Real-Time Realistic Skin Translucency

Jorge Jimenez ■ *Universidad de Zaragoza*

David Whelan ■ *Trinity College Dublin*

Veronica Sundstedt ■ *Blekinge Institute of Technology*

Diego Gutierrez ■ *Universidad de Zaragoza*

A new algorithm renders real-time, photorealistic skin by simulating both reflectance and transmittance of light through a multilayered skin model. Despite this model's simplicity, it reproduces the look of images rendered with techniques such as photon mapping or diffusion approximation.

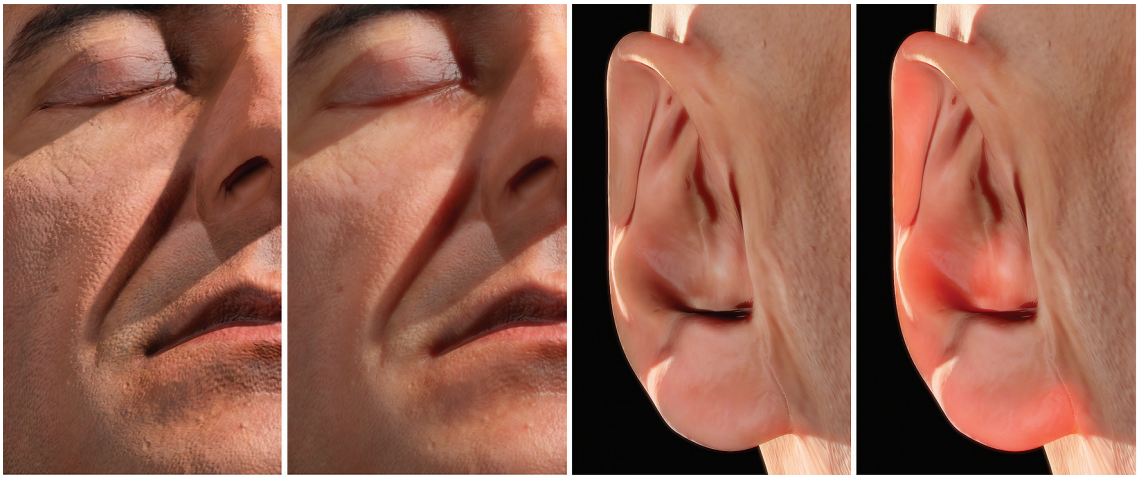
Many materials possess a degree of translucency. Light scatters within translucent objects (such as tree leaves, paper, or candles) before leaving the object at a certain distance from the incidence point. This process is called *subsurface scattering* (SSS). Simulation of SSS in computer graphics is challenging. The rendering process must correctly simulate the light transport beneath an object's surface to accurately capture its appearance (see Figure 1). Figure 2 shows how SSS affects real-world objects.

Human skin is particularly interesting because it consists of multiple translucent layers that scatter light according to their specific composition.[2] This provides the characteristic reddish look to which our visual system seems to be particularly well tuned. Slight simulation errors will be more noticeable in skin than in, say, a wax candle. Correctly depicting human skin is important in fields such as cinematography and games. Whereas the former can count on the luxury of offline rendering, the latter imposes real-time constraints that make the problem much harder. The main challenge is to compute a real-time, perceptually plausible approximation of the complex SSS effects. It should also be easy to implement so that it integrates well with existing pipelines.

Several real-time algorithms for simulating skin exist (for more information, see the "Related Work in Subsurface-Scattering Simulation" sidebar).[3–6] Their common key insight is that SSS mainly amounts to blurring of high-frequency details, which these algorithms perform in texture space. Although the results can be realistic, the algorithms don't scale well; more objects mean more textures to process, so performance quickly decays. This is especially problematic in computer games, in which many characters can appear on-screen simultaneously and real-time performance is needed. We believe this is a main issue keeping game programmers from rendering truly realistic human skin. However, the commonly adopted alternative is to simply ignore SSS, thus decreasing the skin's realism. Additionally, real-time rendering in a computer game context can become much more difficult, with issues such as the background geometry, depth-of-field simulation, or motion blur imposing additional time penalties. In this field, great efforts are spent on obtaining further performance boosts (either in processing or memory usage). This lets us spend the saved resources on other effects, such as higher-resolution textures and increased geometry complexity.

To develop a practical skin-rendering model and thus solve the scalability issues that arise in multicharacter scenarios, we proposed an algorithm that translated the simulation of scattering effects from texture space to screen space (see Figure 3).[1] This algorithm therefore reduced the problem of simulating translucency to a postprocess, with the added advantage of easy adaptability to any graphics en-

 Published by the IEEE Computer Society



(a) (b) (c) (d)

Figure 1. A comparison between (a) ignoring subsurface scattering and (b) accounting for it. The skin's reflectance component is softened from being scattered within the skin. In addition, the figure compares (c) raw screen-space diffusion[1] and (d) screen-space diffusion with transmittance simulation, calculated using the algorithm proposed in this article. Light travels through thin parts of the skin, which the transmittance component accounts for.

gine. The main consequence is that we have less information to work with in screen space, as opposed to algorithms that work in 3D or texture space. We lose irradiance in all points of the surface not seen from the camera, because only the visible pixels are rendered. So, we can no longer calculate the transmittance of light through thin parts of an object.

Eugene d'Eon and his colleagues proposed an algorithm[4] based on translucent shadow maps[7] with good results. However, their solution takes up to 30 percent of the total computation time (inferred from the performance analysis in their paper) and requires irradiance maps, which aren't available when simulating diffusion in screen space. We aim to simulate forward scattering through thin geometry with much lower computational costs, similar to how we've made reflectance practical.[1] From our observations of the transmittance phenomenon, we derived several assumptions on which we built a heuristic that let us approximately reconstruct the irradiance on the back of an object. This in turn let us approximately calculate transmittance based on the multipole theory.[8] The results show that we can produce images whose quality is on a par with photon mapping and other diffusion-based techniques (for a high-level overview of the diffusion approximation on which we based our algorithm, see the "Diffusion Profiles and Convolutions" sidebar). Our technique also requires minimal to no additional processing or memory resources.

Figure 2. Several objects showing varying degrees of translucency. As the middle and right images show, light transmitted through an object can greatly impact its final appearance.

Real-Time Transmittance Approaches

Our algorithm builds on two approaches. The first is Simon Green's approach, which relies on depth maps to estimate the distance a light ray travels inside an object.[9] The scene is rendered from the light's point of view to create a depth map that stores the distance from the objects nearest to the light (see Figure 4). For example, while rendering z_{out1}, this technique accesses the depth map to obtain the depth of z_{in1}, the point nearest to the light. It uses an operation similar to the one used in shadow mapping. However, instead of evaluating a comparison to determine whether a pixel is shadowed, it simply subtracts z_{in1} from z_{out1} of the pixel being shaded, obtaining s_1, the actual distance the light traveled inside the object.

After calculating this distance, Green's approach offers two ways to calculate the attenuation as a function of s:

- using an artist-created texture that maps distance to attenuation or
- attenuating light according to $T(s) = e^{-s\sigma_t}$, where σ_t is the extinction coefficient of the material being

(a)

(b)

Figure 3. Diffusion in texture space and screen space. (a) In texture space, Gaussian convolutions blur high-frequency details. The images show the initial irradiance map and two increasingly blurred versions. (b) In screen space, blurring is selectively applied on the rendered image.

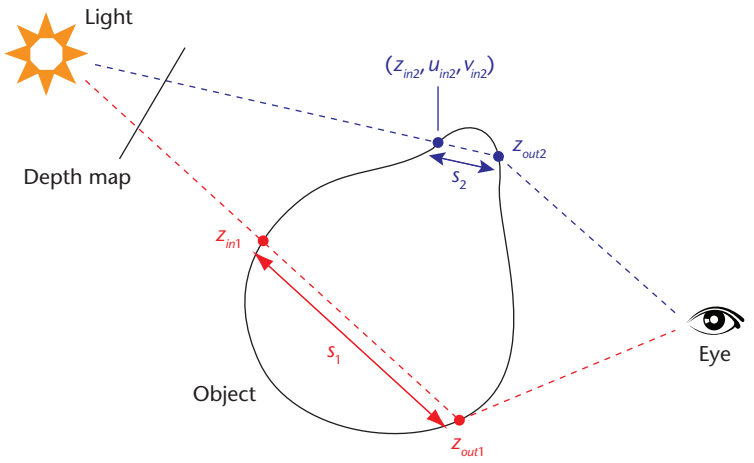


Figure 4. A comparison of Simon Green's approach[9] (red lines) to that of Eugene d'Eon and his colleagues[4,10] (blue lines). The former stores only depth information (z), whereas the latter stores z and the (u, v) coordinates of the points of the surface nearest the light. z_{out} represents the depth of the points where shading is being calculated, while z_{in} is the corresponding depth of the nearest point to the light source. s is the distance between z_{in} and z_{out}.

rendered and $T(s)$ is the transmission coefficient that relates the incoming and outgoing lighting.

An inherent problem with this transmittance approach (which also hinders most approaches based on shadow mapping) is that in theory it works only for convex objects. In practice, however, it approximates the solution well enough with arbitrary geometries.

The second approach is d'Eon and his colleagues' texture-space approach,[4,10] which extends the idea behind translucent shadow maps[7] to leverage the fact that the irradiance is calculated at each point of the surface being rendered. Texture-space diffusion, per se, doesn't account for scattering in areas that are close in 3D space but far in texture space. So, simulation of this effect requires special measurements. Translucent shadow maps store depth z, irradiance, and the normal of each point on the surface nearest the light, whereas the proposed modified translucent shadow maps store z and these points' (u, v) coordinates (see Figure 4). Then, while rendering, for example, z_{out2}, you can access the shadow map to obtain the (u_{in2}, v_{in2}) coordinates, which you can then use to obtain the irradiance at the back of the object. Using z_{out1} and z_{out2}, you can calculate the distance traveled through the object using the depth information from the shadow map, as in Green's approach.

As Figure 5 shows, the approach can approximate the radiant exitance at point C by the radiant exitance $M(x, y)$ at point B—where it's faster to calculate—using the irradiance information $E(x, y)$ around point A in the back of the object:

$$M(x,y) = E(x,y) * R\left(\sqrt{r^2 + d^2}\right). \quad (1)$$

As we saw, d'Eon and his colleagues calculate $R\left(\sqrt{r^2 + d^2}\right)$ using the Gaussian-sum approximation (see Equation D in the "Diffusion Profiles and Convolutions" sidebar):[4]

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

This lets them reuse the irradiance maps convoluted by each $G(v_i, r)$, used for reflectance calculation, for transmittance computations.

With shadow-map-based transmittance, high-frequency features in the depth of the shadow map might turn into high-frequency shading features. This is generally a problem when rendering translucent objects, where a softer appearance is expected. Green recommends sampling multiple points from the shadow map to soften these high-frequency depth changes. In d'Eon's texture-space approach, the distance traveled by the light inside the object is stored in the irradiance maps' alpha channel and blurred together with this irradiance information. The downside is that there's no obvious way to extend to multiple lights, because the alpha channel can store only the distance of one light.

Although Green's approach is physically based, if we use Beer's law instead of artist-controlled attenuation textures, it doesn't account for the attenuation of the light in multilayer materials. On the other side, d'Eon and his colleagues' approach requires texture-space diffusion, because in screen space there are no irradiance maps or irradiance information in the back of the object. Furthermore, the approach requires storing three floats in each shadow map (depth and texture coordinates), whereas regular shadow mapping requires storing only depth. This implies 3× memory usage and 3× bandwidth consumption for each shadow map.

Our Algorithm

Building on these ideas, we present a simple yet physically based transmittance shader. For this, we need a physically based function $T(s)$ that relates the attenuation of the light with the distance traveled inside an object. First, we make four observations:

1. For a great range of thin objects, we can approximate the normal at the back of the object to the reversed normal of the current pixel normal. This approximation will be exact when the front and back surfaces are parallel.
2. When looking at a backlit object from the front—which we consider the most interesting transmittance scenario—the viewer doesn't have accurate information of the irradiance at the back.
3. For materials with a tiny mean free path or for geometry with moderately thick surfaces—such as skin—transmittance is a very-low-frequency phenomenon. This is because the light is diffused as it travels inside an object, hiding most of its high-frequency features.
4. In human skin, the albedo (that is, the surface reflectivity) doesn't vary dramatically over its surface, maintaining a similar skin tone.

From these observations we make three assumptions. First, because of observation 1, we assume that we can replace the exact normal N_a at point A with the reversed normal N_c of the current point C (see Figure 5):

$$N_a = -N_c. \tag{3}$$

Second, because of observation 2, we assume that we can predict the irradiance at the back using some heuristic, such that it will be difficult to notice the difference. (We explain this heuristic later.)

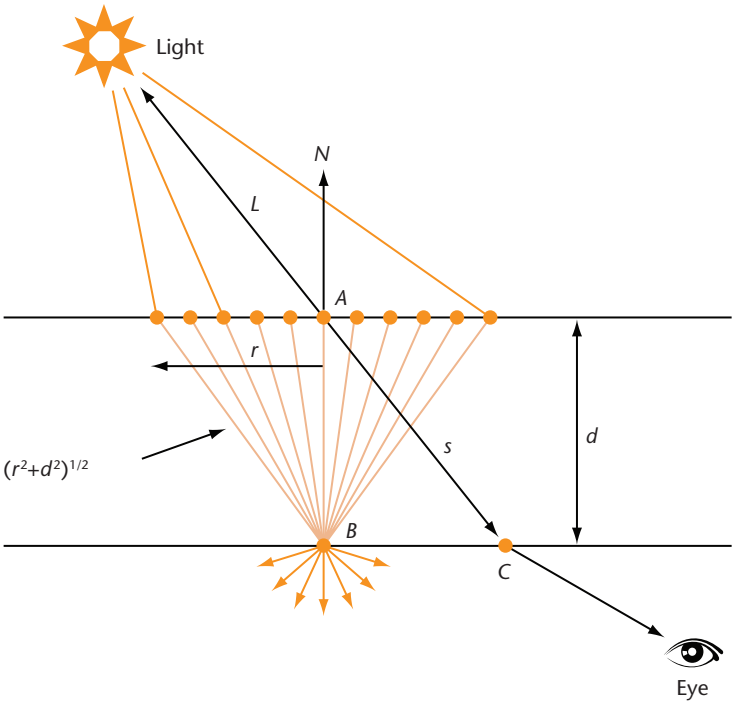


Figure 5. In d'Eon and his colleagues' approach, the radiant exitance at point *C* is approximated by the radiant exitance at point *B*—where it's faster to calculate—using the irradiance information *E* around point *A*.[4,10] *L* represents the light vector, *N* is the surface normal, *d* is the distance between *A* and *B*, *s* is the distance between *A* and *C*, and *r* is the distance from *A* to sampled points around it.

Finally, because of observations 2 and 4, we can safely use the albedo α_c at the front to approximate irradiance at the back of the object. Also, because of observation 3, even if we use high-frequency normals to calculate irradiance around A, we still get low-frequency transmitted lighting. Then, we assume that we can use low-frequency normals and obtain similar results. If we calculate the irradiance in the back using vertex normals—instead of normals from the normal map—we assure this irradiance will be free of high frequencies. (For real-time usage, the high-frequency details are in the normal map and not in the vertex normals.) In this case, the irradiance in the back—around A—will change slowly, so just taking a single irradiance value will produce a result similar to performing the full convolution.

We can assume, then, that irradiance in the back is approximately locally constant; all the points around A in Figure 5 will have the same value as point A:

$$E(x, y) = E = \alpha_c \max(-N_c \cdot L, 0.0). \tag{4}$$

Given a diffusion profile $R(r)$, the transmitted radiant exitance $M(x, y)$ through a planar slab is the convolution of the incoming irradiance with the diffusion profile (see Figure 5):[4,10]

Related Work in Subsurface-Scattering Simulation

Subsurface-scattering (SSS) simulation is usually described in terms of the *bidirectional scattering surface reflectance distribution function* (BSSRDF). Henrik Wann Jensen and his colleagues provided a huge step forward and made SSS practical, publishing techniques that the movie industry rapidly adopted.[1] On the basis of a dipole approximation of light diffusion, they captured the subtle softness that translucency adds to skin's appearance. Craig Donner and Jensen extended their dipole-based model to a multipole model, which also captures the effects of discontinuities at the frontiers of multilayered materials.[2] They show results with a broad range of objects, and their skin simulations are impressive.

George Borshukov and J.P. Lewis used 2D diffuse irradiance textures, with SSS simulated by a Gaussian function with a customizable kernel.[3] The technique maps naturally onto GPUs but doesn't capture the most complex subtleties of multiple scattering in materials. Marc Stamminger and Carsten Dachsbacher introduced translucent shadow maps, a modified version of shadow maps extended with irradiance and surface normal information.[4] In concurrent work, Tom Mertens and his colleagues presented a similar algorithm for local SSS, which solved the BSSRDF equation in image space using importance sampling.[5]

Simon Green's survey of real-time approximations to SSS explains three empirically based techniques that efficiently implement translucency using graphics hardware.[6] The first simply replaces grayscale ramps between absolute darkness and lightness with customized ramps that mimic the SSS influence in the illumination's gradients. The second technique uses depth maps to estimate the distance that light travels inside an object. The final technique empirically blurs 2D diffuse irradiance textures, as Borshukov and Lewis do in their research.[3]

Eugene d'Eon and his colleagues[7,8] simplified Donner and Jensen's model, combining it with diffusion in texture space[3] and translucent shadow maps.[4] They approximate the multipole scheme with a sum of Gaussian convolutions, obtaining separable multilayer diffusion profiles that are combined in a final rendering pass. Their results visually match Donner and Jensen's offline renderings but are achieved at real-time frame rates.

Jorge Jimenez and Diego Gutierrez reintroduced backface culling and view frustum clipping in the texture-space pipeline,[7,8] performing optimal per-object modulation of the irradiance map.[9] John Hable and his colleagues also reintroduced backface culling and proposed an additional optimization based on computing a single 2D convolution at 13 jittered sample points, which account for direct reflection, midlevel scattering, and wide red scattering.[10]

In recent research, Jimenez and his colleagues translated the light diffusion from texture to screen space.[11] They applied the convolution kernel in two ways: using a single 2D convolution with jittered samples (as Hable and his colleagues proposed) and using the six 1D Gaussian convolutions described in earlier research,[7] combining them with a weighted sum in a second pass. In follow-up work, Jimenez and his colleagues further optimized their screen-space algorithm in two ways.[12] First, they accumulated the Gaussian convolutions on the fly, thus reducing memory use and eliminating the need for a final pass to combine them. Second, they used the depth buffer to perform a depth-based Gaussian removal that prevents the Gaussian convolutions' execution in pixels far from the camera.

Although both Mertens and his colleagues and Jimenez and his colleagues work in screen space, their approaches

$$M(x,y) = \iint E(x,y)\,R\left(\sqrt{r^2+d^2}\right)dx\,dy\,. \tag{5}$$

By our first assumption, $E(x, y) = E$, so we have

$$\begin{aligned} M(x,y) &= E\iint R\left(\sqrt{r^2+d^2}\right)dx\,dy \\ &= E\int_0^\infty 2\pi r R\left(\sqrt{r^2+d^2}\right)dr. \end{aligned} \tag{6}$$

Placing Equation 2 into Equation 6, and considering that we define our Gaussian functions to have a unit total diffuse response,[4] we obtain

$$\begin{aligned} M(x,y) &= E\int_0^\infty \sum_{i=1}^{k} w_i e^{-d^2/v_i}\,2\pi r G(v_i,r)\,dr \\ &= E\sum_{i=1}^{k} w_i e^{-d^2/v_i}\int_0^\infty 2\pi r G(v_i,r)\,dr \\ &= E\sum_{i=1}^{k} w_i e^{-d^2/v_i}, \end{aligned} \tag{7}$$

which depends only on E and d. Approximating d by s and rewriting this equation, we can obtain the function $T(s)$ we wanted:

$$M(x, y) = ET(s) \tag{8}$$

$$T(s) = \sum_{i=1}^{k} w_i e^{-s^2/v_i}\,. \tag{9}$$

We can now precalculate $T(s)$ and store it in a look-up texture (see Figure 6), to use it as the attenuation texture of Green's approach. Using this $T(s)$ texture, we manage to produce similar results to the physically based approach[4] while leveraging a simpler technique.

For rendering, we simply need to add the contributions from the reflectance (obtained as usual) and the transmittance. We can safely sum reflected and transmitted lighting instead of blend-

to the BSSRDF differ. The former uses importance sampling, whereas the latter uses a sum-of-Gaussians approach. The latter approach can increase performance because you can use separable 1D convolutions.

Musawir Shah and his colleagues solved the BSSRDF in screen space using a splatting process, rather than a gathering process, for computing the integration.[13] Jimenez and his colleagues devised what's basically a dual technique.[11] Ralf Habel and his colleagues were the first to accurately represent SSS in plant leaves in real time.[14] They precomputed the expensive image convolution required to solve the BSSRDF, storing the results for real-time evaluation using the Half-Life 2 basis. Guillaume François and his colleagues introduced a technique that enables the rendering of single scattering events within multilayered materials in real time.[15] Using relief texture mapping to model the material's interior and two distance approximations to calculate the reduced-intensity L_{ri}, they can produce images on par with ray tracing.

References

1. H.W. Jensen et al., "A Practical Model for Subsurface Light Transport," *Proc. 28th Ann. Conf. Computer Graphics and Interactive Techniques*, ACM Press, 2001, pp. 511–518.
2. C. Donner and H.W. Jensen, "Light Diffusion in Multi-layered Translucent Materials," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 1032–1039.
3. G. Borshukov and J.P. Lewis, "Realistic Human Face Rendering for *The Matrix Reloaded*," *ACM Siggraph 2003 Sketches & Applications*, ACM Press, 2003.
4. M. Stamminger and C. Dachsbacher, "Translucent Shadow Maps," *Proc. 14th Eurographics Workshop Rendering*, Eurographics Assoc., 2003, pp. 197–201.
5. T. Mertens et al., "Efficient Rendering of Local Subsurface Scattering," *Computer Graphics Forum*, vol. 24, no. 1, 2005, pp. 41–50.
6. S. Green, "Real-Time Approximations to Subsurface Scattering," *GPU Gems*, R. Fernando, ed., Addison-Wesley, 2004, pp. 263–278.
7. E. d'Eon, D. Luebke, and E. Enderton, "Efficient Rendering of Human Skin," *Proc. Eurographics Symp. Rendering*, Eurographics Assoc., 2007, pp. 147–157.
8. E. d'Eon and D. Luebke, "Advanced Techniques for Realistic Real-Time Skin Rendering," *GPU Gems 3*, H. Nguyen, ed., Addison-Wesley, 2007, pp. 293–347.
9. J. Jimenez and D. Gutierrez, "Faster Rendering of Human Skin," *Proc. Congreso Español de Informática Gráfica 2008* (CEIG 08), Eurographics Assoc., 2008, pp. 21–28.
10. J. Hable, G. Borshukov, and J. Hejl, "Fast Skin Shading," *Shader X7*, W. Engel, ed., Charles River Media, 2009, pp. 161–173.
11. J. Jimenez, V. Sundstedt, and D. Gutierrez, "Screen-Space Perceptual Rendering of Human Skin," *ACM Trans. Applied Perception*, vol. 6, no. 4, 2009, pp. 1–15.
12. J. Jimenez and D. Gutierrez, "Screen-Space Subsurface Scattering," to be published in *GPU Pro*, W. Engel, ed., AK Peters, 2010.
13. M.A. Shah, J. Konttinen, and S. Pattanaik, "Image-Space Subsurface Scattering for Interactive Rendering of Deformable Translucent Objects," *IEEE Computer Graphics and Applications*, vol. 29, no. 1, 2009, pp. 66–78.
14. R. Habel, A. Kusternig, and M. Wimmer, "Physically Based Real-Time Translucency for Leaves," *Proc. Eurographics Symp. Rendering*, Eurographics Assoc., 2007, pp. 253–263.
15. G. François et al., "Subsurface Texture Mapping," *IEEE Computer Graphics and Applications*, vol. 28, no. 1, 2008, pp. 34–42.

(a) **(b)**

Figure 6. (a) Texture $T(s)$ (defined by Equation 13) encodes the transmitted lighting as a function of distance to be used as a look-up texture in Equation 9. In the left corner of the texture, we have $s = 0$; in the right corner, $s = 4$. (b) A translucent prism rendered using this texture.

ing them—as other approaches do—because we're using the reversed normal for transmittance calculations. This implies that reflected lighting and transmitted lighting can't happen simultaneously, thus avoiding double contribution. Also, although we perform our reflectance SSS calculations in screen space, we obtain the transmittance term in the conventional rendering pass.

As we explained in the section "Real-Time Transmittance Approaches," blurring high-frequency features in the depth map to simulate how light diffuses as it travels through an object is

Diffusion Profiles and Convolutions

For a multilayered material such as human skin, light enters an object, interacts with each of its layers, and exits at various points around the incident point (or is transmitted). Each layer absorbs and scatters the light differently, resulting in a complex process. However, we can describe this interaction in a simple form using *diffusion profiles*. A diffusion profile $R(x, y)$ is a function describing how the light attenuates at each position around the incident point. If we consider homogeneous materials, the attenuation is radially symmetric. So, we can define the diffusion profile as a function of distance r to the incident point $R(r)$.

To obtain such diffusion profiles, we rely on diffusion theory and arrive at the classic diffusion equation:[1]

$$D\nabla^2\phi(x) = \sigma_a\phi(x) - Q_0(x) + 3D\vec{\nabla}\cdot\vec{Q}_1(x). \quad \text{(A)}$$

This equation has a simple solution for a single isotropic point light source in an infinite medium. For finite media, the diffusion equation has no analytical solution. For a semi-infinite plane-parallel medium, this equation has no simple solution. Henrik Wann Jensen and his colleagues managed to approximate a solution,[1] which in turn lets us obtain diffusion profiles for semi-infinite materials using a dipole approximation.

This approach works well for materials such as marble. For complex, multilayered materials such as human skin, we must use the multipole model.[2] In this model, the medium is no longer semi-infinite and has a finite thickness. For finite slabs, we can no longer solve Equation A using the dipole model. By accounting for the transmitted and reflected light at each slab of a multilayered material, we can obtain a numerical diffusion profile (as opposed to the analytic profile obtained from the dipole model).

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

$$M(x,y) = \iint E(x,y)R(r)\,dx\,dy, \quad \text{(B)}$$

where $M(x, y)$ is the radiant exitance at point P, and $E(x, y)$ is the irradiance around P. The integral is basically summing the contribution of each point around P, each one weighted by its own attenuation factor given by the diffusion profile $R(r)$. We can write Equation B as a 2D convolution:

$$M(x,y) = E(x,y) * R(r). \quad \text{(C)}$$

Two-dimensional convolutions are costly for real-time applications. Fortunately, some 2D convolutions are separable, which means we can separate them into two faster 1D convolutions. Gaussian convolutions are one of these separable convolutions. Eugene d'Eon and his colleagues showed that a diffusion profile resembles the aspect of a Gaussian convolution.[3,4] With this observation in mind, we approximate the full 2D convolution by a sum of Gaussian convolutions:

$$R(r) = \sum_{i=1}^{k} w_i G(v_i, r), \quad \text{(D)}$$

which we can separate into faster 1D convolutions.

References

1. H.W. Jensen et al., "A Practical Model for Subsurface Light Transport," *Proc. 28th Ann. Conf. Computer Graphics and Interactive Techniques*, ACM Press, 2001, pp. 511–518.
2. C. Donner and H.W. Jensen, "Light Diffusion in Multi-layered Translucent Materials," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 1032–1039.
3. E. d'Eon, D. Luebke, and E. Enderton, "Efficient Rendering of Human Skin," *Proc. Eurographics Symp. Rendering*, Eurographics Assoc., 2007, pp. 147–157.
4. E. d'Eon and D. Luebke, "Advanced Techniques for Realistic Real-Time Skin Rendering," *GPU Gems 3*, H. Nguyen, ed., Addison-Wesley, 2007, pp. 293–347.

recommended. Instead of blurring the distance traveled inside the object, as previous approaches have done, we simply store the transmittance and reflectance together. We then use the screen-space Gaussian convolutions to blur them, yielding good results.

Implementation Details

Although using the reversed normal for transmittance calculations avoids double contribution, it causes nonsmooth transitions between areas illuminated by reflectance to areas of transmittance-only illumination. In these transitions, the dot product between the normal N and the light vector L is zero for both N and $-N$. To avoid these abrupt illumination changes, we increase the range of the object covered by the transmittance component, using this formula:

$$E = \alpha_c \max\left(0.3 + (-N_c \cdot L), 0.0\right). \quad (10)$$

This means that the transmittance dot product will begin approximately 17 degrees before where it would begin if we used the usual $N \cdot L$ product. The minus is because we're using the reversed normal for transmittance calculations.

A problem of using shadow maps for depth approximations is that artifacts can appear around the projection's edges because pixels from the background are projected onto the object's edges. To solve this problem, Green recommends growing the vertices in the direction of the normals while rendering the shadow maps.[9] This ensures that all points fall onto the object while querying the depth from the shadow map. We opted to shrink the object instead in the normal direction, while querying the depth map. This yields the same result but has the advantage of using standard, unmodified shadow maps.

Figure 7 shows the 25 lines of code that execute our skin shader's transmittance calculations, which highlight its simplicity.

```
float distance(float3 pos, float3 N, int i){
  float4 shrinkedpos = float4(pos − 0.005 * N, 1.0);
  float4 shwpos = mul(shrinkedpos, lights[i].viewproj);
  float d1 = shwmaps[i].Sample(sampler, shwpos.xy/shwpos.w);
  float d2 = shwpos.z;
  return abs(d1 - d2);
}

// This function can be precomputed for efficiency
float3 T(float s) {
  return float3(0.233, 0.455, 0.649) * exp(−s*s/0.0064) +
         float3(0.1,   0.336, 0.344) * exp(−s*s/0.0484) +
         float3(0.118, 0.198, 0.0)   * exp(−s*s/0.187)  +
         float3(0.113, 0.007, 0.007) * exp(−s*s/0.567)  +
         float3(0.358, 0.004, 0.0)   * exp(−s*s/1.99)   +
         float3(0.078, 0.0,   0.0)   * exp(−s*s/7.41);
}

float s = scale * distance(pos, Nvertex, i);
float E = max(0.3 + dot(−Nvertex, L), 0.0);
float3 transmittance = T(s) * lights[i].color *
                       attenuation * spot * albedo.rgb * E;

// We add the contribution of this light
M += transmittance + reflectance;
```

Figure 7. Our shader's transmittance code (using the High Level Shader Language). The depth values obtained from shadow maps are assumed to be linear. This figure highlights the simplicity of our approach.

Results

We developed a skin-rendering algorithm that can simulate the complex mechanics of SSS. It extends the screen-space-based reflectance-only approach[1] by adding transmittance. This important feature adds great realism to the images, as the results show.

Beyond its inherent simplicity, our transmittance algorithm has four advantages. First, it doesn't require irradiance textures as input.

Second, it only requires standard shadow maps as input, which means the rendering pipeline is almost left unaltered. This greatly simplifies adding this feature to any pipeline.

Third, it requires less memory storage than previous approaches. We only require z information, eliminating the need to also store (u, v) coordinates for each pixel of the shadow map. This translates to a one-third memory usage with respect to previous approaches.

Finally, regarding bandwidth, we have to access the memory only twice per pixel (one float each time), once to obtain the depth in the back of the object (z) and once to obtain $T(s)$. This amounts to $2 \times 4 = 8$ bytes per pixel. d'Eon and his colleagues' approach[4] requires three memory accesses (they use only the three widest Gaussian convolutions for transmittance calculations) to access z and the (u, v) coordinates. This amounts to $3 \times 3 \times 4 = 36$ bytes per pixel, which implies 450 percent of memory bandwidth usage with respect to our approach.

To create our renderings, we used an Intel Core i7 CPU 920 at 2.67 GHz and an Nvidia GeForce GTX 295. The head model has 25K triangles with 2,048 × 2,048 color, shadow, and normal maps. We used up to five lights to illuminate the model. Because we perform only a simple texture access, enabling transmittance in a real-time application is an almost free operation. We've measured a performance drop of only 0.97 percent when activating transmittance in our application. Figure 8 compares a hand rendered using photon mapping[11] and our approach. Although some difference exists in the colors of the photon-mapping image, it preserves the characteristic translucency and yellow-to-red gradients. These gradients result from the varying thickness traveled by the light. When the thickness is small, the dermis doesn't color the light. This is the skin layer that colors the light with a reddish tone, because it mostly absorbs other wavelengths. So, the light is influenced only by the epidermis, resulting in a yellowish tone. In the thicker areas, both layers color the light, resulting in a more reddish tone. The photon-mapping images require 15 minutes to compute, whereas our algorithm requires only 5 milliseconds per frame.

Figure 9 compares an ear rendered using the multipole model[12] and our simplified approach. Visual inspection yields similar perceived features, although they're inherently different because of the difference in the geometry. (We didn't have access to the model Craig Donner and Henrik Wann Jensen used,[12] so we used a model from XYZ RGB instead; www.xyzrgb.com.) Both ears exhibit similar red-to-black gradients, a result of the underlying physical model.

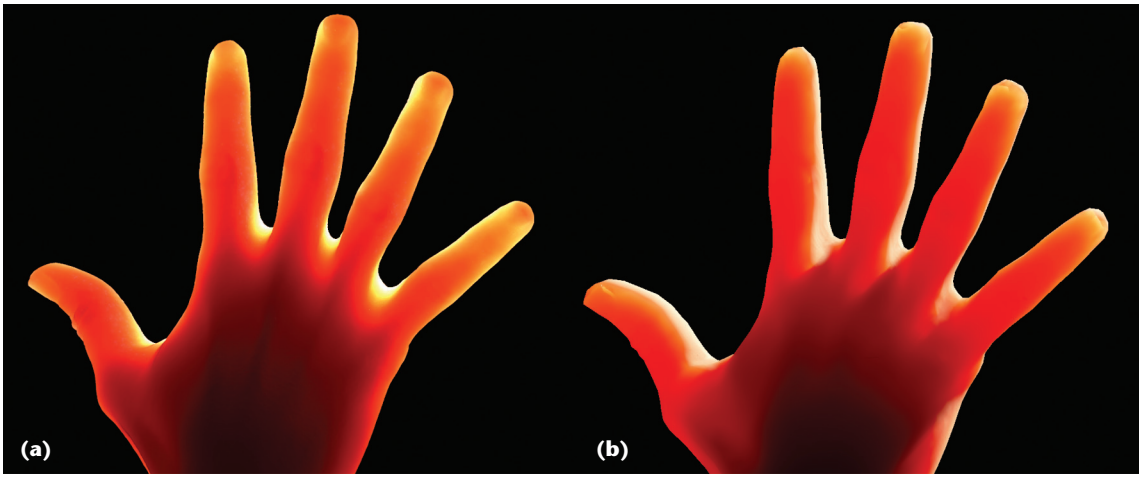


Figure 8. Comparison between (a) photon mapping[11] and (b) our algorithm. Although our algorithm can't match the photon-mapping image's colors exactly, it represents the most characteristic visual cue: the yellow-to-red gradients.

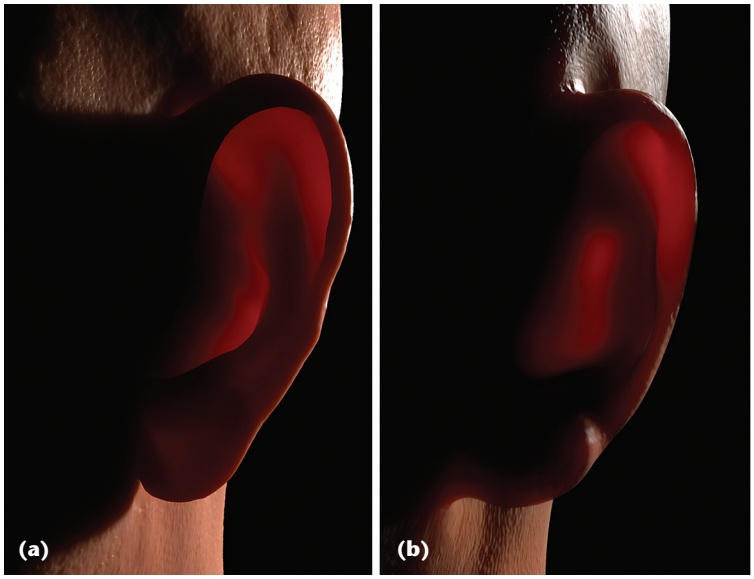


Figure 9. Comparison between (a) the multipole application[12] and (b) our simplified approach. Because the underlying geometry is different, the look doesn't match exactly. Nonetheless, both images have a similar red-to-black gradient.

Figures 10 and 11 show additional renderings featuring the transmission of light in the ear, nostril, and fingers that account for the most evident cases of SSS.

Given some of our assumptions, our algorithm might not work in all cases. First, we can't accurately represent transmittance through overlapping geometry—a limitation common to all shadow-map-based approaches. Second, high-frequency features in the shadow maps could be visible; blurring both transmittance and reflectance ameliorates this, a solution we find acceptable in game contexts. However, a better solution would be blurring the thickness from the light point of view with a Gaussian convolution to soften the transmittance-enabled objects' appearance.

Despite these two limitations, our simplified approach creates very realistic renderings of human skin, including difficult scenarios involving the ears, nostrils, and hands, in which transmittance becomes important. We have shown how our algorithm compares well against other techniques such as multipole diffusion approximation or photon mapping. We believe that the quality of the images produced by our algorithm, and its simplicity, efficiency, and pluggability, make it a good choice for rendering truly realistic skin in the next generation of games.

Acknowledgments

We thank the reviewers for their constructive, detailed comments and XYZ RGB Inc. for the high-quality head scan. The Spanish Ministry of Science and Technology (project TIN2007-63025) and the Aragón government (projects OTRI 2009/0411 and CTPP05/09) partially funded this research. Jorge Jimenez was funded by a research grant from the Aragón government.

References

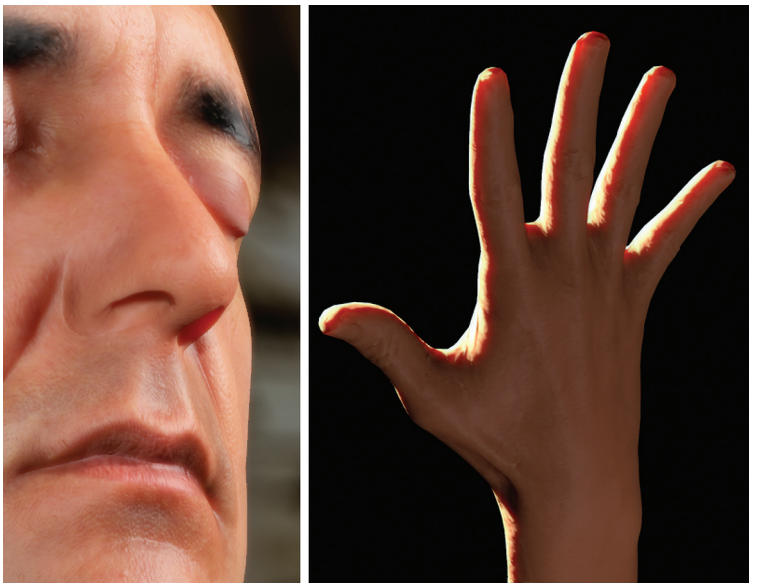
1. J. Jimenez, V. Sundstedt, and D. Gutierrez, "Screen-Space Perceptual Rendering of Human Skin," *ACM Trans. Applied Perception*, vol. 6, no. 4, 2009, pp. 1–15.
2. T. Igarashi, K. Nishino, and S.K. Nayar, *The Appearance of Human Skin*, tech. report, Computer Science Dept., Columbia Univ., 2005.
3. D. Gosselin, "Real Time Skin Rendering," ATI Research, 2004; http://developer.amd.com/media/gpu_assets/D3DTutorial_Skin_Rendering.pdf.

4. E. d'Eon, D. Luebke, and E. Enderton, "Efficient Rendering of Human Skin," *Proc. 2007 Eurographics Symp. Rendering*, Eurographics Assoc., 2007, pp. 147–157.
5. J. Hable, G. Borshukov, and J. Hejl, "Fast Skin Shading," *Shader X7*, W. Engel, ed., Charles River Media, 2009, pp. 161–173.
6. J. Jimenez and D. Gutierrez, "Faster Rendering of Human Skin," *Proc. Congreso Español de Informática Gráfica 2008* (CEIG 08), Eurographics Assoc., pp. 21–28.
7. M. Stamminger and C. Dachsbacher, "Translucent Shadow Maps," *Proc. 14th Eurographics Workshop Rendering*, Eurographics Assoc., 2003, pp. 197–201.
8. C. Donner and H.W. Jensen, "Light Diffusion in Multi-layered Translucent Materials," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 1032–1039.
9. S. Green, "Real-Time Approximations to Subsurface Scattering," *GPU Gems*, R. Fernando, ed. Addison-Wesley, 2004, pp. 263–278.
10. E. d'Eon and D. Luebke, "Advanced Techniques for Realistic Real-Time Skin Rendering," *GPU Gems 3*, H. Nguyen, ed., Addison-Wesley, 2007, pp. 293–347.
11. C. Donner and H.W. Jensen, "Rendering Translucent Materials Using Photon Diffusion," *Proc. Eurographics Symp. Rendering*, Eurographics Assoc., 2007, pp. 243–252.
12. C. Donner and H.W. Jensen, "A Spectral BSSRDF for Shading Human Skin," *Proc. Eurographics Symp. Rendering*, Eurographics Assoc., 2006, pp. 409–417.

Jorge Jimenez *is a PhD student in real-time rendering at the University of Zaragoza. His interests include real-time photorealistic rendering, special effects, and squeezing rendering algorithms to be practical in game environments. Jimenez received his MSc in computer science from the University of Zaragoza. Contact him at jim@unizar.es.*

David Whelan *is an independent programmer. His interests include real-time rendering of translucency in skin. Whelan received his MSc in computer science (interactive entertainment technology) from Trinity College Dublin. Contact him at wheland8@tcd.ie.*

Veronica Sundstedt *is a lecturer in computer graphics at the Blekinge Institute of Technology. Her research interests are graphics, perception, and eye tracking. Sundstedt received her PhD in computer graphics from the University of Bristol. She's on the editorial board of* ACM Transactions on Applied Perception. *Contact her at veronica.sundstedt@bth.se.*



(a) **(b)**

Figure 10. (a) Our rendering model can simulate the transport of light through thin parts such as the nostril. (b) Lighting a hand with a powerful light source generates intense red gradients at the finger boundaries, where the light travels the shortest distance inside the object.

Figure 11. Renderings of realistic skin under different lighting configurations. Even with high-frequency detail in the normal map—as the specular reflections in the images reveal—our simulation of subsurface scattering produces a soft, diffuse appearance.

Diego Gutierrez *is an associate professor of computer graphics at the University of Zaragoza, where he obtained his PhD on the topic of rendering participating media. His research interests include global illumination, perception, and image processing. He's an associate editor of* ACM Transactions on Applied Perception *and* Computers & Graphics. *Contact him at diegog@unizar.es.*