Eurographics Symposium on Rendering 2020
C. Dachsbacher and M. Pharr
(Guest Editors)

Volume 39 (2020), Number 4

A Scalable and Production Ready Sky and Atmosphere Rendering Technique

Sébastien Hillaire [1]

[1]Epic Games, Inc

Figure 1: *Rendered images of different atmospheric conditions and view points using the method presented in this article. Left to right: ground views of an Earth-like daytime and Mars-like blue sunset, and space views of an Earth-like planet and an artistic vision of a tiny planet.*

Abstract

We present a physically based method to render the atmosphere of a planet from ground to space views. Our method is cheap to compute and, as compared to previous successful methods, does not require any high dimensional Lookup Tables (LUTs) and thus does not suffer from visual artifacts associated with them. We also propose a new approximation to evaluate light multiple scattering within the atmosphere in real time. We take a new look at what it means to render natural atmospheric effects, and propose a set of simple look up tables and parameterizations to render a sky and its aerial perspective. The atmosphere composition can change dynamically to match artistic visions and weather changes without requiring heavy LUT update. The complete technique can be used in real-time applications such as games, simulators or architecture pre-visualizations. The technique also scales from power-efficient mobile platforms up to PCs with high-end GPUs, and is also useful for accelerating path tracing.

CCS Concepts
• ***Computing methodologies*** → ***Rasterization; Ray tracing;***

1. Introduction

Rendering natural phenomena is important for the visual simulation of believable worlds. Atmosphere simulation and rendering is important for applications requiring large open worlds with dynamic time of day, or viewing planets from space. Such applications include games, architectural visualization and flight or space simulators. However, current methods have limitations: they are either restricted to views from the ground, can only represent a single atmosphere type, require computationally expensive updates of lookup tables (LUTs) when atmospheric properties are changed, or can even exhibit visual artifacts.

We present a method to render a planet's sky and aerial perspective from a physically based representation of the atmosphere's participating medium in real time. Our contributions in this paper are the following:

- We propose a sky and aerial perspective rendering technique relying on LUTs to evaluate expensive parts of the lighting integral at lower resolution while maintaining important visual features.
- We propose a novel way to evaluate the contribution of light multiple scattering in the atmosphere. It can approximate an infinite number of scattering orders and can also be used to accelerate path tracing.
- The technique supports *dynamic time of day* along with dynamic updates of the atmospheric properties, all while rendering effi-

ciently on a wide range of devices, from a low-end Apple iPhone 6s to consoles and high-end gaming PCs.

This method is used in Epic Games' Unreal Engine[†]. In this paper, we will be using photometric units (luminance/illuminance) instead of radiometric units (radiance/irradiance). This due to the prevalence of these terms in modern game engines [LdR14].

After reviewing previous work in Section 2, we briefly describe participating media rendering (with a focus on the atmospheric case) in Section 3. The atmospheric material model used in this paper is presented in Section 4, and our atmosphere rendering technique is detailed in Section 5. Results and comparisons to a path-traced ground truth and to a previous model are discussed in Section 6. Finally, we report on performance in Section 7 and conclude.

2. Previous work

The first wave of sky rendering techniques were focused on ray marching the atmosphere from the view point. This is what Nishita et al. [NSTN93] first proposed as a method to render an atmosphere from ground and space views. O'Neil [ONe07] proposed integrating the in-scattered luminance per vertex for the sake of performance, and to render the final sky color with the phase function applied per pixel. Wenzel [Wen07] proposes the same idea but with in-scattered luminance stored in a texture that is updated over several frames to amortize the cost. The major drawback of these models is that they ignore the impact that light multiple scattering can have on the look of the sky.

In order to reduce the cost of ray marching and include multiple scattering, analytical models fitted on real measurements [PSS99] or on reference generated using path tracing with spectral information [HW12] have been proposed. These models are very fast to evaluate and benefit from a simple parameterization: for example, a single turbidity value is used to represent the amount of aerosols in the air, resulting in a denser looking atmosphere. However, they are limited to views from the ground and to the single atmosphere type the parameters have been fitted to. For example, it is not possible to render the Mars sky when the model is fitted to the Earth sky.

More advanced models have been proposed for rendering atmospheric effects with multiple scattering, for views ranging from the ground to space. Nishita [NDN96] proposed subdivision of the participating medium into voxels, and the simulation of energy exchange between them. More affordable models that remove the voxel representation have been proposed: they store the result of integrations that can be expensive to evaluate into lookup tables that can be easily queried at run time on GPU. These LUTs can be sampled per pixel at run time (according to view, sun and world information) to compute the transmittance and in-scattered luminance. Bruneton and Neyret [BN08] proposed a 4D LUT while Elek [Ele09] discarded one dimension, effectively ignoring the planet's shadowing of the atmosphere that is visible when the sun is just below the horizon. Because, in these models, in-scattering from the viewer to a mesh surface is evaluated as the subtraction of two values sampled form a LUT, visual artifacts can appear at the horizon due to resolution and parameterization precision issues. Yusov [Yus13] improved the situation through a better parameterization, which works well for Earth-like atmospheres. However, artifacts can still be visible in cases where the atmosphere is denser. For each of these LUT models, multiscattering is achieved by evaluating the in-scattering LUT iteratively: sampling the scattered luminance from the previous scattering order LUT to evaluate the new one. When all are added together, this forms the final in-scattering LUT with multiple scattering orders up to the iteration count. However, such LUTs are cumbersome to update when a game needs to update its atmospheric properties, e.g. due to a change in weather conditions or to match the art direction. It is possible to time slice updates, but this will result in a visual delay between sun movement and sky color [Hil16]. LUT-based models have source code available online [Bru17b; Yus13] and have been used successfully in several games [Hil16; dCK17; Bau19]. Going further, Bruneton [Bru17a] discussed all of those models extensively, and compared their advantages and limitations.

One of the challenges when rendering an atmosphere is to represent volumetric shadowing due to hills and mountains. It is possible to rely on epipolar lines [Yus13], shadow volumes [Bru17b], or a variant of shadow volumes extruding meshes from shadow maps [Hoo16]. These techniques are fast but can only represent sharp shadows from opaque meshes. They will fail to render the soft shadows resulting from cloud participating media or sun disk area light shadow penumbrae. This is an area where ray marching still has a definite advantage in capturing such soft details.

3. Participating media rendering

Rendering participating media can be achieved using ray marching or path tracing. In both cases it involves using a material parameterization representing participating media as described by the radiative transfer equations [FWKH17]. In this framework, for a given position and considering a beam of light traveling in a direction, per-wavelength absorption σ_a and scattering σ_s coefficients (m^{-1}) respectively represent the proportion of radiance absorbed, or scattered, along a direction. The extinction coefficient $\sigma_t = \sigma_a + \sigma_s$ represents the total amount of energy lost due to absorption and out-scattering. Furthermore, when a scattering event occurs, the scatter direction needs to be decided based on a distribution represented by a phase function p of unit sr^{-1}.

Under strong real-time constraints, our approach relies on ray

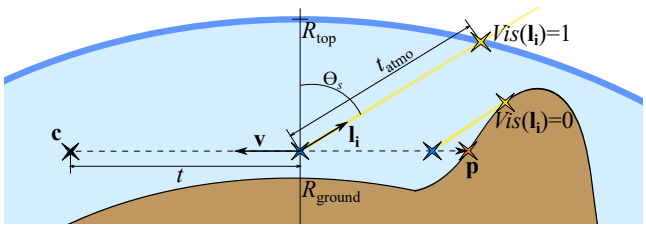

Figure 2: *Sketch illustrating how light single scattering within participating media is computed using Equation 1.*

[†] https://www.unrealengine.com.

marching to first evaluate single scattering, as illustrated in Figure 2. It assumes a set of N_{light} directional lights, e.g. a sun and a moon. It also takes into account a virtual planet with a pure diffuse response of the ground according to an albedo ρ. It involves integrating the luminance L scattered toward an observer as a function of the evaluation of the medium transmittance T, shadow factor S (*Vis* being shadowing from the planet and T from the atmosphere) as well as in-scattering L_{scat} along a path segment using

$$L(\mathbf{c},\mathbf{v}) = T(\mathbf{c},\mathbf{p})\,L_o(\mathbf{p},\mathbf{v}) + \int_{t=0}^{\|\mathbf{p}-\mathbf{c}\|} L_{\text{scat}}(\mathbf{c},\mathbf{c}-t\mathbf{v},\mathbf{v})\,\mathrm{d}t, \quad (1)$$

$$T(\mathbf{x}_a,\mathbf{x}_b) = e^{-\int_{\mathbf{x}=\mathbf{x}_a}^{\mathbf{x}_b} \sigma_{\mathbf{t}}(\mathbf{x})\|\mathbf{dx}\|}, \quad (2)$$

$$L_{\text{scat}}(\mathbf{c},\mathbf{x},\mathbf{v}) = \sigma_{\mathbf{s}}(\mathbf{x}) \sum_{i=1}^{N_{\text{light}}} T(\mathbf{c},\mathbf{x})\,S(\mathbf{x},\mathbf{l_i})\,p(\mathbf{v},\mathbf{l_i})\,\mathbf{E}_i, \quad (3)$$

$$S(\mathbf{x},\mathbf{l_i}) = Vis(\mathbf{l_i})\,T(\mathbf{x},\mathbf{x}+t_{\text{atmo}}\mathbf{l_i}), \quad (4)$$

where $\mathbf{c}$ is the view camera position, $\mathbf{v}$ is the direction toward the view for current position, $\mathbf{p}$ is the intersection surface point, t_{atmo} the ray intersection distance with atmosphere top boundary, and L_o the luminance at $\mathbf{p}$, e.g. lighting on the virtual planet's ground. $\mathbf{l_i}$ and E_i are the i^{th} light direction and illuminance (considering directional light sources).

In this paper, we compare our new ray-marching approach with results from a path tracer. Our path tracer is implemented on GPU to be able to visualize the result being refined in real time at interactive frame rate. It implements Monte Carlo integration with *delta tracking* and importance sampling within participating media [FWKH17]. It also leverages *ratio tracking* [NSJ] for faster convergence when estimating transmittance. This is considered as our ground truth.

4. Atmospheric model

The atmospheric material model we use has been described in previous papers [BN08; Bru17a]. We focus on the simulation of telluric planets, i.e. composed of a solid part made of rock or metal we will call the ground. The planet's ground and atmosphere top boundary are represented by spheres with constant radii. The variable h represents the altitude above the ground. In the case of the Earth, the ground radius is $R_{\text{ground}} = 6360$km and atmosphere top radius can be set to $R_{\text{top}} = 6460$km, representing a participating media layer of 100km. We consider the ground to be a purely diffuse material with a uniform albedo $\rho = 0.3$ [NAS]. When rendering the atmosphere's participating media, we do not consider a wide spectral representation as in [Ele09]. Instead, we focus on typical RGB-based rendering.

An atmosphere consists of several components that are all important to consider in order to achieve the look of the Earth and other planets:

- Rayleigh theory represents the behavior of light when interacting with air molecules. We assume that light is never absorbed and can only scatter around [BN08]. The phase function describing the distribution of light directions after a scattering event is $p_{\text{r}}(\theta) = \frac{3(1+\cos(\theta)^2)}{16\pi}$, where θ is the angle between incident and outgoing scattering directions.

Table 1: *Coefficients of the different participating media components constituting the Earth's atmosphere.*

Type	Scattering ($\times 10^{-6}\text{m}^{-1}$)	Absorption ($\times 10^{-6}\text{m}^{-1}$)
Rayleigh	$\sigma_{\mathbf{s}}^{\mathbf{r}} = 5.802, 13.558, 33.1$	$\sigma_{\mathbf{a}}^{\mathbf{r}} = 0$
Mie	$\sigma_{\mathbf{s}}^{\mathbf{m}} = 3.996$	$\sigma_{\mathbf{a}}^{\mathbf{m}} = 4.40$
Ozone	$\sigma_{\mathbf{s}}^{\mathbf{o}} = 0$	$\sigma_{\mathbf{a}}^{\mathbf{o}} = 0.650, 1.881, 0.085$

- Mie theory represents the behavior of light when interacting with aerosols such as dust or pollution. Light can be scattered or absorbed. The phase function is approximated using the Cornette-Shanks phase function [GK99] $p_{\text{m}}(\theta,g) = \frac{3}{8\pi}\frac{(1-g^2)(1+\cos(\theta)^2)}{(2+g^2)(1+g^2-2g\cos(\theta))^{3/2}}$ where g is the asymmetry parameter in $]-1,1[$ determining the relative strength of forward and backward scattering. By default, $g = 0.8$. Please note that it is also appropriate to use the simpler Henyey-Greenstein phase function.

For simplicity, we omit the parameters of these phase functions in the remaining equations of this paper. We also represent an isotropic phase function as $p_{\text{u}} = \frac{1}{4\pi}$.

Table 1 represents the scattering and absorption coefficients of each component [Bru17a]. Participating media following the Rayleigh and Mie theories have an altitude density distribution of $d^r(h) = e^{\frac{-h}{8\text{km}}}$ and $d^m(h) = e^{\frac{-h}{1.2\text{km}}}$, respectively. Ozone is a specific component of the Earth that has been identified as important for representing its atmosphere, since it is key to achieving sky-blue colors when the sun is at the horizon [Kut13]. Ozone does not contribute to scattering; it only absorbs light. Following Bruneton [Bru17a; Bru17b], we represent the distribution as a tent function of width 30km centered at altitude 25km, $d^o(h) = \max(0, 1 - \frac{|h-25|}{15})$.

5. Our rendering method

5.1. Discussion: observing the sky

We now describe the sky and aerial perspective visual components. It helps to justify the choices we have made when building LUTs and the use of ray marching.

Looking at Figure 3, it appears that an Earth-like sky is of low visual frequency, especially during daytime:

- Rayleigh scattering is smooth.
- The halo around the sun due to the Mie scattering phase function is also fairly smooth for realistic phase g values encountered in nature.
- Multiple scattering is a key component for rendering realistic images. As shown in Figure 3 (bottom row), it also has low visual frequency.
- Higher frequencies are visible toward the horizon because the atmosphere quickly gets denser there and thus light participates more. We must take that into account.

The main source of high frequencies within the atmosphere is due to the planet's shadow (at sunset) and shadows from mountains occluding single scattering events in Equation 3. The solution

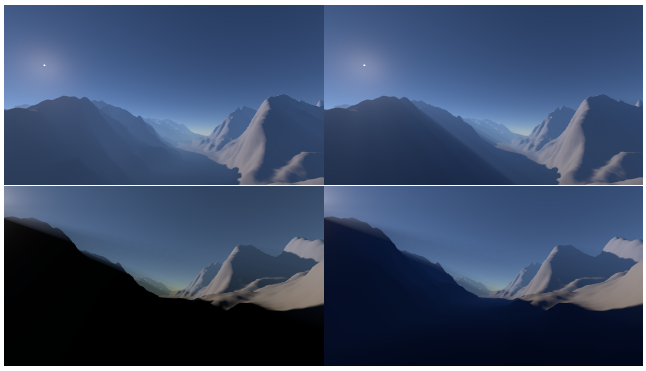

Figure 3: *Top: a scene with sun, sky and aerial perspective without (left) and with (right) volumetric shadows. Bottom: images present a ground view when multiple scattering is evaluated (right) or not (left). Note: global illumination on the terrain is disabled to make observations more visible.*

we propose can render the atmosphere with two modes: volumetric shadow disabled, i.e. taking advantage of the Sky-View LUT for faster rendering (see Section 5.3) or enabled, i.e. for a more accurate but also more expensive render (see Section 7).

5.2. Transmittance LUT

When ray marching is executed to integrate L_{scat}, the shadowing term T — representing the atmospheric medium casting onto itself — must be evaluated. However, executing a second ray march toward the sun for each single scattering sample would be expensive. To accelerate that, the function T is stored as a LUT using the same representation described in Section 4 of Bruneton and Neyret [BN08].

5.3. Sky-View LUT

Given the overall low frequency of the participating media constituting the atmosphere (see Section 5.1), it should be enough to ray march it with a low number of samples. However, doing so for each pixel can be expensive, especially at high resolution such as 4K or 8K. Given the overall low visual frequency of the sky, we should be able to render the sky at a lower resolution and upsample it to higher resolution afterward.

For a given point of view, we propose to render the distant sky into a latitude/longitude texture, oriented with respect to the camera local up vector on the planet's ground for the horizon to always be a horizontal line within it. For an example of this, see Figure 4, where the upper part represents the sky and the lower part the virtual planet ground, with the horizon in the middle. In Section 5.1, we mentioned that higher-frequency visual features are visible toward the horizon. In order to help better represent those, we apply a non-linear transformation to the latitude l when computing the texture coordinate $v \in [0,1]$ that will compress more texels near the horizon. A simple quadratic curve is used: $v = 0.5 + 0.5 * \text{sign}(l) * \sqrt{\frac{|l|}{\pi/2}}$, with $l \in [-\pi/2, \pi/2]$.

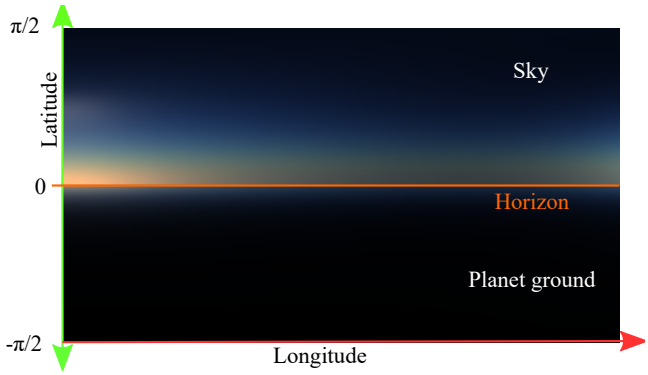

Figure 4: *The Sky-View LUT during daytime. The sun direction can be seen on the left side, where Mie scattering happens.*

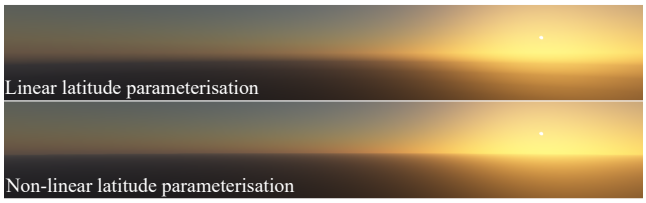

Figure 5: *The non-linear parameterization of the Sky-View LUT helps to concentrate texel details at the horizon, where it visually matters.*

This effectively compresses more pixels close to the horizon and improves the amount of detail present there. It also helps hide the fact that the atmosphere is rendered at a lower resolution, as shown in Figure 5. The sun disk is not rendered as part of that texture because of the low resolution and the non linear mapping. It is composited after applying the Sky-View LUT.

5.4. Aerial Perspective LUT

When rendering a scene, the aerial perspective effects on opaque structures (e.g. terrain, mountains, and buildings) and translucent elements (e.g. glass, fire, or other participating media such as clouds) must be rendered for consistency. Thus, similar to Hillaire [Hil16], we evaluate in-scattering and transmittance towards the camera in a volume texture fit to the view camera frustum (see Figure 6). In-scattering is stored in the RGB channels while the transmittance is stored in the A channel, as the mean of the wavelength dependent RGB transmittance.

The default resolution used in our case is 32×32 over the screen and 32 depth slices over a depth range of 32 kilometers, which is enough for most applications and games. This is the case for Epic Games' Fortnite[‡], having a world map size of 3km^2 with an Earth-like stylized atmosphere setup. If the planet's atmosphere is really dense up to a point where distant objects are less visible, then the

‡ https://www.epicgames.com/fortnite.

depth range can be brought back closer to the view point, in order to increase accuracy over short range.

The aerial perspective volume texture is applied on opaque objects as a post process after lighting is evaluated, at the same time as the Sky-View LUT is applied on screen. For transparent elements in a forward-rendering pipeline, we apply aerial perspective at the per-vertex level. This is because transparent elements are usually small in screen space relative to atmospheric visual variations.

5.5. Multiple scattering LUT

As described in Section 2, previous atmospheric rendering techniques [BN08; Ele09; Yus13] rely on iterative methods to update 3D or 4D LUTs, with one iteration per scattering order. This is an acceptable solution when rendering Earth-like atmospheres where only a multiple scattering order of 5 is required to reach realistic sky visuals. However, it quickly becomes impractical when rendering thicker atmosphere, i.e. when higher scattering orders are important for the atmosphere's look and it is thus necessary to iterate many times over the LUTs. Practically, this operation of complexity $\mathcal{O}(n)$ (where n is the scattering order) is computationally too heavy. This is especially the case when artists are constantly updating atmospheric properties to match art direction or weather changes at different times of day. The computation can be time sliced [Hil16] but this will result in update delays, which can impact the reactivity of other systems such as global illumination or reflection cube maps captured in real time.

Our goal is to propose a cheaper and instant $\mathcal{O}(1)$ method that is independent of the scattering order, to be able to evaluate the light multiple scattering contribution each and every frame without any delay. Maintaining correctness and believability for a wide range of atmosphere setups is also a requirement, as well as being able to render atmospheres across a range of devices (from mobile to high-end PC). Last but not least, we want our approach to rely on a physically based participating media parametrization and to be energy conserving.

5.5.1. Building an intuition about our approximation

Given the overall large scale, long mean free path, and smooth distribution of participating media in the atmosphere, it can be considered that the illuminance E reaching a point in space is the same for all points within a large area around it. Thus integrating luminance resulting from higher-order light scattering events around a sample point can be approximated by integrating the in-scattered light over the surrounding sphere, from neighboring points that receive the same illuminance E, while taking into account the transmittance between those points. This idea of using *global* in-scattered illuminance E as the input to evaluate multiple scattering using the *local* material data is inspired by the *dual scattering* method approximating light multiple scattering in hair [ZYWK08].

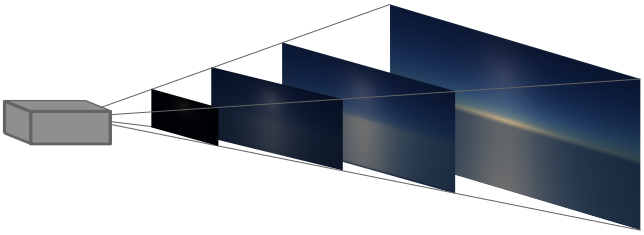

Figure 6: *The camera frustum aerial perspective LUT. This is a visualization of in-scattering for a few slices.*

When light scatters around in a medium, the distribution of scattering directions quickly becomes isotropic [JMLH01; Yan97]. For the sake of performance, we would like our multiple scattering LUT to have a low dimensionality. To this aim, we assume that light bouncing around for scattering orders greater than or equal to 2 will be achieved according to an isotropic phase function, i.e. without any preferred directions. As such, we will ignore the Mie and Rayleigh phase function setup as part of the multiple scattering approximation. We feel that this an acceptable fit considering that the Rayleigh phase function is already smooth. In order to get a better intuition about the approximation for the case of Mie scattering, we refer the reader to the analysis of BSDF shape with respect to scattering orders conducted by Bouthors [Bou08].

Furthermore, it has been shown that a correlation exists between second order scattered luminance and further scattering orders [HG13]. Thus we propose to evaluate the multiple scattering contribution in the atmosphere as a function of the second order of scattered luminance arriving at each sample point.

We build our method from these previous results, and it will be described in depth in Section 5.5.3. Here is a summary of it, together with its approximations when evaluating multiple scattering:

- Scattering events with order greater or equal to 2 are executed using an isotropic phase function p_u.
- All points within the neighborhood of the position we currently shade receive the same amount of second order scattered light.
- We compute the second scattering order contribution $\mathbf{L_{2^{nd}order}}$

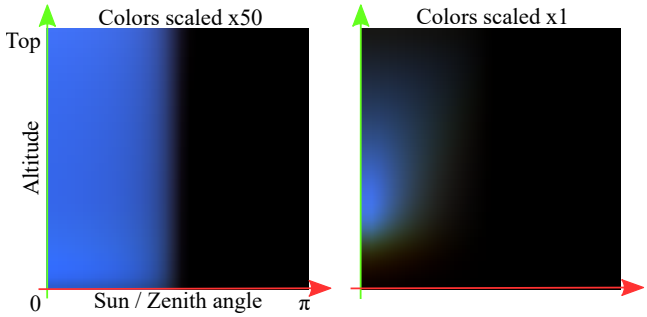

Figure 7: *Visualization of Equation 10* Ψ_{ms} *stored in multiple scattering LUTs. Left: the LUT for the Earth setup. It is broadly uniform, and scattering dominates over transmittance. Right: 50 times denser air, causing Rayleigh scattering with a modified distribution* $d^r(h) = e^{\frac{h}{20km}}$*. The contribution of multiple scattering increases with the density of the medium, until transmittance overtakes it, resulting in a drastic reduction of light reaching the ground. This is especially true when the sun is close to the horizon.*

and a transfer function $\mathbf{f_{ms}}$ (taking into account transmittance and medium variation along altitude) from the participating media around the position we currently shade.
- Finally, we compute the multiple scattering contribution $\Psi_{\mathbf{ms}}$ from these factors, simulating the infinite scattering of the second order light contribution isotropically with respect to the transfer function from neighborhood points, back to the currently shaded position.
- Visibility *Vis* is ignored when evaluating multiple scattering. This relies on the fact that light will scatter around mountains anyway, e.g. the impact of visibility is low for natural atmospheres with a large mean free path.

5.5.2. LUT parameterization

For any point in space, we want to be able to store and query the isotropic multiple scattering contribution to luminance from a LUT. Given that we consider the virtual planet to be a perfect sphere, the multiple scattering contribution to be isotropic, and the distribution of medium in the atmosphere to only vary based on altitude, we represent this LUT as a small 2D texture. The u,v parameterization in $[0,1]^2$ is:

- $u = 0.5 + 0.5\cos(\theta_s)$, where θ_s is the sun zenith angle and ω_s represents its direction.
- $v = \max(0, \min(\frac{h - R_{\text{ground}}}{R_{\text{top}} - R_{\text{ground}}}, 1))$, where the sample position $\mathbf{x_s}$ is at altitude h.

An example of such LUTs and their parameterization can be seen in Figure 7.

5.5.3. High scattering order LUT evaluation

Considering a sample point at position $\mathbf{x_s}$ and altitude h, we integrate the second order scattered luminance $\mathbf{L_{2^{nd}order}}$ towards point $\mathbf{x_s}$ (as illustrated in Figure 8 (left)) using

$$\mathbf{L_{2^{nd}order}} = \int_{\Omega_{4\pi}} L'(\mathbf{x_s}, -\omega)\, p_{\text{u}}\, \mathrm{d}\omega, \tag{5}$$

$$L'(\mathbf{x},\mathbf{v}) = T(\mathbf{x},\mathbf{p})\, L_o(\mathbf{p},\mathbf{v}) + \int_{t=0}^{\|\mathbf{p}-\mathbf{x}\|} \sigma_{\mathbf{s}}(\mathbf{x})\, T(\mathbf{x},\mathbf{x}-t\mathbf{v})\, S(\mathbf{x},\omega_{\mathbf{s}})\, p_{\text{u}}\, E_{\mathcal{I}}\, \mathrm{d}t. \tag{6}$$

In Equation 6, the L' term evaluates the luminance contribution from a single directional light with illuminance $E_{\mathcal{I}}$ and with a direction $\omega_{\mathbf{s}}$, for a position $\mathbf{x_s}$ matching the current LUT entry being built. It also contains the luminance contribution reflected from the ground through L_o (diffuse response according to albedo). $\mathbf{L_{2^{nd}order}}$ should give the second order scattered light towards point $\mathbf{x_s}$ as luminance. But it is evaluated using $E_{\mathcal{I}}$: it is a placeholder for what should be light illuminance $\mathbf{E_i}$. Though in this case it is a unitless factor $E_{\mathcal{I}} = 1$ to ensure that $\mathbf{L_{2^{nd}order}}$ does not return a luminance value, but instead acts as a *transfer function* of unit sr^{-1}, only returning luminance when later multiplied with the actual directional light illuminance. In Equation 5, L_o is also evaluated using $E_{\mathcal{I}}$ but we kept this out for simplicity.

Secondly, we integrate a unitless factor $\mathbf{f_{ms}}$ representing the transfer of energy that would occur from all of the atmospheric medium around and towards the currently shaded sample at position $\mathbf{x_s}$ as

$$\mathbf{f_{ms}} = \int_{\Omega_{4\pi}} L_f(\mathbf{x_s}, -\omega)\, p_{\text{u}}\, \mathrm{d}\omega, \tag{7}$$

$$L_f(\mathbf{x},\mathbf{v}) = \int_{t=0}^{\|\mathbf{p}-\mathbf{x}\|} \sigma_{\mathbf{s}}(\mathbf{x})\, T(\mathbf{x},\mathbf{x}-t\mathbf{v})\, 1\, \mathrm{d}t. \tag{8}$$

This is illustrated in Figure 8 (right). The directional integration over the sphere is computed as $\mathbf{f_{ms}}$, where L_f is integrated along each ray using Equation 8. It is important to skip the sampling of the shadowing term S and phase function in this equation because it is already accounted for when evaluating $\mathbf{L_{2^{nd}order}}$. Thus $\mathbf{f_{ms}}$ is a unitless normalized transfer factor of the energy integrated around and towards $\mathbf{x_s}$, in the range $[0,1]$. To help respect that range, it is recommended to use the analytical solution to the integration of Equation 8 as proposed by Hillaire [Hil15].

As mentioned above, we assume that light reaching any point around $\mathbf{x_s}$ is the same as that reaching $\mathbf{x_s}$ itself for scattering orders greater than to 2. We can use this low spatial variation assumption to evaluate the multiple scattering contribution analytically. Inspired by the dual-scattering approach [ZYWK08], we approximate the infinite multiple scattering light contribution factor $\mathbf{F_{ms}}$ as a geometric series infinite sum

$$\mathbf{F_{ms}} = 1 + \mathbf{f_{ms}} + \mathbf{f_{ms}^2} + \mathbf{f_{ms}^3} + \ldots = \frac{1}{1 - \mathbf{f_{ms}}}. \tag{9}$$

Finally, the total contribution of a directional light with an infinite number of scattering orders can be evaluated as

$$\Psi_{\mathbf{ms}} = \mathbf{L_{2^{nd}order}}\, \mathbf{F_{ms}}, \tag{10}$$

where the second order scattering contribution $\mathbf{L_{2^{nd}order}}$ is amplified by the multiple scattering transfer function $\mathbf{F_{ms}}$. The transfer function $\Psi_{\mathbf{ms}}$ (unit sr^{-1}) is thus simply multiplied with any directional light illuminance (Lux as cd.sr.m^{-2}) to retrieve the multiple scattering contribution to a pixel as luminance (cd.m^{-2}). $\Psi_{\mathbf{ms}}$ is stored in the multiple scattering LUT. For an atmosphere material setup, this LUT is valid for any point of view and light direction around the planet.

To conclude, the light scattering Equation 3 can now be augmented with our multiple scattering approximation, which gives

$$L_{scat}(\mathbf{c},\mathbf{x},\mathbf{v}) = \sigma_{\mathbf{s}}(\mathbf{x}) \sum_{i=1}^{N_{\text{light}}} \left(T(\mathbf{c},\mathbf{x})\, S(\mathbf{x},\mathbf{l_i}) p(\mathbf{v},\mathbf{l_i}) + \Psi_{\mathbf{ms}}\right)\mathbf{E}_i. \tag{11}$$

This simplification avoids a reliance on an iterative method to evaluate the multiple scattering contribution within the atmosphere. For our real-time use case, the integration of $\mathbf{f_{ms}}$ and $\mathbf{L_{2^{nd}order}}$ over the unit sphere is achieved using 64 uniformly distributed directions. For more performance details, please refer to Section 7.

6. Results

We validate our approach to atmosphere rendering by comparing it to two state of the art techniques: the model proposed by Bruneton [Bru17a] and a volumetric path tracer. We compare various scenarios and give the image root mean square error (RMSE) for each of the R, G and B channels as compared to the ground truth

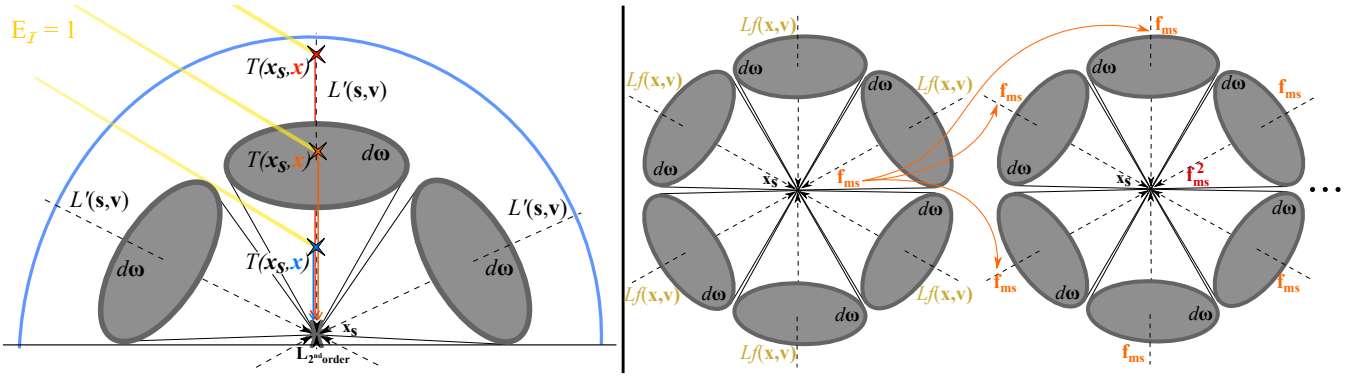

Figure 8: *Sketch presenting on the left how* $\mathbf{L}_{\mathbf{2^{nd}order}}$ *is computed from single scattering* L' *and, on the right, how* $\mathbf{F}_{ms}$ *approximates multiple scattering bounces using a normalized transfer function* f_{ms}*, corresponding to Equation 7, and assuming a sample point neighborhood receive the same amount of energy as the sample point itself, corresponding to Equation 9.*

path tracer. We show the results on a planet with a terrain using a pure-black albedo, and without sun disk, so as not to influence the RMSE measure. The code for this application is open source[§].

Firstly, we verify that our model can faithfully render the Earth's atmosphere — see Figure 9. We present views using single scattering only in order to show the difference when multiple scattering is taken into account. It also shows the three models: Bruneton (B), our model (O) and the reference path tracer (P). At daytime, (B) and (O) RSME are respectively $(1.43, 2.28, 6.07).10^{-3}$ and $(0.94, 1.74, 5.07).10^{-3}$ — both very close to the reference (P). For the sunset case, it is important to note that (B) does not faithfully represent the orange color propagated by Mie scattering. This is because we use a single RGBA 4D scattering LUT, where A represents colorless Mie scattering, rather than a solution requiring two RGB 4D scattering LUTs. This is the typical setup used in real-time applications in order to allocate less memory and to increase

§ https://github.com/sebh/UnrealEngineSkyAtmosphere.

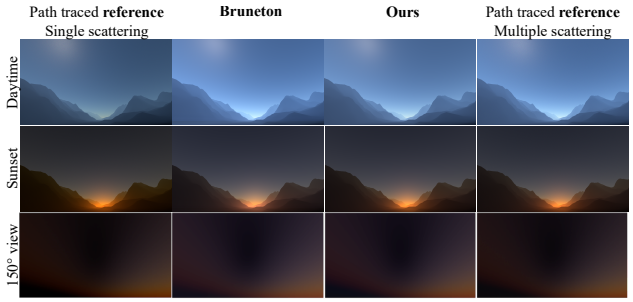

Figure 9: *Rendering Earth's atmosphere with different techniques under different conditions: daytime, sunset, and a 150 degree view of the sky with sun below the horizon revealing the shadow cast by the Earth within the atmosphere. Note: various exposures are used in this figure to ensute that visuals are readable.*

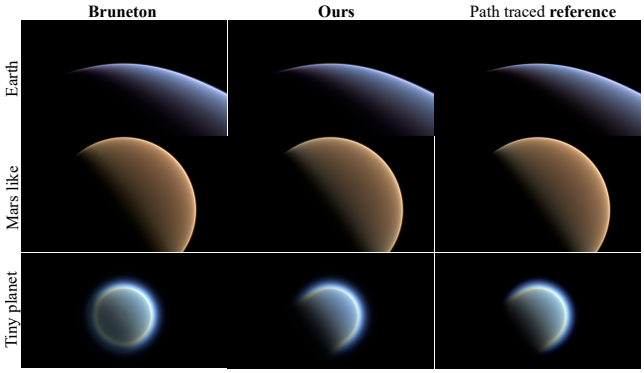

Figure 10: *Space view rendering of different planets: Earth, Mars like and a fictional tiny planet with a thick and dense atmosphere.*

performance (only 1 scattering LUT needs to be updated and it requires less bandwidth to fetch LUT data). The Mie scattering color is recovered using the trick discussed in Section 4 of Bruneton and Neyret [BN08]. It is also interesting to note that both models are able to reproduce the pale scattering color visible in the shadow cast by the Earth within the atmosphere — see bottom of Figure 9.

We also compare the accuracy of those models to achieve space views, see Figure 10. Both (B) and (O) models are able to faithfully reproduce the Earth, with respective RSMEs of $(0.58, 0.67, 1.61).10^{-3}$ and $(0.95, 0.85, 1.23).10^{-3}$, as well as a Mars-like planet atmosphere, RSMEs of $(0.87, 0.97, 0.94).10^{-3}$ and $(1.99, 0.91, 0.56).10^{-3}$. When it comes to artistic tiny planets with thick and dense atmospheres, it appears that (B) is not able to reproduce the volumetric shadowing from the planet's solid core. This is due to the LUT parameterization, which results in a lack of accuracy for small planets inherently featuring a high ground-surface curvature. This limitation of model (B) could be lifted by increasing the 4D light scattering LUT resolution, adding additional memory and computational costs.

For Earth's atmosphere, it has been reported that computing scat-

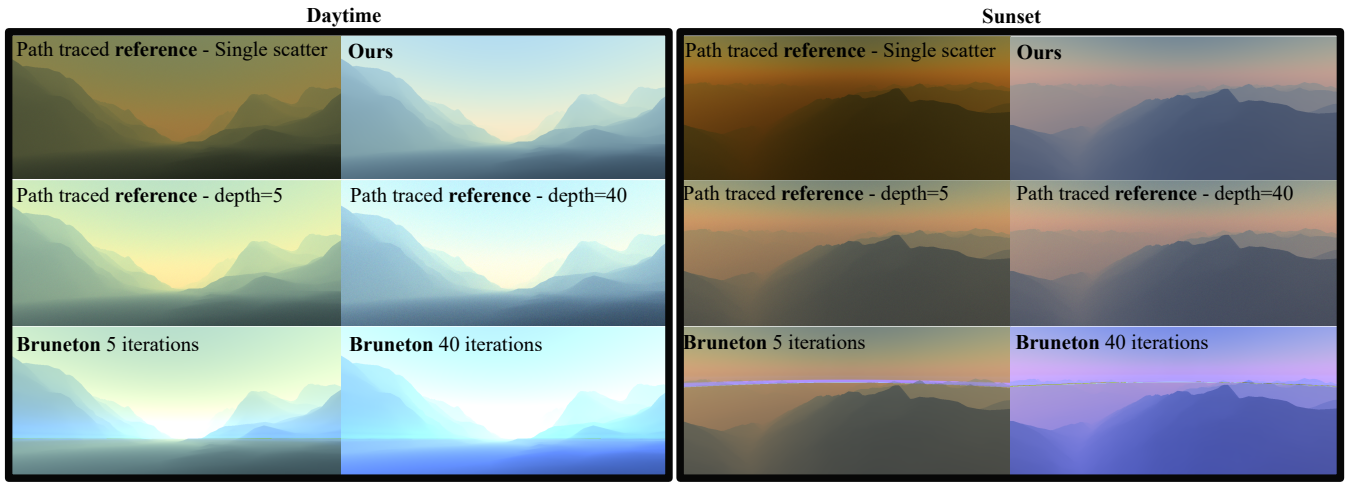

Figure 11: *Daytime on ground (left) and sunset up in the atmosphere (right) views demonstrating that it is important to consider higher scattering orders for denser participating media. Our approach is the only non-iterative technique that can approximate the ground truth.*

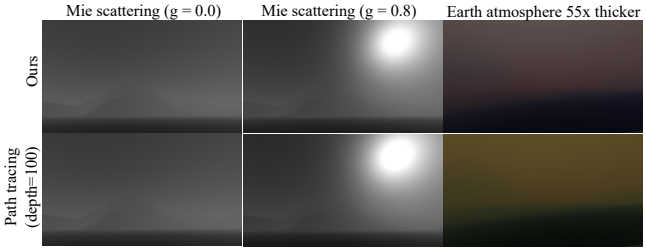

Figure 12: *Limitations of our method when the atmosphere becomes dense. Left and middle: the larger the phase g value, the less accurate it is. Right: a dense atmosphere can result in a different multiple-scattering color.*

tering only up to the 5^{th} order was enough to capture most of the energy [BN08], and we have been able to confirm this by observation. However, when control is given to artists to setup an atmosphere, the atmosphere may get denser and it then becomes important to account for higher scattering orders. While our new model (O) automatically takes that into account, it is not the case for model (B). In this case, there must be as many iterations as there are scattering orders that need to be evaluated, which quickly becomes impractical, even with time slicing. Figure 11 demonstrates that for denser atmospheres, higher-order scattering is crucial for faithfully producing the correct atmospheric color. Our model is able to represent such behavior, while model (B) fails to converge to the correct color for higher scattering orders, and even explodes numerically (Figure 11 (right)). This is likely due to precision issues when sampling the LUTs, even though we are using a 32 bit float representation for the model (B) scattering LUT, instead of a 16 bit float representation that is enough for model (O).

As shown in Figure 12, the new model (O) does have a few issues worth mentioning, each of which are due to the multiple scattering approximation:

- When using very high scattering coefficients, the *hue* can be lost or even start to drift as compared to the ground truth.
- We assume that the light scattering direction is isotropic right after the second bounce. This is in fact a approximation, which is confirmed by a comparison between our model and the reference path tracer. For Mie scattering only, with $g = 0.0$ and $g = 0.8$, RMSE is 0.0058 and 0.039, respectively.

7. Performance and Discussion

On a PC equipped with an NVIDIA 1080, the final on-screen rendering of the sky and atmospheric perspective is 0.14 milliseconds (ms) considering the daytime situation depicted in Figure 9. More detailed timings and the properties of the LUTs generated by our method are provided in Table 2. In the end, the total render time is 0.31 ms for a resolution of 1280×720. For the same view, the Bruneton model [BN08] renders in 0.22ms, but this is without all the LUTs being updated. Updating all the LUTs using the code provided [Bru17b] costs 250ms, where 99% of this cost comes from the many iterations required to estimate multiple scattering. As already shown by Hillaire [Hil16], it is possible to time slice the update over several frames. However, latency would increase when evaluating high scattering orders, and it would take a long time before any result would be available on screen.

When viewing the planet from space, as seen in Figure 10, the Sky-View LUT described in Section 5.3 becomes less accurate because a large part of it is wasted to render empty space. In this case, we seamlessly switch to simple ray marching on screen. The planet and atmosphere render time then becomes more expensive (0.33ms) resulting in a total rendering cost of 0.5ms. But this is often acceptable as planetary views focus on the planet itself, so the rendering budget is likely higher.

Our technique can scale from desktop PC to relatively old Apple iPhone 6s mobile hardware. In this case, LUT resolution and sample count can be scaled down without a huge impact on the resulting visuals. Our setup and performance differences are illustrated

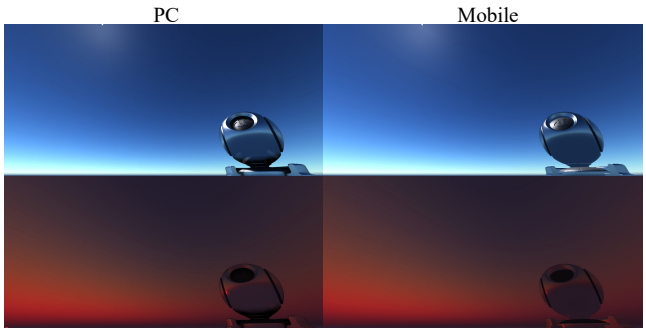

Figure 13: *Visual comparison between PC (NVIDIA 1080) and mobile (iPhone 6s) rendering of the atmosphere. Only the sky is visible at daytime (top) and at sunset with 5× higher Rayleigh scattering coefficients (bottom). Bloom, color grading and other post-processing effects have been disabled.*

in Table 2, while changes in visuals are presented in Figure 13. Visual differences, due to lower LUT quality, are not noticeable to the naked eye. Please note that we do maintain a similar transmittance LUT on both platforms as it is important for ensuring a matching look. Its quality could be further reduce on mobile if more visual differences can be traded for performance. For Epic Games' Fortnite, the total sky rendering cost was roughly 1ms on iPhone 6s.

An important visual effect to reproduce is volumetric shadowing, for example from mountains onto the atmosphere. It is not possible to use epipolar sampling [BCR*10] because the atmosphere is not a homogeneous medium. And it is also not possible to use a shadow volume approach [BN08; Hoo16] because our LUTs do not allow that kind of integral sampling over view ray paths in the atmosphere. Last but not least, these techniques cannot represent soft shadows cast by clouds: we must ray march. Similar to Valient [Val14] and Gjoel [GS16], we recommend using per-ray sample jittering and reprojection to combine samples from previous frames. Jittering can be done according to blue noise [GS16]

Table 2: *Performance for each step of our method, as measured on a PC (NVIDIA 1080) and a mobile device (iPhone 6s).*

PC

LUT	Resolution	Step count	Render time
Transmittance	256×64	40	0.01ms
Sky-View	200×100	30	0.05ms
Aerial perspective	32^3	30	0.04ms
Multi-scattering	32^2	20	0.07ms

Mobile (iPhone 6s)

LUT	Resolution	Step count	Render time
Transmittance	256×64	40	0.53ms
Sky-View	96×50	8	0.27ms
Aerial perspective	$32^2 \times 16$	8	0.11ms
Multi-scattering	32^2	20	0.12ms

Figure 14: *Volumetric shadows from the atmosphere, from left to right: path-traced single scattering, path-traced multiple scattering (depth = 5) and our real-time approach using ray marching and cascaded shadow maps.*

and reprojection can automatically be achieved via a temporal anti-aliasing (TAA) approach [Kar14]. This is illustrated in Figure 14. Using such an approach requires a sample count that is content dependent. In this example, we use 32 samples, which causes the sky and atmosphere rendering time to go up to 1.0ms. To reduce this cost, it is also possible to trace at a lower resolution and temporally reproject and upsample the result. This has already been used, with great results, in a few game engines [Bau19; EPI18]. Results with volumetric shadows are shown in Figure 1.

Furthermore, the multiple-scattering LUT we propose can also accelerate path tracing of the atmosphere participating media, if the approximations described in Section 5.5 and Figure 12 are acceptable. In this case, only single scattering events need to be sampled, e.g. using delta tracking [FWKH17]. When such an event occurs, the traced path can be stopped immediately, at which point the single scattering contribution is evaluated using next event estimation and the contribution from the remaining scattering orders can be evaluated using the multiple scattering LUT. When using this approach with our reference GPU path tracer, the cost for a 720p frame goes down from 0.74ms to 0.29ms for daytime with 5 scattering orders (path depth), as seen in Figure 9. The cost also goes down from 7.9ms to 0.6ms for daytime with 50 scattering orders, as seen in Figure 11.

8. Conclusion

In summary, our method can render sky and atmosphere from many view points efficiently in real time while constantly updating the LUTs, with light multiple scattering simulated, but without requiring cumbersome iterative computations per scattering orders. This is important for lighting artists to be able to achieve their vision and follow a project's art direction, while simulating time of day and changing weather at the same time. We have shown that it gives accurate visual results and, even when it drifts from ground truth due to dense atmosphere or strong anisotropic phase function, the result remains plausible. Because it is physically based and energy conserving, it does not explode. Furthermore, it can be used to accelerate path tracing applications that render sky and atmosphere.

9. Future work

Future work could involve investigating ways to improve the accuracy of the lookup table for anisotropic phase functions and also to support spatially varying atmospheric conditions. We believe it is important at some point to switch to spectral rendering in order to improve the accuracy of the method [EK10]. Last but not least, we

believe that rendering real-time sky and atmosphere using a path tracer coupled with a denoiser is a promising research avenue.

Acknowledgments

We would like to thank the anonymous reviewers for the useful comments, as well as the entire rendering team at Epic Games for reviewing and proofreading the paper, especially Krzysztof Narkowicz, Charles de Rousiers, Graham Wihlidal and Dmitriy Dyomin. We would also like to thank Jean-Sebastien Guay, Jordan Walker, Ryan Brucks, Sjoerd de Jong and Wiktor Öhman for providing level art and evaluating the technique. Lastly, we would like to thank Stephen Hill for proofreading the paper.

References

[Bau19] BAUER, FABIAN. "Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution". *Advances in Real Time Rendering, ACM SIGGRAPH 2019 Courses*. 2019 2, 9.

[BCR*10] BARAN, ILYA, CHEN, JIAWEN, RAGAN-KELLEY, JONATHAN, et al. "A Hierarchical Volumetric Shadow Algorithm for Single Scattering". *ACM Trans. Graph.* 29.6 (2010), 178:1–178:10 9.

[BN08] BRUNETON, ERIC and NEYRET, FABRICE. "Precomputed Atmospheric Scattering". *Proceedings of Eurographics*. 2008, 1079–1086 2–5, 7–9.

[Bou08] BOUTHORS, ANTOINE. "Realistic rendering of clouds in real-time". PhD thesis. Université Joseph Fourier, 2008. URL: http://evasion.imag.fr/~Antoine.Bouthors/research/phd/ 5.

[Bru17a] BRUNETON, ERIC. "A Qualitative and Quantitative Evaluation of 8 Clear Sky Models". *IEEE Transactions on Visualization and Computer Graphics* 23.12 (2017), 2641–2655 2, 3, 6.

[Bru17b] BRUNETON, ERIC. *Precomputed Atmospheric Scattering*. 2017. URL: https://github.com/ebruneton/precomputed_atmospheric_scattering 2, 3, 8.

[dCK17] De CARPENTIER, GILIAM and KOHEI, ISHIYAMA. "Decima Engine: Advances in Lighting and AA". *Advances in Real Time Rendering, ACM SIGGRAPH 2017 Courses*. New York, NY, USA: ACM, 2017 2.

[EK10] ELEK, OSKAR and KMOCH, PETR. "Real-time spectral scattering in large-scale natural participating media". *Proceedings of the Spring Conference on Computer Graphics (SCCG)*. 2010, 77–84 9.

[Ele09] ELEK, OSKAR. "Rendering Parametrizable Planetary Atmospheres with Multiple Scattering in Real-time". *CESCG* (2009) 2, 3, 5.

[EPI18] EPICGAMES. *Unreal Engine 4.19: Screen percentage with temporal upsample*. March 2018. URL: https://docs.unrealengine.com/en-US/Engine/Rendering/ScreenPercentage/index.html 9.

[FWKH17] FONG, JULIAN, WRENNINGE, MAGNUS, KULLA, CHRISTOPHER, and HABEL, RALF. "Production Volume Rendering". *ACM SIGGRAPH 2017 Courses*. 2017 2, 3, 9.

[GK99] GARY E., THOMAS and KNUT, STAMNES. "Radiative transfer in the atmosphere and ocean". *Cambridge Univ. Press* (1999) 3.

[GS16] GJOEL, MIKKEL and SVENDSEN, MIKKEL. "Low Complexity, High Fidelity - INSIDE Rendering". Game Developers Conference. 2016 9.

[HG13] HOLZSCHUCH, NICOLAS and GASCUEL, JEAN-DOMINIQUE. "Double- and Multiple-Scattering Effects in Translucent Materials". *IEEE Computer Graphics and Applications* (2013), 66–76 5.

[Hil15] HILLAIRE, SÉBASTIEN. "Physically Based and Unified Volumetric Rendering in Frostbite". *Advances in Real Time Rendering, ACM SIGGRAPH 2015 Courses*. 2015 6.

[Hil16] HILLAIRE, SÉBASTIEN. "Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite". *SIGGRAPH 2016 Course: Physically Based Shading in Theory and Practice*. 2016 2, 4, 5, 8.

[Hoo16] HOOBLER, NATHAN. "Fast, Flexible, Physically-Based Volumetric Light Scattering". Game Developers Conference. 2016 2, 9.

[HW12] HOSEK, LUKAS and WILKIE, ALEXANDER. "An Analytic Model for Full Spectral Sky-dome Radiance". *ACM Trans. Graph.* 31.4 (2012), 95:1–95:9 2.

[JMLH01] JENSEN, HENRIK WANN, MARSCHNER, STEPHEN R., LEVOY, MARC, and HANRAHAN, PAT. "A Practical Model for Subsurface Light Transport". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 2001, 511–518 5.

[Kar14] KARIS, BRIAN. "High-Quality Temporal Supersampling". *Advances in Real-time Rendering in Games Part I, ACM SIGGRAPH 2014 Courses*. 2014, 10:1–10:1 9.

[Kut13] KUTZ, PETER. *The Importance of Ozone*. 2013. URL: http://skyrenderer.blogspot.se/2013/05/the-importance-of-ozone.html 3.

[LdR14] LAGARDE, SEBASTIEN and de ROUSIERS, CHARLES. "Moving Frostbite to PBR". *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2014 Courses*. 2014 2.

[NAS] NASA. *Earth Fact Sheet*. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html 3.

[NDN96] NISHITA, TOMOYUKI, DOBASHI, YOSHINORI, and NAKAMAE, EIHACHIRO. "Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 1996, 379–386 2.

[NSJ] NOVÁK, JAN, SELLE, ANDREW, and JAROSZ, WOJCIECH. "Residual Ratio Tracking for Estimating Attenuation in Participating Media". *ACM Trans. Graph.* 33.6 (), 179:1–179:11 3.

[NSTN93] NISHITA, TOMOYUKI, SIRAI, TAKAO, TADAMURA, KATSUMI, and NAKAMAE, EIHACHIRO. "Display of the Earth Taking into Account Atmospheric Scattering". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 1993, 175–182 2.

[ONe07] O'NEIL, SEAN. "Accurate Atmospheric Scattering". *GPU Gems 2*. 2007 2.

[PSS99] PREETHAM, A. J., SHIRLEY, PETER, and SMITS, BRIAN. "A Practical Analytic Model for Daylight". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 1999, 91–100 2.

[Val14] VALIENT, MICHAL. "Making Killzone Shadow Fall Image Quality into the Next Generation". Game Developers Conference. 2014 9.

[Wen07] WENZEL, CARSTEN. "Real time atmospheric effects in game revisited". Game Developers Conference. 2007 2.

[Yan97] YANOVITSKIJ, EDGARD G. *Light Scattering in Inhomogeneous Atmospheres*. Springer-Verlag Berlin Heidelberg, 1997 5.

[Yus13] YUSOV, EGOR. "Outdoor Light Scattering". Game Developers Conference. 2013 2, 5.

[ZYWK08] ZINKE, ARNO, YUKSEL, CEM, WEBER, ANDREAS, and KEYSER, JOHN. "Dual Scattering Approximation for Fast Multiple Scattering in Hair". *ACM Trans. Graph.* 27.3 (2008), 32:1–32:10 5, 6.