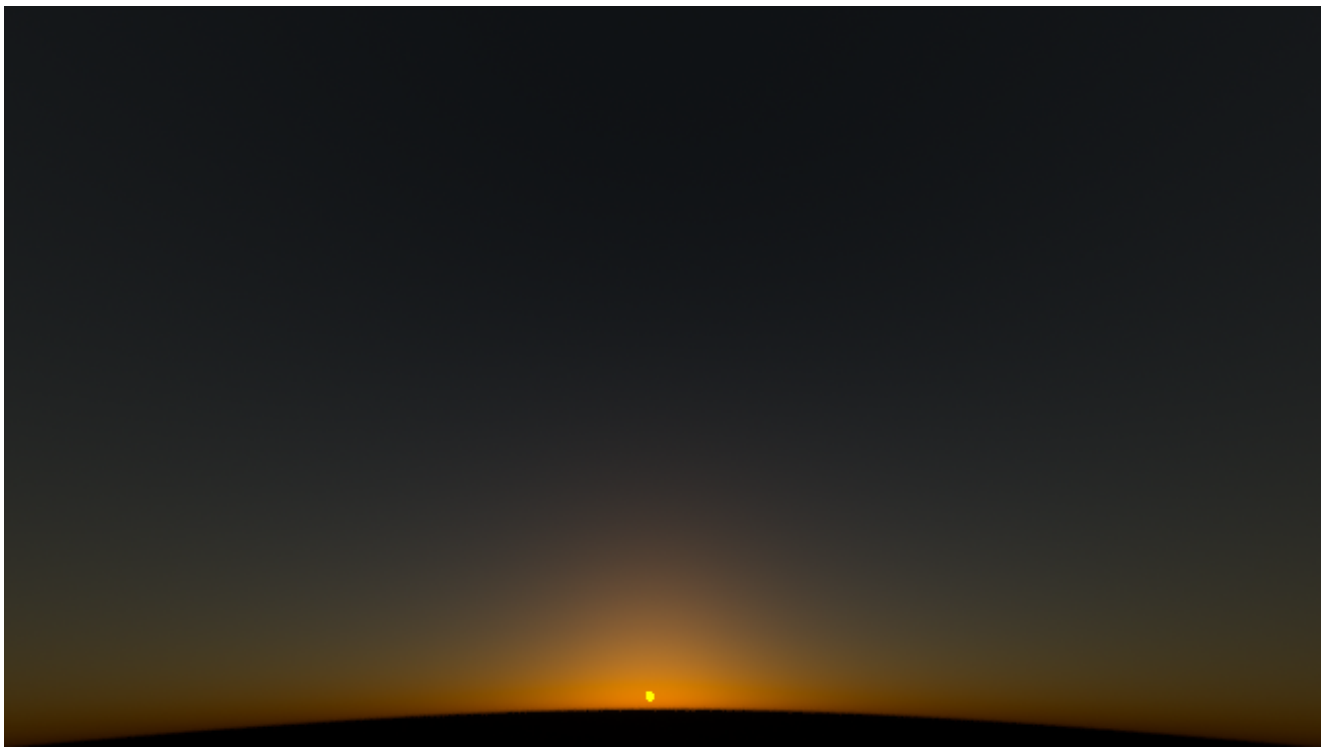
Physical Sun and Sky Models in appleseed



Bachelor Thesis
18.11.2020

by Joel Barmettler, 16-727-877

Supervisors:
Prof. Dr. Renato Pajarola
Lars Zawallich

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

Abstract

The sky is the most prominent – and thus most important – source of light in outdoor scenes. Unfortunately, offline ray-traced rendering engines have often been relying on analytic sky models that deliver poor visual results and inaccurate radiance values for their visual representation of the sky. A physically-based sky model would overcome those issues and provide a more accurate portrayal. In this thesis, the current research state regarding Computer Graphic (CG) sky rendering is assessed and an improved version of the Nishita93 sky model is proposed. The suggested implementation is then compared to other state-of-the-art sky models, including a physically-based sky model implemented by the Cycles rendering engine by Blender Foundation. Moreover, it considers the atmosphere's ozone layer to better account for the blueness of the sky. The results show that the improved Nishita model delivers superior visuals and radiance accuracy values compared to all of the other analytic sky models, while also being faster, more flexible, and more accurate than Blender's implementation. The developed model is contributed to the open-source offline rendering engine appleseed[1] in C++.

[1] Appleseed is a modern open-source rendering engine for animation and visual effects. https://appleseedhq.net.

Zusammenfassung

Der Himmel ist bei Szenerien im Freien die wichtigste Beleuchtungsquelle überhaupt. Trotzdem verlassen sich viele Raytrace-Rendering-Engines immernoch auf analytische Himmelsmodelle, welche nicht nur optisch wenig überzeugende Resultate liefern, sondern auch schlechte Radienzwerte produzieren. Ein physikalisch basiertes Himmelsmodell hingegen würde diese Schwachstellen beseitigen. In dieser Arbeit wird der momentane Forschungsstand von physikalisch basierten Himmelsmodellen aus dem Feld der Computergrafik evaluiert und eine verbesserte Version des Nishita93-Modells vorgestellt. Das verbesserte Modell berücksichtigt den Einfluss von Ozon-Partikeln auf das Sonnenlicht, um die blauen Farben am Morgen- und Abendhimmel besser darzustellen. Das vorgeschlagene Modell wird mit anderen modernen Himmelsmodellen verglichen, insbesondere auch mit einem Modell, welches die Blender Foundation für die Cycles Rendering-Engine implementiert hat. Die Resultate des Vergleiches zeigen, dass das hier vorgeschlagene Himmelsmodell nicht nur herausragend bessere optische und physikalische Werte als alle anderen analytischen Himmelsmodelle produziert, sondern auch schneller, flexibler und akkurater ist als das Modell der Blender Foundation. Das entwickelte Modell wird zur Open-Source Rendering-Engine appleseed[1] als C++ Implementation beigetragen.

[1] Appleseed ist eine moderne Open-Source-Rendering-Engine für Animationen und visuelle Effekte. https://appleseedhq.net.

Contents

1 Introduction

Ray-traced rendering engines like appleseed [BTB+19] are pursuing photo-realistic, physically accurate computer-generated imagery. Due to this, simulating accurate sky illumination has gained importance, since the sky acts as the primary light source for outdoor daylight scenes. While physically-based clear sky models have existed since the early 90s, they are not broadly implemented in rendering engines, most probably due to their rather difficult implementation and high computational complexity compared to more modern analytic models. Many rendering engines still use analytic models to approximate the colour and radiance of the sky which deliver not only less realistic looking scenes, but also more inaccurate global illumination [KKN+14]. In order to further strive for realism in the field of computer graphics, it is essential to have state-of-the-art physically-based sky models as part of offline rendering engines such as appleseed.

The appearance and radiance of the sky are a result of sunlight interacting with particles in the atmosphere. Interactions between light rays and particles of different shapes and sizes lead to absorption and scattering events, altering the intensity and colour of the light. As a visible result, we perceive the sky in a blue colour during the day while it appears to be more red and yellow during sunrise and sunset, when the sun is closer to the horizon. Physical sky models simulate sunlight rays through the atmosphere, taking into account absorption and scattering events caused by different types of particles. Thereby, the intensity and colour of the light reaching a certain viewpoint can be calculated.

Scattering events in the atmosphere can be characterised by two different kinds of scatterings: Raylight scattering and Mie scattering. Raylight scattering is mainly caused by gasses like Nitrogen and Oxygen [GGJ18] whose particles are significantly smaller than the wavelength of the sunlight. These gasses scatter the sunlight in a uniform but wavelength-selective manner, which defines the colour of the sky. Shorter wavelengths that appear blue or green are scattered to a much larger extent than longer wavelengths like orange or red. This is the reason why the sky mainly appears blue. The well known exception is during sunrise an sunset when the sun is low. During these periods, the sunlight passes a long way through the atmosphere, scattering nearly all blue light away from the viewpoint, which results in a yellow and red sky [Str71] [Jar08]. On the other hand, particles that are larger than the wavelength of sunlight, like water droplets, human-made pollution, or dust, cause Mie scattering [GGJ18]. They scatter the light in a strongly forward-direction with little to no wavelength selectivity compared to Mie scattering. This causes the characteristic bright halo effect around the sun disk [Mie08] [Jar08].

However, not all particles cause a scattering of light. For example, ozone particles only absorb incoming light without causing a scattering. The amount of absorbed light depends on its wavelength: Ozone mostly absorbs green, yellow, and orange light. The molecule properties of ozone make the gas a wavelength-selective colour filter [HMS05]. This being said, ozone gives the sky a bluer look during sunset and sunrise when the light has been traveling a long way through the ozone layer, while it has little to no impact on the colour of the sky during the day [Hul53].

Every single light ray experiences numerous scattering events on its way through the atmosphere before eventually reaching the viewpoint. Modelling a large number of such consecutive bounces requires an immense computational effort. Hence, most physically-based sky models only simulate one or two scattering events. This approximation induces only a minimal loss in physical accuracy since higher-order scatterings have decreasing influence in the overall appearance of the sky while they need exponentially more time to compute [NDKY96].

The first models to simulate the sky were based on ray-tracing algorithms and originated with Klassen in 1987 [Kla87]. Due to their high time and space complexity, the physically-based models were (and still are) not suitable for real-time applications such as video games or flight simulators. In 1999, analytic models were invented by Preetham [PSS99]. Most analytic models define a mathematical formula based on Perez' all-weather model [PSM93] and approximate the results of either ray-traced sky models or real-world measurements to achieve adequate colour results in daylight scenes. This makes analytic models fast and therefore applicable for real-time applications. The analytic models are significantly faster when it comes to rendering time compared to physical models, but they suffer from lower accuracy [KKN+14]. Today, analytic models are still widely used, even in offline renderers like appleseed.

The most recent trend in the sky model research field is the development of physically-based, precomputed models. Their computation is based on physical, ray-traced simulations, but most of the expensive calculation is precomputed for various sky conditions (e.g. sun-elevation, haze, etc.) and stored in tremendous, high-dimensional lookup tables. These methods produce lower spectral radiance errors compared to the analytic models [KKN+14] while being rated as highly accurate in perception studies [BN08].

Even the earliest ray-tracing based sky models like the single-scattering model from 1993 by Nishita et al. [NSTN93] are excellent in simulating solar radiance on earth since their implementation is close to a brute force path-traced algorithm. In a comparison of solar and sky-dome illumination accuracy, different sky models were evaluated regarding their strengths and weaknesses. It was observed that physically-based models produce the best results if the execution time is of no concern. Analytic models should be used when plausible results are enough and real-time performance is required [KKN+14]. Physically-based sky models are therefore an excellent extension to the open-source renderer appleseed, which, at the time of writing, only supports analytic sky models like the Preetham or Hosek & Wilkie sky model.

The field of computer graphics strongly needs open-source rendering engines that can hold up to the latest requirements from both academia and the industry. Especially the entertainment industry is pushing for higher and higher realism not only towards a plausible look of the sky in general but also for dependable and accurate radiance illumination to determine objects' material properties accurately [Slo02]. Extending an offline, open-source, ray-traced renderer like appleseed with a physically-based model is, therefore, a natural choice.

This thesis aims to evaluate which physical sun and sky model is the best fit for appleseed's current state and contribute the models C++ implementation to the open-source code repository of appleseed.

2 Background and Related Work

In the following chapter, important background knowledge about the atmosphere, radiometry, and the light transportation equation is provided. However, knowledge about computer graphics, especially ray-tracing and render-engines in general, is assumed. The related work section oversees the current research state of computer generated sky models while differentiating between analytic, physical, and precomputed models.

2.1 Background Knowledge

Implementing a physically-based sky model requires background knowledge in different fields. Basic knowledge of the atmosphere and its composition must be built in order to understand which particles must be simulated. Knowledge of radiometry helps to understand how light behaves and how light rays can be expressed mathematically. Further, the light transportation equation describes how a light ray interacts with particles within a medium and how these interactions alter the appearance of the light.

2.1.1 Atmosphere Composition

The atmosphere consists of a large variety of different gasses [FC13]. To render the look of the sky, the influence of those gasses on the sunlight must be simulated. The characterisation of a gas for this use-case is simplified and based on the following three properties: a.) The gas density in the atmosphere as a function of altitude, b.) the (wavelength-dependent) absorption coefficient of the gas particles, and c.) the phase function, which describes the light's angular distribution after scattering at a certain gas particle. The interaction of those gas particles with incident sunlight makes up for the characteristic brightness and colour of the sky. In sum, this makes clear that to generate a realistic, physically-based sky model, it is essential to simulate the most influential particles accurately and efficiently. Table 2.1 summarizes the properties of the three most important gasses that need to be simulated in order to achieve a realistic looking sky: dry air, aerosols and ozone.

Clear Dry Air Most of the atmosphere consists of clean dry air, formed by four main components: 78.08% nitrogen (N_2), 20.98% oxygen (O_2), 0.93% argon (Ar) and 0.039% carbon dioxide (CO_2) [FC13]. Those gasses consist of particles with very tiny diameters, namely less than one-tenth of the light's wavelength. When light interacts with such small particles, no light waves are absorbed. Rather, the light is scattered into various directions, like a prism. The density of these particles has a roughly exponential fall-off in the atmosphere, with most of the particles concentrated at the bottom [Slo02]. When light travels along a path, it changes its colour

Gas / Particles	Highest Concentration	Particle Size	Scattering Function	Primary Influence on Sky Appearance
Dry air	$0 - 40\,\mathrm{km}$	$< 40\,\mathrm{nm}$	Rayleigh scattering	White/orange glare in circumsolar region
Aerosols	$0 - 7\,\mathrm{km}$	$> 700\,\mathrm{nm}$	Mie scattering	Blue color at daylight, red/orange color during sunrise/sunset
Ozone	$10\,\mathrm{km} - 60\,\mathrm{km}$	$< 1\,\mathrm{nm}$	-	Blue color during sunrise/sunset

Table 2.1: Comparison between dry air, aerosols, and ozone properties that have the most effect on the look of the sky.

and intensity due to dry air scattering out specific wavelengths. This scattering characteristic is also referred to as Rayleigh scattering, a term that will be explained in more detail in Section 2.1.3.

Aerosols Aerosols are solid or liquid particles with sizes of up to several micrometres. Among others, aerosol particles mostly origin from ocean spray, pollen, sandstorms or forest fires, but they can also be caused by humans through industrial air pollution. The terms "haze" or "turbidity" are generally used to refer to the aerosol concentration in the atmosphere. Due to their size, aerosol particles are disposed on the ground and transported into the atmosphere by winds, therefore they occur mainly within the first few kilometres and experience a fast, exponential fall-off in altitude. While aerosols do absorb a small fraction of the light, their primary contribution to the appearance of the sky comes from their scattering characteristics: Aerosols scatter light independently of its wavelength in a strong forward-directed manner [Slo02]. Therefore, when light travels along a path, aerosol particles reduce the intensity of the light by out-scattering but leaving the colour of the light unchanged. This scattering characteristic can be described through the Mie scattering theory, as described in Section 2.1.3.

Ozone Ozone (O_3) mostly occurs in altitudes of $10\,\mathrm{km}$ and $60\,\mathrm{km}$ above ground. In contrast to dry air and aerosols, ozone is more concentrated in higher altitudes. Ozone does not scatter the light. Instead, it influences the colour of the sky by absorbing part of the light in a wavelength-dependent manner. Consequently, ozone has no scattering phase function and solely contributes to the light attenuation through absorption. A light ray traveling through ozone will change its colour and intensity due to ozone molecules absorbing specific wavelengths [Slo02].

2.1.2 Radiometry

Most papers in the field of Computer Graphic (CG) sky rendering assume background knowledge in ray-traced rendering. The following subsection explains the commonly used terminology regarding light and colour theory and dives deeper into the field of *llight transportation in participating media* and the theory of light passing through a particle-filled environment like the atmosphere.

Rendering engines synthesize images from virtual scenes by simulating the physical behaviour of light rays emitted by virtual light sources and their interaction with the virtual objects. The light rays reaching the virtual camera after numerous interactions define the final image output. Since the physical sky acts as such a huge source of global illumination, it is essential to define the standard terminology in the field of radiometry required to discuss the physical properties of light. Radiometry designates the study of physical measurements of electromagnetic radiation, including visible light [Jar08].

Energy All radiometric quantities are effectively different ways of measuring photons that are emitted by a light source. Photons are of a particular wavelength and energy. The wavelength determines how the photon is perceived by the human eye, which can detect photons between roughly 380 and 720 nanometers. The energy Q of a photon is measured in joules J. The relation between the photons' wavelength and the energy it carries is:

$$Q = \frac{hc}{\lambda} \tag{2.1}$$

with h being the Planck's constant and c the speed of light [PH10].

Flux The total amount of energy that hits a surface of area A per unit time is called the radiant power, or flux ϕ, see Figure 2.2. The unit of flux is therefore $\frac{\mathrm{J}}{\mathrm{s}}$, or $Watts$ (W).

$$\phi = \frac{\Delta Q}{\Delta t} \tag{2.2}$$

Flux is also used to define the emission of light sources. Two imaginary spheres of different sizes around a light source both receive the same flux. Even though fewer photons penetrate any local part of the larger sphere due to its bigger surface area, it still receives as much flux as the smaller sphere, where more energy passes through any unit area of the sphere [PH10].

Irradiance The irradiance removes the dependency of the surface area A. Irradiance E describes the flux per unit area and is measured as Watts per square meter $\frac{\mathrm{W}}{\mathrm{m}^2}$. The unit area of $1\,\mathrm{m}^2$ is represented as a dotted sphere in Figure 2.2. Irradiance is formally defined as the differential-power per differential-area at a point p:

$$E(p) = \frac{\Delta\phi(p)}{\Delta A} \tag{2.3}$$

Proceeding with the previous example of the two imaginary spheres around a light source; since the outer sphere has a much larger surface area A, it has a lower irradiance as the inner sphere with the smaller surface. As an effect, the outer sphere is also perceived as less bright. This observed effect is supported by the fact that the amount of energy perceived from a light source at a certain point decreases with the squared distance of the point from the light [PH10]. Furthermore, the rules of irradiance are in line with the Beer-Lambert law, stating that "the amount of light arriving at a surface is related to the cosine of the angle between the light direction and the surface normal" [PH10], Page 283.

Radiance exitance While the irradiance describes flux per unit area arriving at a certain area, the radiant exitance describes the flux per unit area leaving a surface [PH10].

Intensity To explain the term "intensity", the principle of solid angles must be explained first. Solid angles are an extension of regular angles into the third dimension. While a regular angle describes a section of the 2D unit circle, a solid angle describes a surface area on the 3D unit sphere. While regular angles are measured in radians, solid angles are measured in steradians sr. The concept of solid angles is visualized in Figure 2.1: Given a sphere with radius r, the solid angle Ω is defined by the area A divided by the square radius of the sphere:

$$\Omega = \frac{A}{r^2} \tag{2.4}$$

Therefore, the larger the solid angle Ω, the larger the covered surface area A.

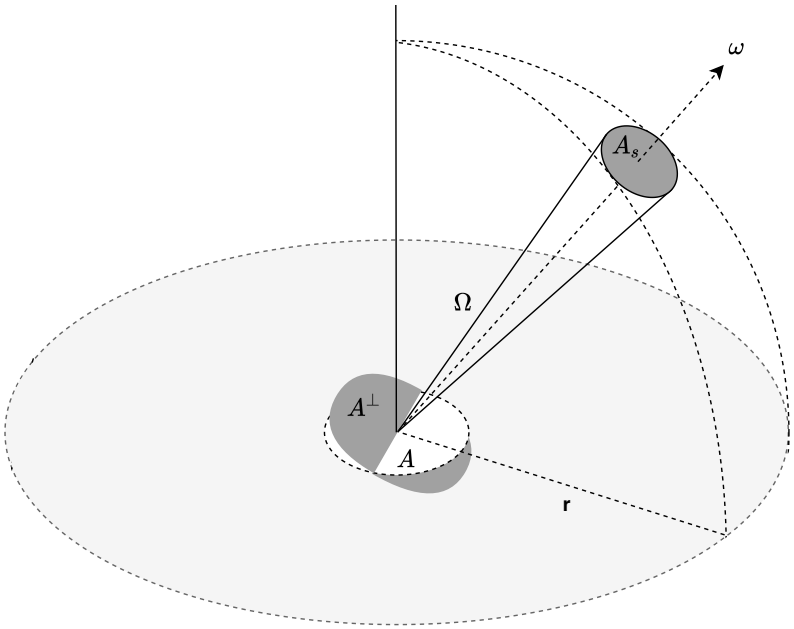


Figure 2.1: A solid angle Ω in direction ω describing the surface area A_s on a sphere with radius r. The area $A^\perp$ is perpendicular to the direction ω.

Using the concept of solid angles, the intensity is defined as the flux emitted by a point light source per unit steradian, describing the directional distribution of light [PH10]. The intensity I has units $\frac{\mathrm{W}}{\mathrm{sr}}$ and is the differential power per differential direction:

$$I = \frac{\Delta\phi}{\Delta\omega} \tag{2.5}$$

Radiance The irradiance describes the average power on an area without taking into account the directional distribution of that power. The radiance describes the irradiance in relation to the solid angle. Radiance measures the amount of light from a differential direction $\Delta\omega$ that passes through the differential area $\Delta A^{\perp}$ that is perpendicular to ω [PH10], see Figure 2.1.

$$L = \frac{\Delta\phi}{\Delta\omega\Delta A^{\perp}} \tag{2.6}$$

The radiance directly describes the perceived brightness of a surface by human eyes [Jar08].

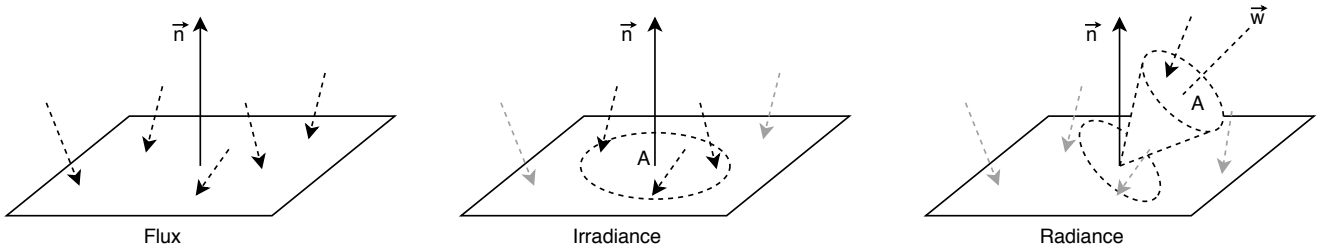


Figure 2.2: Visualization of flux (left), irradiance (center) and radiance (right).

Incident and Exitant Radiance Functions Incident and exitant radiance describe the amount of radiance entering or leaving a point x. We denote $L(x \rightarrow \vec{\omega})$ as exitance radiance, the radiance leaving the point x in direction $\vec{\omega}$, and $L(x \leftarrow \vec{\omega})$ incident radiance, radiance arriving at the point x from direction $\vec{\omega}$. Since the directional vector ω in both cases points out of the surface, the following equation holds in vacuum where radiance remains constant without interacting with any particle [Jar08]:

$$L(x \leftarrow \vec{\omega}) = L(x \rightarrow -\vec{\omega}) \tag{2.7}$$

However, should a particle at point x absorb any energy, the radiance leaving the point x will be smaller than the radiance that arrived at x [Jar08]:

$$L(x \leftarrow \vec{\omega}) > L(x \rightarrow -\vec{\omega}) \tag{2.8}$$

Spectral Power Distribution In physics, spectral irradiance to describe colour can be expressed as a spectral power distribution (SPD). The spectral power distribution function over the human visible wavelength spectrum characterizes the perceived colour of any surface or light source. The rather complex functions are often approximated by sampling the function uniformly with n sample points. The sampled spectrum representation is used to define and manipulate colour within the rendering engine since it is the most physically accurate representation [PH10].

2.1.3 Light Transportation Equation in Participating Media

In a vacuum, incident and exitance radiance are equivalent for any point x. However, the atmosphere is a space full of particles like aerosols, dry air and ozone, later referred to as "particles", which influence the radiance of light rays that are travelling through them. These particles are referred to as "participants" within the atmosphere, hence

the name "participating media". For small virtual scenes, light only travels very short distances through mostly clean air with an extremely low density of participating media. Hence, space with nearly no particles is defined as a vacuum. However, to simulate sunlight through the atmosphere, this is not a reasonable approximation. Sunlight travels long distances through the atmosphere and the particles it contains, where, even at very low density, the particles are very likely to interact with the light rays. The Light Transport Equation in participating media mathematically formulates how to calculate the radiance arriving at any point x within a participating medium. In the following sections, an adaptation of the Light Transportation Equation is explained. All parts of the equation that describe phenomenons that do not occur in the atmosphere (like emission of particles) are left out, while some terms are renamed to be more descriptive for the applied atmospheric use case.

These interactions between the light rays and the atmospheric particles alter the radiance of the light originating from the sun. The probability of an interaction per travelled meter is denoted as σ_t. This probability is also called the *extinction coefficient*, and depends on the particle size and its density within the medium. Whenever an interaction occurs, the photon involved is either absorbed or scattered, see Figure 2.3. The probability of absorption is defined as σ_a and the probability of scattering as σ_s, with $\sigma_t = \sigma_a + \sigma_s$. If any of the two events take place, a change in radiance occurs [Jar08].

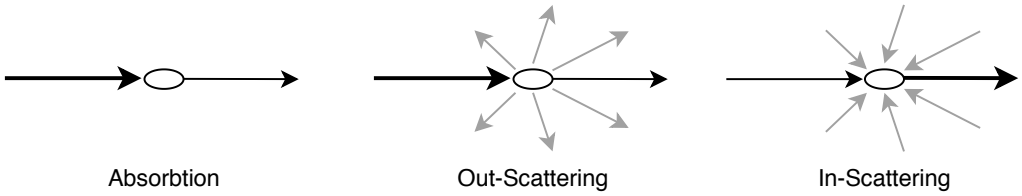


Figure 2.3: Three interaction types when a light ray hits a particle: absorption (left), out-scattering (center) and in-scattering (right). An interaction either increases or decreases the radiance values of the initial light ray.

The ratio between the probability of scattering and probability of extinction, $\frac{\sigma_s}{\sigma_t}$, is called the *albedo*. Therefore the albedo describes a medium's property, indicating how likely a photon is to scatter at point within the participating medium. The albedo is similar to the reflectivity of a surface, with an albedo of 1, meaning the surface scatters 100% of the incoming light. An albedo of 0 indicates that all light is fully absorbed [Jar08].

In the atmosphere, the density of different particles changes with altitude. Consequently, the probability of scattering / absorption must change with altitude and cannot be set constant [Jar08].

Extinction Consider a light ray from the sun at position x in direction $\vec{\omega}$, causing incident radiance $L(x \to \vec{\omega})$. At each step Δt, σ_a percentage of the photons will be absorbed. At the end of a segment t, the differential change in radiance due to absorption is given as:

$$(\vec{\omega} \cdot \Delta_a)L(x \to \vec{\omega}) = -\sigma_a(x)L(x \to \vec{\omega}) \tag{2.9}$$

with Δ_a being the gradient due to absorption and $(\vec{\omega} \cdot \Delta_a)$ the directional derivative in the direction $\vec{\omega}$ [Jar08].

Out-Scattering Beside extinction where photons are being absorbed, photons can also be lost due to out-scattering effects. The radiance change caused by out-scattering is given as:

$$(\vec{\omega} \cdot \Delta_o)L(x \to \vec{\omega}) = -\sigma_s(x)L(x \to \vec{\omega}) \tag{2.10}$$

combined, we get a total loss of radiance being:

$$(\vec{\omega} \cdot \Delta_t)L(x \to \vec{\omega}) = -(\sigma_a(x) + \sigma_s(x))L(x \to \vec{\omega}) = -\sigma_t(x)L(x \to \vec{\omega}) \tag{2.11}$$

The differential Equation 2.11, called Transmittance T_r, gives the percentage of photons that can travel distance t from x into direction $\vec{\omega}$ without being influenced by any particle in the participating medium. By integration, we can express the transmittance simply as:

$$T_r(x' \leftrightarrow x) = e^{-\tau(x' \leftrightarrow x)} \tag{2.12}$$

τ is the *optical depth*, the integral over all extinction along the light ray's path through the atmosphere [Jar08].

The optical depth depends on the particle density, which is not constant within the atmosphere. According to Nishita et al. [NSTN93], the density of dry air and aerosols at a given height h can be determined using a simple exponential function:

$$\rho(h) = e^{\frac{-h}{H_{den}}} \tag{2.13}$$

with H_{den} being the reference altitude for the type of particle. For dry air (mainly the four previously described gases N_2, O_2, Ar, CO_2), a value of $H_{den} = 7994\,\mathrm{m}$ was used by Nishita et al., and a value of $H_{den} = 1200\,\mathrm{m}$ for aerosol particles. When making the particle density dependent on the altitude h, the exponential decrease of particle density with increasing altitude is modelled [Jar08]. For ozone molecules, modelling the particle density accurately as a function of altitude turns out to be rather difficult since the distribution of ozone in the atmosphere changes drastically depending on the geo-location, temperature, etc. Generally, the ozone density in the atmosphere increases roughly linearly, starting at about $10\,\mathrm{km}$ with reaching its maximum at $32\,\mathrm{km}$. Afterwards, the ozone density experiences an exponential decrease until $60\,\mathrm{km}$ above the ground [GRI05].

In-Scattering Until now, only extinction and out-scattering have been explained which are events that reduce the initial radiance intensity. However, radiance can also increase along its path through in-scattering events. Photons can be scattered out of a light ray and join another ray, and thereby increase the radiance of the new ray by the amount of radiance lost by the old ray through the out-scattering event. The radiance of the in-scattering event depends on L_i, the in-scattering radiance, and is defined similarly to the extinction event:

$$(\vec{\omega} \cdot \Delta_i)L(x \to \vec{\omega}) = \sigma_s(x)L_i(x \to \vec{\omega}) \tag{2.14}$$

When the radiance of the sky is calculated, most of the directions that are not directly facing the sun disk will have no initial radiance, since no light is emitted from deep space. All the radiance coming from these directions are results of out-scattering events of the initial sun rays [Jar08].

Finding the total amount of in-scattering radiance at a given point x, $L_i(x \to \vec{\omega})$ requires integrating the incident radiance over all possible directions at x, taking into account the phase function of the underlying medium. The phase function defines the light's angular distribution after scattering. The angular distribution is important

for in-scattering events since the phase function often describes clear tendencies of particles to scatter in a certain direction [Jar08].

The phase function depends on the size and shape of the particles in the medium. An isotropic phase function would imply perfect isotropic scattering. In the atmosphere, two particles lead to two different common phase functions. Rayleigh scattering is caused by particles that are smaller than the wavelength of light, such as air particles. Bigger particles like aerosols lead to Mie scattering [Jar08].

Rayleigh Scattering The Rayleigh phase function is close to isotropic and highly wavelength dependent, meaning the probability of scattering into a certain direction varies greatly with the wavelength of the photon. Rayleigh scattering causes the characteristic colours of the sky, scattering small wavelengths much less than larger ones. The Rayleigh phase function p_R is given as:

$$p_R(x,\theta) = \frac{3}{16\pi}(1 + cos^2\theta) \tag{2.15}$$

with the scattering coefficient

$$\sigma_s = \frac{2\pi^5}{3}\frac{d^6}{\lambda^4}(\frac{n^2-1}{n^2+2})^2 \tag{2.16}$$

where λ is the wavelength of the light, d is the diameter of the particle and n the particles refractive index [Jar08]. The angular distribution of the Rayleigh phase function can be seen in Figure 2.4.

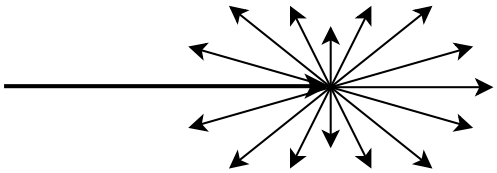


Figure 2.4: With incident light coming from the left, light is rayleigh-scattered at a particle. The arrow length illustrates the probability of scattering into the arrow's direction.

Mie Scattering The Lorenz-Mie theory of scattering is used for particles that are larger than the light's wavelength. Since the Lorenz-Mie scattering functions are complex, Nishita et al. [NMN87] derived two approximations, one more suitable for so called "hazy" atmospheres p_{MH} and the other for "murky" ones p_{MM}.

$$p_{MH}(x,\theta) = \frac{1}{4\pi}(\frac{1}{2} + \frac{9}{2}(\frac{1+cos(\theta)}{2})^8) \tag{2.17}$$

$$p_{MM}(x,\theta) = \frac{1}{4\pi}(\frac{1}{2} + \frac{33}{2}(\frac{1+cos(\theta)}{2})^{32}) \tag{2.18}$$

While these phase functions are not wavelength-dependent, they show an extreme forward tendency. The larger the particle, the stronger the forward scattering tendency [Jar08]. Visualizations of the Mie phase functions can be seen in Figure 2.5

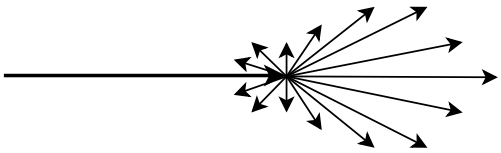


Figure 2.5: With incident light coming from the left, light is mie-scattered at a particle. The arrow length illustrates the probability of scattering into the arrow's direction.

Radiative Transfer Equation, or Volume Rendering Equation The three scattering events, absorption, out-scattering, and in-scattering, can be combined to model the full effect of light while traveling through a participating medium. By combining Equation 2.9, Equation 2.10 and Equation 2.14, the total radiance change at the point x over the ray is given as:

$$(\vec{\omega} \cdot \Delta)L(x \to \vec{\omega}) = -\sigma_a(x)L(x \to \vec{\omega}) - \sigma_s(x)L(x \to \vec{\omega}) + \omega_s(x)L_i(x \to \vec{\omega}) \tag{2.19}$$

This equation is known as the *radiative transfer*, or *radiative transport equation (RTE)*. The boundary condition of such an integral is usually given by the light source or the boundary of the medium. Integrating Equation 2.19 results in:

$$L(x \leftarrow \vec{\omega}) = \underbrace{T_r(x \leftrightarrow x_S)L(x_S \to -\vec{\omega})}_{\text{extinction}} \int_0^S \underbrace{T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t \to -\vec{\omega})dt}_{\text{in-scattering}} \tag{2.20}$$

The first part of the equation corresponds to the reduced radiance due to extinction (absorption and out-scattering), while the second part is the accumulated in-scattering radiance. The two parts together describe the radiance reaching point x from direction $\vec{\omega}$, whereas s is the distance travelled in the medium to the point x, x_s is the location of the light source, and x_t any point on the light ray between x and the point where the light ray entered the medium. The transmittance is given as T_r [Jar08].

Equation 2.20 is usually called the *Volume Rendering Equation*, or short *VRE*. It describes, at any point x in a medium, the radiance that is left after the light ray with direction ω reached point x. The radiance for any point x within the participating medium depends not only on the previous points on the ray, but also on all other points in the whole medium. This makes the VRE a function in five dimensions, namely a function over all possible 2D directions and therefore also all 3D positions in the medium. Due to its high-dimensionality, the VRE is hard to compute [Jar08].

Atmospheric scattering In the case of the sky and its atmosphere, simplifications can be made. First of all, sky models deal only with a single light source, the sun, which is assumed to have parallel light rays. Secondly, only a limited number of scattering events are considered, usually one or two. This reduces the complexity drastically. For any given viewing direction, single scattering must consider only one scattering angle, namely the one pointing into the direction of the sun. All other angles point into outer space, where no light source is present. Lastly, after the light ray left the atmosphere, deep space is assumed to be a perfect vacuum where no participating medium is present [LF14].

Consider a light ray R_1 emitted by the sun with initial intensity I_{sun}, traveling towards the earth. Eventually, it will enter the atmosphere at point P_{scatmo}. It continues its path, until it interacts with a particle at some arbitrary point $P_{\text{interaction}}$. This interaction will cause out-scattering or extinction events, both of which cause reduction of intensity in the initial direction of travel. Eventually, the light ray scatters at a particle at point P_{scat}. However, the intensity of the light ray has already been reduced and is now $I_{\text{bef-scat}} = I_{\text{sun}}T_\tau(P_{\text{scatmo}} \leftrightarrow P_{\text{scat}})$, where τ is of course the optical depth between the point P_{scatmo} and P_{scat}, or the fraction of photons that still follow the initial direction [LF14].

At point P_{scat}, a certain percentage of photons are scattered towards the virtual camera, also called the viewpoint. The intensity of light leaving the point P_{scat} into that direction is determined by the phase function, so either

Mie or Rayleigh, and is given as $I_{\text{aft-scat}} = I_{\text{bef-scat}} * phase(\theta)$, where θ is the angle to the camera and $phase()$ is either the Mie phase function p_M or the Rayleigh phase function P_R. The event happening at point P_{scat} is an in-scattering event [LF14].

The final path segment is between P_{scat} and P_{cam}, where further attenuation will occur, leaving a final intensity of $I_{\text{final}} = I_{\text{aft-scat}} T_\tau(P_{\text{scat}} \leftrightarrow P_{\text{cam}})$ [LF14]. Figure 2.6 visualizes a single light ray as it interacts with a particle at the point P_{scat} and enters the viewing ray, arriving at the point P_{cam}.

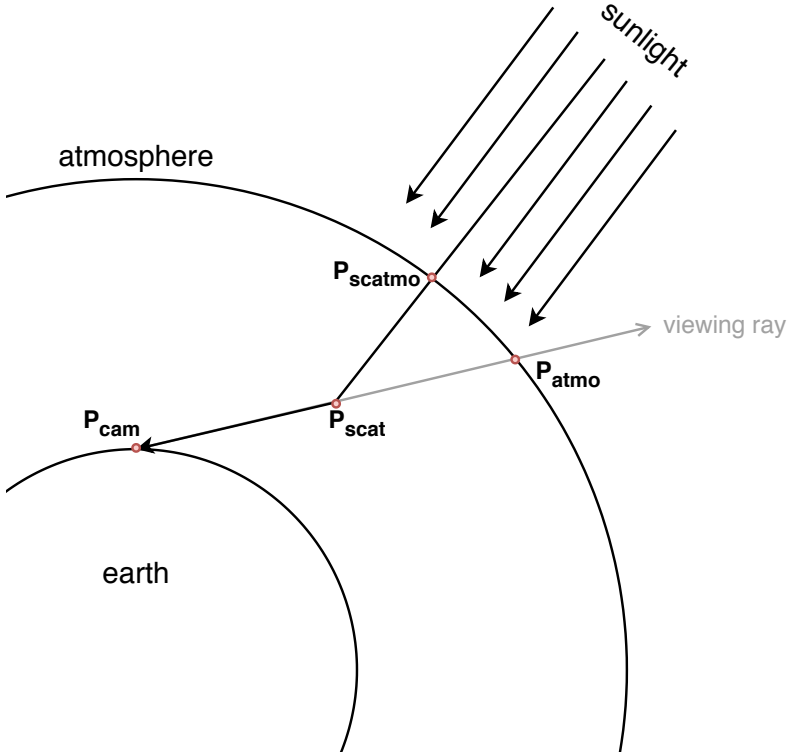


Figure 2.6: A light ray entering the atmosphere at P_{scatmo} being scattered at P_{scat} before arriving at the camera P_{cam}.

The given description of a light ray follows the ray on its logical path from the sun into the camera. However, instead of simulating an infinite amount of parallel light rays, the opposite path is chosen: The total light intensity reaching the camera in a direction ω is given by the sum of all in-scattering events caused by particles along the path between the camera and the point at which the ray exits the atmosphere P_{atmo} in direction ω [LF14].

The final radiance per wavelength λ reaching the camera P_{cam} on a ray pointing to P_{atmo} is therefore calculated as follows:

$$\begin{aligned} I_{cam}(\lambda, s, P_{\text{cam}}) = &\int_{P_{\text{cam}}}^{P_{\text{atmo}}} I_{\text{sun}}(\lambda) T_\tau(P_{\text{scatmo}} \leftrightarrow P_{\text{scat}}) p_R(\theta, \lambda) T_\tau(P_{\text{scat}} \leftrightarrow P_{\text{cam}}) dP_{\text{scat}} + \\ &\int_{P_{\text{cam}}}^{P_{\text{atmo}}} I_{\text{sun}}(\lambda) T_\tau(P_{\text{scatmo}} \leftrightarrow P_{\text{scat}}) p_M(\theta) T_\tau(P_{\text{scat}} \leftrightarrow P_{\text{cam}}) dP_{\text{scat}} \end{aligned} \tag{2.21}$$

where $s = P_{\text{cam}} \rightarrow P_{\text{atmo}}$. The formula above can easily be expressed in words: The total intensity of a light ray from the camera P_{cam} to a point in the atmosphere P_{atmo} along line s in a particular wavelength λ is calculated in two separate parts: first only considering Rayleigh in-scattering, then only considering Mie in-scattering. To determine the light intensity for an individual scatter point P_{scat}, the initial intensity I_{sun} is reduced by attenuation from P_{scatmo} to P_{scat}. The attenuation T_τ is caused by dry air, aerosols and ozone. The remaining intensity $I_{\text{aft-scat}}$, after scattering towards the viewpoint, is calculated using the respective phase function (Rayleigh or Mie). Finally, the leftover intensity is reduced by the attenuation from the scattering point P_{scat} to P_{cam}. The final intensity I_{total}

for a single viewing angle in the camera is found by integrating over all possible scattering points P_{scat} along the path s, and repeating the calculation for all different wavelengths λ.

2.2 Related Work

Sky models serve different use cases: While offline rendering engines like appleseed focus on physical accuracy to generate realistic renderings, games are mainly concerned with the model's speed to keep a high frame rate. Specialized sky models have been developed for these different use cases. They mainly differ in three factors: physical accuracy, runtime complexity and precomputation-time complexity.

Physically-based models deliver extraordinary physical accuracy but have long render times. Analytic models favour low time complexity and low memory complexity over physical accuracy, making them suitable for real-time applications like games or simulations. The most recent development focuses on precomputed models that shift most of the time-complexity of physical models into memory complexity, while still being physically accurate. That allows triple-A games to use physically accurate skies without experiencing frame rate drops. However, precomputed models often use substantial lookup tables that might need to be recomputed for some particular changes in the sky parameters.

2.2.1 Physical Models

In 1987, Klassen introduces the first ever physical sky model in his paper "Modeling the effect of the atmosphere on light"[Kla87]. Physical sky models, also called explicit models or path-traced models, simulate a ray of light through the atmosphere while taking scattering or extinction effects between the light and the participating particles inside the atmosphere into account. Since the colour of the sky is most prominently influenced by scattering and absorption caused by dry air and aerosols, most physical sky models take these effects into account with various degrees of simplifications. Klassen's atmosphere model consists of two layers with constant particle density, neglecting the exponential fall-off of particle density within the atmosphere completely.

Later, Nishita et al. [NSTN93] implement a single scattering model which approximates the particle density using multiple atmospheric shells with various height but constant density within a shell. Three years later, Nishita et al. [NDKY96] extend the previous models by simulating multiple scattering events for more accuracy towards sunrise and sunset scenes. In 2005, Haber, Magnor and Seidel [HMS05] develop a physically-based model for simulating twilight phenomena more accurately than Nishita's models. Their model not only simulates more particles (ozone in particular) but also accounts for factors like humidity and temperature that bend light rays due to a continuous variation of refraction. More recently, Kutz [Kut12] proposed an extensive clear-sky model which takes into account, and models the distribution of the participating particles accurately.

In 2018, Guimera, Gutierrez and Jarabo [GGJ18] continue the development on physical models by connecting the sky parameters to the exact time and location of the viewer. The latter two models are based on a brute-force approach, leading to accurate results while requiring extreme computation times. Haber, Magnor and Seidel claim that their model can be simulated in less than two hours on a conventional machine [HMS05]. Evaluating the model by Guimera et al. takes between 30 minutes and 3 hours on an average consumer PC in the year 2018 [GGJ18]. In contrast, the multi-scattering model by Nishita can be evaluated in 20 seconds, according to measurements on an average consumer PC in the year 2014 by Lopes and Fernandes [LF14]. This illustrates that the more recent development of physical sky models mostly gears towards increasing realism by taking more and more factors into account and mutating them based on the viewer's temporal and geographical conditions.

2.2.2 Analytic Models

The biggest downside of physically-based sky models is their slow execution speed, which makes them far too computationally expensive for real-time applications. Analytic models became popular soon after the release of the first physical sky models, approximating the colour of the sky by an analytic function that, given the view direction and the position of the sun, can determine the colour of the sky. Most of the sky models, such as the famous Preetham model by Preetham, Shirley and Smiths [PSS99] or its improvement by Hosek & Wilkie [HW12],

are based on a function by Perez, Seals and Michalsky [PSM93]. The Perez function offers five Parameters to create a radiance distribution function for the sky.

Analytic models calculate a fit for the five Perez parameters to approximate a reference sky. The resulting analytic model defines the correct Perez parameters based on more intuitive input values, such as the sun's azimuth or the turbidity. While analytic models are simple to implement and extremely fast to evaluate, they are physically less accurate than physical models and are limited in their use cases. Additionally, most physical models are independent of the position of the camera (on the ground or up in the sky) whereas analytic models are usually tied to viewpoints from the earth's surface.

2.2.3 Precomputed Models

The most recent studies in the field of sky models focus on combining the high-performance properties of analytic models with the physical correctness of physical models. These models shift most of the computationally expensive work from execution time to precomputation time. By precalculating nearly all possible values needed at runtime and storing them in large lookup tables, they can achieve real-time performance. While most physical models also perform some precomputation for calculations that have to be done repeatedly throughout the model's evaluation, models like O'Neil [O'N05] precompute every single integral at every possible sun position and viewing direction with every possible camera height.

Later, Bruneton and Neyret [BN08] precalculate physical sky models while taking multiple scattering into account. This results in real-time performance during render time with physically accurate looks. However, the biggest drawback of the precomputed models is their implementation complexity and their inflexibility regarding non-precomputed values: Changing values like the atmosphere turbidity possibly leads to a new precalculation phase, which massively increase the final render time for offline renderers like appleseed [LF14].

3 Nishita Models

The physically-based simulation of the Nishita models has proven to generate phenomenal radiance accuracy [KKN+14], surpassing all of the other existing analytic and precomputed models. In comparison to other physically-based models, the Nishita models are the only ones that merely take seconds to be evaluated [LF14]. The Nishita models are in the sweet spot between high physical accuracy and reasonably fast render times. Nishita et al. have proposed two different sky models: Nishita93 [NSTN93] and Nishita96 [NDKY96]. Nishita93 only implements single scattering events whereas Nishita96 is an extension of the Nishita93 model and supports multi-scattering of any order.

3.1 Nishita93

In 1993, Nishita et al. published a paper titled "Display of The Earth Taking into Account Atmospheric Scattering" in which they propose the sky model Nishita93, the first algorithm for "physically-based image synthesis of the earth viewed from space" [NSTN93]. Although Nishita et al. primarily focused on rendering the earth from outer space while taking the influence of the earth's atmosphere into account, the model has been widely adapted in CG application to render the sky from the ground.

The sky model Nishita93 only models one scattering event per light ray and is therefore referred to as a "single-scattering model". To solve the scattering Equation 2.21, Nishita et al. use numerical integration. A two-dimensional lookup table increases the efficiency of the algorithm: By pre-computing optical depths, a significant load of computational work can be reused during rendering and thus the time to evaluate the sky model can be reduced.

The scattering equation requires the integration of a light ray through the atmosphere. Therefore this integration needs to consider the variation of particle density within the atmosphere, knowing that the air and aerosol particle density decreases exponentially with higher altitudes. Nishita et al. approximate the density variation using a spherical-shell model. Multiple shells with constant particle density but exponentially increasing radii are modelled, allowing for constant integration withing one shell while approximating the effect of exponentially decreasing particle density.

The Nishita93 model considers scattering and absorption effects due to dry air (Rayleigh scattering) and aerosols (Mie scattering). However, as mentioned previously, it ignores multiple scattering of light as well as inter-reflections of light between the earth's surface and particles in the air. Moreover, Nishita et al. neglect absorption in the ozone layer and describe its effects as negligible compared to scattering effects caused by air and aerosols [NSTN93].

3.2 Nishita96

The Nishita93 sky model was later extended in 1996 by Nishita et al. to add support for multiple scattering of any order, where the focus was mainly layed on double-scattering [NDKY96]. The application of single scattering assumes that each light ray from the sun is scattered exactly once. At this specific scattering event, only the direction towards the viewpoint is relevant, since photons progressing in other directions will not be scattered again and will therefore not affect the light reaching the viewpoint. For the application of multiple-scattering, however, light scattering in every direction is important, since the light potentially scatters again before arriving at the viewpoint. To compute double-scattering, an integral over all possible directions needs to be evaluated, which involves a single-scattering computation for each direction according to the Nishita93 sky model. That being said, double-scattering relies on a triple integral which needs to be evaluated for each integration step

along the viewing ray. Since this computation is extremely time consuming, the sky model Nishita96 reduces the number of considered directions for all scattering events to eight. To reduce the evaluation time of the sky model even further, the single-scattering light intensities for each direction of a 3D grid covering the visible space are precomputed and stored in a lookup table. This look up table allows a fast evaluation of the last single-scattering event at each position of the sky by using tri-linear interpolation of values from the lookup table [NDKY96].

3.3 Comparison of Nishita93 and Nishita96

Simulating multiple scattering events is closer to real-world physics compared to single scattering. It is thus not surprising that this method produces more realistic looks, especially during sunrise/sunset scenes, where the amount of scattering events is naturally increased due to the longer travel distances of a light ray within the atmosphere. However, apart from sunrise/sunset scenes, the visible effect of single scattering dominates over multiple scattering during daytime, when the sun has a wider angle from the ground. In Bruneton's [Bru17] perceptual studies of 2016, renders generated by Nishita93 and Nishita96 produced almost identical results. Notably, in Section 13.3 of his perceptual study, Bruneton compares further sky models and states that "participants perceive models whose physical accuracy is 'good enough' as equally realistic" [Bru17]. In sum, both sky models, Nishita93 and Nishita93, deliver comparable visuals. However, Nishita96 is theoretically more physically accurate while suffering from increased implementation complexity, higher rendering times, and bigger memory needs. Furthermore, the heavy precomputation table for the Nishita96 model must be recalculated for every change in the sun's zenith angle, which adds even more rendering time.

For rendering engines such as appleseed, good visuals, reasonable render times, and an implementation complexity that can be maintained by the open-source community is valued more than nearly invisible physical accuracy improvements. For this reason, the Nishita93 model is the more sensible choice for appleseed. Since the single-scattering computations performed in Nishita93 are also part of the Nishita96 multi-scattering model, having the code of the Nishita93 model in place will also contribute to a possible future implementation of the Nishita96 model by the open-source community.

4 Modifications

After Nishita et al. published their latest sky model in 1996, researchers have continued to improve the look and accuracy of physically-based sky models. For example, recent studies have shown that simulating certain additional atmospheric properties like ozone molecules drastically improves the look of the resulting sky under certain conditions. In this study, the Nishita93 model will be taken as the base model for the later implementation, and compatible findings from the latest research are added to the model to improve the final results.

4.1 Ozone Layer

Although the Nishita93 and the Nishita96 model perform well when simulating sunlight through the atmosphere, both models ignore the influence of ozone molecules. While Nishita et al. claim that ozone can be neglected, later studies have highlighted the importance of ozone during sunrise/sunset scenes.

The influence of ozone on the colour of the sky is highlighted by Hulburt [Hul53] in the abstract of his paper "Explanation of the Brightness and Color of the Sky, Particularly the Twilight Sky": "Calculation showed that during the day the clear sky is blue according to Rayleigh, and that ozone has little effect on the color of the daylight sky. But near sunset and throughout twilight ozone affects the sky color profoundly. For example, in the absence of ozone the zenith sky would be a grayish green-blue at sunset becoming yellowish in twilight, but with ozone the zenith sky is blue at sunset and throughout twilight (as is observed), the blue at sunset being due about $\frac{1}{3}$ to Rayleigh and $\frac{2}{3}$ to ozone, and during twilight wholly to ozone."

The importance of ozone is further highlighted by Sloup [Slo02], Haber, Magnor and Seidel [HMS05], Kutz [Kut12] and most recently Guimera, Gutierrez and Jarabo [GGJ18]. Since ozone particles do not scatter light and are only responsible for wavelength-selective absorption, extending Nishita93 with the ozone layer only affects the computation of the optical depths and the dimensionality of the optical depth lookup table, but not the integration of in-scattering events.

To extend the Nishita93 sky model with ozone, a few adjustments must be made. While both air and aerosols are mainly found in the lowest regions of the atmosphere, ozone is found more widespread between $10\,\mathrm{km}$ and $60\,\mathrm{km}$ in height. The density distribution of ozone can be modelled utilizing data provided by Grooß and Russell [GRI05]. Although ozone distribution in the atmosphere varies heavily depending on latitude and date, ozone density can be roughly modelled as a linearly raising function between $10\,\mathrm{km}$ and $32\,\mathrm{km}$ altitude, and an exponentially decreasing function after $32\,\mathrm{km}$. The shell distribution proposed by Nishita et al. in the Nishita93 sky model provides only few shells within the altitudes of $10\,\mathrm{km}$ and $60\,\mathrm{km}$ where ozone is most prominent, leading to a large integration error when computing the optical depths for ozone. However, since the density change of ozone in the atmosphere is generally slower than the drastic change of dry air and dust, the proposed shell model is generally still a valid approximation. Since ozone molecules cause no scattering, only the optical depth calculation must be extended which can be modelled similarly to dry air, because dry air has the same wavelength-selective attenuation properties as ozone.

4.2 Increasing Brightness

In the latest in-depth comparison between different sky models from the year 2016, Bruneton [Bru17] states that the Nishita93 sky model underestimates the sky radiance by one third on average, in comparison to real-world radiance measurements. The underestimation by Nishita93 is a result of neglecting higher-order scattering events which would contribute to a brighter sky. Kider et al. also conclude that Nishita93 generally undershoots the sky radiance values in comparison to real-world measurements [KKN$^+$14]. Hence, the proposed improved Nishita93 implementation scales the final output radiance by a factor of 1.5 to deliver radiance values that are closer to real-world data.

5 Implementation

The key goal of this thesis is to implement a state-of-the-art physically-based sky model into the offline rendering engine appleseed. Sadly, Nishita et al. did not provide a reference implementation for the Nishita93 model. In this chapter, the Nishita93 model is explained in great detail, while adding all of the missing information needed for the implementation, such as important sources for coefficient or radiance values that are not present in the original Nishita93 paper. Further, the changes that need to be made on the Nishita93 model to support ozone molecules are highlighted. Implementation details as well as pseudo-code are provided to make a re-implementation of the proposed model as easy as possible. The source code of the enhanced Nishita93 model can also be found in appleseed's open-source code repository[1].

5.1 Nishita93 Implementation Details

The Nishita93 sky model based on single scattering closely implements the basic principles explained in Equation 2.21. However, the notation made by Nishita et al. often slightly differs from the notation used in Chapter 2. From now on, a mix of both notations will be used: While the definition of points P_x and intensities I_x will be taken as denoted in Chapter 2, the symbols for scattering functions and extinction will be used according to Nishita et al.

Nishita93 Rayleigh Scattering Instead of using the rather complex Rayleigh scattering function from Equation 2.15, Nishita93 uses a simpler, approximated function. Nishita et al. formulate the light intensity after Rayleigh scattering for a given wavelength λ under the angle θ as follows:

$$I(\lambda, \theta) = \frac{I_0(\lambda) K \rho F_r(\theta)}{\lambda^4} \tag{5.1}$$

with K being a constant describing the molecular density at sea level ($0\,\mathrm{km}$), and F_r being the scattering phase function describing the directional characteristics of the scattering function. The phase function F_r is given as:

$$F_r = \frac{3}{4}(1 + cos^2\theta) \tag{5.2}$$

The density ρ depends on the viewpoints altitude h:

$$\rho = e^{\frac{-h}{H_0}} \tag{5.3}$$

where H_0 describes the thickness of the atmosphere if its particle density was uniform. For air particles, H_0 is given as $H_0 = 7994\,\mathrm{m}$.

Nishita93 Mie Scattering Mie scattering is caused by aerosols which are mainly concentrated in the lowest regions of the atmosphere. While the Equation 5.3 is still valid, a different value for H_0 is used when calculating the density ρ. Nishita et al. use $H_0 = 1200\,\mathrm{m}$ to model the distribution of aerosols.

[1] Appleseed's source code is hosted publicly on Github. https://github.com/appleseedhq/appleseed.

Nishita et al. also use a different phase function for Mie scattering which depends not only on the angle θ but also on the asymmetry factor g. They use Cornette's [CS92] improved version of the Henyes-Greenstein function:

$$F_m(\theta, g) = \frac{3(1-g^2)}{2(2+g^2)} \frac{(1+cos^2\theta)}{(1+g^2-2gcos\theta)^{3/2}} \tag{5.4}$$

The asymmetry factor g changes depending on atmospheric conditions such as haze: The more dust particles are present in the atmosphere, the bigger is the asymmetry factor g. The factor g can be defined using the following formula:

$$g = \frac{5}{9}u - (\frac{4}{3} - \frac{25}{81}u^2)x^{-1/3} + x^{1/3} \tag{5.5}$$

$$x = \frac{5}{9}u + \frac{125}{729}u^3 + (\frac{64}{27} - \frac{325}{243}u^2 + \frac{1250}{2187}u^4)^{1/2} \tag{5.6}$$

where u controls the atmospheric condition and can vary from 0.7 to 0.85 [Sek92].

Nishita93 Attenuation Nishita et al. also provide a formula for the extinction ratio per unit length:

$$\beta = 4\pi K a \tag{5.7}$$

The parameter a is different for Rayleigh and Mie scattering. Since Rayleigh scattering is wavelength selective, the parameter a equals $\frac{1}{\lambda^4}$, while for Mie scattering, $a = 1$.

The optical depth of a light ray over a path s over a total distance S for a specific wavelength, described as $t(s, \lambda)$, is given as:

$$t(S, \lambda) = \int_0^S \beta(s)\rho(s)ds = 4\pi a K \int_0^S \rho(s)ds \tag{5.8}$$

Nishita93 Intensity Calculation Given the formulas for scattering and extinction, the intensity of light reaching the viewpoint P_{cam} can be calculated. The light intensity reaching the viewpoint is the remaining light after scattering and absorption between the points $P_{\text{atmo}}P_{\text{cam}}$. The remaining intensity after scattering at any point on the viewing ray is given using Equation 5.1. Before the light can be scattered, it loses intensity due to attenuation (out-scattering and absorption). The light intensity at a scatter point P_{scat} after entering the atmosphere at P_{scatmo} can be calculated using the optical depth Equation 5.8:

$$I_{scat}(\lambda) = I_{\text{sun}}(\lambda) K F_r(\theta) \rho a e^{-t(P_{\text{scat}}P_{\text{scatmo}}, \lambda)} \tag{5.9}$$

I_{sun} describes the light's initial intensity when entering the atmosphere, and $t(P_{\text{scat}}P_{\text{scatmo}}, \lambda)$ is the optical depth between the atmosphere and the scattering point P_{scat}, given by

$$t(P_{\text{scat}}P_{\text{scatmo}}, \lambda) = \int_{P_{\text{scat}}}^{P_{\text{scatmo}}} \beta(l)\rho(l)dl \tag{5.10}$$

where l is just the integration variable.

The light intensity after scattering is attenuated again before reaching the viewpoint P_{cam}:

$$I_{cam}(\lambda) = I_{scat}(\lambda) e^{-t(P_{\text{scat}}P_{\text{cam}}, \lambda)} \tag{5.11}$$

The scattering angle is assumed to be constant since sunlight is parallel. The final equation describing light intensity $I_{cam}(\lambda)$ due to out-scattering and absorption is given as follows:

$$I_{\text{total}}(\lambda) = \int_{P_{\text{cam}}}^{P_{\text{atmo}}} I_{cam}(\lambda)ds = I_{\text{sun}}(\lambda) a K F_r(\theta) \int_{P_{\text{cam}}}^{P_{\text{atmo}}} \rho e^{-t(P_{\text{scat}}P_{\text{scatmo}}, \lambda) - t(P_{\text{scat}}P_{\text{cam}}, \lambda)} ds \tag{5.12}$$

where s denotes the distance from P_{cam} to P_{scat}.

Nishita93 Optical Depth between the Scatter-Point and the Camera The first and easier optical depth needing to be calculated is the optical depth that decreases light intensity after the scattering event, e.g. the optical depth on the viewing ray from the scatter point P_{scat} to the camera P_{cam}

By numerically integrating the Equation 5.8, the optical depth can be found. The integration can be done efficiently by sampling the viewing ray and using trapezoidal integration to calculate the integral for each interval. The optical depth of a sample point P_{samp_i} can be found by adding the optical depth of the interval $P_{samp_{i-1}}P_{samp_i}$ to the optical depth at point $P_{samp_{i-1}}$. It is, therefore, most efficient to start the integration of optical depth from the viewpoint with an optical depth value of 0 and continue with the interval $P_{\text{cam}}P_{samp_1}$. The viewing ray must be sampled to model the exponential decrease of particle density within the atmosphere. At lower altitudes, smaller intervals are desired since the changes in particle density are more drastic. At higher altitudes, where the changes in particle density slows down, larger intervals are sufficient. To account for this behaviour, the atmosphere is modelled as a group of multiple spherical shells with exponentially increasing radii but constant particle density. The particle density is defined to be constant within a shell, allowing to easily integrate from shell to shell. The exponentially increasing shell radii account for the exponential decreasing particle density and ensure that the integration error remains low. A simplified visualization of the Nishita93 shell model with $n = 4$ shells is displayed in Figure 5.1.

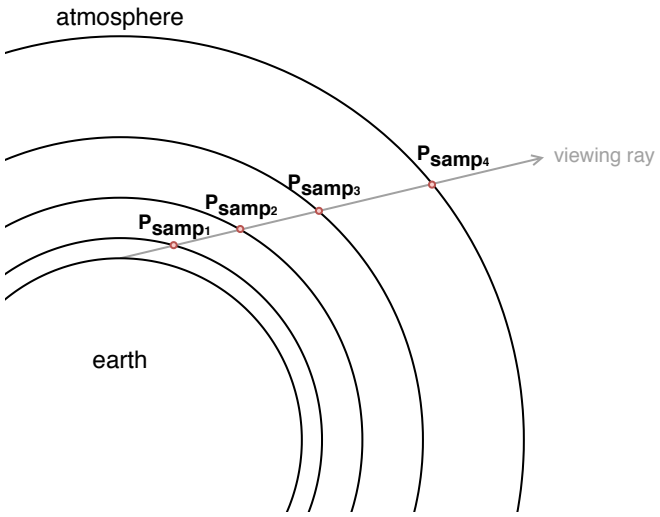


Figure 5.1: Nishita93 multiple-shell model with N=4 shells. The intersections between the shells and the viewing ray act as sample points for the integration.

The optical depth is dominated by Rayleigh scattering, caused by air. Hence, in the original Nishita93 implementation, the radii of the spherical shells are governed by the density distribution function from Equation 5.3 with the reference height of air: $H_0 = 7994\,\text{m}$. As mentioned above, the here proposed model will use a different H_0 to better account for ozone molecules. More details are given in Section 5.2. The optimal radii per shell for a spherical shell model with N shells is in either case given as:

$$r_i = H_0 \log(\rho_i) + R, \tag{5.13}$$

with ρ depending on the number of shells N that are used:

$$\rho_i = 1 - \frac{i}{N} \tag{5.14}$$

where R is the radius of the earth. The first shell, r_1 has the radius of the earth, while the last shell r_n has the radius of the end of the atmosphere. The sampling points for the trapezoidal integration are now taken by the intersection points between the ray and each shell. The density at every sampling point can easily be precomputed once for each shell and reused during rendering. This results in faster computation.

Nishita93 Optical Depth between the Atmosphere and the Scattering-Point The second optical depth must be computed between the scattering point P_{scat} and the atmosphere P_{scatmo}. It is harder to reuse this calculation because both the scattering point P_{scat} as well as the point P_{scatmo} where the scattered ray leaves the atmosphere change. However, making use of the fact that the earth is round and sunlight is parallel, a lookup table for optical depth values can be generated.

For a given point anywhere in the atmosphere, the optical depth can be calculated by integrating from P_{scat} to P_{scatmo}. However, there exists an infinite amount of other points that have the same optical depth. Any defined point P_{scat} at height h has an optical depth l between P_{scat} and P_{scatmo}. A circle is drawn through P_{scat} whose centre lies on the line connecting the centre of the earth with the sun's centre. Every point P on that circle has the same optical depth. Further, the circle can be extruded into the direction of the sun, forming a cylinder. The intersection circles between the cylinder and the atmosphere shells all define circles of uniform optical depth for all points on the circle. Figure 5.2 shows a cross-section of the earth, the atmospheric shells and a cylinder originating at the center of the earth, facing the sun. The intersection point P_{scat} between the cylinder and the shell has the same optical depth as the other intersection point, P'_{scat}.

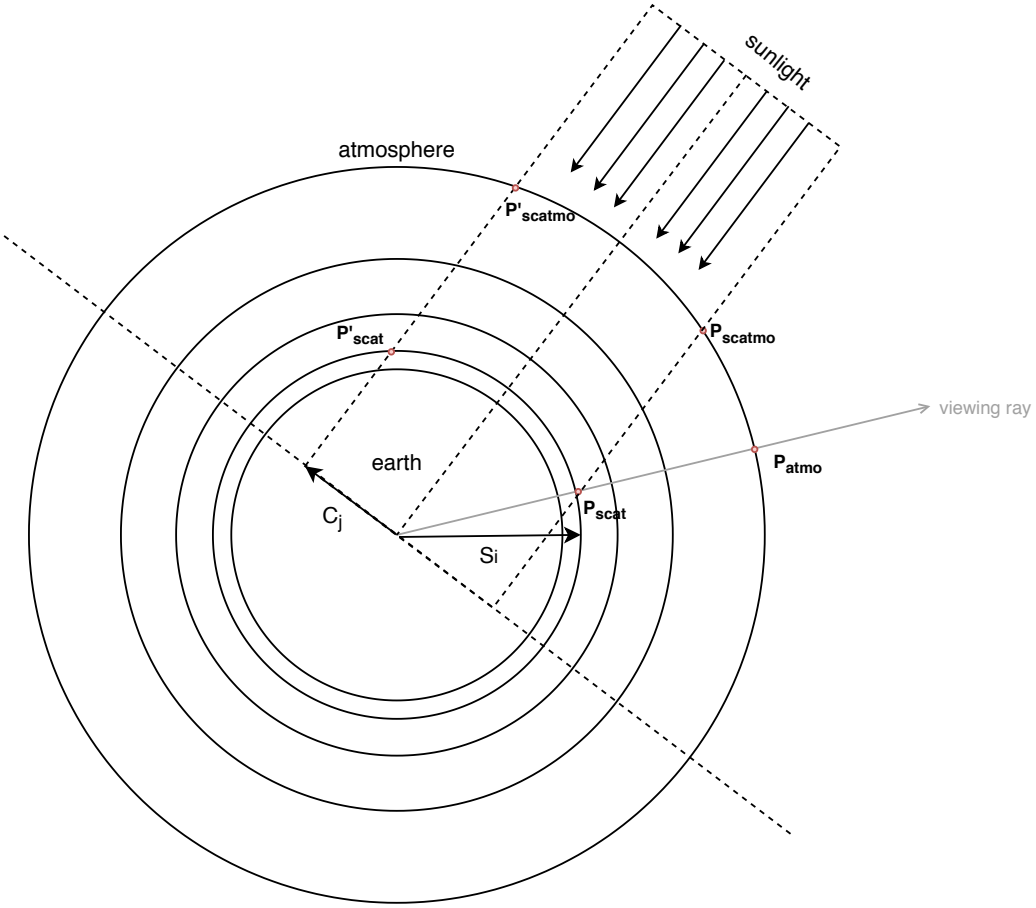


Figure 5.2: Demonstrating the identical optical depths on P_{scat} and P'_{scat} on a shell with the same radius intersecting the cylinder.

Using this concept, a finite set of optical depths between scattering points and the atmosphere can be precomputed and stored in a lookup table. With having N different atmosphere shell radii S_1 to S_N and K different cylinder radii C_1 to C_K, a two-dimensional lookup table (LT) with optical-depth values for a given shell radius S_i and a given cylinder radius C_j can be created[2]. The lookup table can be accessed by defining the shell radius S_i and the cylinder C_j: $[S_i, C_j]$. Given an arbitrary scattering point P_{scat}, its optical depth to the point P_{scatmo} can be determined using the lookup table. First, the surrounding shells S_i and S_{i+1} must be found such that $S_i <= h <= S_{i+1}$ holds, where h is the distance from the point P_{scat} to the center of the earth. Correspondingly, the surrounding cylinders are found, such that $C_j <= d <= C_{j+1}$, where d is the distance between the point P_{scat} and a virtual line connecting the center of the earth with the center of the sun. The optical depth can be looked up by bi-linear interpolation of these values:

$$t(P_{\text{scat}}, P_{\text{scatmo}}) \approx \text{interpolate}(\text{LT}[S_i, C_j], \text{LT}[S_{i+1}, C_j], \text{LT}[S_i, C_{j+1}], \text{LT}[S_{i+1}, C_{j+1}],) \tag{5.15}$$

5.2 Particle Properties

Nishita et al. presented a method to efficiently and accurately simulate light rays and their interactions with particles in the atmosphere. However, the look of the actual sky is determined by the physical properties of the participating particles in the simulated media. While Nishita et al. explain the implementation of basic properties of air and aerosols, some adjustments must be made to the model to allow proper support of ozone particles.

Distribution The particle distribution for air particles and aerosols is modelled with an earth radius of $6\,378\,137\,\text{m}$ and an atmosphere thickness of $60\,000\,\text{m}$. Air and aerosol density distributions were modelled as an exponentially decreasing function, given in Equation 5.3, with $H_0 = 7994\,\text{m}$ for air particles and $H_0 = 1200\,\text{m}$ for aerosols. Since ozone does not follow a simple exponential fall-off, the following density function was used:

$$\rho_{ozone} = \begin{cases} 0, & \text{if h < } 10\,000\,\text{m} \\ (h - 10000)/(32000 - 10000), & \text{if } 10\,000\,\text{m} \leq \text{h < } 32\,000\,\text{m} \\ e^{-(h-32000)/10000}, & \text{if } 32\,000\,\text{m} \leq \text{h} \end{cases}$$

The ozone density function increases linearly from $10\,000\,\text{m}$ to $32\,000\,\text{m}$ and decreases exponentially afterwards.

However, the shell distribution function proposed by Nishita et al. only model very few shells in these heights. This leads to a large integration error for ozone molecules. To better account for ozone, a different shell-distribution must be used. Instead of taking the particle distribution function ρ with the reference height for air molecules, $H_0 = 7994\,\text{m}$, a reference height $H_0 = 15\,988\,\text{m}$ is used, which is doubled the reference height of air molecules. The higher reference height has the effect that more shells are present at higher altitudes. The resulting shell distribution can be seen in Figure 5.3.

The adjusted shell model still concentrates most of the shells in the lowest altitudes but leaves enough shells in the higher regions of the atmosphere to keep the integration error for ozone low. Since the density-change is less severe for ozone molecules than for air or aerosols, having a wider gap between consecutive shells is tolerable. For $n = 64$ shells, the first few shells are already set at $r \approx 250\,\text{m}, r \approx 500\,\text{m}$ and $r \approx 760\,\text{m}$, while the last shells have a radius of $r \approx 46\,500\,\text{m}, r \approx 51\,900\,\text{m}$ and $r = 60\,000\,\text{m}$.

The different density distribution functions, together with the shell heights using the ozone-adjusted shell-model, are shown in Figure 5.3.

[2]Note that Nishita et al. use a slightly different notation: They use the symbol r to denote a shell radius instead of S.

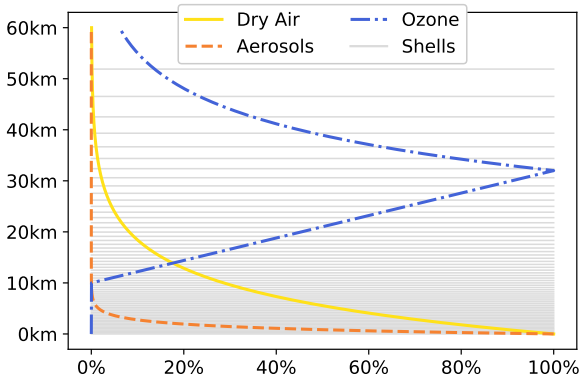


Figure 5.3: Density distribution functions for air, aerosols and ozone. The shell heights from the Nishita93 shell model, adjusted for ozone, are shown in grey. A total number of $n = 64$ shells were used.

Phase Functions The phase functions used for Nishita93 are modelled as three-dimensional functions, describing the probability of scattering events per solid angle. Since both the Rayleigh and the Mie phase functions are axis-symmetric to the incident light direction, their complexity can be reduced to a function of probability per angle.

The resulting phase function describing Rayleigh scattering is given as:

$$F_r(\theta) = \frac{3}{8\pi}(1 + cos^2\theta) \tag{5.16}$$

Similarly, the resulting Mie phase function is given as:

$$F_m(\theta, g) = \frac{3}{8\pi}\frac{1-g^2}{2+g}\frac{1+cos(\theta)^2}{(1+g-2gcos(\theta))^{\frac{3}{2}}} \tag{5.17}$$

The effect on the factor u from Equation 5.6 on the Mie phase function is demonstrated in Figure 5.4.

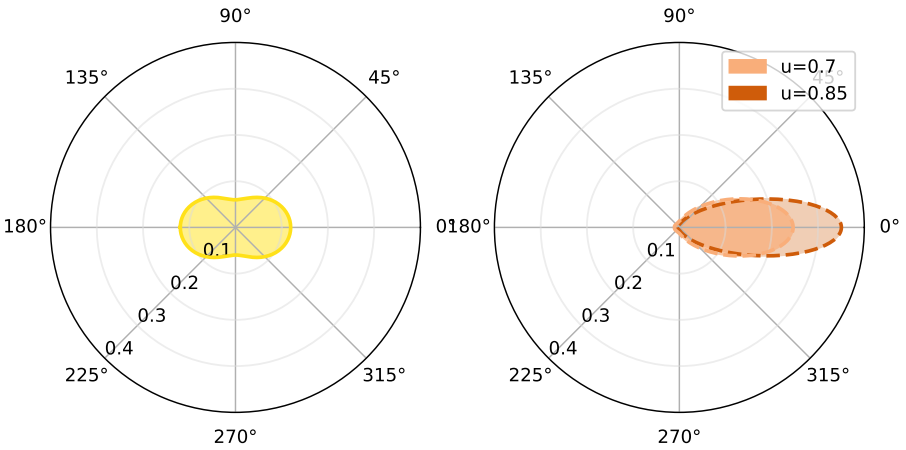


Figure 5.4: Rayleigh (left) and Mie (right) phase functions with incident light from the left. A higher factor u leads to a more narrow but longer Mie phase function.

Coefficients To model the wavelength dependency of the Rayleigh and Mie phase functions, coefficients that describe the scattering amount per wavelengths are used. In the paper "Tables of the Refractive Index for Standard Air and the Rayleigh Scattering Coefficient for the Spectral Region between 0.2 and 20.0 μ and Their Application to Atmospheric Optics", Penndorf [Pen57] publishes a table containing the Rayleigh scattering coefficients for a wide variety of wavelengths. Similar coefficient tables can also be found for ozone particles. Even though they are used similarly, these coefficients originate from different physical properties. While dry air causes wavelength-selective scattering, characterized by the Penndorf Rayleigh coefficients, ozone does not scatter light at all. However, ozone absorbs light wavelength-selective. When modelling the optical depth, both characteristics (out-scattering and absorption) contribute to the same amount and are summarized under the term "attenuation". Hence, calculating the attenuation for dry air is made up solely by out scattering, while for ozone, the attenuation comes from wavelength-selective absorption. Tables for the ozone coefficients are provided by IUP Bremen and can be found online[3].

The effect of the coefficients on the absorption/scattering of particles is visualized in Figure 5.5. The figures show how much of the visible light spectrum between $400\,\mathrm{nm}$ and $700\,\mathrm{nm}$ is scattered/absorbed.

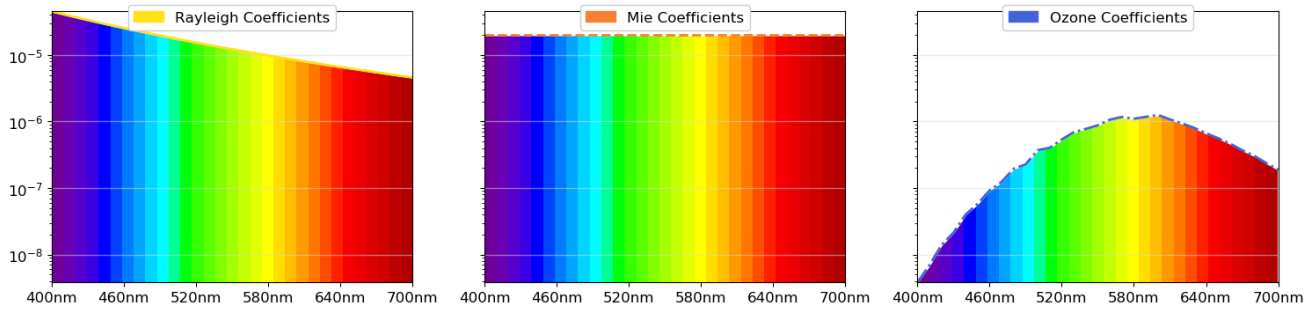


Figure 5.5: Rayleigh (left), Mie (center) and Ozone (right) scattering/absorbtion coefficients with their influence on the wavelength of light, in logarithmic scale. The x-axis holds the wavelength bins from $400\,\mathrm{nm}$ to $700\,\mathrm{nm}$, the y-axis holds the coefficients.

The coefficients of the Rayleigh phase function clearly visualize the tendency of dry air to scatter small wavelengths like blue much more than larger wavelengths. Note that the y-axis of Figure 5.5 is in logarithmic scale, so the tendency to scatter small wavelengths is exponentially higher than scattering large wavelengths. This leads to the overall blue colour of the sky during daytime.

Since the Mie phase function is not wavelength-dependent, its coefficients remain constant over the visible spectrum, as seen in Figure 5.5.

5.3 Main Functions

There have been implemented two sky models by appleseed: The Preetham model and the Hosek & Wilkie model. Appleseed implements all sky models as an environment emission distribution function (EDF). An EDF defines how materials emit light. Therefore, the sky is treated as a source of illumination within appleseed. A normalized three-dimensional vector indicates the direction in which the model shall be evaluated. A radiance spectrum consisting of 31 floating-point numbers, representing bins of $10\,\mathrm{nm}$ size between $400\,\mathrm{nm}$ and $700\,\mathrm{nm}$, will be evaluated for each received directional vector. The input vector can be interpreted as the viewing ray direction for the Nishita93 sky model. The radiance change must be evaluated for each of the 31 wavelength bins.

The implementation of Nishita93 within appleseed has two main components: 1.) Calculation of the radiance for a given input ray starting at the viewpoint, 2.) Calculation of the optical depth for a ray pointing directly at the sun. When using a lookup table, two more main components can be added: 3.) Precomputation and storing of optical depth values, 4.) Lookup and interpolation of optical depth values.

[3] O3 Spectra - Absolutely calibrated ozone absorption cross-sections for 11 temperatures (193K – 293K in steps of 10K) for UV-NIR spectral region. https://www.iup.uni-bremen.de/gruppen/molspec/databases/referencespectra/o3spectra2011/index.html

Viewing-Ray Radiance For every light ray that aims towards the sky, the Nishita Environment EDF determines the ray's initial radiance value. The received vector is set as the viewing ray's direction, while the ray's origin is set to be $1.2\,\mathrm{m}$ above the earth's surface. By setting the "altitude" input parameter of the sky environment EDF, the viewpoint can be lifted into the sky. As Nishita et al. describe in their paper, the viewing ray is integrated from shell-intersection to shell-intersection. The optical depth for each segment is calculated and added to the total optical depth, starting from the viewpoint. In the middle of each segment, the optical depth from the in-scattering ray is evaluated, taking into account the optical depth from the scatter-point to the top of the atmosphere. For each segment, radiance is increased. The value is determined by the total attenuation (caused by the optical depth), the total reduction (caused by angular selectivity of the scattering), the sun radiance and the segment length.

Sun irradiance values from Gueymard [Gue04] are given from $280\,\mathrm{nm}$ to $4000\,\mathrm{nm}$. To get the irradiance values for the 31 bins of size $10\,\mathrm{nm}$ from $400\,\mathrm{nm}$ and $700\,\mathrm{nm}$ that appleseed uses, the irradiance values were averaged. The Rayleigh coefficients are given by Penndorf [Pen57] in Table 3.

An intermediate step of the sky radiance calculation can be seen in Figure 5.6. It illustrates the computation of the sky radiance at the third intersection point between the viewing ray and the atmospheric shells. The segment length marks the part of the viewing ray from the second shell to the third shell. The angle between the viewing ray and the scattering-ray (which points towards the sun) determines the value of the respective phase functions. The optical depth is influenced by both the segment length and the optical depth of the scattered ray, starting at the middle of the segment. Pseudo-code for the sky radiance function can be found in the Appendix 8.1.

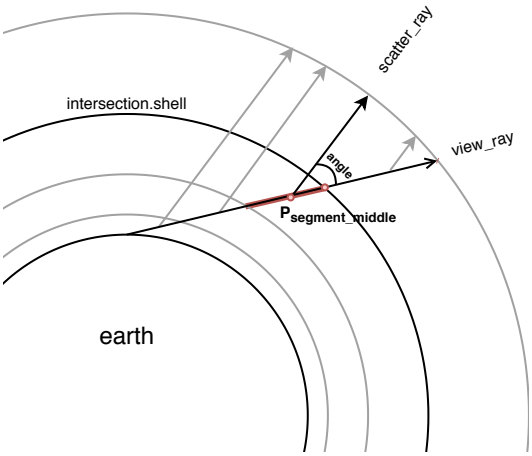


Figure 5.6: Calculation of the radiance at the third shell intersection.

Scattering-Ray Optical Depth To get the optical depth between the scatter point and the top of the atmosphere towards the sun, the scattering-ray must be integrated again. The scattering-ray's origin is set to be the segment middle point, while its directional vector is set to the direction of the sun. This integration is similar to the one inside the sky radiance function. When looping over the intersections of the scattered ray and the atmospheric shells, the optical depth is given by the segment length multiplied with the particle density within the shell. The pseudo-code for the optical-depth calculation is found in Appendix 8.2.

Precomputation of Optical Depths Computing the optical depth for each scattering-ray adds a substantial computational overhead. As described in Nishita93, a lookup table can be used to store optical depth values in a 2-dimensional table, indexed by the cylinder and shell indices. A total of 1024 cylinders were used to achieve nearly indistinguishable results between renderings that have the lookup table enabled/disabled. In the precomputation phase, cylinders facing the direction of the sun are spawned. An optical depth value is evaluated at each intersection point between the cylinder i and shell j. The optical depth calculated at the intersection point is stored in the table at index i, j. The lookup table needs to be recalculated whenever any of the input properties are changed. The pseudo-code implementation is given in Appendix 8.3.

Lookup optical depths With an enabled lookup table, the optical-depth calculation of the scattering-rays is omitted. Instead, the lookup table is used to approximate the optical depth. To get the best possible lookup value, bi-linear interpolation between the four nearest shell-cylinder intersection points with precomputed optical depths is used. The nearest shell-cylinder intersections are formed by the intersections between the biggest-smaller cylinder, smallest-bigger cylinder, biggest-smaller shell and smallest-bigger shell. The weights for the bi-linear interpolations are given by the relative distances from the scatter-point to each shell/cylinder. Pseudo-code for the optical-depth lookup can be found in Appendix 8.4.

Figure 5.7 demonstrates the lookup of an optical depth value from a segment middle point by using the four surrounding cylinder-shell intersection points in which optical depths are stored.

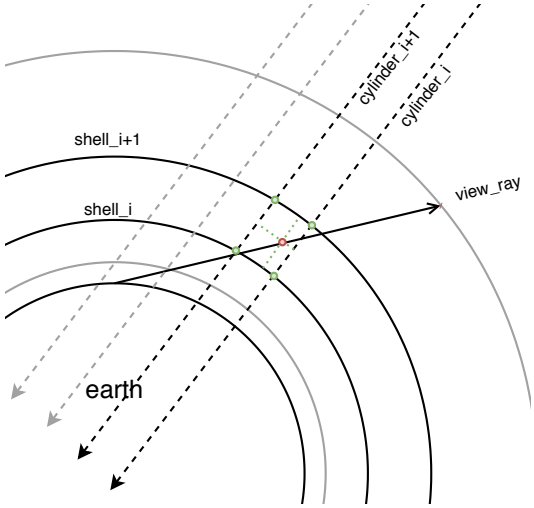


Figure 5.7: Visualization of the four nearest shell-cylinder intersection points (green) for a given scattering-point (red). The weights for the bi-linear interpolation are given by the distance between the scattering-point and the shells/cylinders, visualized as green dotted lines.

6 Results & Analysis

The implementation of the enhanced Nishita93 sky model into appleseed makes it possible to create sky renderings using the proposed model in a ready-to-use rendering engine. Since appleseed also includes two analytic models, the Preetham model and the Hosek & Wilkie model, an accurate comparison between the models in terms of visuals and quantitative factors can be made. To compare the proposed enhanced Nishita93 sky models to other physical sky models currently in use in popular rendering engines, the physical sky model from the open source rendering engine *Cycles* is ported to appleseed and included in the comparison. First, the visual results of all models are analyzed separately, highlighting flaws in the look of the created renders. A comparison between the models illustrates the differences between the models and showcases the strengths of the proposed sky model.

6.1 Results

From this point on, appleseed's enhanced Nishita93 implementation will be referred to as *Applesky20*. The term *Nishita93* will be used when the influence of ozone is disabled and the output as proposed by Nishita et al. is being generated. The results produced by the Applesky20 model are shown in Figure 6.1. The images were rendered with a haze factor $u = 0.8$ at sun theta angles $\theta = 0°$, $\theta = 60°$ and $\theta = 89°$. In order to capture as much of the sky as possible, a Pinhole camera with focal length $f = 0.006$ was used.

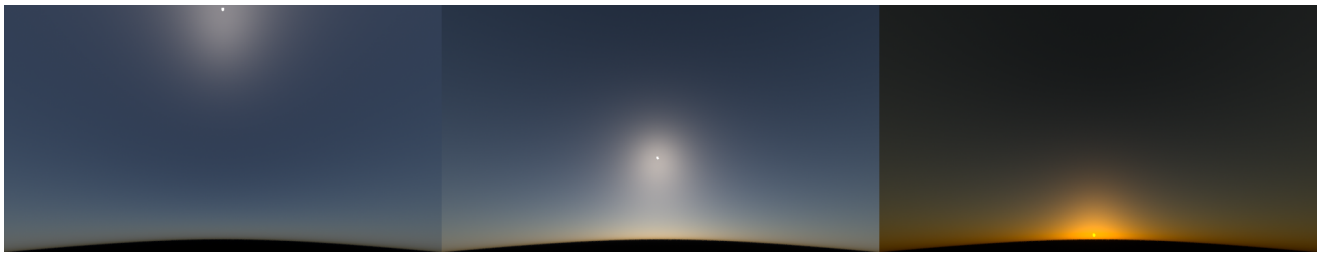


Figure 6.1: Renderings in appleseed from the ground using the Applesky20 model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

Since the Applesky20 model follows a physically accurate, ray-traced approach to calculate the colour of the sky, appleseed users now have the new possibility to render the sky from any position in the atmosphere. The examples in Figure 6.2 show the sky rendered at different altitudes with the camera tilted slightly towards the ground.

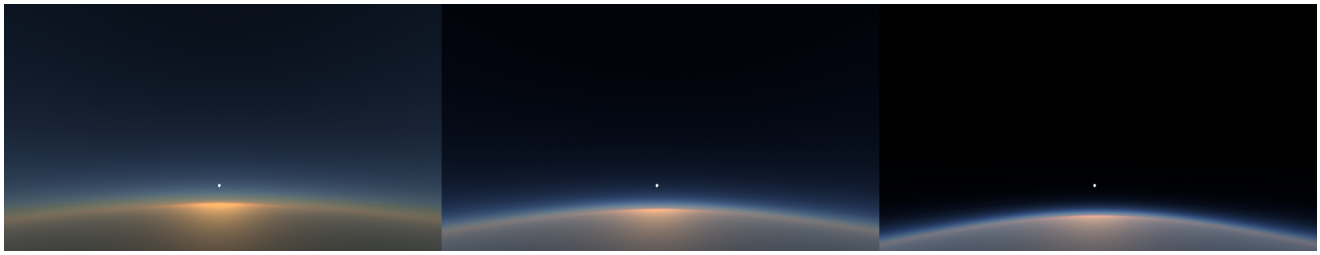


Figure 6.2: Renderings in appleseed using the Applesky20 model with disabled lookup table using a sun theta angle of $\theta = 86°$ at elevations $10\,000\,\mathrm{m}$ (left), $25\,000\,\mathrm{m}$ (middle) and $25\,000\,\mathrm{m}$ (right).

Lifting the camera higher up into space automatically leads to fewer cross-sections with viewing rays and atmospheric shells. Therefore, the computational time needed to evaluate one ray even without a lookup table

becomes smaller. Rendering the sky at the height of $2500\,\mathrm{m}$ with a disabled lookup table takes less time than rendering a viewpoint on the ground with an enabled lookup table. For all atmospheric and space renderings, the lookup table was disabled to give even more accurate results.

The Applesky20 model also supports viewing directions down to earth from outer space, as was the original use case of the Nishita93 model. The demo images from the Nishita93 paper can be reproduced by lifting the camera $1\,500\,000\,\mathrm{m}$ above the ground and looking down to earth, as shown in Figure 6.3.

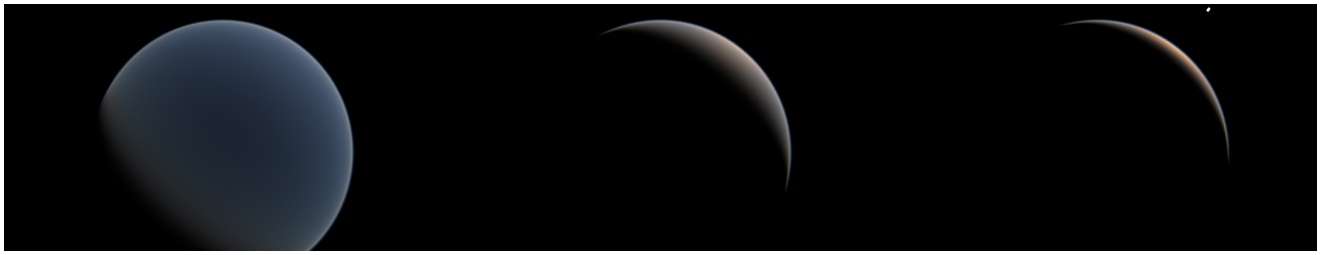


Figure 6.3: Renderings in appleseed using the Applesky20 model with disabled lookup table showing the earth as seen from $1\,500\,000\,\mathrm{m}$ above the ground.

6.2 Model Analysis

In the following section, the analytic sky models that are currently implemented in appleseed, Preetham and Hosek & Wilkie, are analyzed, as well as the new Applesky20 model. The analysis lays the foundation for an in-depth comparison between the sky models available in appleseed. The Applesky20 model will also be compared with a recent Nishita93 adaptation implemented by the ray-traced rendering engine Cycles[1], developed by the Blender Project[2]. The Cycles developers also enhanced the Nishita93 sky model with ozone. However, their sky model implementation does not strictly follow the research paper by Nishita et al. and simplified large parts of the code. This allows an interesting comparison between two different implementation approaches – appleseed on one side, Cycles on the other – both having the same base model but following its underlying research paper to different degrees. To avoid confusion, the Cycles Nishita93 adaptation will be referred to as *Cyclesky20*. Before a comparison can be made, the Cyclesky20 model is analyzed in-detail and its main differences to the Applesky20 models are highlighted.

6.2.1 Preetham Model

The Preetham sky model [PSS99] was one of the first analytic sky models that was fast and easy to implement. Renderings from the Preetham model within appleseed are shown in Figure 6.4. As mentioned by Zotti, Wilkie and Purgathofer, the Preetham model is "one of the most widely used analytic models of skylight luminance in computer graphics" [ZWP07]. Preetham used data from the Nishita96 [NDKY96] model to fit simulated sky radiance data to the Perez [PSM93] sky formula. The most prominent parameter to adapt the look of the Preetham sky model is the turbidity T, which is defined as the ratio of optical depth of the atmosphere that includes haze compared to one only filled with dry air: $T = (\tau_m + \tau_h)/\tau_m$. Preetham supports turbidity values from 2 to 6 and a total of 12 different sun angles. Nevertheless, the Preetham model suffers from a couple of limitations. Firstly, setting extreme turbidity values can even yield negative luminance values near the zenith, which does not occur in nature. Therefore, Preetham should not be used for turbidity values smaller than $T < 1.6$ or sun angles $\theta_s > 85°$. Secondly, Preetham further fails to reproduce the dark regions opposite to the sun near the horizon and instead overshoots the brightness by up to 5 times. Thirdly, Preetham underestimates the brightness at certain regions, most prominently around the sun disk itself [ZWP07].

[1] Cycles is an open-source production rendering engine. https://www.cycles-renderer.org.
[2] Blender is a 3D modelling and rendering package. https://www.blender.org.

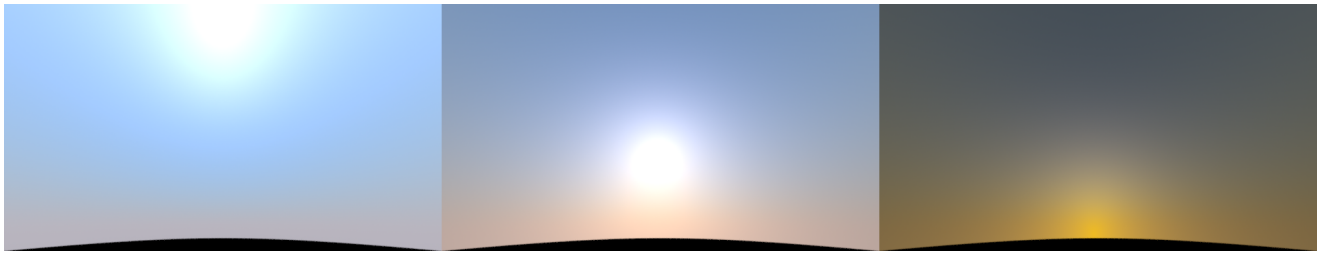


Figure 6.4: Renderings in appleseed from the ground using the Preetham sky model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

6.2.2 Hosek & Wilkie Model

Hosek & Wilkie [HW12] improved on the Preetham sky model in 2012. Instead of taking Nishita96 as ground-truth, Hosek & Wilkie developed their own brute-force path tracer to fit their analytic model to and therefore achieved more realistic results. Hosek & Wilkie also improved on the Perez formula. Demo renderings from the Hosek & Wilkie model within appleseed are shown in Figure 6.5. Since the Preetham model suffered from zero-division errors around the horizon, Hosek & Wilkie introduced a fudge factor of 0.01 that shifts the sky in a way so that the zero division error occurs below the horizon. Since the Hosek & Wilkie sky model only supports viewpoints that face upwards, the zero-division error is shifted to a region below the horizon that is never rendered. Further, they introduced an anisotropic term to better account for Mie scattering. This effect is most prominent in the circumsolar region at high turbidity values, where Mie scattering shows the most impact. Hosek & Wilkie also extended the Perez formula to model the phenomenon that the aureole tends towards the horizon. This can be explained by the high concentration of aerosols at low altitudes, scattering more light into the direction of the viewpoint [Kol12]. The Hosek & Wilkie model was implemented in both appleseed and Cycles.

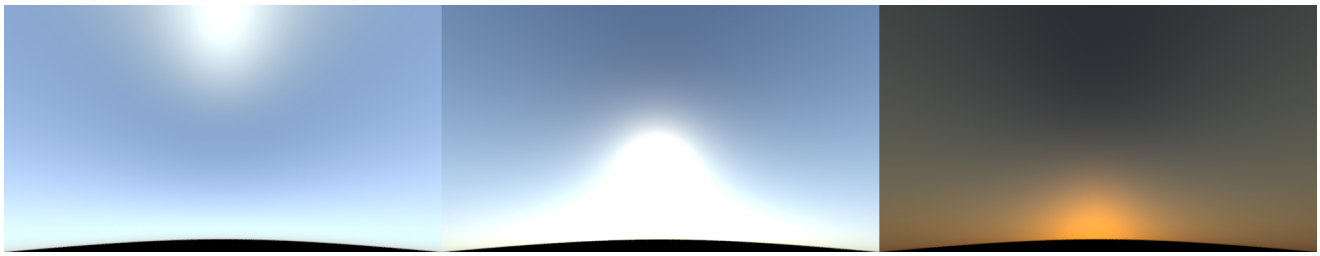


Figure 6.5: Renderings in appleseed from the ground using the Hosek & Wilkie sky model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

6.2.3 Cyclesky20 Model

During the time of writing, the popular open-source 3D creation suite Blender released the new version *Blender v2.9*, which brings a new, physically-based sky model based on Nishita93 to the Cycles rendering engine. In Figure 6.6, sky rendering within Blender are shown.

Compared to the Applesky20 model, it is clearly noticeable that the Cyclesky20 sky model appears much brighter. In fact, turning the strength of the sky texture down to 0.01 within Blender creates results that are comparable to appleseed's Applesky20 model. Renderings of the Cyclesky20 model with reduced brightness are shown in Figure 6.7. The difference in brightness between the models will be discussed in Subsection 6.3.1.

Since Blender – and hence the Cycles rendering engine – is open-source under the GNU General Public License, the implementation of the Cyclesky20 sky model can be evaluated in full detail. Further, when porting the Cyclesky20 model into the appleseed rendering engine, a full performance and visual analysis can be done.

Figure 6.6: Renderings in Blender using the Cycles rendering engine. Renders are taken from the ground using the Cyclesky20 model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

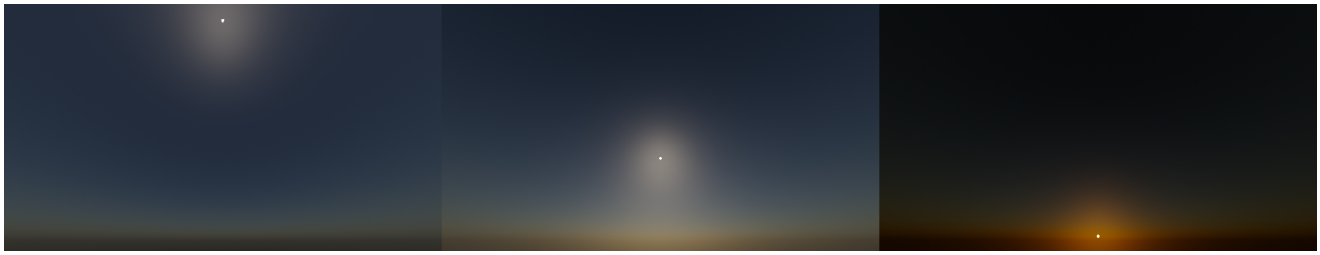


Figure 6.7: Renderings in Blender using the Cycles rendering engine. Renders are taken from the ground using the Cyclesky20 model with 0.01 strength multiplier at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

Although Cycles' source code[3] for the physical sky model is heavily inspired by Nishita93, Cycles' implementation of said sky model differs from the originally proposed Nishita93 model in multiple different ways. A comparison between the Applesky20 and the Cyclesky20 sky model is particularly interesting since both rendering engines have a similar use case and user base. While Applesky20 follows the proven methods by Nishita et al. more strictly, Cyclesky20 is a simplified version of the Nishita93 model. However, both sky models additionally included the effect of ozone.

Source Code Analysis Analysing Cycles' source code reveals the simplifications the Cycles developers made to the original Nishita93 model. The Cyclesky20 model does not make use of the shell model that ensures lower integration errors by assuming constant particles between shells with exponentially increasing radius. Instead, a step-by-step integration approach is used with constant step size and 32 steps between the viewpoint and the atmosphere, and 16 steps between the scatter point and the atmosphere. Neglecting the shell model and only taking into account 16 integration steps for calculating the optical depth will lead to a higher integration error. By leaving out the atmospheric shell model, Cyclesky20 also does not include a lookup table but calculates a double integral for each viewing ray instead. To keep the performance of the sky calculation manageable, the number of integration steps is reduced. Thereby, it is expected that Cyclesky20 performs reasonably well concerning evaluation speed, but is set to deliver wrong radiance values in sunrise/sunset scenes due to large integration errors. Further, Cyclesky20 does not model the earth surface. Therefore, sun rays might travel through the earth and accumulate radiance after leaving the earth's surface again. This leads to strange results when lifting the camera and looking down to the earth. As an effect, evaluating the sky using Cyclesky20 will produce no clearly visible ground.

To calculate the sky radiance for the viewing ray, Cyclesky20 integrates between the ray and the top of the atmosphere. However, for viewpoints in outer space, the integration only happens through deep space, where no radiance can be accumulated. Hence, Cyclesky20 does not support viewpoints in outer space. Even viewpoints that are elevated into the atmosphere are not well handled: Since Cyclesky20 sets its integration limit to the intersection point of the viewing ray and the atmosphere, looking down to the earth while hovering inside of the atmosphere will integrate through the earth until the atmosphere is reached on the other side. Due to the constant

[3]Cycles source-code for Cyclesky20 model. https://developer.blender.org/diffusion/C/browse/master/src/util/util_sky_nishita.cpp.

integration intervals, all integration steps will be inside of the earth. As an effect, all viewing angles facing the earth will receive no radiance values. This effect is shown in Figure 6.9.

Code Porting To eliminate most outside factors when comparing Applesky20 to Cyclesky20, Cycles' source-code of the Cyclesky20 sky model was fully ported to appleseed and integrated as an environment EDF. The source code was only changed so much to make the sky model work inside the appleseed ecosystem. Constants like the sun radiance or coefficients were copied from the Applesky20 model, after ensuring that appleseeds and Cycles' constants only differ marginally. This ensures that all differences in the final renders are caused by implementation differences and not by the use of slightly different constants. The Cyclesky20 model uses a similar set of input parameters. Only the parameter u to control the phase function of Mie particles is missing in Cyclesky20. The Cycles developers chose to set the g value in the Mie scattering equation to a constant value of $g = 0.76$.

Visuals After porting Cyclesky20 into appleseed, it delivers results with comparable brightness to Applesky20. Even without implementing the atmospheric shell model, Cyclesky20 gives remarkably better visuals than Preetham or Hosek & Wilkie, while having the flexibility of a physically-based sky model. The results of rendering Cyclesky20 at different sun theta angles can be seen in Figure 6.8. However, taking a closer look at the sky at low theta angles reveals the lack of accuracy the model provides: Due to the constant step-size in integration, Cyclesky20 greatly underestimates the influence of particles at a low altitude. This is most prominently around the sun disk for a sun angle $\theta = 89°$, where the Cyclesky20 model is too dark. Having only very few integration steps that are evaluated at a height where aerosols are present underestimates Mie scattering. As a result, the characteristic bright glare around the sun appears too dark and rather orange than yellowish.

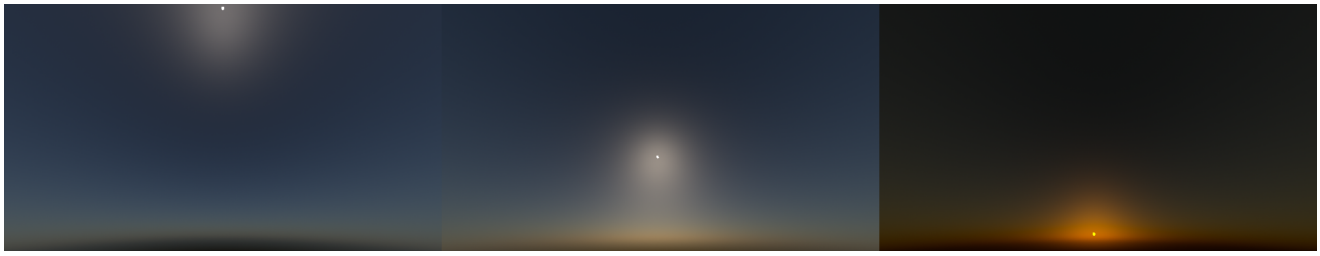


Figure 6.8: Renderings in appleseed from the ground using the Cyclesky20 model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

More drastic errors can be seen when lifting the camera into the sky, a feature that is officially supported by Cyclesky20 and can be controlled inside Blender using the "altitude" parameter. Figure 6.9 shows the camera lifted to altitudes of $10\,000\,\mathrm{m}$, $25\,000\,\mathrm{m}$ and $50\,000\,\mathrm{m}$.

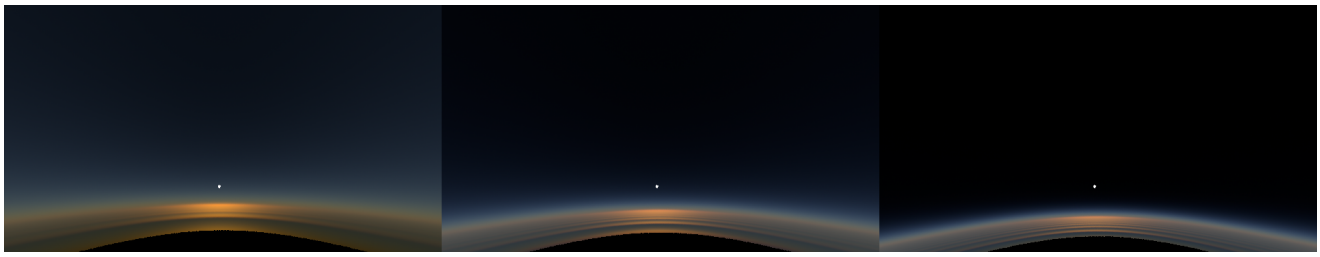


Figure 6.9: Renderings in appleseed using the Cyclesky20 model with a sun theta angle of $\theta = 86°$ at elevations $10\,000\,\mathrm{m}$ (left), $25\,000\,\mathrm{m}$ (middle) and $50\,000\,\mathrm{m}$ (right).

While the sky above the ground is evaluated correctly, every ray pointing towards the surface of the earth results in wrong radiance values. This is caused by the way Cyclesky20 integrates the viewing ray. Rays pointing down to the earth's surface either a.) travel through the earth until leaving the earth's surface again, from where they

continue the integration, or b.) travel through the whole earth and return no radiance values at all. Either case is physically not correct. Visualization of both cases is given in Figure 6.10.

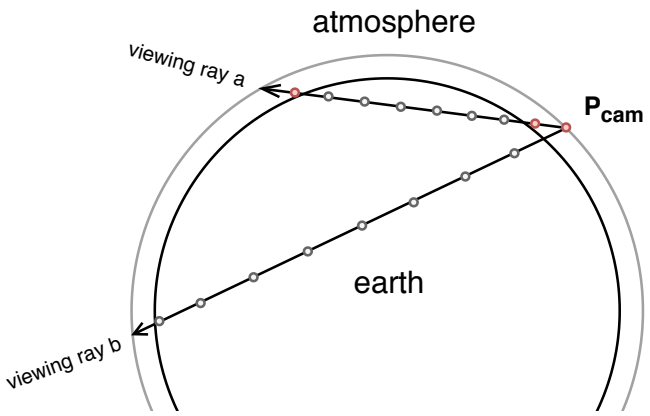


Figure 6.10: Illustration of a viewpoint in the atmosphere and viewing rays pointing down to earth. Using integration with equal step sizes, the viewing ray integrates through the earth. Depending on the ray, the ray a.) leaves the earth on the other side and accumulates wrong radiance values, or b.) returns no radiance at all since all integration steps lie within the earth. Integration points are marked with circles, the ones that lie inside of the earth are drawn in grey colour.

Blender internally overcomes these weaknesses by only rendering rays that point upwards, hence not rendering the view down to earth at all. Results of atmosphere renderings within Blender using the Cyclesky20 model are shown in Figure 6.11.

Figure 6.11: Rendering using Cyclesky20 in Blender with a strength of 0.01 and a sun theta angle of $\theta = 86°$ at elevations $10\,000\,\mathrm{m}$ (left), $25\,000\,\mathrm{m}$ (middle) and $50\,000\,\mathrm{m}$ (right).

Renderings of the Cyclesky20 model from outer space can not be provided because of the previously stated reasons.

6.2.4 Applesky20 Model

The Applesky20 model improves on the original Nishita93 model by adding ozone to the atmosphere simulation and scaling the radiance values by a factor of 1.5 to deliver a more realistic overall sky radiance.

Influence of Ozone The output of the original Nishita93 model can easily be achieved by setting the particle density of ozone to 0, since ozone causes no scattering and only influences the optical depth. Renderings of the Applesky20 model with an ozone particle density of 0.0 are shown in Figure 6.12. A noticeable difference between the Nishita93 and the Applesky20 model can only be seen in the last image, showing a twilight scene where the effect of ozone is most prominent. The results produced by the Nishita93 model show a yellowish sky with minimal blue tones, yielding an unrealistic look.

Ozone particles absorb light mainly in the green, yellow and orange wavelengths. Therefore, by taking ozone particles into account while calculating the optical depth, the sky appears more blue. Since the density of ozone

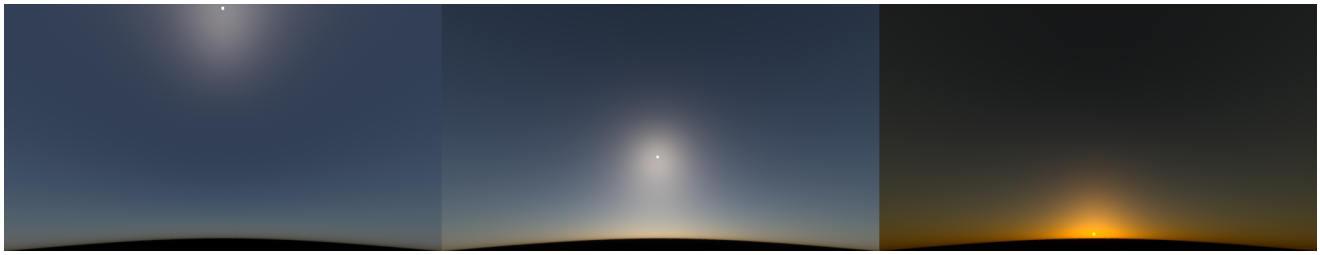


Figure 6.12: Renderings in appleseed from the ground using the Nishita93 model at $\theta = 0°$ (left), $\theta = 60°$ (center) and $\theta = 89°$ (right).

particles is very low, the effect is expected to be most noticeable at low sun theta angles, since light travels a longer distance through the atmosphere. Also, at these low angles, a bluer sky becomes most noticeable, because in normal daylight scenes, the sky is already dominated by a strong blue colour.

Showing the difference in the red, green and blue (RGB) colour channels at high sun theta angles reveals no remarkable difference between rendering the sky with or without ozone. In fact, even for sun theta angles as low as $\theta = 60°$, the effect of ozone is neglectable. Figure 6.13 shows the colour difference between renderings with and without the effect of ozone.

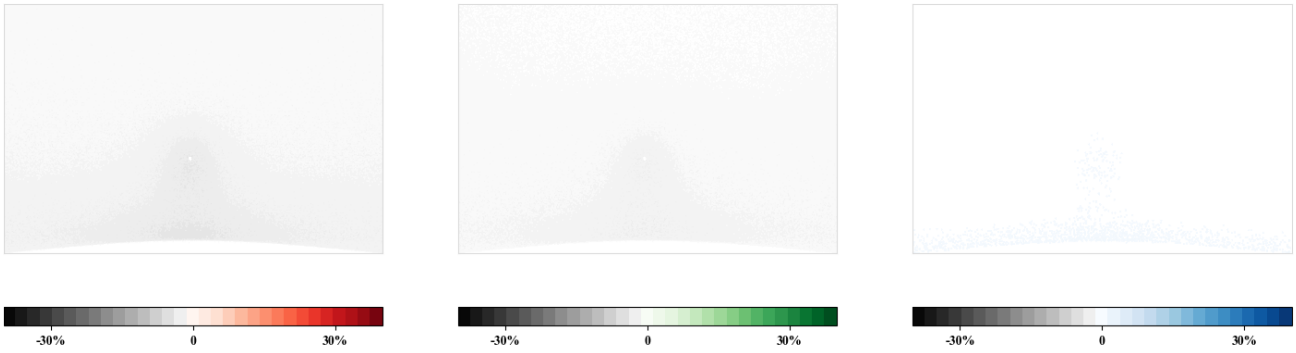


Figure 6.13: Difference in R,G,B values between Applesky20 and Nishita93 for red (left), green (center), blue (right) colour values. All values and relative to the original Nishita93 model. White corresponds to no difference in colour. A stronger colour shows that the Applesky20 model has a brighter colour value, while a dark grey value means that the Applesky20 model has a lower colour value at a certain image region. The images were rendered at a sun theta angle of $\theta = 60°$.

The effect of ozone becomes more apparent for lower sun angles. For sun theta angles as low as $\theta = 89°$, the influence of ozone becomes clearly visible. Figure 6.14 shows the same viewpoints that were compared before but now rendered at a lower theta angle. The red and green colour channels show a significant reduction, implying that ozone particles filtered out more wavelengths at the higher spectrum.

Scaling Factor The radiance scaling factor 1.5 that was suggested by Bruneton has an apparent brightening effect on the resulting sky radiance. Hence a visual analysis is omitted.

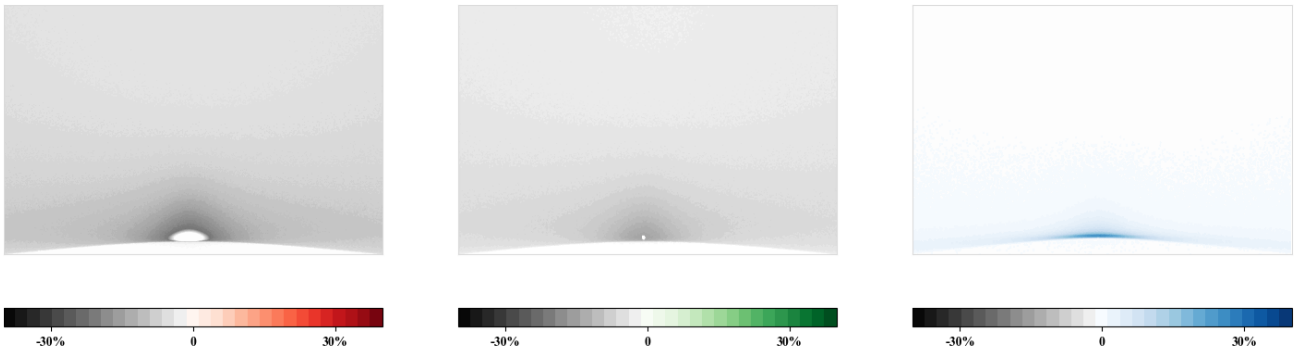


Figure 6.14: Difference in R,G,B values between Applesky20 and Nishita93 for red (left), green (center), blue (right) colour values. All values are relative to the original Nishita93 model. White corresponds to no difference in colour. A stronger colour shows that the Applesky20 model has a brighter colour value, while a dark grey value means that the Applesky20 model has a lower colour value at a certain image region. The images were rendered at a sun theta angle of $\theta = 89°$.

6.3 Model Comparison

An in-depth comparison between the pre-existing sky models in appleseed with the new Applesky20 model and the ported Cyclesky20 model shall reveal the strengths and weaknesses of each model in different situations. The focus of the comparison is twofold: First, an optical study compares the produced visuals between all sky models, showcasing the differences in overall brightness and colour at certain sun theta angles. In the second part, quantitative factors like time and memory complexity as well as possible input parameters are evaluated.

6.3.1 Visual Comparison

Absolute Radiance Scaling Issues Before any in-depth comparison between the sky models can be made, the models' overall differences in brightness must be addressed. When comparing renderings of appleseed's Preetham, Hosek & Wilkie and Applesky20 models, it is clearly noticeable that Applesky20 produces much darker results compared to all other models, while Preetham and Hosek deliver similar brightness values. Renderings of the different sky models in appleseed at a sun theta angle of $\theta = 0°$ are found in Figure 6.15. There are two possible explanations for the noticeable difference in brightness between appleseed's analytic models and Applesky20. Firstly, it is possible that the implementation of the analytic models in appleseed undergo some scaling before being converted to radiance values. This scaling could artificially brighten the sky. Reducing the output luminance of the analytic sky models in appleseed by a factor of π seems to give comparable brightness results for Preetham, Hosek & Wilkie and the Applesky20 sky model. Secondly, it is possible that the sun radiance values used by appleseed are not correct, or that appleseed's environment EDF implementation misinterprets the radiance values produced by the Applesky20 model.

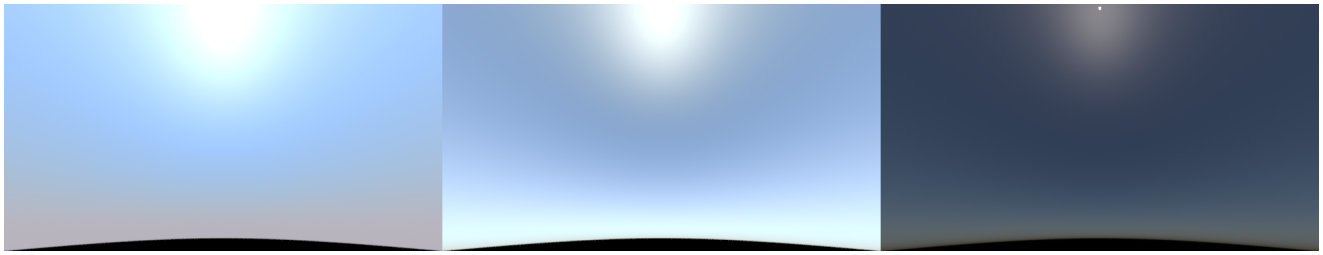


Figure 6.15: Renderings of different sky models in appleseed: Preetham (left), Hosek & Wilkie (center), Applesky20 (right) at a sun theta angle of $\theta = 0°$.

To further investigate this issue, a similar comparison between the Preetham, Hosek & Wilkie and Cyclesky20 sky models is made within Blender. Renderings of all sky models within Blender can be seen in Figure 6.16. Using the default settings for all sky models, the Hosek & Wilkie model looks significantly darker than the Preetham model, and the Cyclesky20 model appears almost white in daylight scenes. This is surprising, since findings from Kider [KKN+14] have shown that both Preetham and Hosek & Wilkie tend to overshoot the sky's brightness, while Nishita93 tends to undershoot. Hence, it would be expected that Cyclesky20 appears darker than the analytic models. It seems that all models undergo some brightness scaling before being displayed. These findings make it impossible to say which sky brightness is actually correct. Thus none of the sky models can be taken as a reference-sky to which appleseed's sky models can be compared to in terms of brightness.

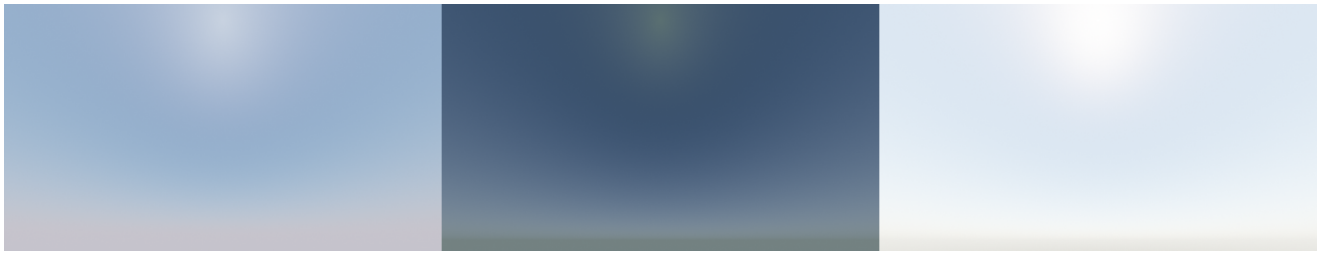


Figure 6.16: Renderings of different sky models in Blender: Preetham (left), Hosek & Wilkie (center), Cyclesky20 (right) at a sun theta angle of $\theta = 0°$.

The analysis raises the question whether the Applesky20 model has an implementation error that causes the sky to be too dark, since Applesky20 produced the darkest looking sky from all sky models encountered so far. However, after porting Cyclesky20 to appleseed, both Applesky20 and Cyclesky20 appear with the same overall brightness. Since the underlying implementation of the Cyclesky20 model did not change when porting the code to appleseed, there seems to be a fundamental difference in how the appleseed and Blender rendering engines interpret the radiance values produced by the sky models. It can be ruled out that Applesky20 suffers from an implementation error that causes the radiance to undershoot. Porting Applesky20 into Blender is expected to deliver sky renderings with a similar overall brightness to Cyclesky20, due to the shared characteristics between Applesky20 and Cyclesky20. The brightness difference by a factor of 100 between the same sky model, rendered in appleseed or in Blender, is a characteristic of the respective rendering engine and not caused by the sky model itself.

For all further analysis, only the sky models within appleseed are considered, and no artificial scaling is applied to any of the Preetham, Hosek & Wilkie, Applesky20 or Cyclesky20 models.

Comparison by Eye In a study by Kider et al. [KKN+14], an in-depth analysis of the radiance accuracy given by different sky models is proposed. It was shown that the sky radiance values by the Nishita93 and Nishita96 models provided greater accuracy than the Preetham or Hosek & Wilkie models.

Hence, the here proposed analysis focuses more on the visual analysis between the models available in appleseed and the Applesky20 and Cyclesky20 adaptations.

Figure 6.17 displays renderings of a test scene at varying sun theta angles for the Preetham, Hosek & Wilkie, Applesky20, and Cyclesky20 sky models, rendered in appleseed. It is clearly visible that both Preetham and Hosek & Wilkie overshoot the sky radiance considerably. This is consistent with the studies done by Kider et al. [KKN+14].

According to Kider et al., radiance values between the sky models differ the most around the sun disk and near the horizon. The analysis of these two illustrates that the renders produced by Preetham tend to have a reddish glow around the horizon while being close to a bright turquoise around the sun disk. Hosek & Wilkie handles the radiance values around the horizon better, while still looking extremely bright around the sundeck at lower sun angles between $\theta = 60°$ and $\theta = 80°$. The least visible difference can be found between the Applesky20 and the Cyclesky20 sky models. Cyclesky20 generally produces more greenish and yellowish results above the horizon, which is a well-known artefact produced by single scattering simulations. These artefacts are less dominant in the

Applesky20 model, although still present to some degree. In lower sun altitudes, the Cyclesky20 model generally produces darker and more orange radiance values around the sun than appleseed's Applesky20 model.

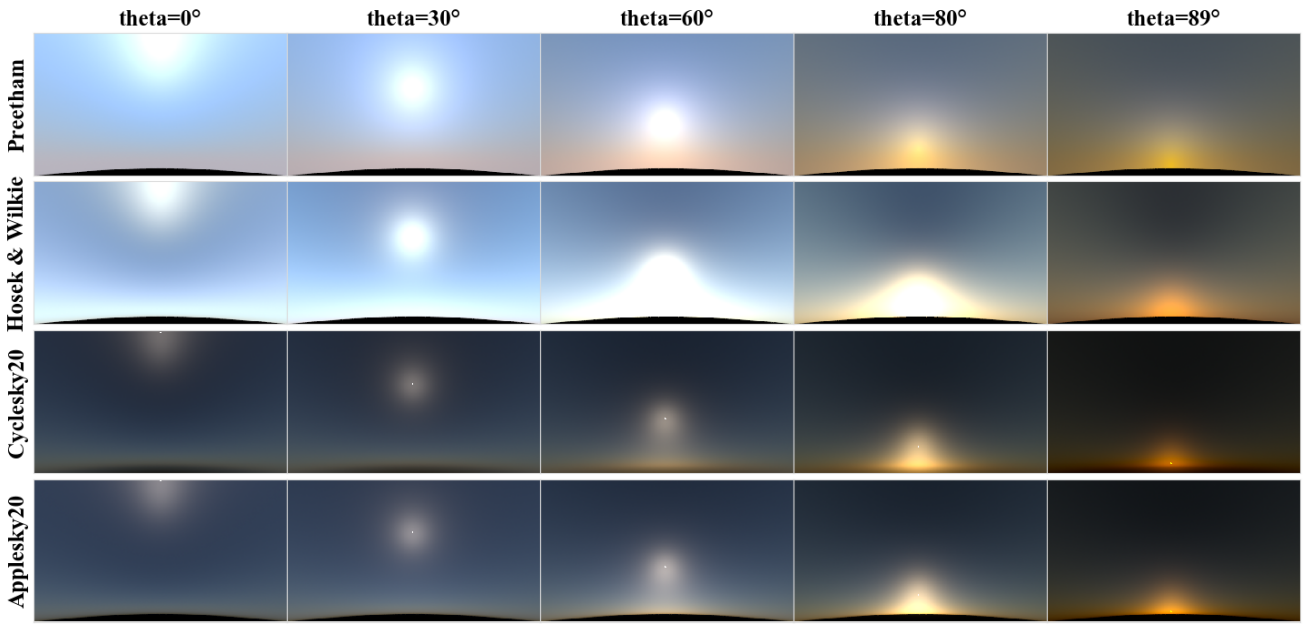


Figure 6.17: Renderings in appleseed from the ground using different sky models: Preetham, Hosek & Wilkie, Cyclesky20 and Applesky20 at different sun theta angles.

Total Brightness Difference The difference in brightness becomes most apparent in Figure 6.18 where the result of a difference image between Applesky20 and the other sky models are plotted as contour plots. Blue values imply that the model is darker than Applesky20; red values imply that the model is brighter than Applesky20. The results of this thesis match the radiance errors measured by Kider et al. [KKN$^+$14]. Preetham produces too bright values throughout the sky, being most accurate around the horizon and around the sun disk. The Hosek & Wilkie model shows similar results but has smaller absolute radiance errors.

Comparing Figure 1b from Kider et al. [KKN$^+$14] with Figure 6.18, it shows that nearly all of observations could be reproduced. A sun theta angle of around $\theta = 60°$ seems to be used in Kider et al. figure. Preetham clearly underestimates the radiance around the sun and all the way down to the horizon. This can also be seen in Figure 6.18. While Hosek & Wilkie also underestimates the sky radiance around the sun disk, it slightly overshoots in the region between the sun and the horizon. This observation was reproduced in this thesis as well.

An interesting comparison can be made between the Cyclesky20 and the Applesky20 models. Cyclesky20 seems to consistently undershoot the radiance values, especially around the sun. This is caused by the large integration error produced by Cyclesky20s integration method. Since these regions are mostly influenced by Mie scattering, caused by aerosols at the lowest region of the atmospheres, the inaccuracy of the values is well understood.

Colour Difference To further investigate in the differences between the sky models regarding brightness and colour, histograms of the different sky models at a sun theta angle $\theta = 30°$ are provided in Figure 6.19. The colour distributions in the Cyclesky20 and Applesky20 sky models have very similar shapes, with Applesky20 fading out more smoothly for brighter colour values. Applesky20 colour values are shifted to the right due to Bruneton's radiance correction factor 1.5. Both analytic models show generally brighter colours, with Preetham being more dominant with red colour values, which is shown clearly around the horizon where Preetham shows a reddish glow. All four models show the same general trend, where the red values lead most towards the darker regions, with green being produced in slightly brighter values, only exceeded by blue.

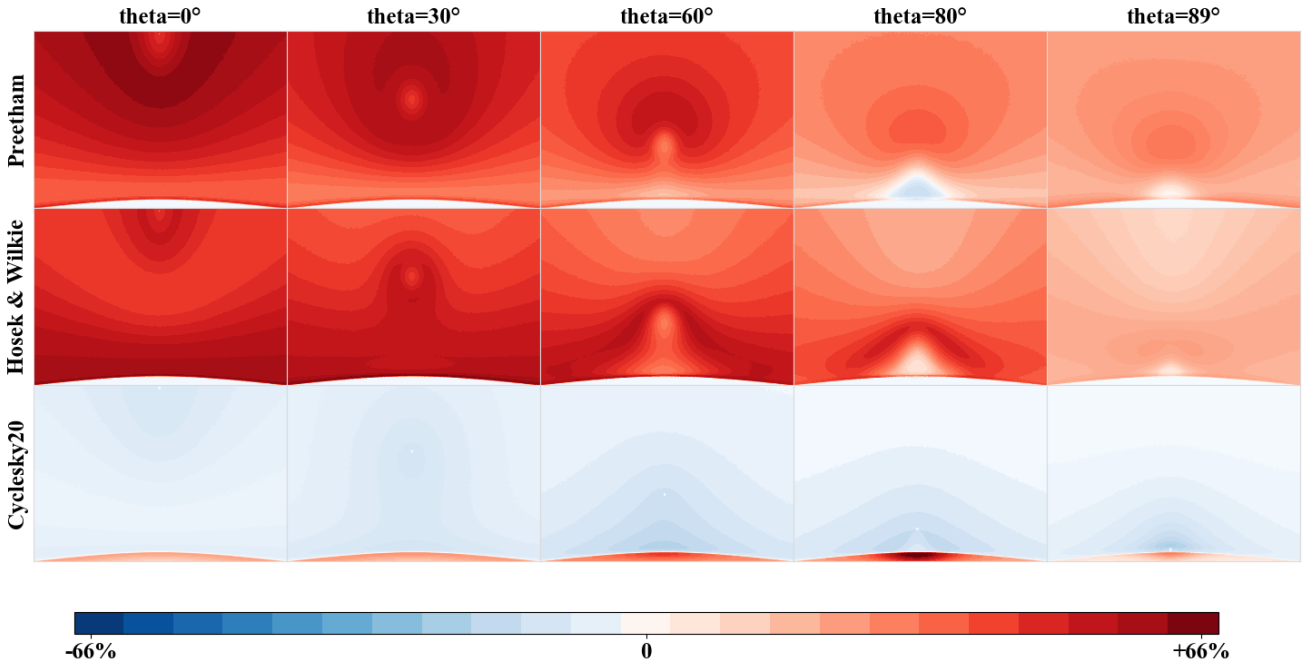


Figure 6.18: Total brighness difference between Preetham, Hosek & Wilkie and Cyclesky20 in comparison with Applesky20, at different sun theta angles. Red values indicate that the model is brighter than Applesky20, while a blue value indicates that the model is darker than Applesky20.

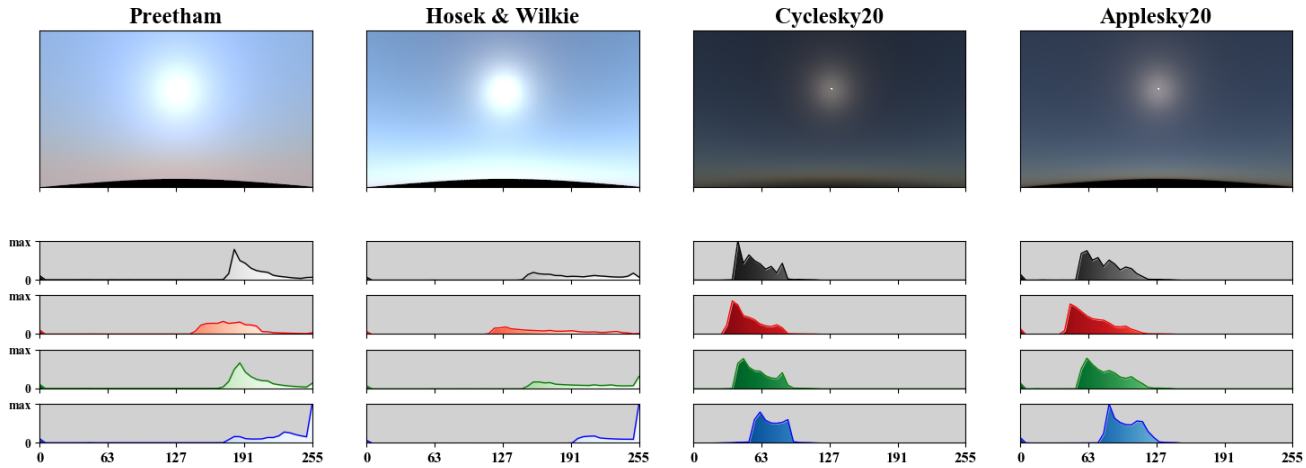


Figure 6.19: Histogram of different sky models, rendered within appleseed from the ground at the sun angle $\theta = 30°$. The topmost histogram applies to a greyscale version of the rendering, the others correspond to the R (red), G (green) and B (blue) color channels. The y-axis of all histograms is scaled from 0 to the maximum color/greyscale value over all images.

When lowering the sun down to the horizon, the sky changes its colours to a more yellowish look. Figure 6.20 shows the different sky models in appleseed rendered at a sun theta angle of $\theta = 85°$. In comparison with Preetham, the Hosek & Wilkie sky model contains much more blue tones and is generally much brighter, showing a clear spike in colour values around 250. Between the physical sky models, the difference is more subtle. While the overall brightness and the red colour channel look almost identical, the blue values in the Applesky20 model show a clear second peak for brighter blue values. Since the blueness of the sunset sky is mostly influenced by ozone, this hints that the Applesky20 model generally does a better job in simulating the influence of the sky than Cyclesky20.

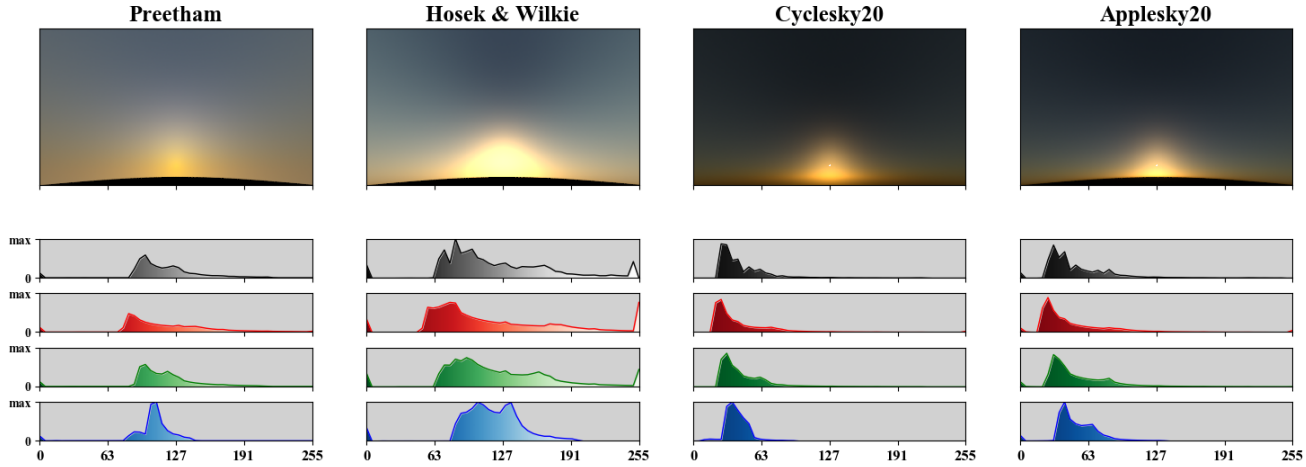


Figure 6.20: Histogram of different sky models, rendered within appleseed from the ground at the sun angle $\theta = 85°$. The uppermost histogram applies to a grey-scale version of the rendering, the others correspond to the R (red), G (green) and B (blue) color channels. The y-axis of all histograms is scaled from 0 to the maximum color/greyscale value over all images.

6.3.2 Quantitative Comparison

CG sky models can be objectively compared by quantitative factors like the number of parameters that can be tweaked (parameterisability), time and memory complexity, and effective rendering times. These quantitative factors show – apart from the visuals – how flexible and usable a certain sky model is. Hence, in the following section, the quantitative factors for the different sky models are being evaluated and compared.

Model Input Parameters Parameterisability is an important aspect of every sky model. Having a large collection of intuitive parameters offers artistic freedom to create different looking skies depending on the use case. Analytic models are mathematical approximations of ground-truth sky data. This makes analytic models inflexible since only scenarios can be created that were provided as inputs for the ground-truth. While it is possible to adjust the sun angle and change – to some extent – the haziness of the sky, most other input parameters only transform the model's output hue, saturation, and brightness.

Physical models, however, offer input parameters that change the physical characteristics in which the light simulation takes place. Changing input parameters like the density of ozone in the atmosphere adapts the output rendering while still being physically correct. This makes it possible to render futuristic-looking atmospheres that are, for example, congested with dust but little to no ozone, while remaining physically correct. By adjusting these parameters, it is even possible to create an atmosphere whose properties are inspired by other planets such as Mars. However, mimicking other planets' atmospheres is limited to modifying the earth's atmosphere's particle densities, since other factors such as the atmosphere thickness and the particle density distributions are not parameterizable by the current implementation of Applesky20. An overview of all possible input parameters for all models is provided in Table 6.1.

Parameter	Preetham	Hosek & Wilkie	Applesky20	Cyclesky20
Sun ϕ angle	✓	✓	✓	✓
Sun θ angle	✓	✓	✓	✓
Air density	x	x	✓	✓
Dust density	x	x	✓	✓
Ozone density	x	x	✓	✓
Elevation	x	x	✓	✓
Haze factor u	x	x	✓	x
Turbidity	✓	✓	x	x
Ground albedo	x	✓	x	x

Table 6.1: Available input parameter for each sky model.

Apart from the suns position parameters ϕ and θ, the input parameters for the analytic and physical models are clearly distinct. However, some parameters trigger the same change in the overall appearance. Changing the turbidity factor of the analytic models has a similar effect as changing the dust density and the haze factor u.

In general, physical sky models offer considerably more parameter flexibility. Only the ground albedo from the Hosek & Wilkie model is present in neither the Applesky20 nor the Cyclesky20 model. Both models assume the ground albedo to be 0 at all times.

Both Preetham and Hosek & Wilkie fitted their analytic models with a sky model of constant air and ozone particles. It is therefore not possible to change these factors. However, depending on the implementation of these analytic models, further parameters to control the brightness, luminance gamma, or saturation might be added. Tweaking these values will bring the sky models further away from delivering realistic values but give artistic freedom to the user to control the look and feel of the sky.

Since the Nishita93 model is physically-based, it is possible to render viewpoints on the ground, in the atmosphere or even from outer space. While Applesky20 fully support all possible viewpoints, Cyclesky20 only supports viewpoints on the ground and has limited support for viewpoints within the atmosphere. The analytic models do not offer this freedom and are bound to the ground.

Time and Memory Complexity The time complexity of the physical sky model can be split into two phases: a.) Precomputation phase and b.) Rendering phase.

Bruneton evaluated the computational complexity of different sky models [Bru17]. His Big-O notation uses an abstract parameter n that is explained as follows in Section 4.1: "A complexity $\mathcal{O}(n^k)$ means that if one uses 2 times more samples in each single numeric integration and along each array dimension, the complexity increases by a factor of 2^k. All the models also have a linear complexity in the number of wavelengths n_λ, which is omitted in the following, *i.e.* $\mathcal{O}(n^k)$ is a shortcut for $\mathcal{O}(n_\lambda n^k)$" [Bru17].

The precomputation phase of the Nishita93 model calculates the optical depth for every shell n_s and every cylinder n_c into the direction of the sun, which is an integral of time complexity $\mathcal{O}(n_s)$. This results in an overall time complexity of $\mathcal{O}(n_s^2 n_c)$.

By using the lookup table, rendering a single ray only involves the integration along the viewing ray and uses precomputed optical depth values for every shell intersection. Having the lookup table in place, the rendering time complexity reduces to $\mathcal{O}(n)$.

In terms of memory complexity, the precomputation table has dimension $n_s x n_c x q$. The quantity q defines the number of values stored in each table cell. For the original Nishita93 model, $q = 2$ since only two optical depths are stored, one for Rayleigh and one for Mie scattering. By adding ozone, a third optical depth value must be stored, resulting in $q = 3$. None the less, memory complexity decreases to $\mathcal{O}(n_s n_c)$ for both models.

The analytic models offer better time and memory complexity than the physical models. All their precomputation effort is permanently saved in a large set of parameters and no precomputation phase is required before rendering an image, leading to a precomputation time and memory complexity of 0 for both the Preetham and the Hosek & Wilkie model. Evaluating the sky radiance during rendering only requires the evaluation of a rather

simple function that is independent of the number of samples n used to create the model, leading to a constant render time complexity of $\mathcal{O}1$.

Effective Render Times Using a computer with 32GB RAM and an AMD Ryzen 1700X processor with 8 cores and 16 threads, a test scene is rendered at different sun theta angle. The scene is rendered at a resolution of 640x360 Pixels with 1 rendering pass and 16 anti-aliasing samples within appleseed studio running in release mode. The total render time is plotted in Figure 6.21.

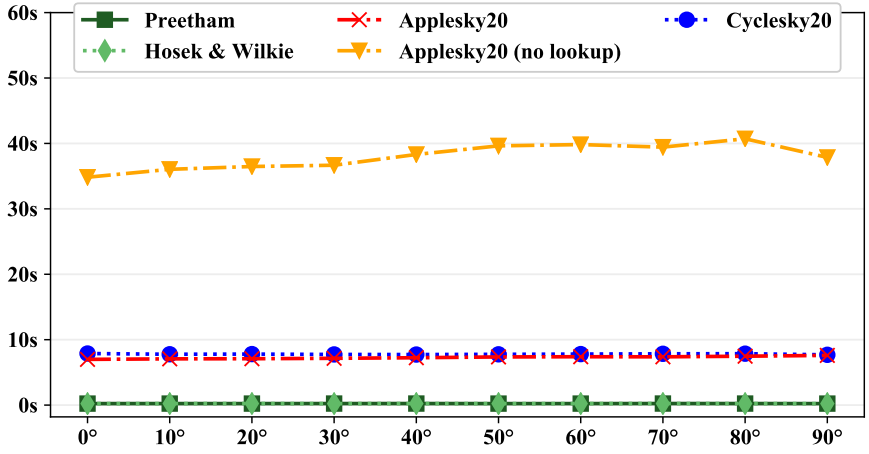


Figure 6.21: Render times in seconds to render Preetham, Hosek & Wilkie, Applesky20, Applesky20 without disabled lookup table, and Cyclesky20 on a Ryzen 1700x with 32GB RAM at different sun-theta angles.

As expected, the analytic models perform much better than the physically-based model. However, using the precomputation table, the render time can effectively be reduced by a factor of ≈ 5. With the help of the precomputation table, the Applesky20 model can consistently outperform Cycles' simplified Cyclesky20 model by $5 - 15\%$.

Sky Model	**Type**	**Supported viewpoints**	**Precompute time**	**Precompute memory**	**Render time**	**Total render time**
Preetham	analytic	ground only	0	0	$\mathcal{O}(1)$	0.229 s
Hosek & Wilkie	physical	ground only	0	0	$\mathcal{O}(1)$	0.225 s
Applesky20	physical	all	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	6.990 s
Applesky20 (no lookup)	physical	all	0	0	$\mathcal{O}(n^3)$	34.832 s
Cyclesky20	physical	ground, atmosphere (limited)	0	0	$\mathcal{O}(n^3)$	7.884 s

Table 6.2: Evaluation of the different sky model parameters in terms of flexibility, complexity and effective render times.

Comparison Table The results of the quantitative analysis can be roughly summarized in Table 6.2 comparing the existing sky models in appleseed (Preetham, Hosek & Wilkie) with the new extended Applesky20 sky model, as well as Cycles' Cyclesky20 model. The Applesky20 sky model with the lookup table disabled was also included in the comparison, since disabling the lookup table has shown to be necessary for rendering the atmosphere under extreme conditions, for example when rendering the earth from outer space while the sun is about to disappear behind the earth.

7 Conclusion and Future Work

The rendering engine appleseed uses ray-tracing to simulate light rays within a virtual scene to create photo-realistic renderings. However, for the most important light source – the sky – appleseed still relies on analytic sky models that deliver inaccurate radiance values based on mathematical approximations. In order to give appleseed's users the possibility to render realistic outdoor scenes, a physically-based sky model which produces physically accurate radiance values is needed. Over the last 30 years, several physically-based sky models have been proposed. The most famous physically-based models are the two sky models by Nishita et al.: the Nishita93 [NSTN93] and Nishita96 [NDKY96] sky models. Both models have been scientifically proven to generate fantastic radiance values that are very close to real-world measurements. Furthermore, the Nishita93 model can be evaluated in just a few seconds on a modern computer, making it applicable for an offline rendering engine like appleseed.

Even though the Nishita models have aged well since their creation in the early 90s, researchers continued to make progress in the field of CG sky simulation and created more accurate and better looking physical sky models. However, most of these models simulate complex phenomenons and therefore take several minutes – sometimes even hours – to compute. Since appleseed is a general purpose rendering engine and must ensure that users are satisfied with the rendering times, implementing these models is not an option.

Nevertheless, some findings from recent research are valid extensions for the Nishita93 model. Recent physically-based sky models often highlight the importance of ozone on the sky's appearance in sunrise and sunset scenes. Unfortunately, the Nishita sky models completely neglect the influence of ozone molecules. As a result, renderings using the Nishita sky models produce a greenish looking sky when the sun is just above the horizon. Moreover, more recent studies comparing physically based sky models highlighted some flaws in the Nishita93 model. In a comparison study, Bruneton [Bru17] discovered that the Nishita93 model underestimates sky radiance by 33% on average.

As a result of these findings from recent literature, a new, physically-based sky model called Applesky20 is proposed in this thesis. It closely implements the Nishita93 single-scattering sky model. Moreover, Applesky20 does simulate ozone particles within the atmosphere, giving its sky a more realistic, blue look in sunrise/sunset scenes. Since the Nishita93 model generally underestimates the sky radiance by 33% and ozone particles reduce the sky radiance even further, a radiance scaling factor of 1.5% was added, which ensures that the sky has an overall brightness that is closer to real-world measurements.

The physically-based Applesky20 model was contributed to the open-source code repository of appleseed. By creating a lookup table to store precomputed optical depth values, rendering times of under $7\,\mathrm{s}$ were achieved. Applesky20 even supports viewpoints from within the atmosphere or from space. Appleseed users therefore have – for the first time ever – the possibility to render the earth as seen from space. Since Applesky20 simulates light rays traveling through the atmosphere, it is possible to modify atmospheric properties like the air, aerosol and ozone density that change the look of the sky, all while remaining physically accurate. By changing these input parameters, it is possible to create futuristic looking skies or – to some extent – even skies of other planets.

During time of writing, the popular 3D creation suite Blender released a new version of its Cycles rendering engine, which also includes a new, physically-based sky model. Cycles' sky model is an adaptation of the Nishita93 model as well. Since Cycles' rendering engine is open-source, the unique possibility to compare two new sky models that were both based on Nishita93 presented itself. An analysis of Blenders source code revealed that Cycles' sky model implemented a simplified version of the Nishita93 model and neglects many of the implementation details given by Nishita et al. For example, Cycles' sky model does not implement the atmospheric shell-model that was one of the key-extensions of the Nishita93 sky model compared to its predecessor Klassen [Kla87]. Nonetheless, Blender also recognized the importance of ozone. Their sky model, which was named Cyclesky20, simulates ozone particles within the atmosphere.

The source code of the Cyclesky20 model was fully ported to appleseed to allow a pixel-perfect visual comparison. Porting Cyclesky20 also ensures that the influence of the underlying rendering engine is eliminated and that rendering times are comparable. Analyzing the source code of Cyclesky20 revealed that its simplified integration method limits the model's usability: Cyclesky20 does not support viewpoints in outer space and cannot render the view down to earth for viewpoints within the atmosphere.

An analysis between appleseed's sky models Preetham [PSS99], Hosek & Wilkie [HW12], Applesky20, and the ported Cyclesky20 model has shown that both physical models (Applesky20, Cyclesky2) generate much more accurate and realistic results than the analytic models (Preetham, Hosek & Wilkie). In a direct comparison between Applesky20 and Cyclesky20, it is evident that Applesky20 produces more accurate radiance values due to its superior integration method, has no viewpoint limitations, offers a wider range of input parameters to tweak the look of the sky, while also having faster rendering times than Cyclesky20.

The visual analysis between all sky models revealed brightness issues for the two physically-based models: Compared to the analytic models, both Applesky20 and Cyclesky20 look significantly darker. The brightness difference must be an artifact of the appleseed rendering engine, since renderings using the same Cyclesky20 models within Blender produces skies that are approximately 100x brighter than appleseed's renderings. It is likely that appleseed uses wrong radiance values for the sunlight. Thus, further investigation is needed to resolve this issue. Meanwhile, Applesky20 offers a radiance-multiplier parameter that can be tweaked to artificially brighten the sky. Setting the brightness multiplier to 3 generates renderings that have a comparable overall brightness to the Preetham or Hosek & Wilkie models.

The proposed Applesky20 model is a relevant extension for the appleseed rendering engine. As a future improvement, the Applesky20 model could be extended to support multiple-scattering according to the Nishita96 model. Even though the influence of single-scattering dominates over multiple-scattering during most times of the day, multiple-scattering becomes extremely important when rendering twilight scenes. Adding multi-scattering on top of the already existing single-scattering would give the appleseed users the flexibility to choose how many scattering events should be simulated, letting them effectively trade physical accuracy with rendering time, if desired.

The Applesky20 model is – of all the models it has been compared to – the only one that supports rendering the earth from outer space. However, the earth's surface is always rendered black. By giving users the possibility to assign a material to the earth's surface, highly realistic earth renderings from outer space could be created. Similarly, assigning the earth a reflective material would simulate the ground albedo, the only parameter Hosek & Wilkie currently offer that Applesky20 is still missing.

Lastly, because of the way Applesky20 simulates light rays through the atmosphere, Applesky20 is theoretically not limited to only render the earth's sky. By changing the physical properties and composition of the atmosphere, atmospheres from other planets could be simulated as well. However, multiple parameters are currently hard coded into the Applesky20 source code that restrict the simulation of other planets. Such parameters are: the radius of the planet, the thickness of the atmosphere and the density distribution functions for all particles. By giving users the possibility to set these parameters as inputs for the Applesky20 model, they could simulate the atmosphere of any planet, as long as the look of the planet's sky is mainly influenced by dry air, aerosols and ozone. The current open-source implementation of the Applesky20 model even makes it possible to easily extend the source code and simulate more particles.

Appleseed's new Applesky20 model is highly competitive in comparison with the newest sky models of other rendering engines such as Cycles. Fortunately, there is still room for improvement and with some additional effort, Applesky20 could become a multi-planetary sky model suited for nearly all situations.

8 Appendix

8.1 Pseudocode

Listing 8.1: Compute the sky radiance for a given viewing-ray and sun-direction

```
function sky_radiance(viewing_ray, sun_dir, use_lookup_table)
    output_spectrum = Spectrum(0)
    optical_depth = 0

    // Determine Rayleigh and Mie phase by angle between view- and sun direction
    angle = angle_between(viewing_ray, sun_dir)
    rayleigh_phase = rayleigh_phase_function(angle)
    mie_phase = mie_phase_function(angle)

    // Loop over intersection points with viewing ray and atmosphere shells
    // The intersections are sorted by increasing distances
    for each ray_shell_intersection:

        // Determine distance that ray travels in the current shell
        segment_length = current_intersection.distance - previous_intersection.distance

        // Continue along viewing ray to P_scat to the middle point of the segment
        segment_middle_point = viewing_ray + intersection.distance - segment_length / 2

        // Create new ray originating at the scatter point and points towards the sun
        scatter_ray = Ray()
        scatter_ray.origin = segment_middle_point
        scatter_ray.direction = sun_dir

        // Stop if segment lies behind earth and is therefore not visible
        if intersects_earth(scatter_ray) or behind_earth(scatter_ray):
            break

        // Increase optical depth by particle density values of current shell
        rayleigh_density = intersection.shell.rayleigh_density
        mie_density = intersection.shell.mie_density
        optical_depth.rayleigh += segment_length * rayleigh_density
        optical_depth.mie += segment_length * mie_density

        // Lookup/compute the optical depth from the scatter point to the sun
        if use_lookup_table:
            light_optical_depth = lookup_optical_depth(scatter_ray)
        else:
            light_optical_depth = optical_depth(scatter_ray)

        // The total optical depth is the optical depth from the sun to the scatter
        // point + the optical depth from the scattering-point to the viewpoint
        total_optical_depth = light_optical_depth + optical_depth

        // Total extinction is the product of the coefficients and the optical depth
        total_extinction = rayleigh_coefficient_spectrum * total_optical_depth.rayleigh
                         + mie_coefficient_spectrum * total_optical_depth.mie
        total_attenuation = e^(-total_extinction)

```

```
        // Factor in the phase functions and the wavelength dependent coefficients
        total_reduction = rayleigh_phase * rayleigh_density *
            rayleigh_coefficient_spectrum
                        + mie_phase * mie_density * mie_coefficient_spectrum;

        // Decrease sun radiance by attenuation and reduction over the segment
        output_spectrum += total_attenuation * total_reduction * sun_radiance_spectrum *
             segment_length;
    return output_spectrum
```

Listing 8.2: Compute optical depth for a given scatter-ray pointing towards the sun

```
function optical_depth(scatter_ray)
    optical_depth = 0

    // Loop over intersection points with scatter ray and atmosphere shells
    // The intersections are sorted by increasing distances
    for each ray_shell_intersection:

        // Determine distance that ray travels in the current shell
        segment_length = current_intersection.distance - previous_intersection.distance

        // Increase optical depth by particle density values of current shell
        rayleigh_density = intersection.shell.rayleigh_density
        mie_density = intersection.shell.mie_density
        optical_depth.rayleigh += segment_length * rayleigh_density
        optical_depth.mie += segment_length * mie_density

    return optical_depth
```

Listing 8.3: Precompute all optical depths and store them in the lookup table

```
function precompute_optical_depths(sun_dir)
    // Create vector perpendicular to sun direction in the earths center
    sun_dir_perpendicular = create_perpendicular_vector(sun_dir)

    // Loop over number of cylinders
    for n_cylinder in n_cylinders:
        cylinder_radius = n_cylinder * cylinder_width

        // Create ray that travels along the border of the cylinder towards the sun
        cylinder_border_ray = Ray()
        cylinder_border_ray.origin = sun_dir_perpendicular * cylinder_radius
        cylinder_border_ray.direction = sun_dir

        // Loop over number of shells
        for n_shell in n_shells:
            shell = shells[n_shell]

            // Check if cylinder border intersects shell
            if cylinder_border_ray in shell:
                // Find intersections point
                intersection_point = shell.intersect_with(cylinder_border_ray)

                // Create new ray from intersection point towards the sun
                scatter_ray = Ray()
```

```
                scatter_ray.origin = intersection_point
                scatter_ray.direction = sun_dir

                // Calculate and store the optical depth of the scatter ray
                light_optical_depth = optical_depth(scatter_ray)
                lookup_table[n_shell][n_cylinder] = light_optical_depth
            else:
                // Take value of previous cylinder
                lookup_table[n_shell][n_cylinder] = lookup_table[n_shell][n_cylinder -
                    1]
```

Listing 8.4: Lookup optical depth value for a given scatter-ray from the lookup table

```
function lookup_optical_depth(scatter_ray)
    sun_dir = scatter_ray.direction

    // Determine radius of cylinder that touches scatter_ray
    radius = distance_point_line(ray.origin, earth_center, sun_dir)
    cylinder_raw_index = radius / cylinder_with

    // Raw index of 3.2 indicates that interpolation has to be done between
    // cylinder 3 and 4, but cylinder 3 has 80% of the weight and cylinder
    // 4 only 20% since 3.2 is nearer to 3 than 4
    cylinder_index = int(raw_cylinder_index)
    second_cylinder_weight = cylinder_raw_index - cylinder_index
    first_cylinder_weight = 1 - second_cylinder_weight

    // Determine index of closest shell that includes scatter_ray
    shell_index_raw = find_shell_index(ray)

    // Shell index of 3.2 indicates that interpolation has to be done between
    // shell 3 and 4, but shell 3 has 80% of the weight and shell
    // 4 only 20% since 3.2 is nearer to 3 than 4
    shell_index = int(raw_shell_index)
    second_shell_weight = shell_raw_index - shell_index
    first_shell_weight = 1 - second_shell_weight

    // Calculate weighted averages
    avg_cylinder_depths_1 = lookup_table[shell_index][cylinder_index] *
        first_cylinder_weight + lookup_table[shell_index][cylinder_index + 1] *
        second_cylinder_weight;
    avg_cylinder_depths_2 = lookup_table[shell_index + 1][cylinder_index] *
        first_cylinder_weight + lookup_table[shell_index + 1][cylinder_index + 1] *
        second_cylinder_weight;
    looked_up_depth = avg_cylinder_depths_1 * first_shell_weight + avg_cylinder_depths_2
         * second_shell_weight;

    // Return weighted average of enclosing shell-cylinder intersection values
    return looked_up_depth;
```

Bibliography

[BN08] Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. *Computer Graphics Forum*, 27(4):1079–1086, 2008.

[Bru17] E. Bruneton. A qualitative and quantitative evaluation of 8 clear sky models. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2641–2655, 2017.

[BTB+19] François Beaune, Esteban Tovagliari, Luis Barrancos, Stephen Agyemang, Sagnik Basu, Mandeep Bhutani, Lovro Bosnar, Rafael Brune, Matt Chan, João Marcos Mororo Costa, Herbert Crepaz, Junchen Deng, Jonathan Dent, Mayank Dhiman, Dorian Fevrier, Karthik Ramesh Iyer, Dibyadwati Lahiri, Kevin Masson, Gray Olson, Achal Pandey, Jino Park, Sergo Pogosyan, Bassem Samir, Oleg Smolin, Thibault Vergne, Luke Wilimitis, and Lars Zawallich. appleseed, 2019.

[CS92] William M. Cornette and Joseph G. Shanks. Physically reasonable analytic expression for the single-scattering phase function. *Applied Optics*, 31(16):3152–3160, June 1992.

[FC13] G. Farmer and John Cook. Introduction to earth's atmosphere. In *Climate Change Science: A Modern Synthesis*, pages 179–198. Springer, December 2013.

[GGJ18] David Guimera, Diego Gutierrez, and Adrian Jarabo. A physically-based spatio-temporal sky model. In *Proceedings of the XXVIII Spanish Computer Graphics Conference*, CEIG '18, page 29–37, Goslar, DEU, 2018. Eurographics Association.

[GRI05] J.-U. Grooß and James M. Russell Iii. Technical note: A stratospheric climatology for o_3, H_2O, CH_4, NO_x, HCl and HF derived from haloe measurements. *Atmospheric Chemistry and Physics*, 5(10):2797–2807, October 2005.

[Gue04] Christian A Gueymard. The sun's total and spectral irradiance for solar energy applications and solar radiation models. *Solar energy*, 76(4):423–453, 2004.

[HMS05] Jörg Haber, Marcus Magnor, and Hans-Peter Seidel. Physically-based simulation of twilight phenomena. *ACM Transactions on Graphics*, 24(4):1353–1373, October 2005.

[Hul53] E. O. Hulburt. Explanation of the brightness and color of the sky, particularly the twilight sky. *Journal of the Optical Society of America*, 43(2):113–118, February 1953.

[HW12] Lukas Hosek and Alexander Wilkie. An analytic model for full spectral sky-dome radiance. *ACM Transactions on Graphics*, 31(4), July 2012.

[Jar08] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. Citeseer, 2008.

[KKN+14] Joseph T. Kider, Daniel Knowlton, Jeremy Newlin, Yining Karl Li, and Donald P. Greenberg. A framework for the experimental comparison of solar and skydome illumination. *ACM Transactions on Graphics*, 33(6), November 2014.

[Kla87] R. Victor Klassen. Modeling the effect of the atmosphere on light. *ACM Transactions on Graphics*, 6(3):215–237, July 1987.

[Kol12] R. Timothy Kol. *Analytical sky simulation*. Master Thesis, Utrecht University, Utrecht, Netherlands, 2012.

[Kut12] Peter Z Kutz. Physically-based atmosphere rendering. 2012.

[LF14] Diogo AR Lopes and António Ramires Fernandes. Atmospheric scattering - state of the art. 2014. Publisher: Instituto Politécnico de Leiria.

[Mie08] Gustav Mie. Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der Physik*, 330(3):377–445, 1908.

[NDKY96] Tomoyuki Nishita, Yoshinori Dobashi, Kazufumi Kaneda, and Hideo Yamashita. Display method of the sky color taking into account multiple scattering. In *Pacific Graphics*, volume 96, pages 117–132, 1996.

[NMN87] Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 303–310, New York, NY, USA, 1987. Association for Computing Machinery.

[NSTN93] Tomoyuki Nishita, Takao Sirai, Katsumi Tadamura, and Eihachiro Nakamae. Display of the earth taking into account atmospheric scattering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 175–182, 1993.

[O'N05] Sean O'Neil. Accurate atmospheric scattering. *GPUGems 2: Programming Techniques for High-Performance Graph-ics and General-Purpose Computation*, 2:253–268, 2005.

[Pen57] Rudolf Penndorf. Tables of the refractive index for standard air and the rayleigh scattering coefficient for the spectral region between 0.2 and 20.0 μ and their application to atmospheric optics. *Journal of the Optical Society of America*, 47(2):176–182, February 1957.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[PSM93] Richard Perez, Robert Seals, and Joseph Michalsky. All-weather model for sky luminance distribution—preliminary configuration and validation. *Solar energy*, 50(3):235–245, 1993. Publisher: Pergamon.

[PSS99] Arcot 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*, pages 91–100, 1999.

[Sek92] Seishi Sekine. Optical characteristics of turbid atmosphere. *J Illum Eng Int Jpn*, 71(6):333, 1992.

[Slo02] Jaroslav Sloup. A survey of the modelling and rendering of the earth's atmosphere. In *Proceedings of the 18th Spring Conference on Computer Graphics*, SCCG '02, page 141–150, New York, NY, USA, 2002. Association for Computing Machinery.

[Str71] John W Strutt. Xv. on the light from the sky, its polarization and colour. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(271):107–120, 1871.

[ZWP07] Georg Zotti, Alexander Wilkie, and Werner Purgathofer. A critical review of the preetham skylight model. *WSCG 2007 Short Communications Proceedings I*, pages 23–30, 2007. Publisher: Václav Skala-UNION Agency.