INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Subsurface Texture Mapping

Guillaume Francois & Sumanta Pattanaik & Kadi Bouatouch & Gaspard Breton

N° ????

juillet 2006

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Subsurface Texture Mapping

Guillaume Francois * & Sumanta Pattanaik † & Kadi Bouatouch ‡ & Gaspard Breton §

Thème COG — Systèmes cognitifs

Projet Siames

Rapport de recherche n° ???? — juillet 2006 — 28 pages

Abstract: Subsurface scattering within translucent objects is a complex phenomenon. Designing and rendering this kind of material requires a faithful description of their aspects as well as a realistic simulation of their interaction with light. This paper presents an efficient rendering technique of multilayered translucent objects. We present a new method for modeling and rendering such complex organic materials made up of multiple layers of variable thickness. Based on the relief texture mapping algorithm, our method calculates the single scattering contribution for this kind of material in real-time using commodity graphics hardware. Our approach needs the calculation of distances traversed by a light ray through a translucent object. This calculation is required for evaluating the attenuation of light within the material. We use a surface approximation algorithm to quickly evaluate these distances. Our whole algorithm is implemented using pixel shaders.

Key-words: Realtime graphics, GPU, Subsurface scattering

En collaboration avec France Télécom R&D Rennes et avec l'Université de Floride Centrale (UCF)

* gfrancoi@irisa.fr
† sumant@cs.ucf.edu
‡ kadi@irisa.fr
§ gaspard.breton@francetelecom.com

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Subsurface Texture Mapping

Résumé : La diffusion de la lumière à l'intérieur de matériaux participants est un phénomène complexe. Pour modéliser et rendre de tels matériaux, il est nécessaire d'avoir une description adaptée de ceux-ci ainsi qu'une simulation réaliste de leurs interactions avec la lumière. Ce papier présente une technique de rendu adaptée aux matériaux multicouches. Cette nouvelle méthode permet de modéliser des matériaux organiques complexes composés de couches multiples à épaisseur variable. Basée sur l'algorithme du *relief mapping*, notre méthode permet le calcul temps réel de la diffusion simple pour ce type de matériau, et ce en exploitant les performances des cartes graphiques. Notre méthode nécessite le calcul des distances parcourues par la lumière à l'intérieur des différentes couches du matériau. Ce calcul est nécessaire pour l'évaluation de l'atténuation de la lumière à l'intérieur du matériau. Nous proposons d'utiliser un algorithme d'approximation de surface pour réaliser ce calcul rapidement. Notre algorithme est implementé à l'aide de pixel shader.

Mots-clés : Rendu temps reel, GPU, Diffusion sous-surfacique

newpage

Contents

1 Introduction **4**
1.1 Related Work and Motivations 5
1.2 Overview 6

2 Our Method **8**
2.1 Subsurface Texture Mapping 8
2.2 Reduced Intensity Estimation 11
2.3 Algorithm 15

3 Results **19**

4 Conclusion and Future Work **23**

1 Introduction

Rendering realistic materials is essential for creating convincing images. The interactions between light and materials are often modeled by a BRDF (Bidirectional Reflectance Distribution Function) which assumes that light enters and exits the surface at the same point. This assumption does not hold for many materials such as marble, wax, milk, leaves and human skin. For these translucent materials, light does not only reflect off the surface, but scatters inside the medium before leaving the surface. This phenomenon is called subsurface scattering. Since such materials are often seen in our day-to-day life, it is essential to offer solutions to the rendering of translucent materials. The effects of subsurface scattering are multiple: the objects appear smooth and soft since light is diffused beneath their surface. For some complex materials, this phenomenon can also exhibit their inner composition. Veins are visible under human skin, for instance.

Figure 1: Subsurface texture mapping.

This paper presents a rendering technique for materials made up of multiple layers, the thickness of which is not necessarily considered as constant unlike existing methods. In such a case, the inner composition of the material can be perceived as shown in Figure 1, exhibiting blurry details inside the translucent object where the scattering parameters vary. We used the concept of relief texture mapping [1] to model the interior of a material. However, instead of representing the surface details, we use this method to represent the inner structure of the object. Therefore, the layers of the material are described by a simple 2D texture, where each channel encodes a thickness. Since our method is not limited to locally flat surfaces, a useful surface approximation is used to quickly estimate the single scattering term.

We propose a simple but realistic method that offers a real-time estimation of the single scattering term for such complex materials. Our method could be considered between subsurface rendering methods based on surfaces and 3D texture-based algorithms. Furthermore, our

solution also provides a compact way to design translucent objects using a small amount of data, whereas 3D texturing requires a large amount of memory storage.

This paper is structured as follows. Section 1.1 summarizes the related works and presents our motivations while Section 1.2 outlines our method that is described in detail in Section 2. Section 3 shows some results and Section 4 concludes this paper.

1.1 Related Work and Motivations

The radiative transport Equation [2], describing the subsurface scattering phenomenon, is too complex to be fully computed at interactive frame rates. Furthermore, the parameters of this equation such as the scattering coefficient or the phase function, can vary in the case of non uniform media or translucent object. Therefore, for materials such as marble, smoke, or layered materials such as skin or plant leaves, we cannot use the approximations usually proposed for simplifying the equations.

Light scattering within participating media or translucent objects can be computed using multiple methods such as path tracing [3], volumetric photon mapping [4] and diffusion approximation [5]. Most of these methods offer an accurate estimation of subsurface scattering for offline rendering but does not allow a fast rendering of translucent materials. However, in most cases, the computational cost of those methods prevents their use for real-time rendering of translucent objects.

The dipole based method due to Jensen et al [6] and proposed in real-time by [7, 8], deals with uniform materials, using a BSSRDF model (bidirectional subsurface scattering reflectance distribution function). Donner et al. [9] recently proposed a multipole based method to estimate subsurface scattering within layered materials. This new method, extension of the previous dipole approximation, provides realistic results close to Monte Carlo estimations. Nevertheless, if the algorithm proposes realistic and close to physically correct results, it does not offer interactive frame rates useful in many applications. Another real-time method [10] addresses scattering for general participating media but does not deal with multilayered materials.

3D textures [11] is the commonly used method to describe complex and translucent objects. They allow to describe the inner composition of an object, such as the organs of a human body and thereby, as well as the rendering of the interior of the objects. Path tracing can be used for a correct estimation of subsurface scattering. Implementation relying on 3D textures and using the GPU have been proposed in [12]. However, 3D textures require large amounts of data which can be a major constraint for real-time rendering. Note that some objects do not require a complex description deep beneath the surface. For example, the human skin presents particularities, such as veins, within a small depth beneath the surface. In this case, 3D textures appear unnecessary and limiting. For that particular case, we need both a surface description and a volume description, giving further details close to the object's surface only.

In this paper, we propose a novel description of complex materials. Rather than using the well-known 3D textures, we propose the use of simple 2D textures well handled by graphics hardware. Our method can be considered as an alternative method for computing single scattering within multilayered materials in real-time. Our approach allows a realistic description of the layers of variable thickness using subsurface texture mapping. Our method follows a similar concept than relief texture mapping introduced by Policarpo et al. [1] and recently proposed to describe non-height-field surface details with multiple layers[13]. Relief texture mapping offers highly realistic appearance to synthetic objects while using a simple geometry: fine details upon the surface are represented with a 2D texture. However, in our case, the details are not above but beneath the surface, which creates, for instance, particular veins effect.

1.2 Overview

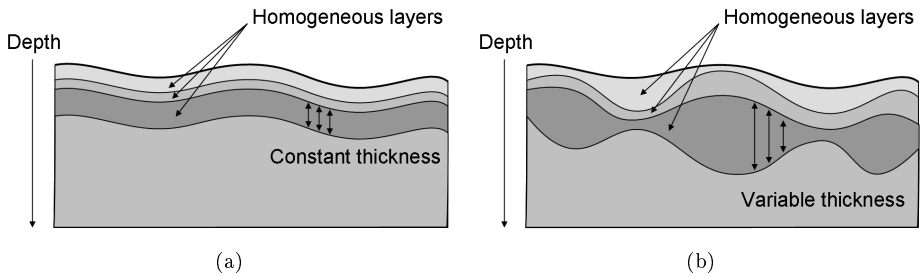


Figure 2: Figure 2(a) presents a simple layered material and figure 2(b) a material composed of layers with variable thickness.

In this paper we propose a new approach to modeling and real-time rendering of translucent organic materials. The materials considered in our case are composed of multiple layers, each one having particular properties. In contrast to methods using planar layers, our method computes single scattering into layers of variable thickness (cf. Figure 2). With these material properties one can render new effects not visible for layers of constant thickness. In order to provide a real-time but realistic rendering, we limit our computation to single scattering. Our method, implemented on graphics hardware, uses a ray marching algorithm illustrated in Figure 3.

Modeling of layered material requires information about the thickness of the layers since this thickness varies beneath the surface. We propose in Subsection 2.1 the use of a 2D

texture to store the thickness of each layer. We use this texture to determine the scattering parameters. The computation of subsurface scattering within this kind of non homogeneous material requires a knowledge of the layers' parameters. To this end, we propose a point and path localization method.
As shown in Figure 3, the reflected luminance at point P is due to single scattering events occurring along the viewing ray underneath the object's surface. We compute the contributions of a certain number of sample points M on the viewing ray until a certain point M_{max} after which scattering is considered as negligible.

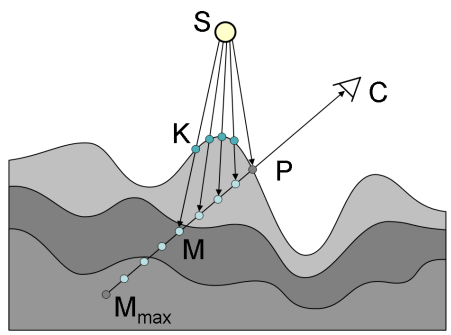


Figure 3: Ray marching algorithm in a multilayered material.

The single scattering contribution, $L_M(P, \omega_{out})$, at a visible point P and due to an inner sample point M on the viewing ray, is expressed as:

$$L_M(P, \omega_{out}) = Q(M, \omega_{out}) e^{\int_M^P -\sigma_t(s)ds} \quad (1)$$

$$Q(M, \omega_{out}) = \sigma_s(M) p(M, \omega_{out}, \omega_{in}) L_{ri}(M, \omega_{in}), \quad (2)$$

where $L_{ri}(M, \omega_{in})$ is the reduced intensity at point M coming from direction ω_{in} and represents the amount of incident light arriving at the inner point M after attenuation. The term $e^{\int_M^P -\sigma_t(s)ds}$ represents the attenuation of the scattered radiance due to absorption and scattering along the path from M to P. $p(M, \omega_{out}, \omega_{in})$ is the phase function and describes how the light coming from the incident direction ω_{in} is scattered in the outgoing direction ω_{out} at point M.

The total scattered radiance due to single scattering arriving at the point P is the sum of the contributions of all points M on the viewing ray:

$$L(P, \omega_{out}) = \int_P^{M_{max}} L_M(P, \omega_{out}) dM \quad (3)$$

We evaluate this last equation using a ray marching algorithm, which discretizes the equation as:

$$L(P,\omega_{out}) = \sum_{P}^{M_{max}} L_M(P,\omega_{out})\delta_M \quad (4)$$

$$= \sum_{P}^{M_{max}} Q(M,\omega_{out}) e^{\int_M^P -\sigma_t(s)ds} \delta_M \quad (5)$$

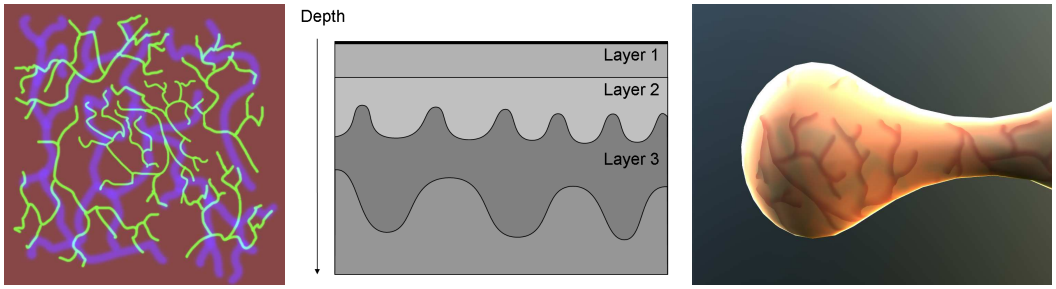
where δ_M is the sampling step along PM_{max}.
The term $e^{\int_M^P -\sigma_t(s)ds}$ is estimated by a ray marching algorithm. This latter performs depth comparisons to determine the extinction coefficient σ_t for each layer. The estimation of the term $Q(M,\omega_{out})$ of Equation 1 requires the calculation of the reduced intensity. We show in Subsection 2.2 that the estimation of the reduced intensity needs an estimate of the distance $\|KM\|$ traversed by the light in the material. We propose a method to compute an approximate value of this distance. This is one of the contributions of this paper. The single scattering result $Q(M,\omega_{out})$ at an inner point M also depends on the properties of the material. Thereby, we need to determine the layer in which is located the point M, to know for instance the corresponding scattering coefficient $\sigma_s(M)$. To this end, we propose a simple material description that allows fast point localization guided by textures. With these contributions, detailed in the next section, real-time rendering of subsurface scattering in such materials can be achieved using graphics hardware. In Subsection 2.3 we propose implementation solutions for graphics hardware.

2 Our Method

We propose a simple but realistic solution to the rendering of multilayered materials. In our case, we do not assume the layers with a constant thickness. We limit our computation of subsurface scattering to single scattering to meet the real-time constraint. The next two subsections present the subsurface texture mapping idea and a novel method to estimate the reduced intensity on arbitrary, non planar, polygonal surface described by a mesh. The solutions provided by these two subsections are combined in our rendering algorithm presented in Subsection 2.3.

2.1 Subsurface Texture Mapping

In this section, we present the notion of subsurface texture used to describe the internal structure of a translucent object. Our method, built on the same idea as relief texture mapping proposed by [1], uses a texture as a depth map which allows to describe complex objects when mapped onto a flat polygon as shown in Figure 4. We propose to use a 2D



(a) (b) (c)

Figure 4: Subsurface Mapping : The Figure 4(a) presents the subsurface texture. The red channel encodes a layer with a constant thickness, the green and blue ones describe layers composed of veins with different width. The alpha layer, not visible here, encodes a layer with a uniform thickness. Figure 4(b) shows the representation (cross-section) of such a material. Figure 4(c) shows the resulting image obtained when applying such a material on an arbitrary surface.

texture to describe our multilayered material. Indeed, the depth of each non planar layer can be encoded using a relief map. The presence of the four RGBα channels of a texture allows to define the thickness of four layers. The red channel is related to the first layer and the other channels to the following layers.

As shown in Figure 5, the depth stored in each channel represents the distance of the layers to the surface in the direction of the normal. The thickness of a layer can be obtained using the depth information stored into two consecutive channels. N textures can describe up to $4N$ layers, which allows a complex definition of a multilayered material. An algorithm using ray tracing can be used to compute the subsurface texture when each layer is described by a mesh.

Estimating the single scattering at a point M beneath the surface requires information on the layer in which is located M. By comparing the depth of the point M and the related depth of each layers stored into the texture, we can determine the layer of interest and use its specific parameters.
The process of mapping our texture of layer thickness onto a polygonal surface is presented in Figure 6 and explained as follows:

- determine PM, with the point M inside the medium and lying on the viewing ray.
- Project the vector PM in the tangent space (defined by the tangent T, normal N and bi-normal B vectors).
  $(ds, dt) = (PM \cdot T, PM \cdot B)$

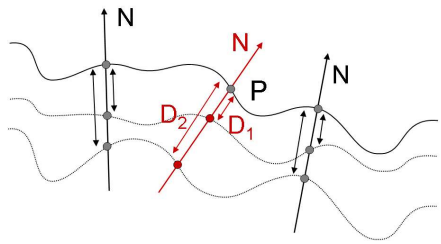
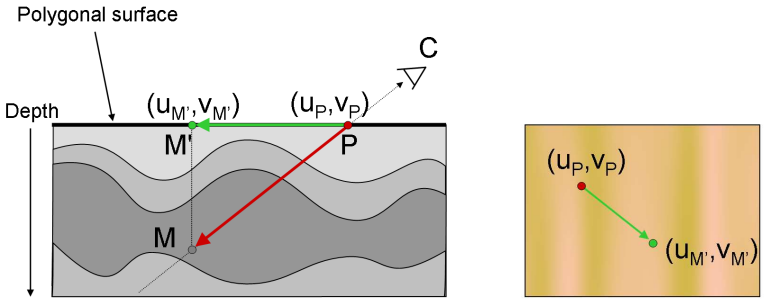


Figure 5: Definition of the subsurface map

- use the projected PM, (ds, dt), and the texture coordinates of the point P, (u_P, v_P), to compute the texture coordinates of M': $(u_{M'}, v_{M'}) = (u_P, v_P) + (ds, dt)$
- determine the layer of the point M by comparing its depth with the depths of the layers stored in the subsurface texture at $(u_{M'}, v_{M'})$.



(a) Profile of the material (b) Projected vector into the texture

Figure 6: Projection in the tangent space.

This process is illustrated in Figure 6. Point P has a depth equal to zero. 6(b) presents the projection of the vector PM into the texture space. Note that the tangent and bi-normal are related to the texture coordinates. The tangent gives the direction of variation of the u texture coordinate and the bi-normal gives the direction of the variation of the coordinate v. For more details see [14] and [1].

This mapping allows a complex description of layered materials with simple localization of points within it using a single texture lookup. We can describe up to $4N$ layers using N

RGBα textures. This is one of our contributions. We will see in the Subsection 2.3 how to compute the single scattering using such a material description.

2.2 Reduced Intensity Estimation

This section gives details on the calculation of the reduced intensity $L_{ri}(M, \omega_{in})$ (see Equation 2) which accounts for the attenuation of light within a multilayered material before being scattered. Refer to Figure 7(a) for the explanations given hereafter.

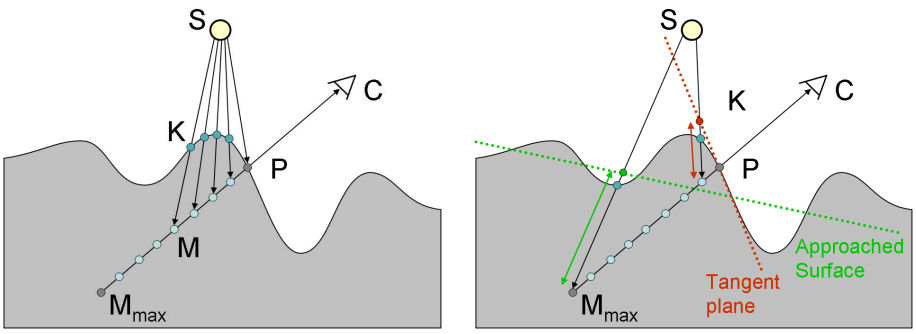
For the sake of clarity, we first consider the case of a single layer material having a constant extinction coefficient σ_t. The calculation of the reduced intensity depends on the amount of light $L_i(K, \omega_{in})$ impinging onto the surface of the medium at a point K and depends on the distance $||KM||$ traversed by the light ray within the medium. The reduced intensity is given by:

$$L_{ri}(M, \omega_{in}) = L_i(K, \omega_{in}) e^{-\sigma_t ||KM||} \tag{6}$$

The problem can be reduced to the computation of the distance $\|KM\|$, the term $L_i(K, \omega_{in})$ is given by the light source intensity. This distance can be accurately obtained using ray tracing. Nevertheless, using the graphics hardware, we cannot compute easily the intersection between the light ray SM and the surface. For a fast estimation of the distance $||KM||$, without the need of detailed information about the surface, we propose to use planar approximations of the surface which, when intersected by the light ray SM, gives a reasonable estimate of the position of the point K (see Figure 7).
Note that in [15] the authors present a method addressing a similar problem and implemented on the GPU. They use a planar approximation method to compute the point hit by a reflected or a refracted ray. Their method provides more accurate results and performs planar approximations at runtime based on the model's geometry and rays (reflected or refracted). Our method is faster since all the planar approximations are performed in a simple preprocessing step, only based on the model's geometry and the material properties (extinction coefficient).

First, let us consider the points M on the ray path close to point P. In this case, we observe that replacing the surface by its tangent plane at P offers a reliable estimate of the distance. This is due to the proximity of the intersection between the real surface and the light ray and the intersection of the light ray with the tangent plane. Most of the light scattered at points close to P strikes the surface at points also close to the point P. Figure 7(b) presents the surface approximation (in red) that can be used for these points M. Note in Figure 7(b) that this approach is not reliable when the light enters at points far from P since the tangent plane gives an incorrect estimate of the light entry point. Nevertheless, in these latter cases, the attenuation of light is significant compared to the attenuation of light impinging onto a close neighborhood of P. This is due to the distance of light traversal. We consider then that the tangent plane offers a reliable approach of the surface to estimate the scattering

contributions, even if it does not give accurate results for some low-level contributions as explained before.



(a) Single Scattering (b) Planar Approximations

Figure 7: Reduced intensity estimation using an estimate of the distances.

For more distant points beneath the surface, the tangent plane at point P does not offer a satisfying approximation of the surface. Therefore, the tangent plane at point P does not approximate well enough the surface points at which the light enters before being scattered deeply within the medium. In Figure 7(b), the intersection between the tangent plane and the light ray SM_{max} is far away from the real intersection point. We propose to use another planar approximation of the surface which allows a more accurate estimation of the distance traversed by the light within the medium. This new plane is drawn in green in the Figure 7(b). For these distant points, to calculate the distance $\|KM\|$, we need to make use of a

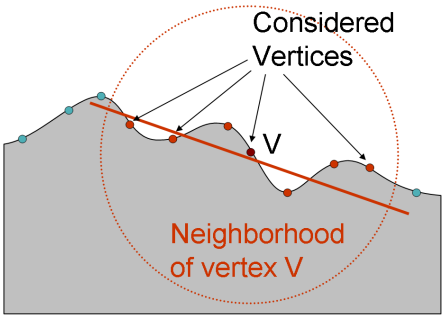


Figure 8: Neighborhood for the estimation of the plane related to M_{max}.

plane that better approximates the surface around the point P where the light intersects it. Such a plane is determined in a preprocessing step using a least square fitting algorithm

operating on the vertices of the surface included in a spherical neighborhood centered at point P.
The Figure 8 shows the points used for the calculation of such a plane whose intersection with the view ray yields a reliable estimate of $\|KM\|$. For each vertex V of the surface, we calculate its spherical neighborhood following the radiative transfer equation: points far from the vertex V do not contribute to its outgoing radiance. Since the light follows an exponential attenuation within the medium, the vertices are considered as 'too far' from the considered vertex V when their distance from V is higher than a given maximum radius. The radius of the neighborhood, represented by a sphere, depends on the extinction coefficient of the material σ_t. The attenuation for a straight traversal of the light from a vertex L to the vertex V is equal to $e^{-\sigma_t\|VL\|}$ (in our case, σ_t is the extinction coefficient of the first layer). If we consider that the light coming from the vertex taken into account in the plane calculation as to be attenuated less than a user defined ϵ, it leads to the equation of the radius r of the sphere, derived as follows:

$$\begin{array}{ll} e^{-\sigma_t\|VL\|} & \leq \epsilon \\ \|VL\| & \leq -\frac{\log(\epsilon)}{\sigma_t}, \end{array} \tag{7}$$

which leads to:

$$r = -\frac{\log(\epsilon)}{\sigma_t} \tag{8}$$

In Figure 8 the vertices out of the sphere are not taken into account for the plane estimation. The approximated plane is then obtained with a surface fitting technique using a root mean square method. The Figure 7(b) shows the surface approximation in green and the corresponding intersection with the light ray.

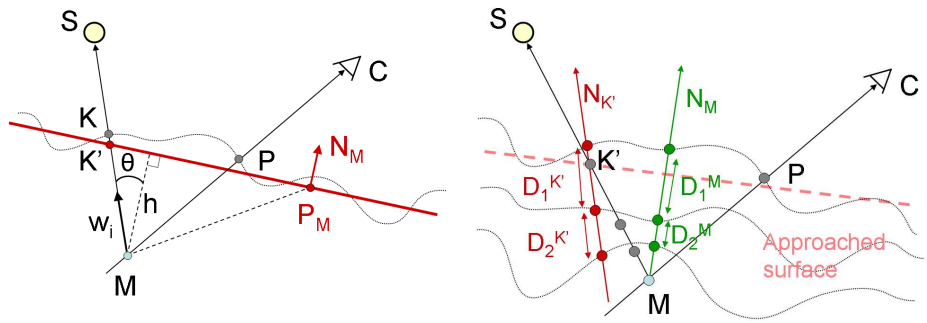
For a point M of intermediate depth, i.e. between point P and point M_{max} on the ray path, we propose to use another approximating plane for the calculation of the attenuation distance KM. This new plane is obtained using the tangent plane and the approximated plane computed for the point M_{max}, calculated as described above. Each plane is represented by a normal and a point. The plane related to M is denoted $\Pi_M\{\vec{N}_M; P_M\}$. The normal vector to this plane is obtained by interpolation of the normal to the tangent plane, i.e., the normal of the surface at point P, and the normal to the plane related to the point M_{max}.

$$\Pi_M : \left\{ \begin{array}{ll} \vec{N}_M & = \frac{1}{(\alpha+\beta)}|[\alpha\vec{N}_P + \beta\vec{N}_{M_{max}}] \\ P_M & = \frac{1}{(\alpha+\beta)}[\alpha P_P + \beta P_{M_{max}}], \end{array} \right. \tag{9}$$

where $\alpha = \|MM_{max}\|$ and $\beta = \|MP\|$

The calculation of $\|K'M\|$, estimate of $\|KM\|$, is presented in Figure 9(a), where: $h = |P_M M \cdot N_M|$, $h/\|K'M\| = \cos(\theta)$ and $N_M \cdot \omega_{in} = \cos(\theta)$, which leads to:

$$\|K'M\| = \frac{|P_M M \cdot N_M|}{|N_M \cdot \omega_{in}|} \tag{10}$$



(a) Calculation of the distance using the plane approximation

(b) Estimation of the distances with the depth at point M and point K'

Figure 9: Distance Calculation.

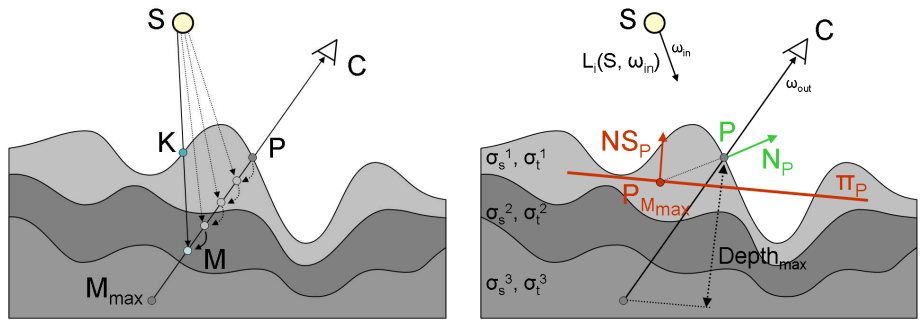
In the case of a multilayered material, the attenuation of the incident light depends on the parameters of each layer. For a three layers material, the Equation 6 is modified as follows:

$$L_{ri}(M, \omega_{in}) = L_i(K, \omega_{in}) e^{-\sigma_t^1 d_1} e^{-\sigma_t^2 d_2} e^{-\sigma_t^3 d_3}, \tag{11}$$

where d_i and σ_t^i are respectively the distance of the light path inside the ith layer and the extinction coefficient of this layer. We propose to estimate the distances using the same idea proposed above. To obtain a correct information on the thickness of the layers along the light ray path, we require a ray tracing or ray marching algorithm with a large number of texture lookup into the subsurface map. Since we want a fast estimate of the reduced intensity, we do not use such algorithms and still calculate the distance $\|K'M\|$ as presented above, without taking the multiple layers into account. To estimate the distances d_i, distances traversed by the light inside the ith layer, we use the depths D_j^M and $D_j^{K'}$ stored into the subsurface texture map for points M and K' respectively, as shown in Figure 9(b). We obtain the texture coordinates of the points M and K' by projecting the vector PM, respectively PK', into the tangent space and adding the results to the texture coordinates of point P. If we suppose that the layer thicknesses between point M and point K' are constant, then we can estimate the light path length in each layer:

$$d_i \approx \min\left(\|K'M\| - \sum_{j=0}^{i-1} \frac{D_j}{\cos(\theta)}, \frac{D_i}{\cos(\theta)}\right), \tag{12}$$

where $D_j = \frac{D_j^M - D_j^{K'}}{2}$ is an estimate of the jth layer's thickness. This estimate gives reliable results in most cases and can be quickly evaluated.



(a) Ray marching

(b) Input variables. Note that the tangent T and the binormal B are not represented

Figure 10: Illustration of algorithms 1 and 2.

2.3 Algorithm

The implementation of the different ideas exposed before is described by the Algorithms 1 and 2. Single scattering computation uses a ray marching algorithm from the point P to the point M_{max} (see Figure 10(a)). Performed in image space, our method is efficiently implemented on commodity graphics hardware. Some implementation details are described at the end of this section.

Our method consists of three steps: (1) estimation of the reduced intensity L_{ri} that depends on the light attenuation along KM, K being the entry point of the light into the medium, (2) computation of the light scattered at point M and (3) attenuation calculation of the scattered light along PM.
Firstly, we compute the reduced intensity as explained by the Algorithm 2. This latter computes the light ray/surface intersection as well as the distances traversed by light within each layer as seen in Subsection 2.2. The output of this algorithm is the light attenuation along KM, useful for the calculation of the reduced intensity.
Secondly, the single scattering term $Q(M, \omega_{out})$ is computed following Equation 2 and using the phase function as well as the scattering coefficient of the layer containing point M.
Finally, the attenuation term $e^{\int_M^P -\sigma_t(s)ds}$ along the path $\|MP\|$ is computed. As we are using a ray marching algorithm, the attenuation at the next sample point along the viewing ray can be computed using the attenuation at the previous sample point. In this way, for two consecutive sample points M_N and M_{N+1} on the path PM_{max} we have then:

$$\begin{aligned} e^{\int_{M_{N+1}}^P -\sigma_t(s)ds} &= e^{(\int_{M_{N+1}}^{M_N} -\sigma_t(s)ds + \int_{M_N}^P -\sigma_t(s)ds)} \\ &= e^{\int_{M_N}^P -\sigma_t(s)ds} e^{\int_{M_{N+1}}^{M_N} -\sigma_t(s)ds} \end{aligned} \tag{13}$$

In the two algorithms, the scattering terms σ_t^i,σ_s^i and p^i correspond to the extinction coefficient, the scattering coefficient and the phase function of the ith layer, respectively. Given a sample point M, the objective is to determine the layer containing it and the associated values of the scattering terms accordingly. When comparing the depth of point M with the layer depths obtained by a single texture look-up, one can obtain the layer to which belongs M. To perform the texture lookup, PM is projected into the tangent space. This projection is obtained using the projection of vector PM_{max}, (ds, dt), in Algorithm 1.

The functions **FindLayer()** and **LayerThickness()** used in Algorithms 1 and 2 are texture look-up functions giving respectively the number of the layer containing a point (using a depth comparison) and the thickness of the layers for given texture coordinates.

Following these algorithms, the outgoing radiance of each visible point of the projected translucent object is computed in the fragment shader. Some details of the rendering process using the graphics hardware are presented hereafter.

The planar approximation, needed for the estimation of the reduced intensity and presented in Subsection 2.2, is done in a preprocessing step for each vertex of the mesh representing the surface of the considered translucent object. To each vertex V, we apply a surface fitting algorithm that results in a normal NS_V to the approximated plane and a point P_V lying on this plane. Indeed, only NS_V has to be saved since P_V can be recovered using:

$$P_V = V + \|NS_V\| N_V \tag{14}$$

where N_V is the normal at vertex V.

At runtime, for each visible point P of the translucent object, a rasterization step calculates the normal N_P at point P and the approximated plane normal NS_P corresponding to P. This approximated plane normal is obtained by interpolating the normals of the approximated planes precomputed for each vertex of the mesh's triangle containing P. This step also provides the texture coordinates (u_P, v_P) of the point P as well as the tangent and bi-normal vectors at P, say T and B. Then, these data are sent to the fragment shader which computes the single scattering.

Before running the ray marching algorithm, the fragment shader determines the point M_{max} (lying on the viewing ray and within the medium) beyond which scattering is considered as negligible. M_{max} is obtained as:

$$M_{max} = P - \frac{Depth_{max}}{NS_P \cdot \omega_{out}} \omega_{out} \tag{15}$$

NS_P is the normal of the approximated plane associated with point P. As shown in Figure 10, the direction of the normal at point P does not point precisely to the medium depth, in particular when the surface is bumpy. On the contrary, the normal NS_P of the approximated plane can be used to describe the global depth direction without extra computations.
The other parts of our method can be implemented straightforwardly following Algorithms 1 and 2.

Algorithm 1 Subsurface Texture Mapping

Input
See Figure 10(b)
SubsurfaceTexture
$numberofsamples = N$

Initialization
$L(P, \omega_{out}) = 0$
$AttenuationPM = 1.0$
//Index of the current layer: $CurrentLayer = 0$
$M_{max} = P - \frac{Depth_{max}}{NS_{P \cdot \omega_{out}}} \omega_{out}$
$PMstep = \dfrac{PM_{max}}{numberofsamples}$
//Project the vector PM_{max} into the tangent space
$(ds, dt) = (PM_{max}.T, PM_{max}.B)$
$TextureCoordinatesStep = \dfrac{(ds, dt)}{numberofsamples}$
$(u_M, v_M) = (u_P, v_P)$
$depthM = 0$
$depthStep = PMstep.N$

Ray marching algorithm
for $i = 0$ to $numberofsamples$ **do**

//Point M localization:
$CurrentLayer =$ **FindLayer**($depthM$,(u_M, v_M), SubsurfaceTexture)

//Estimate the reduced intensity:
$L_{ri}(M, \omega_{in}) =$ ReducedIntensityComputation(P,M,S,SubsurfaceTexture)
//(see Algorithm 2).

//Estimate the single scattering at point M:
$L(M, \omega_{out}) = \sigma_s^{CurrentLayer} \times L_{ri}(M, \omega_{in}) \times p^{CurrentLayer}(M, \omega_{in}, \omega_{out})$

//Attenuate the scattered radiance along PM and add the contribution of point M:

$L(P, \omega_{out}) += L(M, \omega_{out}) \times AttenuationPM \times PMstep$

//Move to the next sample M and compute its texture coordinates and attenuation:
$(u_M, v_M) += TextureCoordinatesStep$
$depthM += depthStep$
$M += PMStep$
$AttenuationPM \times = e^{-\sigma_t^{CurrentLayer} \|PMstep\|}$
end for

return $L(P, \omega_{out})$

Algorithm 2 Reduced intensity computation

Input
See Figure 10(b)
i : sample number
currentLayer
Point M
Texture coordinates (u_M, v_M)

Initialization
$L_{ri}(M, \omega_{in}) = 0$
$AttenuationKM = 1.0$

//Obtain the planar surface approximation:
$\omega_{in} = normalize(MS)$
$N_M = \dfrac{(i.N_P + (numberofsamples - i).NS_P)}{(numberofsamples)}$
$P_M = \dfrac{(i.P_{M_{max}} + (numberofsamples - i).P)}{(numberofsamples)}$
$\cos(\theta) = N_M \cdot \omega_{in}$

//Calculate the distance $\|KM\|$:
$\|KM\| = \frac{\|P_M M.N_M\|}{\cos(\theta)}$

//Compute the thickness of the layers at point M:
$(D_1^M, D_2^M, D_3^M, D_4^M)$ =**LayerThickness**(M,(u_M, v_M), SubsurfaceTexture)

//Compute the thickness of the layers at point K:
$K = M + \|KM\|\omega_{in}$
$(u_K, v_K) = (u_P, v_P) + (PK.T, PK.B)$
$(D_1^K, D_2^K, D_3^K, D_4^K)$ =**LayerThickness**(K,(u_K, v_K), SubsurfaceTexture)

//Average thicknesses:
$(D_1, D_2, D_3, D_4) = \frac{1}{2}((D_1^M, D_2^M, D_3^M, D_4^M) + (D_1^K, D_2^K, D_3^K, D_4^K))$

for $j = 0$ to 4 **do**
  $d_j = \min\left(\|KM\| - \sum_{n=0}^{j-1} \frac{D_n}{\cos(\theta)}, \frac{D_j}{\cos(\theta)}\right)$
end for

//Calculate the attenuation along KM:
$AttenuationKM = e^{-\sigma_t^1 d_1}.e^{-\sigma_t^2 d_2}.e^{-\sigma_t^3 d_3}.e^{-\sigma_t^4 d_4}$

//Estimate the reduced intensity:
$Lri(M, \omega_{in}) = L_i(K, \omega_{in}).AttenuationKM$

return $Lri(M, \omega_{in})$

3 Results

We have implemented our subsurface scattering method using fragment programs written in nVidia Cg and experimented with several translucent objects. All the RGBα subsurface textures used in this paper have a resolution of 800×600. The results given in this paper have been obtained on a 3.8 GHz PC with 1024 MB of memory and a GeForce 7800GTX with 256 MB of memory.

For the sake of more realism, we have added specular (glossy) and diffuse reflection components to the fish model to give the fish's skin a scaly and shiny appearance. Note that the glossiness emphasizes the translucent appearance of our objects (see [16] for more details). The data used for the fish model are given by Figures 12 and 11. Even though the geometry of the fish model is coarse, as shown in Figure 11, our method allows to exhibit finer details as shown in Figure 13. This is due to our fine subsurface relief textures.

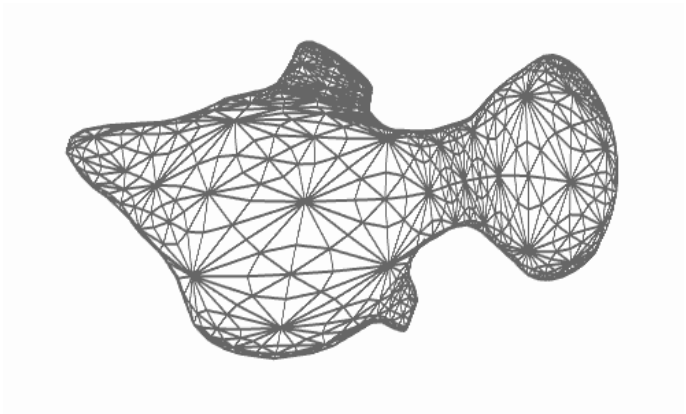
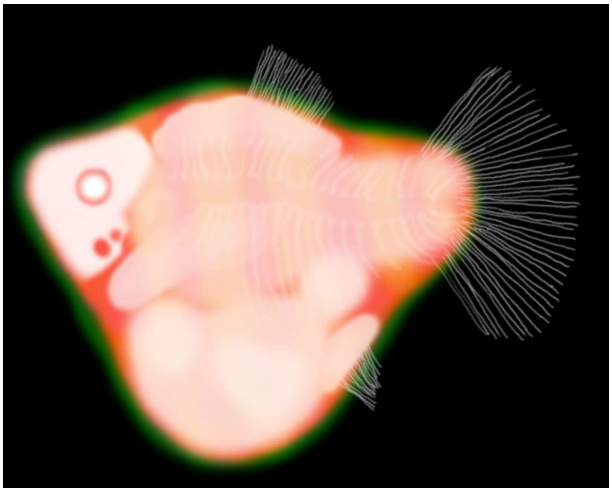
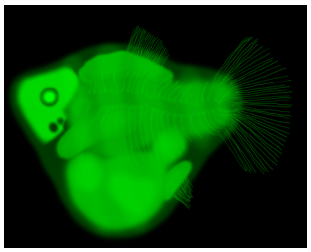


Figure 11: Our method does not require densely tesselated objects, hence reducing the cost of vertex processing.

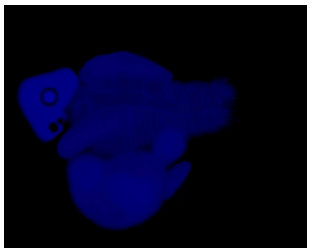
An RGBα subsurface texture encodes the thickness of the layers. One channel of the texture refers to one layer. Note that each channel is created as a single grayscale image describing the distance from the layer to the surface of the translucent object. A subsurface texture is easily created as follows. First, we use an image processing software to create grayscale images, where the intensity of a pixel is inversely proportional to the thickness. In other words, a high intensity corresponds to a low thickness. Next, we map this subsurface texture on the meshed model using a 3D modeler. Figure 12(a) represents the compositing of the channels corresponding to three layers. The channel α is not used for the fish model. the red channel is almost constant and describes a first layer with a constant thickness. The next layers presented in Figures 12(b) and 12(c) correspond to the green and blue channels.



(a) RGB Subsurface texture map



(b) Green channel



(c) Blue channel

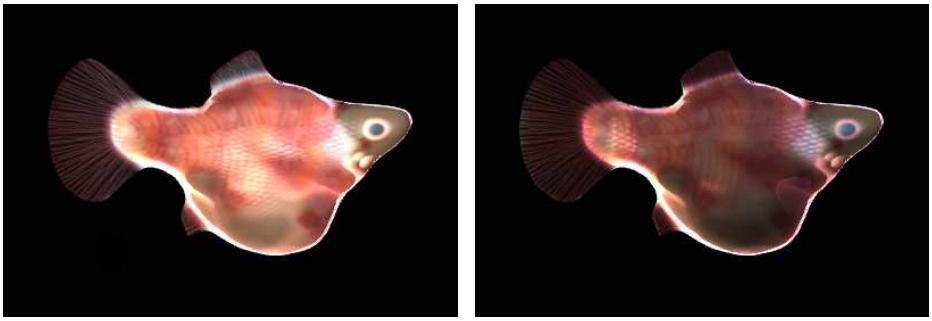
Figure 12: Subsurface texture map of the fish model.

Table 1 gives the scattering coefficients used for the fish model. We have used the Schlick phase function $p(\theta) = \frac{1-g^2}{4\pi(1+g\cos(\theta))^2}$ where θ is the angle between ω_{in} and ω_{out} and g is called the average cosine describing the degree of anisotropy of the phase function. Note that the parameter g of the phase function as well as the extinction and scattering coefficients are not uniform.

	$\sigma_s(mm^{-1})$			$\sigma_t(mm^{-1})$			
Layers	R	G	B	R	G	B	g
Dermis	2.0	1.0	1.0	2.6	1.6	1.6	0.25
Blood	6.0	3.0	2.0	6.6	3.6	2.6	0.40
Organs	1.0	1.0	2.0	1.5	1.5	2.5	0.80

Table 1: Scattering coefficients used for the fish model.

Figure 13 shows images provided by our method and corresponding to two different light configurations. Three point light sources are used, one behind, one beneath and one in front of the object. One can easily and accurately distinguish the internal structure of the fish model.

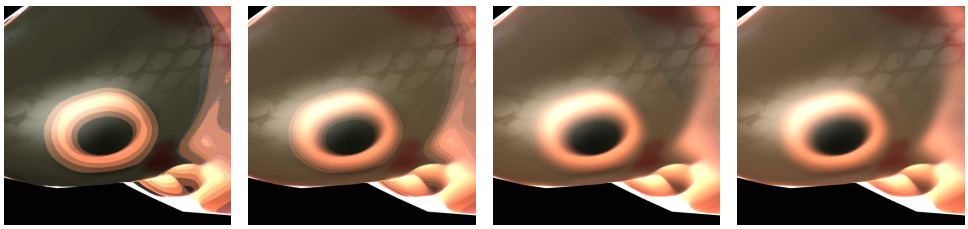


(a) (b)

Figure 13: Fish model under two different lighting configurations.

The number of light sources increases the rendering time but does not represent the main bottleneck of the algorithm which is the ray marching step. The different pictures presented in this paper have been computed with 100 sampling points along the viewing ray. Table 2 gives a comparison of the rendering times for different models with different sample numbers (see Figure 14). All the models have been rendered at a resolution of 800×600 pixels.

Images rendered with only 10 samples suffer from aliasing artifacts. This aliasing problem depends on the global depth of the layers. Nevertheless, artifacts disappear in most cases for ray marching with 50 samples or more. The images created with 50 samples and 100 samples are almost similar (some small differences can appear at grazing angles).

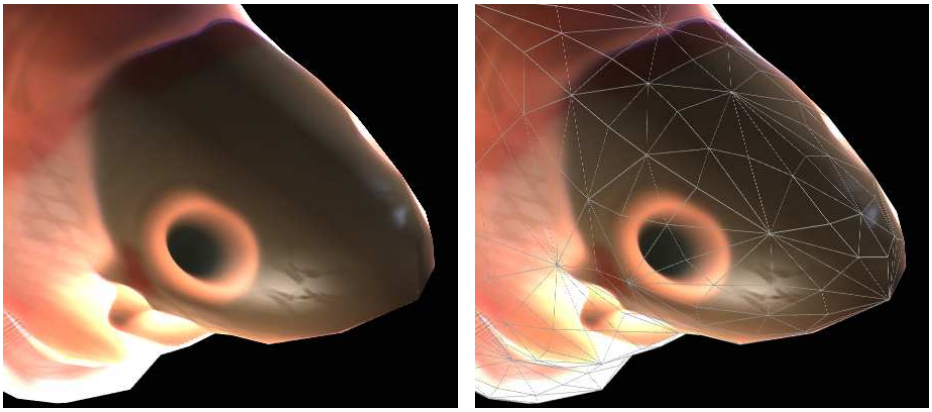


(a) 10 samples (b) 20 samples (c) 50 samples (d) 100 samples

Figure 14: Fish model rendered with different number of samples.

number of samples	fish model	gecko model
10	126 fps	120 fps
50	51 fps	33 fps
100	20 fps	16 fps

Table 2: Comparison of the rendering time for different sample numbers



(a) (b)

Figure 15: Close-up of the fish model, with subsurface mapping and geometry.

Figure 15 shows a closer view of the fish model. Notice the volumetric appearance of the interior, even with a coarse mesh. The specular effect gives the viewer a hint of the position of the model's surface, giving then a good idea of the inner distances.

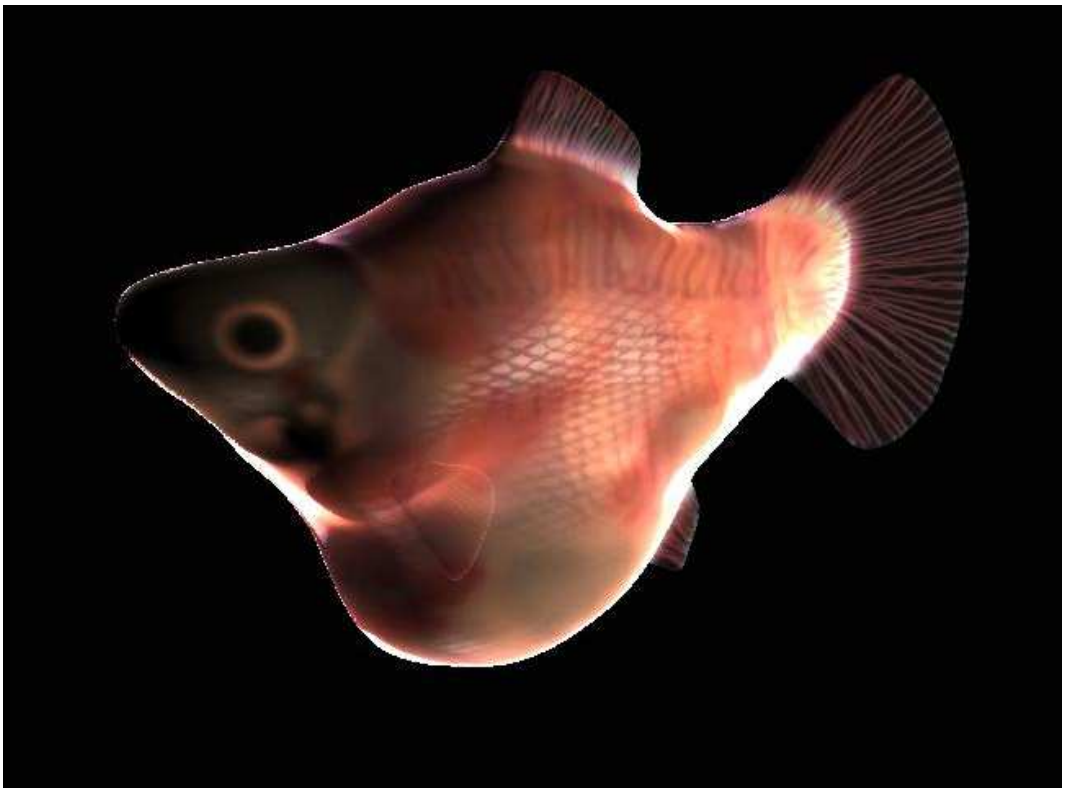
Our rendering algorithm provides the objects with a translucent appearance with new effects due to the variability of the layer thicknesses. Indeed, the eyes and the abdomen of the fish appear darker because of their proximity to the surface. The bumps on the gecko skin are more or less visible depending on the position of the viewpoint. These volumetric effects are commonly observed in our daily life and can be easily interpreted by an observer.

The Figures 17 and 18 give some results obtained using two other objects: a gecko model and a human head model. For the gecko model, subsurface texture mapping is used to describe its translucent bumpy skin. Since the thickness of the skin is small, the relief appearance is less visible compared to the fish model. As for the human head model, the skin is described using a veins subsurface texture comparable to the one presented in Figure 4(a). The skin of these two models is composed of three layers.

4 Conclusion and Future Work

We have proposed a method for modeling and rendering subsurface scattering in real-time using programmable GPUs. Our method allows an intuitive description of complex materials made up of multiple layers of variable thickness, offering then new effects often observable in our daily life. Our layer description is simple since it uses classical texturing already available in commodity graphics cards. With this description it is possible to represent multilayered translucent objects at a low cost (low memory storage) using the different channels of a texture. Computing subsurface scattering requires the determination of the points at which a light ray enters the translucent object. We have proposed a fast method based on locally planar approximation of object's surface to approximate these points.

Our method computes single scattering for objects lit by point light sources. Our next goal is to compute single scattering under environment lighting condition. The use of an environment map would lead to complex estimates of the reduced intensity. Another goal is to tackle the problem of multiple scattering computation. Our current algorithm can also be improved following the ideas proposed in [13] describing non-height-field surface details with multiple layers.



(a)

Figure 16: Fish model : 1396 vertices, 2788 triangles

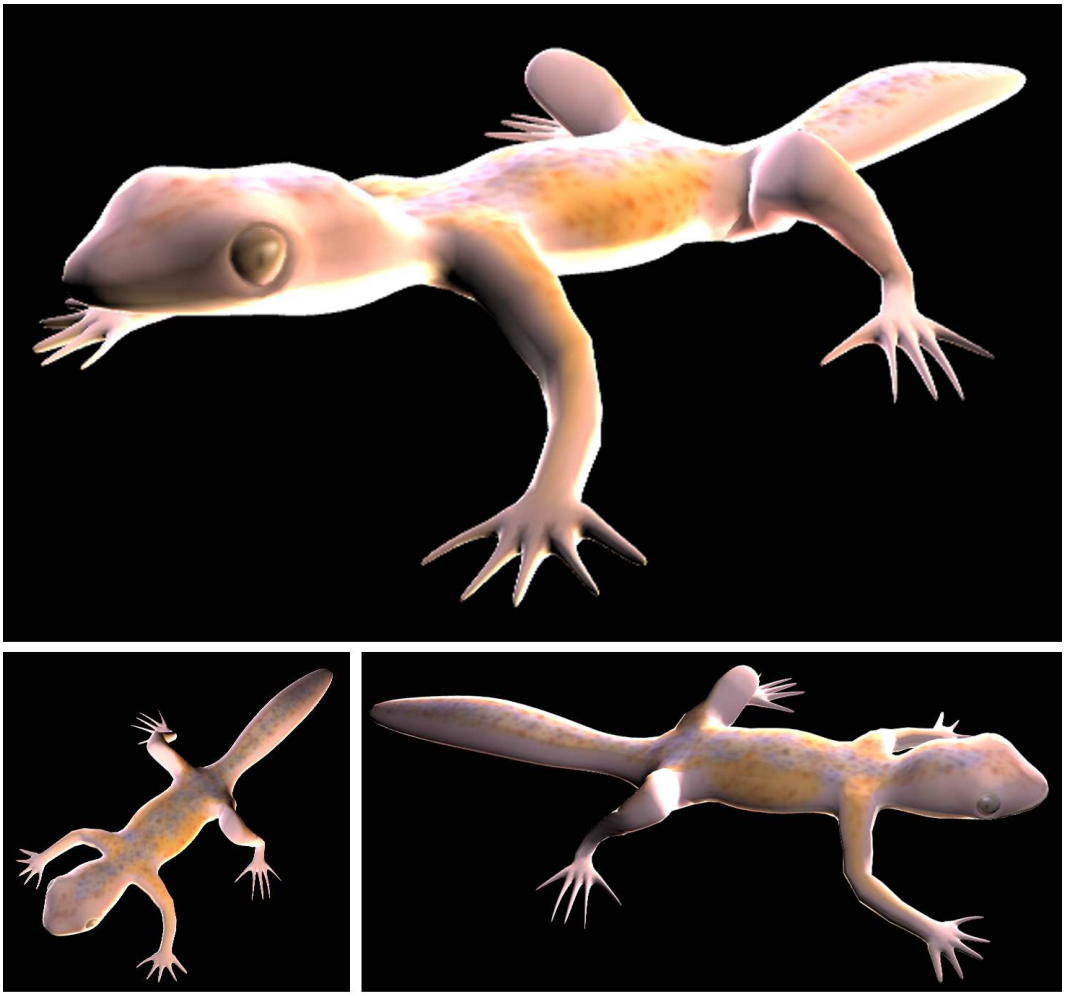


Figure 17: Rendering of a gecko model : 9774 vertices, 10944

Figure 18: Rendering of a human head model : 36572 vertices, 73088 triangles.

References

[1] Fabio Policarpo, Manuel M. Oliveira, and Joao L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, New York, NY, USA, 2005. ACM Press.

[2] S. Chandrasekhar. *Radiative Transfer*. Clarendon Press, Oxford, reprinted Dover Pub., 1960, 1950.

[3] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1993. ACM Press.

[4] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320, New York, NY, USA, 1998. ACM Press.

[5] Jos Stam. An illumination model for a skin layer bounded by rough surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 39–52, London, UK, 2001. Springer-Verlag.

[6] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, 2001. ACM Press.

[7] Rui Wang, John Tran, and David Luebke. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph.*, 24(3):1202–1207, 2005.

[8] Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and Hans-Peter Seidel. Efficient rendering of local subsurface scattering. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 51, Washington, DC, USA, 2003. IEEE Computer Society.

[9] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph.*, 24(3):1032–1039, 2005.

[10] Kyle Hegeman, Michael Ashikhmin, and Simon Premoze. A lighting model for general participating media. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 117–124, New York, NY, USA, 2005. ACM Press.

[11] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

[12] Joe Kniss, Simon Premoze, Charles Hansen, Peter Shirley, and Allen McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, 2003.

[13] Fabio Policarpo and Manuel M. Oliveira. Relief mapping of non-height-field surface details. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 55–62, New York, NY, USA, 2006. ACM Press.

[14] Mark Peercy, John Airey, and Brian Cabral. Efficient bump mapping hardware. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 303–306, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[15] Laszlo Szirmay-Kalos, Barnabas Aszodi, Istvan Lazanyi, and Matyas Premecz. Approximate Ray-Tracing on the GPU with Distance Impostors. *Computer Graphics Forum*, 24(3):695–704, 2005.

[16] Roland W. Fleming, Henrik Wann Jensen, and Heinrich H Bülthoff. Perceiving translucent materials. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 127–134, New York, NY, USA, 2004. ACM Press.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399