Real-Time Spectral Scattering in Large-Scale Natural Participating Media

Oskar Elek
Faculty of Mathematics and Physics
Charles University in Prague

Petr Kmoch
Faculty of Mathematics and Physics
Charles University in Prague

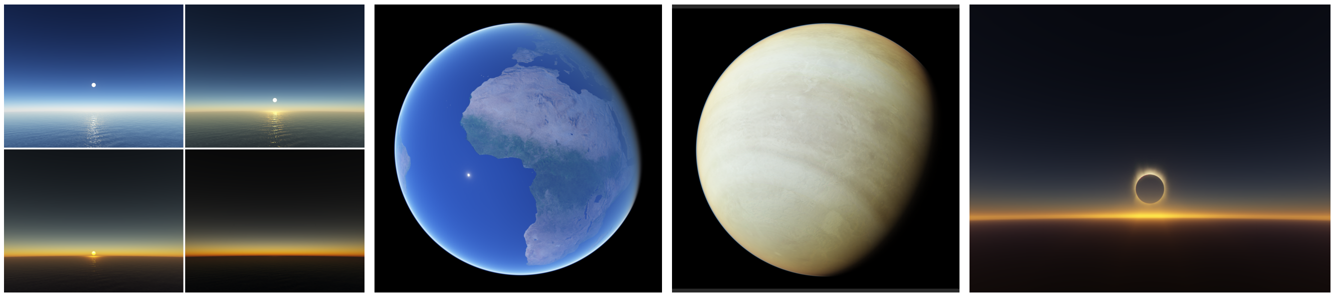

Figure 1: Examples of the presented method — sequence of the setting Sun, orbital views of the Earth and Venus, and Sun eclipse.

Abstract

Real-time rendering of participating media in nature presents a difficult problem. The reason is that realistic reproduction of such media requires a proper physical simulation in all cases. In our work we focus on real-time rendering of planetary atmospheres and large areas of water. We first formulate a physically-based model for simulation of light transport in these environments. This model accounts for all necessary light contributions — direct illumination, indirect illumination caused by the scattered light and interreflections between the planetary surface and the atmospheric volume, as well as reflections from the seabed. We adopt the precomputation scheme presented in the previous works to precompute the colours of the arbitrarily dense atmosphere and large-scale water surfaces into a set of lookup tables. All these computations are fully spectral, which increases the realism. Finally we utilize these tables in a GPU-based algorithm that is capable of rendering a whole planet with its atmosphere from all viewpoints above the planetary surface. This approach is capable to achieve hundreds of frames per second on today's graphics hardware.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: atmospheric and aquatic light scattering, natural phenomena, participating media

1 Introduction

Planetary atmosphere and large areas of water, such as the oceans, are perhaps the most common participating media around us, as we know at least the first one from our everyday experience. Thus any rendering system that strives for realistic display of any general outdoor scene needs to incorporate a believable method for rendering these. This includes 3D computer games, flight and driving simulations, virtual worlds and even animated movies. Still, applications like GoogleEarth, NASA WorldWind or for instance SecondLife use simplistic methods for rendering of the Earth's atmosphere and water areas. 3D computer games are more advanced in this direction, but they still use some combinations of static skyboxes and substantially simplified physically-based algorithms.

In this paper we present a method for physically-based real-time rendering of general planetary atmospheres and water bodies. We build on the methods presented in [Bruneton and Neyret 2008] and [Elek 2009], which are equivalent — both rely on precomputation of the scattering data into a set of lookup tables in an incremental manner (one scattering order at a time). We extend these methods by precomputing and rendering large aquatic bodies, which can range from small lakes to oceans. We also generalize the atmospheric model to account for atmospheres with arbitrary density. Finally, our work is the first to utilize fully spectral scattering data for these environments in real-time rendering. Fig. 1 shows some of the results of our work.

Related work is discussed in Section 2. We formulate the physical model in Section 3 and describe its precomputation in Section 4. Then we show how to render the precomputed data in Section 5 and after the results and implementation figures in Section 6 we conclude the paper in Section 7.

2 Related Work

Simulation of planetary atmosphere appearance has been the point of interest in the field of computer graphics for a long time. The absence of sufficiently powerful graphics hardware implied that for long only non-interactive methods were used to display atmospheric light scattering effects.

[Nishita et al. 1993] presented equations for computing single scattering in the Earth's atmosphere; this model is still widely used although it does not enable changing the atmosphere's density. [Nishita et al. 1996] was based on division of the skydome into a set of cells and theoretically capable of computing an arbitrary number of scattering orders, but practically usable only up to the

SCCG 2010, Budmerice, Slovakia, May 13 – 15, 2010.
© 2010 ACM 978-1-4503-0558-7/10/0005 $10.00

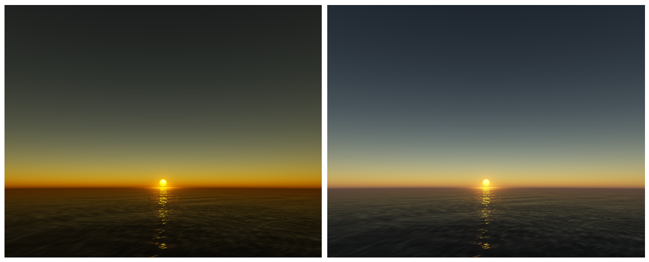

Figure 2: Comparison of single and multiple scattering.

second order, due to computational intensity. A method for displaying the sky from the ground was presented in [Haber et al. 2005]. It was based on the simulation of radiative transfer in the body of the skydome and capable of simulating an arbitrary number of scattering orders. The method also enabled the simulation of a wide range of atmospheric conditions. However, despite numerous optimizations, it remained computationally expensive. A completely different approach was used in [Preetham et al. 1999], where an analytic solution for calculation of the colour of the Earth's sky was presented. However, as pointed out in [Zotti et al. 2007], the model was behaving incorrectly under some specific conditions and could even yield negative intensities.

Fully accurate real-time methods for realistic rendering of planetary atmospheres are still not feasible today. However, by exploiting graphics hardware's texturing capabilities and various approximations, it is now possible to devise a real-time method while keeping an acceptable quality of its output. All of the following approaches can render the Earth's atmosphere in real-time.

[O'Neil 2004] presented an approximative method for rendering the Earth's atmosphere based on the model from [Nishita et al. 1993]. O'Neil here suggested usage of 2D lookup tables for storing the optical depth and predicted that the whole single-scattering integral should be precomputable into a 3D lookup table. Then in [O'Neil 2005] he took a different direction and presented a set of *ad hoc* analytic functions for the atmosphere's colour, evaluated in vertex shader. [Wenzel 2006] presented a fast method periodically precalculating the single-scattering integral into a 2D texture. Schafhitzel et al. [Schafhitzel et al. 2007] figured out the way to fully precompute the single-scattering equations into a 3D lookup table. However, it was found a full parameterization requires a 4D table.

As previously mentioned, [Bruneton and Neyret 2008] and [Elek 2009] presented two equivalent precomputation schemes for multiple atmospheric scattering. Both are based on an incremental computation of the scattering dataset. One scattering order is calculated at time, always from the data for the previous order. While the first method uses a full 4D parameterization of the main scattering table, the second tries to approximate the dimension which corresponds to sun↔view azimuth by an *ad hoc* phase function during the real-time evaluation. We build on these methods by generalizing them to arbitrarily dense atmospheres and large water bodies.

In the area of real-time rendering of water bodies, research falls into two categories — generation of surface wave geometry and simulation of light transport in the water volume. As we are not concerned with wave geometry, we will limit ourselves to the second group. For a good overview of methods from the first category, please refer to [Bruneton et al. 2010], who also present a hierarchical representation of the ocean surface combining geometry, perturbed normals and a modified BRDF.

As for the second group, there are surprisingly few works that concern themselves with simulation of the light scattering within water bodies. [Nishita et al. 1993] presented, along with their atmospheric model, also an analytical expression of the first scattering order in water in order to render oceans. This has been further developed by [Iwasaki et al. 2003], where the authors added the computation of the second scattering order; then they used virtual sampling planes to render the water volume. However, the derivation of their equations remains unclear and the data they use are difficult to obtain. [Premoze and Ashikhmin 2000] presented a simplified radiative transfer algorithm, which they used to render various types of natural waters, for example tropical or muddy water. [Cerezo and Serón 2002] show how different phytoplankton concentrations in water influence its appearance. It is important to note however, that none of these methods runs in real-time (except for [Iwasaki et al. 2003], which reaches interactive framerates).

3 Physical Model

In this section we describe the physical model used in our work, for sake of completeness. We first present the equations for the atmospheric computations and then describe the differences when calculating the optical properties of water.

3.1 Scattering in atmosphere

Scattering coefficient At first we need the amount of light scattered by one air particle. This depends on the Rayleigh and Mie mean particle polarizability constants α_R and α_M [Born and Wolf 1999] expressed as

$$\alpha_{\{R,M\}} = \frac{2\pi^2(n_e^2-1)^2}{3N^2_{e_{\{R,M\}}}} \tag{1}$$

where n_e is the index of refraction of the Earth's atmosphere at sea level and $N_{e_{\{R,M\}}}$ is the Rayleigh/Mie particle molecular number density in the Earth's atmosphere at sea level. These are calculated from measured parameters of the Earth's atmosphere. From this we can express the Rayleigh/Mie scattering coefficient $\sigma_{\{R,M\}}$ as

$$\sigma_R(\lambda) = 4\pi\frac{N_R}{\lambda^4}\alpha_R \qquad\qquad \sigma_M = 4\pi N_M \alpha_M \tag{2}$$

where $N_{\{R,M\}}$ denotes molecular number density of Rayleigh/Mie particles in the desired atmosphere. Note that unlike σ_R, the coefficient σ_M is λ-independent. This formulation of the scattering coefficient allows us to simulate atmospheres with arbitrary density (unlike for instance [Nishita et al. 1993], [Schafhitzel et al. 2007] or [Bruneton and Neyret 2008]). As shown in Figures 1 and 4, this can produce qualitatively different appearances.

Scattering intensity The Rayleigh/Mie scattering intensity of the first order $L^{(1)}_{S_{\{R,M\}}}$ expresses the amount of light deviated by a given angle θ during a scattering event at point P (see Fig. 3 for illustration). It depends on spectral wavelength λ and is expressed by the following equation:

$$L^{(1)}_{S_{\{R,M\}}}(\lambda,\theta,P) = I_I(\lambda)\cdot F_{\{R,M\}}(\theta)\cdot\rho_{\{R,M\}}(P)\cdot\sigma_{\{R,M\}}(\lambda) \tag{3}$$

where I_I is the spectral intensity of direct incident light and $\rho_{\{R,M\}}(P) = \exp(-\frac{h}{H_{\{R,M\}}})$ is the Rayleigh/Mie density scale function. This function expresses the decrease of atmospheric density in dependence on h, the altitude of P over the ground. $H_{\{R,M\}}$ is

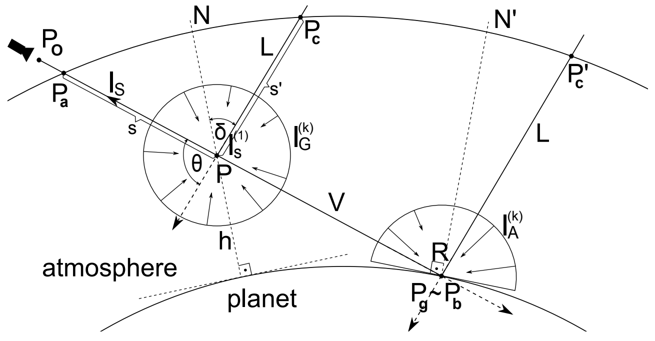

Figure 3: A schematic view of the atmosphere.

the atmospheric Rayleigh/Mie scale height expressing the altitude where the density of the respective type of particles scales down by a $1/e$ term. For the Earth's atmosphere $H_R \approx 8$km and $H_M \approx 1.2$km. $F_{\{R,M\}}$ is Rayleigh/Mie phase function, which expresses angular dependence of scattered light intensity on the scattering angle θ:

$$F_R(\theta) = \frac{3}{4}(1+\cos^2(\theta)) \quad (4)$$

$$F_M(\theta,g) = \frac{3(1-g^2)}{2(2+g^2)}\frac{(1+\cos^2(\theta))}{(1+g^2-2g\cos(\theta))^{3/2}} \quad (5)$$

where Eq. 4 is the standard Rayleigh phase function and Eq. 5 is the improved Henyey-Greenstein approximation of the general Mie phase function by [Cornette and Shanks 1992]. $g \in \langle -1;1\rangle$ describes Mie scattering anisotropy.

Single scattering The single-scattering equation describes the intensity of light $I^{(1)}_{S_{\{R,M\}}}$ that reaches an observer P_O looking in the direction V (see Fig. 3), after exactly one scattering event:

$$t(S,\lambda) = \sum_{i\in\{R,M\}} \sigma_i(\lambda)\int_0^S \rho_i(s')ds' \quad (6)$$

$$I^{(1)}_{S_{\{R,M\}}}(P_O,V,L,\lambda) = R(P_O,V,L,\lambda) + I_I(\lambda)F_{\{R,M\}}(\theta)\cdot \quad (7)$$

$$\cdot\frac{\sigma_{\{R,M\}}(\lambda)}{4\pi}\int_{P_a}^{P_b}\rho_{\{R,M\}}(P)e^{-t(PP_c,\lambda)-t(P_aP,\lambda)}ds$$

$$R(P_O,V,L,\lambda) = \bar{a}(\lambda)(-L\cdot N')I_I(\lambda)e^{-t(P_gP'_c,\lambda)-t(P_aP_g,\lambda)} \quad (8)$$

where L is the direction to the light source and the sample point P is parameterized by s. P_a and P_b are the first and the last point where the density of the atmosphere is nonzero (for observer situated inside the atmosphere, $P_a = P_O$). P_c is the intersecting point of L with the upper atmospheric boundary when starting in P. t is the attenuation coefficient in accordance with the Beer-Lambert-Bouguer law. We add R into Eq. 7 to account for the reflection of incident light from the planetary surface in the calculation of higher scattering orders. Here, N' is the surface normal at P_g and $\bar{a}$ is the mean albedo of the planetary surface (we use $\bar{a} = 0.45$). If V intersects the planetary surface, then $P_g = P_b$, otherwise $R = 0$.

Multiple scattering Any model which strives for realism must account for multiple scattering (see Fig. 2). The key to make calculation of multiple scattering possible is to split the formulation of the radiative transfer equation into terms that will allow us to precompute it. Otherwise, one would have to resort to recursive distributed Monte-Carlo evaluation, which would have an exponential computational complexity (in respect to the number of calculated scattering orders, denoted K).

At first we define a gathered light intensity of k^{th} order $I^{(k)}_{G_{\{R,M\}}}$ at some point in the atmosphere P, an ambient light intensity of k^{th} order $I^{(k)}_{A_{\{R,M\}}}$ at some surface point P_g and a composite ambient light intensity of k^{th} order $A^{(k)}_{\{R,M\}}$ at some observer position P_O, in the view direction V and with the light source direction L as

$$I^{(k)}_{G_{\{R,M\}}}(P,V,L,\lambda) = \int_{4\pi} F_{\{R,M\}}(\theta)I^{(k)}_{S_{\{R,M\}}}(P,\omega,L,\lambda)d\omega \quad (9)$$

$$I^{(k)}_{A_{\{R,M\}}}(P_g,L,\lambda) = \int_{2\pi}(N'\cdot\omega)I^{(k)}_{S_{\{R,M\}}}(P_g,\omega,L,\lambda)d\omega \quad (10)$$

$$A^{(k)}_{\{R,M\}}(P_O,V,L,\lambda) = \bar{a}(\lambda)I^{(k)}_{A_{\{R,M\}}}(P_g,L,\lambda)e^{-t(P_aP_g,\lambda)} \quad (11)$$

where θ is the scattering angle between V and ω and $I^{(k)}_{S_{\{R,M\}}}$ is the scattered light intensity of k^{th} order. These formulae denote the amount of light, which has undergone exactly k scattering events, reflected (in-scattered) into the direction $-V$ at P (Eq. 9), reaching the point P_g (Eq. 10) and reaching P_O from P_g (Eq. 11).

Having all we need we can now define the scattered light intensity of k^{th} order $I^{(k)}_{S_{\{R,M\}}}$ at the observer position P_O coming from the direction V as

$$I^{(k)}_{S_{\{R,M\}}}(P_O,V,L,\lambda) = A^{(k-1)}_{\{R,M\}}(P_O,V,L,\lambda) + \frac{\sigma_{\{R,M\}}(\lambda)}{4\pi}\cdot \quad (12)$$

$$\cdot\int_{P_a}^{P_b} I^{(k-1)}_{G_{\{R,M\}}}(P,V,L,\lambda)\rho_{\{R,M\}}(P)e^{-t(P_aP,\lambda)}ds$$

where the notation stays similar to Eq. 7. If we now define the total intensity of the k^{th} order of scattered light as $I^{(k)}_S = I^{(k)}_{S_R} + I^{(k)}_{S_M}$ we can finally express the total scattering intensity of first K orders as

$$\boxed{I_S = \sum_{k=1}^{K} I^{(k)}_S} \quad (13)$$

3.2 Scattering in water

The calculation of light scattering in water is similar to the calculation of atmospheric scattering. Most parts of the model stay the same, only now the volume of the medium is not located between the upper atmospheric boundary and the planetary surface, but between the water level and the seabed. Moreover there are also other specificities that we must account for.

Even though the atmosphere can contain absorbing particles, we assume in Section 3.1 that it does not (although they can easily be added by incorporating the absorption coefficient into Eq. 6). For water this is not possible, as it exhibits significant absorption, mainly in the red part of the spectrum. To obtain absorption and scattering coefficients for clear water, ς^a and ς^s, we use the data from [Buiteveld et al. 1994], finely sampled at 2nm. Water can also contain additional organic and inorganic compounds (see Fig. 8).

Another difference to take into account is the fact that water density is constant. Therefore, we do not need the density scale function ρ, which can simply be removed from all equations. This greatly simplifies the matter, for example the attenuation coefficient t^{water} can now be expressed as

$$t^{\text{water}}(S,\lambda) = \|S\|(\varsigma^a+\varsigma^s)(\lambda). \quad (14)$$

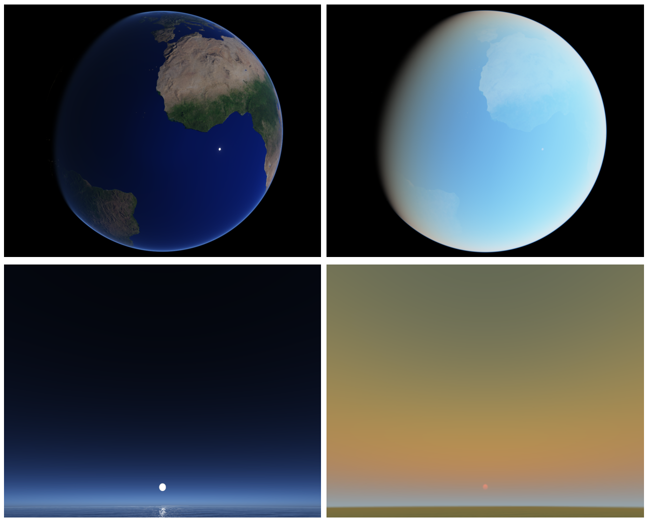

Figure 4: Comparison of different densities — atmosphere 10 times sparser (left) and 10 times denser (right) than the Earth's one.

From now on, we will indicate the terms defined for water with the superscript *water* for distinction.

It can also be noticed that we do not explicitly model Mie scattering in water. There are two reasons for that. First, the data for Mie scattering coefficient in seawater (pure water does not contain any Mie-scattering compounds) are not available, as it is already difficult to obtain the data for the Rayleigh scattering coefficient. Second, as in most cases the behaviour of Mie scattering is strongly anisotropic with a dominant forward lobe, it would only have noticeable effect for an underwater observer looking towards the light source. Since we account only for observer positions above the water level, we can neglect Mie scattering in water.

4 Precomputation

4.1 Parameterization

To make the precomputation possible for every observer position P_O, every view direction V and every light source direction L, a convenient parameterization to fit the scattering data into a 3D texture readable by the graphics hardware must be defined. To do this, we follow the ideas from [Schafhitzel et al. 2007; Bruneton and Neyret 2008] and make some assumptions:

1. The star (light source) is so far away that all light rays from it can be considered parallel
2. The planet is perfectly spherical (neglecting terrain morphology for now)
3. The density of the atmosphere is changing in respect to altitude ($\rho_{\{R,M\}}(P)$) but not in respect to latitude and longitude
4. The atmosphere is a spherical shell and is symmetrical around the plane between L and the zenith vector in P_O

Thanks to these assumptions, P_O, V and L can be reduced into 4 scalar parameters — altitude $h \in \langle 0; H_{max}\rangle$ (where H_{max} is the upper atmosphere boundary altitude), view↔zenith angle $\vartheta \in \langle 0;\pi\rangle$, sun↔zenith angle $\delta \in \langle 0;\pi\rangle$ and sun↔view azimuth $\phi \in \langle 0;\pi\rangle$. We typically use $H_{max} = 65$km (although for denser atmospheres it may be necessary to use $H_{max} \approx 100$km or even more). We use tiling to emulate a 4D table in a 3D texture and manual linear interpolation of the 4th dimension during rendering.

Our task is now to remap the parameters of our lookup table h, ϑ, δ and ϕ into the 4D table coordinate space $UVWR$, i.e. to design a remapping function $f : \langle 0;H_{max}\rangle \times \langle 0;\pi\rangle^3 \rightarrow \langle 0;1\rangle^4$:

$$\begin{aligned}
U(h) &= Z_1/Z_2 \\
V(\vartheta) &= 1/2 + (h\cos(\vartheta) + \sqrt{Z_3})/(2Z_1) \\
&\quad \text{if } h\cos(\vartheta) < 0 \text{ and } Z_3 > 0, \text{ or otherwise} \\
&= 1/2 - (h\cos(\vartheta) - \sqrt{Z_3 + Z_2^2})/(2Z_1 + 2Z_2) \\
W(\delta) &= (1 - e^{-2.8\cos(\delta) - 0.8})/(1 - e^{-3.6}) \\
R(\phi) &= (1 + \cos(\phi))/2
\end{aligned}$$

with $Z_1 = \sqrt{h^2 - r_P^2}$, $Z_2 = \sqrt{r_A^2 - r_P^2}$, $Z_3 = r^2\cos^2(\vartheta) - Z_1^2$, r_P and r_A being the radii of the planet and the atmospheric shell, respectively. The optimized remapping functions U, V and W are adopted from [Bruneton and Neyret 2008]. We use a slightly changed W, to truly account for all sun angles where the scattering intensity is nonzero even in denser atmospheres or atmospheres with very high H_{max}, since the original W from [Bruneton and Neyret 2008] was specifically designed for the Earth's atmosphere.

For water, the situation is different. Despite the fact that we want to model only observers above the water level, we still need to calculate light scattering for points inside the water volume, if we want to model multiple scattering in water (see Section 4.3 for further details). Moreover, we want to model water for more than one depth D, as, unlike the upper atmospheric boundary, the depth of water areas is usually not constant. Adding the three angles to complete the parameterization, this leaves us with the need of a 5D table to store the simulation results. Because manipulation with an emulated 5D table would be complicated and its size would be too large, we need to get rid of at least one dimension.

We decided to exclude the sun↔view azimuth ϕ from the parameterization of the water texture, because our observations and tests show that the intensity of scattered light varies negligibly with respect to ϕ. The reason for this is most probably the strong locality of light transport in water (visible light which travels 100m in water retains only $0.6 \cdot 10^{-20}\%$ to 1.5% of its original intensity, depending on the wavelength) — the areas with different ϕ are much closer to each other than in the atmosphere (assuming L stays the same) and the difference between them is small.

On the plus side, the change of the scattering intensity with respect to the rest of the parameters is also not as dynamic as in the atmosphere. We therefore do not need any complicated nonlinear parameterization; a simpler linear parameterization will suffice here:

$$\begin{aligned}
U(D) &= D/D_{max} \\
V(d) &= d/D \\
W(\vartheta) &= (1 + \cos(\vartheta))/2 \\
R(\delta) &= (1 + \cos(\delta))/2
\end{aligned}$$

where D_{max} is the peak depth to account for and d is observer depth. There is no point in using $D_{max} > 100$m, due to the aforementioned low transmissivity of water (see Fig. 5).

4.2 Precomputing atmospheric scattering

We will now review the precomputation algorithm as presented in [Bruneton and Neyret 2008] and [Elek 2009]. The formulation

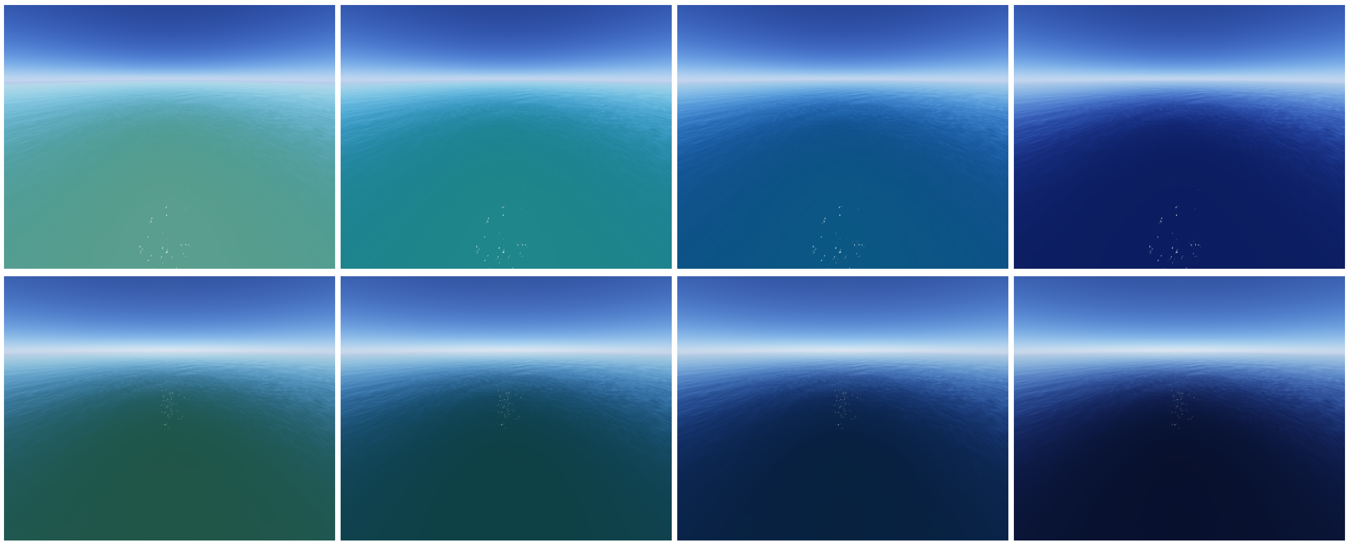
Figure 5: Showing the effect of varying water depth with a sandy seabottom. From left to right: water depth of 2m, 4m, 10m and 100m at noon (top row) and at 4 PM (bottom row). Fisheye camera is used to cover a larger field of view.

of the physical model allows us to precompute multiple scattering in an incremental manner, one scattering order at the time. At first the $I_S^{(1)}$ is computed exactly, according to Eq. 7 and the results of this computation are stored in the lookup table as described in Section 4.1. Then we can compute the desired number of higher scattering orders — for calculation of the $I_S^{(k)}$ (Eq. 12), we need to obtain $A^{(k)}$ (Eq. 11) and $I_G^{(k)}$ (Eq. 9). This can be done in two ways — either repeatedly computing them directly during the evaluation of Eq. 12 for $I_S^{(k)}$, or precomputing them into auxiliary lookup tables in each iteration prior to the calculation of $I_S^{(k)}$; in either case, the lookup table for $I_S^{(k-1)}$ is used during the computation. Both ways are possible, but, despite the first method being more accurate, it is very convenient to use the precomputed auxiliary textures, as this is much faster and the actual difference in quality is minimal. We need a 1D table to store $A^{(k)}$ and a 4D table to store $I_G^{(k)}$. It is even possible to use a 2D table for $I_G^{(k)}$, neglecting the viewing direction V — this way, the directional information provided by the phase function is lost, but the error is still very small, because the integration over the sphere in Eq. 9 largely averages the phase function.

However, there are some considerations that [Bruneton and Neyret 2008] and [Elek 2009] omit. To complete the computation we must correct the resulting 4D scattering texture for the light that is reflected off the ground. This includes Eqs. 8 and 11, because they use the mean surface albedo $\bar{a}$ that treats the planetary surface as a homogeneous reflector. That is because during rendering, we want to account for surfaces with different albedos and reflection characteristics, so we want to remove these contributions from the scattering texture and evaluate them separately at runtime. For this, we construct a single-purpose correction texture with the exactly same resolution and parameterization as the main scattering texture and fill it with correcting light intensity $I_{C_{\{R,M\}}}$ defined as follows:

$$I_{C_{\{R,M\}}}(P_O,V,L,\lambda) = R(P_O,V,L,\lambda) + \sum_{k=1}^{K-1} A^{(k)}_{\{R,M\}}(P_O,V,L,\lambda) \quad (15)$$

Note that $I_C = 0$ if V does not intersect the planet. The correction is then done by subtracting intensity values of the correction texture from the main scattering texture on a per-texel basis. For the complete precomputation procedure please refer to Algorithm 1.

Moreover, it is necessary to consider the fact that while the limited angular resolution of the 4D scattering table (see Section 6 for concrete figures) is sufficient for Rayleigh scattering, it is not sufficient for accurate reproduction of Mie scattering, mainly in the areas where the angle θ between V and L is small, as $F_M(\theta,g)$ has a very high gradient there. To overcome this problem we use the similarity theory [Wyman et al. 1989]. During the precomputation process we consider Mie scattering to be isotropic (i.e. $F_M(\theta,g)=1$) and use the effective scattering coefficient σ'_M defined as

$$\sigma'_M = (1-g)\sigma_M. \quad (16)$$

We then precompute $F_M(\theta,g)$ into a small 2D texture and evaluate it in the fragment shader during rendering. This also allows us to change the anisotropy parameter g in real-time (even though it is partially hardwired in σ'_M; still the error is negligible here). Even though this approach is somehow mathematically incorrect, it reproduces the effects caused by Mie scattering surprisingly well, most probably due to the dominance of the first order of Mie scattering. Arguably, one could use a convolution of multiple $F_M(\theta,g)$ for precomputation into the 2D texture, instead of the $F_M(\theta,g)$ alone, as described in [Riley et al.]. It is also worth mentioning that we treat I_{S_M} as being monochromatic, so it occupies only one channel in the scattering texture. We then shade it during rendering by multiplication with the attenuated direct illumination I'_I (see Section 4.3 for the definition; alternatively, one could use the proportionality rule from [Bruneton and Neyret 2008]).

4.3 Precomputing scattering in water

As mentioned in Section 3.2 the physical model stays similar for water volumes. With some modifications, this also applies for the precomputation algorithm. It is necessary to use different physical constants (Section 3.2) and parameterization (Section 4.1) for the water texture, as well as the formula for the attenuation coefficient t^{water} (Eq. 14). It is also important to note that the incident light in all computations of scattering in water no longer has the same

```
Data: n-dimensional textures denoted as TexnD
Result: 4D texture TS containing $I_S$ and 1D texture TA
        containing $I_A = \sum_{k=1}^{K-1} I_A^{(k)}$
1  begin
2      initialize all constants and parameters;
3      Tex4D TS $\overset{compute}{\longleftarrow} I_S^{(1)}$;                    // Eq. 7
4      Tex4D TS$_\Delta$ $\leftarrow$ TS;
5      Tex1D TA $\leftarrow$ 0;
6      Tex1D TA$_\Delta$ $\leftarrow$ 0;
7      Tex4D TG $\leftarrow$ 0;
8      for $k \leftarrow 2$ to $K$ do
9          TG $\overset{compute}{\longleftarrow} I_G^{(k-1)}$[TS$_\Delta$];              // Eq. 9
10         TA$_\Delta$ $\overset{compute}{\longleftarrow} I_A^{(k-1)}$[TS$_\Delta$];             // Eq. 10
11         TS$_\Delta$ $\overset{compute}{\longleftarrow} I_S^{(k)}$[TG,TA$_\Delta$];           // Eq. 12
12         TS $\leftarrow$ TS + TS$_\Delta$;
13         TA $\leftarrow$ TA + TA$_\Delta$;
14     end
15     Tex4D TC $\overset{compute}{\longleftarrow} I_C$[TA];                   // Eq. 15
16     TS $\leftarrow$ TS $-$ TC;
17 end
```
Algorithm 1: Pre-computing multiple scattering.

spectral composition as in the computations of atmospheric scattering, because it always has to pass through the layer of atmosphere, which attenuates it. The attenuated incident light can be expressed as $I_I'(\lambda) = I_I(\lambda)e^{-t(P_g P_c', \lambda)}$ (see Fig. 3). Another difference from the atmospheric scattering precomputation is that while the 4D texture containing the atmospheric colour data is directly utilizable in rendering, the 4D texture containing the water colour data should be reduced into a 3D texture after the precomputation step. This is because it contains the data for all possible observer depths, which we do not need for observer positions above the water surface. Therefore we use for rendering only the subset of the 4D texture which corresponds to the observer depth $d = 0$m. On the other hand, if observers below the water surface needed to be considered, it would be possible to use the full 4D texture to render underwater scenes as well. This would be similar to observers in various altitudes in the atmosphere — here, the observer depth d would be used to fetch the water scattering texture (along with the rest of the parameters).

It can be noted that if high accuracy is not a priority, it's sufficient to use only the first scattering order in water. The reason for this is that water is a weakly scattering medium, so most of the water colour comes from reflection off the seabed. This simplifies and speeds up the precomputation, as it is now necessary to precompute the first scattering order only (Lines 2 and 3 in Algorithm 1), and it also allows us to decrease the dimensionality of the scattering table for water — we can exclude $V(d)$ from the entire precomputation.

4.4 Spectral precomputation

To accurately simulate light scattering effects and especially multiple scattering, using the RGB space for the simulation is not sufficient (see Fig. 6). Although the involved spectra are not extremely spiky, they are not linear either (especially the water absorption spectrum). We therefore use a full spectral computation, resulting in smoother, less saturated and thus more natural look of the atmosphere and water areas. As hardware-compliant textures can contain 4 colour channels at most, an array of 4D textures must be used to emulate a spectral texture with an arbitrary amount of spectral channels. For the simulation with M spectral samples an array of $\lceil (M+1)/4 \rceil$ textures has to be used ('+1' for storing I_{S_M}).

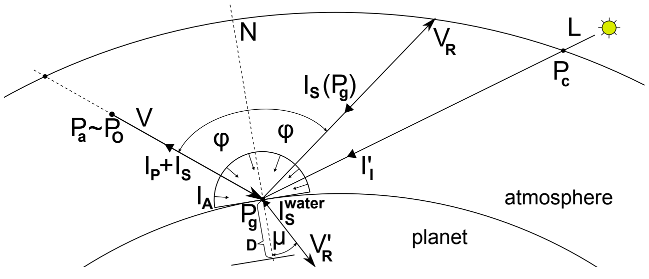

Figure 7: A scheme illustrating the calculation of the light contributions from the planetary surface.

Upgrading RGB computation to spectral is not so difficult. First, the constants needed for the simulation have to be spectral, which is not a problem at most cases, as the available data are often already sampled at more wavelengths than three. The rest can be calculated, for instance we obtain the incident light intensity as blackbody radiation at 5778K, according to Planck's law (the desired light source temperature can be changed, of course). Second, the algorithm itself does not change, except that now it is necessary to work with the array of 4D textures. This implies that instead of fetching a single texture, each texture in the array must be fetched using exactly the same texture coordinates (the same holds for writing). Last, the conversion from spectral to RGB colour values must be performed in order to utilize the resulting texture on graphics hardware. For this we first convert the spectral colour values to CIE XYZ space using the standard XYZ primaries (available at http://www.cis.rit.edu/mcsl). This XYZ tristimulus value is then moved to the gamut of sRGB space and converted to sRGB by multiplication with an XYZ→sRGB conversion matrix. The result is then stored in a single 4D texture for use in rendering.

Another important consideration is that even if one chooses to perform only RGB precomputation instead of the full spectral one for some reason, it is not correct to use the obtained scattering values directly as RGB colour components (even if this produces fairly good results). This is because the R, G and B components do not correspond to any concrete values of λ. Instead, such data have to be treated as spectra with only 3 samples, and the conversion described above should be used to obtain usable RGB colour values.

5 Rendering

We do not discuss the rendering of terrain morphology, since this is not the topic of our work. For the description of how to utilise the scattering lookup texture precomputed for a perfectly spherical planet in a terrain renderer, please refer to [Bruneton and Neyret 2008] or [Schafhitzel et al. 2007]. For our purposes we represent both the planet and the atmospheric shell as finely tessellated spheres rendered with frontface culling (atmosphere) and backface culling (planet) enabled.

To avoid costly real-time calculation of the extinction coefficient t, we precompute it into a 2D lookup texture parameterized by the observer altitude h and zenith angle ϑ. This texture contains $t(PP_c)$, where P has altitude h and P_c is the intersection of the ray that starts at P under zenith angle ϑ with the upper atmosphere boundary.

For rendering the atmosphere we just have to fetch the appropriate colour value from the scattering texture. Fetching means evaluating

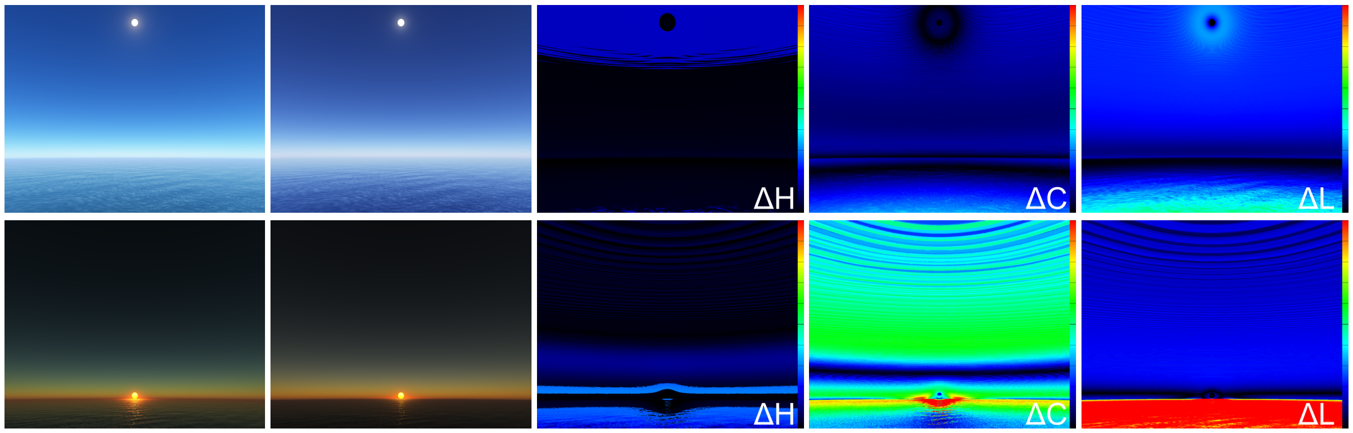

Figure 6: Using spectral precomputation (second column) produces more natural results than RGB precomputation (first column). The differences are not only in lightness, but also in chroma (spectral data are less saturated) and even slightly in hue component, and are more substantial during sunset, when higher scattering orders are stronger. The scale represents a maximum difference of 10%.

the remapping function for the atmosphere parameterization f from the spatial position of the observer and retrieving two values from the scattering texture (to perform a manual linear interpolation for the fourth dimension). The fragment colour I_F is then obtained simply as $I_F = I_S$. If the observer is situated outside the atmosphere, we compute the intersection point P_a between the view ray and the sphere with the atmosphere radius and evaluate f for this point, as there is no scattering or attenuation outside the atmosphere.

For shading the planetary surface we must also evaluate contributions other than the scattered light. For the ground, which is generally diffuse, we must account also for the direct illumination I'_I as well as indirect ambient illumination I_A caused by the scattering. For water surfaces, which are specular, it is necessary to account for the reflected skylight and sunlight I_R and of course for the light from within the water volume I_S^{water}. In either case the light has to be finally attenuated on its path from P_g to the observer P_O. See Fig. 7 for the situation depiction.

Therefore, to obtain the light coming from the planetary surface I_P, we have to define two distinct formulae:

$$\begin{aligned} I_P(P_O,V,L,\lambda) &= I_D(\lambda)(-L\cdot N)I'_I(P_g,L,\lambda)+ \\ &\quad +I_D(\lambda)I_A(P_g,L,\lambda) \qquad \text{if } P_g \text{ on ground} \\ I_P(P_O,V,L,\lambda) &= \hat{F}(\varphi)(I_S(P_g,V_R,L,\lambda)+I'_I(P_g,V_R,\lambda))+ \\ &\quad +I_S^{\text{water}}(P_g,V'_R,D,\lambda) \qquad \text{if } P_g \text{ on water} \end{aligned}$$

Here, I_D stands for the diffuse colour of the ground, fetched from a surface texture, for example. $\hat{F}(\varphi)$ is the Fresnel term approximation denoting the amount of light reflected off the water surface. We don't use the full Fresnel term; instead we use the fast and accurate approximation from [Lazányi and Szirmay-Kalos 2005] (which can furthermore be precomputed). V'_R is the refraction ray, which we compute in the fragment shader using the `refract()` intrinsic function of the Cg language. I_S^{water} is obtained by fetching the water scattering texture using the depth D of the point where V'_R intersects the seabottom. $I'_I(P_g,V_R,\lambda) = I'_I(P_g,L,\lambda)$ if V_R hits the star disc, otherwise it is zero.

To obtain I_F, we just sum the (attenuated) surface colour and the scattering contribution:

$$\boxed{I_F = I_S + I_P \cdot e^{-t(P_aP_g,\lambda)}}$$

Note that I_F has a very high dynamic range. To correct this we apply a very simple tone mapping operator from [O'Neil 2005]: $I_F = 1-\exp(-E\cdot I_F)$, where E is the exposure constant.

6 Implementation and Results

Our implementation uses a CPU-based program for the precomputation algorithm and a GPU-based renderer. The precomputation could be also carried out by a GPU (as shown in [Bruneton and Neyret 2008]), but we have chosen CPU-implementation because of its flexibility suitable for the spectral precomputation.

The used resolutions of the lookup textures are $32\times128\times32\times8$ for the atmospheric scattering 4D table (packed in a $32\times128\times256$ 3D texture), $32\times64\times64$ for the water scattering 3D texture, 256×512 for the 2D extinction texture and 256 for the 1D ambient texture. Their size together is 10MB (using 16b floating-point values per channel). We sample the visible spectrum at 15 evenly distributed samples. For numerical sampling of all integrals a regular trapezoidal evaluation rule with 30 samples for each integral was used. Using higher sampling rates does not improve the apparent quality of the calculated data.

The hardware configuration used for testing was a desktop PC with *Intel Core 2 Duo 1.86 Ghz* CPU and *NVidia GeForce 8800GT* graphics adapter. The precomputation time for the entire dataset was about 1.5 hours for 6 scattering orders when using the aforementioned sampling rates. Using more than 6–7 scattering orders is not needed, since at this point the solution is well converged. However, for very dense atmospheres, using up to 10 scattering orders might be necessary. The performance of our rendering application is always in real-time framerates. For the screen resolution of 1024×768 the average measured framerate was 180 FPS and the minimal framerate was 85 FPS. For the screen resolution of 2560×2048 the average measured framerate was 60 FPS and never dropped below 29 FPS. The scene in both measurements contained approximately 1.1 million static triangles.

Our method is a bit slower than the one of [Bruneton and Neyret 2008], since we also evaluate the reflection and scattering from water bodies. The reason why they report lower framerates is that they also account for the terrain morphology, while we do not.

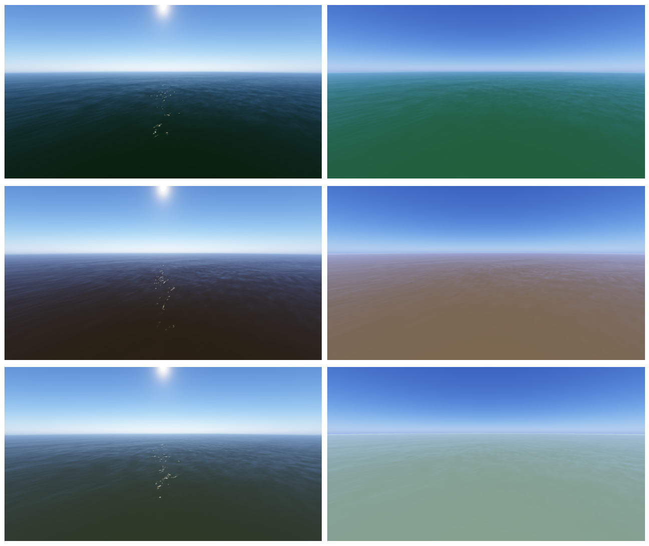

Figure 8: Comparison of various water types appearance at morning and afternoon. Top: Water with high algae concentration. Middle: Muddy water. Bottom: Water with high phytoplankton abundance (phytoplankton scattering and absorption data obtained from [Cerezo and Serón 2002]).

7 Conclusion

We have presented a method for physically-based real-time rendering of planetary atmospheres and water surfaces. Our method builds on the works of [Bruneton and Neyret 2008] and [Elek 2009], but is more capable, allowing simulation of arbitrarily dense planetary atmospheres and adding support for large-scale water bodies within the same physical model and precomputation framework. By doing fully spectral computations we also increase the realism of the obtained data. We have shown how to utilize these data in a real-time renderer, accounting for all involved light contributions.

Our work primarily targets real-time applications, such as flight simulators and games, various virtual environments, but also widespread software like Google Earth, where our model could be used after some simplifications. Especially applications which need to render both the atmosphere and large water volumes will benefit from this work. We also think that some non-interactive renderers could utilize the data from the precomputation step, as full simulation of large-scale multiple scattering is not feasible even in these.

In the future we would like to generalize the model even more to account for local changes of atmospheric parameters, so we can drop the second half of Assumption 3. Another direction worth exploring would be to include other types of participating media, often suspended in the atmosphere and natural waters.

Acknowledgements We would like to thank Alexander Wilkie and Jaromír Plášek for their useful comments and advice during the creation of this work.

References

BORN, M. F., AND WOLF, E. 1999. *Principles of Optics*, 7th ed. Cambridge University Press.

BRUNETON, E., AND NEYRET, F. 2008. Precomputed atmospheric scattering. In *Proceedings of EGSR '08*.

BRUNETON, E., NEYRET, F., AND HOLZSCHUCH, N. 2010. Real-time realistic ocean lighting using seamless transitions from geometry to BRDF. *Comput. Graph. Forum 29*, 2.

BUITEVELD, H., HAKVOORT, J. M. H., AND DONZE, M. 1994. The optical properties of pure water. In *SPIE Proceedings on Ocean Optics XII*.

CEREZO, E., AND SERÓN, F. J. 2002. Rendering natural waters: Merging computer graphics with physics and biology. In *Proceedings of CGI '02*.

CORNETTE, W. M., AND SHANKS, J. G. 1992. Physical reasonable analytic expression for the single-scattering phase function. *Applied Optics 31*, 16.

ELEK, O. 2009. Rendering parametrizable planetary atmospheres with multiple scattering in real-time. In *Proceedings of the Central European Seminar on Computer Graphics*.

HABER, J., MAGNOR, M., AND SEIDEL, H.-P. 2005. Physically-based simulation of twilight phenomena. *ACM Transactions on Graphics 24*, 4.

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2003. A volume rendering approach for sea surfaces taking into account second order scattering using scattering maps. In *Proceedings of Volume Graphics '03*.

LAZÁNYI, I., AND SZIRMAY-KALOS, L. 2005. Fresnel term approximations for metals. In *WSCG 2005 Short Communications Proceedings*.

NISHITA, T., SIRAI, T., TADAMURA, K., AND NAKAMAE, E. 1993. Display of the earth taking into account atmospheric scattering. In *Proceedings of SIGGRAPH 93*.

NISHITA, T., DOBASHI, Y., KANEDA, K., AND YAMASHITA, H. 1996. Display method of the sky color taking into account multiple scattering. In *Proceedings of Pacific Graphics*.

O'NEIL, S. 2004. Real-time atmospehric scattering. http://www.gamedev.net/reference/articles/article2093.asp.

O'NEIL, S. 2005. Accurate atmospheric scattering. *GPU Gems 2*. Addison-Wesley Professional.

PREETHAM, A. J., SHIRLEY, P., AND SMITS, B. 1999. A practical analytic model for daylight. In *Proceedings of SIGGRAPH 99*.

PREMOZE, S., AND ASHIKHMIN, M. 2000. Rendering natural waters. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*.

RILEY, K., EBERT, D. S., KRAUS, M., TESSENDORF, J., AND HANSEN, C. Efficient rendering of atmospheric phenomena.

SCHAFHITZEL, T., FALK, M., AND ERTL, T. 2007. Real-time rendering of planets with atmospheres. *Journal of WSCG 15*.

WENZEL, C. 2006. Real-time atmospheric effects in games. In *ACM SIGGRAPH 2006 Courses*.

WYMAN, D. R., PATTERSON, M. S., AND WILSON, B. C. 1989. Similarity relations for the interaction parameters in radiation transport. *Applied Optics 28*.

ZOTTI, G., WILKIE, A., AND PURGATHOFER, W. 2007. A critical review of the preetham skylight model. In *WSCG 2007 Short Communications Proceedings I*.