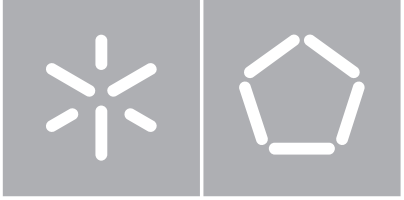
Universidade do Minho
Escola de Engenharia

Diogo Alberto Rocha Lopes

State of the Art on Atmospheric Scattering

Outubro de 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Diogo Alberto Rocha Lopes

State of the Art on Atmospheric Scattering

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor António José Borba Ramires Fernandes

Outubro de 2014

ACKNOWLEDGEMENTS

I take this opportunity to thank the people who have helped me during the creation of this thesis.

First I would like to thank my adviser Antonio Ramires Fernandes for his help and support throughout the project.

I would also like to thank my family, my brothers and sisters of LEI and also to my friends for their support.

ABSTRACT

The colour of the sky has always fascinated people throughout the ages. The blue sky, the colour of the sun, the orange of the sunsets, the clouds, etc...

For centuries, physicists and mathematicians tried to explain with formulae what artists like Leonardo da Vinci reproduced in their paintings, like the colour of the sky, the bluish colour of distant mountains, fog, and several other nature effects.

In the area of computer graphics there is a great interest in recreating these natural effects, observable everyday, as real as possible. The reproduction of these effects is essential for certain applications such as flight simulators, video-games and other scientific applications, hence, the relevance of rendering these effects in real time.

This project aims to present a state of the art on atmospheric scattering and so consolidate knowledge on the subject.

Much work has been done with the purpose of recreating these effects digitally. In result many algorithms, implemented using different techniques, are now available.

When evaluating atmospheric scattering there are several aspects that are considered such as the light source, the density and size of particles in the atmosphere, among others.

In this thesis, only daylight models will be considered and in these models, the sun is considered the only light source. The particles on the atmosphere can be divided in 2 main entities: air molecules and aerosol particles.

The phenomenon can be explained as the result of the interaction between the rays that come from the sun and the particles in the atmosphere. The most relevant interaction, regarding colour, is scattering.

The atmospheric scattering models present methods and techniques to evaluate and recreate this phenomenon. These models are very diverse but they base their techniques with the physics behind the phenomenon.

Their evolution showed a path that went from quality to efficiency. Part of this evolutionary trait came with the evolution on both hardware and software.

The early AS models had a greater concern in recreating with the most possible accuracy the atmospheric scattering phenomenon. That concern passed, over the years, to recreate this phenomenon the fastest way possible.

When talking about rendering the colours of the sky, is natural to want to render the other natural effects such as clouds and light shafts. Therefore it is also valuable to present information of some algorithms that can recreate these effects.

RESUMO

A cor do céu sempre fascinou as pessoas ao longo dos tempos. O azul do céu, a cor do sol, os pores-do-sol alaranjados, as nuvens, etc...

Durante séculos, físicos e matemáticos tentaram explicar com fórmulas o que artistas como Leonardo da Vinci reproduziram nas suas pinturas, como a cor do céu, a cor azulada de montanhas distantes, névoa, entre outros.

Na área de computação gráfica há um grande interesse em reproduzir com o maior realismo possivel estes efeitos naturais observados no quotidiano. A reprodução destes efeitos é fundamental para certas aplicações como simuladores de vôo, jogos e outro tipo de aplicações mais científicas, daí, a importância em reproduzir estes efeitos em tempo real.

Este projeto tem como objetivo apresentar um estado da arte em dispersão atmosférica e assim consolidar o conhecimento sobre o assunto.

Muito trabalho já foi feito com o propósito de recriar estes efeitos digitalmente. Para isso muitos algoritmos, implementados usando diferentes técnicas, foram apresentados.

Ao avaliar o fenómeno de dispersão atmosférica existem vários aspectos que são considerados, tais como a fonte de luz, a densidade e tamanho de partículas na atmosfera, entre outros.

Nesta tese, apenas os modelos diurnos serão considerados e nestes modelos, o sol é considerada a única fonte de luz. As particulas existentes na atmosfera podem ser divididas em dois tipos principais: moléculas de ar e aerosóis.

O fenómeno pode ser visto como o resultado da interacção entre a luz que vem do sol e as partículas na atmosfera. A interação mais relevante, em relação a cor, é a dispersão.

Os modelos de dispersão atmosférica apresentaram métodos e técnicas que conseguem avaliar e recriar este fenómeno.

Estes modelos são diferentes entre eles, mas todos têm como base a física por de trás do fenómeno.

A evolução destes modelos mostra um caminho que passou de maior qualidade para uma maior eficiência. Parte desta característica evolutiva veio com a evolução tanto do hardware como do software.

Os primeiros modelos, tinham como preocupação principla recriar o fenómeno de dispersão atmosférica com a maior precisão possível. Essa preocupação passou, ao longo dos anos, para recriar este fenómeno da maneira mais rápida possível.

Ao falar sobre como renderizar as cores do céu, é natural falar também sobre outros efeitos naturais tais como nuvens e feixes de luz. Por isso, também é importante apresentar alguns algoritmos que conseguem recriar estes efeitos.

CONTENTS

Contents

LIST OF FIGURES

List of Figures

List of Figures

LIST OF TABLES

LIST OF ALGORITHMS

1

INTRODUCTION

1.1 CONTEXTUALISATION

In the area of computer graphics there is a great interest in rendering the colours of the sky and recreate some of the natural phenomenons that can be seen everyday.

Atmospheric scattering is the natural phenomenon mainly responsible for the sky and sun colours, and it can be described as the change in the direction of light due to the interaction with the particles of the atmosphere.

The atmospheric scattering effects has always fascinated the computer graphics community. Any attempt to recreate the beauty of images like the ones present in `figure` 1 is praiseworthy.

Figure 1: Images of the sky during different times of day (sunrise, noon and sunset) - by Paul Bates in "http://paulbates.com"

Recreating the colours of the sky, sunsets and clouds is a real challenge because the nature of scattering is complex.

Over the years many algorithms were presented to reproduce this phenomenon. The majority of these algorithms are based on the theoretical foundations of the physicists Lord Rayleigh (Rayleigh (1871a,b, 1881, 1889)) and Gustav Mie (Mie (1908)).

In this thesis, only daylight models will be considered, and the colour of the sky and sun are the main concern. In these models, the sun is considered as the only light source. Due to the distance between the Sun and Earth, it is assumed that the light rays/photons travel parallel to each other, and unmolested through the empty space until they enter the earth's atmosphere.

Upon entering the atmosphere these rays will interact with the particles in the atmosphere. When evaluating atmospheric scattering there are several aspects that are considered such as the light direction, the density and size of particles in the atmosphere, among others. Most models consider two types of particles: air molecules and aerosol particles, the latter being much larger than the former. Air molecules can be found everywhere in the atmosphere where aerosols are more common closer to earth's surface. The result of the interaction between a photon and a particle is a function of the photon's wavelength and the size of the particle.

These interactions are responsible for the colour of the sky and sun throughout the day, as well as the colour of other elements such as clouds and landscape.

1.2 MOTIVATION

The existing models of atmospheric scattering can produce high-quality results. However, the information about these models is too widespread, and there isn't a very consistent evaluation about the strong and weak points of each of these models. Also, there is not much information about getting these models to work, i.e., there is not enough information about the values used in the atmospheric scattering models that were used to obtain the results (images) present in the articles/proceedings.

Therefore its highly relevant to gather this information and make a thorough analysis of these algorithms and thus consolidate the knowledge on this subject.

This project tackles this issue attempting to provide a complete description of the algorithms, covering the mathematical side with formulae, as well as the programmers view with code snippets.

The main focus of this work will be the atmospheric scattering phenomena under the context of clear sky colour. For completeness a brief overview of other phenomena such as light shafts, clouds, rainbows and lightning will also be a part of this thesis.

1.3 OBJECTIVES

The first goal is to understand the mechanics behind atmospheric scattering, namely how is a ray affected by the interaction with a particle. This will be decisive to explain the colours of the sky and sun.

After the theoretical part is understood, the goal will be to study how computer graphics researchers have implemented this phenomenon. It is fundamental to understand what assumptions were made. As this is a highly complex phenomenon, most models tend to make assumptions that simplify some aspects of the theory to make it more performance friendly. Also, due to the complexity of the problem some researchers proposed the pre-computation of some part of the computations involved in an attempt to speed up their algorithms. These, and other, approaches will be detailed for each method.

The overall goal is to write a state of the art on the subject of sun and sky colour due to atmospheric scattering. For completeness, a brief overview of other atmospheric related phenomena will also be included.

1.4 DOCUMENT STRUCTURE

In `chapter` 2 there will be a description of the atmospheric scattering phenomenon based on the general mechanics of the atmospheric scattering models. Later, in `chapter` 3, a detailed description of of the most relevant models and algorithms will be presented as well as some results obtained by the implementation of some of those models. In `chapter` 4 it is presented content related to models that can reproduce a variety of other weather phenomenons: clouds, fog, rainbows, light shafts, rain, and lightning. Conclusions and considerations about the work done are presented in `chapter` 5

2

ATMOSPHERIC SCATTERING - THE THEORY

The general theory behind this phenomena will be introduced in this chapter. This includes the mathematical formulation of the problem and an understanding of the mechanics of the photon-particle interactions and their implication in the colours we perceive.

The first sections will detail how scattering affects the photons travelling direction, and ultimately the light intensity and colour that reaches our eyes.

The last section will describe how scattering affects the colour of the sky, the sun, and the landscape itself.

2.1 AN INTRODUCTION TO THE SCATTERING EFFECT

The sun is a source of light that sends a countless number of rays/photons, every second, towards the Earth. Upon entering the atmosphere, sun rays will interact with particles in the atmosphere. There are several types of interaction that can occur, but the most relevant one regarding colour is scattering.

Scattering occurs when a photon's electromagnetic field hits a particle's electric field in the atmosphere and is deflected into another direction (Rayleigh (1871a)).

To better illustrate the implications of the light scattering phenomenon, lets follow a ray's path since it enters the atmosphere until it reaches the camera (ray R_1 in `figure 2`). Lets assume that a ray is a set of photons travelling along a common trajectory.

Ray R_1 travels from the sun and reaches the top of the atmosphere (P_{sun1}) with a certain intensity I_0. Following the same direction we see that the ray interacts by the first time with a particle at point P_1.

This interaction has the potential to deflect the ray's photons in all directions. Hence, only part of the photons will maintain the initial direction, which results in the attenuation of the rays intensity along the original direction. Each time the ray interacts with a particle its intensity gets further attenuated.

Still considering the initial direction, R_1 further interacts with other particles until it reaches P_{scatt}. The intensity of the ray upon reaching P_{scatt}, I_1, can be computed with `equation 1`.

$$I_1 = I_0 att(p_A) \tag{1}$$

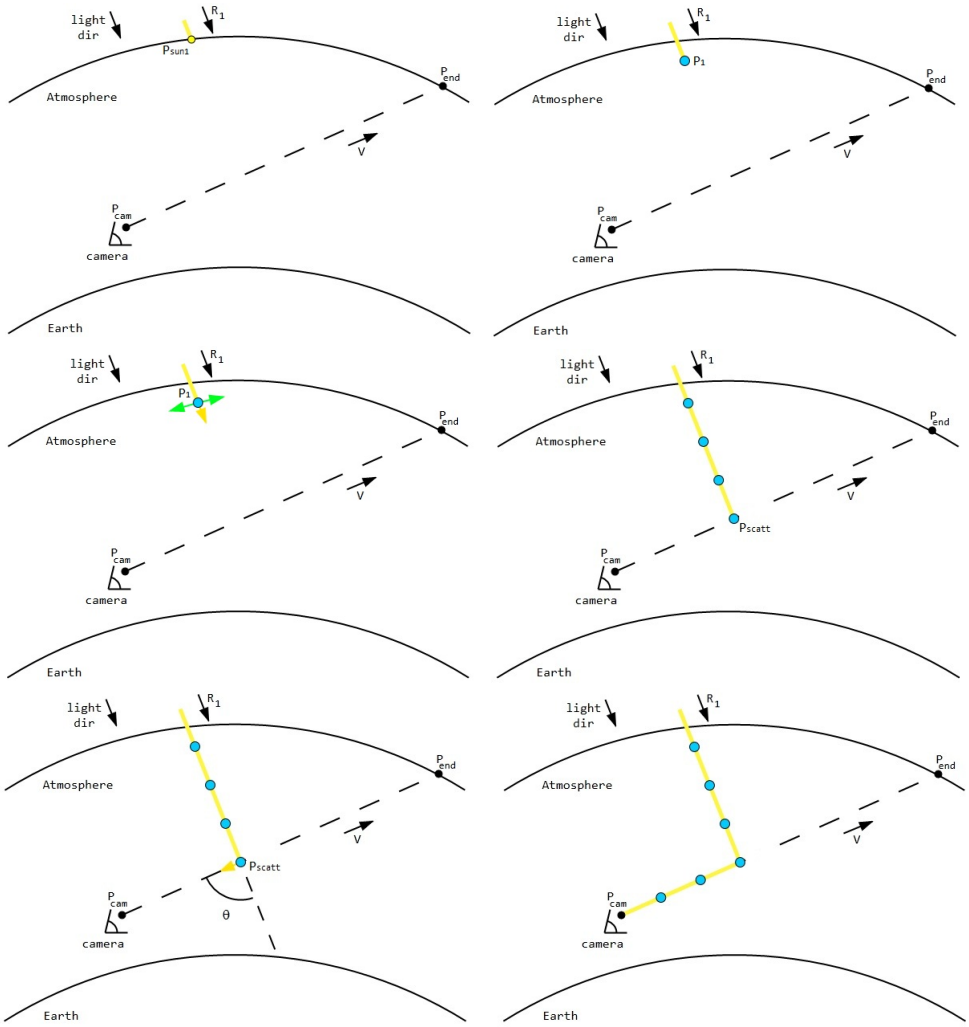


Figure 2: Path of a ray in the atmosphere

where p_A is the path from P_1 to P_{scatt}. In this case, the value of $att()$ represents the light attenuation, which can be translated as the percentage of photons that will follow the same direction after travelling a certain path in a medium, in this case the atmosphere.

Consider that at P_{scatt} some of the photons are scattered towards the camera. So a ray directed towards the camera will leave P_{scatt} with an intensity value of I_{scatt} (`equation 2`).

$$I_{scatt} = I_1 phase(\theta) \tag{2}$$

where $phase(\theta)$ determines the percentage of the photons that will travel in a direction with an angle θ (`figure 2`), relative to the original direction. The process of deflecting the light towards a given direction, in this case the direction towards the camera, is also called in-scattering.

Between P_{scatt} and the camera location, P_{start}, the ray will suffer further attenuation as it interacts with particles on the way, hence, the ray will reach the camera with an intensity I_{final} (equation 3).

$$I_{final} = I_{scatt} att(p_B) \tag{3}$$

where p_B is the path between P_{scatt} and the camera. The loss of intensity by the deflection of photons out of the original direction is also called out-scattering.

The total amount of intensity reaching the camera in a direction v can be computed as the sum of all the in-scattering contributions caused by rays, parallel to R_1 interacting with particles in the path from the camera to the exit point (P_{end}) of the atmosphere along v.

A representation of the in-scattering and out-scattering events can be seen in (figure 3).

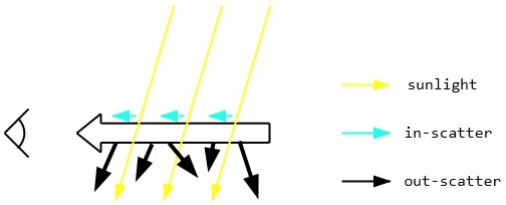


Figure 3: In-scattering and Out-scattering

Is important to mention that (as figure 3 shows) out-scattering has more influence than in-scattering. This happens due to the nature of these types of scattering. While in-scattering can be represented as a probability of a photon being deflected towards the camera through a specific solid angle, out-scattering can be represented as the probability of a photon being deflected in the remaining angles which represents an higher probability. Therefore an higher amount of photons are out-scattered than in-scattered. Furthermore, scattering depends on the photon's wavelength, hence not all colours scatter with the same distribution.

John Strutt, also known as Lord Rayleigh, studied the light scattering phenomenon and presented a series of equations capable of describing the behaviour of the interactions between light and small particles (Rayleigh (1871a,b, 1881, 1889)). Gustav Mie later presented a more general light scattering model that includes particles of all sizes (Mie (1908)).

The complete equation to compute the intensity arriving at the camera due to R_1, according to the diagram in figure 3 is:

$$I_{final} = I_0 \; att(p_A) \; phase(\theta) \; att(p_B) \tag{4}$$

The next sections will detail the scattering phenomena based on the work of Rayleigh (section 2.2) and Mie (section 2.3). Afterwards the process to compute the attenuation is presented (section 2.4). Once both scattering and attenuation are presented it is time to show how everything fits together

(section 2.5). To conclude, a brief discussion on the effects of these processes on the colours we perceive (section 2.6).

2.2 RAYLEIGH SCATTERING

According to the Oxford Dictionary, Rayleigh scattering is *The scattering of light by particles in a medium, without change in wavelength* (Oxford-Dictionaries (2014)). Rayleigh scattering describes the interaction between light and small sized particles (10 times inferior to the photon wavelength) like air molecules. To better understand the theory of Rayleigh lets consider a set of photons with different wavelengths coming from the sun.

When the photons enter the atmosphere, they will interact with the particles in the atmosphere and some of them will be deflected into different directions.

What Rayleigh realised is that photons with a certain wavelength (for example $475nm$) react differently when interacting with the particles of the atmosphere when comparing with photons with a wavelength of $670nm$, i.e., some photons have more potential of being scattered than others (figure 4).

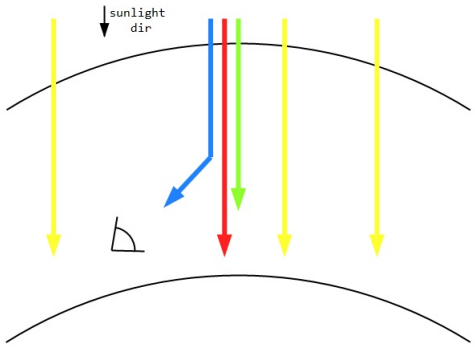


Figure 4: Rayleigh scattering light behaviour - scratchapixel.com

According to Rayleigh, the intensity of light scattered by an air molecule, in a given direction θ and for a given wavelength λ, can be described by the scattering cross section and the scattering phase function, see equation 5.

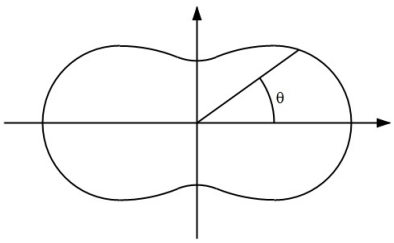
$$I(\lambda,\theta) = I_0\sigma_R(\lambda)\frac{phase_R(\theta)}{4\pi} \tag{5}$$

In equation 5, $phase_R(\theta)$ represents the Rayleigh scattering phase function for air molecules ("mol"), $\sigma_R(\lambda)$ represents the Rayleigh scattering cross section and I_0 represents the intensity prior to the interaction.

A phase function is a function of the deflection angle that describes the scattering distribution of light.

2.2. Rayleigh Scattering

Rayleigh's phase function determines the probability of light being scattered in a certain direction due to the the interaction with small sized particles (`equation` 6).



$$phase_R(\theta) = \frac{3}{4}(1 + cos^2(\theta)) \tag{6}$$

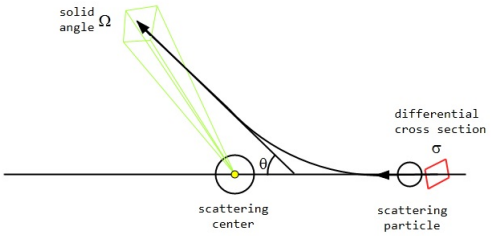
Figure 5: Rayleigh phase function

Clearly, the probability of a photon being scattered at an exact angle value is 0. Hence, we need the concept of cross section. In physics, according to Oxford Online Dictionaries, cross section is *a quantity having the dimensions of an area which expresses the probability of a given interaction between particles*.

In other words, scattering cross section is a hypothetical area which describes the likelihood of light being scattered by a particle, i.e., it explains the scattering potential of light into a solid angle due to its interaction with a particle.

Is important to mention that phase functions have the property of energy conservation, i.e., the integral over all directions must be equal to 1. Therefore, the phase function must be divided by the all range of directions, in this case, by all range of areas (since the scattering event occurs over a solid angle). So, the phase function must be divided by the all set of areas, in the case of a sphere, by 4π which is the surface area of a sphere (`equation` 5).

A visual example of the concept of cross section can be seen in `figure` 6 (Wikipedia (2014m)). On the right side of this figure there is the equation of Rayleigh's scattering cross section.



$$\sigma_R(\lambda) = \frac{24\pi^3(n_s^2 - 1)^2}{\lambda^4 N_s^2(n_s^2 + 2)^2} \tag{7}$$

Figure 6: Rayleigh scattering cross section

In `equation` 7, n_s is the refractive index of air, λ is the light photon's wavelength and N_s is the particle density of air molecules (normally, $n_s = 1.0003$ and $N_s = 2.54743 * 10^{19} cm^{-3}$).

Another important aspect to retain concerns the difference in scattering potential regarding light/photon's wavelength (figure 7).

In equation 7 Rayleigh states that the light scattering potential is inversely proportional to the fourth power of wavelength $(1/\lambda^4)$, which means that, rays with shorter wavelengths, for example blue ($475nm$), have far more scattering potential than rays with larger wavelengths like red ($650nm$).

In figure 7 there is a representation of the percentage scattering of direct sunlight according to light (photon's) wavelength.

This difference in light scattering potential is the main reason for the blue colour in the sky during most of the day. As blue wavelengths have one of the lowest frequencies, as well as violet, there will be more of these violet and blue photons reaching the eye. According to this, violet should be more predominant in the skies, however the colour is blue because the amount of violet is smaller than blue (table 2 in Preetham et al. (1999)) and the human eye reacts more to blue than to violet.

Further ahead (section 2.6.1), a more thoroughly explanation of the variations on the colours of the sky will be presented.

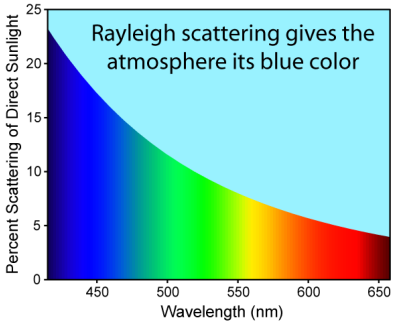


Figure 7: Colour intensity of Rayleigh Scattering - wikipedia

2.3 MIE SCATTERING

Mie scattering theory, also known as the Mie solution to Maxwell's equations, is a general theory applicable to scattering caused by particles of any size (Mie (1908)).

However, in scattering models Mie scattering is mainly used to explain the result of the interaction between light and larger particles, such as aerosols (larger than the ones considered in Rayleigh scattering).

In fact Rayleigh scattering is a subset of Mie scattering. The scattering behaviour of light when interacting with small and large particles can be seen in figure 8 (small particles on the left with Rayleigh scattering; center and right are larger particles with Mie scattering).

Mie's equations are highly complex because they require the calculation of infinite series and complex integrals (Van De Hulst (1982); Box (1983); Hahn (2009); C.P. (2012)). They are more complex and difficult to evaluate than the scattering equations of Rayleigh, therefore, and for matters of simplification, only some of Mie's equations will be analysed next.

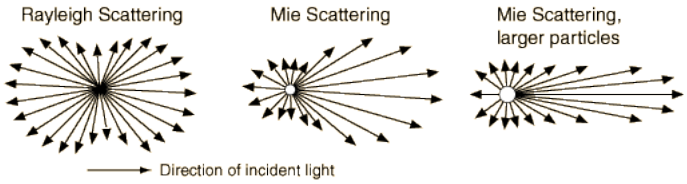


Figure 8: Types of scattering and description of its behaviour - hyperphysics.phy-astr.gsu.edu

Mie scattering, which can be applied to all sized particles, computes the intensity of scattering with `equation` 8.

$$I(\lambda,\theta) = I_0(\lambda)\sigma_M(\lambda)phase_M(\theta) \tag{8}$$

where σ_M is Mie's scattering cross section and $phase_M(\theta)$ is Mie's normalised phase function.

According to Mie's theory the scattering of an electromagnetic wave by particles is usually characterised by four scattering amplitudes S_1, S_2, S_3 and S_4 (Box (1983)). In the case of spherical particles (the subject of Mie theory), symmetry requires that $S_3 = S_4 = 0$. Therefore the scattering of light can be described resorting to S_1 and S_2:

$$S_1(\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)}[a_n\pi_n(cos\ \theta) + b_n\tau_n(cos\ \theta)] \tag{9}$$

$$S_2(\theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)}[b_n\pi_n(cos\ \theta) + a_n\tau_n(cos\ \theta)] \tag{10}$$

were $\pi_n(cos\ \theta)$ and $\tau_n(cos\ \theta)$ are angular functions and are expressed in terms of the Legendre polynomials (Wikipedia (2014g)), and a_n and b_n are complex parameters expressed in terms of spherical Bessel functions (Wikipedia (2014a)) evaluated at x (size of the particle) and n_s (refraction index).

These mentioned parameters (a_n and b_n) and the angular functions ($\pi_n(cos\ \theta)$ and $\tau_n(cos\ \theta)$) are not expanded here, but they can be found in (Box (1983); Hahn (2009)).

$S_1(\theta)$ and $S_2(\theta)$ describe not only the amplitude/intensity of light of a certain wavelength and according to the angle of scattering, but they also describe the behaviour of the interactions of light when interacting with particles with a certain size x. The way of computing these series depend on the number of elements/series used to compute the equations $S_1(\theta)$ and $S_2(\theta)$.

In turn this number of series depends mainly on one factor: particle size. If the particle is small, then the number of series necessary to describe the light-particle interaction behaviour will also be small. On the other hand, when considering larger particles, the number of series will be higher.

To evaluate the scattered intensity of light, these series are usually expressed via Stokes parameters, see `equation` 11 (Wikipedia (2014n)):

$$\begin{pmatrix} I^s \\ Q^s \\ U^s \\ V^s \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} & 0 & 0 \\ M_{21} & M_{22} & 0 & 0 \\ 0 & 0 & M_{33} & M_{34} \\ 0 & 0 & M_{43} & M_{44} \end{pmatrix} \begin{pmatrix} I_i \\ Q_i \\ U_i \\ V_i \end{pmatrix} \tag{11}$$

were I_i, Q_i, U_i and V_i represent the Stokes flux vector.

Regarding the scattering cross section and scattering phase function, from these parameters, the most important is M_{11} because it identifies the intensity of light that is scattered (without polarisation) and is given by:

$$M_{11}(\lambda,\theta) = (|S_1(\theta)|^2 + |S_2(\theta)|^2)\frac{\lambda^2}{4\pi^2} \tag{12}$$

The total scattering cross section is given by:

$$\sigma_M(\lambda) = 2\pi \int^{\pi} M_{11}(\lambda,\theta)\ sin\ \theta\ d\theta \tag{13}$$

According to Box (1983), M_{11} can also be seen as a combination of the phase function and the scattering cross section:

$$M_{11}(\lambda,\theta) = phase_M(\theta)\sigma_M(\lambda) \tag{14}$$

These equations show that in fact `equation` 8 could be given by $I(\lambda,\theta) = I_0(\lambda)M_{11}(\lambda,\theta)$.

Mie's scattering equations are complex and the solution depends on the difference between the particle size and the photons wavelength. Hence, Rayleigh scattering is commonly used for smaller particles. For larger particles, computational models usually resort to an approximation of Mie's equations described in `section` 2.3.1.

2.3.1 *Henyey-Greenstein Approximation*

Henyey and Greenstein devised an expression which mimics the angular dependence of light scattering by particles, used to describe scattering of light by interstellar dust clouds.

The Henyey-Greenstein scattering function has proven to be useful in approximating the angular scattering dependence of single scattering events. In other words, the Henyey-Greenstein phase function can also be used to evaluate the effect of the scattering events in the atmosphere.

In fact, its results are similar to the ones obtained with the Mie phase function, so, due to the high complexity of the Mie scattering functions, some applications use the Henyey-Greenstein method in order to enhance performance.

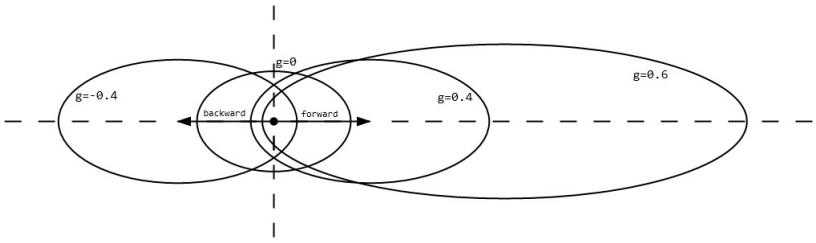


Figure 9: Henyey-Greenstein phase function

$$phase_{HG}(\theta) = \frac{1-g^2}{(1+g^2-2gcos(\theta))^{3/2}} \tag{15}$$

`Equation` 15 depends on a parameter g, ranging from -1 to 1, that represents the scattering orientation (`figure` 9). Positive values for g benefits forward-scattering, and negative values benefits back-scattering. If $g = 0$ its values (for light distribution) are similar to the ones obtained with Rayleigh scattering. When comparing this image with `figure` 8 we can see that, the values of g should be chosen according to the size of the particle, i.e., if the particle size is huge, then the value of g is closer to 1, if the particle size is smaller than, the value of g becomes closer to 0.

In `figure` 9 is a representation of the Henyey-Greenstein phase function when describing the interaction of light with different values of g.

Is important to mention that, as in Rayleigh's, the phase function used in the AS models could be slightly different. The differences were a result of the experiments and colour testing that according to their authors, presented a better approximation to reality. One example of this adaption can be found in Nishita et al. (1993) (function proposed in Cornette and Shanks (1992) - `equation` 16).

$$P(\theta) = \frac{3(1-g^2)}{2(2+g^2)} \frac{1-g^2}{(1+g^2-2gcos(\theta))^{3/2}} \tag{16}$$

2.4 COMPUTING LIGHT ATTENUATION

Computing the attenuation requires, among other things, knowing the particle density. In AS models, Earth's atmosphere is assumed to have air molecules and aerosols as particles and they are distributed all over the atmosphere. A possible representation can be found in `figure` 10.

In general, according to the works of Kaneda et al. (1991) and Nishita et al. (1993), the particle density for a particular height h can be computed as

$$\rho(h) = exp(-h/H_T) \tag{17}$$

where H_T is the scale height, i.e. the height at which the atmpsheric pressure decreases by a factor of e Wikipedia (2014l). For example, in Nishita et al. (1993), for air molecules $H_{mol} = 7994m$ and

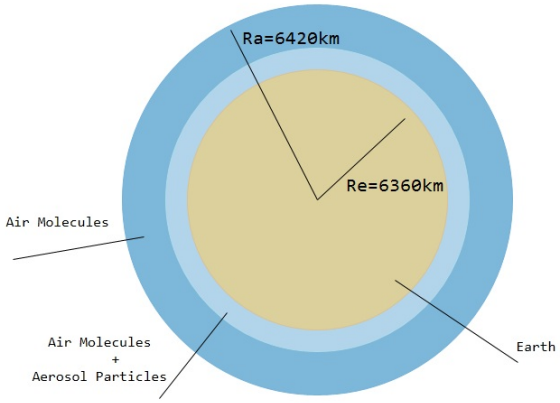


Figure 10: Earth's atmosphere density distribution - scratchapixel.com

for aerosol particles $H_{aero} = 1200m$. We shall refer to the density of air molecules as ρ_{mol} and the density of aerosols as ρ_{aero}.

This expression reflects the fact that both types of particles have an exponential distribution along the atmosphere being more dense near the surface and this expression also shows that the concentration of aerosols is only meaningful much closer to the surface.

The size of the particles also affects the interaction with different wavelengths. Attenuation is therefore a function of both the wavelength, λ, and particle density, ρ.

The Beer-Lambert's law (derivation of the works in Beer (1852) and Lambert (1760)) describes the extinction/attenuation of light due to out-scattering events. In other words, this law describes the portion of light transported through a certain medium, in this particular case, the atmosphere.

This law, considering its application in a path ($P_A \rightarrow P_B$), is usually represented as:

$$att(P_A \rightarrow P_B) = exp\Big(- \int_{P_A}^{P_B} \beta(x)dx \Big) \tag{18}$$

were β is the extinction coefficient, i.e., β represents the optical thickness of the atmosphere particles (air molecules and aerosol particles in this case).

We can see that according to `equation` 18 the value of attenuation varies exponentially with the distance travelled by the light in the atmosphere.

As been mentioned before, in the case of atmospheric scattering models, the value of light attenuation is the portion of photons that are out-scattered due to air molecules (Rayleigh scattering) and aerosol particles (Mie or Henyey-Greenstein scattering functions).

Using these scattering equations we can give values to the extinction coefficient in `equation` 18, because, those functions explains how much photons(light) are out-scattered from their direction due to interactions with a volume of particles in the atmosphere.

Equation 19 shows how to compute the attenuation (for particles of type T, where T can be *aero* (aerosol particles) or *mol* (air molecules)) of a ray after travelling in path s.

$$att_T(s,\lambda) = exp\Big(-4\pi\sigma_T(\lambda)\int^s \rho_T(p)dp\Big) \tag{19}$$

where $\sigma_T(\lambda)$ is the scattering cross section and $\int^s \rho_T(p)dp$ is the expression for optical depth. In some AS models the value for $\sigma_T(\lambda)$ is replaced by a notation $\frac{K}{\lambda}$ were this constant K is computed according to the type of particle being considered, for example, in O'Neil (2005), for air molecules $K_{mol} = 0.0030$, and in Nishita et al. (1993), K_{mol} is represented as a constant for the standard atmosphere (molecular density at sea level.) For aerosols, because the mie scattering cross section is complex to evaluate, some AS models only adjust the value of Rayleigh using the scale of particle distribution (in O'Neil (2005) for aerosol the value of K is half of the value for air molecules $K_{aero} = 0.0015$). The values used for $\sigma_T(\lambda)$ will be mentioned with more detail in the AS models in chapter 3).

When considering attenuation due to both types of particles, the value of attenuation of light when travelling a path s in a medium with both air molecules (*mol*) and aerosol particles (*aero*) becomes:

$$att(s,\lambda) = exp\Big(-(4\pi\sigma_{mol}(\lambda)\int^s \rho_{mol}(p)dp + 4\pi\sigma_{aero}(\lambda)\int^s \rho_{aero}(p)dp)\Big) \tag{20}$$

2.5 PUTTING IT ALL TOGETHER

Consider figure 11.

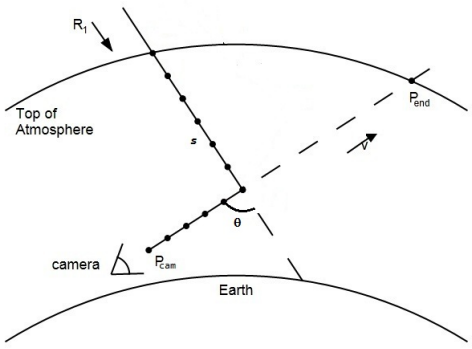


Figure 11: Evaluation of the intensity of a scattered ray along its path

The equation to compute the light intensity for a single ray, in a single wavelength λ that reaches the camera located at P_{cam}, along a path s, and for a given type of particle T (*aero* or *mol*) is given by:

$$I_T(\lambda, s, P_{cam}, P_{scatt}) = I_0(\lambda)att_{Tsun}(P_{scatt},\lambda)phase_T(\theta)att_{Tview}(P_{scatt},\lambda) \tag{21}$$

2.5. Putting it all Together

But the full amount of light that reaches the camera in a single wavelength λ that reaches the camera located at P_{cam}, along a path s, and for a given type of particle T (*aero* or *mol*) is given by:

$$I_T(\lambda, s, P_{cam}) = \int_{P_{cam}}^{P_{end}} I_0(\lambda) att_{Tsun}(p, \lambda) phase_T(\theta) att_{Tview}(p, \lambda) dp \tag{22}$$

where $att_{Tsun}(p)$ is the attenuation, for particles of type T, between the entrance in the atmosphere until point p in path s, and $att_{Tview}(p)$ is the attenuation between p and the camera position.

When considering the result from both types of particles (*aero* and *mol*), the final intensity can be computed as:

$$I(\lambda, s, P_{cam}) = \int_{P_{cam}}^{P_{end}} I_0(\lambda) att_{sun}(p, \lambda)(phase_{mol}(\theta) + phase_{aero}(\theta)) att_{view}(p, \lambda) dp \tag{23}$$

In `Equation` 23, the attenuation functions (att_{view} and att_{sun}) are the ones represented by `equation` 20, where $phase_{mol}$ is the phase function for air molecules (Rayleigh), $phase_{aero}$ is the phase function for aerosol particles (Mie or Henyey-Greenstein).

As we can see, in `equations` 23 and 20 there is a necessity to solve integrals. The main difference between the AS models resides in the way they solve these integrals. Some approximate the real value by performing ray marching (Wikipedia (2014k)) or they simplify these mathematical expressions by analytical functions.

So far, only single scattering events are being considered, i.e., ray R_1 suffered only one event of in-scattering towards the camera (`figure` 11). The full value of in-scattered light must also take into account the scattering of light from rays that have been scattered multiple times before. An example appears in `figure` 12.

Ray R_2 (unlike R_1) is scattered multiple times until it is finally scattered into the viewing direction at point P_{scatt}.

In mathematical terms, the main difference between R_1 and R_2 is in the way att_{sun} is computed.

While att_{sun} for R_1 represents the amount of particles that remained in the same direction, for R_2, att_{sun} has a constant change in paths (see `figure` 13), therefore the intensity of R_2 that reaches P_{scatt} can be seen as:

$$att_{R_2 sun}(\lambda) = I_0 * (att(P_{m1}) phase(\theta_{P_{m1}})) * (att(P_{m2}) phase(\theta_{P_{m2}})) \tag{24}$$

where $att(P_{m1})$ and $att(P_{m2})$ represent the attenuation of the rays intensity along the paths ($P_{m1} \rightarrow P_{m2}$) and ($P_{m2} \rightarrow P_{scatt}$) respectively, and where $phase(\theta_{P_{m1}})$ and $phase(\theta_{P_{m2}})$ represent the amount of light that was scattered into different directions (from point P_{m1} to P_{m2} and from point P_{m2} to P_{scatt} respectively).

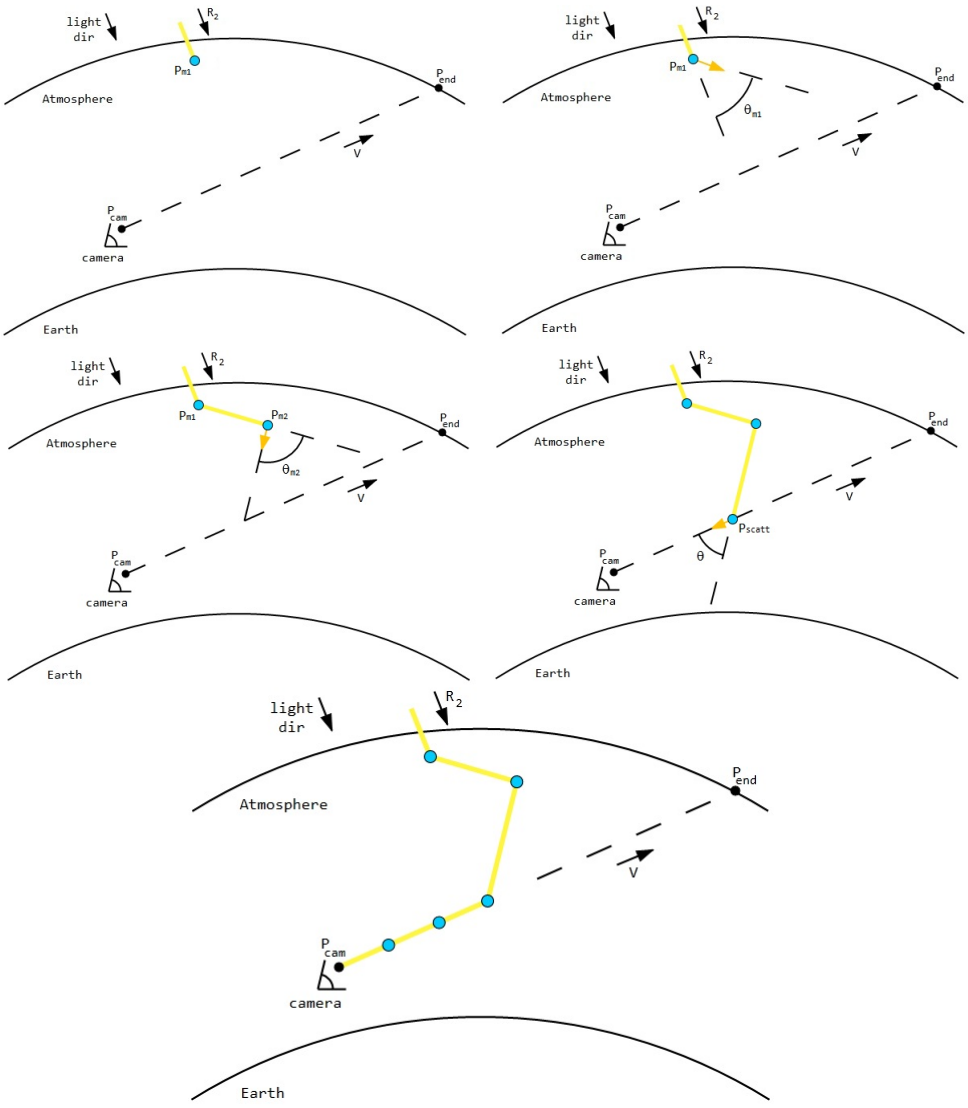


Figure 12: Path of a ray in the atmosphere (multiple scattering)

Hence, to take into account multiple scattering up to a n order of scattering events, (being $n = 1$ for single scattering, and $n > 1$ for the other orders) the term att_{sun}, from `equation` 22, needs to be replaced by a more complex expression (att_{sunMS}) that can sum the contributions from all these rays, which can be seen in `equation` 25.

$$att_{sunMS}(n,\lambda) = \sum_{i=1}^{n} \prod_{j=1}^{j=i} \sum_{k=1}^{k=n_k} att(path(j),\lambda) phase(\theta_j) \tag{25}$$

where $path(j)$ is the path travelled by the ray between two consecutive interactions, $j-1$ and j, $phase(\theta_j)$ is the angle between the direction of the ray prior to interaction j and the direction after interaction j, and n_k is the number of ray samples from which the light contributions are gathered,

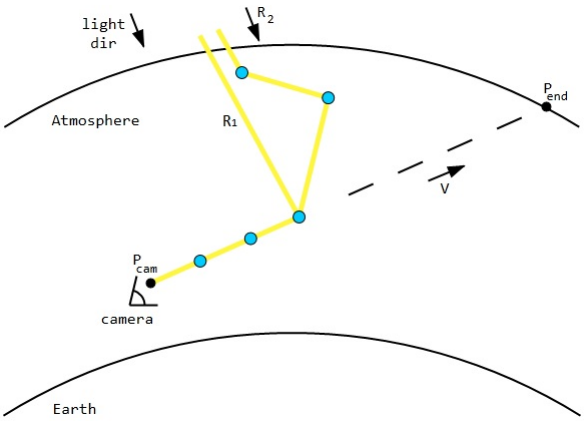


Figure 13: Representation of 2 rays, one scattered once (R_1) and another scattered 3 times (R_2)

i.e., unlike single scattering, where for one sample in the view direction we only consider a single ray with a certain intensity, in multiple scattering more rays are considered which implies the usage of sampling.

It is important to mention that, when evaluating the contribution from multiple scattering, all rays reaching the atmosphere must be taken into account, as well as all scattering directions in every light-particle interaction.

Evaluating the amount of scattered light ((R_1 - `figure` 13)) is already complicated and costly just considering single-scattering. With multiple scattering ((R_2 - `figure` 13)) the complexity becomes prohibitive from a computational point of view.

Since the amount of light coming from multiple scattering events is far less significant when compared to the contribution of single scattering, due to the successive application of the phase functions, some models evaluate only single scattering. Other models proposed to consider only a small number of scattering events.

The usage of these phase functions is crutial to evaluate the amount of light scattered in the atmosphere. For a better understanding on the comparison in using Rayleigh to evaluate the scattering of light when interacting with small particles and Mie to evaluate the scattering of light when interacting with large particles lets consider `figure` 14

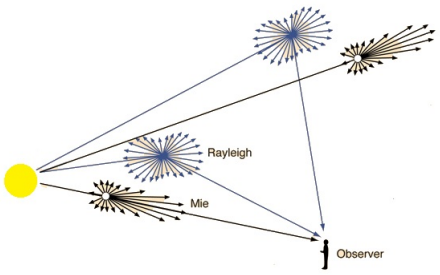


Figure 14: Mie and Rayleigh scattering comparison

This image shows the difference in scattering potential between small and large particles. In fact the image shows how the Mie function contributes more for the colour of the sun, than to the colour of the sky, where Rayleigh contributes more for the colours of the sky.

2.5.1 *Sun Direction*

There is an important factor that was not yet approached but that is crucial for any AS model: the direction of the sun.

Due to the huge different in size between the sun and earth (sun approximately 109 times larger in diameter than earth - Wikipedia (2014o)), most AS models consider the sun component as a simple direction of light.

There are a lot of algorithms that make possible to determine the position of the sun at any time of the day and at any position on the Earth based on coordinates of latitude and longitude.

In here is presented an algorithm from Preetham et al. (1999) that provides the sun light direction according to geographical coordinates, latitude and longitude, for a given time of day and date. With this algorithm is possible to obtain the direction of sunlight from any place at any time (image 15).

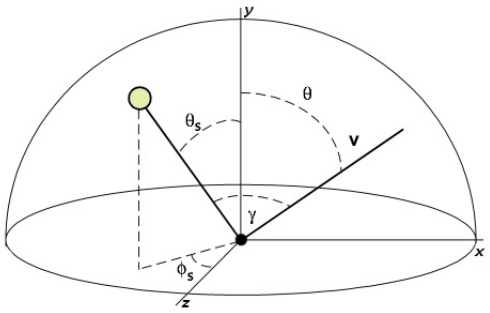


Figure 15: Preetham sky model - Preetham et al. (1999)

The algorithm considers the following input data:

- the time of day in decimal hours
- the day number (1... 365)
- latitude and longitude in radians
- the standard meridian for the time zone in radians.

The standard meridian time can be computed as the hour difference to Greenwich Meridian Time times 15 (an hour equals 15 radians).

Given a latitude (lat) and longitude (lon), the standard meridian for the time zone (Z), a time of day (ts) and a day (D), the solar time in decimal hours, t, can be computed as:

$$t = t_s + 0.170sin(\frac{4\pi(D-80)}{373}) - 0.129sin(\frac{2\pi(D-8)}{355}) + \frac{12(Z-lon)}{\pi} \quad (26)$$

The solar declination in radians, γ, can be approximated by

$$\gamma = 0.4093 sin(\frac{2\pi(D-81)}{368}) \tag{27}$$

Finally the sun position is given by (θ_s, ϕ_s), where θ_s is given by

$$\theta_s = \frac{\pi}{2} - arcsin(sin(lat)sin(\gamma) - cos(lat)cos(\gamma)cos(\frac{\pi t}{12})) \tag{28}$$

and ϕ_s is given by

$$\phi_s = arctan(\frac{-cos(\gamma)sin(\frac{\pi t}{12})}{cos(lat)sin(\gamma) - sin(lat)cos(\gamma)cos(\frac{\pi t}{12})}) \tag{29}$$

The direction of the sun is given in spherical coordinates, so, to obtain the vector of light direction its only necessary to do the following:

- $x = -sin(\theta_S)sin(\Phi_S)$
- $y = cos(\theta_S)$
- $z = sin(\theta_S)cos(\Phi_S)$

2.6 THE COLOUR OF THINGS

Scattering affects the perceived colour from the sky, the sun and the landscape itself. This section describes the reasons for these colours.

2.6.1 *The Sky and Sun*

Lets focus in a particular spot in the sky. During the day the sky is generally blue (with clear sky weather), and at sunset, near the horizon, the sky becomes reddish.

As mentioned before, according to Rayleigh, photons with shorter wavelengths (blue photons) possess more potential of being scattered when compared to photons with larger wavelengths (red photons).

The particle size plays an important role, affecting the dispersion of light. Small particles scatter light almost equally in all directions while large particles scatter more light in directions similar to the original direction of the ray (`figure 8`).

The distribution of particles in the atmosphere is also a relevant factor. The density increases with the approximation to the Earth's surface. So, the concentration of particles is larger near the surface than at higher altitudes, particularly for particles of larger dimension like aerosols.

Another important factor remains in the direction of light. Sunlight direction changes during the day. This direction affects the amount of blue, red and green photons that our eyes receive. `Figure 16` illustrates this statement.

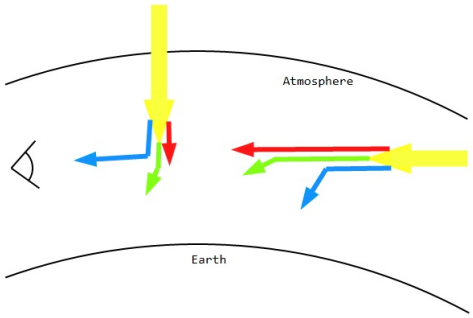


Figure 16: Path of different photons in the atmosphere at different times of day - scratchapixel.com

Lets take as an example sunlight direction at noon. If we look towards the sun, the colour our eyes perceive is closer to yellow, however if we look to any other place in the sky the colour is blue. That happens because, blue photons are more likely to scatter than green and red (`figure 16`), and by the same reason, if we look at the sun we will receive more green and red photons (because they are less likely deflected) and less blue photons that when looking towards the sky.

At sunset the light travels a greater path in the atmosphere, hence, not only blue but also green photons are significantly out-scattered, causing an orange sun. Also, at sunset the sunlight direction is closer to our normal view direction (we normally look towards the horizon) than at noon and that favours the red photons that have less scattering potential.

To summarise, the colour of the sky and sun depends on several factors such as particle size and light direction, but the major factor is the wavelength of the photons.

2.6.2 *The Landscape and other Objects*

As mentioned before, the only light source considered in an atmospheric scattering model is the Sun.

So the colour of an object will depend mainly on the intensity and direction of the sun light. The major contribution for the colour of the object is the reflection of the sunlight from the object.

When looking towards the landscape, normally, distant entities (mountains for example) seems bluer than closer ones. In fact this is a detail that helps us to perceive the relative distance between objects, and it happens due to in-scattering and attenuation. The more distant an object is from the eye, the more (blue) light will be scattered towards the eye, and at the same time, more of the object's "light" will be out-scattered from the viewers direction. This effect is usually called aerial perspective.

The bluish colour happens for the same reason that the sky is blue during the day. Photons with low wavelengths, like blue, have more scattering potential, so more blue light will be scattered in the view direction towards the eye.

Figure 17 Shows the perceived colour of an object lit by the sun as a result of the reflected light that travels from the object to the eye.

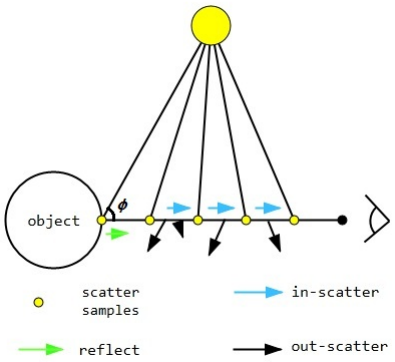


Figure 17: Reflection of a sun ray from an objects surface

The light reaching the eye is a result of the attenuation on the colour reflected by the object Obj plus, the in-scattered light along the path between a camera at P_{cam} and the position of the object P_{end}.

In mathematical terms, the amount of light that reaches the eye in the path between the object and the camera (considering only single scattering) is given in equation 30.

$$I_{obj}(Obj, P_{cam}, P_{end}) = I_{obj}(Obj, \phi) att_{view}(P_{cam} \rightarrow P_{end}) + I(\lambda, s, P_{cam}) \tag{30}$$

were $I_{obj}(Obj, \phi)$ is the intensity that is reflected at a angle ϕ (equation 31) and $I(\lambda, s, P_{cam})$ is the amount of in-scattered light in path s (equation 22).

The reflected intensity also suffers attenuation, hence, the value of $I_{obj}(Obj, \phi)$ (considering only single scattering) can be described as:

$$I_{obj}(Obj, \phi) = I_0(\lambda) att_{sun}(path_{sun-obj}) colour(Obj) reflect(\phi) \tag{31}$$

were $att_{sun}(path_{sun-obj})$ represents the attenuation in the path between the top of the atmosphere and the object's surface (and is given by equation 20), $colour(Obj)$ is the colour property of the object and $reflect(\phi)$ is the amount of light that is reflected in an angle ϕ according to the object properties.

The equations above justify the fact that distant objects/nature elements, such as mountains, have a more bluish/attenuated colour than closer objects.

3

ATMOSPHERIC SCATTERING - THE MODELS

In Atmospheric Scattering rendering there are two main approaches:

- Physically based Model - Usage of methods based on ray marching mechanisms to solve complex integrals using sampling
- Analytical Model - simplification of complex integrals and evaluating methods based on physical values

The physically based models usually present more accurate results when compared to the analytical model, however, its performance is far worse.

The first rendering models like the models of Klassen (Klassen (1987)) are physically based models.

They perform ray marching to obtain samples and evaluate the amount of light reaching the camera. In mathematical terms, this means that `equation` 23 becomes:

$$I(\lambda, s, n_{samples}) = \sum_{p=0}^{n_{samples}} I_0(\lambda) att_{sun}(p)(phase_{mol}(\theta) + phase_{aero}(\theta)) att_{view}(p) \tag{32}$$

Dobashi et al. (Dobashi et al. (1997)) presented models of an analytical nature.

Most of the models use Henyey-Greenstein phase function (or the version proposed by Cornette and Shanks) instead of Mie's since it gives a reliable approximation to the Mie's phase and avoids the high complexity of Mie's equations. In fact, of the models analysed here, only the model in Preetham et al. (1999) uses Mie's equations (analytical approximation with pre-computed values).

The most complete physically based models will evaluate multiple scattering events along the ray's path in order to obtain more rigorous values of light/colour intensity, resulting in more calculations and, therefore, more rendering time (Nishita et al. (1996a) and Bruneton and Neyret (2008)).

On the other hand, the analytical models prefer to simplify the integrals and use variables that module the intensity variation (for example Preetham et al. Preetham et al. (1999)). These methods obtain results faster at the expense of accuracy.

There are other concerns regarding the process of rendering the colours of the sky, namely spectrum of values and tone mapping.

Some AS models evaluate the amount of in-scattered light by evaluating up to 30 different light/photon wavelengths (e.g. aerial perspective model in Preetham et al. (1999)). This produces a colour spec-

trum far richer than RGB, but it does not allow direct representation of the colours on RGB model displays.

In order to represent these values in a computer, this spectrum of values must be converted. For example, in Preetham et al. (1999), the colour spectrum is converted to XYZ and then to RGB. The result obtained with the colour spectrum is physically more accurate than the models that just compute the colours of the sky for only 3 values of light wavelength.

Another issue is the values of light intensity. Sometimes those values overflow causing the resultant colour to be too bright. To address this issue some models propose to use tone maps in order to fit the values of light into valid range.

In the models, for a matter of simplicity, lets assume the concept of ray (or sun ray) as a group of photons. Sometimes a ray will be described as a group of photons of different wavelengths and other times as a group of photons of the same wavelength. The first concept is used to explain a general idea, and the second is used for computational matters.

3.1 ATMOSPHERE WITH TWO LAYERS

In 1987, Victor Klassen (Klassen (1987)) proposed a single scattering model of the atmosphere composed by 2 layers, each with constant density (`figure` 18).

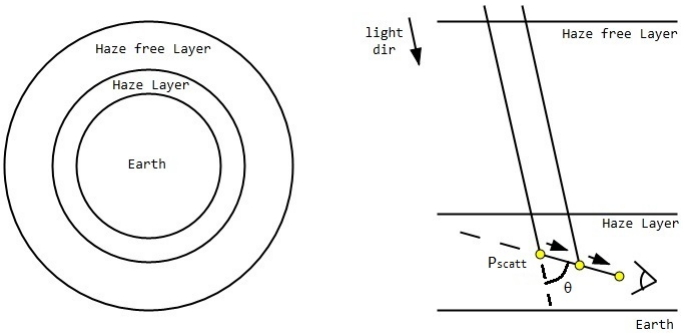


Figure 18: Earth's atmosphere with 2 layers, outer layer with air molecules, and the inner layer with air molecules and haze particles

Furthermore, the top layer is composed only by air molecules (haze free layer). The bottom layer, closest to the Earth's surface, is composed by both particle types (haze and air molecules). Klassen's model considers only single scattering and further assumes that it is very unlikely that an in-scattering event will occur in the top layer.

Consider the mechanism of light/photon scattering as described in `section` 2.1 and `figure` 18. The ray coming from the sun enters the atmosphere with an intensity I_0. Since in-scattering events occur only in the haze layer, in the top layer the attenuation is proportional to the distance covered by

the rays in that layer. When entering the haze layer the intensity of the ray (I_1) is given by `equation` 33.

$$I_1 = att(s_t, D_t)I_0 \tag{33}$$

where s_t is the path covered by the ray since it enters the atmosphere until it reaches the haze layer, and D_t is the density of the top layer.

The intensity at the in-scattering point P_{scatt} can be computed as the product of I_1 and the attenuation between the entrance point on the haze layer, and the intersection with the view direction. So, the intensity of the ray at that point $I_{P_{scatt}}$ is given by `equation` 34.

$$I_{P_{scatt}} = att(s_h, D_h)I_1 \tag{34}$$

where s_h is the path covered by the ray in the haze layer until it intersects the view direction, and D_h is the density of the haze layer.

At P_{scatt} part of the ray is deflected towards the camera. The ray leaves P_{scatt} with an intensity of I_2 that is given by:

$$I_2 = phase(\theta)I_{P_{scatt}} \tag{35}$$

where $phase(\theta)$ represents the phase function, and θ is the deflection angle. From P_{scatt} to the camera the light travels through path s_e. The ray's intensity when it reaches the camera is I_{cam} and is given by:

$$I_{cam} = att(s_e, D_h))I_2 \tag{36}$$

Putting it all together, considering this example the intensity would be given by:

$$I_{cam} = att(s_e, D_h)phase(\theta)att(s_h, D_h)att(s_t, D_t)I_0 \tag{37}$$

The full amount of light in the view direction is obtained by sampling the view direction (marching a ray), between the camera and the exit point of the haze layer, and computing at each sample the in-scattering contribution. The equation, that evaluates the full amount of light reaching the camera, is:

$$I_{final} = \sum_{i=0}^{n} att(s_{ei}, D_h)phase(\theta)att(s_{hi}, D_h)att(s_{ti}, D_t)I_0 \tag{38}$$

where s_{ei} is the path between the camera and sample i, θ is the deflection angle between the view direction and the sun rays, s_{hi} is the path of the sun ray inside the haze layer, and s_{ti} is the path of the sun ray inside the haze free layer.

3.1. Atmosphere with two layers

The attenuation itself is greatly simplified due to the assumptions of constant density within each layer. For the top layer the attenuation can be given by equation 39 and for the bottom layer by equation 40.

$$att_{top}(s, D_{top}) = exp(-4\pi\sigma_{mol}(\lambda))length(s)D_{top}) \tag{39}$$

$$att_{haze}(s, D_{haze}) = exp(-(4\pi\sigma_{mol}(\lambda) + 4\pi\sigma_{aero}(\lambda)))length(s)D_{haze}) \tag{40}$$

For the phase function, as in the bottom layer there are both aerosols and air molecules, the expression of $phase(\theta)$ in equation 38 can be written as:

$$phase(\theta) = phase_R(\theta) + phase_{HG}(\theta) \tag{41}$$

Using these equations, a CPU ray-marching solution was implemented in order to render the colours of the sky using Klassen's AS model. The results can be seen in figure 19.

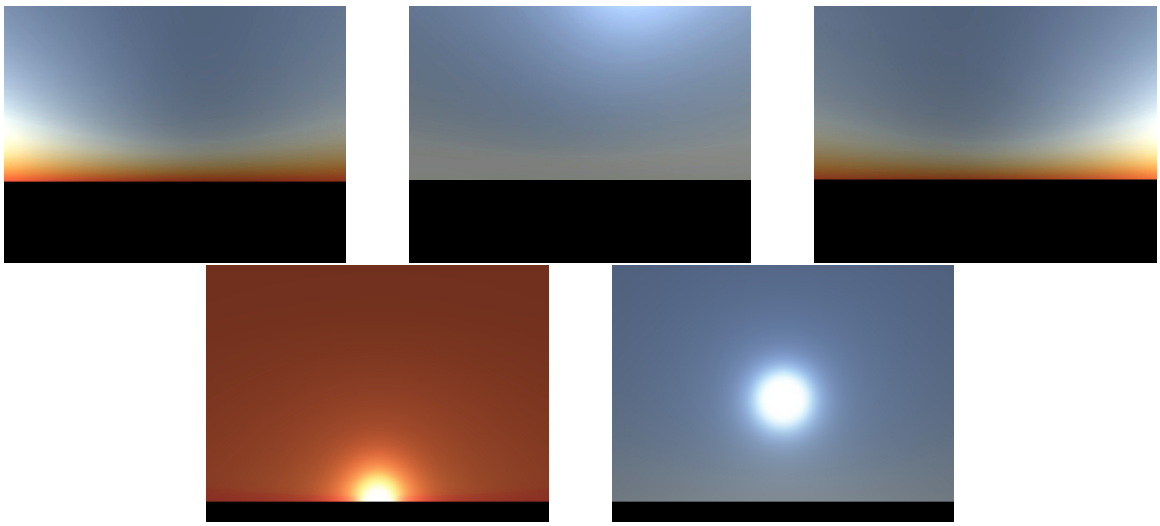


Figure 19: Images using Klassen model of the atmosphere with 2 layers. Top: Sky at Sunrise, Midday and Sunset, respectively. Bot: Sun at sunset and 16 hours

Klassen's model can reproduce various types of day, like foggy days or days with clear sky by adjusting the particle density and the height of the haze layer.

The results (see figure 19) were obtained by using the scattering coefficient values present in Nishita et al. (1993) and Bucholt (1995), namely $D_{haze} = 4$, $D_{top} = 2$, $\sigma_{mol} = \frac{K_{mol}}{\lambda^4}$ and $\sigma_{aero} = K_{aero} = K_{mol}/(10^{-25})$ ($K_{mol} = 4.5864 * 10^{-32}$). Eight samples were taken in each view direction, and only three wavelengths were considered for red, green and blue using for each one of them a

initial value of intensity of $I_0 = 20$. The g coefficient of Henyey-Greenstein phase function used was $g = 0.99$.

Algorithm 1 Klassen AS model

```
for each view direction view view_V do
  sample_V = (intersectSphere(camPos, view_V, hazeRadius) − camera_P)/numSamples;
  sample_P = camera_P + sample_V ∗ 0.5;
  sampleLength = length(sample_V);
  viewOffset = 0.5 ∗ sampleLength;
  θ = getAngle(view_V, sunDir_V);
  color = {0, 0, 0};
  for n = 0..numSamples do
    haze_P = intersectSphere(sample_P, sunDir_V, hazeRadius);
    topAtm_P = intersectSphere(sample_P, sunDir_V, atmRadius);
    hazeDist = length(sample_P, haze_P);
    hazeFreeDist = length(haze_P, topAtm_P);
    for n = 0..3 do
      color[i]+ = att(viewOffset, D_h, wavelength[i])∗        // equation 40
                  phase(θ)∗                                   // equation 41
                  att(hazeDist, D_h, wavelength[i])∗          // equation 40
                  att(hazeFreeDist, D_t, wavelength[i]);      // equation 39
    end for
    viewOffset+ = sampleLength;
    sample_p+ = sample_v;
  end for
  render(viewV, color);
end for
```

The rendering process to obtain the images with Klassen's AS model is present in `code` 1, where $intersectSphere(pos, dir, radius))$ represents the point of intersection of a ray, starting from point Pos and with direction dir, with a sphere of radius $radius$. In `code` 1 $camera_P$ represents the camera position relative to the center of the earth, $hazeRadius$ the radius of the sphere that delimits the haze layer, $atmRadius$ is the total radius of the atmosphere, $sunDir$ is the direction pointing towards the sun, and $wavelength$ is an array with the three wavelengths for RGB in meters.

On the same paper Klassen (1987) presents a model of fog, where fog is represented by another layer very close to the Earth's surface. In this model fog possesses an opacity coefficient and its used to evaluated also the colour of objects. This will be detailed in `section` 4.3.

3.2 ATMOSPHERE WITH EXPONENTIALLY DECREASING DENSITY

As mentioned before the atmosphere is not made of two distinct zones with constant density. In fact the atmosphere is a single layer where the particle density has an exponential variation. The next AS models become physically more accurate by considering this exponential variation.

3.2. Atmosphere with Exponentially Decreasing Density

The exponential variation of the atmoshpere density was first explored in Kaneda et al. (1991), where it implements single scattering with a continuous atmosphere as described in chapter 2.1. This work uses Rayleigh's scattering equations for air molecules and Henyey-Greenstein's equations for aerosols.

Optical depth integrals are approximated by marching a ray. While in Klassen's model the ray is only sampled from the view direction (v), this model extends the sampling to the full path of the ray since it enters the atmosphere just like in figure 20 .

With a non-uniform atmosphere density, this model needs to evaluate the density of air molecules and haze particles for each of these samples resulting in a closer approximation to the attenuation equation (equation 19).

A comparison between the models of Klassen and Kaneda et al. can be seen in figure 20.

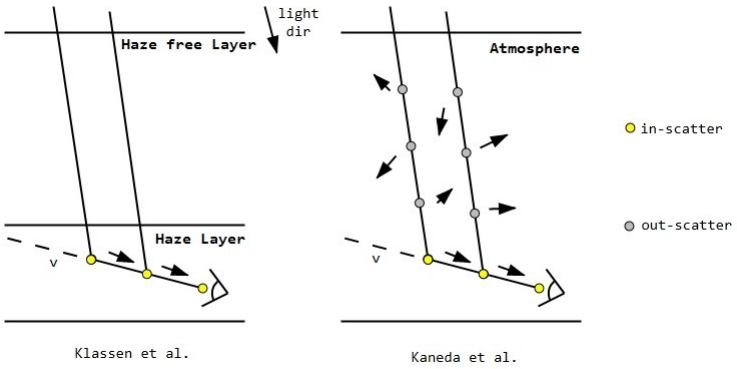


Figure 20: Comparison of Klassen's AS model with 2 layers and Kaneda et al. with one layer with heterogeneous density

The increase in accuracy comes with a penalty regarding performance, as the model by Kaneda et al. is significantly more expensive computationally wise. In Nishita et al. (1993) several performance wise improvements were proposed. The main objective was to reduce the number of calculations during rendering.

Consider a set of spheres as in figure 21 (left). According to Nishita et al., the best way to position these spheres across the atmosphere is in a decreasing exponential distribution, matching the density distribution of the atmosphere, such that the density amplitude between any two adjacent spheres is similar. The proposed formula to compute the radius of these spheres is given by equation 42.

$$r(i, n_{samples}) = r_{earth} + log_2(1 - \frac{i}{n_{samples}}) * H \tag{42}$$

where r_{earth} is the radius of the Earth and $H = atm_{radius} - earth_{radius}$.

Consider now a set of cylinders with the axis aligned with the sun direction and passing through the centre of the earth. Figure 21 (right) shows a cylinder matching this description.

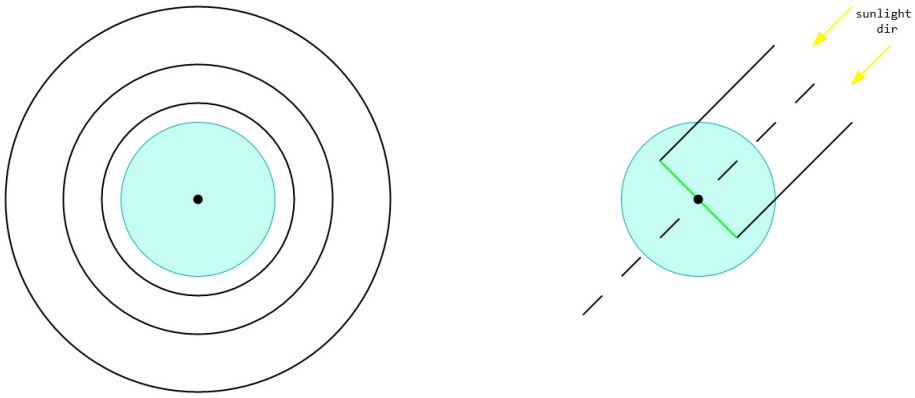


Figure 21: Discretization of the atmosphere

Assuming the sun rays travel parallel to each other, and considering `figure` 22, it is clear that sun rays entering the atmosphere have the same optical depth when reaching P_1 and P_2, since they cover the same portion of the atmosphere. This is true for the any point in the circle that is the result of the intersection of the cylinder with radius r_C and the sphere with radius r_S.

Nishita et al. propose to take advantage of this fact by pre computing the optical depths for pairs (r_C, r_S) and storing them in a lookup table. Considering the atmosphere radius (R_{atm}) and the earth radius (R_{earth}), $0 < r_C < R_{atm}$ and $R_{earth} < r_S < R_{atm}$.

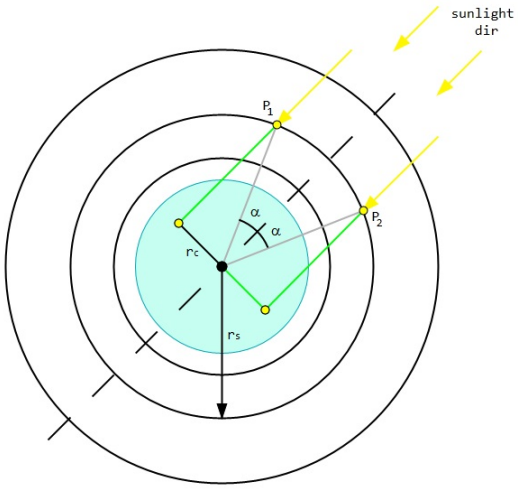


Figure 22: Symmetry property for optical depths

The lookup table would provide, through interpolation, the optical depth for all rays in the sun direction. Considering only single scattering, this would assist in the computation of the attenuation of the sun rays for every sample point along the view ray. To perform the interpolation is mandatory to identify at run-time the radius of the sample point and the radius of the cylinder. The radius is

obtained with the distance between the earth's centre and the sample point. For the cylinder radius, its used the angle relation seen in `figure` 22, where this angle is made between the sunlight direction, and the vector $centre_{earth} \rightarrow P_1$. So, the radius of the cylinder is $r_c = cos(\pi/2 - \alpha)$.

Recall the equation to compute the attenuation along a path (`equation` 43), and its usage to compute the intensity arriving at the camera (`equation` 44).

$$att(s,\lambda) = exp\Big(- (4\pi\sigma_{mol}(\lambda) \int^{s} \rho_{mol}(p)dp \; + \; 4\pi\sigma_{aero}(\lambda^4) \int^{s} \rho_{aero}(p)dp)\Big) \tag{43}$$

$$I(\lambda, s, P_{cam}) = \int_{P_{cam}}^{P_{end}} I_0(\lambda) att_{sun}(p)(phase_{mol}(\theta) + phase_{aero}(\theta)) att_{view}(p)dp \tag{44}$$

The proposed pre computation corresponds to the integrals in `equation`43, for att_{sun} in `equation` 44, which is a significant part of the required computations.

Results of using the model proposed by Nishita et al. can produce images as in figure 23:

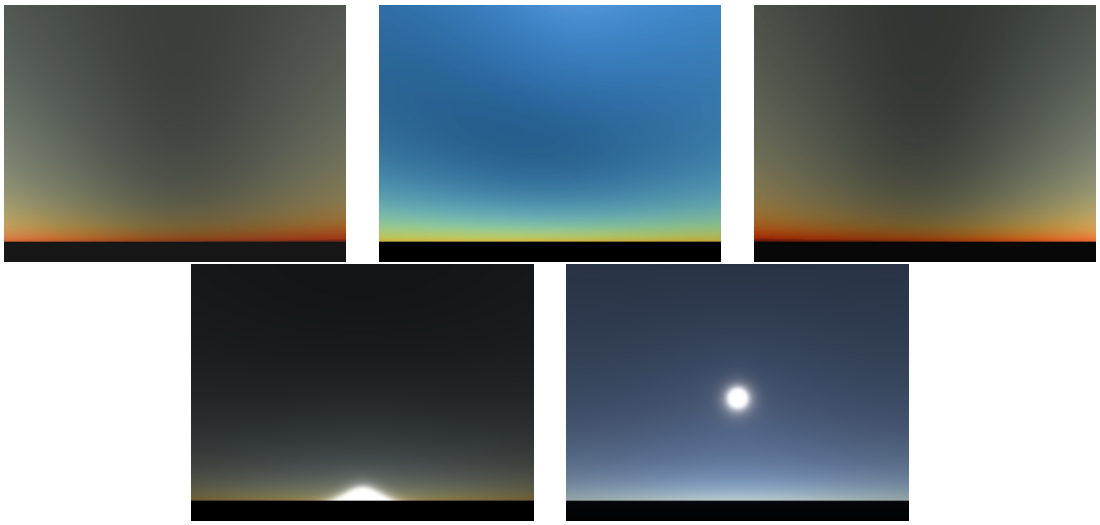


Figure 23: Images using Nishita model of the atmosphere in clear sky. Top: Sky at Sunrise, Midday and Sunset, respectively. Bot: Sun at sunset and 16 hours

These results were obtained by using coefficients like air and aerosol particles density, present in Nishita et al. (1993), Bucholt (1995) and Nishita et al. (1996a), which were mentioned above in `chapter` 2.1. The values for were the same used for the model of Klassen, namely, $\sigma_{mol} = \frac{K_{mol}}{\lambda^4}$ and $\sigma_{aero} = K_{aero} = K_{mol}/(10^{-25})$ ($K_{mol} = 4.5864 * 10^{-32}$). The photons wavelength (λ) was used in meters. Eight samples were taken in each viewing direction and sunlight direction, and only three wavelengths were considered: red, green and blue using for each one of them a initial value of intensity of $I_0 = 20$.

Nishita et al. (1993) AS model uses Rayleigh's phase function for air molecules ((equation 6), and for aerosol particles uses a modified version of the Henyey-Greenstein phase function as presented in equation 16. The g coefficient of used was $g = 0.99$.

In code 2 it is presented an algorithm to pre-compute the values of optical depth for a series of sampling spheres ($n_{spheres}$), sampling cylinders ($n_{cylinders}$) and for a number of samples in the process of marching the ray ($num_{Samples}$). $atmRadius$ is the total radius of the atmosphere and $sunDir$ is the direction pointing towards the sun.

Algorithm 2 Nishita et al. single scattering model - pre-computation

$initSphereRadius()$ // equation 42
$initCylinderRadius()$
for each pair in (r_s, r_c) **do**
  $start_P = getPointInIntersectionSphereCylinder(r_C, r_S)$
  $end_P = intersectSphere(start_P, sunDir_V, atmRadius)$
  $sample_V = (end_P - start_P)/num_{Samples}$
  $opticalDepth_{aero} = opticalDepth_{mol} = 0$
  $sample_P = start_P$
  $sampleLength = length(sample_V)$
  for $n = 0..numSamples$ **do**
    $opticalDepth_{aero} += exp(-height(sample_P)/H_{aero}) * sampleLength$ //$H_{aero} = 1200$
    $opticalDepth_{mol} += exp(-height(sample_P)/H_{mol}) * *sampleLength$ //$H_{mol} = 7994$
    $sample_P += sample_V$
  end for
  $lookup[r_C][r_S].aero = opticalDepth_{aero}$
  $lookup[r_C][r_S].mol = opticalDepth_{mol}$
end for

In code 3 it is presented an algorithm to compute the colours of the sky. In the code there is explicit the process of computing the attenuation value of the ray since it enters the atmosphere until it reaches the eye. There is also the part with the condition of the lookup table improvement proposed by Nishita et al. (lookup process). Like in code 2, atm_{Radius} is the total radius of the atmosphere and $sunDir$ is the direction pointing towards the sun.

The functions of $closest_{sphere}$ and $closest_{cylinder}$ will fetch the closest (and at the same time the lowest) entry in the lookup table for the radius of the sphere, and of the cylinders respectively.

Like Kaneda et al., Nishita et al. also presents a way to render clouds and extended to render also the colour of the sea. The mechanism of scattering of light and optical depth is use to evaluate the intensity of light that reaches the clouds and the sea, and in the case of clouds, the same mechanism for atmosphere particles is used but with different extinction coefficients. Regarding the sea, the calculation is based on the work of Gordon and McCluney Gordon (1973) McCluney and Center (1974). This will be further detailed in section 4.2 further ahead.

Algorithm 3 Nishita et al. single scattering model - run-time

for each view direction $view_V$ **do**
  $end_P = intersectSphere(camera_P, view_V, atmRadius)$
  $sample_V = (end_P - camera_P)/numSamples$
  $sample_P = camera_P + sample_V * 0.5$
  $sampleLength = length(sample_V)$
  $viewOffset = 0.5 * sampleLength$
  $\theta = getAngle(view_V, sunDir_V)$
  $opticalDepth_{aero} = opticalDepth_{mol} = 0$
  $zeros(attenuation)$
  for $n = 0..numSamples$ **do**
    $opticalDepth_{mol} = exp(h(sample_P)/H_{mol}) * viewOffset$
    $opticalDepth_{aero} = exp(h(sample_P)/H_{aero}) * viewOffset$
    $r_C = getCylinderRadius(sample_P)$
    $r_S = getSphereRadius(sample_P)$
    $lightOpticalDepth_{mol} = lookupAndInterpolate_{mol}(r_C, r_S)$
    $lightOpticalDepth_{aero} = lookupAndInterpolate_{aero}(r_C, r_S)$
    for $i = 0..3$ **do**
      $\beta_{mol} = (opticalDepth_{mol} + lightOpticalDepth_{mol}) \frac{4\pi.K_{mol}}{waveLength[i]^4}$
      $\beta_{aerosol} = (opticalDepth_{aero} + lightOpticalDepth_{aero}) 4\pi.K_{aero}$
      $attenuation[i] += exp(-(\beta_{mol} + \beta aero))$
    end for
    $sample_P += sample_V$
    $viewOffset += sampleLength$
  end for
  for $n = 0..3$ **do**
    $color[n] = I_0[waveLength[i]] * attenuation[n] * (phase_{mol}(\theta) + phase_{aero}(\theta))$
  end for
  $render(viewV, color)$
end for

3.3 MULTIPLE SCATTERING IN THE ATMOSPHERE

The AS models presented above only considered single scattering events. In 1996, Nishita et al. proposed a model to compute the colours of the sky by considering also the contribution of multiple scattering (Nishita et al. (1996a)) providing a better approach to the real phenomenon.

The inclusion of multiple scattering in the evaluation of light scattering implies the inclusion of more light samples to evaluate. The difference with these light samples is that instead of belonging to the path from a certain point v to the sun, they belong to the path from v to other points vs inside the atmosphere (`figure` 24).

For that purpose an AS model was proposed in which the atmosphere is divided in voxels, and they will be used to store pre computed values of light intensity (both direct and in-scattered light).

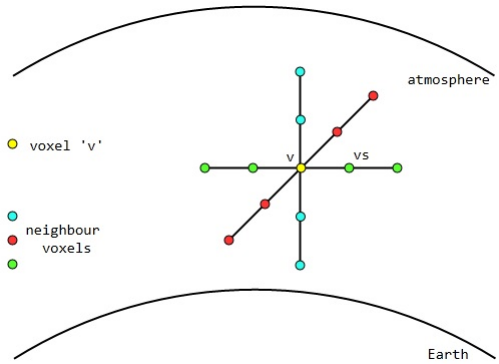


Figure 24: Nishita et al. atmosphere with voxels

The basic idea is to gather in each voxel v (`figure` 24) not only the direct light intensity arriving from the sun, but all light coming from pre defined sampling directions gathering light from neighbour voxels (vs). In order to gather light into voxels, Nishita et al. presents a multi-pass algorithm that accumulates different values of light at each pass. In turn, each pass represents an order of scattering, i.e., a two pass algorithm will gather direct light and a 1^{st} order of scattering (single scattering), a three pass will gather direct light, single scattering and 2^{nd} order of scattering, and so on (`equations` 22 and 25).

To gather direct light contribution for each voxel v, it is only necessary to compute the light attenuation of a sun ray with wavelength λ, travelling in a path s, starting at v and in the direction of the sun since the moment it enters the atmosphere until it reaches v ($I_{v,0} = I_0 * att_{sun}(s, \lambda)$).

To explain the process to gather the higher orders of scattering lets consider the gathering of single scattering in a voxel v (see `figure` 25).

To gather the 1^{st} order of scattering in voxel v, it is necessary to gather the values of light that are scattered at the sampling voxels vs (red circles in `figure` 25) on the various sampling directions ($sample_{dir}$), towards voxel v. Calculation is made using `equation` 21 using L_0 as the value of

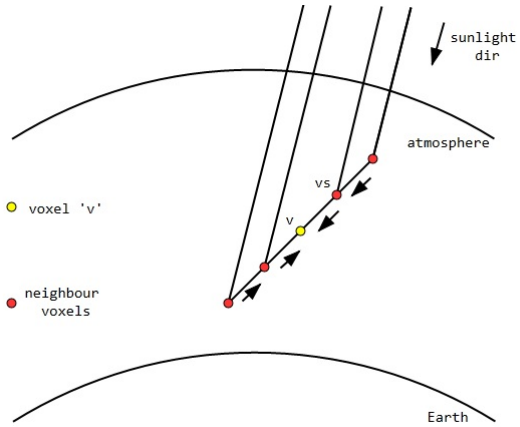


Figure 25: Voxels 1^{st} order of scattering calculation

direct light stored in vs ($I_{v,0}$) and the angle θ as the angle made between the sunlight direction and the sampling direction ($sample_{dir}$).

The value that reaches voxel v from each voxel vs is $I_{v,1,v_s} = I_{v_s,0} * phase(\theta) att_{view}(path(vs,v))$. So the amount of light in voxel v at this point is the sum of $I_{v,0} + I_{v,1,v_s}$. This process is repeated for a number of neighbours.

To gather a 2^{nd} order of scattering a similar process is done (see `figure` 26).

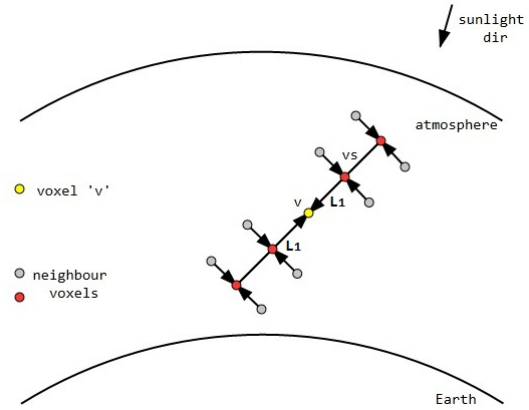


Figure 26: Voxels 2^{nd} (or n^{th}) order of scattering calculation

Just like in the 1^{st} order of scattering, the objective is to gather scattered light on the sample voxels v from the neighbour voxels vs. The difference is that the amount of light that goes from neighbour voxels vs to voxel v is given by $I_{v,2} = I_{v_s,1} * phase(\theta) att_{view}(path(vs,v),\lambda)$. This step is repeated n times.

3.3. Multiple scattering in the atmosphere

It is important to mention that at each step (i) the value of light ($I_{v,i}$) that is gathered from each neighbour voxel becomes considerably smaller due to the phase function, which means that the value of a 2^{nd} order of scattering is more significant than a 3^{rd} order of scattering.

Gathering light into voxels is performed as a pre computation. In run time the algorithm gathers these values for each sampling direction and applies a phase function to determine which portion will be deflected towards the camera. Note that the n orders of scattered light gathered in the voxels will become a $n+1$ order when considering the point of view from the camera, i.e., the direct contribution (order 0) will become the single scattering (order 1) for the camera, the single scattering for the voxel (order 1) will become the second order of scattering for the camera (order 2), and so on.

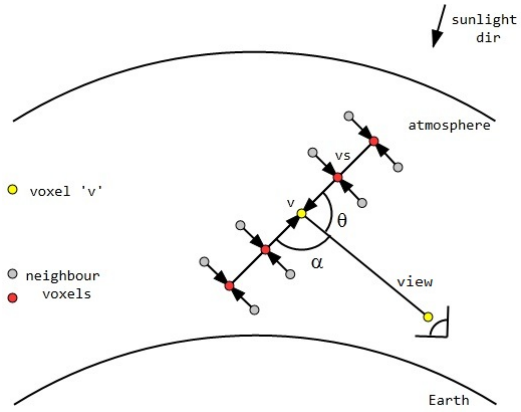


Figure 27: Gather of scattered light

For each sample along the view direction it is necessary to determine the eight voxels closer to it and interpolate these values. In the algorithm in `code 5` the variable *ivoxel* is a virtual voxel centred at the sample point and whose values are the result of the interpolation from the eight neighbour voxels.

The final result of this model can produce results like the ones present in `figure 28`.

According to Nishita et al. the values of intensity that comes with more than 2 orders of scattering is very low, so respecting that, this results were obtained by gathering just 2 levels of scattering, and coefficients such as air and aerosol particles density are those presented in Nishita et al. (1993), Bucholt (1995) and Nishita et al. (1996a), and are the same as the ones presented in the previous subsection. The g coefficient of Henyey-Greenstein phase function used was $g = 0.99$, $\sigma_{mol} = \frac{K_{mol}}{\lambda^4}$ and $\sigma_{aero} = \frac{K_{aero}}{\lambda^{0.84}} = K_{mol}/(10^{-19})$ ($K_{mol} = 4.5864 * 10^{-32}$). The photons wavelength (λ) was used in meters.

The number of samples for both the pre processing stage as well as the run time computation was 8.

Only three wavelengths were considered: red, green and blue using for each one of them a initial value of intensity of $I_0 = 20$.

Algorithm 4 Nishita et al. multiple scattering model - pre-computation

```
createVoxels voxels[numVoxels]
// init voxels with direct light and directions to neighbours
for each voxel in voxels do
  for n = 0..numNeighbours do
    voxel.neighbour[n] = getVoxelNeighbour(n)
    voxel.dir[n] = voxel.pos − voxel.neighbour[n].pos
  end for
  voxel.directLight = computeDirectLight(voxel.pos, sunDir)
end for
// compute scattered light
for for each voxel in voxels do
  // 1st order pf scattering
  // for each neighbour
  for d = 0..numDir do
    v = voxel.neighbour[d]
    path = voxel.pos − v.pos
    attenutation = att(v.pos, voxel.pos)
    // collect light from all neighbours
    for k = 0..numDir do
      θ = getAngle(voxel.dir[d], v.dir[k])
      voxel.order[1][d] = v.directLight ∗ phase(θ) ∗ attenutation
    end for
  end for
  // 2nd and subsequent orders of scattering
  for n = 2 .. order of scattering do
    for d = 0..numDir do
      v = voxel.neighbour[d]
      path = voxel.pos − v.pos
      attenutation = att(v.pos, voxel.pos)
      for k = 0..numDir do
        θ = getAngle(voxel.dir[d], v.dir[k])
        voxel.order[n][d]+ = v.order[n − 1][k] ∗ phase(theta) ∗ attenutation
      end for
    end for
  end for
  // accumulate the intensities coming from the same direction
  // considering different orders of scattering
  for d = 0..numDir do
    voxel.totalIntensity[d] = voxel.directLight
    for n = 1..ordersofscattering do
      voxel.totalIntensity[d]+ = voxel.order[n][d]
    end for
  end for
end for
```

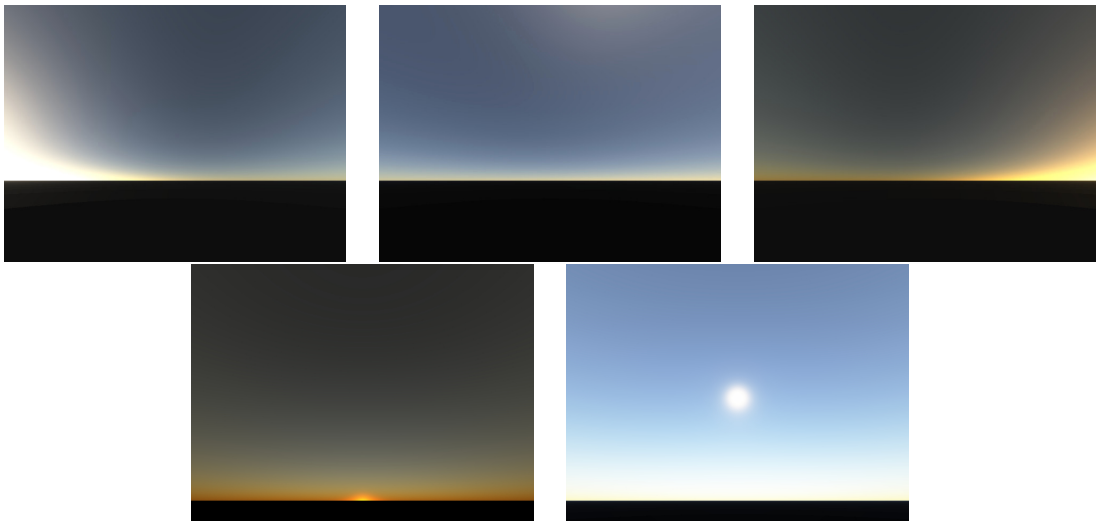


Figure 28: Images using Nishita et al. model of the atmosphere in clear sky with multiple scattering. Top: Sky at Sunrise, Midday and Sunset, respectively. Bot: Sun at sunset and 16 hours

In code 5, there is a description of the process to compute the colours of the sky using the AS model of Nishita et al. with multiple scattering.

Algorithm 5 Nishita et al. multiple scattering model - run-time

for each view direction $view_V$ **do**
  $sample_V = (camera_P - intersectSphere(camera_P, view_V, atmRadius))/numSamples$
  $sample_P = camera_P + sampleV * 0.5$
  $sampleLength = length(sampleP)$
  $intensity = 0$
  for $n = 0..numSamples$ **do**
    // create a voxel with interpolated values
    // considering the 8 voxels closer to $sample_P$
    $ivoxel = getVoxelInterpolatedValues(sample_P)$
    //collect direct light
    $\theta = getAngle(sample_V, sunDir_V)$
    $sampleIntensity = ivoxel.directLight * phase(\theta)$
    // collect light coming from the other directions
    for $d = 0..numDirs$ **do**
      $\theta = getAngle(sample_V, ivoxel.dir[d])$
      $sampleIntensity+ = ivoxel.totalIntensity[d] * phase(\theta)$
    end for
    $sampleIntensity* = att(camera_P, sample_P)$
    $intensity+ = sampleIntensity$
    $sample_P+ = sample_V$
  end for
  $render(viewV, intensity)$
end for

3.4 MODELLING WITH BASIS FUNCTIONS

The main issues of the previous models, based on ray-marching, is the computational time required to produce results with good quality. So, Dobashi et al. (1997) in 1995 (and Preetham et al. (1999) in 1999) proposed to replace the scattering integrals with simpler analytical expressions to reduce the number of computations necessary to obtain the colours of the sky.

To achieve this reduction both proposed a set of analytical functions that are capable of, given a viewing direction and the position of the sun, describe the light intensity for a given light wavelength, and consequently, the colour of the sky in that viewing direction.

In Dobashi et al. (1997) three alternatives are proposed to approximate the scattering integral. In common to all these alternatives there is a preprocessing stage where samples are computed. The model of Kaneda et al. (Kaneda et al. (1991)) is used to obtain the initial set of values of intensity.

Some alternatives imply the usage of basis functions, where as the simplest is based only on linear interpolation.

In this work a coordinate system based on spherical coordinates is used where the sky is an hemisphere whose centre is the viewpoint/camera (`figure` 29).

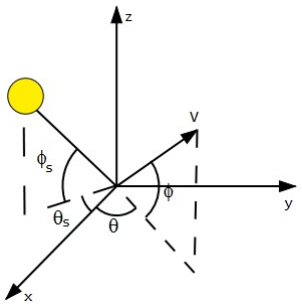


Figure 29: Dobashi et al. coordinate referential

In `figure` 29 ϕ and θ are the spherical coordinates of the viewing direction, with ϕ being the elevation angle and θ the azimuth. The sun direction can also be characterised in spherical coordinates as (ϕ_s, θ_s).

Assuming that θ_s is zero, i.e. the sun is always in the XZ plane, the intensity distribution of the sky in a given view direction $v(\phi, \theta)$ and for a given wavelength λ can expressed as $L_{sky}(\lambda, \phi, \theta, \phi_s)$. Note that if θ_s is different from zero the above characterisation is still valid considering $L_{sky}(\lambda, \phi, \theta - \theta_s, \phi_s)$.

As mentioned before the simplest alternative proposed by Dobashi et al. is to use simple interpolation. For that purpose the sky dome hemishpere is divided as a grid, and for each vertex of the grid, where a vertex defines a viewing direction, it is stored a set of intensity values corresponding to the the possible pairs of wavelength and sun altitude. According to Dobashi et al. this approach would be extremely fast as only a bilinear interpolation operation would be required at runtime. The main

argument presented against this approach is the memory required to store the precomputed values. Although a valid argument at the time of the proposal, this would be a viable alternative today with common graphics hardware having over 2GB of RAM.

The other alternatives imply the usage of basis functions. Note that the interpolation alternative can also be described in terms of basis functions considering delta functions as noted by the authors, but it is simpler to consider it simply as an interpolation approach.

Spherical harmonics (Green (2003a)) had been used before to represent reflectance functions as in Cabral et al. (1987). For the majority of the sky this approach could be applied successfully as the colour variation is smooth, i.e. there is no high frequency detail (Green (2003b)). However, according to Dobashi et al., the number of degrees of spherical harmonic functions required to express the sharp peak of intensity around the sun would be very high. Up to 20 degrees were used and still the errors reported are too high for that particular area.

The third approach proposed by Dobashi et al. is uses cosine basis functions. According to the authors, this method presented, when the paper was written (1995), the best trade-off between memory usage and time consumption.

In this scenario the sky dome is divided in bands (`figure` 30). For each slice a set of cosine basis functions are used for each wavelength and sun altitude angle. In each slice the value of L_{sky} can be computed as described in (`equation` 45).

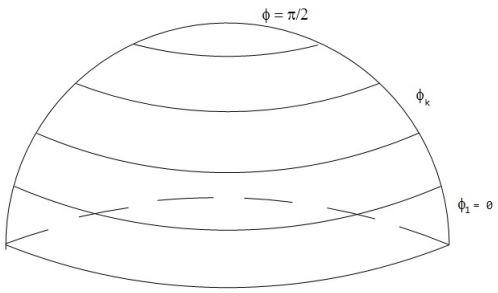


Figure 30: Dobashi et al. skydome

$$L_{sky}(\lambda, \phi, \theta_j, \phi_s) = \sum_{i=0}^{N_j} c_i \, W_{i,j}(\lambda, \phi_s) \, cos(i\theta), \quad (j = 1, ..., M) \tag{45}$$

In `equation` 45 M is the number of vertical divisions, θ_j is the altitude angle for slice j, N_j is the number of cosine functions used to represent the intensity distribution in direction $theta_j$, $W_{i,j}(\lambda, \phi_s)$ is the weight for cosine function i in slice j, and c_i is a normalising factor. When $i = 0$, $c_i = \sqrt{\frac{1}{\pi}}$, and for other values of i, $c_i = \sqrt{\frac{2}{\pi}}$.

The weights are computed using numeric integration as shown in equation 46.

$$W_{i,j}(\lambda,\phi_s) = c_i \Delta\theta_w \sum_{k=0}^{K_w} (cos(ik\Delta\theta_w) L_{sky}(k\Delta\theta_w,\phi_j,\phi_s) \quad (j = 1...M, i = 1...N) \tag{46}$$

where K_w is the number of sample points used in the numerical integration, and $\Delta\theta_w = \pi/K_w$. In this work the values of L_sky are produced by the AS model of Kaneda et al. (Kaneda et al. (1991)).

Dobashi et al. also explores the optimal number of cosine functions for each slice. This helps to keep the number of basis functions under control, while at the same time not sacrificing quality. For instance, when the altitude of the sun coincides with slice more functions are required to depict accurately the area around the sun. On the other hand, if the sun is high the changes in the sun intensity along a slice are smaller.

The authors present an error function to limit the error obtained in each slice. The method starts with a single cosine function and evaluates an error measure. If the error value is above a user defined threshold then the number of basis functions is incremented. The process is repeated until the error is below the threshold.

The proposed measure is a mean square error between the the values of L_{sky} obtained using Kaneda et al. (1991), and the intensity approximation using the basis functions (equation 47).

$$E(\lambda,i,j) = \int_0^{\pi} \left(L_{sky}(\lambda,\phi,\theta_j,\phi_s) - \sum_{k=0}^{i} c_k \, W_{k,j}(\lambda,\phi_s) \, cos(k\theta) \right)^2 d\theta \tag{47}$$

Since, as noted by the authors, cosine functions are orthogonal to each other, and normalised by c_k, this error measure can be rewritten as in equation 48.

$$E(\lambda,i,j) = \int_0^{\pi} \left(L_{sky}(\lambda,\phi,\theta_j,\phi_s) \right)^2 d\theta - \sum_{k=0}^{i} W_{k,j}(\lambda,\phi_s) \tag{48}$$

To simplify the choice of the threshold the error can be normalised as in equation 49.

$$E_N(\lambda,i,j) = \frac{E(\lambda,i,j)}{\int_0^{\pi} \left(L_{sky}(\lambda,\phi,\theta_j,\phi_s) \right)^2 d\theta} \tag{49}$$

To obtain the colours of the sky for a certain view direction $v(\theta,\phi)$ first it is necessary to compute the intensity values at the slices above and below θ. Let θ_k be the altitude angle of the slice below θ. Using equation 45 the values for $L_{sky}(\lambda,\phi,\theta_k,\phi_s)$ and $L_{sky}(\lambda,\phi,\theta_{k+1},\phi_s)$ are computed. Afterwards the intensity for view direction (ϕ,θ) can be computed as described in equation 50.

$$L_{sky}(\lambda,\phi,\theta,\phi_s) = \frac{\phi_{k+1}-\phi}{\phi_{k+1}-\phi_k} L_{sky}(\lambda,\phi,\theta_k,\phi_s) - \frac{\phi-\phi_k}{\phi_{k+1}-\phi_k} L_{sky}(\lambda,\phi,\theta_{k+1},\phi_s) \tag{50}$$

Figure 31: Images of the sky at noon and at sunset and using basis functions - Dobashi et al. (1997)

This method is not as physically accurate as the ones presented before. Note that the error is only limited in the slices. The interpolation process can produce results with error higher than the threshold, which can be particularly meaningful in the area closer the sun. Nonetheless, it reduces much of the necessary calculations and therefore reduces the amount of time necessary to produce images at run-time.

3.5 ATMOSPHERE AND TURBIDITY

In 1999, Preetham et al. followed the same idea as Dobashi et al. but, instead of using basis functions to compute the sky colours, a different set of functions proposed in Perez et al. (1993) were used. Preetham et al. developed a method that, based on measurements of light intensity distributions in different weather conditions, modelled analytic expressions to reproduce the colour of the sky.

For his model, Preetham et al. considers a term called turbidity that is a measure of the fraction of scattering due to haze as opposed to molecules Preetham et al. (1999), hence, turbidity is a measure of the thickness of the atmosphere. A higher value of turbidity means a more dense haze generating fog, and a lower value means clearer sky.

As Preetham et al. states, "Although turbidity is a great simplification of the true nature of the atmosphere, atmospheric scientists have found it a practical measure of great utility. Because it does not require complex instrumentation to estimate turbidity, it is particularly well-suited for application in graphics" - Preetham et al. (1999).

Preetham et al. presents 2 different models to compute the amount of light that is in-scattered towards the camera. A skylight model, in which the colours of the sky are computed and an aerial perspective model that is used to compute the amount of in-scattered light when computing the colour of an object .

The skylight model proposed by Preetham et al. intended to use an analytical expression that given a light direction provides give the spectral radiance, i.e, the amount of light that reaches the camera.

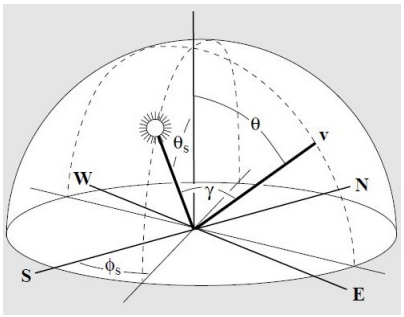


Figure 32: Preetham sky model - Preetham et al. (1999)

First Preetham et al. simulated and computed the sky spectral radiance function for a large number of sun positions and values of turbidity.

For this simulations they used the method in Nishita et al. (1996a) (considering therefore multiple scattering). With the results obtained with these simulations they needed a luminance formula capable of reproducing the variations of colour and light in the sky. For that purpose they compared the models of luminance present in CIE (1995), Dobashi et al. (1997) and Perez et al. (1993) and according to Preetham et al. they found out that the luminance model presented in Perez et al. (1993) was more accurate.

This model of luminance would give the values of spectral radiance in the xyY colour space (Wikipedia (2014c)), in which the values x and y represent chromaticity and Y represents luminance. These values in the xyY colour space can than be converted to XYZ and then to RGB (Wikipedia (2014c)).

The colour of the sky in a certain weather condition T (turbidity) for a given sunlight direction $s = (\theta_s, \phi_s)$ and a view direction $v = (\theta, \gamma)$ (`figure` 32) can be given by values of cromaticity (x and y), and by a value of luminance (Y), which are given by:

$$Y = Y_z \frac{F(\theta,\gamma)}{F(0,\theta_s)}, \quad y = y_z \frac{F(\theta,\gamma)}{F(0,\theta_s)}, \quad x = x_z \frac{F(\theta,\gamma)}{F(0,\theta_s)} \tag{51}$$

where $F(\theta,\gamma)$ represents the luminance function of Perez et al. which describes the sky light distribution according to the direction of incident sunlight and some factors, such as, darkening or brightening of the horizon (A), luminance gradient near the horizon (B), relative intensity of the circumsolar region (C), width of the circumsolar region (D) and relative backscattered light (E) (see `equation` 52).

$$F(\theta,\gamma) = (1 + Ae^{B/cos\theta})(1 + Ce^{D\gamma} + Ecos^2\gamma) \tag{52}$$

The values for the luminance function (A, B, C, D and E) as well as the values for the zenith luminance (Y_z) and zenith cromaticities (x_z and y_z) were pre-computed according to the value of turbidity and are present in the appendices in Preetham et al. (1999).

Figure 33: Images using Preetham model of the atmosphere in clear sky

The turbidity value used to obtain the results in `figure` 33 is 2. All the other values were mentioned before. The images 33 were obtained using Preetham et al. model values and tables according to its paper Preetham et al. (1999).

The work of Preetham et al. allowed to obtain the colours of the sky faster without a significant loss in quality due to the parameter fitting according to a high quality model Nishita et al. (1996a).

The work of Preetham et al. still is the base for many of the modern video-games and flight simulators.

3.6 IMPROVEMENT STRATEGIES IN ATMOSPHERIC SCATTERING MODELS

One of the first GPU optimisations was presented by Dobashi et al. in 2002 Dobashi et al. (2002). The process of computing the colours of the sky is very similar to the one proposed in Nishita et al. (1993). The main difference consisted in using sampling planes instead of sampling directions and for each of the points in each of the planes `equation` 22 is used to compute light intensity values.

The objective was to use the GPUs capacity to process multiple calculations simultaneously. The following workflow was proposed:

- Compute the camera parameters such as the view frustum and set the number of sampling planes to be sent to the GPU and place them along the view frustum, perpendicular to the center view direction;
- Order the sample planes by distance and send each plane at a time to the GPU (starting with the more distant plane);
- For each point of sample in the sampling plane (vertices) compute the in-scattering value using `equation` 21;
- Accumulate the result using blending.

This model presents a strategy to implement physical based models and perform ray-marching using the GPU.

The contribution of Dobashi et al. (2002) is more focused on light shafts and cloud rendering. The details for these light shafts and clouds will be discussed with more detail further ahead in `sections` 4.1 and 4.2.

In 2004 O'Neil (2004) analysed Nishita et al. (1993) and proposed a performance optimisation improving the lookup table used in the original work.

The lookup table proposed by Nishita et al. was able to reduce the number of calculations by pre-calculating the attenuation of the sunlight path (att_{sun}). It is still required to compute the attenuation from the point of scattering to the camera (att_{view}), and the integral of in-scattering (the evaluation of samples along the view ray).

O'Neil proposed a lookup table that pre-computed the optical depth of any ray based on the height of the point where the ray is fired and its vertical angle.

The lookup table has two entries (x, y), where x represents a specific altitude from the ground and y represents a vertical angle. The value of x goes from 0 (on the ground) to 1 (top of the atmosphere), and the value of y goes from 0 to 180 degrees and represents the vertical angle (see `figure` 34).

Each table cell contains 4 channels, 2 for Rayleigh and 2 for Mie. For both Rayleigh and Mie channels, one channel is used to store the atmosphere scale density at a given altitude x and the other stores the optical depth of a ray fired from altitude x with a vertical angle y.

This re-arrangement of the lookup table allowed to not only the fast gathering of the optical depth of the light in its path from the sun into any point in the atmosphere, but also the optical depth of light in the viewers direction. According to O'Neil, if the lookup tables are precise, i.e. a large number of samples is used to build them, the number of runtime samples to approximate in-scattering integral can be significantly lower than previous methods to obtain good result, with O'Neil reporting good results with as few as 4 samples.

Recall the equation to compute the attenuation along a path (`equation` 53), and its usage to compute the intensity arriving at the camera(`equation` 54).

$$att(s,\lambda) = exp\Big(- (4\pi\sigma_{mol}(\lambda) \int^{s} \rho_{mol}(p)dp \; + \; 4\pi\sigma_{aero}(\lambda^4) \int^{s} \rho_{aero}(p)dp)\Big) \tag{53}$$

$$(\lambda, s, P_{cam}) = \int_{P_{cam}}^{P_{end}} I_0(\lambda) att_{sun}(p)(phase_{mol}(\theta) + phase_{aero}(\theta)) att_{view}(p)dp \tag{54}$$

A table lookup provides the values for both integrals found in `equation` 53 considering a ray starting at a given height, and with a given direction provided by a vertical angle. To solve `equation` 54 it is required to compute the attenuation for multiple sun rays and view rays.

To better understand how to obtain these integral approximations consider `figure` 34.

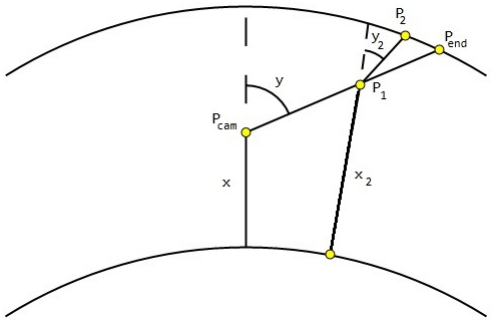


Figure 34: lookup table elaboration by O'Neil

The process to obtain the optical depths to compute att_{sun} is very simple. Consider the path $P_1 \rightarrow P_2$. To obtain the values relative to point P_1 in a view direction with a vertical angle y_2 a single lookup (with interpolation) with the pair of values (x_2, y_2) is required.

On the other hand, the process to obtain the optical depths to compute att_{view} is not so straight forward. This is because the values in the lookup table store values of optical depth of a path that goes from a given point in the atmosphere to the top of the atmosphere. To compute the optical depth of the path $P_{cam} \rightarrow P_1$ two lookups are required.

To get the optical depth in path $P_{cam} \rightarrow P_1$, it is necessary to get the optical depth of P_{cam} with values (x, y) and subtract the optical depth of P_1 (by performing a lookup with the pair of values (x_2, y)). This would give the value of optical depth desired.

Lets consider that the view ray from point P_{cam} intersects the earth. If the ray intersects the earth in a point P_{earth}, the way to compute the optical depth from P_{cam} to P_{earth} is exactly like the same method above because according to O'Neil the optical depth is computed just like there was no planet. This means that to obtain the desired value, subtract the value of optical depth at P_{earth} to the value of optical depth at P_{cam}.

In O'Neil (2005) a different approach was proposed. His goal was to implement atmospheric scattering using only shader model 2.0, which does not allow texture fetches in the vertex shader. The solution was to replace the lookup table with functions that would fit the computed data.

O'Neil plotted the results obtained for the various heights of x with the several angles of y and proposed the usage of two analytical expressions: one that explained the exponential distribution in x, and a scale/fit function that explained the difference in scale caused by varying angle y.

O'Neil used a scale of the Earth's parameters in which there are two important values relative to scale. The atmosphere thickness, that according to O'Neil is about 2.5 percent of the planet radius, and the relative height at which the density achieves its average value, $sDepth$. O'Neil's uses a value of 0.25.

With this in mind, O'Neil plotted the results obtained for the various heights of x with the several angles of y and realised that the lines of the graph followed a exponential distribution but the scale of each line on the graph varied dramatically. He then decided to normalise the lines and found out that the lines varied according to a an exponential function:

$$exp_x(x) = exp(-4x) \tag{55}$$

Part of the problem was solved, as O'Neil says "the x dimension (height) of the lookup table is being handled by exp(-4x)" - O'Neil (2005).

According to O'Neil, the only part of the optical depth that was not handled by the function $exp(-4x)$, was the scale used to normalise the lines on the graphs mentioned previously. Through a new graph, where along the values of x were plotted the values of scale of y, O'Neil found that the values fitted a polynomial curve (`equation` 56).

$$scale_y(\theta) = \frac{1}{sDepth} * exp(-0.00287 + c * (0.459 + c * (3.83 + c * (-6.80 + c * 5.25)))) \tag{56}$$

where $c = 1 - cos(\theta)$.

These simplifications allow to replace the lookups mentioned before by the computation of these two functions. Hence, instead of using a lookup, to compute the optical depth of a ray starting from a given height x and a direction provided by an angle θ `equation` 57:

$$opticalDepth(x,\theta) = exp_x(x)\ scale_y(\theta) \tag{57}$$

With this optimisations it was possible to avoid the shader bottleneck caused by the lookups.

The results of this model can be seen in `figure` 35. In sum, these simplifications were necessary to make the algorithm capable of producing results it real-time. One important factor to retain is that the results obtained by O'Neil are not very different in quality when comparing to the results obtained by Nishita et al, despite the simplifications performed.

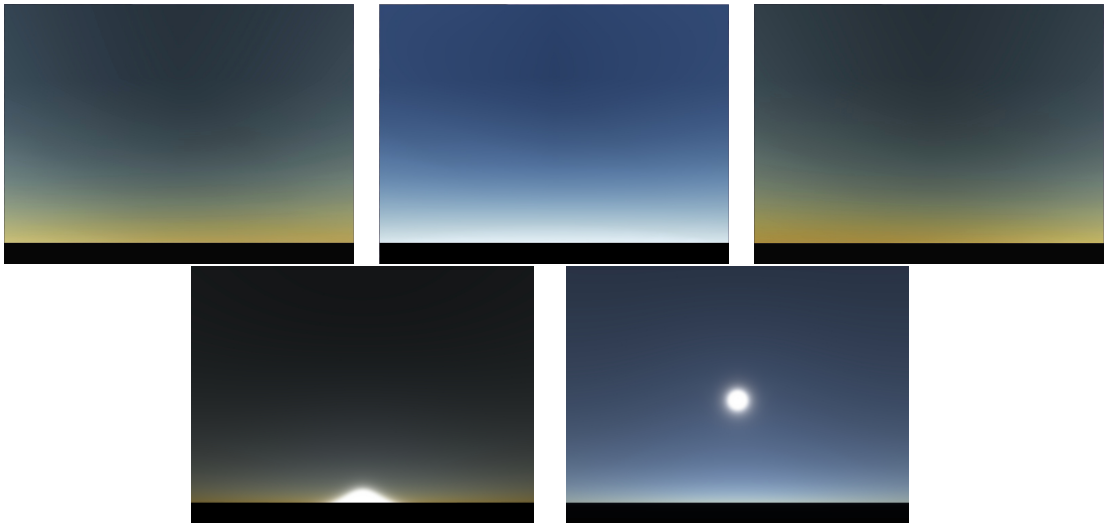


Figure 35: Images using O'Neil model of the atmosphere in clear sky. Top: Sky at Sunrise, Midday and Sunset, respectively. Bot: Sun at sunset and 16 hours

The scattering coefficients used to obtain these images, namely $\sigma_{mol}(\lambda) = \frac{K_{mol}}{\lambda^4}$ where $K_{mol} = 0.003$ and $\sigma_{aero}(\lambda) = \frac{K_{aero}}{\lambda^{0.84}}$ where $K_{aero} = 0.0015$. The photons wavelength (λ) was used in micrometers. For each view ray, 5 samples were obtained.

Only 5 samples were computed for each view ray.

The process to render the colours of the sky is exposed in the `code` 6 and refers to the vertex and fragment shaders to obtain the colours of the sky. The vertex shader is responsible for calculating the optical-depth of both in-scattering and out-scattering per vertex, using `equation` 57 and the fragment shader is responsible for using the interpolated values from the vertex shader and compute the Rayleigh and Mie phase functions, based on the approximation proposed in Cornette and Shanks (1992), and therefore the final value of colour of the pixel.

The following variables and constants were used in the code:

- $camera_P$: Camera position relative to a referential with origin in the center of the earth
- $scale$: $\frac{1}{atmosphereradius - earthradius}$
- $sunDir_V$: normalised vector pointing towards the sun
- $scaleDepth$: 0.35

- $scaleOverScaleDepth$: $\frac{scale}{scaleDepth}$
- $sunIntensity$: The sun intensity.

Algorithm 6 O'Neil AS model shaders

for each vertex v **do**
// view direction from camera to vertex in the sky dome
$view_V = v - camera_P$
// distance from camera to the vertex (top of the atmosphere)
$distance = length(view_V)$;
// normalised view direction
$view_V / = distance$
// compute scattering offset
$cameraHeight = length(camera_P)$
// vertical angle of $view_V$
$angle = dot(view_V, camera_P)$
$sDepth = exp(scaleOverScaleDepth * (earthRadius - cameraHeight))$
$opticalOffset = sDepth * scale(angle)$;
// compute the first sample and sample vector
$sampleLength = distance / numSamples$
$sample_V = view_V / numSamples$
$sample_P = camera_P + sample_V * 0.5$
// compute real scale sample length
$scaledLength = sampleLength * scale$
$color = 0, 0, 0$
for $n = 0..numSamples$ **do**
$height = length(sample_P)$
$sDepth = exp(scaleOverScaleDepth * (earthRadius - height))$
$sunAngle = dot(sunDir, sample_P) / height$
$cameraAngle = dot(view_V, sample_P) / height$
// optical depth from camera to sample point
$opticalDepth = opticalOffset - sDepth * scale(cameraAngle)$
// plus scatter for sun ray
$opticalDepth+ = sDepth * scale(sunAngle)$
// compute attenuation summing Rayleight and Mie components
$$att = exp\left(- opticalDepth * \left(\frac{1}{wavelengths^4} \times K_{mol} \times 4 \times PI + K_{aero} \times 4 \times PI \right) \right)$$
$color+ = att * sDepth * scaledLength$
$sample_P+ = sample_V$
end for
// these are the variables that are passed to the fragment shader
$mieColor = color * K_{aero} * sunIntensity$
$rayleightColor = color * \frac{1}{wavelengths^4} * K_{mol} * sunIntensity$
$direction = view_V$
end for
for each pixel p **do**
$angle = dot(sunDir, direction)$
$color = rayleightPhase(angle) * rayleighColor + miePhase(angle) * mieColor$
end for

3.7 PHYSICALLY BASED MODELS AND PRE-COMPUTATION

After the work of O'Neil, Schafhitzel et al. in 2007 (Schafhitzel et al. (2007)) and Bruneton and Neyret in 2008 (Bruneton and Neyret (2008)) tried to improve even further the model of Nishita et al. (Nishita et al. (1993)). For that purpose, these models wanted to combine high performance with rendering quality. Hence, they tried to perform the highest number of pre-computations possible in order to perform less operations in rendering time. While Schafhitzel et al. only consider single scattering, Bruneton and Neyret propose a model that includes multiple scattering events.

Nishita et al. (1993) proposed to pre-compute optical depth for sunlight. O'Neil (O'Neil (2004)) went further and pre-computed the optical depth for any ray. Schafhitzel et al. pre-computed every single integral by assuming not only every possible sunlight and viewing direction but also every possible camera height, and store them in a 3D texture (`figure` 36).

Just like in the cases of Nishita et al. (1993) and O'Neil (2004), Schafhitzel et al. proposed to store the values of optical depth in a table. The difference is that these values are stored in a 3D table/texture. This texture parametrisation can be seen in `figure` 36.

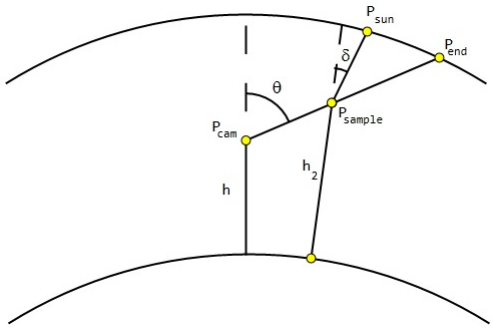


Figure 36: lookup table elaboration by Schafhitzel et al.

The values stored in the texture 3D are computed using `equation` 32 for all combinations of view direction angle (θ), angle to the sun/solar elevation (δ) and height of the point in the atmosphere (h). This parametrisation is similar to the one proposed in O'Neil (2004) but with the consideration of two angles (view direction and sunlight) its possible to precompute the light intensity that reaches the camera.

The intensity stored in the 3D texture represents the amount of light that reaches point P_{cam} in the entire path $P_{cam} \rightarrow P_{end}$ with a sunlight direction δ (`equation` 58).

$$texture3D(h,\theta,\delta) = \int_{pos(h)}^{v(\theta)} I_0(\lambda) att_{sun}(p,\delta)(phase_{mol}(\theta-\delta) + phase_{aero}(\theta-\delta)) att_{view}(p,\theta) dp \tag{58}$$

where h is the height of the camera, θ is the vertical angle of the view direction, and δ is the vertical angle for the sun direction.

This means that, for any view direction, the colour of the sky can be obtained by a simple lookup in the table. With this approach, the calculations of complex integrals have been replaced for the calculations of indexes in a 3D table, and so the amount of in-scattered light can be given by `equation` 59.

$$I(\lambda, h, \theta, \delta) = texture3D(h, \theta, \delta) \tag{59}$$

Schafhitzel et al. considered only the wavelengths of RGB colour space.

This lookup is only valid if there are no objects or terrain to render. If indeed, are objects or terrain that is to render, then the number of lookups change from 1 to 2. This process is very similar to the one mentioned above when analysing the work of O'Neil (O'Neil (2004)) and is represented in `figure` 37.

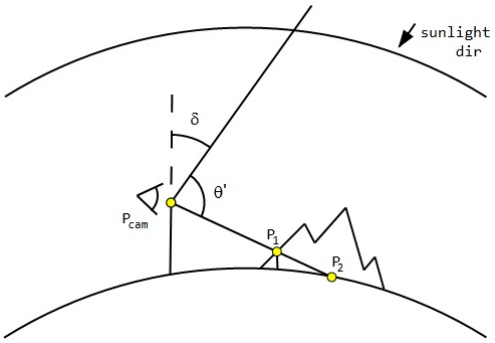


Figure 37: lookup table elaboration by Schafhitzel et al.

To evaluate the amount of in-scattered light in the path $P_{cam} \rightarrow P_1$ its only necessary to perform a lookup at point P_{cam} and subtract to the value of the lookup at point P_1 (see `equation` 60).

$$I(P_{cam}, P_1) = texture3D(h(P_{cam}), \theta, \delta) - texture3D(h(P_1), \theta, \delta) \tag{60}$$

where $h(P_{cam})$ *and* $h(P_1)$ represent the height at P_{cam} and P_1 respectively, and $\theta = \theta' + \delta$. The same can be said about P_1. Another detail in Schafhitzel et al. model, is that for the terrain they pre-compute all components of the scattering integral with the phase function separately in order to obtain a better light precision.

Because the values of light intensity are stored in a table, this allows to compute `equation` 32 with as much as samples as desired.

These improvement allow to obtain the colours of the sky with even less calculations than the AS model of O'Neil.

In 2008, Bruneton and Neyret went further and pre-computed calculations for multiple scattering as well (pre-computing att_{sunMS} in `equation` 25). For this purpose Bruneton and Neyret pre-compute and store the values of multiple scattering in tables. In a process similar to the one used by Nishita et al. with mutiple scattering (Nishita et al. (1996a)), Bruneton and Neyret model uses multiple passes to compute this table, and at each pass the result table sums the values of an order of scattering for each sunlight direction (s), viewing direction (v) and camera position/height (x).

The light contributions in this AS model were decomposed in four main components (see `figure` 38):

- Direct light ($L_0(x,v,s)$) - direct light coming from the sun in direction s
- Reflection ($R(x,v,s)$) - contribution of light due to reflection on earth's surface
- Single scattering ($S(x,v,s)$) - contribution of light from single scattering events
- Multiple scattering ($MS(x,v,s)$) - contribution of light from multiple scattering events

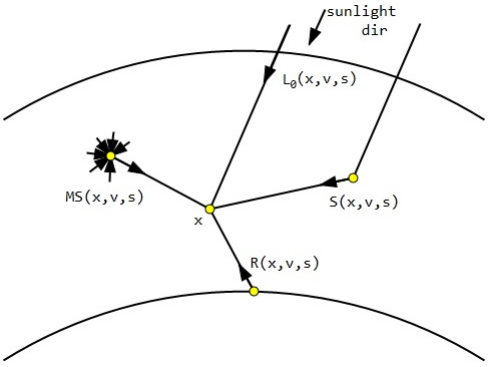


Figure 38: Bruneton and Neyret atmospheric scattering model components

For all these components attenuation is evaluated. The values of attenuation are evaluated just like in O'Neil (2004), i.e., for all combinations of camera height x and view direction v the attenuation values are pre-computed and stored in a table ($att(x,v)$ using `equation` 43).

The component of direct light is evaluated during run-time and is evaluated to obtain the colour of the sun.

The value of in-scattered light is evaluated in a very similar process to the one presented in Schafhitzel et al. (2007). For all combinations of camera height x, view direction v and sun light direction s, evaluate the amount of scattered light along the view direction using a variation of `equation` 32, where the value of the phase function is only computed at run-time, which means that the value stored in the table $S(x,v,s)$ is $S(x,v,s) = \int_x^{length(v)} I_0 att_{sun}(s,p) att_{view}(v,p) dp$.

The component of reflection is evaluated according to the amount of light that reaches a certain point x in the earth's surface, which is modelled as a Lambertian surface (equal value of reflection

independent of the camera position - Wikipedia (2014e)) with a reflection function $reflect(w)$ that describes the amount of light reflected into direction w.

The amount of light reaching the surface at point x is given not only from the sun, but also from the multiple scattering of light.

To evaluate multiple scattering contributions consider `figure 39`.

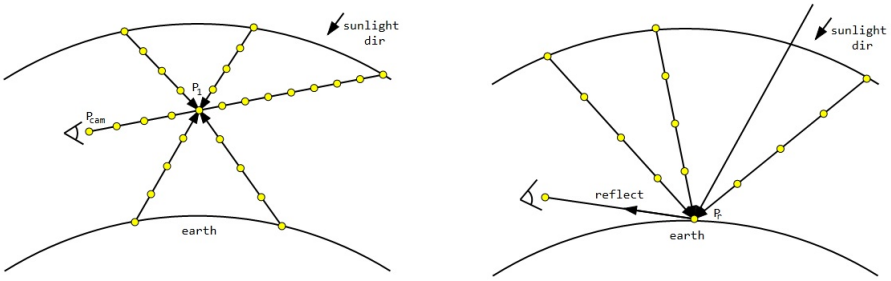


Figure 39: Multiple scattering evaluation

Bruneton and Neyret evaluate and store these values in two different tables. The values of multiple scattering for all combinations of camera height x, view direction v and sun light direction s are stored in one table $MS(x,v,s)$ (`figure 39` on the left), and the values of multiple scattering for all combinations of a surface point x_0 and sunlight direction s are stored in another table $Ir(x_0,s)$ (`figure 39` on the right).

To evaluate the value to store in $Ir(x_0,s)$, for each point x_0 (P_r in right `figure 39`) multiple rays are marched into different sampling directions until the intersection of the top of the atmosphere. For each sample along these rays, multiple rays are marched recursively, thereby providing the multiple scattering effect. The degree of recursion determines the number ofscattering orders. It is also evaluated the amount of light that that reaches x_0 directly from the sun. Hence, the value of $Ir(x_0,s)$ will take into account contributions from direct light, single scattering and n orders of scattering.

The value of $Ir(x_0,s)$ that will reach the camera (`figure 39`), considering a view direction v, is then given by:

$$I(x_0,s,v) = reflect(-v) * Ir(x_0,s) \tag{61}$$

The components of direct light and single scattering can be seen as follows:

1. for direct light: $Ir(x_0,s) = I_0 * att(x_0,s)$
2. for single scattering: $Ir(x_0,s) = \sum_{j=0}^{n_{dir}} \int_{x_0}^{length(j)} I_0\ att(s,p) phase(\theta_j) att(j,p) dp$,
   where $ndir$ is the number of sampling directions at x_0 and θ_j is the angle between s and the sampling direction j. So, in fact, as single scattering is pre-computed and stored in table $S(x,v,s)$, then $Ir(x_0,s)$ can also be seen as $Ir(x_0,s) = \sum_{j=0}^{n_{dir}} phase(\theta_j)\ S(x_0,j,s)$.

3.7. Physically based models and pre-computation

The process to evaluate the values of multiple scattering that will be used to compute the multiple scattering contribution in $Ir(x_0, s)$ and that will be stored in $MS(x, v, s)$, for each point x (P_{cam} in left `figure` 39) consists in marching a ray into the view direction until the intersects the atmosphere or the earth's surface. For each sample p in that ray and in n steps its evaluated the amount of light that is in-scattered from multiple sampling directions s_{dir} at p towards the camera. The value of $MS(x, v, s)$ will contain the n orders of scattering including single scattering and the light reflected at the earth. $MS(x, v, s)$ can be seen as:

$$MS(x, v, s) = \sum_{i=2}^{n} \sum_{j=0}^{n_{dir}} \Big(\int_{x_0}^{length(j)} att(q, j) \sum_{k=0}^{n_{msdir}} inscatt(I_{i-1}, x, k, j) dq \Big) \tag{62}$$

where $inscatt(I_{i-1}, x, k, j)$ is given by:

$$inscatt(I_{i-1}, x, k, j) = phase(\theta_{kj}) I_{i-1}(p, k, j) \Big(\int^{length(k)} att_{view}(k, p) dp \Big) \tag{63}$$

where $I_{i-1}(p, k, j)$ is the total value of light of order $i-1$ that was computed before at a point with position p, view direction k and light direction j. This means that each pass i in n orders of scattering, the value of in-scattering with this combination will be computed and stored in a temporary table of type $\Delta MS(x, v, s)$. So, $I_{i-1}(p, k, j) = \Delta MS(p, k, j)$.

$I_{i-1}(p, k, j)$ value depends if the sampling ray intersects the earth's surface or the atmosphere. If the sampling ray intersects the atmosphere then $I_{i-1}(p, k, j)$ depends only on scattering from the samples in the sampling ray, otherwise, if the sampling ray intersects the earth's surface then its necessary to add the contribution of reflected light. Lets consider single scattering. If the sampling ray intersects the atmosphere, then $inscatt(I_{i-1}, x, k, j) = S(p, k, j)$ which was pre-computed before. If the ray intersects the earth's surface $inscatt(I_{i-1}, x, k, j) = S(p, k, j) + Ir(x_0, s)$, where s is the sun light direction and x_0 is the point in the earth's surface where the sampling ray hit. For the following n scattering orders, it is only necessary to fetch the values pre-computed in the previous step.

The final value of intensity reaching the camera for a given view direction, sunlight direction is given by:

$$Colour(x, v, s) = L_0(x, v, s) + reflect(-v)\ Ir(x_0, s) + phase(\theta)\ MS(x, v, s) \tag{64}$$

where θ is the angle between the v and s.

Comparing with the multiple scattering computation with Nishita et al. of 1996, this model reduces its complexity at run-time. Results can be seen in `figure` 40.

These images were obtained using the coefficients present in Nishita et al. (1993) and Nishita et al. (1996a). The number of orders of scattering computed in this images were 4. For the attenuation pre-computations, 500 samples were used, for the scattering calculations of $S(x, v, s)$ 50 samples per view

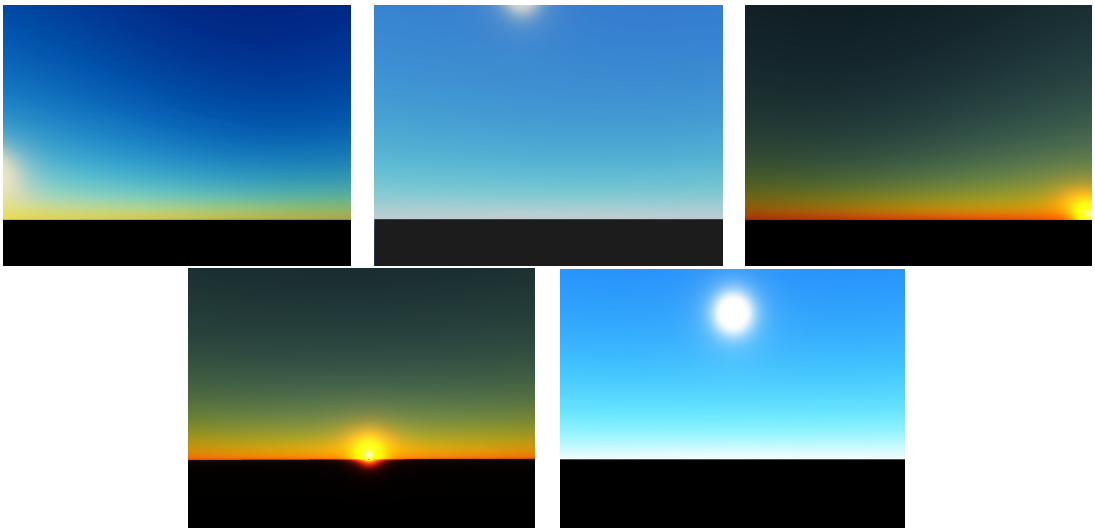


Figure 40: Images using Bruneton model of the atmosphere in clear sky. Top: Sky at Sunrise, Midday and Sunset, respectively. Bot: Sun at sunset and 16 hours

direction and 16 sample directions were used to get multiple scattering. The values used to compute the attenuation, were, $\sigma_{mol} = \frac{K_{mol}}{\lambda^4}$ and $\sigma_{aero} = K_{aero} = K_{mol}/(10^{-25})$ ($K_{mol} = 4.5864 * 10^{-32}$). The photons wavelength (λ) was used in meters.

3.8 TURBIDITY MODEL REVISITED

The work of Preetham et al. had a huge impact in the computer graphics community with many applications still using their AS model.

Their work was deeply analysed in Zotti et al. (2007). The authors compared the results in Preetham et al. (1999) with actual measurements and the ISO/CIE 2003 on Illumination CIE (2004) for standard general sky luminance distributions and conclude that the results for values of turbidity below 2.0 are not correct. Furthermore, the Preetham model does not properly reproduce the noticeable darkening of the sky in the antisolar hemisphere when the sun is low, with luminance values about 2–5 times higher than expected. Also, the brightness peak towards the sun is not as steep as it can be measured or as modelled by the CIE Clear-Sky models.

Based in the work of Zotti et al., Hosek and Wilkie (2012) proposed a more accurate analytical method to the real values of skylight.

For that purpose, Hosek and Wilkie developed a path tracer that performed a huge variety of simulations in different scenarios such as different weather conditions and values of ground albedo. These simulations created a data base of results of the sky light distribution using the scattering equations of Rayleigh, Mie and Henyey-Greenstein. These results were used to fit a new analytical function capable of reproducing with accuracy the values obtained with the path tracer.

The result was an analytical function which consists in a variation of the function presented by Perez et al. (Perez et al. (1993)).

Hosek and Wilkie realised that Perez equation had some limitations. One of those limitations concerns the lack of reproducing a phenomenon (circumsolar ring) that resembles a glow around the solar point. For that purpose Hosek and Wilkie introduced an anisotropy factor $X(g,\alpha)$, where g controls the glow around the solar point. Factor $X(g,\alpha)$ resembles Henyey-Greenstein phase function (`equation` 66).

Another issue concerns the extension of this glow. This extension is more notorious towards the horizon than towards the zenith. For that purpose they added a term capable of applying this effect ($Icos^{\frac{1}{2}}(\theta)$).

There was another issue with Perez et al. equation that would produce a problematic value when the view direction would be at the horizon, i.e., when the value of $\theta = 90$. This originated a division by 0 ($e^{B/cos\theta}$) in `equation` 52). To solve that, a small fudge factor was included turning $e^{B/cos\theta}$ in $e^{B/(cos\theta+0.01)}$.

The major changes mentioned above can be seen in `equation` 65:

$$F(\theta,\gamma) = (1 + Ae^{\frac{B}{cos\theta+0.01}}).(J + Ce^{D\gamma} + Ecos^2\gamma + G\ X(H,\gamma) + Icos^{\frac{1}{2}}\theta) \tag{65}$$

where $X(g,\alpha)$ is given by:

$$X(g,\alpha) = \frac{1+cos^2\alpha}{(1+g^2-2g.cos\alpha)^{\frac{3}{2}}} \tag{66}$$

`Equation` 65 includes some factor when comparing to the original version of Perez et al. The only terms not mentioned yet are G and J.

The G term controls the glow intensity introduced by anisotropy factor X. The term J is applied due to the inclusion of the anisotropy factor. This term will act as a factor to normalise the value of luminance obtained wiht `equation` 65.

As opposed of the skylight model of Preetham et al. that normalises the value of luminance coming in a certain view direction by computing the luminance at the zenith, in Hosek et al. skylight model, they decided to have a different approach, which can be seen in `equation` 67:

$$L(\lambda,\theta,\gamma) = F(\theta,\gamma)L_{M\lambda}(\gamma) \tag{67}$$

where $L_{M\lambda}(\gamma)$ is the mean value of intensity at a sun elevation angle γ. This makes $F(\theta,\gamma)$ as a fitting factor where if $F(\theta,\gamma) = 1$, that means that the mean value of light intensity for an angle γ coincides with its mean value.

There is yet another difference between this model in relation to the model of Preetham et al. and that regards the values from $A..J$ in `equation` 65. While in Preetham et al. those values are a linear function of the values of turbidity, Hosek et al. proposed to model these values using also bezier curves (Wikipedia (2014b)), in order to obtain a better fit of the values of skylight when comparing to the results obtained with the path tracer.

The authors claim to significantly improve the rendition of sunsets and high atmospheric turbidity setups.

From a mathematical point of view the models are similar, with Hosek and Wilkie (2012) adding more coefficients to gain more control and be able to produce more accurate results. These model can produce images like the ones in `figure` 41.

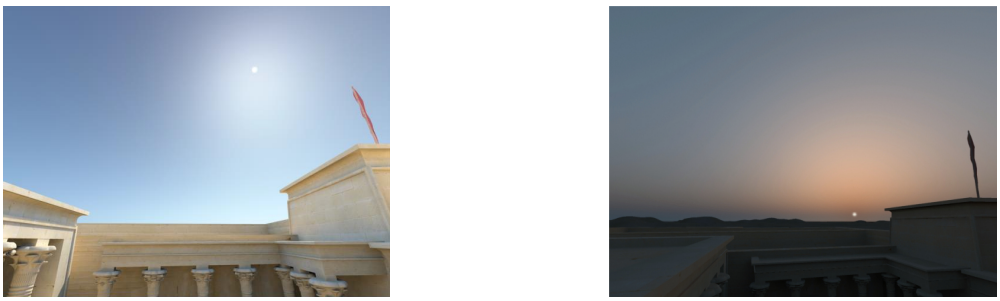


Figure 41: Hosek model images for during the day (left) and at sunset (right) - Hosek and Wilkie (2012)

3.9 MODEL ANALYSIS

The first model studied in this thesis was proposed by Klassen (1987). It is a single scattering model that simplifies the atmosphere density model, considering only two layers with constant density. Furthermore, it was assumed that in-scattering events only occurred in the bottom layer.

The first implementation with an exponential decreasing density atmosphere is proposed by Kaneda et al. (1991). It is also a single scattering model and it computes the intensity values according to the equations that describe the phenomena.

Nishita et al. (1993) proposed to pre compute the values of optical depth for sun rays, thereby simplifying a significant part of the runtime algorithm. The model is similar to Kaneda et al.

Later, Nishita et al. (1996a) extended their original work to include multiple scattering events. In order to achieve this in run time, a pre computation is performed that accumulates the multiple scattered light in a voxel grid surrounding the planet. The quality of this work has been recognised in subsequent analytical based works that the values proposed by this work as the ground truth for function fitting in later works.

Dobashi et al. (1997) proposed to approximate the integrals with cosine basis functions. It is the first work that attempts an analytical approach. The model is only single scattering, and data from Kaneda et al. (1991) was used as ground truth for function fitting.

In Preetham et al. (1999) the main parameter is turbidity, a measure of atmosphere thickness. The model from Preetham et al. is multiple scattering. It is an analytical method that uses the Perez function for luminance and defines a set of coefficients, linearly dependent on the turbidity value, such that sky and sun colours can be reproduced.

The first usage of shaders to compute the colours of sky and sun is by Dobashi et al. (2002). It is basically the same method as in Nishita et al. (1993), but instead of considering sampling points it considers sample planes, allowing for an efficient implementation on the CPU.

The work in O'Neil (2004) goes one step further than Nishita et al. (1993) and pre computes the optical depth for any ray given an height and vertical angle. This approach allowed a very significant relief of the work load during run time.

Later, in O'Neil (2005), the pre computations are replaced with an analytical approach based on function fitting. The original goal was to implement his previous algorithm using only Shader Model 2.0, which does not allow for texture access in the vertex shader (required for the lookup tables). O'Neil's functions effectively replace the lookup table in the previous work.

Schafhitzel et al. (2007) goes back to lookup tables, and adds the sun direction to the tables in O'Neil (2004) as the third entry in the table. In this way, most of the single scattering process is pre computed, leaving only the phase function for the run time (to render the terrain), since it provides high frequency detail.

Following this trend is the work in Bruneton and Neyret (2008), adding multiple scattering, light shafts and ground reflection. The goal is to pre compute as much as possible. As in Schafhitzel et al.

(2007), Bruneton et al. also leave the phase function out of the pre computations but not only to the colour of the terrain but also for the colours of the sky.

The last work described in here is Hosek and Wilkie (2012). This work is based on Preetham et al. (1999) and a deep analysis by Zotti et al. (2007) of Preetham et al. results. Note that one of the authors in Zotti et al. (2007) is also an author in Hosek and Wilkie (2012). The goal is to fix the issues that were identified in the analysis. To achieve that the analytical expression used by Preetham et al. was expanded to include more terms and coefficients. To provide more control these coefficients are now the result of quintic interpolation, as opposed to linear in Preetham et al.

Figure 42 presents a graph containing the dependencies between these methods.

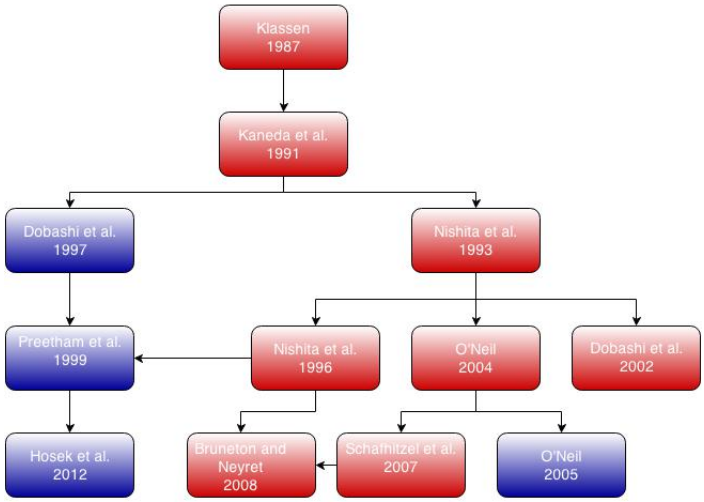


Figure 42: Connections between physically based AS models (red) and Analytical AS models (blue) along time

A summary of the features of the AS models described in this chapter are summarised in table 1.

AS Model	Scattering Eval	Approach	Pre-computation
Klassen (1987)	Single	Physical	No
Kaneda et al. (1991)	Single	Physical	No
Nishita et al. (1993)	Single	Physical	Yes
Nishita et al. (1996a)	Multiple	Physical	Yes
Dobashi et al. (1997)	Single	Analytical	Yes
Preetham et al. (1999)	Multiple	Analytical	Yes
Dobashi et al. (2002)	Single	Physical	Yes
O'Neil (2004)	Single	Physical	Yes
O'Neil (2005)	Single	Analytical	No
Schafhitzel et al. (2007)	Single	Physical	Yes
Bruneton and Neyret (2008)	Multiple	Physical	Yes
Hosek and Wilkie (2012)	Multiple	Analytical	Yes

Table 1: Table with some aspects relative to the AS models presented in this chapter

Only some of these models opted to compute multiple scattering, namely the models in Nishita et al. (1996a), Preetham et al. (1999), Bruneton and Neyret (2008) and Hosek and Wilkie (2012). All the others assumed that the contribution of the multiple scattered rays was not enough to compensate the computational cost. Nevertheless, when comparing the results obtained in terms of quality the multiple scattering approach gets brighter and more saturated skies, due to the increased contribution gathered from multiple scattering.

The difference can be seen specially at sunset when looking towards the horizon. In the comparison between the results of Nishita et al. single scattering (`figures` 23) and Bruneton and Neyret multiple scattering (`figures` 40) we can see that the model with multiple scattering can reproduce with more accuracy the colours of the sunset by having an orange horizon, while in the single scattering model of Nishita et al. at sunset that characteristic orange horizon does not show, and due to the lack of bright the colours are somewhat too dark.

When analysing the differences between the analytical and physical approaches to the problem of evaluating atmospheric scattering, it is clear that there is no clear winner. While analytical models are based on physical data, and hence provide only an approximation to the ground truth, in the analytical this approximation is highly cost effective allowing for accurate models running in real-time.

4

ATMOSPHERIC SCATTERING - OTHER EFFECTS

This chapter's goal is to provide some pointers relating to other atmospheric effects. It is included mainly for completeness, rather than an attempt to do a full STAR on each of the subjects presented.

In this chapter other effects related with the atmospheric scattering phenomenon will be addressed. These include light shafts (sec. 4.1), cloud modelling and rendering (sec. 4.2), fog effects (sec. 4.3), simulating rain (sec. 4.4), rainbow rendering (sec. 4.5), and lightning recreation (sec. 4.6).

4.1 LIGHT SHAFTS

The term light shafts describes an optical phenomenon that is perceived as beams of light in a shadowed environment, radiating from the light source (Prast and Fruehstueck (2013)). These shafts of light can be distinct as beams from the darker ambiance because the small particles in the air scatter the incoming light in different directions, namely into our eyes.

Examples of this phenomenon can be found everywhere, like beams of light piercing through a cloud, or beams of light coming from a window, or even from trees in the forest (`figure` 43).

Figure 43: Light shafts in the forest by Greg Paupst

As light shafts are perceived due to scattering of light coming from a light source, then rendering light shafts could be done using atmospheric scattering render equations, as long as shadows are taken

into account. In figure 44 is represented a simple view of the problem. Along the view direction, only some sections receive direct light from the sun.

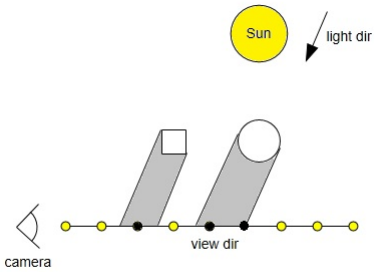


Figure 44: Light shafts representation

So a way of computing this effect consists in using the equations 22 and 19 as follows:

$$I(\lambda,\theta,s,n) = \sum_{i=0}^{i=n} I_0(\lambda) att_{sun}(s,n,i) phase(\theta) att_{view}(s(i),n) \tag{68}$$

where $att_{view}(s,n)$ is given by:

$$att_{view}(s,n) = exp(\frac{4\pi K}{\lambda^4} \sum_{i=0}^{i=n} \rho(s(i))ds) \tag{69}$$

and $att_{sun}(s,n,i)$ is given by:

$$att_{sun}(s,n,i) = exp(\frac{4\pi K}{\lambda^4} \sum_{j=0}^{j=n} \rho(s(i))ds) shadow(i)\ ,\ shadow(i) \in \{0,1\} \tag{70}$$

The difference between these equations and the originals reside in the $shadow(i)$ factor, i.e., if the sample point i is in shadow, then it does not receive direct light from the sun ($shadow(i) = 0$), therefore the camera does not receive in-scattered light from that sample point. Otherwise, the sample receives direct light from the sun ($shadow(i) = 1$) and the process to compute the light that reaches the camera is the same as before.

The process described above is only valid if we just consider single scattering. For multiple scattering the calculations becomes more complex. Nevertheless, if we use as base, equation 25, than a possible expression to evaluate the shadow areas taking into account multiple scattering can be:

$$att_{sunMS}(s,n,i) = \sum_{k=0}^{n} \prod_{j=1}^{j=k} att(path(j)) phase(\theta_j) shadow(i,j)\ ,\ shadow(i,j) \in \{0,1\} \tag{71}$$

where $shadow(i,j)$ is 1 if between sample point i and sample point j is any obstacle that prevents the light from point j to reach i. Otherwise $shadow(i,j) = 0$.

Regarding the identification of the shadowed areas, the presented algorithms to render light shafts resort to shadow maps and shadow volumes.

One of the first methods proposed to render light shafts was presented in Kaneda et al. (1991) (although they mention it as light beams in fog) and it consisted in sampling the viewing ray along its path and performing shadow tests for each sample, so, for each lit sample, compute light intensity using `equation` 23.

The method was improved, performance wise, in 2002 by Dobashi et al. (2002). The main difference was the usage of GPUs. Instead of sampling the view direction, Dobashi et al. used sampling planes. Their strategy was to use the structure of the view frustum, pyramid shape, and place some sampling-planes along that pyramid.

If the difference in the values between planes was very different, for example, if the values of intensity in sample (i,j) of plane k was very different fro the values in sample (i,j) of plane $k+1$ (which happens due to shadow zones), then it was necessary to place additional sub-planes between those planes in order to increase the detail in those areas.

The algorithm work-flow can be described as follows:

1. n sampling planes are placed equally distant along the view frustum/pyramid
2. for each sample $s(i,j)$ of every plane k, compute the amount of light reaching it and test if it is in shadow
3. for every consecutive plane k and $k+1$, if the values of the samples are similar then continue, otherwise if $s(i,j)$ are very different (value surpassing a threshold value η), then place sub-samples and repeat this step until the value of difference is below η
4. collect the values of all sampling-planes and blend them

A representation of this sub-planes can be seen in `figure` 45.

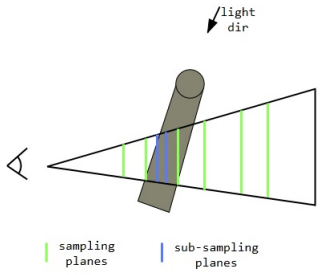


Figure 45: Evaluation of shadow using planes (green) and sub-planes to add detail (blue)

This is made in order to to reduce the number of computations necessary to obtain the values of light intensity reaching the camera.

The problem of this method is that it stills need to use sampling to solve the scattering equations and to identify the shadowed areas.

Some years later in Bruneton and Neyret (2008) from 2008, Bruneton and Neyret, proposed a different approach. Taking advantage of pre-computing the attenuation of light for every possible camera position, view angle and sun position, they proposed the following:

- Using a geometry shader they apply an algorithm that extrudes the silhouette edges of objects observed from the sun, as in a shadow volume.
- With this it is possible to compute the full length of the shadowed areas (`figure` 46 $out - in$)
- Knowing the length and using the pre-computed values is possible to compute the amount of light that would be in-scattered in the shadowed zones in and out and subtract it to the final value of light reaching the camera

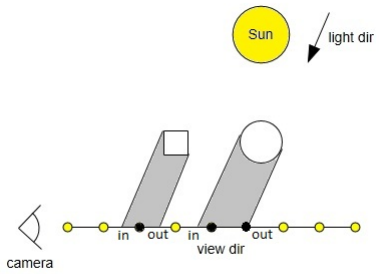


Figure 46: Light shafts

So the final value of light intensity could be seen as:

$$I(\lambda,\theta,s,n) = I(\lambda,\theta,s,n) - I(\lambda,\theta,in\_shadow(s),n) \tag{72}$$

In Engelhardt and Dachsbacher (2010), Engelhardt and Dachsbacher proposed a rendering technique for textured light sources in single-scattering media. They realised that the in-scattered light values varies orthogonally, but most smoothly, along crepuscular rays and light shafts. This aspect allowed the sampling in image space by using epipolar geometry and can be seen in `figure` 47.

We can see in `figure` 47, that along the red line the value of in-scattered light varying in a predictable way. This red line can be seen as a epipolar line (line that goes from the light source (sun in this case) to the end of the screen space).

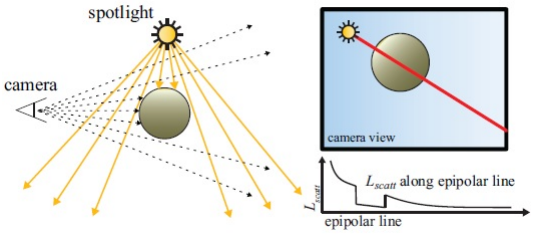


Figure 47: Epipolar analysis of the light intensity - Engelhardt and Dachsbacher (2010)

So, the proposal of Engelhardt and Dachsbacher passes by sampling several epipolar lines in order to compute the amounts of light reaching the camera taking into account the occluded/shadow areas. The shadowed areas are computed and stored in shadow/depth maps.

It is important to mention that this method requires to change the coordinate system from rectangular coordinates to epipolar coordinates and vice-versa.

The epipolar lines are usually placed like in `figure` 48).

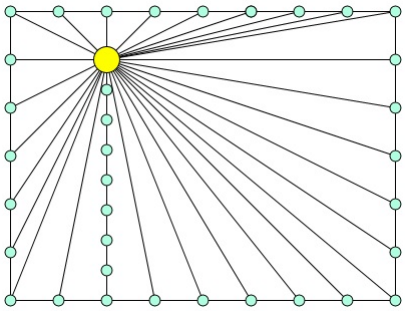


Figure 48: Epipolar Sampling

These marked samples on the view space (epipolar samples) will be used to generate a 3D texture were values of in-scattered light will be stored. So, the entries in the 3D texture, that coincide with the view rays passing by the epipolar samples, will compute the amount of in-scattered light using `equation` 68. The other entries in-scattered light values will be interpolated from the other entries.

This method was improved in 2011 by Chen et al. Chen et al. (2011). They realised that it was possible, by using epipolar geometry, to create an efficient shadow map made essentially from a one-dimensional height map (`figure` 49).

With this structure, shadow test is intrinsically a check if current position on the ray is under this height map or above it. So they proposed constructing a 1D min/max binary tree and use this structure to identify lit and shadowed regions on the ray. This structure was able to speed up the calculations.

The methods discussed before were discussed in Yusov (2013), and from that an implementation came that can produce results like the ones in `figure` 50.

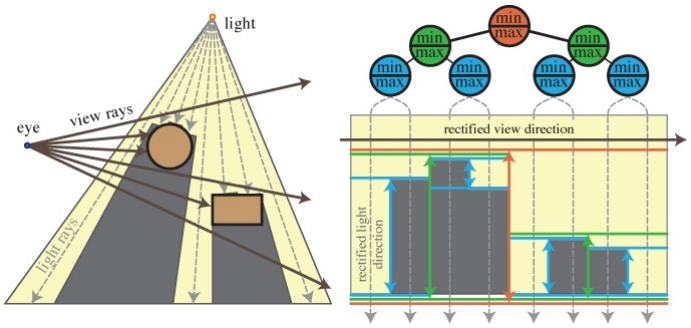


Figure 49: Epipolar Sampling and improved shadow map - Chen et al. (2011)

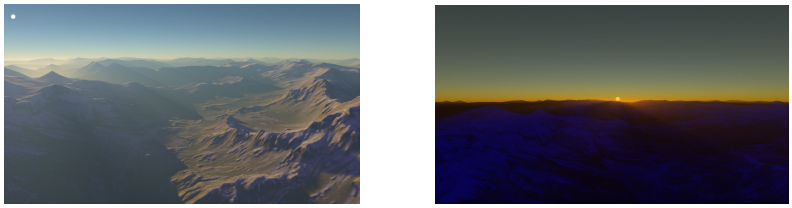


Figure 50: Images of the sky and light shafts using an implementation of Egor Yusov - Yusov (2013)

4.2 CLOUDS

When we look at the sky we usually don't only see the sun and a vast blue mantle, but also clouds. Clouds are an integral feature of the sky and without them synthetic outdoor scenes seem unrealistic (Harris (2002)). They have many shapes and sizes and they result of air in Earth's atmosphere becoming saturated due to cooling and addition of water vapour.

Figure 51: Clouds - "Fir0002/Flagstaffotos"

Luke Howard was the first to present a nomenclature system for clouds in 1802 (Wikipedia (2014h)). From that moment on, several researches were made to understand all the aspects behind cloud formation, radiometry, dynamics and so on.

The computer graphics community have always had an interest in rendering clouds. The rendering of clouds generated a huge amount of publications on the subject. Different approaches were made to address this issue from modelling the clouds into physically based animation and illumination.

It is possible to divide the cloud rendering algorithms in 3 parts: modelling, illumination and dynamics.

For cloud modelling, there were many proposed algorithms such as clouds based of particle systems, metaballs/blobs structures (`figure` 52), voxel volumes and clouds made from procedural noise and textured solids.

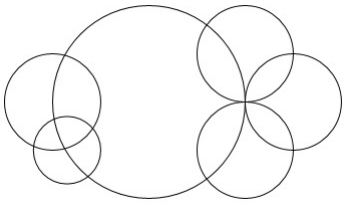


Figure 52: Scheme of a cloud using metaballs

Clouds based in particle systems, present in Reeves (1983) and Harris and Lastra (2001), are systems where the particles represent volumes and they possess some attributes like radius, colour and texture. The clouds are made of a set of these particles.

Other models are based in metaballs/blobs and ray marching. These blobs represent volumes with centre and radius that may or may not overlap (Nishita et al. (1996b)).

Some models use voxel volumes/grids to store the information of the clouds positioning (Kajiya and Von Herzen (1984) and Harris et al. (2003)). Textured solids result from the application of noise on the surface of a solid like an ellipsoid (Gardner (1985)).

Some of these models use procedural noise (Perlin (1985)) to model clouds density.

To simulate cloud dynamics there were 2 different methods: physical methods that solved equations of motion, fluid and thermodynamics, and simple methods based in simpler assumptions.

`Figure` 53 shows the action of wind on a cloud (in this case just cloud shift).

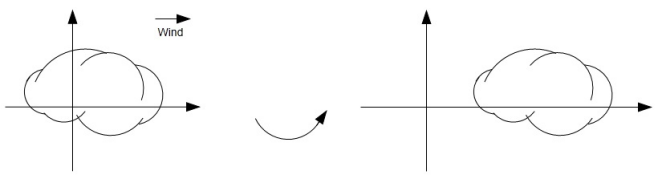


Figure 53: Action of wind

On the early models there was a concern on the memory usage in order to render clouds. Another concern was the time required to render the clouds. That fact led to the usage of algorithms based

in assumptions and analytical expressions that could get realistic but achieving those results in a fair amount of time.

An example of this is the model proposed by Dobashi et al. in 2000 Dobashi et al. (2000). In their work, they proposed a model of generating and moving clouds according to cellular automata. The space was divided in voxels, and in each voxel there was a logical system capable of regulating the generation and dissolution of cloud cells. This system is based in attributing a series of variables to each cell like, humidity, temperature, cloud, extinction and so on. These variables have values of 0 or 1 and they are changed according to probabilities.

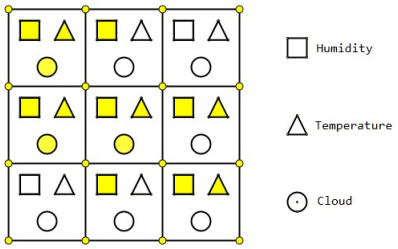


Figure 54: Voxel system of Dobashi et al.

This means that the cell has to pass by some stages before being a cloud cell, i.e, to form a cloud cell, first it must have enough water vapor and humidity. Then if the temperature becomes cold enough, then the water vapor begins to condensate. After this process the cloud cell is formed.

In `figure` 54 is possible to see a representation of Dobashi et al. system. The logical variables, humidity and water vapor (square), temperature (triangle) and cloud (circle), are regulated using random generated numbers, i.e, in each cell, for both humidity, temperature and cloud there is associated a variable (between 0 and 1) that has the value of probability of that state becoming active (coloured symbols in `figure` 54). So the process can be seen as follows:

1. The cell starts with all logical variables equal to zero, which means that there are no clouds
2. In the next frame, for each cell 3 random numbers (humidity, temperature, cloud) are generated and if those numbers are larger than the cells variables, then the cell change its state according to the rules:
   - If the cell as not the conditions of humidity required, than none of the other conditions can be valid (top cell on the right)
   - If the humidity conditions are valid (random number for humidity is larger than the value for humidity in the cell), then the cell becomes in the state humidity (cell with square filled - middle bottom cell)
   - If the cell is in the humidity state and if the conditions of temperature are valid (cold enough to condensate) then the cell passes to the stage temperature (cell with filled triangle and square - bottom right cell)

- If the cell is in the temperature state and if the conditions to create a cloud cell becomes valid, than the cell passes to the state cloud (cell with filled completely filled figures - cell on the top left)

Regarding motion, Dobashi et al. simulates wind effect by using a vector that moves the cells information into neighbouring cells according to the vector direction and magnitude. This is very simplistic but it is very fast to calculate.

With the evolution of graphics hardware, it was possible to use more physically based method. Hence, in 2003 Harris et al. (2003) solves a series of equations to simulate cloud moving and creation like the Navier-Stokes equations (Wikipedia (2014i)). Like in Dobashi et al. (2000), Harris et al. also divides the space in a voxel grid but for each cell it computes the equations of motion, the thermodynamic equation, and the water continuity equations. The result of these is a grid with the information of the voxels that are cloud cells.

Another work presented in 2014, more concretely, the work in Duarte et al. (2014), presented a method to generate and create clouds from real weather forecast data, using diagrams. Instead of solving the Navier-Stokes equations like in Harris et al. (2003), (which according to Duarte et al. (2014) consume a huge amount of resources) they propose to use SkewT/LogP diagrams (weather service (1979)). These diagrams allow to determine 3 fundamental parameters in the cloud generation process: temperature, condensation and equilibrium level. By evaluating these 3 parameters is possible to simulate cloud generation and movement along the atmosphere.

To evaluate these values is first necessary to create the diagrams and for that, values gathered by weather balloons are posted in websites like "http://www.woweather.com", will feed information and therefore create the SkewT/LogP diagrams.

For each of the 3 parameters (temperature, condensation and equilibrium level), the diagram will represent 3 lines relating amtmosphere temperature with pressure. By analysing a particle temperature and pressure conditions is possible to determine its state, i.e., if the particle is, or not, in a cloud state.

These diagrams are used to simulate the cloud in 2 steps: cloud particle generation and ascension.

The particle generation occurs when the value of temperature at ground level reaches a certain limit (visible in the SkewT/LogP diagram), particles start to be generated. Then the particle ascend and its temperature and pressure starts to diminish. When the value of temperature and pressure of the particle reaches the ambient temperature (also visible in the SkewT/LogP diagram) the particle starts to condensate and passes to a water vapour state. Afterwards, the particle continues to ascend until it reaches a state where it does not move vertically. That happens when the particle reach the equilibrium level. With this the particle reaches the cloud state.

To generate a cloud, several particles are generated. The usage of these diagrams allow to perform a faster simulation of the cloud generation than using complex equations.

For cloud illumination most models use light scattering equations. When computing atmospheric scattering, it is possible to use only single scattering to evaluate the colour of the sky with a certain amount of quality. With clouds that is not possible, because the density of atmospheric particles is

much smaller than the density of the particles inside the clouds. This density requires that, to evaluate the colour of the clouds, it is necessary to compute multiple events of light scattering (`figure` 55).

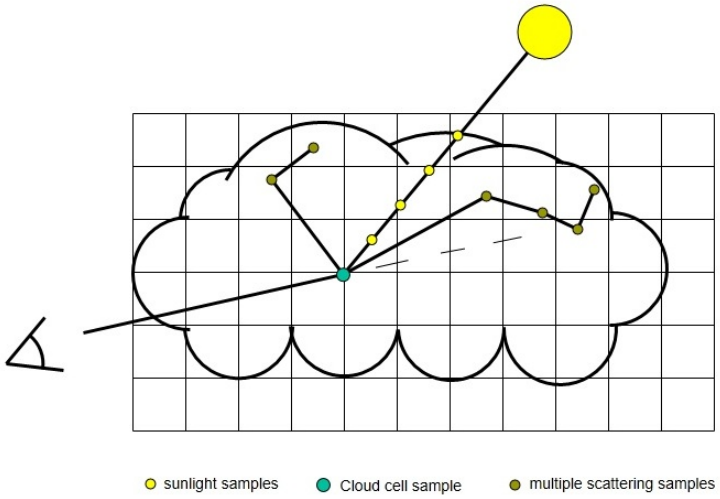


Figure 55: Illumination of a cloud using metaballs and voxel grid

To solve this problem it is necessary to compute the atmospheric scattering effect. The common approach is to use analytical expression (basis functions and spherical harmonics) or ray marching with sampling.

Some use ray marching to evaluate single scattering and get multiple scattering by using analytical expressions Kajiya and Von Herzen (1984). Other models like Nishita et al. (1996b) get single and multiple scattering by gathering in a voxel grid, the values of scattering (multi-pass algorithm).

In some more recent methods like Harris et al. (2003) and Bouthors et al. (2008), the problem of multiple scattering is simplified by reducing the direction of the scattering events to only a forward direction (scattering can deflect the photons to any direction).

In 2014, Yusov presented an algorithm for cloud rendering where the problem of cloud illumination is solved by pre-computing the values of light scattering and attenuation and storing them in a set of textures (Yusov (2014)).

In this model, the space is divided in a 3D grid and each cell of the grid has a value of density, where cloud cells will have bigger values of density than other cells. Yusov generates particles in the grid and attributes value of density according to a composition of 3D noise functions.

Using the idea present in Bruneton and Neyret (2008), Yusov pre-computed the values of multiple-scattering and stored them in a 4D table, with the difference of the table considering not only the height of the camera, the view and sunlight direction, but also a parametrised value of the particle density.

With this, the final value of colour for each view ray is obtained by doing lookups in the 4D table according to the values of density in the 3D grid.

The algorithms presented above are not mutually exclusive. Most of the work done on cloud rendering was an attempt to obtain the best possible trade-off between quality and rendering time by using a mix of these algorithms.

As an example of cloud rendering, lets focus on the algorithm in Yusov (2014). and explain the normal steps taken into the process to obtain the colour of the cloud.

The mathematical expression to obtain the colour of the cloud, with the camera in a position P_{cam} in a view direction v is very similar with the expression in `equation` 23 with multiple scattering (att_{sunMS} em vez de att_{sun} - `equation` 25). The major change remains in the evaluation of the cloud particles, i.e, the interaction of the sunlight with the other particles in the atmosphere its still used both Rayleigh and (in this case) Henyey-Greenstein phase functions, but for cloud particles, only the phase function of Henyey-Greenstein is used. This happens because cloud particles are considered as large particles.

With this in mind a possible expression to represent the value of light intensity reaching the camera in a view direction v can be given by `equation` 73:

$$L(v) = L_{in}(v) + L_{cloud}(v) + att_{cloud} att_{in} L_{bg}(v) \tag{73}$$

where $L_{in}(v)$ represents the amount of light that is in-scattered due to particles on the atmosphere between the camera and the cloud, $L_{cloud}(v)$ represents the amount of in-scattered light coming from inside the cloud, $L_{bg}(v)$ represents the amount of in-scattered light behind the cloud particles and that is attenuated (attenuation computed with the attenuation due to the cloud particles (att_{cloud}) and the atmosphere particles (att_{in}))

Expressions L_{in} and L_{bg} are given by `equation` 23, and L_{cloud} is given by `equation` 22 with Henyey-Greenstein phase function.

As mentioned before, Yusov idea passes by pre-computing the value of single and multiple scattering for every view and sunlight direction not only for the normal atmospheric scattering (L_{in} and L_{bg}) but also, pre-compute values for a cloud particle (L_{cloud}). For matters of simplicity lets represent the pre-computed table for atmospheric scattering as $table_{scatt}$ and the pre-computed table for clouds with $table_{cloud}$.

It was also mentioned that the clouds are modelled with spherical volumes positioned in the atmosphere. These spherical volumes possess a centre ($c_{x,y,z}$) and a radius r. In order to be able to pre-compute the optical depth inside these volumes, the density in the volumes is parametrised according to the distance to the centre of the volume. So, the pre-computed 4D table consists in the calculations for every view direction ((θ_v, ϕ_v)), sunlight direction (θ_s) and parametrised density.

To understand the render process lets focus on `figure` 56.

The render process consists in for every view ray v, intersect the ray with the cloud particles, and if the ray intersects with a cloud, obtain the point of entry (P_{entry}) and exit (P_{exit}) of the view ray in the cloud. With this information perform 2 lookups on $table_{scatt}$ to obtain L_{in} and L_{bg} according to a light direction l_{dir} ($L_{in} = lookup(P_{cam}, P_{entry}, l_{dir})$ and $L_{bg} = lookup(P_{exit}, P_{end}, l_{dir})$). Perform another

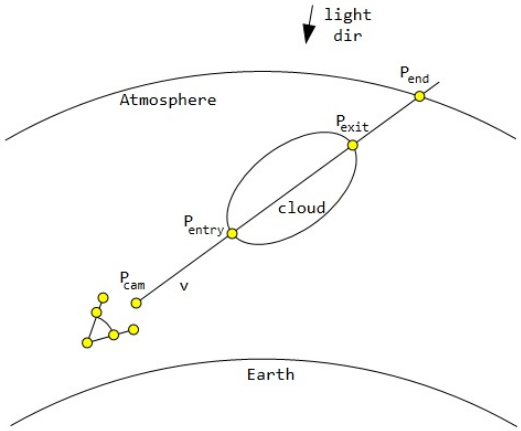


Figure 56: Cloud illumination

lookup on $table_{cloud}$ to obtain L_{cloud} ($L_{cloud} = lookup(P_{entry}, P_{exit}, l_{dir}, d)$, where d is the value of density inside the cloud particle).

To obtain the values of attenuation att_{cloud} and att_{in}, another auxiliary tables are used to store the pre-computed values for optical depth (for cloud and atmospherical scattering).

This model can produce results like the ones in `figure` 57

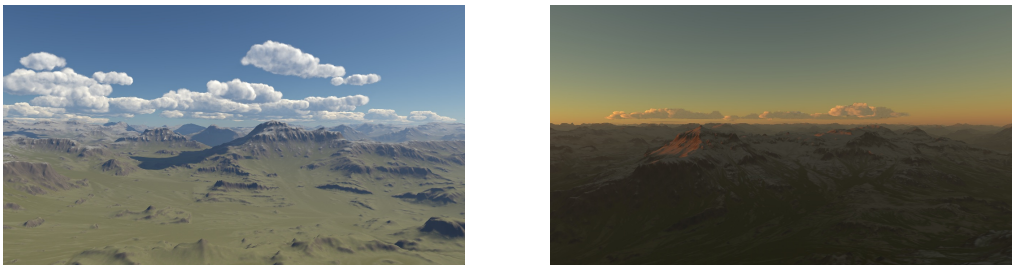


Figure 57: Images of clouds using Yusov model - Yusov (2014)

Some other aspects concerning efficiency in this model can be further explored in Yusov (2014).

4.3 FOG

Fog appears when the particle density in the atmosphere become higher. Fog reduces our vision range because the higher density of particles will cause the scattering away of all the coming from objects (`figure` 58).

The majority of the methods proposed to render fog derive from methods of computing atmospheric scattering, so most of them compute the fog effect with light scattering equations.

Figure 58: Fog in the landscape

The early attempts in rendering fog were very simple. The fog was considered a layer/component with constant density and was rendered with ray marching with light scattering equations. An example is in Klassen (1987) and Kaneda et al. (1991).

The assumptions made were an extension of the rendering method for atmospheric scattering. Layers were adjusted, or a layer with higher density was added. The higher density would cause a greater attenuation of light. By controlling the density values it was possible to control the intensity of the fog.

In 1999, Preetham et al. proposes an analytical method based on a value of turbidity (Preetham et al. (1999)). This value would control the optical thickness of the medium and generate the density of the atmosphere.

The problem of these early models were the assumption of fog as a layer with constant density and with a global range. In fact, fogs are volumes near surface with higher and non-homogeneous particle density (`figure` 59).

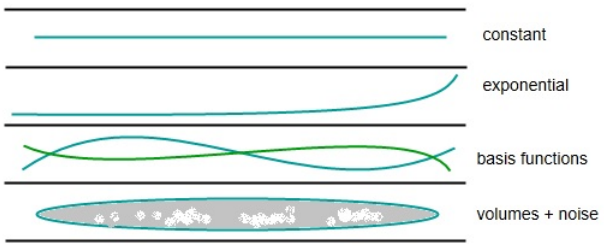


Figure 59: Fog density evaluation

Some researchers addressed this issue, namely in Biri et al. (2002) were the fog was modelled by a set of cosine functions. Other example of usage of basis functions can be seen in Giroud and Biri

(2010), were the basis function are used to draw the area of the volume as a fog map. That fog map would have different values that would correspond to different densities inside the fog.

Another mechanism to implement fog volumes was proposed by Wenzel (2006). These volumes would be as simples as boxes or ellipsoids, but the density of the fog inside the volumes would be modelled with 3D noise.

In Wronski (2014) there was a proposal to render volumetric fog. The fog would be modelled with perlin noise functions and vertical attenuation. This information would be stored in a grid aligned with the camera frustum (up, right, depth) so that the gathering of light would be just a 2D ray marching in that grid (marching in depth).

So in a certain perspective, fog can be modelled by adjusting the atmosphere density and its possible to obtain fog by using the AS models presented in `section` 1.8.

However, there is an interest in rendering the localised phenomenon. This localisation brings also something important that is, the majority of the particles in the fog can considered as large, which means that this variation in density is not only localised, but also of a specific type of particles (in this case the increase of aerosol particles). The immediate consequence is the increase of importance of the Mie scattering equations (although many models use the function of Henyey-Greenstein).

An example of rendering foggy scenes using the method in Wronski (2014), can be found in `figure` 60

Figure 60: Images of clouds using Wronski model - Wronski (2014)

The process to render foggy scenes using scattering equations is very similar to the ones described along `chapter` 3.

4.4 RAIN

Rain is a complex phenomenon where water particles drop from clouds and descend to Earth's surface. The process to render such a phenomenon accurately is complex.

Along the years several algorithms were proposed to render the various effects of rain, specially rainfall (`figure` 61). Other effects like rain droplet splashing were also studied.

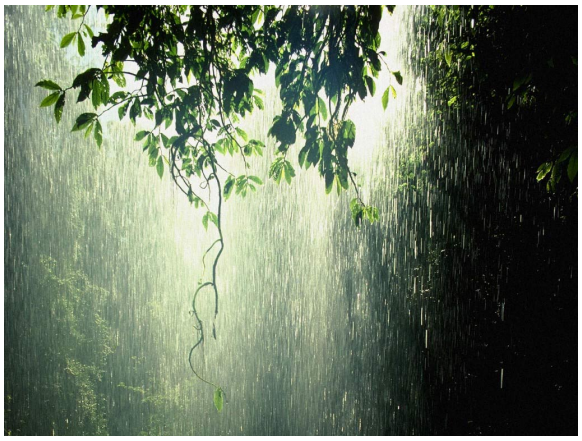


Figure 61: Rain

Some of the most important work on rain rendering appeared in 2006 with the work of Garg et al. Garg and Nayar (2006) and Garg and Nayar (2007). In their works they proposed methods to render rainfall and splash distribution.

The method in Garg and Nayar (2006) was off-rendering where they take in videos with information of depth and gathered rain drops from other images and them implement them in the videos.

However these methods were not applied in real-time. In 2006 Wang et al. and using particle systems they proposed a method to render realistic rain. Each particle consists of a rain stroke matte that is randomly chosen from rain stroke samples, a sphere model, and attributes such as velocity Wang et al. (2006).

In 2008 Changbo et al. proposed a method to simulate rainfall by generating a series of rain streaks made from previously elaborated models.

According to them, rain streaks are not only lines, but the product of the oscillations that are caused by the aerodynamic forces and the surface tension acting on the water droplet Changbo et al. (2008). So based on real rain streaks, they generated different rain streaks shapes and store them in textures. These different streaks are then associated to rain particles and blended to the scene.

In 2010 Lopez (Lopez (2010)) studied the methods of rain rendering and proposed a new method that consisted in using particle systems. His algorithm had 2 stages. A pre-computed stage were the information of the rain would be generated. This information consists in the zones of the particle generation and the shapes of the rain streaks.

The zone of origin of the particles consist in setting a spacial reference element, for example an horizontal oriented quad, and in that reference set a rain particle generation probability modelled or by a function or by a texture.

In other words, for each sample point in that reference generate a random number, and if that number surpasses a certain value, then generate a rain particle.

According to Lopez, this generation could also be simulated for the scene and in this case, a groups of particles that would be generated at the same time would be set in a pack (figure 62). These packs would have information about the velocity of descent, vector of descent, and would be rendered at the same time.

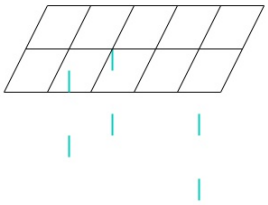


Figure 62: Rain particle generation

The second pass would be the generation of particles according to the pre-computed information and rendered using blending functions. The rain streak packets will be represented as billboards, i.e, for each packet particle, evaluate the velocity of descent and vector and fetch the correspondent rain streak.

The method proposed takes into account occlusion, so that the particles behind an object are not rendered.

To better understand the process to render rain streaks, lets focus on the model proposed by Lopez:

1. Generate the spaces were the rain will be generated. For example place a texture (with values between 1 and 0) horizontally oriented in world space
2. For each time t of the simulation, traverse the texture in step 1 and generate for each sample a random number between 0 and 1. If the value is higher than the value on the texture generate particles and compute their position over a 3D grid with associated values of velocity and length
3. for each group of particles generated in each time t with the same characteristics associate to a packet
4. In run-time generate the packets according to the simulation:
    a) In the vertex shader compute the positions of the packets on the world space and exclude the ones out of the space generated in step 1 and that are occluded by any object or landscape (camera depth buffer test)
    b) In the geometry shader for each particle passed by the vertex shader associate a billboard with the rain streak (line with a certain length according to the values of velocity and length of the particles)
    c) In the fragment shader blend the rain streak into the scene

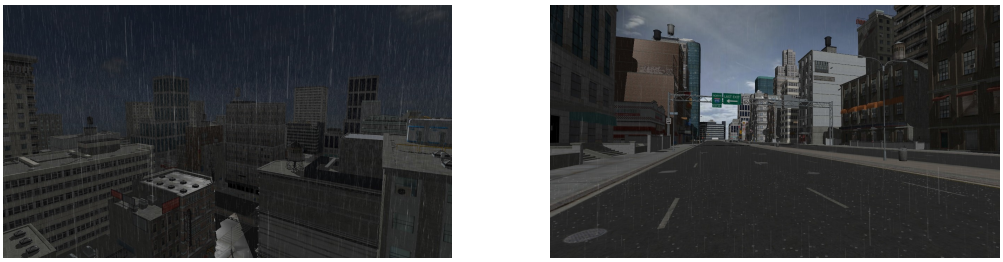


Figure 63: Images of rain using Lopez model - Lopez (2010)

An example of the results using this method can be seen in figure 63.

4.5 RAINBOWS

After rain, sometimes is possible to see a natural colourful phenomenom - rainbows. The rainbow phenomenon has always fascinated people, so its inclusion and rendering in graphical contexts is important.

Figure 64: Rainbow

Some models presented to render rainbows can also reproduce fogbows. The difference between rainbows and fogbows is that instead of rendering colourful bows due to rain, fogbows are rendered due to fog. From the several works presented about the subject there were some that attempted to render rainbows using texture based mechanisms (nVidia (2004)). These mechanism were based on using texture colour lookup using software MiePlot (Laven (2014)) to produce the values of colour according to scattering angle and water droplet radius.

To render the rainbows it is necessary to perform a lookup the colour in the texture and blend it into the scene (figure 65). In nVidia (2004) the blend is done according to a moisture factor that is set by the scene.

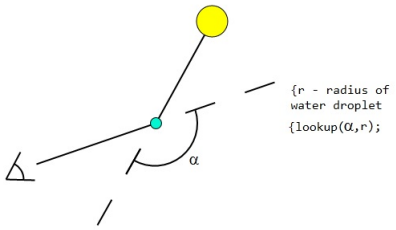


Figure 65: Rainbow lookup

Frisvad et al. (2007) proposed to use an even simpler method based on a theory by Aristotle in approximately 350 B.C. Aristotle considered the sky as a hemisphere and that rainbows appear at a certain angle between the sunlight direction and the view direction. With this assumption the colours of the rainbow are rendered in a sky dome.

In 2011 Sadeghi et al. proposed a physically based model to render rainbows (Sadeghi et al. (2011)). Their proposal relies in simulating accurate phase functions for rainbows instead of using the phase functions of the Mie theory.

Their argument is that water droplets are not spherical and, as Mie's theory only considers spherical particles, then it is more accurate to use a phase function that takes into account the geometrical shape of a water droplet.

So, to generate a new phase function, Sadeghi et al. module a series of object with different possible water droplet shapes. To each one of those objects trace a series of parallel rays (in order to simulate a directional light source), and for each one of these rays trace their path through a n number of interactions (reflection and refraction) and collect the exit ray in an outer sphere.

This outer-sphere will have the general distribution of the light behaviour when interacting with the water droplet. This distribution analysis will allow to module a function that according to an angle (relation between a direction of light and point of exit) will give the distribution of light, just like a normal phase function.

The simulation itself takes a huge amount of time because its necessary to do it for each wavelength (in Sadeghi et al. (2011) they spent 350 minutes for 33 wavelengths). On the other and this phase function generation allows to compute the behaviour for different types of water droplets.

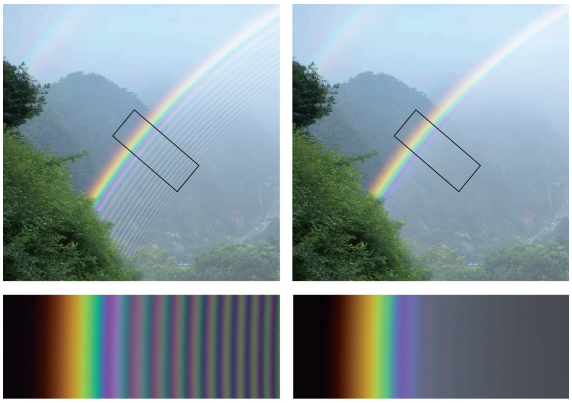
As an example of rainbow rendering lets focus on the model in Nvidia's white paper (nVidia (2004)).

The first step consists in using a software like "Mieplot" to compute to a texture the values of colour according to scattering angle and water droplet radius.

MiePlot generates Lee diagrams (Lee (1998)) and these diagrams show how the colour of a rainbow changes as the radius of a water droplet changes. Lee diagrams are 2D images in which each pixel represents the colour of scattered light corresponding to a given angle of deviation (scattering angle) on one axis and radius of water droplet on the other axis (nVidia (2004)).

For each view direction compute the angle of scattering and get the value of colour of the texture obtained in the first step(depending on the water droplet radius considered in the scene). Then, compute the scene colours and blend the value of step 2 in the scene.

The models described before, for example, the model in Sadeghi et al. (2011) can produce results like the ones in figure 66.



(a) (b)

Figure 66: Images of rainbow with Sadeghi et al. model comparing the usage of Mie theory(a) and the new method(b) - Sadeghi et al. (2011)

4.6 LIGHTNING

Lightning is a sudden flow of electricity or electrostatic discharge that occurs during an electric storm. Lightning occurs when there is a high electric potential and a high resistant medium (figure 67). Like other natural phenomena, there are attempts to reproduce this effect in CG.

Figure 67: Lightning

One of the first attempts to generate a lightning model was proposed by Niemeyer et al. in 1984, where they analysed the fractal dimension of various types of dielectric breakdown, including lightning (Niemeyer et al. (1984)). To model the path of the lightning Niemeyer et al. uses the Laplace equation (Wikipedia (2014f)) to assign probability of the path of the lightning branches on a mesh (figure 68).

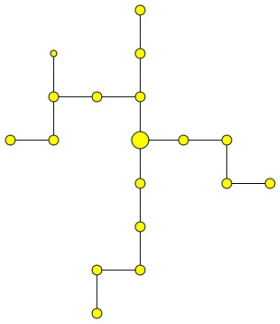


Figure 68: Lightning shape using a grid

In 2004 Kim et al., proposed a method to render lightning based on the model of Niemeyer et al. (1984) (Kim and Lin (2004)). They generate the paths of the lightning and then use APSF (Atmospheric Point Spread Function) to evaluate the amount of light that flows in each of those paths. APSF are a particular type of point spread functions (PSF Wikipedia (2014j)) that are related to the atmosphere, i.e., APSF are functions that describe the response of light distribution in a medium, in this case atmosphere, around a particular point of light. In Kim and Lin (2004), the APSF are a series expansion of the Henyey-Greenstein phase function.

The paths are rendered using lines and the values of light intensity are used for a glow effect in order to turn the paths more similar to lightning.

In Kim and Lin (2007), Kim et al. improved the method of render by using octrees in order to simulate the 3D path generation of the lightning. Kim et al. proposed to solve the path generation and solving the Laplace equations by using balanced octrees. This structure allowed to obtain speed ups in an order of magnitude of 2.

In LaPointe and Stiert (2009), they pointed out that the rendering process in Kim and Lin (2004) and Kim and Lin (2007), presents artifacts and a flat appearance.

To better understand out to render lightning lets focus on the model of LaPointe and Stiert (2009).

LaPointe and Stiert, proposed an algorithm of 2 main steps: lightning shape and colour.

In a first step they use L-Systems (Wikipedia (2014d)) to simulate the shape and create a volumetric lightning. This step consists in taking in a line segment S and using the L-System's rules, subdivide the segment at the midpoint and displaced at an angle perpendicular to itself with a magnitude M (figure 69).

This process is repeated for both line segments in S' until the number of levels of detail, n, is reached. According to LaPointe and Stiert, on each iteration, the maximum random magnitude that M

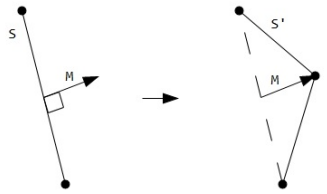


Figure 69: Lightning shape using a l-system

can be is halved. This origins a single line with a series of deformations. But if we look at lightning they usually have several branches.

To do that, they applied in the subdivision step, a probability of a new branch being created (in LaPointe and Stiert (2009) the probability is about 0.5). An example of this new branch can be seen in `figure` 70.

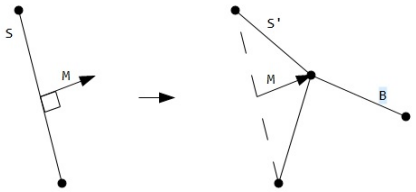


Figure 70: Lightning branch

The line segments are then stored in a volume buffer.

The second step consists in give to each segment a value of colour and intensity. The colour is usually the same for all segments (in LaPointe and Stiert (2009) they use $colour = (0.5, 0.5, 1.0, 1.0)$). For intensity, the main segment S has the highest value of intensity I ($I = 1$) and for each branch the value is halved, i.e., 1-level branches will have $I = 0.5$, $2 - level$ branches will have $I = 0.25$ and so on.

The values of intensity are then brightened to give a glow effect to the lightning.

To render this structure, in LaPointe and Stiert (2009) they put the lightning inside an axis aligned bounding box, and use a ray-tracing algorithm to intersect view rays with the lightning structure. Depending on the distance the value of colour is attenuated according to a δ.

An example of the results with this method can be seen in `figure` 71.

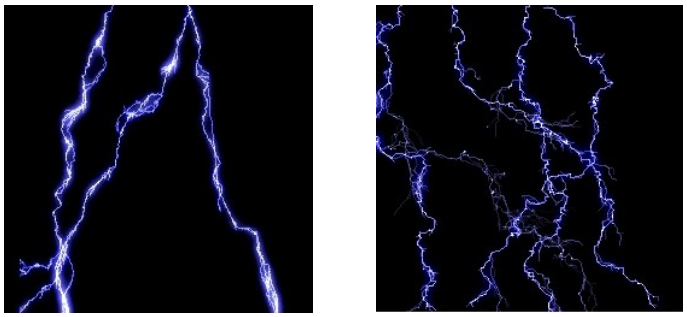


Figure 71: Images of simple (left) and multiple (right) lightnings using the method of LaPoint and Stiert - LaPointe and Stiert (2009)

5

CONCLUSION

The natural phenomenon of atmospheric scattering was thoroughly studied along the years. A large set of models were proposed to recreate, among other natural effects, the beautiful colours of the sky.

All AS models had the concern to compute these effects with most efficiency and quality as possible. The core of these models was the mechanism used to solve the scattering equations presented by Lord Rayleigh in the late 19^{th} century and Gustav Mie in the early 20^{th} century.

The atmospheric models evolution showed a path that went from quality to efficiency.

The early AS models, like the ones presented by Klassen in 1987, Kaneda et al. in 1991 and Nishita et al. in 1993, had a greater concern in recreating with the most possible accuracy the atmospheric scattering phenomenon.

That concern passed, over the years, to recreate this phenomenon the fastest way possible. With this appeared the AS models presented by Preetham et al. in 1999, followed by the GPU models of O'Neil in 2005 and, Bruneton and Neyret in 2008.

Despite the improvement in efficiency, the modern AS models had the concern of speeding up the process of rendering without loosing quality.

With respect to the quality of the results, one thing that can be concluded is that when comparing analytical and physically based models, the differences, in some circumstances are so tiny, that is difficult to perceive differences. One example of this can be seen in the comparison of the results obtained using the physically based algorithm in Nishita et al. (1993) (`figure` 23) and the analytical simplification made in O'Neil (2005) (`figure` 35).

When talking about rendering the colours of the sky, is natural to want to render the other natural effects such as clouds and light shafts.

So, along side the evolution of the AS models, there was also an evolution on the algorithms that could recreate clouds and shadows. For example, on the light shafts, there was an evolution on the shadow map/volumes techniques, on the fog there was an evolution in the modelling of local densities.

The possibility of rendering the colours of the sky in real-time, would be almost unthinkable in 1987, when Klassen presented his AS model. The computers capacity at that time was very limited. This is also a factor to take into account: the models/algorithms have also evolved along side with the computers hardware and software.

Another important thing to mention is that the information about the atmospheric scattering models, and the algorithms behind light shafts, clouds and lightning, is huge and at the same time is highly dispersed.

So there is an interest in the analysis and evaluation of these methods and algorithms, because many different and creative methods were presented in these models and there is a possibility in the future that these different methods may be used together and create a better model to render the colours of the sky. For example, there might be a way to compute a complex dynamic using these methods that could allow to simulate in real-time something like the appearance of clouds that will lead to rain and consequent rainbow formation.

One interest thing that appeared in some of the models analysed in this project concerns the usage of real data to create and evaluate atmospheric phenomenons. The model in Duarte et al. (2014) uses data coming from websites like the values of atmosphere pressure and temperature and uses them to simulate cloud generation.

This can be extended to other atmospheric scattering models, i.e., it was an interesting idea to use real-time data like atmospheric particle density, temperature and intensity of incoming light to calculate in a certain place and time (on Earth) the colours of the sun and sky.

The atmospheric scattering models allowed the creation and improvement of a series of applications that goes from flight simulators, that are useful tools in pilot training, to applications like video-games and films.

BIBLIOGRAPHY

(1995). Spatial distribution of daylight-luminance distributions of various reference skies, cie publication 110, central bureau of the cie, vienna, austria, 1994, softbound, pp. 33, $51(members)/77$ (non-members). *Color Research & Application*, 20(1):80–81.

Beer (1852). Bestimmung der absorption des rothen lichts in farbigen flüssigkeiten. *Annalen der Physik*, 162(5):78–88.

Biri, V., Michelin, S., and Arques, D. (2002). Real-time animation of realistic fog.

Bouthors, A., Neyret, F., Max, N., Bruneton, E., and Crassin, C. (2008). Interactive multiple anisotropic scattering in clouds.

Box, M. A. (1983). Power series expansion of the mie scattering phase function. *Australian Journal of Physics*.

Bruneton, E. and Neyret, F. (2008). Precomputed atmospheric scattering. *Comput. Graph. Forum*, 27(4):1079–1086.

Bucholt, A. (1995). Rayleigh-scattering calculations for the terrestrial atmosphere.

Cabral, B., Max, N., and Springmeyer, R. (1987). Bidirectional reflection functions from surface bump maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 273–281, New York, NY, USA. ACM.

Changbo, W., Wang, Z., Zhang, X., Huang, L., Yang, Z., and Peng, Q. (2008). Real-time modeling and rendering of raining scenes. *Vis. Comput.*, 24(7):605–616.

Chen, J., Baran, I., Durand, F., and Jarosz, W. (2011). Real-time volumetric shadows using 1d min-max mipmaps.

Cornette, W. M. and Shanks, J. G. (1992). Physically reasonable analytic expression for the single-scattering phase function. *Appl. Opt.*, 31(16):3152–3160.

C.P., D. (2012). Dust scattering off particles. *Universität Heidelberg*.

Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., and Nishita, T. (2000). A simple, efficient method for realistic animation of clouds. pages 19–28.

Dobashi, Y., NISHITA, T., KANEDA, K., and YAMASHITA, H. (1997). A fast display method of sky colour using basis functions. *The Journal of Visualization and Computer Animation*, 8(2):115–127.

Dobashi, Y., Yamamoto, T., and Nishita, T. (2002). Interactive rendering of atmospheric scattering effects using graphics hardware. pages 99–107.

Duarte, R. P. M., Morgado, J. F. M., and Gomes, A. (2014). Visualização dinâmica de nuvens através de diagramas termodinâmicos. *Livro de atas do 21º encontro português de computação gráfica*.

Engelhardt, T. and Dachsbacher, C. (2010). Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. pages 119–125.

Frisvad, J. R., Christensen, N. J., and Falster, P. (2007). The aristotelian rainbow: from philosophy to computer graphics. pages 119–128.

Gardner, G. Y. (1985). Visual simulation of clouds. pages 297–303.

Garg, K. and Nayar, S. K. (2006). Photorealistic rendering of rain streaks. *ACM Trans. Graph.*, 25(3):996–1002.

Garg, K. and Nayar, S. K. (2007). Vision and rain. *Int. J. Comput. Vision*, 75(1):3–27.

Giroud, A. and Biri, V. (2010). Modeling and rendering heterogeneous fog in real-time using b-spline wavelets.

Gordon, H. R. (1973). Simple calculation of the diffuse reflectance of the ocean. *Appl. Opt.*, 12(12):2803–2804.

Green, R. (2003a). Spherical harmonic lighting: The gritty details. [Online; accessed 24-November-2014].

Green, R. (2003b). Spherical Harmonic Lighting: The Gritty Details. *Archives of the Game Developers Conference*.

Hahn, D. W. (2009). Light scattering theory.

Harris, M. J. (2002). Real-time cloud rendering for games. pages 21–29.

Harris, M. J., Baxter, W. V., Scheuermann, T., and Lastra, A. (2003). Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '03, pages 92–101, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Harris, M. J. and Lastra, A. (2001). Real-time cloud rendering.

Hosek, L. and Wilkie, A. (2012). An analytic model for full spectral sky-dome radiance. *ACM Trans. Graph.*, 31(4):95:1–95:9.

Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18(3):165–174.

Kaneda, K., Okamoto, T., Nakamae, E., and Nishita, T. (1991). Photorealistic image synthesis for outdoor scenery under various atmospheric conditions. *The Visual Computer*, 7(5-6):247–258.

Kim, T. and Lin, M. C. (2004). Physically based animation and rendering of lightning. pages 267–275.

Kim, T. and Lin, M. C. (2007). Fast animation of lightning using an adaptive mesh. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):390–402.

Klassen, R. V. (1987). Modeling the effect of the atmosphere on light. *ACM Trans. Graph.*, 6(3):215–237.

Lambert, J. H. (1760). *I.H. Lambert Photometria, sive, De mensura et gradibus luminis, colorum et umbrae [microform]*. V.E. Klett Augustae Vindelicorum.

LaPointe, C. and Stiert, D. (2009). Volume lightning rendering and generation using l-systems.

Laven, P. (2014). Mieplot. [Online; accessed 10-July-2014].

Lee, R. L. (1998). Mie theory, airy theory, and the natural rainbow. *Appl. Opt.*, 37(9):1506–1519.

Lopez, C. C. (2010). Real-time realistic rain rendering.

McCluney, W. and Center, G. S. F. (1974). *Ocean Color Spectrum Calculations*. Goddard Space Flight Center.

Mie, G. (1908). Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Ann. Phys.*, 330(3):377–445.

Niemeyer, L., Pietronero, L., and Wiesmann, H. J. (1984). Fractal dimension of dielectric breakdown. *Phys. Rev. Lett.*, 52:1033–1036.

Nishita, T., Dobashi, Y., Kaneda, K., and Yamashita, H. (1996a). Display method of the sky color taking into account multiple scattering.

Nishita, T., Dobashi, Y., and Nakamae, E. (1996b). Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 379–386, New York, NY, USA. ACM.

Nishita, T., Sirai, T., Tadamura, K., and Nakamae, E. (1993). Display of the earth taking into account atmospheric scattering. pages 175–182.

nVidia (2004). Rainbows and fogbows: adding natural phenomena, sdk white paper.

on Illumination CIE, I. C. (2004). Spatial distribution of daylight - cie standard general sky.

O'Neil, S. (2004). Real-time atmospheric scattering.

O'Neil, S. (2005). Accurate atmospheric scattering.

Oxford-Dictionaries (2014). light-scattering. [Online; accessed 20-October-2014].

Perez, R., Seals, R., and Michalsky, J. (1993). All-weather model for sky luminance distribution—preliminary configuration and validation. *Solar Energy*, 50(3):235 – 245.

Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296.

Prast, S. and Fruehstueck, A. (2013). Caustics, light shafts, god rays.

Preetham, A. J., Shirley, P., and Smits, B. (1999). A practical analytic model for daylight. pages 91–100.

Rayleigh, J. (1871a). On the light from the sky, its polarization and colour. *Philosophical Magazine Series 4*, 41:107–120, 274–279.

Rayleigh, J. (1871b). On the scattering of light by small particles. *Philosophical Magazine Series 4*, 41:447–454.

Rayleigh, J. (1881). On the electromagnetic theory of light. *Philosophical Magazine Series 5*, 12:81–101.

Rayleigh, J. (1889). On the transmission of light through an atmosphere containing small particles in suspension, and on the origin of the blue of the sky. *Philosophical Magazine Series 5*, 47:375–394.

Reeves, W. T. (1983). Particle systems—a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108.

Sadeghi, I., Muñoz, A., Laven, P., Jarosz, W., Seron, F., Gutierrez, D., and Jensen, H. W. (2011). Physically-based simulation of rainbows. *ACM Transactions on Graphics (Presented at SIGGRAPH)*, 31(1):3:1–3:12.

Schafhitzel, T., Falk, M., and Ertl, T. (2007). Real-time rendering of planets with atmospheres. *Journal of WSCG*, 15(1-3):91–98.

Van De Hulst, H. (1982). *Light Scattering by Small Particles*. Peter Smith Publisher, Incorporated.

Wang, L., Lin, Z., Fang, T., Yang, X., Yu, X., and Kang, S. B. (2006). Real-time rendering of realistic rain. (MSR-TR-2006-101):18.

weather service, A. (1979). the use of the skew t, log p diagram in analysis and forecasting.

Wenzel, C. (2006). Real-time atmospheric effects in games. pages 113–128.

Wikipedia (2014a). Bessel functions. [Online; accessed 22-September-2014].

Wikipedia (2014b). Bézier curve. [Online; accessed 10-September-2014].

Wikipedia (2014c). Cie xyy colour space. [Online; accessed 10-June-2014].

Wikipedia (2014d). L-system. [Online; accessed 12-September-2014].

Wikipedia (2014e). Lambertian reflectance. [Online; accessed 4-August-2014].

Wikipedia (2014f). Laplace equation. [Online; accessed 23-September-2014].

Wikipedia (2014g). Legendre polynomials. [Online; accessed 22-September-2014].

Wikipedia (2014h). Luke howard. [Online; accessed 20-October-2014].

Wikipedia (2014i). Navier-stokes equations. [Online; accessed 9-October-2014].

Wikipedia (2014j). Point spread function. [Online; accessed 11-September-2014].

Wikipedia (2014k). Ray marching. [Online; accessed 4-August-2014].

Wikipedia (2014l). Scale height — wikipedia, the free encyclopedia. [Online; accessed 24-November-2014].

Wikipedia (2014m). Scattering cross section. [Online; accessed 10-July-2014].

Wikipedia (2014n). Stokes parameters. [Online; accessed 22-October-2014].

Wikipedia (2014o). Sun. [Online; accessed 12-January-2014].

Wronski, B. (2014). Volumetric fog: Unified compute shader based solution to atmospheric scattering.

Yusov, E. (2013). Practical implementation of light scattering effects using epipolar sampling and 1d min/max binary trees.

Yusov, E. (2014). High-Performance Rendering of Realistic Cumulus Clouds Using Pre-computed Lighting. pages 127–136.

Zotti, G., Wilkie, A., and Purgathofer, W. (2007). A critical review of the preetham skylight model. pages 23–30.