* [24] showed that diffusion can be performed directly in screen space, rather than unwrapping the model to a texture and performing it in texture space, whilst maintaining a perceptually plausible result. By working directly in screen space Jimenez *et al.* achieved a significant performance increase over the work of d'Eon *et al.* [10]. A similar approach in take for the calculation of transparency.

2. The second requirement is for the algorithm to perform acceptably. Jimenez and Gutierrez [23] report that the simulation of translucency as implemented by d'Eon *et al.* can consume 30% of the algorithm run time using a single light source. Adding additional light sources would further degredate the performance of the algorithm. We aim to show that our algorithm performs better than this.

3. The third requirement is to determine a set of algorithm parameters that best fits the head model and demonstrates the flexibility of the algorithm to work with other models.

3.3 Methodology

The algorithm was built using the following combination of programming tools:

- C#
- XNA 3.0
- HLSL [15]

- FBX model file
- Visual Studio 9 (2008) development environment.

A three dimensional human head model was provided in the `.fbx` 3D modeling format for use in this dissertation. It is a high resolution image of an actor's (Doug Jones) head scanned and provided by XYZRGB Inc. [35]. We received three versions of this model each with a different polygon resolution. This model is provided solely for academic use.

For demonstration purposes and to extract images for the perception test, the ability to manipulate the model, camera and light sources as well as the parameters of the forward scattering algorithm. This required a menu / parameter manipulation system to be added to the application and the ability to save and load scene data from external files.

The development cycle that was used is loosely termed as "'iterative experimental"'. It begins with a simple premise, that forward scattering can be simulated in screen-space as a reasonable approximation of the effect generated by using modified translucent shadow maps. The algorithm is then refined over several iterations.

3.4 Validation

Validation of the algorithm is performed using two methods. Firstly comparisons between still images captured for the application and sample images from other algorithms and photographs. Secondly a perceptual validation of the transmitted colour is performed with to determine the amount of colour transmittance required for each of the colour channels.

3.4.1 Perceptual Validation

A user study was conducted to determine the colour transmittance values to be applied to our test model. Based on both research in computer graphics and in the medical imaging domain, it has been shown that different wavelengths of light will exhibit different depths of transmittance before being absorbed or scattered back through the same surface as they entered. The study will determine from a perception perspective the

relative relationships between the primary colour frequencies (red, green, blue). This data can be compared with the data from other research to determine if a correlation exists and is used to set the algorithm colour transmittance parameters.

Participants

Twelve volunteering participants (nine male, three female) took part in the experiment, all being unaware of the purpose of the experiment. Some subjects had experience with computer graphics and all reported normal or corrected to normal vision. The age of the participants ranged from 23 to 58

Stimuli

The test consisted of six test stimuli (scenes), each differentiated by a different camera and light position. Samples of these can be seen in appendix A. The subjects were asked to evaluate the realism of the scene specifically with reference to the colour pigmentation of the translucent portions of the scene. Typically the scenes were setup so a majority of lighting was from translucent portions of the model. The subjects modified the scene by selecting colours using a colour selection widget and would see a real-time update of their changes. Once satisfied with the image, they press the space bar and the algorithm settings are recorded to file.

Chapter 4

Implementation

Analysing the work of d'Eon *et al.* [10] it is clear that using a modified TSM is a limiting factor in the scalability of the algorithm. Generation of the TSM requires an overhead in terms of the memory requirements of each TSM and the computational cost of building the TSM at each simulation frame. Introducing additional light sources to the scene requires that a corresponding TSM be generated for each light source. The cost of doing this is prohibitive and scenes are typically lit with just a single dynamic shadow casting light source at any one time.

An example of this avoidance of multiple shadow maps in real-time games can be observed in the game Left4dead [8] by Valve, each of the four characters uses a torchlight. From the first person perspective of any character, their torch will cast shadows on the scene. However observing other characters from this perspective shows that these characters are not casting shadows.

4.1 Initial premise

Our initial premise was to calculate a thickness of the model as observed from the perspective or the camera. This is done by rendering two depth maps. One for the front facing and one for the back facing polygons of the model. This is done from the perspective of the camera, rather than the light source as is typically done in shadow mapping and is independent of the number of light sources in the scene.

Combining these two depth maps and subtracting the front and back depth values

Figure 4.1: Comparison of lighting without (left) and with (right) forward scattering. Light source is directly behind object.

of each screen pixel, we calculate a screen space map of the thickness of the model being rendered (as shown in Figure 4.3). For the purpose of this implementation we shall refer to this as the 'thickness map'. This map is physically accurate in translucency calculations where the light source is directly behind the model. However if the light source is offset at a different position, then the calculation will not be accurate and adjustments need to be made to compensate for the inaccuracies. This is addressed later in the implementation.

Calculating translucency is done as a function of the thickness. A function is required to transform the pixel value in the thickness map to a three vector translucency value, which we define to be in the range $< 0, 1 >$ for the three colour channels, where 0 represents an absence of translucency and 1 represents complete translucency.

The simplest empirical method of doing this is to linearly transform the calculated screen space thickness to a transmittance value. This is achieved by defining a maximum thickness, whereby any pixel with a thickness value exceeding this is excluded from the forward scattering calculation given in equation (4.1), where T_λ is the transmittance for a given wavelength λ, d_p is the scene depth at pixel p and $d_{max\lambda}$ is the maximum depth in which transmittance occurs for a given wavelength λ.

$$T_\lambda = 1 - d_p/d_{max\lambda} \tag{4.1}$$

As the calculated value for translucency is only accurate for the situation where

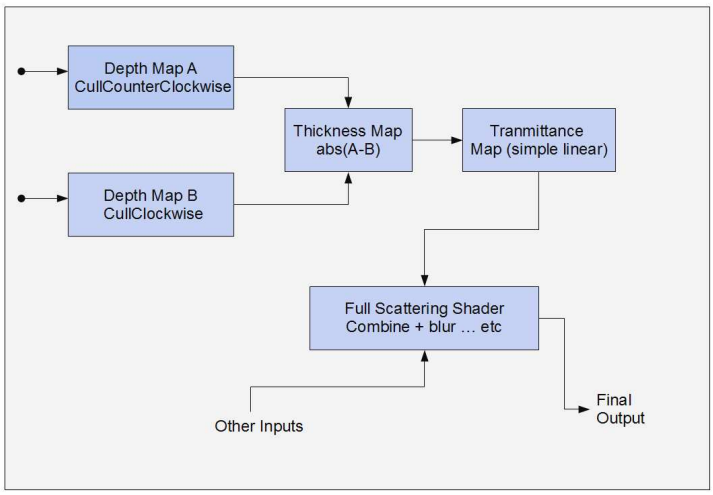

Figure 4.2: The high level application pipeline expressed in terms of the HLSL components.

the light source is directly behind the object, we need to apply a correction to the calculated value compensating for the actual lighting direction. Initially the simplest way to achieve this is to calculate the *dot product* between the camera direction vector and light source direction vector. If the calculated *dot product* value is positive then then both the light and camera are pointing in the same relative direction and the translucency component will be zero. For values between $< 0 >$ and $< -1 >$ we multiply the translucency component by the *dot product* according to the following

$$T_f = 1 - (1 + D_{cl}) \tag{4.2}$$

where T_f is the final transmittance value and D_{cl} is the *dot product* of the camera direction and light source direction. The result of calculating and applying the translucency is shown in Figure 4.3.

The translucency is calculated within a HLSL shader and the application stores the result in a texture via executing the shader against a `RenderTarget`. The final skin shader (responsible for rendering the screen) will accept this texture as a parameter and will sample from it in the fragment shader using screen space coordinates, using the Direct3D 9 shader model VPOS semantic (a shader semantic that returns the current

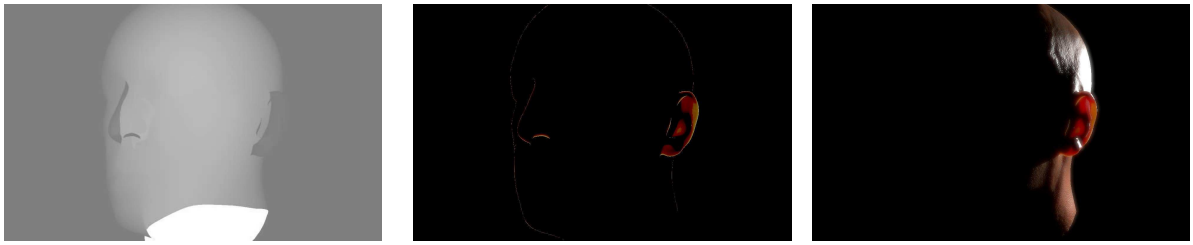

Figure 4.3: A calculated screen space thickness map of the model (left). Darker areas represent thinner regions, lighter areas represent thicker regions. The corresponding calculated transmittance map from the model (center) and the final combined image (right).

pixel's screen space coordinates), to determine the amount of additional lighting to apply in the final lighting calculation.

The final skin shader performs a screen space approximation of the Gaussian blur and we take advantage of this fact by applying the translucency to the scene before the blurring. This gives the benefit of applying the blurring to both the forward and back scattering lighting. Figure 4.3 shows the results of apply forward scattering.

4.2 Experimentation Framework

In order to test and validate the algorithm, a framework was required in which scenarios could be described, manipulated, and tested. The various components of this framework are described below.

4.2.1 Camera Management

A mouse driven first person camera was developed to allow the free positioning of the camera in the scene. It was implemented to follow typical first person gaming conventions, the keys `w, s, a, d` for movement and mouse for free looking. It enables the user to place the camera in various positions in the scene to analyze the rendered image.

4.2.2 Light Management

Similar to the camera, the scene light needs to be controllable by the mouse in order to place it in specific locations in the scene. This is achieved similar to the camera.

4.2.3 Menu System

As the application was developed additional features and parameters were added to the algorithm. Rather than have to manually modify application parameters in code and reload the application, a two layer menu system was developed which allowed the user to dynamically modify application parameters and enable and disable features. This was necessary to allow real-time adjustments of the renderer and immediately evaluate their effect. An example of this can be seen in Figure 4.4. As the application grew, so did the set of adjustable parameters.

Figure 4.4: The Menu system implemented for manipulating translucency parameters. Other menus exist for other parameter manipulation.

4.2.4 Data Capture and Loading

In order to perform perception testing, scene data and application parameters must be described. A scene/parameter saving mechanism was developed which allows the user to capture the current scene and algorithm parameter set and write it to an external file. Similarly the loading mechanism reads the file and sets all application parameters to the loaded values. This allows each rendering for our perception test to be defined and recorded to and external file. XML was chosen as the file format and the inbuilt XML processing API's of C# were leveraged.

4.3 Depth Map Revisions

The first iteration of the algorithm gave mixed results. However there were encouraging signs that the algorithm had merit. The major issue with the initial revision concerned the calculated thickness map. It correctly generated screen space thickness, but could not take into account the direction of the light.

Two major errors existed in the initial rendering.

1. Any thin region with an occluder behind it was not included in the translucency calculations due to the thickness map.

2. Any thin regions that were visible from the camera but contained a large occluder (usually the head) between the light source and the thin region were included in the translucency calculation.

Both these issues resulted in some thin regions being lit inappropriately and other thin regions not being lit appropriately. Figure 4.5 shows one of the situations where this occurs

To solve the first problem an alternative method of generating the depth maps was used. By changing the initial depth values and setting the depth comparison function to `CompareFunction.LessEqual`, the generated thickness map would record the depth of the closest surface to the camera disregarding any surfaces behind that surface. Making this simple change allowed thin regions that previously were not highlighted to receive translucency.

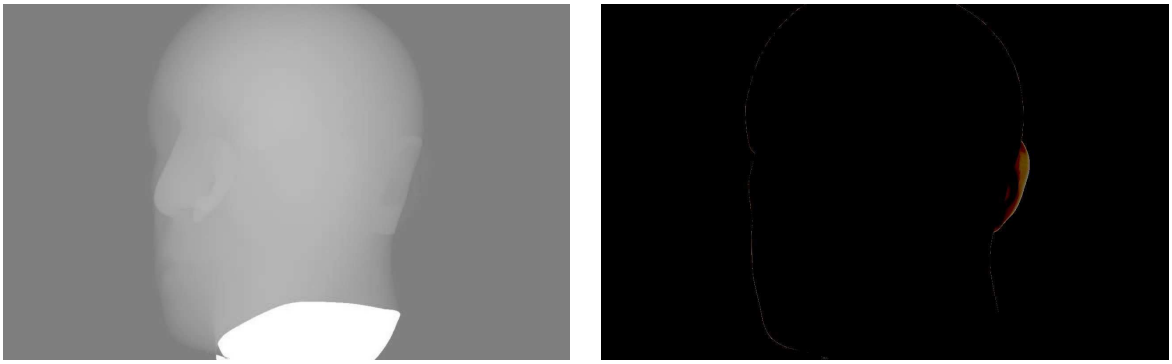

Figure 4.5: Using a simple approach to the depth maps results in thin areas not being lit when there are other surfaces behind them. In the thickness map (left) we can see that the portion of the ear protruding from the head has its thickness correctly calculated, the rest of the ear does not. The transmittance map (right) shows how this translates into an incorrectly rendered image.

The second major issue with the implementation was that it rendered translucency in thin regions that should not receive any light. Since the algorithm works in screen space, it does not have reference to the light perspective to determine if occluders exist. The simplest and most elegant solution to this problem is to make use of the shadow map that already exists. The shadow map stores the distance closest to a light source for a given pixel on screen. In the main skin rendering pass, shown in Figure 4.2, where the forward scattering contribution is added to the standard lighting contribution, the shadow map information is also available.

We can make use of this information and compare the distance from current pixel to the light source to the corresponding value in shadow map. If this is greater than our maximum thickness parameter, it can be concluded that this pixel should not receive translucency lighting. As can be seen in Figure 4.6, the left ear should be obscured by the rest of the head, but as it is a thin region forward scattering is calculated for it. Without using the shadow map it would be difficult to determine if this region should be lit or not. However as can be seen on the right, the forward scattering is not added to the scene for the left ear, resulting in a correctly lit image.

This method has the drawback of relying on the shadow map being available. Other techniques are investigated to determine if it is possible to break the link to the shadow map completely. As will be reported, later no technique was found that could conclu-

Figure 4.6: Rendering without using the shadow map to eliminate pixels (left) versus using the shadow map (right). The light sources is on the right as the camera looks at the scene.

sively match the result from using the shadow map.

4.4 Transparency Intensity Modulation

In the initial implementation described in Section 4.1, the *dot product* between the camera and light direction vector was used to modulate the transparency received by a screen pixel in the transparency pixel shader. Based on observations, this relationship does not describe particularly well the expected transmittance value for a screen pixel. It give too much weight to transmittance for large (close to 90 degrees) angles between the two vectors.

Examining the relationship between both these vectors led to a direct relationship between the *dot product* and the cosine function assuming both vectors are normalised. This is shown in equation 4.3.

$$\begin{aligned} a.b &= |a||b|cos\Theta \\ &= 1 \times 1 \times cos\Theta \\ &= cos\Theta \end{aligned} \tag{4.3}$$

Plotting this relationship in Figure 4.7, it is observed that the *dot product* will become negative once the angle between the two vectors is greater than $\frac{\pi}{2}$ radians (90 degrees). Since the *dot product* in this case is simply the cosine function, the output quickly increases with small angle increments, resulting in a large amount of translucency for small incidence angles. This is undesirable and it would be preferable to limit translucency at smaller angles (90-150 degrees) while maintaining it at higher angles (150-180 degrees).

It should be noted that these discussed values assume a minimum *dot product* of zero before translucency occurs. This should not strictly be the case as this *dot product* is constant for the scene. The camera direction vector is calculated to be at the center of the scene and therefore for pixels not at the center of the scene this is not a correct value to use. To allow experimentation, a new shader variable is introduced noted as `minDotProductValue¿` representing the minimum value of the *dot product* to be considered for a translucency calculation.

All of the modulation functions proposed assume that the allowable *dot product* exists only in the range $< \frac{\pi}{2}, \pi >$. Therefore when this minimum *dot product* value is set to not zero, the allowable range $<$`minDotProductValue`$, -1 >$ must be scaled to the range $< 0, -1 >$.

The first improvement implement was to apply the inverse cosine function to the *dot product*. As shown in Figure 4.7, this turns the non-linear cosine function to a linear function in the range $< \frac{\pi}{2}, \pi >$ (90-180 degrees). Normalising the output of the inverse cosine function gives a linear relationship between the incidence angle and the transmittance amount.

Doing this produced a smoother transition, when panning the camera around the model. On observation it was felt that the transition still occurred too quickly and as such, a better function than the inverse cosine was required. This function should have low transmittance amount at low incidence angles, in a sense simulating the Fresnel effect.

Mathematica was used to identify an appropriate quadratic function to represent the curve. A set of data points were selected to approximate the desired curve. Using the inbuilt Mathematica function `FindFit` on the dataset, a quadratic formula (equation 4.4) was produced. This was transcribed in to HLSL and added to the application.

By moving to a cubic curve (equation 4.5) and fitting to the same data points, we

get a better approximation. We added both curves to the shader to compare if the cubic provides any noticeable improvement over the quadratic. Figure 4.8 shows the generated functions added to the application.

$$0.595752x^2 - 2.27844x + 2.14506 \tag{4.4}$$

$$0.650897x^3 - 3.9017x^2 + 7.75534x - 5.07621 \tag{4.5}$$

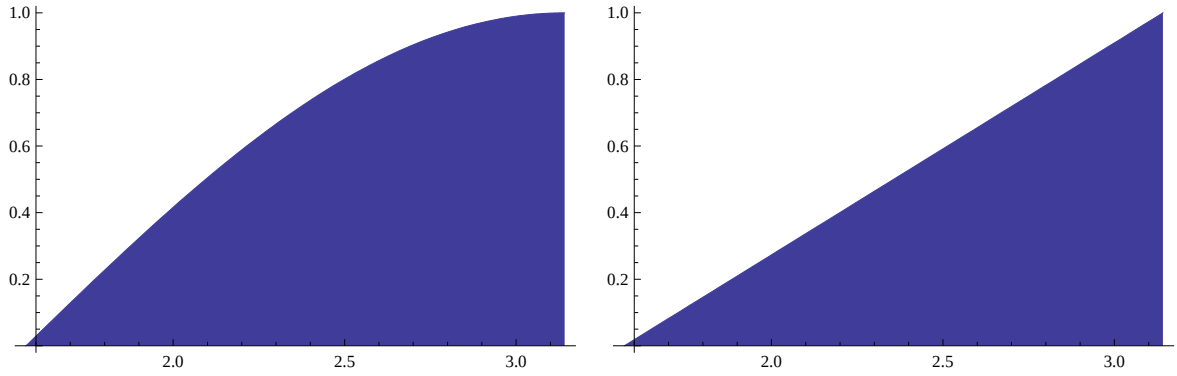

Figure 4.7: Dot product relationship in the range $< \frac{\pi}{2}, \pi >$. We can see that this provides too much transmittance at low incidence angles. (left). Taking the inverse cosing of the *dot product* in the range achieves a linear relationship (right).

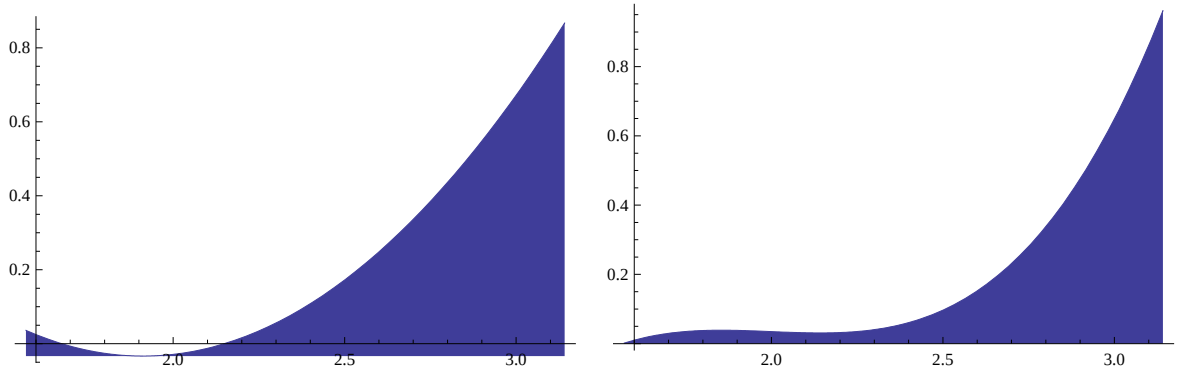

Figure 4.8: Applying a quadratic function shifts the transmittance influence towards straighter incidence angles (left). A cubic provides a more refined curve in comparison to the quadratic (right).

It should be noted that these curves are all derived to give an output in the range

$< 0, -1 >$, assuming an input angle in the range $< \frac{\pi}{2}, \pi >$. Pre-scaling is performed on the input value to the function to ensure that it fits within this range. In the application the pre-scaling function can be enabled or disabled to allow further investigation.

4.5 Modulating Colour Transmittance

Research from both Igarashi *et al.* [18] and Zhao and Fairchild [37] show that different wavelengths of light will show varying transmittance properties. In general red light will transmit deeper into skin relative to blue light with other wavelengths somewhere in between. Our algorithm needs to cater for this to achieve a high image fidelity.

Based on this knowledge, modifications to the algorithm were investigated to determine if the image fidelity could be improved by varying the transmission profile of the three primary colour components (red, green, and blue). Currently the transmittance map as calculated by the shader is a floating point value in the range ¡0, 1¿ and is colour channel independent. As shown experimentally by Zhao and Fairchild [37], the depth to which light transmits before being absorbed and thus the total transmittance of light through thin regions is wavelength dependent. To simulate this, controls were added to the application to adjust transmittance values of the red, green and blue colour channels. The HLSL shader was extended to cater for calculations based on three independent colour channels.

Implementing this into our empirical shader, we decided to extend the transmittance shader to accept red, green and blue filtering amounts and multiply the outputted transmittance value colour channels by these filtering amounts. We speculated that different skin types would demand different colour channel coefficients to produce the most perceptually realistic renderings.

Based on the work of Zhao and Fairchild [37] as well as the spectral absorption coefficients identified by Donner and Jensen [12] and the fact that the model we used was a Caucasian model, we enhanced the red channel transmission relative to the green channel. We then subdued the blue channel, again relative to the green channel. Figure 4.9 demonstrates an enhanced fidelity by making this change.

To more accurately simulate this colour transmittance, we enhanced this further, to redefine the maximum thickness variable in the transmission shader to a three channel vector rather than a single float. Each component of the vector represents the maximum

Figure 4.9: Comparison of the rendering with transmittance equal in all wavelengths (left) compared with transmittance set individually using the parameters [red = 0.55, green = 0.29, blue = 0.28] (right)

thickness that light in that colour channel can penetrate through, relative to the global maximum thickness used in the transmittance shader. The transmittance shader was then updated to calculate for each colour channel, rather than just a global calculation.

4.6 Removing Reference to the Shadow Map

In order to eliminate areas of the model that should not be lit through forward scattering, the standard lighting shadow maps are used. One of the goals of the algorithm was to investigate methods that do not rely on the shadow map. Various methods were investigated but none could provide a comprehensive solution.

4.6.1 Varying the Minimum Dot Product

Based on this, the solution was to parametrize the shader to accept a minimum *dot product* threshold. In the previous revision, we assumed that any value less than 0 (incidence angle greater than 90 degrees) should be considered in the transmittance calculation. Rather than assuming that 0 was the correct value, the algorithm was parameterised to allow this variable to be set for the scene. The shader was extended to accept this threshold parameter. All pixels below the threshold were scaled to a range $\langle 0, 1 \rangle$ from the $\langle minimumthreshold, -1 \rangle$ using the scaling formula in equation

4.6.

$$S(x) = 1 - ((1 + Dot_{c.l})/(Dot_{min} + 1)) \tag{4.6}$$

where $S(x)$ is the scaled transmittance value, $Dot_{c.l}$ is the *dot product* of the camera and light direction vectors and Dot_{min} is the parameterisable minimum *dot product* value

This provided a solution but still had the limitation of fixing a global value for the Dot_{min}. This compromise meant that this solution only works for a few scene scenarios and degrades the quality in all other scene combinations.

4.6.2 Dynamic Variation of the Minimum Incidence Angle

One of the major issues is avoiding adding forward scattering lighting to areas of the model that should not be lit. This is a function of working purely in screen space. The main problem is that there is no specific value of the minimum incidence angle that works for all scenarios.

The solution identified is to vary Dot_{min} (minimum incidence angle) depending on how the model is viewed. This was implemented by describing a vector in world space that is related to the model. For this, an imaginary vector is described pointing directly from the front of the face. This is referred to as the 'face vector'. The face vector is translated and rotated with the model. Using the face vector and the standard camera direction vector we calculate the *dot product* of these vectors. This is used as an index into a look up table (LUT) that returns the minimum *dot product* value to be used in the calculation. This was implemented as part of the XML scene description file and could be modified outside the application for tweaking. The LUT was divided into 10 degree angles around the Y-axis and the lookup would select the appropriate Dot_{min} for the computed angle of face and camera vectors.

In the application, the LUT is queried in the C# code to select the appropriate value based on the result of the face-camera *dot product*. This lookup will select both the appropriate previous and next values in the lookup table and linearly interpolate between these based on the incidence angle. The result of the interpolation is the minimum *dot product* value use by the translucency calculation.

This is a solution only for camera rotations around the Y axis and should be

generalised into three dimensions. This would result in a LUT that emulates points on the surface of a sphere. Having a three dimensional table then means we can effectively control the parameters for all situations.

However this solution is both overly complex, hard to configure and does not produce the same image quality as using the shadow map. Therefore the shadow map is used the algorithm It was not possible to decouple the transmittance algorithm from the general shadow map while retaining the same rendering quality.

4.7 Optimizations

This section discusses the various optimizations that have been made to the base code in order to reduce the computation overhead of the shader. Since this shader is not physically based, minimising the overhead was a priority.

4.7.1 Elimination of Unnecessary Calculations

The advantage of working within screen space is that the algorithm only needs to be run under certain conditions. As we have introduced a minimum incidence angle for forward scattering calculation, we can use this in the C# code to determine if the transmittance should be calculated. For any situation where this condition is not met, the execution speed simply reverts to that of the normal skin shader.

4.7.2 Combine HLSL Shaders

The shader was built with multiple `RenderTargets`, to aid the tweaking of the shader and to view intermediate states of the shader pipeline. However this brings additional overheads in terms of render target switches and performing some calculations unnecessarily. This optimization will eliminate the thickness pass by combining it with the transmittance pass. Rather than pass the thickness of the model in camera to the transmittance effect, we will pass the depth textures and calculate the thickness and transmittance at the same time.

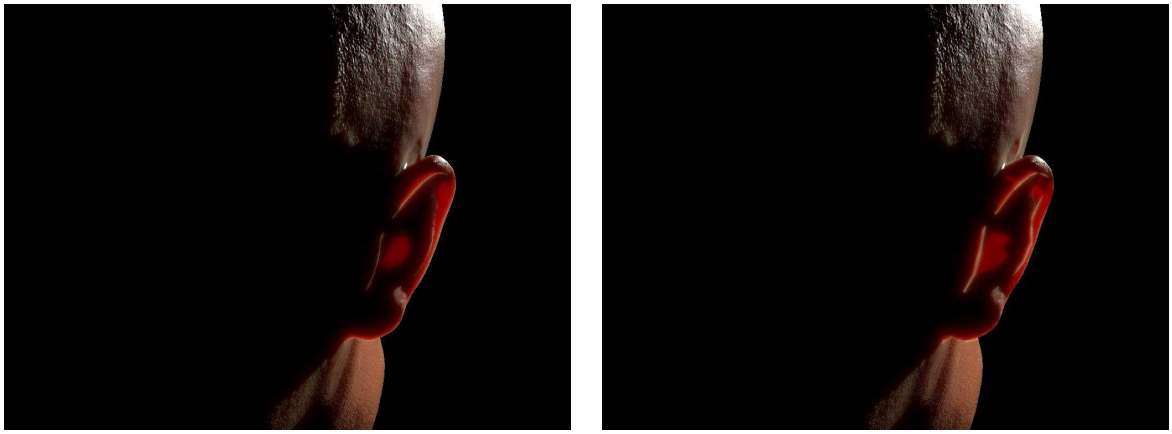

Figure 4.10: Comparison using full model (left) versus limited polygon model (right) for calculation of transmission component. While the same general areas are lit, we find that using the less detailed model produces more artifacts in the forwards scattered areas.

4.7.3 Reduced Model Quality for Transmission Calculations

During development, it was hypothesized that the quality of the model used for that transmission calculation did not need to be of the same level of detail as the model used to render the final images. We introduced the optimisation to perform all transmission calculations with the least complex model in terms of polygon count. This provided an additional performance boost with little discernible loss of quality. To further improve the performance of the algorithm, a transmission specific model could be used, whereby areas where transmission is expected to occur, such as nose, ears, hands etc., could use a more complex polygon mesh and the other parts of the model could be kept with a simpler mesh or eliminated completely.

4.7.4 Calculate Depth Maps on Alternative Frames

A primary overhead in the algorithms is the requirement for it to switch `RenderTarget`s during the computation. A `RenderTarget` switch will block the GPU while it takes place, until such time as the switch has completed. The generation of both the front facing and back facing depth maps require such a switch. Therefore in order to reduce the time in which the GPU is blocked, each frame will render either the front or back depth map, alternating between frames. Translucency calculations will then use the

newly calculated depth map along with the cached version of the other depth map from the previous frame. This effectively removes a `RenderTarget` switch from the algorithm.

4.7.5 Coherence

The performance of the algorithm can be further improved by caching the previously calculated transmittance map and reusing it in the next frame, when it is determined that the scene has not changed significantly. Doing this avoids the recalculation of the two depth maps, the thickness map and the transmittance map. A delta is defined between the camera-light *dot product* from the frame that last computed the transmittance map and the camera-light *dot product* in the current frame. Only if this delta is greater than the configured threshold is the transmittance recomputed. For all other frames the cached transmittance map is used. It was found from observation that a threshold of 0.005 was sufficient for most scenarios. This could be increased in certain scenarios such as when the camera and light vectors were near right angles and decreased when the vectors are near parallel. The performance benefit could be further improved by adaptively changing this threshold, depending on the *dot product* of these vectors.

It should be noted that the coherence and depth map optimisations cannot be combined, as this results in a jittering of the translucent portion of the rendered image when the scene changes (i.e. camera movement or light movement).

4.8 Summary

This chapter presents a practical model for rendering translucency in human skin. It is demonstrated that using the screen space technique to generate translucency values and applying these to the general lighting algorithm before applying the screens space blur, produces images of reasonable fidelity while maintaining high performance. This technique, when used with the algorithm proposed by Jimenez *et al.* [24], provides a novel empirical method of realistically rendering skin, that compares well with the solution of d'Eon *et al.* [10]. The challenges and their solutions have been presented

here and provide a platform for further optimisations and improvements.

It has also been shown that the technique lends itself well to optimisation and as is shown in Chapter 5, these optimisations can significantly improve the performance of the algorithm. The most costly part of this technique is the generation of the front and back face depth maps, due to having to process the complete model twice.

Chapter 5

Evaluation

The algorithm is evaluated using three metrics, performance, transmittance colouration and comparisons to d'Eon *et al.*'s [10] approach.

5.1 Performance Evaluation

Table 5.1 shows the performance benchmarks of the application in frames per second. The data was obtained using a single point light source, a shadow map resolution of 1024 x 1024, 2988 triangles per head rendered at 1280x720 resolution (720p). The images were rendered on a machine with a Intel Q6600 Quad core processor, a Nvidia GTX275 GPU with 896MB DDR3 RAM.

The test consisted of rotating the head on the Y axis while also rotating the light source around the Y axis at a faster rate. The test runs for 60 seconds.

As can been observed from the benchmarks, in the worst case scenario (whereby the transmittance is calculated every frame regardless of whether it is necessary) adding transmittance to the algorithm causes the frame rate to drop from 304 to 217. This is a significant difference but still a very positive result. It is shown that it is possible to optimise this result to significantly improve the frame rate and in the best case (most optimised) scenario the cost of the algorithm only causes an average of 22 FPS reduction (i.e running at 90% of the Jimenez *et al.* [24] screen-space shader).

To determine the component cost of the algorithm, the test was rerun with each component disabled, and the results for the captured in Figure 5.2.

Run	Test Type	FPS
1	Baseline (with shadow mapping and diffuse lighting)	540
2	1 + Jimenez Screen Space shader	304
3	2 + Transmittance	217
4	3 + Transmittance when necessary	248
5	3 + Coherence (0.005)	246
6	3 + Coherence (0.01)	269
7	3 + Transmittance when necessary + Coherence (0.005)	266
8	3 + Transmittance when necessary + Coherence (0.01)	282

Table 5.1: Performance (in frames per second) of various configurations of the algorithm.

Run	Test Type	FPS
1	Full Transmittance	222
2	1 less the transmittance shader	239
3	2 less thickness shader	258
4	3 less back depth calculations	278
5	3 less all depth calculations	304

Table 5.2: Performance breakdown of the individual components of the algorithm.

As can be seen from Table 5.2, the cost of computing the transmittance relative to the overall cost of the algorithm is quite low. From the performance data it was speculated that switching `RenderTarget`s was causing many wasted computation cycles. Switching from one `RenderTarget` to another and saving its output to a texture for use by the next shader blocks the GPU from computation during the switch.

This is the case for both the thickness shader and the two depth shaders. In order to test the optimisation, the algorithm was benchmarked using the same conditions for three further scenarios. Results can be seen in Figure 5.3. In the first scenario, the thickness shader and the transmittance shader were combined into a single shader. In the second scenario, the first scenario is combined with the computation of each depth map on alternative frames, rather than both per frame. The third scenario combines the first scenario is combined with the coherence optimisation.

The performance test shows that the screen space algorithm is not a computationally expensive algorithm. A majority of the overhead of the algorithm come from having to switch `RenderTarget`s. By focusing the optimisation on eliminating the switches, it has been demonstrated that the algorithm performs well in real-time. All

Run	Test Type	FPS
1	Baseline (with shadow mapping and diffuse lighting)	540
2	1 + Jimenez Screen Space shader	304
3	2 + Transmittance	217
4	3 + Combined Thickness/Transmittance Shader	239
5	4 + Alternate Frame Depth map Calculations	261
6	4 + Coherence (0.005)	265
7	All optimisations	296

Table 5.3: Performance (in frames per second) of the algorithm when combining the thickness shader and translucency shader. Compared with Figure 5.1 and improvement can be seen in all optimisation conditions.

optimisation show no significant loss in image quality with the exception of running both the coherence and alternate frame depth maps optimisations together. Doing this results in the transmittance portion of the rendering jittering as the camera or light source moves. Therefore only one of these optimisations should be used.

5.2 Colour Modulation Evaluation

The psychophysical evaluation consisted of participants selecting a colour value that gave them the most perceptually believable image. It was hypothesised that the relationship between the red, green and blue channels should be similar to the transmittance depth relationship shown in experiments by Zhao and Fairchild [37], which shows empirical evidence that red wavelengths will transmit deeper into skin tissue that shorter wavelength.

A two way ANOVA was performed to establish the relationship between the colour channels for 12 participants and six test scenes. Two main factor are investigated, the color channel relationship and scene as well as the interaction between them.

As per the hypothesis, we see that the red channel is selected as the maximal channel by all participants. The interesting data is in the relationship between the green and blue channels. In Figure 5.1, using the ANOVA, it is shown that there is a measurable statistical difference in four of the six scenes for the relationship between the green and blue data points.

It is hypothesized that this is caused by two factors. Firstly, in certain scenes small

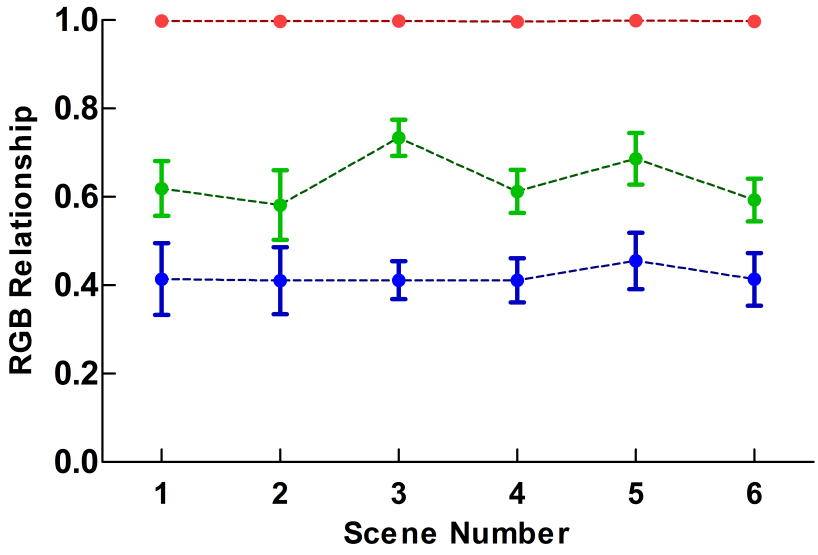

Figure 5.1: A statistical comparison of the results. The graph shows the selected relationship between the three colour channels across the six scenes. Mean values are represented here for all six scene. Standard error of mean is also shown, represented by the error bars.

Scene	t statistic value	P value	Significance
1	2.926	<0.05	* (small significance)
2	2.440	>0.05	ns (no significance)
3	4.606	<0.001	*** (high significance
4	2.870	<0.05	* (small significance)
5	3.299	<0.01	** (medium significance)
6	2.562	>0.05	ns (no significance)

Table 5.4: ANOVA results for green versus blue channel for each of the scene. This demonstrates that participants gave a varied set of answers to the scene data.

changes in the colour relationship does not produce a significant observable change in the rendered image. This leads to a more varied result set from participants. This is especially true for scenes three and five, which as shown in Figure 5.4 are the most statistically interesting. Secondly, the lighting conditions of the rendered scenes are an uncommon occurrence in general day to day observations for most people. Therefore they are not as tuned to observing these scenarios and as such when selecting responses,

a wider spread of data is observed.

From this we can hypothesise that the precise colour of the translucency effect when implemented in conjunction with subsurface scattering from a perceptual perspective is not as important to the final rendered image as other aspects of subsurface scattering. This is backed up from the test data (see Figure A) where discrepancies between the scenes in the green channel are frequent.

5.3 Image Fidelity Evaluation

In order to validate the quality of the images produced by the algorithm, it was compared with images from d'Eon *et al.*'s algorithm. An initial observation on our algorithm was that it was configured to produce too much translucency relative to the other algorithm and required tweaking.

Figure 5.2: Without taking into account the distance and intensity of the light source, the translucency effect is too intense (left). Modulation of the translucency produces a more realistic image (right).

It was realised that to generate a similar colour intensity to the test image, the translucency map needs be modulated based on the distance from the pixel to the light source as well as the intensity of the light source. Due to time constraints this was not completed, however to simulate this modulation, we fixed values for the light source to replicate the scene setup used by d'Eon *et al.*. Figure 5.2 shows the result of this modification. It can be clearly seen that the modulated image is an improvement on

our original translucency image.

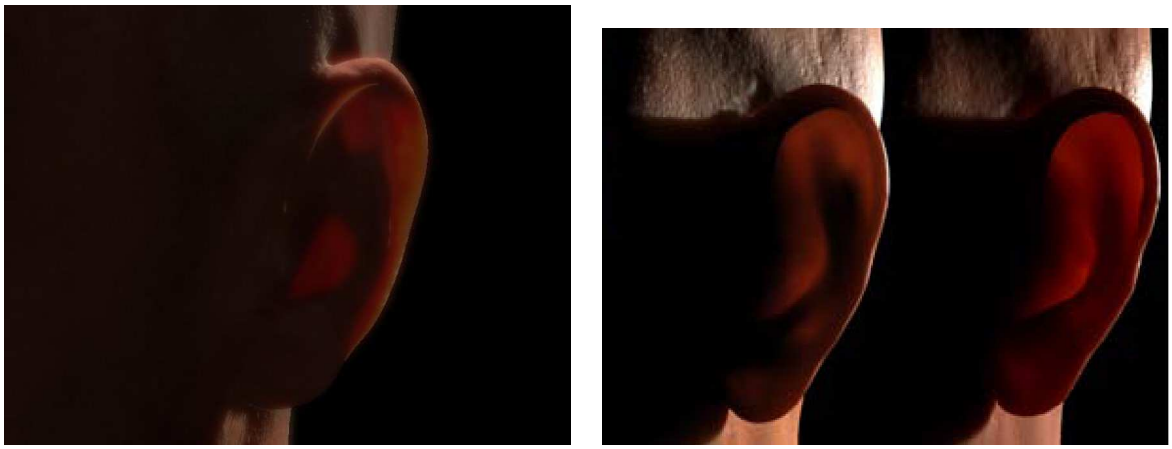

Figure 5.3: A Comparison of the ear as rendered by our algorithm (left), d'Eon *et al.*'s algorithm (center), and Monte Carlo methods (right).

In Figure 5.3 the output from both algorithms is shown. The key difference between them is that the screen-space algorithm incorrectly lights the ear because of the screen-space depth calculations. This is more prominent when the camera and light vector tend towards orthogonality. This is a mostly unavoidable attribute of the algorithm. While there are discrepancies, the general output of the algorithm is similar and this artifact is not present in all scenarios and is difficult to notice when the scene in motion.

5.3.1 Other Observations

The proposed optimisation to reduce the polygon count of the model when computing the translucency map did not give any performance increase. However the loss in quality from this was quite noticeable. Therefore it is recommended to ensure that the model used in the translucency calculation is the same as the main model using in the rendering.

In regards to the transparency intensity modulation due to the light incidence angle, it was observed that using the *dot product* relationship produced poor transitions and is not recommended. Applying the inverse cosine generated a believable transition from non-translucent regions to translucent regions (for instance as the light moves around the model). The quadratic and cubic curves did not significantly improve the quality of inverse cosine.

5.4 Summary

While currently these techniques are not used in real-time game engines, it is apparent that at the current rate of progress, real-time games will need to make the same leap in terms of skin rendering that the film industry made, as described in Section 2.6.1. Research into real-time subsurface scattering is still in its infancy. However recent work has shown that it is possible to include these effects in games. It is demonstrated that using a combination of screen-space irradiance convolution and our screen space translucency algorithm, it is possible to render a perceptually viable approximation of the physical lighting model at a low computational cost.

It has been demonstrated that the screen-space algorithm, although an approximation, could be better suited to video game engines than the physically correct algorithms, because of the performance benefits. This is because of the low computational and memory requirements of the algorithm.

Chapter 6

Conclusions

The purpose of this thesis as set out in Chapter 1 was to investigate and implement a screen space translucent shader for skin. Using an initial premise that it was possible to simulate the effect of light diffusing through skin, producing a perceptually plausible synthetic image, while performing well in real-time.

6.1 Conclusions

This thesis presents a screen-space algorithm that compliments previous subsurface scattering work done in the screen-space domain. The main goals of the algorithm were to produce a plausible image at the lowest cost as possible. While previous work on real-time subsurface scattering has produced excellent image quality in the case of d'Eon *et al.* [10], the trade off is poor scalability. In the case of Jimenez *et al.* [24] a performance increase is shown at the cost of not implementing translucency.

The presented work demonstrates that Jimenez's work can be extended well to include translucency. By working in screen space we have retained the optimisations that are available in the GPU and identified by Jimenez *et al.*. A set of optimisations that can be used at the discretion of the implementor are also demonstrated. It has been shown that the performance of the algorithm even without using any optimisation is impressive and the presented optimisations do not reduce the image quality for a large majority of situations while producing a significant performance improvement.

The user trial produced interesting results showing that there is variance in how

different users perceive the same scene in relation to the simulated colours. Further investigation could be warranted to identify why this occurred.

6.2 Future Work

We set out potential further improvements and enhancements that could be made to the algorithm.

6.2.1 Generalisation

Currently as the algorithm stands it is tailored specifically to the skin domain. It could be worth investigating if the technique can generalise to other types of translucent materials. To do this investigations into parameterising the algorithm based on the subsurface scattering coefficients as defined and used in other research would be required.

6.2.2 Improving the colour transmittance

The algorithm uses a three colour channel transmission model which is constructed from empirical observations. This model is sufficient for a simple model of light photons transmitting through skin. Further investigations could be made into replacing this model with a more complex model which could account for transmittance based on light wavelength. This could be achieved by investigating light wavelength to RGB conversion methods. Methods to relate the thickness of a region to the wavelengths that transmit though it and the intensity at which they transmit are also required. This could potentially produce more a accurate and believable use of colour in translucent regions.

Skin types

In this dissertation, only a single skin type was examined. Future work should include investigation into how the algorithm works with other types of skin. The color transmittance algorithm should be validated against various skin types and extended or improved to support any differences found.

6.2.3 Integration with Games Engines

Integrating the algorithm with an existing 3D games engine and investigating the results would be a logical next step for testing the benefits of this algorithm. Further perception tests could be performed in a more complete scene setting, which could include more interesting lighting scenarios. This would help determine if this algorithm can generate more realistic real-time renderings.

6.2.4 Further Optimisations

Further optimisations could be explored are described below

Reducing the number of passes

As has been discussed in chapter 4, swapping `RenderTarget`s is a costly operation. A further optimisation that was not implemented is to combine the transmittance shader with the second depth calculation. Therefore the algorithm reduces to a two pass algorithm, the first pass consisting of calculating the depth of the front facing polygons. In the second pass the second depth (back facing polygons) calculation is performed in the transmittance shader, eliminating a `RenderTarget` swap. It is estimated that this would produce a frame rate equivalent to the alternate frame depth map calculation optimisation, while not suffering from any jitter artifacts in fast moving scenes.

It is also a logical conclusion that since this optimisation and others previously documented make use of the screen-space nature of the algorithm, they could potentially be extended to the work of Jimenez *et al.* to improve the performance of their shader.

Mark Transmittance Areas on the Model

This is a proposed optimisation and was not implemented for this application due to restrictions on the ability to manipulate the model file. The proposed optimisation is to reduce the amount of polygons that have to be rendered to calculate transmission. the model to be rendered is marked to identify areas in which transmittance can occur. The proposed way of doing this is to use the alpha channel on each vertex to identify if it should partake in the translucency calculation.

In this way we reduce the amount of polygons that are rendered to each of the depth maps and thus reduce the number of calculations required to build the transmission map.

6.2.5 Fixing the Frame Rate

Modern game engines make use of all available resources in order maximize performance. This is especially true for game consoles. Because of the design of this algorithm it is easily possible to allow a game engine to adaptively control the algorithm's parameters in order to make best use of the available computational cycles available for a given frame. As has been demonstrated, running with all the optimisations enabled, the cost of the shader is very low. In scenarios where other complex effects are being rendered which take precedence in the scene, the game engine would be able to adjust the algorithm parameters to ensure minimal computational cost. In less costly scenes the engine could ensure that the algorithm's fidelity was enhanced.

Appendix A

Perception Test Information

A listing of the scenes used for the perception test and the user responses.

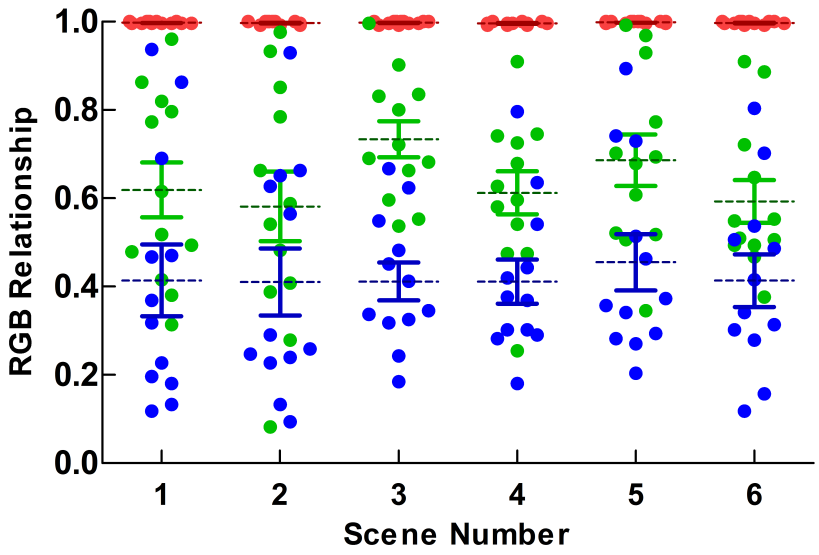

Figure A.1: A scatter plot showing individual values obtained from each participant for each scene. Mean values for each colour are represented by the dotted lines while standard error of mean are represented by error bars.

Figure A.2: Scene 1 (left) and scene 2 (right) of the perception test.

Figure A.3: Scene 3 (left) and scene 4 (right) of the perception test.

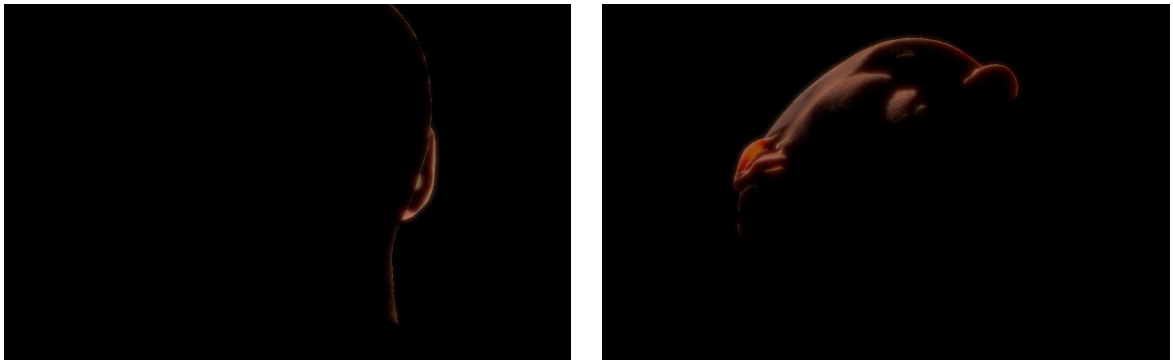

Figure A.4: Scene 5 (left) and scene 6 (right) of the perception test.

Bibliography

[1] C. Beeson and K. Bjorke. Skin in the" Dawn" demo. *ACM SIGGRAPH Computer Graphics*, 38(2):14–19, 2004.

[2] J. F. Blinn. Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, 1977.

[3] G. Borshukov and JP Lewis. Realistic human face rendering for" The Matrix Reloaded". In *International Conference on Computer Graphics and Interactive Techniques*, pages 1–1. ACM New York, NY, USA, 2003.

[4] B. C. Burnaby. The Quest for Realism: Skin Rendering Techniques in Games. 2003.

[5] N. A. Carr, J. D. Hall, and J. C. Hart. GPU algorithms for radiosity and subsurface scattering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 51–59. Eurographics Association Aire-la-Ville, Switzerland, Switzerland, 2003.

[6] S. Chandrasekhar. *Radiative transfer*. Courier Dover Publications, 1960.

[7] P. H. Christensen, B. Bashforth, D. Batali, C. Bernardi, T. Jordan, D. Laur, E. Tomson, G. Quaroni, W. Wooten, and C. Hery. Renderman, theory and practice. *Siggraph Course Note*, 2003.

[8] Valve Corporation. Left4Dead. http://www.l4d.com/home.html, 08 2009.

[9] C. Dachsbacher and M. Stamminger. Translucent shadow maps. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 197–201. Eurographics Association Aire-la-Ville, Switzerland, Switzerland, 2003.

[10] E. dEon, D. Luebke, and E. Enderton. Efficient rendering of human skin. In *Eurographics Symposium on Rendering*, pages 147–157, 2007.

[11] C. Donner and H. W. Jensen. Light diffusion in multi-layered translucent materials. *Proceedings of ACM SIGGRAPH 2005*, 24(3):1032–1039, 2005.

[12] C. Donner and H. W. Jensen. A spectral BSSRDF for shading human skin. *Rendering Techniques*, 2006:409–418, 2006.

[13] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. K. Pedersen. Modeling and rendering of weathered stone. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 225–234. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1999.

[14] P. Dutre, P. Bekaert, and K. Bala. *Advanced global illumination*. AK Peters, Ltd., 2003.

[15] W. Engel. Implementing Lighting Models with HLSL, 2003.

[16] T. J. Farrell, M. S. Patterson, and B. Wilson. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. *Medical Physics*, 19:879, 1992.

[17] H. Gouraud. Continuous shading of curved surfaces. *IEEE transactions on computers*, 100(20):623–629, 1971.

[18] T. Igarashi, K. Nishino, and S. K. Nayar. The appearance of human skin. *Columbia Univ. Comput. Sci., Tech. Rep*, 2005.

[19] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 576–581, New York, NY, USA, 2002. ACM.

[20] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320. ACM New York, NY, USA, 1998.

[21] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518. ACM New York, NY, USA, 2001.

[22] H.W. Jensen. Global illumination using photon maps. *Rendering Techniques*, 96:21–30, 1996.

[23] J. Jimenez and D. Gutierrez. Faster rendering of human skin. *CEIG08*, 2008.

[24] J. Jimenez, V. Sundstedt, and Gutierrez D. Screen-Space Perceptual Rendering of Human Skin. *ACM Transactions on Applied Perception, 2009 (to appear).*, 2009.

[25] J. T. Kajiya. The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4):143–150, 1986.

[26] H. P. A. Lensch, M. Goesele, P. Bekaert, J. Kautz, MA Magnor, J. Lang, and H.P. Seidel. Interactive rendering of translucent objects. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 214–224, 2002.

[27] Callan McInally. Ruby: the doublecross. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Computer animation festival*, page 244, New York, NY, USA, 2004. ACM.

[28] M. Mittring. Finding next gen: CryEngine 2. In *International Conference on Computer Graphics and Interactive Techniques*, pages 97–121. ACM New York, NY, USA, 2007.

[29] F. Nicodemus, J. Richmon, J. Hsia, I. Gisberg, and T. Limperis. Geometric considerations and nomenclature for reflectance, NBS monograph 160. *National Bureau of Standards*, 1977.

[30] M. Oren and S. K. Nayar. Generalization of the Lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, 1995.

[31] M. Pharr and P. Hanrahan. Monte Carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proceedings of the 27th annual conference on*

Computer graphics and interactive techniques, pages 75–84. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000.

[32] Pixar. Renderman. https://renderman.pixar.com/, 08 2009.

[33] H. Rijpkema, G. Steele, and M. Derksen. It's not easy being green. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2008.

[34] C. Schlick. An inexpensive BRDF model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Citeseer, 1994.

[35] xyzrgb. XYZ RGB Ltd. http://www.xyzrgb.com/, 08 2009.

[36] T. T. Yu, J. Lowther, and C. K. Shene. Photon mapping made easy. *ACM SIGCSE Bulletin*, 37(1):201–205, 2005.

[37] Z. Q. Zhao and P. W. Fairchild. Dependence of light transmission through human skin on incident beam diameter at different wavelengths. In *Proceedings of SPIE*, volume 3254, page 354, 1998.