Eurographics Symposium on Rendering 2003
Per Christensen and Daniel Cohen-Or (Editors)

Translucent Shadow Maps

Carsten Dachsbacher† and Marc Stamminger‡

Computer Graphics Group, Department of Computer Science, University of Erlangen-Nuremberg, Germany

Abstract

Shadow maps are a very efficient means to add shadows to arbitrary scenes. In this paper, we introduce Translucent Shadow Maps, an extension to shadow maps which allows very efficient rendering of sub-surface scattering. Translucent Shadow Maps contain depth and incident light information. Sub-surface scattering is computed on-the-fly during rendering by filtering the shadow map neighborhood. This filtering is done efficiently using a hierarchical approach. We describe optimizations for an implementation of Translucent Shadow Maps on contemporary graphics hardware, that can render complex translucent objects with varying light and material properties in real-time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

Shadow maps [11] are widely used for adding shadow to arbitrary scenes in real-time computer graphics. Each texel of the shadow map texture – rendered with a virtual camera placed at the lights' position – stores the distance of the closest surface to the light source. This information is used in the rendering pass of the final image to determine whether a point lies in shadow by comparing its light source with the shadow map distance, which can be achieved on modern graphics hardware as a simple texture lookup and is thus very cheap. But also in offline rendering systems like RenderMan shadow maps are widely used due to their efficiency and generality.[8]

In this paper, we extend the binary shadow map lookup to a shadow map filter that implements sub-surface scattering. As sub-surface scattering all light reflection processes are summarized that happen underneath an optical boundary surface. By this, the lighting on soft objects, in particular shadows, is blurred, and thin object parts become translucent. Bidirectional reflection distribution functions (BRDFs) cannot model this behavior, their definition assumes that the reflection point is the point of incidence. Thus, many computer generated objects look too harsh and lack a natural softness in their appearance. Researchers early recognized that for realistic rendering of natural objects sub-surface scattering is essential. However, the correct simulation is challenging and requires complex computations. Jensen et al.[6, 4] described an approximation that breaks the computation down to a radiosity-like 2D-integral over the object surface, which allows faster, but not yet interactive rendering.

The idea of Translucent Shadow Maps is to extend a shadow map so that all information to compute translucency is available. A pixel in the Translucent Shadow Map stores not only the depth and thus 3D-position of the sample, but also the irradiance entering the object at that position and the surface normal. The translucency integral from Jensen[6] can then be computed as a filter on the color values, where the filter weights depend on the corresponding depth and normal values. By using a hierarchical filtering technique based on mip-maps, this filtering can be computed in real time in a fragment program on contemporary graphics hardware, where our implementation is restricted to parallel light sources.

2. Previous Work

The light scattering in translucent materials can be treated like global illumination in other, usually less dense participating media, e.g. by Monte-Carlo or finite element based global illumination algorithms [1, 5, 10]. In Hanrahan et

† e-mail: dachsbacher@cs.fau.de
‡ e-mail: stamminger@cs.fau.de

al.[3], a model has been presented that adds sub-surface scattering effects for the special case of layered surfaces. Jensen at al.[6] presented a more general model for light reflection on solid translucent objects, based on BSSRDFs (bidirectional surface scattering distribution function), which are a generalization of BRDFs with possibly distinct light incidence and exitance points. The higher dimensionality of BSSRDFs compared to BRDFs results in longer computation times, but the method is still much faster than a full Monte-Carlo simulation. A more rapid, hierarchical evaluation technique for this model was introduced by Jensen et al.[4]. The separation of incident light computation and BSSRDF integration as well as a hierarchical integration technique dramatically reduce computation time to a few seconds. Although this strategy is similar to the method presented in this paper, we accomplish this concept in a completely different way. By using a shadow map as intermediate irradiance representation, we are restricted to illumination from point or parallel light sources, however, the computation of the irradiance and the hierarchical integration are even more simplified and can be completely passed to the graphics processing unit (GPU).

Another accelerated computation technique for Jensen's BSSRDF model has been presented by Lensch et al.[7]. In this paper, the authors use a mixture of radiosity-like finite element computations and texture filtering to evaluate the BSSRDF integral. The method requires significant precomputation for a texture atlas, form factors, and filter kernels. After this, the models can be rendered interactively with moving light sources at several frames per second.

3. Translucent Shadow Maps

In the original sub-surface model of Jensen[6] sub-surface scattering is described as a sum of a single and a multiple scattering term. As in Lensch et al.[7] we restrict ourselves to multiple scattering effects. This is arguable for materials with high albedo – like marble, milk or skin – because single scattering has only small contribution to the re-emitted radiance in these cases[6]. Note that in the case of multiple scattering any relation between directions of incident light and exitance is lost, which makes it easier to handle.

The sub-surface scattering in highly scattering material can be separated into three phases. First, the light incident at a surface point x_{in} is scattered into the material according to the Fresnel term F_t. For an irradiance impulse $I(\omega_{in})$ from a point or parallel light source this is simply:

$$E(x_{in}) = F_t(\eta, \omega_{in})|N(x_{in}) \cdot \omega_{in}|I(\omega_{in}), \qquad (1)$$

where η is the optical density of the material. The Fresnel term F_t can be well approximated as proposed by Schlick[9].

Second, light diffuses through the material. This diffusion process is approximated by the diffuse sub-surface reflectance function $R_d(x_{in}, x_{out})$ with $x_{in}, x_{out} \in S$, which can be compared with the geometric term in radiosity. R_d describes the transport of incident light at x_{in} through the object to x_{out}. This 4D-function does not only depend on $||x_{out} - x_{in}||$, but also on the angle between $x_{out} - x_{in}$ and the surface normal in x_{in}:

$$B(x_{out}) = \int_S E(x_{in}) R_d(x_{in}, x_{out}) dx_{in} \qquad (2)$$

In this integral it is assumed that the path from x_{in} to x_{out} is completely within the object, which is in general the case only for convex objects. So errors occur for concave objects, however, in practice this kind of error is often visually not important. Finally the light leaves the object, again weighted by F_t:

$$L(x_{out}, \omega_{out}) = \frac{1}{\pi} F_t(\eta, \omega_{out}) B(x_{out}) \qquad (3)$$

The simulation path is also depicted in Fig. 1.

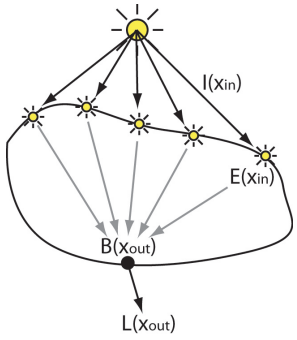

Figure 1: *Computing translucency by integration.*

With Translucent Shadow Maps (TSMs), the simulation is separated (Fig. 2). The first phase (Equ. 1) is computed during the generation of the TSM. Additionally to depth, a TSM stores irradiance $E(x_{in})$ and the surface normal $N(x_{in})$ with every pixel (see Fig. 3). The surface normal is required later to compute R_d. Note, that $E(x_{in})$ is wavelength dependent, thus we store a red, green and blue color component which allows us to use arbitrary textured surfaces.

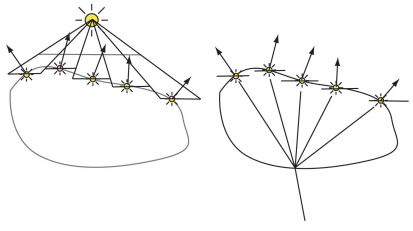

Figure 2: *The Translucent Shadow Map stores irradiance samples (left). The radiance leaving the object is then computed by filtering these irradiance samples (right).*

Having this information, the integral in Equ. 2 can be computed during rendering of the user's view as a filter

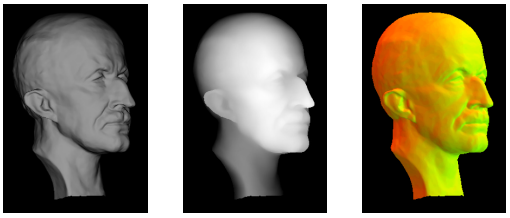

Figure 3: *The Translucent Shadow Map contains irradiance, depth, and surface normal.*

of the TSM color values, with weights given by R_d. For the evaluation of R_d, x_{in} is computed from the TSM depth value and the required normal $N(x_{in})$ is taken from the TSM (Fig. 4).

The filter weights heavily depend on Δz. For large Δz, the weigths are small and decrease only slowly with $(\Delta x, \Delta y)$. If Δz is small, the central weights are large and decrease quickly. Generally, the filter must be large enough to cover all regions where R_d is significantly large – for natural materials this can be in the order of centimeters. Because large filters are expensive, we use a hierarchical filter based on a mip-map of the TSM. In our filter sample pattern of 21 sample positions, shown in Fig. 4, sampling density decreases with the distance to the origin. The outer samples represent a larger filter area, thus their irradiance, depth and normal values area are taken at coarser mip-map levels (see Fig. 4). The pattern reflects the fact that R_d decreases rapidly with the sample distance. The pattern is based on heuristics; as shown above, the filter value distribution heavily depends on Δz, so there is no globally optimal pattern.

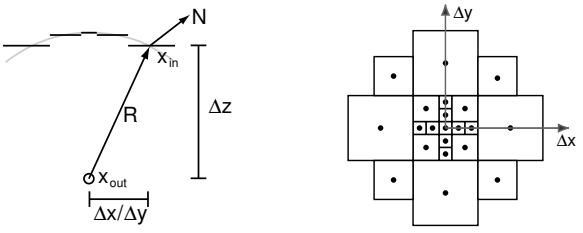

Figure 4: *Filtering the TSM. Left: computing R, Δx and Δy from the TSM. Right: Sampling pattern.*

The TSM of an object also contains background pixels, i.e. pixels not showing the object. In order to avoid that invalid values from these pixels diffuse into the result through the mip-map filter, we add an alpha image. For background pixels, this alpha image, irradiance, and normal are set to zero, otherwise pixels get an alpha of one. After mip-mapping, we weight all mip-mapped values by $1/\alpha$, resulting in an average value for non-background pixels only.

As in Lensch et al.[7] we further optimize the computation by splitting up the evaluation of the sub-surface scattering into two independent steps: the local response and the global response (see Fig. 6). The latter is evaluated as described above and represents sub-surface scattering at larger distances. This can be either done per fragment or per vertex (and interpolated across polygons during rendering) since this appears as a smooth function. The local response, which describes the short distance sub-surface light transport, could be evaluated exactly the same way, but we apply simplifications to speed up the rendering: for the calculation of the transmitted light in this case we ignore the difference of the depth values associated with the sample texel and x_{out}. This results in a very simple filter with a fixed sampling scheme. All texels in the neighborhood of x_{out} (in light space) with a depth value in a certain proximity to the depth value of x_{out} are weighted and summed up. Samples outside this depth proximity are handled by the global response.

reduced scattering coefficient	σ_s'
absorption coefficient	σ_a
relative refraction index	η
Fresnel transmittance factor	$F_t(\eta,\omega)$
in- and out-scattering location	x_{in}, x_{out}
in- and out-scattering direction	ω_{in}, ω_{out}
surface normal at x_{in}	N_{in}

$$
\begin{aligned}
R_d(x_{in},x_{out}) &= \frac{\alpha'}{4\pi}\left[z_r(\sigma_{tr}d_r+1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3}\right. \\
&\quad \left. + z_v(\sigma_{tr}d_v+1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right] \\
z_r &= 1/\sigma_t' \quad z_v = z_r + 4AD \\
d_r &= ||x_r - x_{out}||, \text{with} x_r = x_{in} - z_r \cdot N_{in} \\
d_v &= ||x_v - x_{out}||, \text{with} x_v = x_{in} + z_v \cdot N_{in} \\
A &= \frac{1+F_{dr}}{1-F_{dr}} \quad D = \frac{1}{3\sigma_t'} \\
F_{dr} &= -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta \\
\sigma_{tr} &= \sqrt{3\sigma_a\sigma_t'} \quad \sigma_t' = \sigma_a + \sigma_s' \quad \alpha' = \frac{\sigma_s'}{\sigma_t'}
\end{aligned}
$$

Figure 5: *Quantities and equations describing the BSSRDF diffusion term.*

4. Implementation

For the implementation of the Translucent Shadow Maps we used OpenGL and the extensions introduced with the DirectX9 generation of graphic processing units, namely programmable vertex and fragment processing.

In the first rendering pass of each frame the Translucent Shadow Map is generated. We use two simultaneous render

targets: in the first one we store the amount of light penetrating the materials boundary (in the red, green and blue component). This fraction is described by the Fresnel term which we approximate as proposed by Schlick[9]. The second render target contains the distance from the light source to the visible surfaces, thus the light space depth-buffer, which corresponds to a standard shadow map. Furthermore the 2D-projection of the (normalized) surface normals onto the plane orthogonal to the lights' direction is stored as color. Since visible surfaces (in the Translucent Shadow Map) are oriented towards the light direction, this information suffices to reconstruct the surface normal.

In the subsequent rendering passes we evaluate the local and global response for each surface point x_{out} rendered in the camera view. The point's coordinates are transformed into light space to determine its texture coordinates $(u,v)^T$ in the Translucent Shadow Map and its distance d to the light source.

The local response is determined by an image filter with a size of 7×7 texels where the neighborhood of $(u,v)^T$ serves as input. The incoming radiance at the neighboring texels only contributes to the outgoing radiance, if the texel's depth values lies within a certain proximity to d. If the stored depth value at $(u,v)^T$ is significantly smaller than d, then the currently processed surface point is shadowed by the surface part represented by this texel. All contributing texels are summed up using constant, precomputed weights, ignoring depth difference. In the rendering passes for the local response we also calculate the local illumination depending on the Fresnel term for reflected light. All this work is done by a vertex and a fragment program. The number of required rendering passes depends on the deployed graphics hardware.

For the global response the incoming light from the hemisphere (pointing inward the object) of x_{out} needs to be calculated. The sub-surface light transport from a surface point x_{in} to x_{out} is evaluated according to the formulae in Fig. 5. It depends on the vector $R = x_{in} - x_{out}$ and the normal $N = (n_x, n_y, n_z)^T$ at x_{in}. Although this function could be computed per fragment, we use a pre-computed, quantized version, which can be quickly evaluated as a texture lookup. Considering the fact that only textures up to a dimension of three are provided by graphic hardware, we need to reduce the five input dimensions (R is 3D, N is 2D) to three.

The first dimension reduction – and so the first texture coordinate S – is obtained by quantizing the surface normal at x_{in} with: $\frac{\lfloor w(\frac{1}{2}n_x+\frac{1}{2})\rfloor + \mathrm{frac}(\frac{1}{2}n_y+\frac{1}{2})}{w}$ with $\mathrm{frac}(x) = x - \lfloor x \rfloor$. The variable w represents the half resolution (in texels) of the texture in the S dimension. It is necessary to defer the quantizing of the normal to this stage, because we want to obtain mip-mapped normals from the Translucent Shadow Map.

As second texture coordinate T we use the z-component of R which is the difference of the depth values of x_{in} and x_{out}. Note that small values of T are treated in the local response and are excluded in this context by adapting the lookup texture appropriately.

The third texture coordinate R enumerates the 21 samples from our hierarchical sampling scheme. Each sample has constant offset relative to $(u,v)^T$ and a particular mip-map bias value for the TSM lookup. If our light source is parallel, (r_x, r_y) is constant for every sample and can thus be assumed to be constant within each texture slice with $R = const$. This quantization trick is not possible with point lights, in this case more intricate computations become necessary.

All this ends up in a three dimensional texture consisting of 21 slices. Each slice contains for a particular sample of our sampling scheme the filter value $R_d(x_{in}, x_{out})$ for all possible quantized normals (first slice dimension) and r_z (second slice dimension).

The texture slices of the material property 3D texture contain only low frequency color gradients. Therefore, a low resolution 3D texture of the size $64 \times 64 \times 32$ for each type of material proved to be sufficient. Note that the third dimension of the texture has an extent of 32, which is the next power of two to 21. It is generated in a pre-processing step during initialization in approximately 150ms on a Pentium 4 processor with 2.4GHz, so material properties can be changed interactively with almost no delay.

Model	FPS	#Vertices	#Triangles
Simple Cube	99.7	8	12
Triceratops	80.0	2832	5660
Horse-10k	50.2	5068	10132
Horse-100k	10.0	48485	96966
Bunny-10k	30.5	5187	10370
Bunny-70k	11.9	35947	69451
Bird-54k	15.3	27002	54000
Bird-100k	9.6	48668	97332
Max Planck	14.2	25044	49999
Max Planck II	5.7	100086	199996

Table 1: *This table lists the frames per second at a resolution of* 512×512 *and the number of vertices and triangles for models we tested with our implementation.*

5. Results

Table 1 shows the performance of our implementation of the Translucent Shadow Map method using a resolution of 512×512 for the shadow map and the camera view. The timings were collected on an Intel Pentium 4 Processor with 2.4GHz and an ATI Radeon 9700 graphics card. Note that the object and the light source can be moved interactively and freely by the user. Complex objects can be rendered at interactive frame rates, simpler objects in real-time. In the

first case the rendering speed is limited by the vertex processing of the GPU because we need multiple render-passes, due to the limited number of instructions and texture lookups in the fragment processing stage. For simpler objects the frame rate is bounded by the fragment processing speed. With newer GPUs like the Radeon 9800 or GeForce FX we will be able to increase rendering speed by saving render passes. The rendering quality will benefit from this aspect, too: intermediate results need no longer be accumulated in the precision limited frame buffer, thus the calculation gets more accurate, due to higher internal precision of the GPUs. With the Radeon 9700 we need 7 render passes for the local and 7 for the global response.

Fig. 7 shows the bunny model under different lighting conditions. Fig. 8 and 9 show models with varying material density. In our examples we used the material properties given in Jensen[6].

6. Discussion and Future Work

We extended the concept of shadow maps by storing the surface normal and irradiance for every texel. The resulting Translucent Shadow Maps allow rendering of translucent objects on contemporary hardware in real-time and performance and accuracy of our approach benefit from the newest generation of GPUs.

The low-frequency global response could be calculated per vertex, as proposed by Lensch et al. [7]. On nowadays GPUs this could be done with our method, but the results have to be read back from the frame-buffer stalling the graphic pipeline. With future GPUs, when textures can be sampled at the vertex processing stage, this will be possible without performance penalty.

A Translucent Shadow Map captures – typical for shadow maps – only the first visible surface seen by the light space camera. They do not contain all information necessary for rendering sub-surface scattering effects for concave or hollowed objects. In the future we would like to solve this problem by using a depth peeling technique as used by Everitt et al.[2], which reveals all necessary depth information.

References

1. Philippe Blasi, Bertrand LeSaëc, and Christophe Schlick. An importance driven monte-carlo solution to the global illumination problem. In *Fifth Eurographics Workshop on Rendering*, pages 173–183, June 1994. 1

2. Cass Everitt. Order-independent transparency. *http://developer.nvidia.com/view.asp?IO=order_independent_transparency*, 2001. 5

3. Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 165–174, August 1993. 2

4. Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics*, 21(3):576–581, July 2002. 1, 2

5. Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 311–320, July 1998. 1

6. Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 511–518, August 2001. 1, 2, 5

7. Hendrik P. A. Lensch, Michael Goesele, Philippe Bekaert, Jan Kautz, Marcus A. Magnor, Jochen Lang, and Hans-Peter Seidel. Interactive rendering of translucent objects. In *Proc. Pacific Graphics 2002*, pages 214–224, 2002. 2, 3, 5

8. W. Reeves, D. Salesin, and R. Cooke. Rendering antialiased shadows with depth maps. In *Proceedings of ACM SIGGRAPH 1987*, Computer Graphics Proceedings, Annual Conference Series, pages 283–291, August 1987. 1

9. Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13(3):233–246, 1994. 2, 4

10. François X. Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, June 1994. 1

11. Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, volume 12, pages 270–274, August 1978. 1

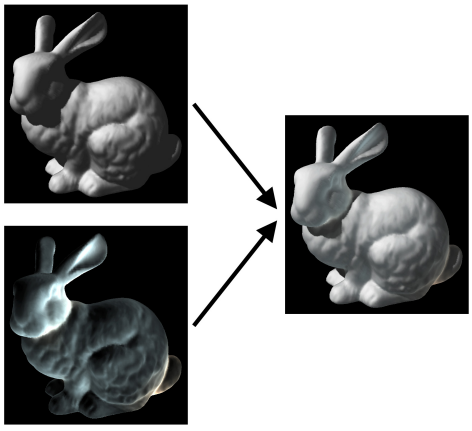

Figure 6: *The sub-surface scattering is evaluated in two steps: the local response and the global response (displayed brightened in the lower left image).*

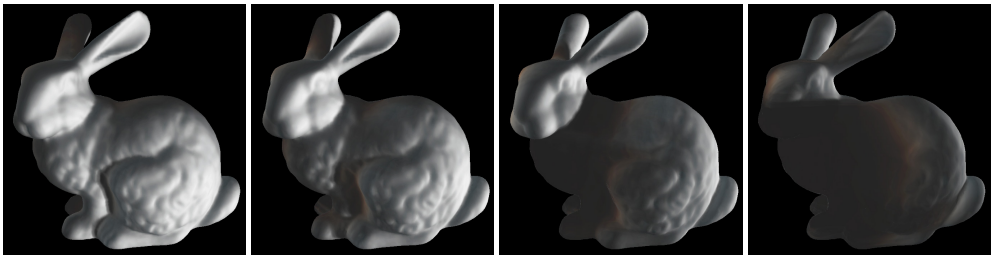

Figure 7: *The bunny model rendered with sub-surface scattering illuminated by a rotating parallel light source.*

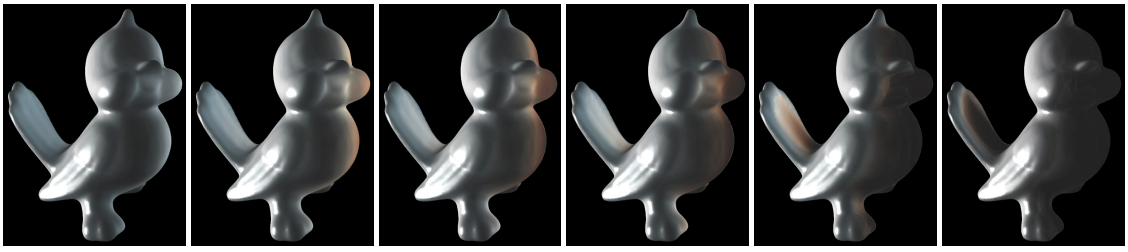

Figure 8: *This model is illuminated by parallel light from the left. From left to right the material density increases.*

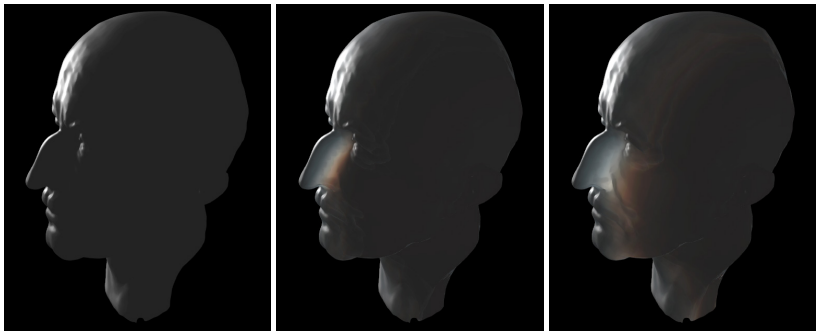

Figure 9: *The Max Planck model is back-lit with varying material density.*