Universidad Zaragoza
1542

Master Thesis

Physically-Based Sky and Atmosphere Rendering in Real-Time

Author

Fernando García Liñán

Supervisor

Adolfo Muñoz Orbañanos

Master in Robotics, Graphics and Computer Vision
Escuela de Ingeniería y Arquitectura
2023

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura
Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Fernando García Liñán ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de Estudios de la titulación de Máster Universitario en Robótica, Gráficos y Visión por Computador (Título del Trabajo)

Physically-Based Sky and Atmosphere Rendering in Real-Time

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 27 de enero de 2023

Firmado por GARCIA LIÑAN FERNANDO - ***8906** el día 27/01/2023 con un certificado emitido por AC FNMT Usuarios

Fdo: Fernando García Liñán

Abstract

In this Master Thesis, we propose a physically-based method to render the atmosphere of Earth in real time from any arbitrary viewpoint. Atmosphere rendering is a difficult problem to solve due to its inherent complexity. The physics equations that describe the scattering of light feature multiple nested integrals that must be computed numerically or approximated. Moreover, the atmospheric medium is strongly dependent on the wavelength of incoming light, which makes spectral rendering a requirement to faithfully represent the colors of sky. Our main contribution relies on an approximation to spectral rendering, where instead of computing dozens of spectral samples, only four carefully chosen samples are used to evaluate the final color. This makes our method exceptionally cheap and accurate, as it is able to produce images that are perceptually indistinguishable from those generated by a fully-fledged spectral renderer, at a fraction of the computational cost. We also propose an approximation to multiple light scattering that significantly reduces the computational cost of evaluating the light transport equations. We tackle the issue of developing an atmospheric model that is highly configurable and able to represent a wide variety of atmospheric conditions, while still being physically-based and suitable for real-time applications. Finally, we implement our method on Shadertoy to prove its high portability and ease of implementation, and we demonstrate how the results compare to a ground truth path tracing simulation.

Acknowledgements

I would like to thank my supervisor, Adolfo Muñoz, for all his help and advice during the entire project. I would also like to offer my special thanks to Adrián Jarabo, for all his insightful comments and suggestions.

Lastly, I would like to thank my family, for giving me the opportunity to pursue the things I am interested in and supporting me all the way.

Contents

1

Introduction

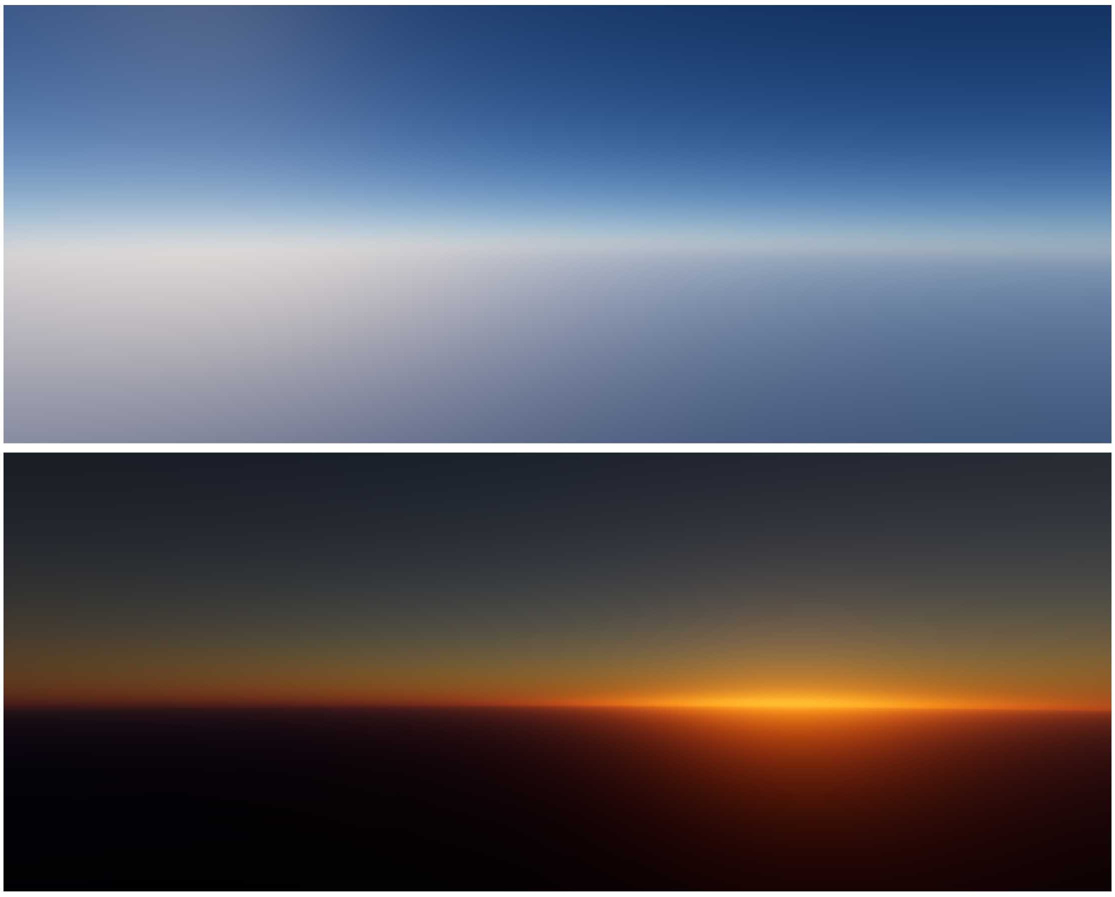

Figure 1.1: Renders obtained using our method.

The atmosphere is one of the most salient features of realistic and believable outdoor scenes. Applications such as films, games and flight simulators make use of these realistic environments to provide a successful and meaningful experience.

Unfortunately, simulating the atmosphere is no trivial task, which explains why rendering the sky is an ongoing problem in computer graphics and new techniques are still being researched today.

The complex interactions that occur within the atmospheric medium are responsible for the blue color of the sky, as well as the golden-red sunrises and sunsets. The physics behind these interactions is well understood, hence it is possible to simulate them using light scattering theory. Given infinite computational power, we are only limited by the quality of the available data, so obtaining accurate measurements is key to developing a successful atmospheric model. For example, in the case of the Earth's atmosphere, research in the field of atmospheric sciences is a great source for this kind of data. It is also helpful that the field is thriving nowadays due to the prevailing necessity of monitoring our atmosphere as closely as possible to study and prevent climate change.

In the real world we do not have unlimited computational power though. Light scattering physics can be solved offline given enough time and resources, but not at interactive framerates. Light scattering within the atmosphere requires several nested integrals to solve, especially if we allow light to scatter multiple times. Any integral is numerically tough to solve, and much more difficult to solve in real-time.

In this work we propose a method to render the atmosphere of Earth in real-time using a physically-based model. Our main contribution is a way to correctly take into account the strong wavelength dependency of the atmospheric medium. Most state of the art atmosphere rendering techniques entirely forgo spectral rendering in favour of traditional RGB rendering, which yields incorrect results. Our method is able to render images that are completely indistinguishable from offline spectral renders, at a tiny fraction of their cost. We also propose a multiple scattering approximation that takes advantage of the low density of the Earth's atmosphere, as well as the relatively simple reflectance response of the Earth's surface.

In the end our model is able to dynamically render any state of the Earth's atmosphere at any time of day, from any point of view above the surface, including space views. The atmospheric properties can be updated in real time, allowing us to simulate a wide range of atmospheric conditions dynamically.

We contribute an implementation of our approach as a Shadertoy shader to showcase its simplicity and portability. We have also developed a custom GPU-accelerated volumetric path tracer to obtain ground truth data for our simulations.

2

Related Work

In this section we review existing literature on atmosphere rendering. It is important to note that we focus on the rendering aspect of atmospheric simulation, i.e. we only take into consideration the literature that explicitly tackles the problem from a computer graphics perspective. We will not be citing research in the domains of physics and atmospheric sciences, except as sources for our atmospheric model data. For an in-depth comparison of several atmospheric models, we refer to the work of Bruneton [Bru16].

One of the first attempts in rendering the atmosphere involved ray marching the atmosphere from the view point [Nis+93], which allowed Nishita et al. to render both ground and space views. This model was limited to single scattering, but was later revised to account for 2nd order scattering using precomputed tabulated values [Nis+96]. To reduce the cost of ray-marching, Preetham et al. [PSS99] proposed an analytical model fitted on real measurements. Similarly, Hosek and Wilkie [HW12] improve upon this idea by building an analytical model that is fitted on the results of a path tracer, fixing some of the issues with the Preeham model. These analytical models are very fast to evaluate, making them ideal for real-time applications. They have consequently gained a lot of popularity over the years, although they have some glaring issues, the main one being that they are limited to ground views. Their parametrization is also too straightforward and inflexible, making it impossible to render more complex atmospheric conditions.

Considering the limitations of analytic approaches, O'Neil [ONe05], in an attempt to make Nishita's approach run on a GPU in real-time, proposed deferring as many calculations as possible to the vertex shader, as well as fitting the optical depth along arbitrary rays to a function that is very fast to evaluate. Bruneton and Neyret [BN08], instead of fitting to a function, use precomputed lookup tables to store the transmittance and in-scattering. Elek [Ele09] proposed removing one of the dimensions of the lookup tables to simplify the calculations, in exchange for the ability to correctly reproduce the Earth's shadow on the atmosphere in twilight conditions.

Schüler [Sch12] provided an alternative to precomputations by approximating the light transport integrals to reach a closed-form solution. However, this approach is quite complex to implement and generally performs worse than the precomputed alternative. It is also limited to atmospheres with exponential fall-off, so it does not scale to more complex atmospheric models.

More recently, Hillaire [Hil20] proposed a way to render the atmosphere in real time using small lookup tables with low dimensionality, including multiple scattering. The reduced size and dimensions greatly improve performance, as well as simplify the resulting implementation. Guimera et al. [GGJ18] propose a physically-based atmosphere with more parameters than previous models, using research from atmospheric sciences. The model takes into account the time of the year and the particular location of the observer by dynamically changing the atmospheric properties, such as the ozone concentration and the type of aerosols present.

In the topic of spectral rendering, Elek [EK10] proposes a method to perform brute force spectral rendering of the atmosphere in real time, essentially doing the same thing that offline spectral renderers do but using ray marching and an approximation similar to [BN08] for multiple scattering. Spectral renderers need to know the emission spectra and the surface albedos at each wavelength, but this kind of information is not usually available for common surface types. Peters [Pet+19] tackles this problem, which is effectively the inverse of our problem (obtaining spectral samples from RGB triplets).

We leverage some of the rendering techniques described in [BN08] and [Hil20], and combine them with the atmosphere model described by [GGJ18]. Our own contributions (multiple scattering and spectral rendering) complete our method.

3

Participating Media Rendering

Rendering participating media is an important aspect of most modern renderers. The atmosphere of Earth is a particular case of participating media, and is characterized by its overall low density and predictable behaviour based on altitude. In this chapter we will be providing an overview of the concepts required to understand volume rendering, and how they can be applied to the atmosphere.

3.1. Volume Properties

A volume can be described as a collection of particles, ranging from atoms and molecules to any particle size. As a photon travels through the volume, it may collide with the particles that make up the volume. Therefore, it is common to describe participating media with the collision coefficients, which represent the probability of a photon colliding per unit distance traveled inside the volume. The physical unit of a collision coefficient is inverse length (SI units m^{-1}). It is also important to note that the collision coefficients are not only spatially varying, but they also depend on the wavelength of the incoming photon. In this chapter we will not be expressing this dependency explicitly in the equations to avoid overloading the notation.

We can define a volume using the absorption and scattering coefficients, together with the phase function and emission:

Absorption. An absorption event occurs when a photon is absorbed by the volume, i.e. the energy of the photon is added to the kinetic energy of the atoms in the volume. The probability of such event is given by the absorption coefficient μ_a.

Scattering. When a scattering collision occurs, the photon is absorbed by the volume and then re-emitted into a different direction defined by the phase function. The probability of this event is given by the scattering coefficient μ_s.

It is worth noting that we usually separate scattering events into two distinct types: in-scattering and out-scattering. In-scattering accounts for the increased radiance due to scattering from other directions, while out-scattering accounts for the reduced radiance due to scattering into other directions.

Phase function. The phase function models the angular distribution of scattered light. It is usually modeled as a function of the angle between the incoming and outgoing light directions (before and after the scattering event).

Emission. Volumes can emit radiance by themselves, a classic example being fire. However, we will not be discussing this property of volumes as it does not apply to the atmospheric medium.

Another useful parametrization of the collision coefficients are the extinction coefficient μ_t:

$$\mu_t = \mu_a + \mu_s \tag{3.1}$$

and the single-scattering albedo α:

$$\alpha = \frac{\mu_s}{\mu_t} \tag{3.2}$$

3.2. Radiative Transfer Equation

The distribution of radiance in participating media is described by the radiative transfer equation (RTE) [Cha60]. The RTE is a differential equation that models how as a radiance beam travels, it loses energy to absorption and out-scattering, and gains energy by emission and in-scattering. The differential form can be directly used in finite element methods such as radiosity, but we are interested in its integral form, also known as the volume rendering equation (VRE). For a given ray $x_t = x - t\omega$, we define the radiance L arriving at point x from direction ω as:

$$L(x,\omega) = T(x,x_d)\,L_d(x_d,\omega) + \int_{t=0}^{d} T(x,x_t)\,\sigma_t(x_t)\,L_s(x_t,\omega)\,dt \tag{3.3}$$

where $x_d = x - d\omega$ is the position where the ray ends at a surface, L_d is the incident radiance at the end of the ray, σ_t is the extinction coefficient, T is the transmittance, and L_s is the in-scattering. Note how there is no emission term as it does not apply to the atmospheric medium.

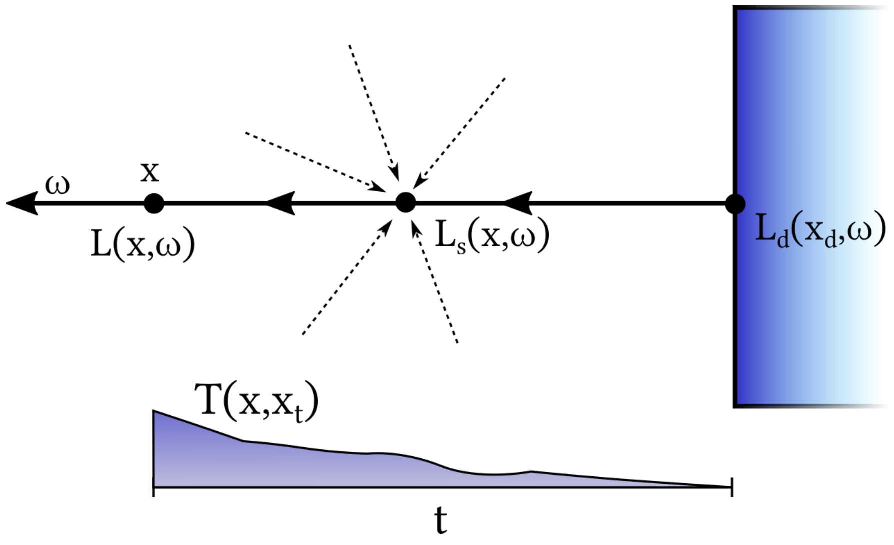

Figure 3.1: Visualization of the volume rendering equation (VRE).

The transmittance T is the fraction of radiance at point x_b reaching point x_a, and gives the probability of a free flight. It is given by the Beer-Lambert law:

$$T(x_a, x_b) = \exp\left(-\int_{x_a}^{x_b} \sigma_t(x)\, \|dx\|\right) \tag{3.4}$$

The in-scattering L_s is the contribution from the out-scattering of all other directions ω' at x, into direction ω:

$$L_s(x, \omega) = \int_{S^2} \alpha(x)\, f_p(x, \omega, \omega')\, L(x, \omega')\, d\omega' \tag{3.5}$$

where S^2 denotes the spherical domain around position x (4π steradians), f_p denotes the phase function, α is the single-scattering albedo, and L is the radiance described in the VRE (equation 3.3), making the formula recursive.

Two important concepts we will be using all throughout this work are single scattering and multiple scattering. Single scattering, as its name implies, represents light that is scattered exactly one time on its path from a surface to the eye. Multiple scattering, on the other hand, represents light that undergoes multiple scatter events. In later chapters we will see how separating the in-scattered radiance into these two parts will become very useful.

The VRE integrals can be evaluated using a Monte Carlo estimator, leading to what is known as volumetric path tracing. Due to the high computational cost, this solution is usually favored by offline renderers only. In Chapter 5 we will be discussing some approximations that allow solving the VRE in real time.

3.3. Volumetric Path Tracing

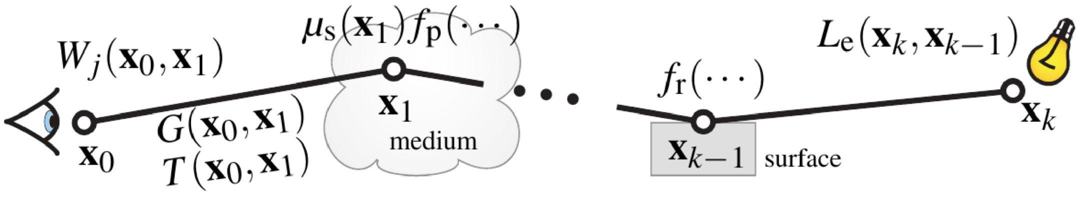
$W_j(\mathbf{x}_0, \mathbf{x}_1)$
$\mu_s(\mathbf{x}_1) f_p(\cdots)$
$\mathbf{x}_0$
$G(\mathbf{x}_0, \mathbf{x}_1)$
$T(\mathbf{x}_0, \mathbf{x}_1)$
$\mathbf{x}_1$
medium
$f_r(\cdots)$
$\mathbf{x}_{k-1}$
surface
$L_e(\mathbf{x}_k, \mathbf{x}_{k-1})$
$\mathbf{x}_k$

Figure 3.2: Complete path from an emitter to the sensor, interacting with multiple surfaces and participating media. [Nov+18]

Even though in this work we concentrate on real-time applications, we need to study offline rendering to obtain a ground truth to compare against. As we mentioned in the previous section, we can use Monte Carlo methods to solve the VRE. It is not fast, especially for highly scattering media like the atmosphere, but it gives the correct solution for complex cases like ours. In our case, volumetric path tracing is used to provide a ground truth solution to the rendering equation.

3.4. Ray Marching

An alternative way to solve the VRE integrals is through ray marching, also known as volumetric ray casting or volumetric ray tracing. To integrate incoming light along a ray due to in-scattering, we break down the volume that the ray passes through into small volume elements, and later combine the contribution of each individual element to arrive at a final solution.

Mathematically, we are approximating the integral by doing a Riemann sum of a finite number of volume segments. The size of each segment, which we call the step size, will determine the quality of the approximation. As the step size gets smaller and smaller, the approximation approaches its real value. This is an advantageous trait for our use-case, as we can choose a bigger or smaller step size depending on the device where we are going to be rendering the atmosphere, hopefully reaching a good balance between quality and performance.

The most problematic issue with ray marching is evaluating the in-scattering integral. Since we are evaluating the integrals numerically, and the VRE is inherently recursive, we would need to evaluate an infinite number of in-scattering integrals to arrive at a final radiance value. This is obviously not possible in practice, so most ray

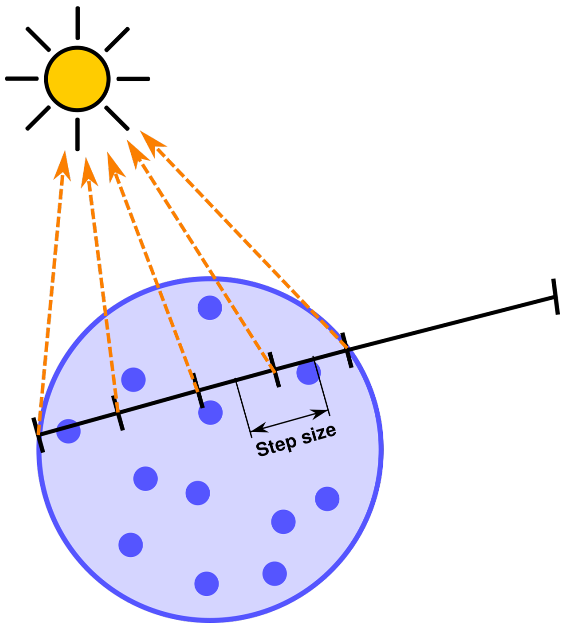

Figure 3.3: Ray marching along a ray in small, regular steps.

marching solutions settle for evaluating the in-scattering integral just once, which effectively yields the single scattering contribution. In our case, we only calculate the single scattering using ray marching, and approximate the recursive integral with a closed-form solution, as we will see in Chapter 4.

4

Physically-Based Atmosphere Model

The atmospheric model described in this section is very similar to the Guimera model [GGJ18]. We simulate a perfectly spherical planet of radius $R_{earth} = 6360\,\text{km}$, with an atmosphere that extends to the Kármán line, $100\,\text{km}$ above sea level. The Earth surface is considered to be purely diffuse, with a varying albedo ρ depending on the type of surface around the observer. Using spheres to model the Earth and its atmosphere allows us to simulate the phenomenon where the Earth itself casts a visible shadow on the atmosphere. This would not be possible in simpler models such as the plane-parallel model commonly used in atmospheric sciences.

To model the atmosphere of Earth, or any participating media in general, we need to specify the sources for the absorption and scattering coefficients mentioned in section 3.1:

- Rayleigh scattering from molecular oxygen (O_2) and nitrogen (N_2). They are responsible for the blue color of the sky.
- Absorption due to ozone (O_3).
- Scattering and absorption due to aerosols. Aerosols are relatively large suspended particles from different sources (dust, condensed water, human-made pollution...), that vary significantly depending on the position in the planet.

There exist other sources for absorption and scattering within the atmosphere, such as carbon dioxide (CO_2), water vapour, or nitrogen dioxide (NO_2). However, we find that these constituents do not improve image fidelity enough to justify their inclusion in a computer graphics context. For the same reason, we also choose to ignore other phenomena such as polarization.

4.1. Rayleigh Scattering

Rayleigh scattering explains the scattering of light by particles much smaller than the wavelength of the incoming radiation. In the case of the Earth's atmosphere, the main molecules responsible for this type of scattering are (O_2) and nitrogen (N_2). This type of scattering gives the sky its characteristic blue color at daytime, as well as the yellowish to reddish hue when the Sun is close to the horizon.

The distribution of these molecules is fairly constant across the globe, i.e. it is not very dependent on the latitude and longitude of the observer. However, it is strongly dependent on the altitude of the observer, as the concentration of molecules decreases exponentially with altitude.

The molecular scattering coefficient μ_s^m is given by:

$$\mu_s^m(\lambda, h) = N(h)\, \sigma_s^m(\lambda) \tag{4.1}$$

where N is the molecular number density at a given pressure and temperature, or altitude, h, and σ_s^m is the total Rayleigh scattering cross section per molecule. N can be obtained from different sources, but we use data from the U.S. Standard Atmosphere model [Uni76]. For σ_s^m, we use the formulation proposed by Bucholtz [Buc95]:

$$\sigma_s^m(\lambda) = \frac{24\pi^3(n_s^2 - 1)^2}{\lambda^4 N_s^2 (n_s^2 + 2)^2} \left(\frac{6 + 3\rho_n}{6 - 7\rho_n} \right) \tag{4.2}$$

where λ is the wavelength, n_s is the index of refraction for standard air, N_s is the molecular number density ($2.54743 \times 10^{19}\,\mathrm{cm}^{-3}$) for standard air, and ρ_n is the depolarization factor, a term that accounts for the anisotropy of the air molecule and varies with wavelength.

Although $N(h)$ can be stored on a lookup table and queried at runtime, we chose to fit the data to a function. We found absolutely no decline in image fidelity, and performance is improved due to the removal of memory fetches on the GPU. To simplify even further, we store the molecular scattering coefficient at sea level $\mu_s^m(\lambda, 0)$, and multiply by a fitted function:

$$\mu_s^m(\lambda, h) = \mu_s^m(\lambda, 0)\, e^{-ah^b} \tag{4.3}$$

where $a = 0.07771971$ and $b = 1.16364243$ for an h expressed in kilometers.

For the phase function, we use the classic Rayleigh phase function:

$$f_p(\omega, \omega') = \frac{3}{16\pi}(1 + \cos^2\theta) \tag{4.4}$$

where θ is the dot product between ω and ω'. There exists a more precise form proposed by Chandrasekhar [Cha60] which accounts for the effect of depolarization anisotropy, but the improvement is imperceptible.

4.2. Ozone

Ozone (O_3) is responsible for absorbing most of the Sun's ultraviolet radiation. It concentrates at the ozone layer, a region of the Earth's stratosphere, although is also present in lower quantities in the rest of the atmosphere. Since ozone only absorbs light, we define the following absorption coefficient:

$$\mu_a^o(\lambda, h) = C_a^o(h)\,\sigma_a^o(\lambda) \tag{4.5}$$

where $\sigma_a^o(\lambda)$ is the cross section obtained from the work by Gorshelev et al. [Gor+14], and $C_a^o(h)$ is the concentration measured by Ramanathan and Kulkarni [RK53] and Dütsch [Düt74]. The measurements by Dütsch change with the time of the year, so we introduce the month as another parameter to our atmospheric model.

4.3. Aerosols

Aerosols are fine solid particles or liquid droplets suspended in air. They can be natural, like fog and dust, or artificial, like smoke, soot and pollution. The wide variety of aerosols present in the atmosphere makes it difficult to categorize them and study their optical properties. However, we can attempt to categorize them based on geographical location. Based on Zimmermann et al. [Zim+89] and Jaenicke [Jae93], we model the following geographical environments: Background, Desert-Dust, Maritime-Clean, Maritime-Mineral, Polar-Antarctic, Polar-Artic, Remote-Continental, Rural and Urban.

We assume that aerosols are solid and spherical, with optical properties defined by their index of refraction η and radius. Using Lorentz-Mie theory [Hul57], we can calculate the optical properties of a single aerosol, namely their scattering and absorption cross sections (σ_s^a and σ_a^a). The required parameters to apply Mie theory were obtained from the work of D'Almeida et al. [DKS91]. We could also calculate their phase function, but we chose to use the simpler Henyey-Greenstein phase function [HG41]:

$$f_p(\omega, \omega') = \frac{1 - g^2}{4\pi(1 + g\cos\theta)^2} \quad g \in [-1, 1] \tag{4.6}$$

where θ is the dot product between ω and ω', and g is an asymmetry parameter that controls the amount of forward and backward scattering.

The same process can be used to obtain the scattering and absorption cross sections of all aerosol types. For a more in-depth explanation, we again refer to the work by Guimera et al. [GGJ18].

The concentration of aerosols is not constant along the vertical axis, so following [AIA99] we define the vertical distribution of aerosols C_t^a as

$$C_t^a(h) = C_t^a(0)\left(\exp\left(\frac{-h}{H_p}\right) + \frac{C_b}{C_t^a(0)}\right) \tag{4.7}$$

where $C_t^a(0)$ is the concentration at ground level, and both H_p and C_b are parameters specific for each type of aerosol, also obtained from [AIA99].

Finally, with the cross sections and the vertical distribution of the aerosols defined, we can express the scattering μ_s^a and absorption μ_a^a coefficients as

$$\mu_s^a(\lambda, h) = \sigma_s^a(\lambda)\, C_t^a(h) \tag{4.8}$$

$$\mu_a^a(\lambda, h) = \sigma_a^a(\lambda)\, C_t^a(h) \tag{4.9}$$

5

Our Approach

In this chapter we describe how we solve the volume rendering equation (VRE) in real-time, as well as how we apply the atmospheric model described in Chapter 4 to the generic participating media framework from Chapter 3.

5.1. Transmittance LUT

The first step towards a real-time solution to the VRE is to find a way to compute the transmittance T in equation 3.4 along any arbitrary ray inside the atmosphere. A real-time application must be able to do this in $O(1)$. Unfortunately, there is no general, trivial solution to this equation applied to the atmospheric medium, since the air density decreases exponentially with altitude, and the altitude above a spherical planet varies non-linearly along straight paths.

If we assume that we are dealing with a perfectly spherical atmosphere, we notice that the transmittance only depends on two parameters due to spherical symmetry: the altitude h of the observer and the zenith angle θ (angle between the vertical direction and the Sun direction) [ONe05]. The transmittance can thus be pre-computed and stored in a lookup table.

If we also assume that the air density decreases exponentially with altitude, an alternative approach would be to solve the transmittance integral analytically. Such integral is known in the physics community as the Chapman function. However, this analytical formulation breaks down in non-exponential atmospheres, has some numerical instabilities, and can be quite complex to implement. We also found that the lookup table works faster and is easier to implement in practice, so we opted for the lookup table approach.

We implement the table as a small 2D texture. When the atmospheric properties change (a change in the Sun's direction does not count), we run a shader that numerically integrates equation 3.4 by ray marching through the atmosphere for every pair of h and $\cos\theta$ values. This procedure is identical to [BN08].

5.2. Single Scattering

The VRE is a recursive integral that accumulates radiance across multiple scattering events. As we mentioned before in Section 3.4, if we consider the demanding constraints of real-time rendering, it is useful to split the VRE into two parts: one that is responsible for the first scattering event inside the atmosphere, and another for the rest of the scattering events. To render the first scattering event (or single scattering), we change the in-scattering term of the VRE (equation 3.5) to the following:

$$L_s(x, \omega) = \alpha(x)\, f_p(x, \omega, \omega_s)\, E_s \tag{5.1}$$

where E_s is the spectral solar irradiance, and ω_s is the Sun's direction. This effectively removes the recursion and allows the integral from equation 3.3 to be solved, again, using ray marching. For every pixel in the final image, we take the corresponding view direction ω_c and we traverse a ray originating from the camera position x_c, obtaining the radiance value $L(x_c, \omega_c)$ for that pixel.

The overall low frequency of the participating media constituting the atmosphere can be taken advantage of by rendering to a texture instead of directly to the screen. This texture can then be upscaled without a significant loss in detail, considerably reducing the computational cost of ray marching, especially if the screen has a very high resolution. This approach, as described by Hillaire [Hil20], renders the sky to a texture using an equirectangular projection from the point of view of the camera, so that the horizon is always an horizontal line within the texture. The same paper also proposes using a non-linear mapping of the texture coordinates to dedicate more texels to the horizon, where most of the high frequency detail is present. We opted to just increase the height of the texture instead.

5.3. Multiple Scattering

The rest of the scattering events are much more difficult to simulate in real time. As we mentioned multiple times, the problem is inherently recursive, so the complexity of the algorithm will always be $O(n)$ unless we make some approximations. Ideally, our multiple scattering approximation has to be cheaper and have $O(1)$ complexity, while still maintaining image fidelity.

One contributor to multiple scattering, especially around daytime, is the diffuse response from the ground. We find that the second order scattered radiance is enough to approximate the total contribution from a purely diffuse surface with albedo ρ, and that further scattering orders have minimal impact. We also notice that when light scatters around in a medium, the distribution of scattering directions quickly becomes isotropic [Jen+01; Yan97], prompting us to use the isotropic phase function $f_p = \frac{1}{4\pi}$ to simplify even further. Applying the in-scattering equation 3.5, we define L_g as the second order scattered radiance from the ground towards point x:

$$L_g(x) = \int_\Omega \frac{\alpha(x)}{4\pi} L'(x, -\omega)\, d\omega \tag{5.2}$$

$$L'(x, \omega) = T(x, p)\, L_{\text{diff}}(p, \omega) \tag{5.3}$$

$$L_{\text{diff}}(x, \omega) = T(x, x_s)\, \frac{\rho}{\pi}\, E_s\, (n \cdot \omega_s) \quad x_s = x - t\omega_s \tag{5.4}$$

where Ω denotes the portion of the spherical domain around x subtended by the Earth, p is the intersection ground surface point corresponding to each direction in the integral, n is the surface normal at p, t is the ray parameter for the atmospheric boundary or nearest intersection, E_s is the solar spectral irradiance, and ω_s is the Sun's direction.

We still have to integrate over the sphere around x to take into account all possible directions that the ground radiation could be coming from. We could solve this numerically, but we make another approximation to arrive at a completely closed-form solution. If the camera is sufficiently close to the ground, which is always the case for views inside the atmosphere, we can assume that L' will be very close to constant across all directions. Since L' no longer depends on direction, and the rest of the terms are constant as well, we are left with:

$$L_g(x) = \frac{\alpha(x)\, T(x, p)\, T(p, x_s)\, \rho\, E_s\, (n \cdot \omega_s)\, \Omega(x)}{4\pi^2} \qquad x_s = x - t\omega_s \tag{5.5}$$

Ω is the solid angle subtended by the Earth at point x, which is a distance d from the

Earth's center:

$$\Omega(x) = 2\pi\left(1 - \frac{\sqrt{d^2 - R_g^2}}{d}\right) \quad d \geq R_g \tag{5.6}$$

where R_g is the radius of the Earth.

Even though the ground albedo is a major contributor to multiple scattering, we also have to take into account the scattering coming from the atmosphere itself. This type of scattering is specially important during twilight conditions, where the longer average ray length is responsible for more scattering events. This value is also the hardest to obtain in real time as we cannot make many simplifying assumptions as in the case of ground albedo.

We opt to make an aggressive approximation by establishing that the multiple scattering coming from the atmosphere at any point within it is directly proportional to the inverse fourth power of the wavelength. We theorize that, since the distribution of scattering directions becomes isotropic very quickly, Rayleigh scattering is going to play a key role in the color of multiple scattering.

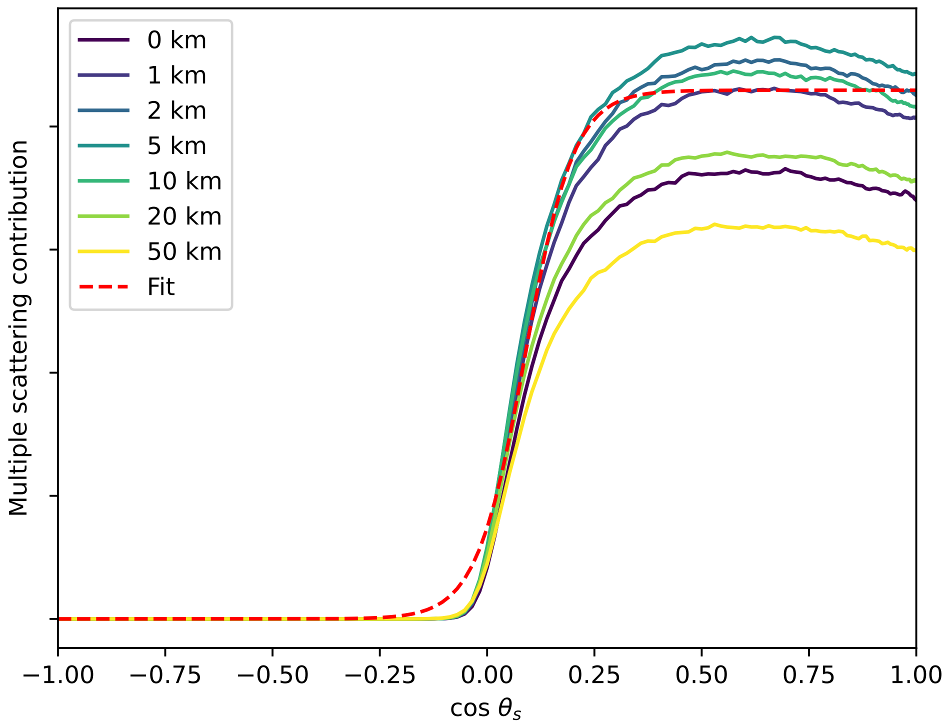

Figure 5.1: Multiple scattering contribution plotted against the dot product of the Sun's direction and the zenith, for points at different altitudes above sea level. The red dashed line corresponds to our fit.

The contribution from multiple scattering is also proportional to the dot product between the zenith direction and the Sun's direction $\cos\theta_s$, due to the fact that points that are close to being occluded by the Earth are less likely to receive light from surrounding points. We found that a simple linear relationship is not enough to represent this behaviour, as can be seen on Figure 5.1, so a logistic function is used instead.

The entire approximation is formally defined as follows:

$$L_{ms}(\theta_s, \lambda) = E_s\,\lambda^{-4}\left(\frac{k}{1 + ae^{-b\cos\theta_s}}\right) \tag{5.7}$$

where k, a and b are constant values obtained by fitting this function to the plot of the ground truth multiple scattering contribution for all possible $\cos\theta_s$ values at an altitude of 10 kilometers above sea level. From the plot in Figure 5.1 we can tell that any other altitude would have worked just as well, except near sea level or the top of the atmosphere. A point near the surface receives significantly less in-scattering, because there is less air below compared to higher altitudes, so light does not scatter as much below it. Same reasoning can be applied to points near the top of the atmosphere, but above instead of below.

It is important to note that this approximation breaks down with higher density atmospheres, as multiple scattering becomes more and more relevant and its contribution starts to play a key role in the final appearance of the sky. Luckily for us, in the case of the Earth's atmosphere, single scattering carries most of the weight.

In general, the approximation is good enough to maintain image fidelity with a negligible performance impact, especially when compared to evaluating the recursive radiance integral multiple times. We get the following final expression for the rendering equation:

$$L(x,\omega) = T(x,x_d)\,L_d(x_d,\omega) + \int_{t=0}^{d} (T(x,x_t)\,\sigma_s(x_t)\,f_p(x,\omega,\omega_s)\,E_s + L_g(x) + L_{ms}(n\cdot\omega_s))dt \tag{5.8}$$

The equation we have just derived is very dependent on the wavelength of the incoming radiation, as we mentioned in Chapter 3. In the next section we will take a look at our solution for taking this dependency into account.

5.4. Spectral Rendering

An often ignored aspect of sky and atmosphere rendering is the strong wavelength dependency of the atmospheric medium. This is caused by the optical properties of both molecular scatterers and ozone being highly variable depending on the wavelength of light interacting with them, as we have seen on Chapter 4. To take this relationship into account in a light transport simulation, offline renderers usually resort to uniform spectral sampling or more elaborated spectral sampling methods like Hero Wavelength Sampling [Wil+14], while real-time renderers often choose to completely ignore this behaviour and perform traditional RGB rendering.

In uniform spectral sampling, we pick n_λ uniformly distributed samples in a portion of the electromagnetic spectrum (usually the visible spectrum). For each pixel in the rendered image, n_λ rays are fired from the camera, each with an associated wavelength, making all the interactions between the rays and the scene wavelength-dependent. Consequently, compared to traditional RGB rendering, spectral rendering requires substantially more data to represent the optical properties of the scene, which can be challenging to obtain in most cases. Luckily, in the case of the atmosphere of Earth, there is plenty of available data thanks to atmospheric sciences research.

Spectral rendering can sometimes offer more accurate results since it is able to capture optical phenomena that would otherwise be impossible to recreate, like iridescence. However, spectral rendering is quite sensitive to which portion of the electromagnetic spectrum is used to perform the simulation. For example, if we have a scene that contains a single light source that does not emit any electromagnetic radiation in the near-UV range, and we simulate light transport for the near-UV range in an attempt to improve image fidelity, we are not going to observe any improvements. Due to cases like this, widening the range of simulated wavelengths or increasing the number of sampled wavelengths will usually, but not always, yield a better final result.

The prohibitively high computational cost of offline spectral rendering techniques remains the most significant problem though, making them unsuitable for real-time applications. We propose a method to greatly decrease the overall computational cost while still offering similar results to a fully-fledged spectral renderer.

5.4.1. Tristimulus Color from Spectral Data

For a given spectral power distribution (SPD), $L(\lambda)$, the XYZ tristimulus values can be computed as:

$$\begin{aligned} X &= \int_\lambda \bar{x}(\lambda)L(\lambda)d\lambda \\ Y &= \int_\lambda \bar{y}(\lambda)L(\lambda)d\lambda \\ Z &= \int_\lambda \bar{z}(\lambda)L(\lambda)d\lambda \end{aligned} \tag{5.9}$$

where $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ are the CIE standard observer functions, also known as the CIE color matching functions (CMFs). [CIE32] The integrals are computed over the visible spectrum. If it has not been specified otherwise, we will consider the visible spectrum to range from 390nm to 780nm.

As discussed in previous sections, $L(\lambda)$ does not have a closed form analytical expression that represents it. Instead, the function exists as discrete samples. Numerical integration must then used to solve the integrals in equation 5.9. It is important to note that the spans and spacings of the standard observer functions must match those of the spectral power distribution. Resampling and interpolation is used to accomplish this.

Now it is clear why n_λ needs to be sufficiently high to correctly approximate $L(\lambda)$. If too few samples are selected, the result of the numerical integration will not be accurate. We are looking for a successful approximation which uses the least amount of spectral samples while still maintaining visual fidelity. We also take advantage of the fact that we are integrating the product of $L(\lambda)$ and the CMF, thus some samples of $L(\lambda)$ will be weighted down significantly by the CMF, making their impact on the final image unimportant.

5.4.2. Numerical Integration

We can perform numerical integration of the integrals in equation 5.9 using a Riemann sum. For clarity purposes, we will use the tristimulus value X, although the same criteria also applies to Y and Z. The visible spectrum is divided into n_λ rectangular partitions that together form a region that is similar to the region below the function inside the integral. The height of each rectangle corresponds to the value of $L(\lambda) \cdot \bar{x}(\lambda)$ at any λ belonging to the partition, while the width corresponds to the wavelength interval covered by the partition. The sum of the areas of the partitions approximates the integral we are trying to compute. This idea is shown in Figure 5.2.

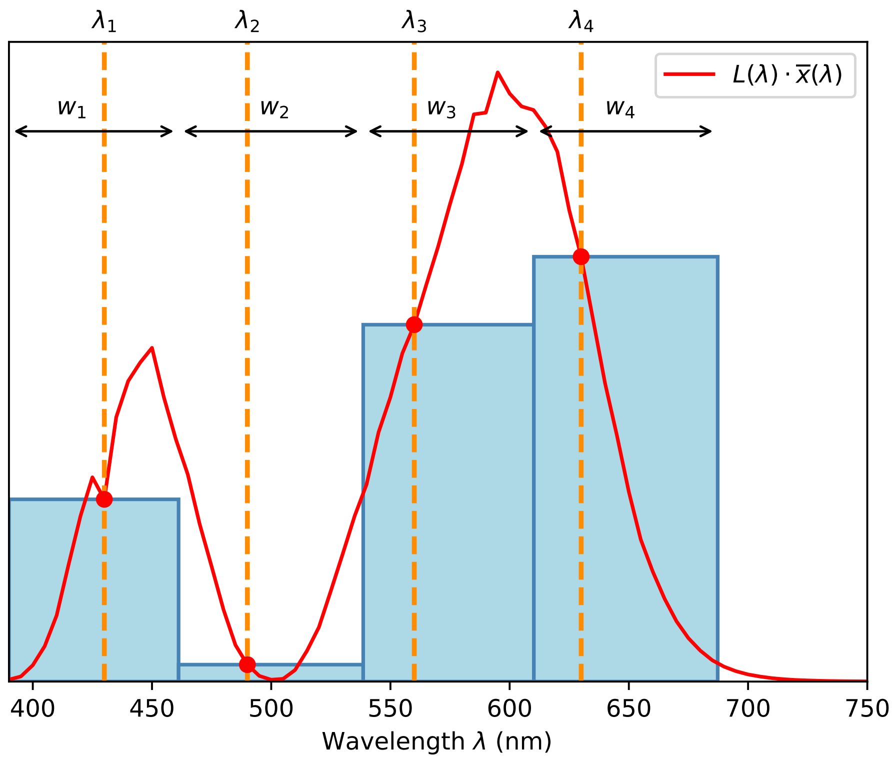

Figure 5.2: Riemann sum of four partitions, each described by λ_i and w_i. The sum of the areas of all four partitions approximates the integral of $L(\lambda) \cdot \bar{x}(\lambda)$. The same idea can be applied to the Y and Z values.

The Riemann sum for each tristimulus value can be expressed as:

$$\begin{aligned} X &= \sum_{i=1}^{n_\lambda} \bar{x}_i L_i w_i \\ Y &= \sum_{i=1}^{n_\lambda} \bar{y}_i L_i w_i \\ Z &= \sum_{i=1}^{n_\lambda} \bar{z}_i L_i w_i \end{aligned} \tag{5.10}$$

where w_i is the wavelength interval $\Delta\lambda$ covered by each partition, which we will call the weight of the partition. Going back to uniform spectral sampling, w_i is constant in this case as the interval covered by each partition remains constant. In our solution we allow the intervals to change, giving us four more degrees of freedom.

These summations of products can also be written as a matrix multiplication, which is specially useful in order to perform the computations efficiently on the GPU:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \bar{x}_1 w_1 & \bar{x}_2 w_2 & \dots & \bar{x}_n w_n \\ \bar{y}_1 w_1 & \bar{y}_2 w_2 & \dots & \bar{y}_n w_n \\ \bar{z}_1 w_1 & \bar{z}_2 w_2 & \dots & \bar{z}_n w_n \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_n \end{bmatrix} \tag{5.11}$$

5.4.3. Optimization

We have seen how using more spectral samples tends to yield a more accurate image, but due to limitations inherent to real-time rendering, we cannot sample dozens of wavelengths like offline renderers do. First, we have to settle for a number of wavelengths to sample.

We have chosen four ($n_\lambda = 4$) to take advantage of the SIMD instructions available in most modern hardware. The data paths and registers supporting SIMD operations are usually at least 128-bits wide in both CPUs and GPUs, corresponding to four full 32-bit floats. Our algorithm greatly benefits from SIMD vectorization since the light transport simulation is identical for each wavelength; only the data varies.

Four wavelengths might not seem sufficient, but we found that if we choose the best four wavelengths for which to sample $L(\lambda)$, and the correct weight for each partition w_i, we are able to have all the perks of a spectral renderer with just a single SIMD instruction.

We are left with four partitions, each with a representative wavelength λ, and a weight w, totalling eight parameters. The resulting spectrum to tristimulus conversion is:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \bar{x}_1 w_1 & \bar{x}_2 w_2 & \bar{x}_3 w_3 & \bar{x}_4 w_4 \\ \bar{y}_1 w_1 & \bar{y}_2 w_2 & \bar{y}_3 w_3 & \bar{y}_4 w_4 \\ \bar{z}_1 w_1 & \bar{z}_2 w_2 & \bar{z}_3 w_3 & \bar{z}_4 w_4 \end{bmatrix} \begin{bmatrix} L(\lambda_1) \\ L(\lambda_2) \\ L(\lambda_3) \\ L(\lambda_4) \end{bmatrix} \tag{5.12}$$

To find the values of the eight parameters that offer the best results when compared to a ground-truth offline simulation, we pose an optimization problem. Mean squared error (MSE) is used as the loss function due its simplicity and adequate results, and is defined as follows:

$$\mathcal{L} = \frac{1}{n} \sum_{\phi,\theta_v,\theta_s} (L_{\phi,\theta_v,\theta_s} - \hat{L}_{\phi,\theta_v,\theta_s})^2 \tag{5.13}$$

where $L_{\phi,\theta_v,\theta_s}$, $\hat{L}_{\phi,\theta_v,\theta_s}$, are the ground truth and approximated radiance values at a given view direction (ϕ, θ_v) and Sun elevation angle θ_s, and n is the total number of combinations of ϕ, θ_v, θ_s.

This pixel-wise loss function asserts that all view directions and Sun elevation angles are equally important, thus it is important to note that their distribution will affect the optimization. We have deliberately skipped higher Sun elevation angles to put more emphasis on sunrise/sunset conditions, in order to avoid overfitting to the clearly dominant blue color present in daylight. We can observe how the relative value of each wavelength changes with Sun elevation in figure 5.3.

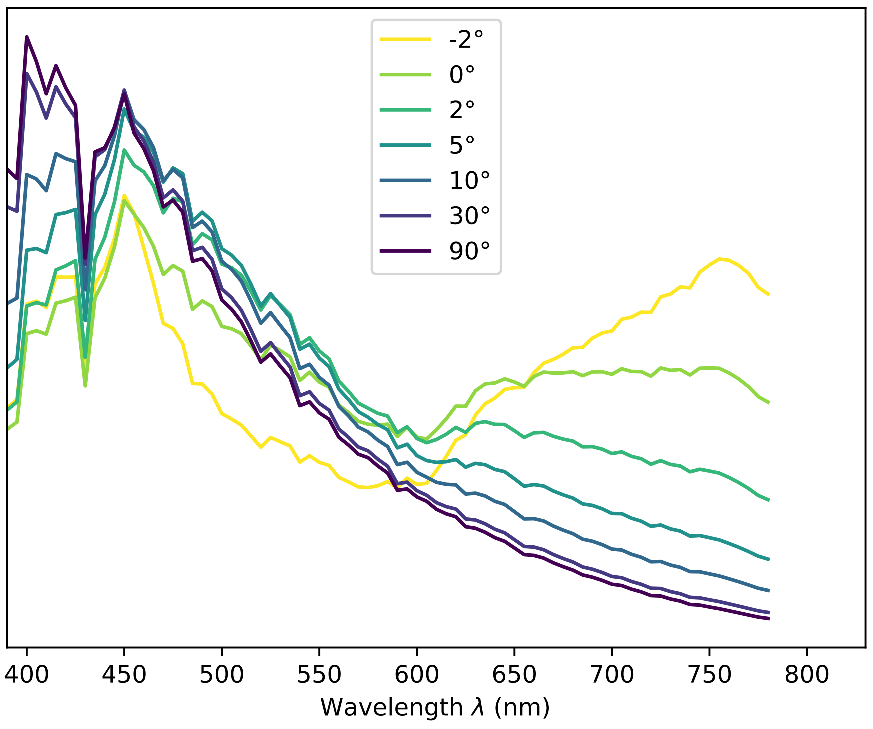

Figure 5.3: Normalized spectral radiance for several Sun elevations at sea level. The values are averaged across many view directions. Note how long wavelengths (yellows, oranges, reds) become more relevant the lower the Sun elevation angle (sunrise and sunset).

For the optimization process itself, first we performed a brute force grid search of the eight parameters (λ_1, λ_2, λ_3, λ_4, w_1, w_2, w_3, w_4), and then we refined the results using the Nelder–Mead method [NM65].

It is important to note that this optimization only works for a particular atmospheric model. If we optimize for a model, but attempt to use the resulting optimized parameters with another model, the end result will not be correct. In our case, we use the Rayleigh scattering and ozone absorption parts of our atmospheric model, and ignore the aerosols. Aerosols can vary dramatically with the viewer's position, making them unsuitable for obtaining a general solution to the eight optimization parameters. Luckily for us, they are not as dependent on wavelength as molecules due to their relatively large size, so we found no practical difference between optimizing with and without aerosols.

5.4.4. Rendering

During actual real-time rendering we compute the spectral radiance $L(\lambda)$ as seen in Chapters 3 and 5 for each of the optimized wavelengths (λ_1, λ_2, λ_3, λ_4), and then apply equation 5.12. The optimization process we described in the previous section has been performed offline, which means that, at runtime, our method solely requires four spectral samples and a matrix multiplication. Now it is clear why we chose four spectral samples instead of any other number. We can compute $L(\lambda)$ for the four samples at the same time thanks to SIMD, which makes our method as cheap as traditional RGB rendering.

The end result is a highly accurate render comparable to fully-fledged spectral renderers that require dozens of spectral samples. The image fidelity is also greatly improved in contrast to state to the art algorithms that incorrectly use three spectral radiance values directly as an RGB triplet, as we will see in Chapter 6.

6
Results

The correctness of our approach is validated by comparing it against the results from a volumetric path tracer. The path tracer uses delta tracking for sampling vertices in the path, and next-event estimation for connecting path's vertices with the Sun. The transmittance has been estimated with ratio tracking [NSJ14]. Our first version of the path tracer was based on an OpenGL shader to make the rendering times as painless as possible. Later versions moved to CUDA to be able to render offscreen, but ultimately the code was moved to the CPU to leverage the multithreading capabilities of servers with many cores. Path tracing is an inherently incoherent operation because each ray can go a different direction or collide with different parts of the atmosphere, which degrades GPU performance very severely. This makes CPU rendering a very attractive option, and the one we went for in our final renders. For rapid iteration we still used the GPU version though.

We present fisheye renderings of our volumetric path tracer and our real-time approach, together with the relative square error (RSE) in Figure 6.1. We observe that our approach is very successful at approximating the unbiased results from the path tracer, only failing at very low Sun elevation angles where our multiple scattering approximation starts to break down. Still, the error is very small and only noticeable if we pay very close attention.

We can also observe how our model correctly simulates the projected shadow of the Earth onto the atmosphere at lower Sun elevation angles. The specially tricky gold/redish tones present at sunset and sunrise are also correctly simulated, and practically indistinguishible from the path traced ground truth.

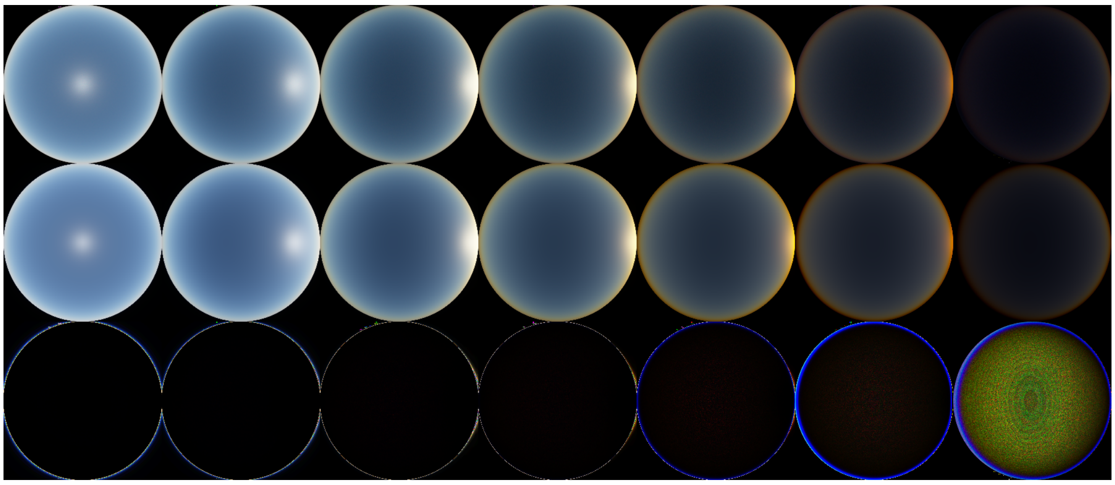

Figure 6.1: Fisheye skydome rendering of the final radiance value obtained with the ground truth path tracer (1st row), our approach (2nd row), and the Relative Squared Error (RSE) between the 1st and 2nd rows (3rd row). The models output a linear sRGB value, which is then tonemapped with $1 - e^{-kL}$ and gamma corrected. From left to right: Sun elevation of 90°, 30°, 10°, 5°, 2°, 0°, -2°.

To quantitatively evaluate the physical accuracy of our model, we have also compared the absolute luminances of our approach and the volumetric path tracer in Figure 6.2. We can observe some differences in the luminance around the horizon, especially during daylight, whereas the halo around the Sun and the horizon during twilight conditions have correct photometric values.

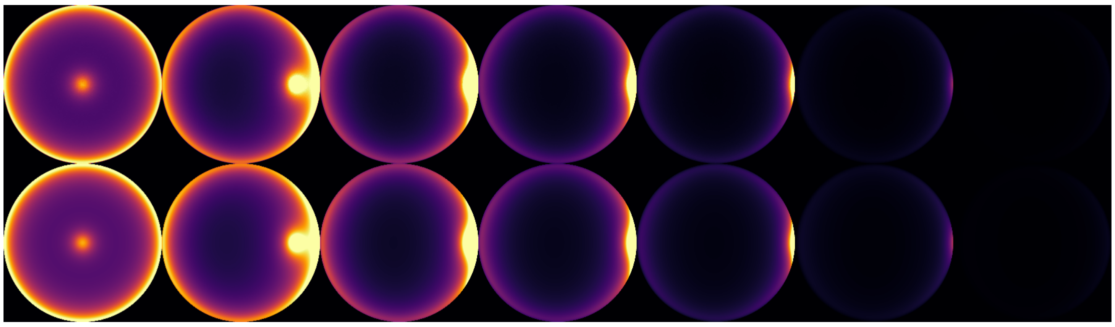

Figure 6.2: The absolute luminance in $cd \cdot m^{-2}$ of each model. First row: ground truth path tracer. Second row: our approach. From left to right: Sun elevation of 90°, 30°, 10°, 5°, 2°, 0°, -2°.

Since our main contribution in this work is a method to faithfully take into account the strong wavelength dependency of the atmospheric medium, we show how traditional RGB rendering would fare under identical atmospheric conditions in Figure 6.3. In RGB rendering we take three spectral samples at 680nm for red, 550nm for green, and 440nm for blue. These samples are used directly as a linear sRGB color, which is then tonemapped and gamma corrected. This approach is incorrect, as noted by other works [Bru16], although quite popular in the state of the art. Our approach provides much more realistic results with the same computational cost. Also note that in the traditional RGB rendering case we multiply the final radiance value by an arbitrary factor to match the absolute luminance. They do not match by default because the theoretically correct RGB color is the result of integrating with the CIE color matching functions as described in Section 5.4.1, but in RGB rendering we use the three spectral samples directly as an RGB value, bypassing the color matching integral entirely.

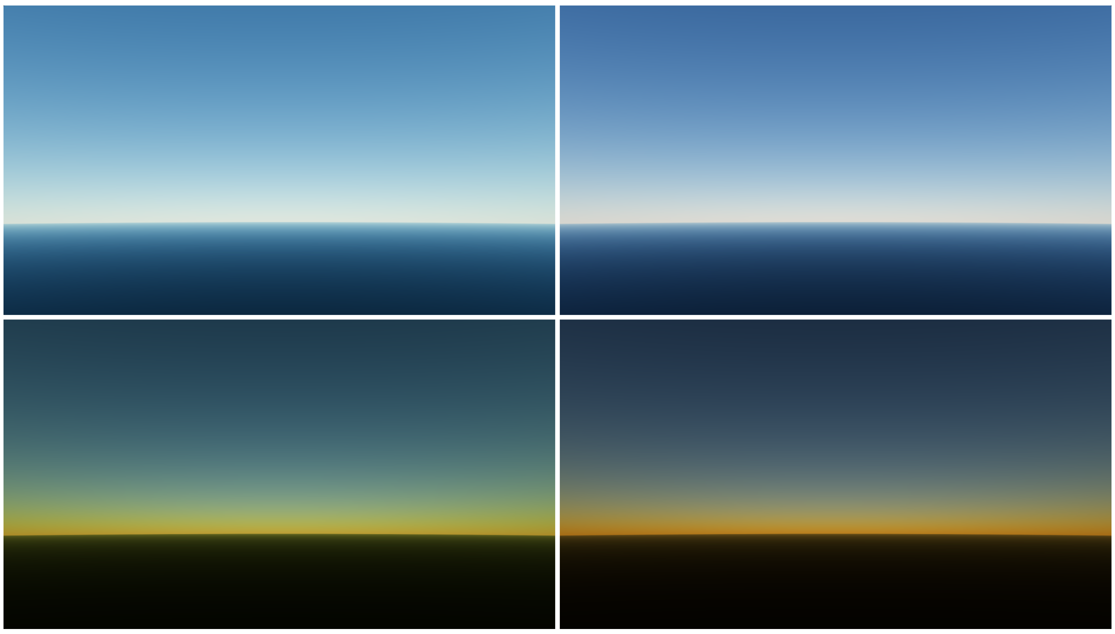

Figure 6.3: Traditional RGB rendering with 3 wavelengths (680nm, 550nm, 440nm) (left), compared to our spectral rendering method (right). To highlight the differences between both approaches we have removed aerosols as they do not depend on wavelength as much as other constituents.

To show our atmospheric model in action, we render two different atmospheres, each with one of the aerosol types we described in Section 4.3. The results can be seen in Figure 6.4. It is clear that aerosols have a huge impact on the overall appearance of the atmosphere, that is why we consider Guimera's atmospheric model to be superior than other real-time atmospheric models.

Lastly, we would like to evaluate if our model is able to represent the atmosphere of Earth from any arbitrary viewpoint. Ground or near-ground viewpoints have already been tested in the figures above, so we show a space view in Figure 6.5.

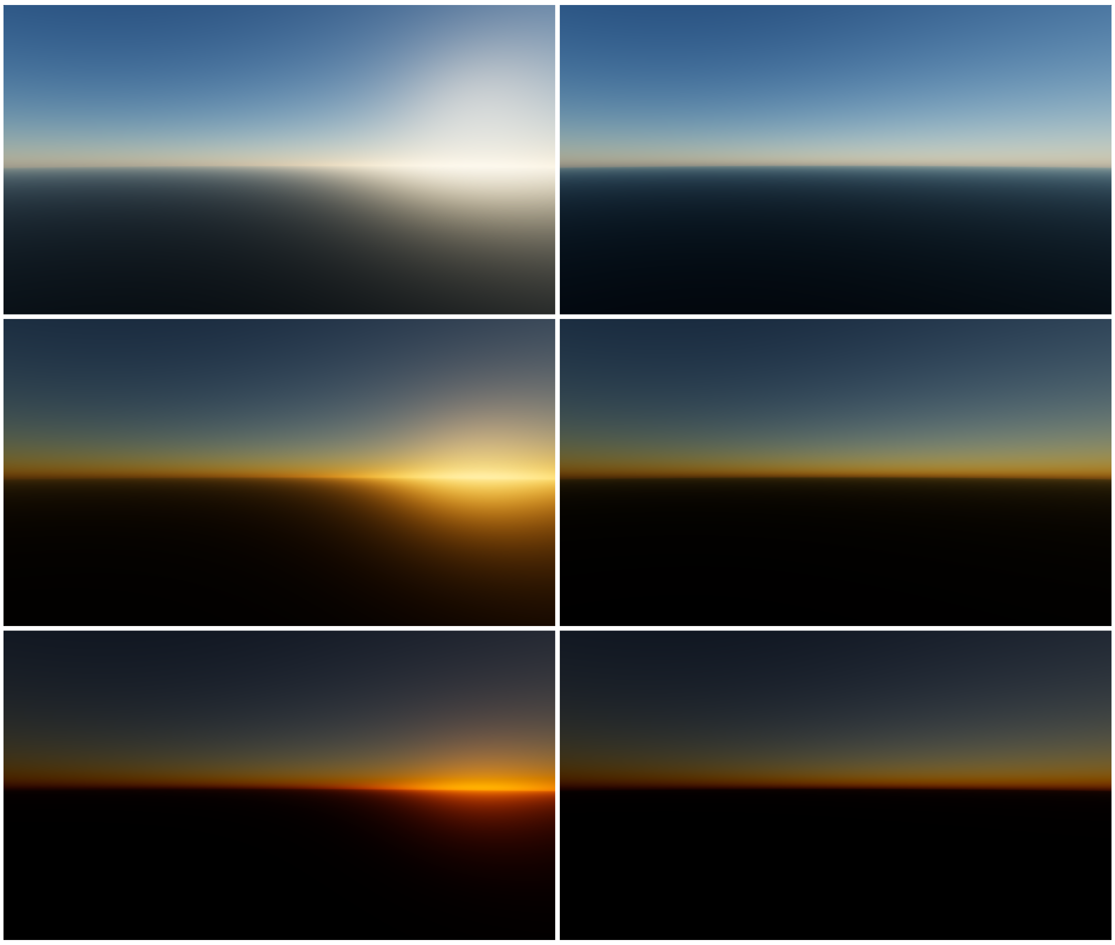

Figure 6.4: Impact of aerosols on the atmosphere. Left: Urban type aerosols. Right: Maritime Clean aerosols. Top to bottom: Sun elevation of 10°, 5° and 0°.

6.1. Performance

Since the implementation of our method is Shadertoy-based, and profiling WebGL applications is not very well supported, we run the shaders locally and profile them using the graphical debugging tool RenderDoc. We establish the threshold to consider an algorithm real-time in 16 ms, which equates to around 60 frames per second.

The hardware used for testing is a Nvidia GeForce GTX 1060 6GB GPU and an AMD Ryzen 5 1600 CPU with 16GB of RAM. Both pieces of hardware were released in 2017.

There are two distinct processes that make up the entirety of the computing time per frame: rendering the transmittance lookup table described in Section 5.1, and rendering the sky texture described in Sections 5.2 and 5.3. The transmittance texture takes **0.05 ms** to render, while the sky texture takes **0.16 ms**. Together they add up to **0.21 ms** in total, leaving plenty of frame time for other processes. This confirms that our approach is indeed real-time, as its computing time is much lower than the

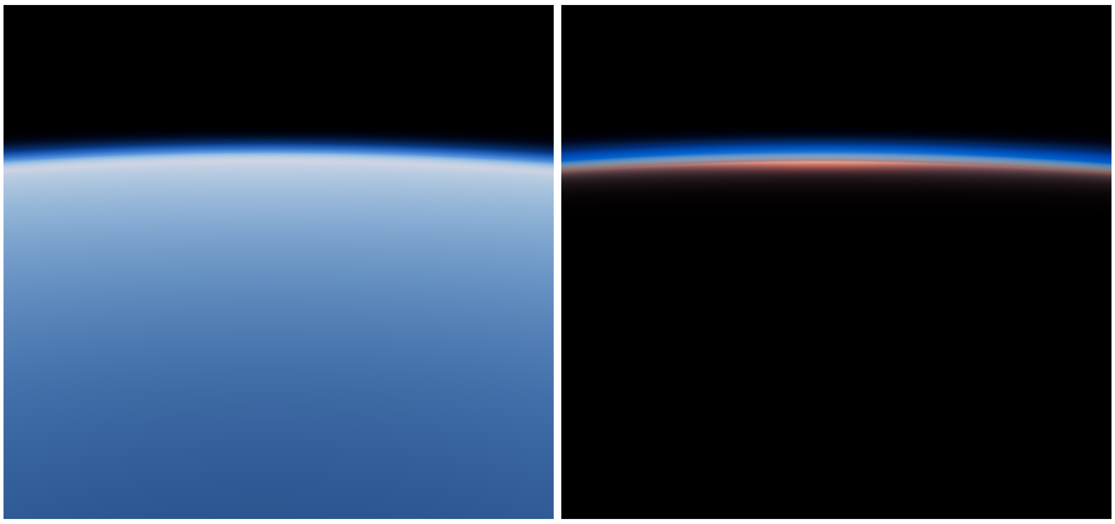

Figure 6.5: Space view of the atmosphere 80km above the surface.

established threshold of 16 ms.

The main parameter that we can adjust to vary the performance of our algorithm is the number of ray marching steps for both textures. For our results we have used 32 steps for the sky texture and 64 for the transmittance texture. If we would like our method to run on a mobile device, we can lower the sky texture ray marching steps to 8. This would lower the quality of the final result though.

It is also important to note that both textures have to be recomputed only if the atmospheric conditions have changed, or if the Sun's direction has changed in the case of the sky texture. This means that the actual performance impact on a real application would be much lower if we do not recompute both textures every frame.

The memory requirements of our approach are also quite low. The size of the transmittance texture is 256x64, while the size of the sky texture is 256x256. The format of both textures is RGBA32F (4 channels, 32-bits each) in order to have as much precision as possible, which equates to a total memory consumption of **1.25 MiB**.

To put things into perspective, our path tracer would take around 10 minutes to render a 256x256 image of the sky with sufficiently low noise. The improvement in performance with respect to offline methods is clear.

7

Conclusions

In summary, our method is able to successfully render the Earth's atmosphere from any arbitrary viewpoint in real-time. We have shown how our results compare to those from a path tracer, demonstrating our method's ability to maintain its accuracy even in tough atmospheric conditions. Furthermore, we have implemented this work on a public Shadertoy so that it is hopefully iterated upon by other people. The source code for the spectral path tracer that has been used to generate the reference ground truth images is also provided.

7.1. Future Work

We believe that atmosphere rendering is still an unsolved problem in computer graphics. Multiple scattering is still not as accurate as it could be, and our method relies on certain assumptions about our atmosphere which quickly break down when dealing with other planets or environments.

The phase function used for aerosols could be improved, as the Henyey-Greenstein phase function fails to capture the complex shape and appearance of the real one obtained through Mie theory. This intricate phase function could be fitted to a simple function or stored into a lookup table to make it viable in real-time.

Lastly, we think that our spectral rendering framework is applicable to other use-cases and is not limited to the atmosphere. Any participating media which has optical properties that are strongly dependent on wavelength can benefit from our approach, such as human skin and hair, which feature prominent subsurface scattering. All of the techniques and the general workflow described in Section 5.4 can be applied to these volumetric media.

Bibliography

[AIA99] AIAA. *Guide to Global Aerosol Models (GAM).* American Institute of Aeronautics and Astronautics, Inc., Jan. 1999.

[BN08] Eric Bruneton and Fabrice Neyret. «Precomputed Atmospheric Scattering». In: *Computer Graphics Forum.* Special Issue: Proceedings of the 19th Eurographics Symposium on Rendering 2008 27.4 (June 2008), pp. 1079–1086.

[Bru16] Eric Bruneton. «A Qualitative and Quantitative Evaluation of 8 Clear Sky Models». In: *IEEE Transactions on Visualization and Computer Graphics* PP (Oct. 27, 2016), pp. 1–1.

[Buc95] Anthony Bucholtz. «Rayleigh-scattering calculations for the terrestrial atmosphere». In: *Applied Optics* 34.15 (May 20, 1995), pp. 2765–2773.

[Cha60] S Chandrasekhar. *Radiative transfer.* New York: Dover Publications, 1960.

[CIE32] CIE. *Commission internationale de l'Eclairage proceedings.* Cambridge: Cambridge University Press, 1932. 691 pp.

[CS92] William M. Cornette and Joseph G. Shanks. «Physically reasonable analytic expression for the single-scattering phase function». In: *Applied Optics* 31.16 (June 1, 1992), pp. 3152–3160.

[DKS91] Guillaume A. D'Almeida, Peter Koepke, and Eric P. Shettle. *Atmospheric aerosols: global climatology and radiative characteristics.* Studies in geophysical optics and remote sensing. Hampton, Va., USA: A. Deepak Pub., 1991. 561 pp.

[Düt74] H. U. Dütsch. «The Ozone Distribution in the Atmosphere». In: *Canadian Journal of Chemistry* 52.8 (Apr. 15, 1974), pp. 1491–1504.

[EK10] Oskar Elek and Petr Kmoch. «Real-time Spectral Scattering in Large-scale Natural Participating Media». In: Proceedings - SCCG 2010: 26th Spring Conference on Computer Graphics. May 13, 2010, pp. 77–84.

[Ele09] Oskar Elek. «Rendering Parametrizable Planetary Atmospheres with Multiple Scattering in Real-Time». In: May 20, 2009.

[Fon+17] Julian Fong et al. «Production volume rendering: SIGGRAPH 2017 course». In: *ACM SIGGRAPH 2017 Courses.* SIGGRAPH '17. New York, NY, USA: Association for Computing Machinery, July 30, 2017, pp. 1–79.

[GGJ18] David Guimera, Diego Gutierrez, and Adrián Jarabo. «A Physically-Based Spatio-Temporal Sky Model». In: *Spanish Computer Graphics Conference (CEIG)* (2018).

[Gor+14] V. Gorshelev et al. «High spectral resolution ozone absorption cross-sections - Part 1: Measurements, data analysis and comparison with previous measurements around 293 K». In: *Atmospheric Measurement Techniques* 7.2 (Feb. 24, 2014), pp. 609–624.

[HG41] L. G. Henyey and J. L. Greenstein. «Diffuse radiation in the Galaxy.» In: *The Astrophysical Journal* 93 (Jan. 1, 1941), pp. 70–83.

[Hil20] Sébastien Hillaire. «A Scalable and Production Ready Sky and Atmosphere Rendering Technique». In: *Computer Graphics Forum* 39.4 (2020), pp. 13–22.

[Hul57] Hendrik Christoffel van de Hulst. *Light Scattering by Small Particles.* Courier Corporation, 1957. 502 pp.

[HW12] Lukas Hosek and Alexander Wilkie. «An analytic model for full spectral sky-dome radiance». In: *ACM Transactions on Graphics* 31.4 (July 1, 2012), 95:1–95:9.

[Jae93] Ruprecht Jaenicke. «Chapter 1 Tropospheric Aerosols». In: *International Geophysics.* Ed. by Peter V. Hobbs. Vol. 54. Aerosol–Cloud–Climate Interactions. Academic Press, Jan. 1, 1993, pp. 1–31.

[Jen+01] Henrik Wann Jensen et al. «A practical model for subsurface light transport». In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 511–518.

[Kal+17] Simon Kallweit et al. «Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks». In: *ACM Transactions on Graphics* 36.6 (Nov. 20, 2017), pp. 1–11.

[Ket+21] Markus Kettunen et al. «An unbiased ray-marching transmittance estimator». In: *ACM Transactions on Graphics* 40.4 (Aug. 31, 2021), pp. 1–20.

[Nis+93] Tomoyuki Nishita et al. «Display of the earth taking into account atmospheric scattering». In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques.* SIGGRAPH '93. New York, NY, USA: Association for Computing Machinery, Sept. 1, 1993, pp. 175–182.

[Nis+96] Tomoyuki Nishita et al. «Display method of the sky color taking into account multiple scattering». In: (Jan. 1, 1996).

[NM65] J. A. Nelder and R. Mead. «A Simplex Method for Function Minimization». In: *The Computer Journal* 7.4 (Jan. 1, 1965), pp. 308–313.

[Nov+18] Jan Novák et al. «Monte Carlo methods for volumetric light transport simulation». In: *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37.2 (May 2018).

[NSJ14] Jan Novák, Andrew Selle, and Wojciech Jarosz. «Residual Ratio Tracking for Estimating Attenuation in Participating Media». In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 33.6 (Nov. 2014).

[ONe05] Sean O'Neil. «Accurate Atmospheric Scattering». In: *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005, pp. 253–268.

[Pet+19] Christoph Peters et al. «Using Moments to Represent Bounded Signals for Spectral Rendering». In: *ACM Trans. Graph.* 38.4 (July 2019).

[PSS99] A. J. Preetham, Peter Shirley, and Brian Smits. «A practical analytic model for daylight». In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., July 1, 1999, pp. 91–100.

[RK53] K. R. Ramanathan and R. N. Kulkarni. «Height distribution of atmospheric ozone». In: *Proceedings of the Indian Academy of Sciences - Section A* 37.2 (Feb. 1, 1953), pp. 321–331.

[Sch12] Christian Schüler. «An Approximation to the Chapman Grazing-Incidence Function for Atmospheric Scattering». In: *GPU PRO 3*. A K Peters/CRC Press, 2012.

[Uni76] United States Committee on Extension to the Standard Atmosphere. *U.S. standard atmosphere, 1976*. Washington, D.C.: U.S. Government Printing Office, 1976.

[Wil+14] A. Wilkie et al. «Hero Wavelength Spectral Sampling». In: *Computer Graphics Forum* 33.4 (2014), pp. 123–131.

[Yan97] Edgard G. Yanovitskij. *Light Scattering in Inhomogeneous Atmospheres*. Springer Berlin Heidelberg, 1997. 400 pp.

[Yus14] Egor Yusov. «High Performance Outdoor Light Scattering Using Epipolar Sampling». In: *GPU Pro 5*. A K Peters/CRC Press, 2014.

[Zim+89] P. H. Zimmermann et al. «A global three-dimensional source-receptor model investigation using 85Kr». In: *Atmospheric Environment* 23 (Jan. 1, 1989), pp. 25–35.