UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech
FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

Day and Night Cycle, Development of a Physically Based Real-Time Solution

A Thesis submitted in partial fulfillment of the requirements of
MASTER IN INNOVATION AND RESEARCH IN INFORMATICS
SPECIALIZATION IN COMPUTER GRAPHICS AND VIRTUAL REALITY

by

Guillem Pérez Delgado

Supervised by

Pere-Pau Vázquez

Barcelona, October 2022

Day and Night Cycle

Development of a Physically Based Real-Time Solution

Guillem Pérez Delgado

Facultat Informatica de Barcelona
Universitat Politècnica de Catalunya
Barcelona
October 10, 2022

Abstract

Over the last few years, open world videogames have been gaining lots of interest in the gaming industry. Open world videogames not only allow the player to freely roam over a vast terrain but also aim to recreate a believable dynamic world. Thus, one of the basic elements that such a videogame should feature is a day and night cycle. In this thesis, all of the intricacies that are involved in developing a physically based day and night cycle solution in a real-time rendering context are discussed. The main topics that will be covered are atmosphere rendering, celestial bodies positioning, celestial bodies rendering and nighttime scenes rendering.

Contents

Chapter 1

Introduction

The development of a complete day and night cycle solution encompasses a wide variety of distinct elements. In the past, most of these elements could be successfully achieved by using a wide variety of independent techniques, the appearance of the sky, for example, could be achieved by using artist-created HDR textures or simple color gradients. Even though these previous approaches work well and are able to provide great results, they do not scale well into dynamic environments where lots of manual tweaking is required, especially to obtain consistency between the various elements. It is clear then, that more advanced solutions are required in order to support this dynamicity that an increasing number of videogames demand these days. [Hil16]

As we will see over the course of this thesis, using a physically based approach, it is possible to obtain a complete day and night cycle solution that unifies all the elements it encompasses and requires almost no manual tweaking to obtain consistent results. As well as presenting the theoretical foundations of each of these elements, as part of this thesis, I develop a C++ and OpenGL implementation where all of these come together to create a fully working system that demonstrates the techniques presented. For each of the elements of the solution, it is common to find not one but several solutions/models/approaches to it, hence some decisions have to be taken when developing this implementation. Each of the decisions taken and the reasoning behind them is accordingly justified where appropriate.

It is important to note that the solution being physically based does not imply that there is no room for artistic control, as demonstrated by the fact that techniques similar to those explained in this thesis have been used in videogames of very different styles: from very realistic ones such as *Far Cry 5* [McA18], *Red Dead Redemption 2* [Bau19] or *Ghost of Tsushima* [Pat21] to stylized ones such as *Fortnite* [Hil20]. Figure 1.1 shows two examples of these. Artistic control will be emphasized throughout the thesis in the form of freely controllable parameters.

Figure 1.1: Screenshots of two videogames that feature a physically based day and night cycle. Top: *Red Dead Redemption 2*. Bottom: *Fortnite*.

The goal of this thesis is to develop a system that is able to render scenes similar to the ones shown in Figure 1.1, as well as the dynamic transitions in them. In those screenshots, the two main visual cues that convey the distinct time of day are the appearance of the sky and the celestial bodies that appear in them: the Sun in the *Red Dead Redemption 2* screenshot as it depicts a sunset, the Moon and stars in the *Fortnite* screenshot as it depicts night. These will be then the topics covered in the thesis. More precisely, Chapter 2 shows how such sky colors can be obtained. Regarding celestial bodies, it is not only important to faithfully depict them, a topic which is discussed in Chapter 4, but also to correctly position them in the sky as the time of day passes, which is discussed in Chapter 3. Finally, Chapter 5 presents some special considerations for rendering nighttime scenes.

Being all of these topics so independent, I tried as much as possible to make the chapters independent and self contained. When introducing new concepts, I made my best effort in order to make it affordable and easy to follow, providing the intuition behind them without delving much into the complicated details but also not leaving any relevant details out.

For this same reason, the reader of this thesis has little to no prerequisites other than having some computer graphics background. Finally, if one wants to delve deeper into any of the topics presented, plenty of references are provided.

Chapter 2

Atmosphere Rendering

Both the blue appearance of the Earth's sky at noon and the orange shades it takes during dawn and dusk, as well as other effects such as **aerial perspective** (depicted in Figure 2.1), are due to the presence of an **atmosphere**: a layer of constant height that sits on top of the Earth's surface and that is filled with particles such as dust, pollution, water droplets or gases. Particles inside the atmosphere interact with the different wavelengths of light traveling through it and produce visual effects that are otherwise not present when light travels in a vacuum.

Figure 2.1: Real photo showing aerial perspective at increasing levels of strength as the mountains are further away. Aerial perspective makes distant objects look hazier. Note how the color of the more distant mountains almost blends with that of the sky. Photo by Valerius Tygart with CC BY-SA 3.0 license.

These volumes filled with particles that interact with light are more generally known as **participating media** and several elements can be modeled as such, for example, smoke, clouds or fog. When rendering in participating media, the usual rendering equation [Kaj86] is not valid since the assumption that radiance along rays remains constant does not hold [Jar08]. Then, another formulation, which is introduced in Section 2.1.4, has to be used for rendering in participating media.

In this chapter, a brief overview of the concepts of participating media that are relevant when trying to realistically render the Earth's atmosphere is provided. After that, the atmosphere rendering model that I used for the implemented solution is introduced.

2.1 Participating Media[1]

As mentioned before, when light travels through a participating medium it is continuously interacting with the particles that compose it. These interactions are typically referred to as **scattering events**. As opposed to when rendering in a vacuum where radiance remains constant along a ray, in a participating medium, radiance changes with each scattering event. These changes can be classified into the following four types and are depicted in Figure 2.2:

- **absorption**: light is absorbed by the particles in the medium, this event happens at a given rate denoted by σ_a with units of m^{-1}.
- **emission**[2]: light is emitted by the particles in the medium.
- **out-scattering**: light is scattered away by bouncing off the particles in the medium, this event happens at a given rate denoted by σ_s with units of m^{-1} and the distribution of the bouncing direction follow the phase function p.
- **in-scattering**: in the same way that light is scattered off, it can also be scattered in the current light ray by the same principles described in the previous item.

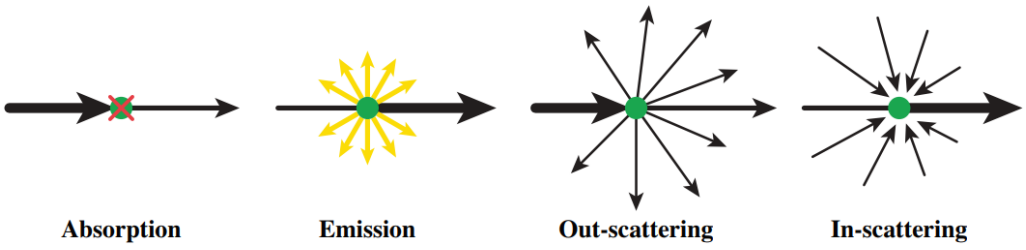

Figure 2.2: The four type of radiance changes that can occur in a participating medium. Figure extracted from [Jar08].

[1]The theory presented in this section is an adapted version of the material available in Section 2 of the course notes of [Hil16] with some additions of the excellent [Jar08], for more detailed information and/or references consult the original sources.

[2]This type is mentioned here for completeness but will be ignored for the rest of the discussion as it is not relevant for our case.

The rates σ_a and σ_s are known as **absorption coefficient** and **scattering coefficient** respectively and can be **wavelength dependent**.

Each of these four types of radiance changes is present to a greater or lesser extent depending on the size of the particles that compose the medium. More precisely, consider $x = \frac{2\pi r}{\lambda}$ to be the relative size of a particle, being r the radius of the particle and λ the wavelength of the light considered. Then, according to x, the following three types of scattering can be identified:

- **Rayleigh scattering** when $x \ll 1$
- **Mie scattering** when $x \approx 1$
- **Geometric scattering**[3] when $x \gg 1$

2.1.1 Rayleigh Scattering

Rayleigh scattering models the scattering of light off particles much smaller than its wavelength. It is highly wavelength dependent and shows little to no absorption, this means that a triplet of coefficients $\sigma_s = (\sigma_{sR}, \sigma_{sG}, \sigma_{sB})$ is used to quantify it and $\sigma_a = 0$. Its phase function is a symmetrical two lobe function that can be seen in Figure 2.3 and its mathematical expression is:

$$p(\theta) = \frac{3}{16\pi}(1 + \cos^2\theta).$$

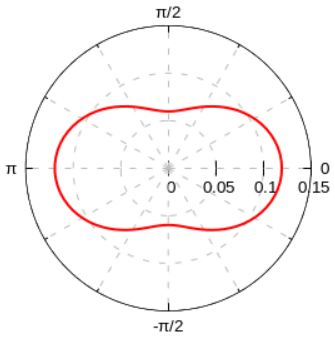

Figure 2.3: Polar plot of the Rayleigh scattering phase function.

In the case of the Earth's atmosphere, the scattering caused by air molecules can be modeled as Rayleigh scattering and **it is the cause of the blue color of the sky at noon and its orange shades at dusk and dawn**.

[3]Again, this type is mentioned here for completeness but will be ignored for the rest of the discussion as it is not relevant for our case.

2.1.2 Mie Scattering

Mie scattering models the scattering of light off particles of similar size to its wavelength. It is, in general, more complex to determine its scattering and absorption behaviors as well as its phase function [Bru17a]. For that reason, its phase function is typically just approximated by other functions such as the Cornette-Shanks phase function [Hil20], the Henyey-Greenstein phase function or the Schlick approximation. These different phase functions can be seen in Figure 2.4 and have the following mathematical expressions:

$$p_{c-s}(\theta) = \frac{3}{8\pi} \frac{(1-g^2)(1+\cos^2\theta)}{(2+g^2)(1+g^2-2g\cos\theta)^{3/2}}$$

$$p_{h-g}(\theta) = \frac{1}{4\pi} \frac{1-g^2}{(1+g^2-2g\cos\theta)^{3/2}}$$

$$p_{schlick}(\theta) = \frac{1}{4\pi} \frac{1-k^2}{(1+k\cos\theta)^2}$$

where $k \approx 1.55g - 0.55g^3$ in the Schlick approximation and g is a parameter in the $(-1, 1)$ range that controls the amount of backward/forward scattering.

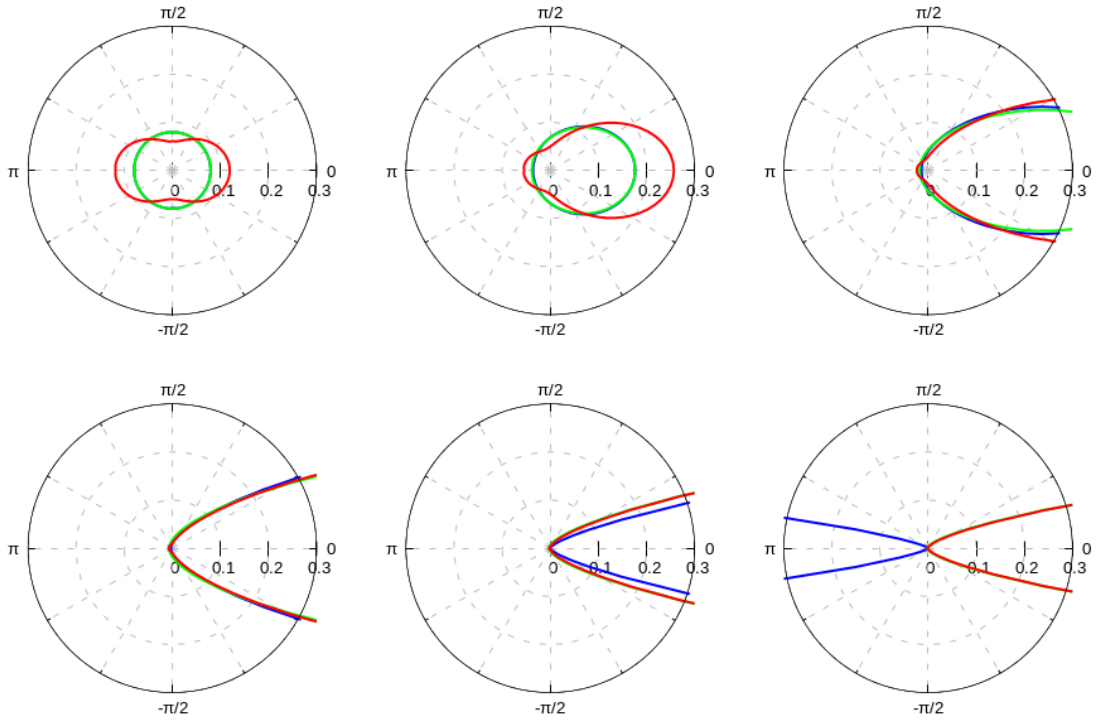

Figure 2.4: Polar plot of the different approximations of the Mie scattering phase function. Red: Cornette-Shanks phase function. Green: Henyey-Greenstein phase function. Blue: Schlick approximation. From left to right and top to bottom, values of $g = 0.00$, 0.25, 0.50, 0.75, 0.90 and 0.95. Note how for values of g very close to 1.00, the Schlick approximation flips, which is an undesired behavior. Also note how, the Cornette-Shanks phase function for $g = 0.0$ is exactly equal to the Rayleigh scattering phase function in Figure 2.3.

In the case of the Earth's atmosphere, the scattering caused by large particles near the ground (whose concentration depends a lot on weather conditions and/or pollution) can be modeled as Mie scattering and **it is the cause of the bright halo usually visible around the Sun**.

2.1.3 Rayleigh Scattering and Mie Scattering Comparison

In order to better understand the different effects Rayleigh and Mie scattering cause, Figure 2.5 shows them at different levels of strength. On the one hand, notice how reducing Rayleigh scattering (top left) makes the sky become darker producing a sense of having no atmosphere while increasing it (bottom left) makes the blue color to be mostly absorbed similarly to what happens at dawn and dusk, where light has to travel a thicker part of the atmosphere. On the other hand, notice how decreasing/increasing Mie scattering produces a sense of cleaner/more polluted air.

Figure 2.5: Left: the effect of modifying Rayleigh scattering. Right: the effect of modifying Mie scattering. From top to bottom: reduced scattering, Earth-like scattering, increased scattering. Figure extracted from [Hil16].

2.1.4 Rendering in Participating Media

Due to the aforementioned changes of radiance along rays caused by scattering events, rendering in a participating medium is much more involved than rendering in the vacuum. Figure 2.6 shows a sketch of all the elements involved in it.

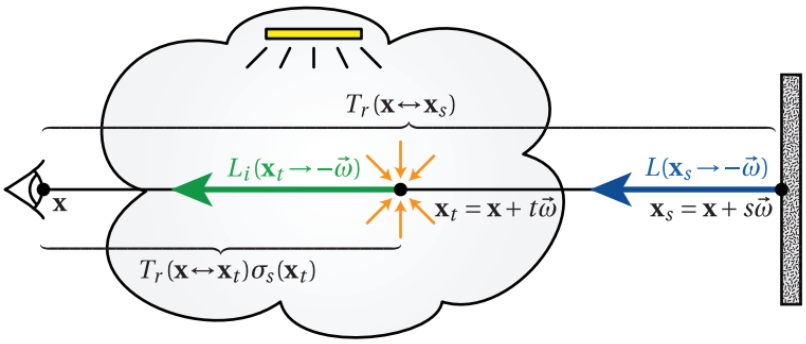

Figure 2.6: Sketch depicting the elements involved when rendering in a participating medium. Figure extracted from [Jar08].

As in any rendering problem, the goal is to compute the radiance reaching the observer's position $\mathbf{x}$ from all directions $\vec{\omega}$ in the viewing frustum, this will be denoted by $L(\mathbf{x} \leftarrow \vec{\omega})$ (not depicted in the sketch). Along the ray originating at $\mathbf{x}$ with direction $\vec{\omega}$ the first opaque surface that is encountered is at point $\mathbf{x}_s = \mathbf{x} + s\vec{\omega}$ with an exitant radiance of $L(\mathbf{x}_s \rightarrow -\vec{\omega})$ (blue arrow in the sketch). In a non-participating media rendering context, it would hold that

$$L(\mathbf{x} \leftarrow \vec{\omega}) = L(\mathbf{x}_s \rightarrow -\vec{\omega}) \tag{2.1}$$

and $L(\mathbf{x}_s \rightarrow -\vec{\omega})$ would be simply computed by using the usual rendering equation [Kaj86].

In a participating medium, however, due to the scattering events that are produced in it, this **radiance is attenuated as it travels** from $\mathbf{x}_s$ to $\mathbf{x}$, which means that only a fraction of it will reach $\mathbf{x}$. This fraction is the **transmittance** between $\mathbf{x}$ and $\mathbf{x}_s$, which is denoted by $Tr(\mathbf{x} \leftrightarrow \mathbf{x}_s)$ and computed as

$$Tr(\mathbf{x} \leftrightarrow \mathbf{x}_s) = e^{-\int_0^s \sigma_t(\mathbf{x}_t)dt} \tag{2.2}$$

where $\sigma_t = \sigma_a + \sigma_s$ is the **extinction coefficient**. Among others, the transmittance has a very useful property of being multiplicative for colinear points i.e. for all points $\mathbf{x}'$ in the segment that joins $\mathbf{x}$ and $\mathbf{x}''$ it holds that

$$Tr(\mathbf{x} \leftrightarrow \mathbf{x}'') = Tr(\mathbf{x} \leftrightarrow \mathbf{x}')\, Tr(\mathbf{x}' \leftrightarrow \mathbf{x}''). \tag{2.3}$$

Incorporating the transmittance term into Equation 2.1 then becomes

$$L(\mathbf{x} \leftarrow \vec{\omega}) = L(\mathbf{x}_s \rightarrow -\vec{\omega})\, Tr(\mathbf{x} \leftrightarrow \mathbf{x}_s). \tag{2.4}$$

However, this is not the final formulation yet. As well as attenuating the radiance from the encountered surface, **the participating medium can also add to this radiance**

by scattering light coming from other directions (yellow arrows in the sketch) onto $-\vec{\omega}$. This is the in-scattering as described in Section 2.1 which is denoted by $L_i(\mathbf{x}_t \rightarrow -\vec{\omega})$ (green arrow in the sketch) and computed as

$$L_i(\mathbf{x} \rightarrow \vec{\omega}) = \int_{\Omega} p(\mathbf{x}, \theta)\, L(\mathbf{x} \leftarrow \vec{\omega}')d\vec{\omega}' \tag{2.5}$$

where Ω denotes the whole sphere of directions, $p(\mathbf{x}, \theta)$ is the phase function and θ is the angle formed between $\vec{\omega}$ and $\vec{\omega}'$. This in-scattering can happen at any point $\mathbf{x}_t = \mathbf{x} + t\vec{\omega}$ between $\mathbf{x}_s$ and $\mathbf{x}$ thus the term added will be an integral for all the range of values of t. Incorporating this term into Equation 2.4 gives us the final formulation that will be used for rendering in a participating medium

$$L(\mathbf{x} \leftarrow \vec{\omega}) = L(\mathbf{x}_s \rightarrow -\vec{\omega})\, Tr(\mathbf{x} \leftrightarrow \mathbf{x}_s) + \int_0^s L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\, \sigma_s(\mathbf{x}_t)\, Tr(\mathbf{x} \leftrightarrow \mathbf{x}_t)dt. \tag{2.6}$$

By inspecting this final formulation we can observe that computing the desired radiance $L(\mathbf{x} \leftarrow \vec{\omega})$ in a participating medium is then just a matter of

1. **computing the radiance of the encountered surface** $L(\mathbf{x}_s \rightarrow -\vec{\omega})$ using the usual rendering equation
2. **multiplying by the transmittance** $Tr(\mathbf{x} \leftrightarrow \mathbf{x}_s)$
3. and finally **adding the in-scattering term** $\int_0^s L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\, \sigma_s(\mathbf{x}_t)\, Tr(\mathbf{x} \leftrightarrow \mathbf{x}_t)dt$.

As in the usual rendering equation, note the recursivity in this formulation: L_i depends on L which at the same time depends on L_i. The accuracy of the rendering method is then determined by the number of bounces taken into account when computing L_i: **if only direct light is considered then the method is said to only support single scattering, if direct light and also light that has bounced once is considered then the method is said to support second order scattering, and so on for higher orders of scattering**.

2.2 Atmosphere Rendering Models

As of the writing of this thesis, **there exist plenty of atmosphere rendering models** that allow to render the visual effects caused by the presence of an atmosphere. In [Bru17a], eight of these models are extensively evaluated both in a qualitative and quantitative manner. Essentially, the models presented in [Bru17a] can be split into two categories: **analytical and numerical models**. While analytical models attempt to derive a mathematical formula that directly provides the sky spectral radiance i.e. the sky color, numerical models use ray marching and integration techniques to compute the terms required to evaluate Equation 2.6. Analytical models sound then very appealing since nothing more than the evaluation of a mathematical function is required to render the sky, however, these are very limited as they only support views from the ground and without taking into

account aerial perspective. On the other hand, numerical methods are more flexible as most of them support the whole range of viewpoints (from the ground up to outside the atmosphere) as well as aerial perspective, of course, this comes at the expense of memory consumption and computation time. These numerical methods work by precomputing some high dimensional LUTs that allow to efficiently evaluate Equation 2.6 at rendering time. Luckily, with some simplifying assumptions and exploiting some properties such as the Earth's spherical symmetry and the transmittance property shown in Equation 2.3, these high dimensional tables can be reduced to manageable dimensionalities such as 2D, 3D or 4D at most [Bru17b]. Moreover, since the details in the sky have a low frequency [Hil20], the size of the tables along each dimension can be small and thus making the whole LUTs of a manageable size.

2.3 The Bruneton Model

Among the atmosphere rendering models from [Bru17a], the Bruneton model [BN08] is the one I have chosen for my implementation. According to the quantitative evaluation, it is **the most physically accurate model**, having support for all scattering orders and only matched in accuracy by an extension of it [EK10][4]. Besides having good real-time performance, reasonable precomputation time and reasonable memory consumption, there is available a C++ and OpenGL open source reference implementation of it provided by the authors which I will be using and that makes it easier to integrate into existing renderers [Bru17b].

The Bruneton model **is a numerical model** and as such works by precomputing a set of LUTs, in this particular case three are the precomputed LUTs: transmittance, scattering and irradiance. The transmittance and scattering LUTs are 2D and 4D respectively and allow to efficiently compute the transmittance and in-scattering terms of Equation 2.6 at rendering time. On the other hand, the irradiance LUT is also 2D and is used internally by the model in its precomputation phase for computing the higher order scattering terms, moreover, it is also useful at rendering time to shade ground objects without requiring to perform an environment capture and using IBL techniques.

Besides the aforementioned Rayleigh and Mie scattering components, the Bruneton model **also takes into account Ozone**: a specific component of the Earth's atmosphere that has been proven important for correctly obtaining the blue colors of the sky especially when the Sun is close to the horizon [Hil20]. Each of the absorption and scattering coefficients of these components and other properties such as the radius of the planet or its ground albedo color are **customizable through several parameters**, thus allowing to not only render Earth-like atmospheres but all kinds of different atmospheres. In the implementation, all of these parameters are exposed through a graphical interface so they are easy to change interactively and get a grasp of what effect has each parameter on the visual result. By default, these parameters are set to the values that can be found in the table in Appendix B which model a physically correct Earth's atmosphere.

[4] In fact, under certain circumstances explained in [Bru17a], these two models are exactly equal.

One of the limitations that Bruneton and all of the models from [Bru17a] have by design is that they only support one light source, namely the Sun. However, for our use case of developing a day and night cycle solution, we require two light sources: the Sun and the Moon[5]. To overcome this limitation and as suggested by [McA18], in my implementation I will use **two separate instances of the model**: one for each celestial body.

[5]Technically speaking the Moon is not a light source, however during night and thanks to the reflection of light coming from the Sun, it is considered the main source of natural light. This topic will be further discussed in Chapter 4.

Chapter 3

Celestial Bodies Positioning

Positioning celestial bodies such as the Sun, the Moon and stars in an appropriate coordinate system is a classical astronomical problem. As such, there already exist several solutions that are able to provide these positions at a high degree of accuracy. These solutions are often presented as formulas that, unfortunately for our interests, have two issues: they are highly complex and are usually given in terms of non-standard units.

In [Jen+01], the authors provide simplified low-accuracy formulas that are derived from these aforementioned high-accuracy classical formulas. These simplified formulas are given in units that are much more familiar to typical computer graphics applications and thus are easier to implement. These simplified formulas are the ones that I use in my implementation.

Before presenting the formulas by the end of this chapter, a brief discussion of two important concepts is required, namely, time and coordinate systems in the context of astronomy.

3.1 Astronomical Time

In astronomical contexts, an instant in time is usually specified as a **Julian Date**. Without entering much into the technicalities, a Julian Date is a decimal quantity that counts the number of days since a set reference epoch. Given an instant of time specified as a date $D/M/Y$ ($Y > 1582$) and a time $h : m : s$ ($0 \leq h < 24$), its Julian Date, denoted by JD, can be computed as

$$\begin{aligned} \text{JD} = \quad & 1720996.5 - \lfloor Y'/100 \rfloor + \lfloor Y'/400 \rfloor + \lfloor 365.25Y' \rfloor \\ + \quad & \lfloor 30.6001(M'+1) \rfloor + D + (h + (m + s/60)/60)/24 \end{aligned} \tag{3.1}$$

where $Y' = Y - 1$ and $M' = M + 12$ if M is 1 or 2, otherwise they are just $Y' = Y$ and $M' = M$.

Unfortunately, just using this formula as provided is not totally accurate and the reason is a **discrepancy that exists in the way of measuring time in astronomy versus the way of measuring time in our daily lives**, also known as civil time. On the one hand, astronomical theories of the positioning of celestial bodies are derived using mathematical formulas which require a uniformly increasing time scale. This standard time scale is known as Terrestrial Time (TT). On the other hand, civil time is based on UTC which is adjusted to the slightly erratic rotation of the Earth around the Sun. This implies that leap seconds are usually introduced to UTC time, making TT and UTC drift apart. This discrepancy which is measured in seconds is known as ΔT and defined as $\Delta \mathrm{T} = \mathrm{TT} - \mathrm{UTC}$. The value of ΔT is constantly changing and there exist several models that give its value for different periods of time. As of the writing of this thesis, its current value is about $73s$ and this fixed value is the one that I will be using for the implementation. Using a fixed value should provide good enough accuracy at least for periods of time in the order of decades up to a few centuries due to the slow variation of this value. [Gen10]

Then, given that the time is always provided in UTC terms, in order to obtain an accurate Julian Date using Equation 3.1, the correction term ΔT has to be introduced by adding it to the variable s.

Finally, the formulas provided in Section 3.3 will be given in terms of the variable T, which correspond to Julian centuries since January the 1st, 2000 and is computed as

$$\mathrm{T} = (\mathrm{JD} - 2451545)/36525.$$

3.2 Astronomical Coordinate Systems[1]

When viewed from the ground, the sky looks like a dome above us, which is the reason why the use of **spherical coordinate systems** is so present in positional astronomy. A spherical coordinate system is defined by providing three elements: a reference point that serves as the origin, a reference plane[2] that contains the origin and a reference direction that lays in this reference plane. Given a point, its position in such a spherical coordinate system can be determined by three coordinates: two angles generically known as longitude and latitude (but that can take other names as we will see later) and a radius which corresponds to the distance of the point to the origin of the coordinate system. To obtain its longitude and latitude consider the segment that joins the point with the origin. The latitude is the signed angle formed between the reference plane and the segment. The longitude is the counterclockwise angle formed between the reference direction and the projection of that segment onto the reference plane.

For positioning the Earth, the Sun and the stars, three are the coordinate systems that will be used: the **horizon coordinate system**, the **equatorial coordinate system** and

[1]The definition of the coordinate systems presented in this section may differ from their usual definition found in other astronomical texts. It is important to follow the definitions made in this section in order for the formulas presented at the end of this chapter to work.

[2]This reference plane has to be oriented by providing a normal vector so that the plane separates the space into positive and negative half-spaces. This is crucial for unambiguously defining both latitude and longitude.

the **ecliptic coordinate system**. Figure 3.1 shows a diagram of these three coordinate systems.

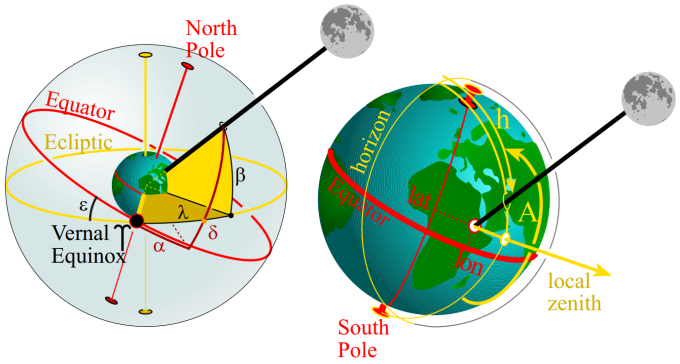

Figure 3.1: Position of the Moon in different coordinate systems. Left: equatorial and ecliptic coordinate systems depicted in red and yellow respectively. Right: horizon coordinate system. Figure extracted from [Jen+01].

3.2.1 Equatorial and Ecliptic Coordinate Systems

Both equatorial and ecliptic coordinate systems have their origin at the center of the Earth and only differ in their reference plane. While the equatorial coordinate system uses the equatorial plane[3] oriented by the North Pole, the ecliptic coordinate system uses the plane of orbit of the Earth around the Sun oriented so that the dot product of the normals of these two reference planes is positive. As the name of the coordinate system suggests, this plane is known as the ecliptic plane. The angle formed by the two reference planes is commonly known as the obliquity of the ecliptic and denoted by ε. Both coordinate systems use the same reference direction which corresponds to the intersection of their reference planes. This direction is commonly known as the Vernal Equinox and denoted by Υ.

In the equatorial coordinate system, the longitude and latitude coordinates are commonly known by right ascension and declination and denoted by α and δ respectively. In the ecliptic coordinate system, the longitude and latitude coordinates are commonly denoted by λ and β respectively.

3.2.2 Horizon Coordinate System

The final coordinate system that will be used is the horizon coordinate system. This is, in fact, the most useful coordinate system for our purposes since **it provides the**

[3]Despite its similarities, the equatorial coordinate system does not have to be confused with its terrestrial counterpart the geographic coordinate system. While both are spherical coordinate systems that use the equator as its reference plane, one is used for positioning celestial bodies and the other is used for positioning objects on the surface of the Earth.

coordinates of the celestial bodies as seen from an observer on the surface of the Earth[4] that is looking at the sky.

The origin of this coordinate system is then the position of the observer and the reference plane is the plane tangent to the Earth at that precise point oriented by the local zenith. The reference direction is the South cardinal direction.

In this coordinate system, the longitude and latitude coordinates are commonly known by azimuth and elevation and denoted by A and h.

3.2.3 Coordinate Systems Conversion

Converting between coordinate systems is more easily done in their associated rectangular coordinate systems rather than the spherical coordinate systems. To obtain the coordinates of a given point in its associated rectangular coordinate system, the following conversion formula is used

$$\begin{cases} x &= r\cos(\text{latitude})\cos(\text{longitude}) \\ y &= r\cos(\text{latitude})\sin(\text{longitude}) \\ z &= r\sin(\text{latitude}) \end{cases}$$

and to obtain once again the spherical coordinates its inverse conversion is used

$$\begin{cases} r &= \sqrt{x^2+y^2+z^2} \\ \text{longitude} &= \arctan 2(y,x) \\ \text{latitude} &= \arcsin(z/r). \end{cases}$$

Once in rectangular ecliptic coordinates, converting a point to rectangular equatorial coordinates is just a matter of rotating around the x axis by the obliquity of the ecliptic i.e. applying the rotation matrix $R_x(\varepsilon)$, with $\varepsilon = 0.409093 - 0.000227\text{T}$.

Converting to horizon coordinates is a bit harder and has to take into account the position of the observer on the Earth, here denoted by (lon, lat). The matrix for converting from rectangular equatorial coordinates to rectangular horizon coordinates is then

$$R_y(\text{lat} - \pi/2)R_z(-\text{LMST})P$$

where

$$P = R_z(0.01118\text{T})R_y(-0.00972)R_z(0.01118\text{T})$$

and

$$\text{LMST} = 4.894961 + 230121.675315\text{T} + \text{lon}.$$

Important to note that, as opposed to the rest of the formulas presented in this chapter, in the computation of LMST, T is computed without taking into account the ΔT correction.

[4]Thus, the horizon coordinate system is not just one but several of them, depending on the position of the observer on the surface of the Earth.

3.3 The Formulas

The formulas for both **the Sun's and the Moon's positions are given in ecliptic coordinates** and can later be converted to any of the other coordinate systems by applying the conversion formulas from the previous section. All angles are given in radians and the radii are given in astronomical units[5].

3.3.1 Sun Position

The Sun's position in ecliptic coordinates is given by

$$\begin{cases} \lambda &= 4.895048 + 628.331951\mathrm{T} + (0.033417 - 0.000084\mathrm{T})\sin M + \sin 2M \\ \beta &= 0 \\ r &= 1.000140 - (0.016708 - 0.000042\mathrm{T})\cos M - 0.000141\cos 2M \end{cases}$$

where $M = 6.24 + 628.302\mathrm{T}$.

3.3.2 Moon Position

The Moon's position in ecliptic coordinates is given by

$$\begin{aligned} \lambda = l' \\ &+ 0.1098\sin(m') \\ &+ 0.0222\sin(2d - m') \\ &+ 0.0115\sin(2d) \\ &+ 0.0037\sin(2m') \\ &- 0.0032\sin(m) \\ &- 0.0020\sin(2f) \\ &+ 0.0010\sin(2d - 2m') \\ &+ 0.0010\sin(2d - m - m') \\ &+ 0.0009\sin(2d + m') \\ &+ 0.0008\sin(2d - m) \\ &+ 0.0007\sin(m' - m) \\ &- 0.0006\sin(d) \\ &- 0.0005\sin(m + m') \end{aligned}$$

[5] $1\mathrm{au} = 1.496 \cdot 10^{11}\mathrm{m}$

$$\begin{aligned}\beta = &+0.0895\sin(f)\\ &+0.0049\sin(m'+f)\\ &+0.0048\sin(m'-f)\\ &+0.0030\sin(2d-f)\\ &+0.0010\sin(2d+f-m')\\ &+0.0008\sin(2d-f-m')\\ &+0.0006\sin(2d+f)\end{aligned}$$

$$r = \frac{1}{23455\pi'}$$

where

$$\begin{aligned}\pi' = &+0.016593\\ &+0.000904\cos(m')\\ &+0.000166\cos(2d-m')\\ &+0.000137\cos(2d)\\ &+0.000049\cos(2m')\\ &+0.000015\cos(2d+m')\\ &+0.000009\cos(2d-m)\end{aligned}$$

and

$$\begin{aligned}l' &= 3.8104 + 8399.7091\mathrm{T}\\ m &= 6.2300 + 628.3019\mathrm{T}\\ f &= 1.6280 + 8433.4663\mathrm{T}\\ m' &= 2.3554 + 8328.6911\mathrm{T}\\ d &= 5.1985 + 7771.3772\mathrm{T}\end{aligned}$$

3.3.3 Stars Positions

Due to the great distances at which they are located, **distant stars can be considered to be at fixed positions**, at least for time periods of up to five centuries. This is why, rather than using formulas to compute their positions, these are obtained from star catalogues such as the *Yale Bright Star Catalogue* [HJ91] or the *Hipparcos and Tycho Catalogues* [ESA97]. The approach used to position stars in the implementation will be discussed in Section 4.2.

Chapter 4

Celestial Bodies Rendering

Correctly rendering the appearance of the Sun, the Moon and the stars, each comes with its own set of particularities. In this chapter, the different techniques that I use to render these elements in the implemented solution are presented. These are mainly based on the works by Jensen et al. [Jen+01] and Muller et al. [MED12].

4.1 Sun and Moon Rendering

Both the Sun and the Moon are **rendered as disk-shaped billboards** i.e. as flat geometry located at a unit sphere around the camera and that is always oriented towards it. The size of these billboards is based on the **angular diameter** of the celestial bodies they represent. The angular diameter of a celestial body is a quantity often used in astronomy to provide its size, it is usually denoted by δ and measured in either degrees or radians. It quantifies the amount of visual angle the celestial body covers. Figure 4.1 shows a sketch of the elements that are involved in the computation of angular diameter.

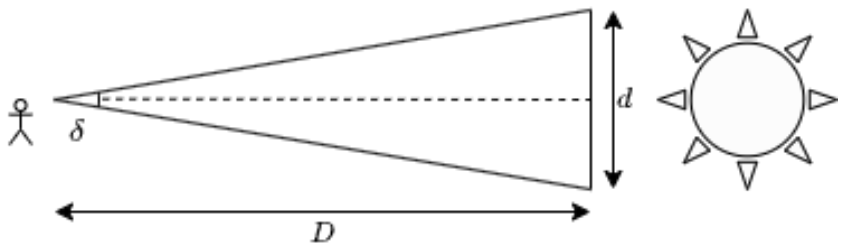

Figure 4.1: Elements involved in the computation of angular diameter. D denotes the distance between the observer and the celestial body, d the diameter of the celestial body and δ the angular diameter.

According to the sketch and some basic trigonometry, the following relationship which allows to compute the angular diameter can be obtained

$$\tan\left(\frac{\delta}{2}\right) = \frac{d/2}{D}. \tag{4.1}$$

Given that the billboards are placed at a unit sphere and must have an angular diameter equal to that of the celestial body they represent, the following relationship holds

$$\frac{b/2}{1} = \frac{d/2}{D} \tag{4.2}$$

where b is the diameter of the billboard. This finally allows to compute the diameter of the billboard as

$$b = \frac{d}{D} = 2\tan\left(\frac{\delta}{2}\right). \tag{4.3}$$

By using the distances obtained with the formulas from the previous chapter and the sizes of the celestial bodies from [Wil21], **the billboards represent the sizes of the celestial bodies in a physically accurate manner**. Unfortunately, this means that, at standard fields of view, the sizes of the Sun and the Moon are too small to appreciate any detail. For that reason, multiplicative parameters that allow modification of their size are added in the implementation.

4.1.1 Sun Shading

The appearance of the Sun is relatively simple to model since its appearance is only due to its **emissive nature**. The only small detail that has to be taken into account is **limb darkening**: a visual effect that makes the Sun to be perceived as a disk of non-uniform radiance when viewed from the Earth. This is due to the fact that more light is received when viewing it in directions close to its normal i.e. close to the center of the disk than when viewing it in directions tangent to the normal i.e. close to the edge of the disk.

In [Hil16], the implementation of two physically based models [HM98] [Nec96] that allow to render the Sun's disk with the appropriate limb darkening is provided. These two models provide **gradients that depend on the distance to the center of the Sun's disk** and when multiplied by the base radiance of the Sun, provide the radiance of any given point on the Sun's disk. These gradients can be seen in Figure 4.2.

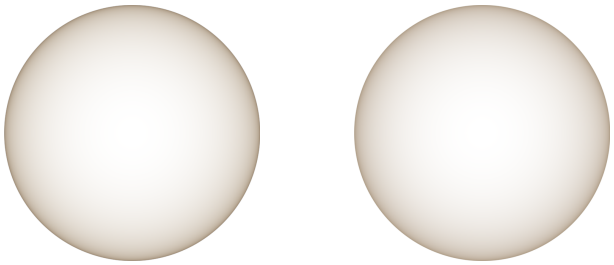

Figure 4.2: Left: Limb darkening gradient from [Nec96]. Right: Limb darkening gradient from [HM98].

As can be observed, both models provide very similar resulting gradients only differing at the edges where the [Nec96] model has a slightly stronger darkening. The implementation of both models is pretty simple and easy to integrate into any existing shader and on par in terms of performance, thus I see no compelling reason to choose one model over the other, so both are implemented in the developed solution. Figure 6.4 shows a comparison of these models applied on top of the Sun's disk.

4.1.2 Moon Shading

The most important aspect that has to be taken into account when shading the Moon is the simulation of **lunar phases** i.e. the periodic changes in its apparent shape when viewed from the Earth. As opposed to the Sun, the Moon is not an emissive surface and its **appearance is mainly due to the reflectance of light coming from other sources**. Being located in the solar system, the **main source of light that illuminates the Moon is, of course, the Sun**. This reflective nature, combined with the spherical shape of the Moon and the change in the relative position of the Moon and the Earth with respect to the Sun as the Moon orbits the Earth, are the causes of the distinct lunar phases, which can be seen in Figure 4.3.

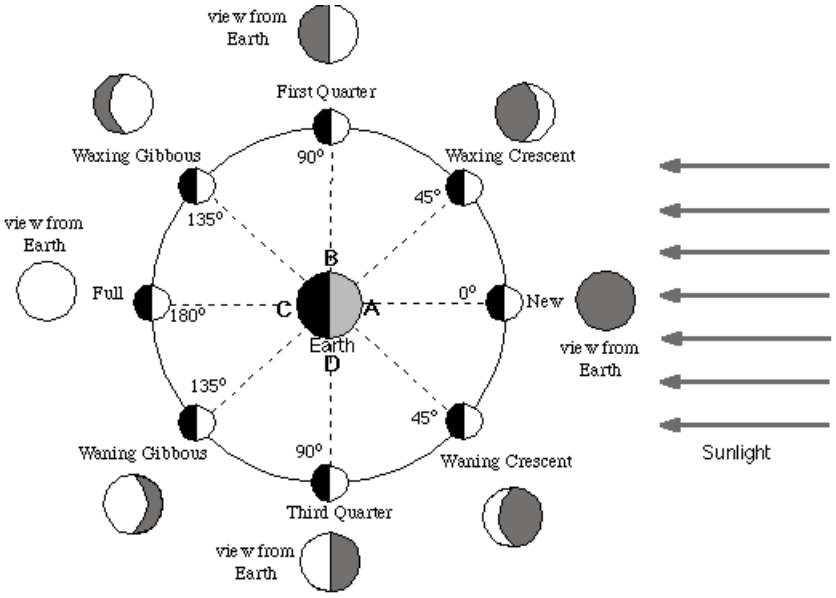

Figure 4.3: Diagram showing different lunar phases according to the relative position of the Earth and the Moon. Sun is considered to be infinitely far away. Note that independently of the position of the Moon, in the zenithal view depicted by the inner circles, the Moon is always lit in the same manner. The only thing that changes is the view from Earth, depicted by the outer circles. Diagram by Nick Strobel extracted from `www.astronomynotes.com`.

Even though direct Sun light is the main source of illumination of the Moon, it is not the only one. Light coming from the Sun and reflected on the Earth also illuminates the

Moon. This is known as the **earthshine** and is important to consider it when shading the Moon because it produces the visual effect shown in Figure 4.4.

Figure 4.4: Real photo showing the Moon at crescent phase. While only the thin slice at the bottom is lit by the Sun, the totality of the Moon's disk is visible due to earthshine. Figure extracted from [Agr15].

Having identified the light sources that illuminate the Moon, it is also **important to determine their irradiance**. As per [Jen+01] I take the irradiance of the Sun to be $1905\,\mathrm{W}\cdot\mathrm{m}^{-2}$ both at the top of the Earth's atmosphere as well as at the surface of the Moon[1], these are denoted by $E_{S\to E}$ and $E_{S\to M}$ respectively. Determining the irradiance of the earthshine at the surface of the Moon is a bit more complex since it depends on their relative position.

By the same principles described before that cause the lunar phases, the Earth also shows different phases when viewed from the Moon, which **causes the earthshine irradiance at the surface of the Moon to fluctuate**. As accuracy is not extremely important for the earthshine irradiance, in [Jen+01] they propose an approximation that consists of multiplying the maximum earthshine irradiance corresponding to Earth at full phase of $0.19\,\mathrm{W}\cdot\mathrm{m}^{-2}$ with the fraction of lit Earth visible from the Moon.

This fraction can be computed by using **phase angles**, which allow to quantitatively determine the phases of both the Moon and the Earth. More precisely, the lunar phase angle, denoted by φ, is the angle formed between the Sun and the Earth as viewed from the Moon. The Earth's phase angle, here denoted by φ', is defined analogously as the angle formed between the Sun and the Moon as viewed from the Earth. Figure 4.3 shows the Earth's phase angles (in degrees) according to the relative position of the Earth and the Moon. As shown in Figure 4.5, these two angles satisfy $\varphi+\varphi'=\pi$, so the lunar phase angle can then be computed as $\varphi=\pi-\varphi'$, while the Earth's phase angle is computed by using the positions obtained with the equations from the previous chapter.

[1]The irradiance is a radiometric quantity that depends on the distance. As such, the irradiance from the same source at different points in space can be different. For the case of the irradiance of the Sun at the top of the Earth's atmosphere and at the Moon's surface, it is a reasonable approximation to take the same value given the relatively small distance between the Earth and the Moon when compared to those to the Sun.

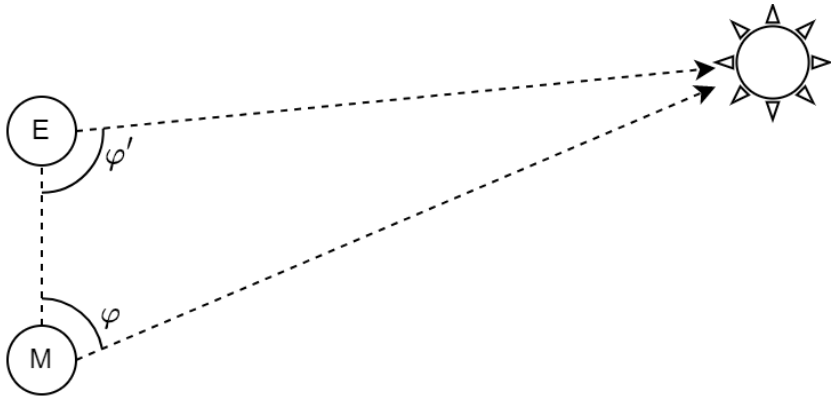

Figure 4.5: Diagram showing the phase angles of both the Moon and the Earth. The angles of the triangle formed by the Sun, the Moon and the Earth add up to π. By considering the Sun to be infinitely far away, the two dashed lines become parallel so the angle formed by the Earth and the Moon as seen from the Sun becomes null and $\varphi + \varphi' = \pi$.

Finally, coming back to the irradiance computation, the irradiance of the earthshine at the surface of the Moon, $E_{E\to M}$, is computed as

$$E_{E\to M} = 0.19\,\frac{1}{2}\left[1 - \sin\left(\frac{\varphi'}{2}\right)\tan\left(\frac{\varphi'}{2}\right)\ln\left(\cot\left(\frac{\varphi'}{4}\right)\right)\right] \tag{4.4}$$

where the expression between square brackets is a sigmoid function that computes the aforementioned fraction of lit Earth visible from the Moon. Using a similar expression, the irradiance received from the Moon at the top of the Earth's atmosphere, $E_{M\to E}$, can be computed

$$E_{M\to E} = \frac{2Cr_m^2}{3d^2}\left\{E_{E\to M} + E_{S\to M}\left[1 - \sin\left(\frac{\varphi}{2}\right)\tan\left(\frac{\varphi}{2}\right)\ln\left(\cot\left(\frac{\varphi}{4}\right)\right)\right]\right\} \tag{4.5}$$

where $C = 0.062$ is the average albedo of the Moon, r_M the radius of the Moon and d the distance between the Moon and the Earth. This is used as a parameter for the atmospheric model.

As mentioned before, the spherical shape of the Moon is crucial for the existence of lunar phases. It **is then important to consider the spherical geometry of the Moon instead of the flat geometry of the billboard** with which it is actually rendered. The way of doing so is by recovering in the fragment shader the normal of the sphere that the billboard represents as $\mathrm{N} = (x, y, \sqrt{1 - x^2 - y^2})$ where (x, y) are the texture coordinates of the point of the billboard to shade.

Rather than modeling the Moon as a regular diffuse surface and shading it with the usual Lambertian BRDF, due to its specific photometric properties, **a specialized BRDF** is used. Denoting by θ_i the angle between the incident light and the normal of the surface, θ_r the angle between the reflected light and the normal of the surface and φ the angle between the incident light and the reflected light, this specialized BRDF is given by

$$f(\theta_i, \theta_r, \varphi) = \frac{2}{3\pi}B(\varphi)S(\varphi)\frac{1}{1 + \cos\theta_r/\cos\theta_i} \tag{4.6}$$

where $B(\varphi)$ is a retrodirective function with the following expression

$$B(\varphi) = \begin{cases} 2 - \frac{\tan\varphi}{2g}\left(1 - e^{-g/\tan\varphi}\right)\left(3 - e^{-g/\tan\varphi}\right) & \varphi < \pi/2 \\ 1 & \varphi \geq \pi/2 \end{cases}$$

and $S(\varphi)$ is the average scattering of lunar surface particles given by

$$S(\varphi) = \frac{\sin|\varphi| + (\pi - |\varphi|)\cos|\varphi|}{\pi} + t\left(1 - \frac{1}{2}\cos|\varphi|\right)^2.$$

The parameters g and t are chosen as $g = 0.6$ and $t = 0.1$ as suggested in [Jen+01]. The f function of Equation 4.6 is then multiplied by the spectral albedo of the Moon in order to obtain the actual BRDF. This spectral albedo is taken as $c = 0.072 \cdot (1.2525, 1.04125, 0.8625)$ for the usual triplet of RGB wavelengths of $(680, 550, 440)$, also as per [Jen+01]. Figure 4.6 shows a comparison between the usual Lambertian BRDF and the Moon's specialized BRDF. Note how the retroreflective term of the specialized BRDF makes the sphere noticeable brighter when it is hit by light coming from the same viewing direction (bottom left) as opposed to when it is hit by light from other directions (bottom right). This behavior is consistent with the one exhibited by the real Moon where **full Moon appears much brighter than other phases**. Another feature of this specialized BRDF is the presence of **less limb darkening** which makes the sphere look like a flat disk rather than an actual sphere. Even though it is not that noticeable in the comparison, I measured this reduction in limb darkening from a ratio of around 3 between the brightest parts at the center to the darkest parts at the edge of the Lambertian sphere to a ratio of 2.5 measured in the sphere shaded with the specialized BRDF.

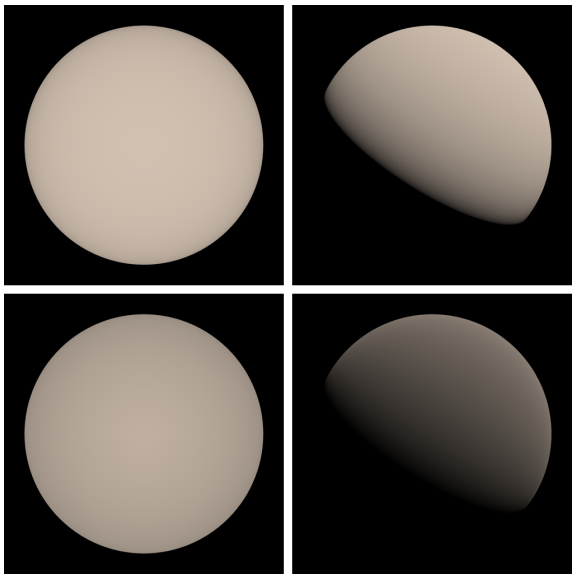

Figure 4.6: Top: Lambertian BRDF. Bottom: Moon's specialized BRDF. Left: Light coming from camera direction. Right: Light coming from the top right. Both the intensity of the light and the albedo of the surface remain constant in order to provide an accurate comparison. Renders performed in [dfr20].

Besides using the specialized BRDF and the correct physical value for the albedo, the usual **color mapping and normal mapping techniques are also applied** when shading the Moon. High resolution versions of the textures required to apply these techniques can be obtained directly from the NASA. In particular, at [Stu19], they provide both color and displacement textures that I will use for the implementation. The displacement texture can then be converted to a normal texture by using specialized software and the color texture can be used as is to modulate the BRDF by multiplying it. The result obtained after applying these techniques can be seen in Figure 4.7. Note how the detail added by the normal mapping technique is very subtle and just barely noticeable close to the terminator line i.e. the line that separates the lit part from the dark part.

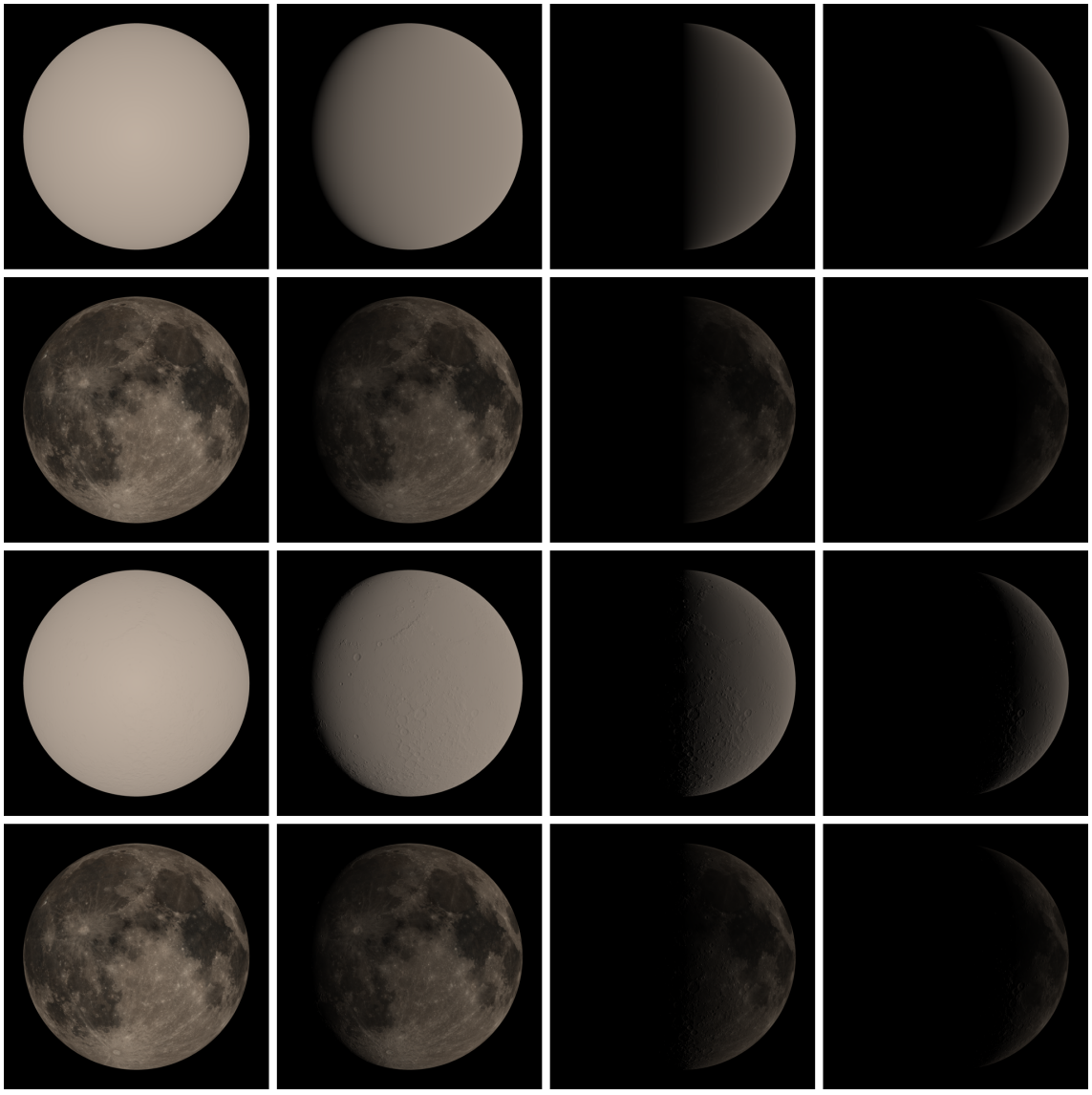

Figure 4.7: Moon shaded at different phases. From top to bottom: None of the techniques applied, color mapping applied, normal mapping applied and both color and normal mapping applied. Renders performed in [dfr20].

4.2 Stars Rendering[2]

As mentioned in Section 3.3.3, distant stars can be considered to be at fixed positions in the galaxy, thus **a static HDR texture suffices to render them**. Once again, at [Stu20], the NASA provides a set of HDR textures that can be used for this purpose. The totality of the stars of the galaxy are separated into two textures: a foreground texture that contains the brightest stars obtained from the *Hipparcos and Tycho Catalogue* [ESA97] and a background texture that contains a set of smaller, undistinguishable by the naked eye, stars that form the Milky Way. Unfortunately, **the units in which these textures come are not specified** and thus I was not able to correctly use them as physical values. The solution I chose is then to sacrifice physical accuracy by allowing the intensity of these textures to be freely controlled by multiplicative parameters. These parameters are independent, one for each texture, so that there is more control over the final result.

These textures come in equatorial coordinates, so **they are easy to render by just using a full screen quad** and recovering the horizon coordinates of the view direction at the fragment shader. By using the conversion from Section 3.2.3, these horizon coordinates are then converted into equatorial coordinates with which the textures are finally sampled and composited.

[2]See Appendix A for an alternative approach.

Chapter 5

Nighttime Scenes Rendering

Rendering nighttime scenes is not as simple as rendering daytime scenes. If one were to render a nighttime scene without having any extra consideration, the outcome would be similar to the one that can be seen in Figure 5.1, where the **nighttime scene is barely distinguishable from its daytime counterpart** other than for the presence of stars in the sky.

Figure 5.1: Same scene rendered at different times of the day and exposed to have the same luminance. Top: Noon. Bottom: Midnight. Figure extracted from [McA18].

In this chapter, before explaining in Section 5.2 the approach I follow in my implementation to render compelling nighttime scenes, a brief overview of the problematics it tries to overcome is presented in Section 5.1. After taking into account the considerations presented in this chapter, one might be able to obtain **results that present more of a "night feeling"** as can be observed in Figure 5.2.

Figure 5.2: Same scene as in Figure 5.1 rendered with a proper treatment for nighttime scenes. Figure extracted from [McA18].

5.1 The Problem

Physically based rendering methods such as the ones presented in this thesis produce as a result a more or less faithful reproduction of the radiances/luminances of the scene. Due to the limitations of display systems and the way human vision works, directly displaying them in typical computer screens usually does not produce the appealing results. Sections 5.1.1 and 5.1.2 present two related concepts that have to be considered in order to obtain convincing results.

5.1.1 Tone Reproduction[1]

Both in computer graphics and photography, the mapping of scene luminances to display luminances is referred to as **tone reproduction**. The development of tone reproduction algorithms, also known as tonemapping operators, that preserve the viewing experience a human observer would obtain by viewing the real scene is a complex task since it deals with perceptual effects that occur in the human visual system and thus usually can't be measured.

A key concept to better understand the difficulties of tone reproduction is that of **dynamic range**. Loosely speaking, the dynamic range of some variable is the ratio between its largest measurable value and its smallest measurable value. It is usually measured in terms

[1]Even though the tone reproduction problem is not exclusive to nighttime scenes, in fact, it has to be dealt with in any type of physically based rendering, it is presented here because it serves as a good starting point to later introduce other concepts that are related to nighttime rendering.

of some logarithmic scale such as decibels or stops. In the particular case of luminances that will be discussed in this chapter, the unit used is $\log_{10}$ units.

The first difficulty one encounters when mapping scene luminances to display luminances is the vastly different dynamic ranges they have. While real scenes can span a wide range of luminances of around 8 $\log_{10}$ units, typical monitors in which these scenes have to be displayed, only have a dynamic range of around 2 to 3 $\log_{10}$ units [Fer+96]. It is then clear that the tone reproduction algorithm has to perform a **dynamic range compression**: failing to do so will cause undesirable results in scenes with high dynamic range. This problem is shown in Figure 5.3, where a simple, linear tone mapping operator is used resulting in a loss of detail either in the dark or bright areas of the scene. A more complex mapping that correctly compresses the dynamic range is also shown. Note that a human observer viewing that scene would be able to see detail both in the dark and bright areas of it, thus making the more complex mapping a more accurate and desirable reproduction.

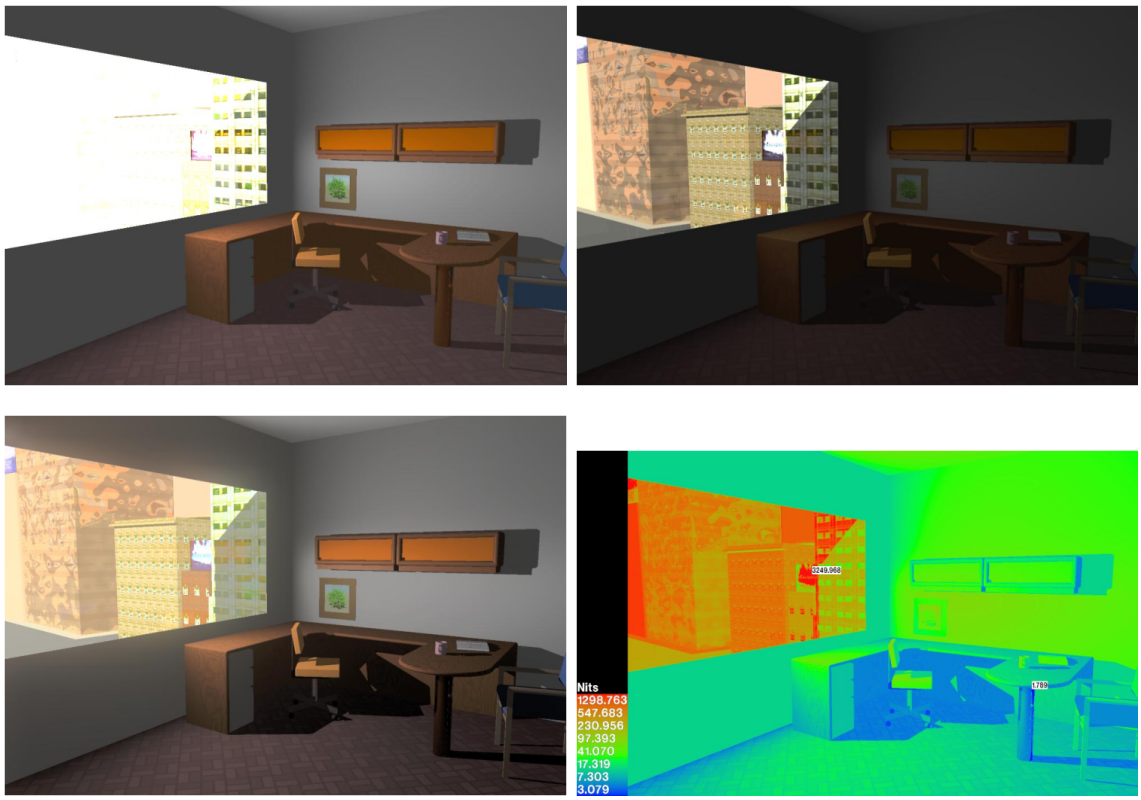

Figure 5.3: Top: Linear mappings of the scene. Bottom left: Mapping that better compresses the dynamic range of luminances in the scene. Bottom right: Luminances of the scene. Figure extracted from [LRP97].

As opposed to display systems that have a fixed and rather low dynamic range, the human visual system has a much higher dynamic range. However, at a given instant in time, only a portion of it is available: **the totality of the dynamic range of the human visual system is achieved through a combination of different adaptation**

mechanisms [Fer+96]. The presence of these adaptation mechanisms can be noticed, for example, when exiting a dark tunnel on a sunny day or when turning the lights off in an, otherwise, completely dark room. These phenomena are known as light and dark adaptation respectively and it is important to incorporate them in some applications such as driving simulators where the visibility conditions have to be accurately represented.

Even though these adaptation mechanisms provide vision over a wide range of luminances, the **human visual system works differently at different levels of adaptation**. Sorted from high to low level of luminance adaptation, three different types of vision can be distinguished: **photopic vision**, **mesopic vision** and **scotopic vision**. While photopic vision is the usual cone-based vision that provides color perception, scotopic vision is mostly monochromatic and rod-based [2]. Finally, mesopic vision is the transition between the former two and is not that well understood [Fer+96]. Figure 5.4 shows the range of luminances each of these adaptation levels span.

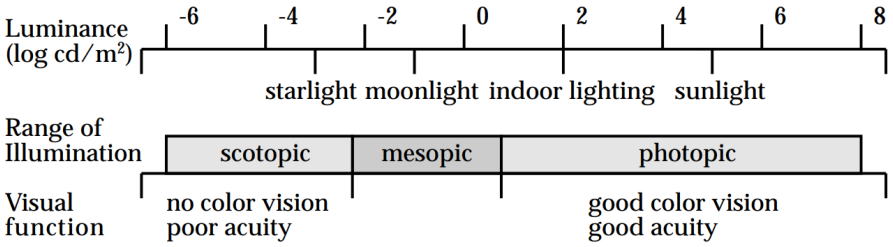

Figure 5.4: Luminance ranges and their corresponding type of vision. Figure extracted from [Fer+96].

5.1.2 Scotopic Vision

In this section, the distinctive features that characterize scotopic vision are discussed. This is of interest to us because scotopic vision is the one that prevails in nighttime scenes where there are mostly low luminance conditions.

Purkinje Effect

Under scotopic vision, the perception of luminance is different than that under photopic vision. This perceptual effect can be quantified thanks to the **luminous efficiency functions** provided by the *CIE* [3], which measure the average sensitivity of human visual perception to different wavelengths of light. Figure 5.6 shows several of these luminous efficiency functions at different levels of adaptation. Note how the peak in sensitivity shifts from long wavelengths i.e. red for photopic conditions to short wavelengths i.e. blue for scotopic conditions. This shift in the peak sensitivity **causes red colors to look**

[2] Cones and rods are the two types of photoreceptor cells that contribute to human vision.

[3] *CIE* stands for *Commission internationale de l'éclairage* (or *International Commission on Illumination* in English) and is the international authority on topics related to light, color and vision. See more at the *CIE* website.

relatively darker in low light levels. This phenomenon is known as the *Purkinje Effect* and an example of it is shown in Figure 5.5.

Figure 5.5: Perceived appearance of the same scene under different levels of adaptation. From left to right: photopic vision, mesopic vision and scotopic vision.

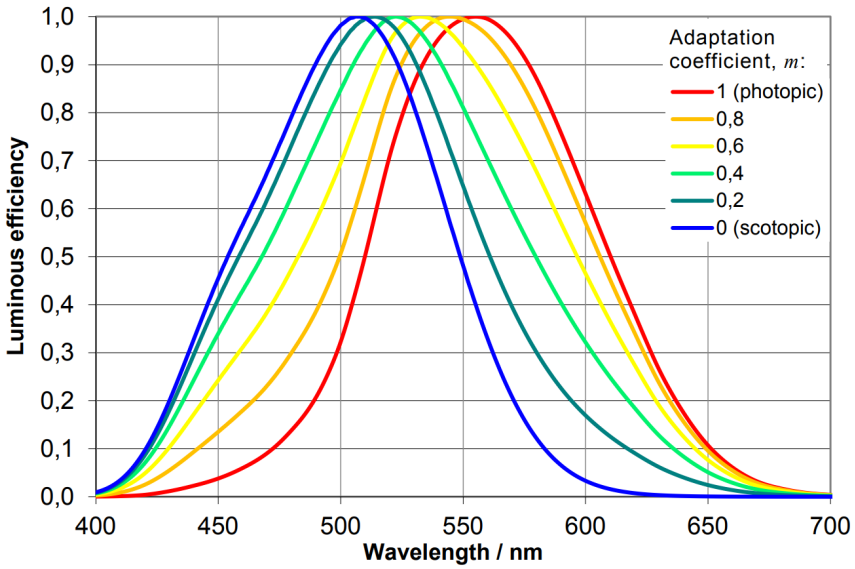

Figure 5.6: Luminous efficiency function for different levels of adaptation of the human visual system. Figure extracted from [Goo+16].

Blue Shift

In artistic fields such as painting or cinema, **night scenes have been traditionally depicted with a slight tendency to favor blue tones**. This is not a mere artistic convention but rather a representation of what is believed to be an actual perceptual effect of scotopic vision. [Jen+00] [TSF02] Figure 5.7 shows an example of a painting having a strong blue shift.

Figure 5.7: *The Starry Night* by Vincent Van Gogh.

Loss of Visual Acuity

Due to the lowest density of rods in the central part of the retina, **the visual acuity of scotopic vision is greatly reduced** [TSF02]. Visual acuity quantifies the ability of the visual system to resolve spatial detail and it is often clinically measured with a Snellen chart [Fer+96] as shown in Figure 5.8.

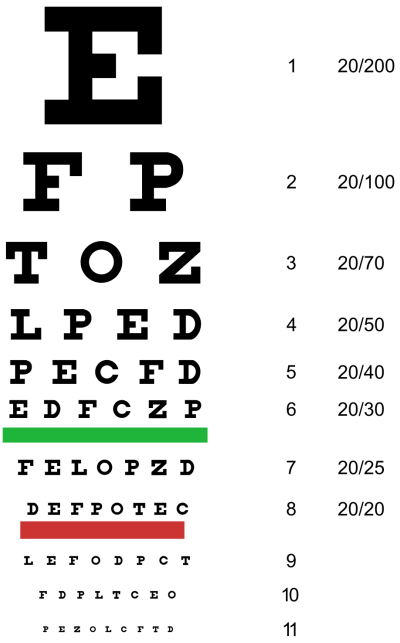

Figure 5.8: Snellen chart. This chart is designed to be viewed at a standard distance of 20ft. Observer acuity is then determined by the smallest row that it can read accurately. Normal acuity is represented by row 8 with a 20/20 qualification and from there other acuity values such as 20/40 mean that the observer has to be at a distance of 20ft in order to accurately distinguish letters that a person with normal acuity would be able to see at 40ft. Loosely speaking this observer is said to have half the normal acuity. Photo by Jeff Dahl with CC BY-SA 3.0 license.

Visual Noise

Another perceptual effect of scotopic vision that is discussed in [TSF02] is the **presence of noise**. This effect is not very well understood since there are many sources where the noise could be originated. This noise is modeled as additive.

5.2 My Approach

In order to both solve the tone reproduction problem and simulate the aforementioned characteristic traits of scotopic vision, there exist several successful approaches such as the ones presented in [Fer+96] and [LRP97]. These approaches are able to simulate a subset of the perceptual effects of scotopic and/or photopic vision, as well as determine the adaptation level required for the scene and tonemap the scene according to it. The downside of these approaches is that they both require building in real-time the histogram of the luminances of the scene, which is not a trivial task to do. For my implementation, I chose a simpler approach based on the works by Thompson et al. [TSF02] and Hellsten [Hel07]. This approach works as a relatively simple **post-processing algorithm** and so can be easily implemented in any existing renderer.

The input for the Thompson post-processing algorithm is an already displayable RGB image i.e. it has **already been tonemapped** using one of the several tonemapping operators available out there. For my implementation, I will use the simple, yet effective Reinhard operator introduced in [Rei+02]. This tonemapping operator comes in two variants, a simple variant given by

$$L_d = \frac{L}{1 + L} \tag{5.1}$$

where L is the luminance of a pixel and L_d its tonemapped luminance (the subscript $\cdot_d$ standing for "display"), and an extended variant given by

$$L_d = \frac{L \cdot \left(1 + \frac{L}{L_{\text{white}}^2}\right)}{1 + L} \tag{5.2}$$

where L_{white} is a configurable parameter that gives more control over the final result by burning luminances over that of L_{white}. Observe how the formulation from Equation 5.2 is indeed an extension of Equation 5.1 as they become equal for $L_{\text{white}} \to +\infty$. In my implementation, I will use the extended operator as it provides greater control at almost no extra cost. For even finer control over the final image, before applying the Reinhard tonemapping operator, the luminance L can be multiplied by a constant k which acts as an **exposure adjustment constant**.

Note that this operator **works on the luminance of the pixel rather than its individual RGB channels**, thus given a color C represented by the three vector component (R, G, B), the correct way of computing the tonemapped color C_d (again, the subscript $\cdot_d$ stands for "display" but can also be interpreted as "day" as we will see later) is by first

computing its luminance as

$$L = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B \tag{5.3}$$

then applying Equation 5.2 to obtain L_d and finally obtain C_d by performing a simple scaling

$$C_d = C \cdot \frac{L_d}{L}. \tag{5.4}$$

After performing these preliminary steps, the actual Thompson post-processing algorithm is applied. The first step is to **compute the scotopic luminance**, here denoted by V. This is done by using the approximation from [LRP97] which requires the input color to be in the CIE XYZ color space. The conversion from RGB space to CIE XYZ is performed by a single matrix multiplication as follows

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{5.5}$$

and once in the CIE XYZ color space, the scotopic luminance V can be approximated by

$$V = Y \cdot \left[1.33 \cdot \left(1 + \frac{Y + Z}{X} \right) - 1.68 \right]. \tag{5.6}$$

Important to note that, as mentioned before, the Thompson algorithm assumes the input image to be already tonemapped, thus all the operations mentioned here are performed on C_d rather than C and the X, Y, Z, R, G, B, V from Equations 5.5 and 5.6 become X_d, Y_d, Z_d, R_d, G_d, B_d, V_d. This scotopic luminance V_d gives us a grayscale image that is then **perturbed by adding some noise**, here denoted by N. In my implementation, I used the 2D noise function from [GM22] which can be animated by modifying the alpha parameter over time. If the 2D noise function of choice does not allow to easily be animated by a parameter, the alternative way of proceeding is by taking slices of a 3D noise function. Finally, to obtain the night image denoted by C_n, the grayscale perturbed image is **multiplied by a constant grayish blue color** C_b, which gives us

$$C_n = (V_d + N) \cdot C_{\mathrm{b}}. \tag{5.7}$$

This is not the final result yet, as in order to properly represent regions of the image that might be in a photopic state such as a region directly lit by a light, this night image C_n **is combined with the regular day image** C_d. This is done by performing a linear interpolation

$$C_f = (1 - n) \cdot C_d + n \cdot C_n \tag{5.8}$$

where C_f represents the final image, and the factor n is a weight in the $[0, 1]$ range that is computed based on the photopic luminance Y_d. If Y_d is above a certain threshold M_p then vision is considered to be under photopic conditions and n is set to 0, if Y_d is below a certain threshold M_s ($M_s < M_p$) then vision is considered to be under scotopic conditions and n is set to 1, otherwise ($M_s < Y_d < M_p$) the vision is considered to be under mesopic conditions and n is linearly interpolated.

Observe how, by construction, this **method correctly simulates the *Purkinje Effect*, the blue shift and the visual noise**, but does not take into account the loss of visual

acuity. The original method as described in [TSF02] and [Hel07] also take the loss of visual acuity into account by performing a Gaussian blur, with the amount of blur dependent on the illumination conditions. For my implementation, I chose not to do this because of the complexity it added.

Chapter 6

Conclusions

Over the course of this thesis, all of the elements that a day and night cycle solution encompasses are presented. Being such a wide variety of complex topics, I have not had the opportunity to delve much into the technical details of each, rather than provide an overview of them. In terms of the solution that I developed, this means that the approaches followed to tackle the problems that have arisen are also not the most advanced ones. It is then possible to consider alternative, more sophisticated approaches in order to improve some aspects of these elements. These are discussed in Section 6.3. Nonetheless, the developed solution, which source code can be consulted on GitHub, is fully functioning and a discussion of the results obtained is presented. As in any computer graphics method discussion, the two most important aspects that have to be discussed are performance, in Section 6.1, and visual results in, Section 6.2.

6.1 Performance

Even though performance has not been extensively evaluated, in the PC used for developing and testing the solution that consists of an *Nvidia GTX 970* GPU, an *Intel i7 6700K* CPU and 16GB of RAM[1], executing the whole pipeline (computing celestial bodies positions, rendering atmosphere, rendering celestial bodies and finally post-processing) at the standard 1920x1080 resolution performs well beyond the usual 60 FPS target and, in fact, stays consistently over 120 FPS. Which makes the solution presented very good in terms of performance for real-time rendering contexts. The only aspect of the solution that does not work in real-time is the change of some atmospheric parameters which require the recomputation of the LUTs, a process that takes just 1 to 2 seconds in the test PC. As explained in [Hil20], it is possible to make this work in real-time by time slicing the update of the LUTs which comes at the expense of some visual inconsistencies.

[1] As of the writing of the thesis, these specifications imply a mid-range PC in terms of performance.

6.2 Visual Results

In terms of the visual results, the implemented solution works well and provides very convincing results, especially in daytime scenes. In nighttime scenes the results provided are not so convincing which I believe can be mainly attributed to two main factors: first, the approach used for rendering the stars which is very simple and not well calibrated due to the issue mentioned in Section 4.2 and secondly to the simple tonemapping approach used. Another situation in which the implementation does not shine is when providing some extreme values to the atmospheric parameters, which causes visible artifacts near the horizon due to numerical precision issues. This is a known issue of the Bruneton model (and other methods that rely on LUTs) as explained in [Hil20].

Find below some images that present the visual results.

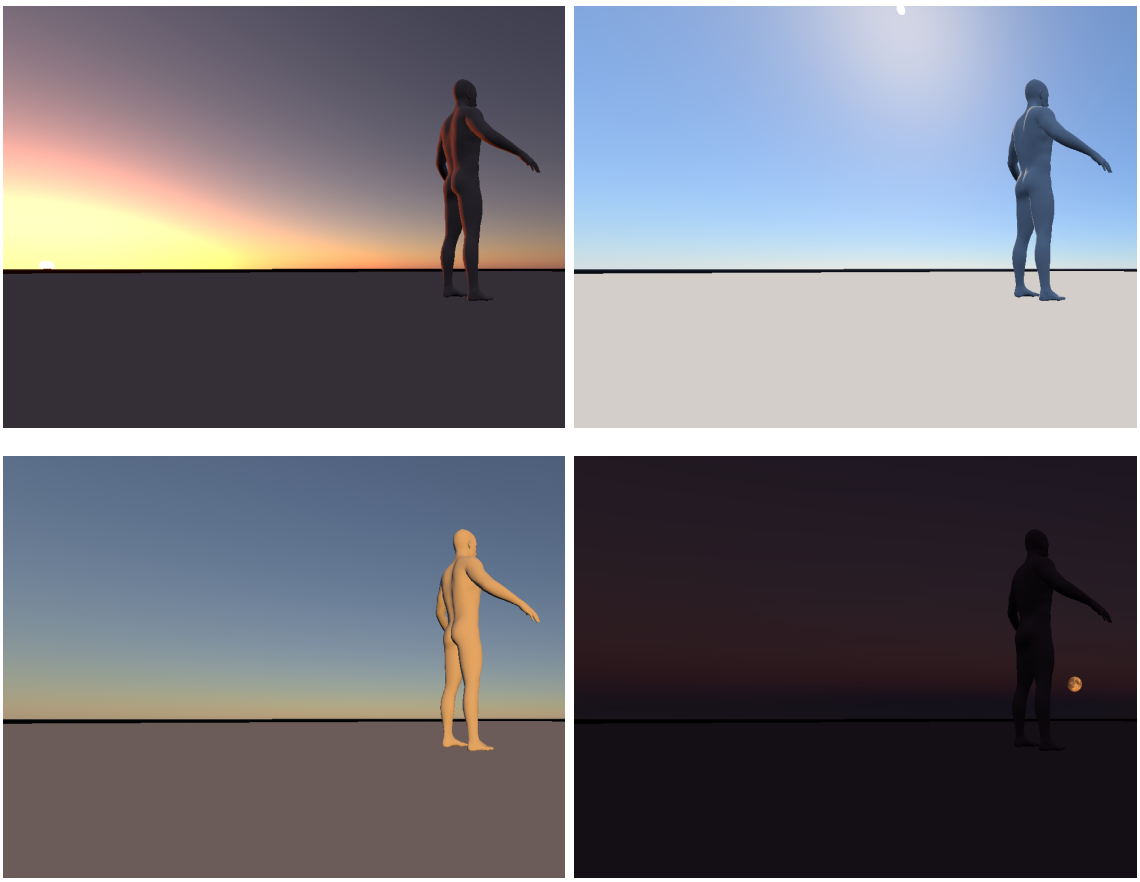

Figure 6.1: Earth atmosphere at different times of the day. Top left: Dawn. Top right: Noon. Bottom left: Before dusk. Bottom right: After dusk.

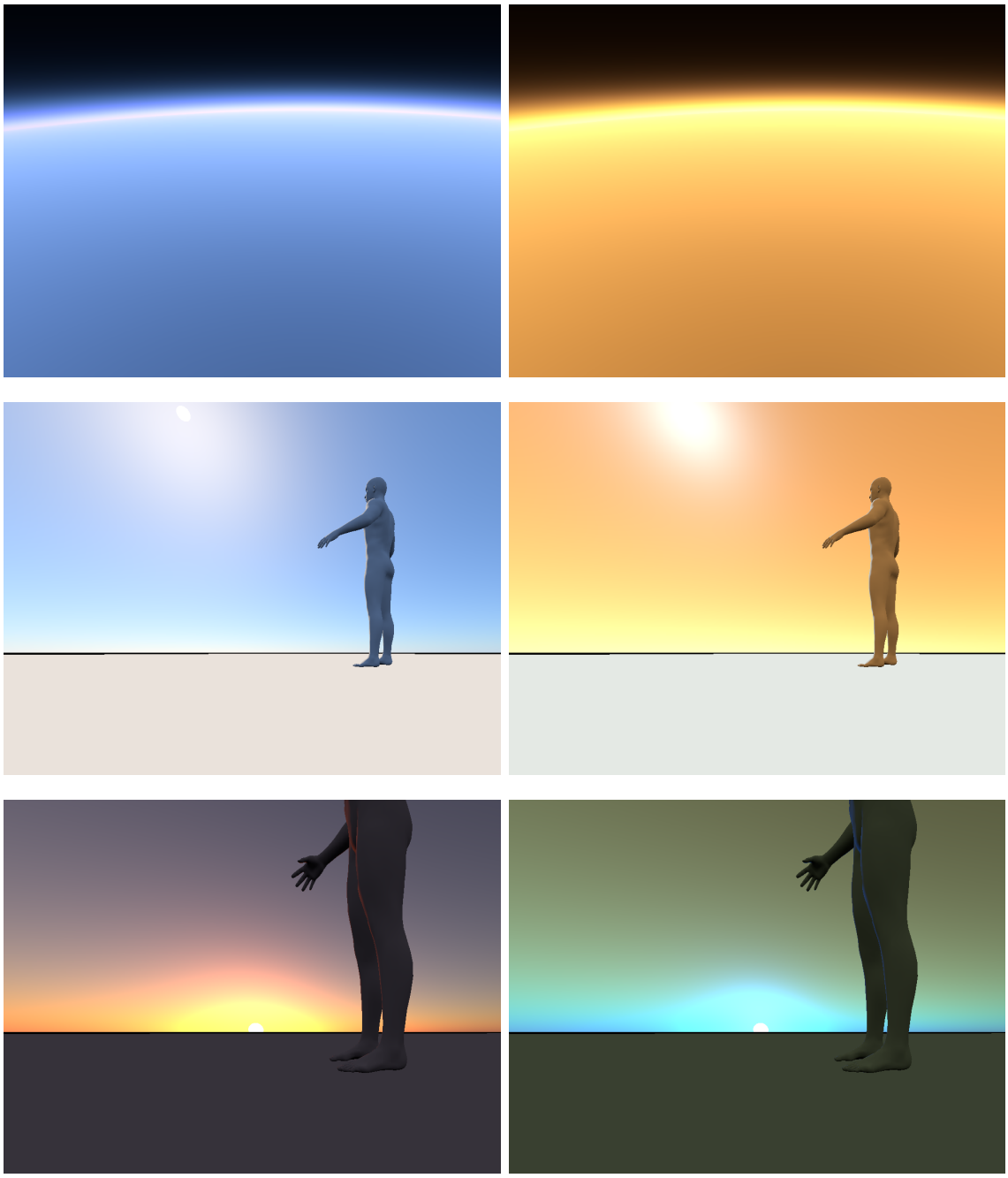

Figure 6.2: Earth atmosphere and custom atmosphere comparison. Left: Earth atmosphere. Right: Custom atmosphere. From top to bottom: View from outside the atmosphere and views from inside the atmosphere at noon and dusk.

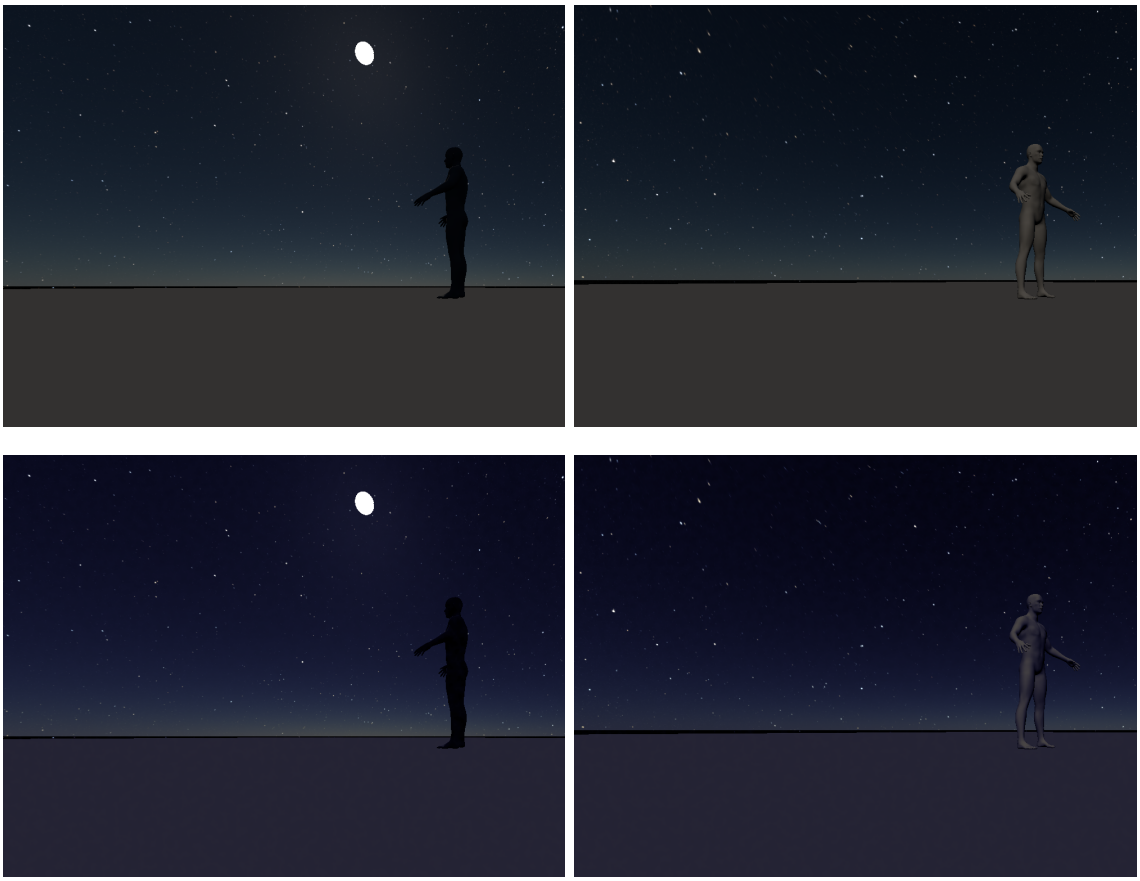

Figure 6.3: Different views of Earth atmosphere at nighttime. Top: Before applying Thompson post-processing algorithm. Bottom: After applying Thompson post-processing algorithm.

Figure 6.4: Comparison of Sun limb darkening models. Left: None. Center: [Nec96]. Right: [HM98]. Note that the effect is very subtle and only noticeable at very low levels of exposure.

Figure 6.5: Different phases of the Moon viewed from the Earth atmosphere. Top: Nighttime. Bottom: Daytime. Note the effect of earthshine in the bottom left image.

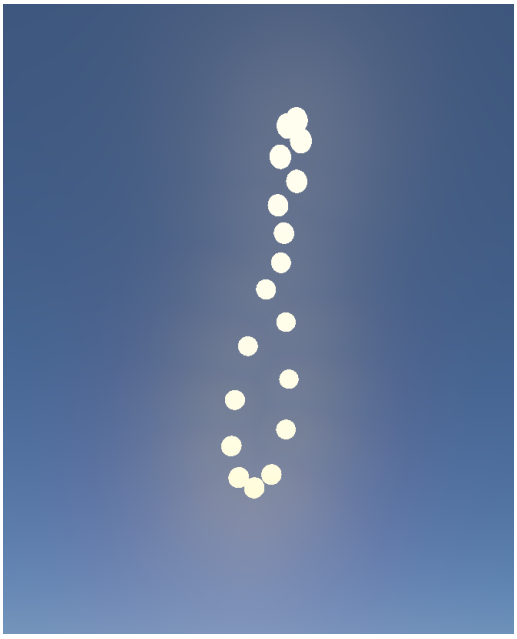

Figure 6.6: Superimposed images showing the position of the Sun at the same time of the day at 18 evenly spaced days along a year. The infinity shaped curve described is known as analemma.

6.3 Future Work

When it comes to the accuracy of the atmosphere rendering model, there are not many improvements that could be made. It would still be interesting to explore novel approaches such as the one used in *Unreal Engine* [Hil20], where no high dimensional LUTs are required and atmospheric parameters can be changed in real-time.

When positioning celestial bodies, I use in my implementation a fixed ΔT value which works well enough for dates around the year 2022. However, some applications such as the prediction/rendering of eclipses, require not only extreme accuracy but also formulas that work for extended time periods. In that case, some model for the computation of ΔT should be introduced in the implementation, such as the one in [Esp]. Another aspect that has not been dealt with in the implementation is the orientation of the Moon. Even though the Moon is said to be tidally locked to the Earth i.e. it is always showing the same face, there are small rotations called librations that cause small changes in the visible face of the Moon. These librations have not been taken into account in my implementation but are responsible for showing an extra 9% of the total lunar surface [Jen+01].

The rendering of stars by means of HDR panoramic textures is one of the weak points of the solution developed. Even though these are correctly positioned in the sky, there are several problems that arise from the usage of HDR panoramic textures. As mentioned before, the physical unit used in the textures is not known thus physical accuracy is lost in favor of artistic control and its intensity has to be adjusted in terms of what "looks good". Moreover, the projections used to map the stars to the panoramic textures have singularities at the poles that cause visible artifacts. Last but not least, its usage gives the sky a very static feeling that does not match the real feeling of the sky, where the stars show a scintillating effect i.e. small fluctuations in their perceived color and brightness. All of these aspects might be improved by using the approach from [MED12], where stars are rendered individually as billboards, and their position, color and intensity are based on the data from the *Yale Bright Star Catalogue* [HJ91].

Finally, in terms of the rendering of nighttime scenes, the simple approach presented works well and increases the "night feeling" of the final images, but still requires the manual tweaking of exposure. Implementing an automatic exposure adjustment system would be a requirement in order to make the solution production ready. Further improvements could be made by implementing one of the histogram-based methods such as [Fer+96], [LRP97] or [DD00].

It is important to note that some of the alternative approaches discussed in this section only suppose minor improvements that for some applications such as videogames might not be required or even noticeable, but for other applications such as simulators might be desirable.

Bibliography

[Kaj86] James T. Kajiya. "The Rendering Equation." In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, pp. 143–150. ISBN: 0897911962. DOI: 10.1145/15922.15902.

[HJ91] Dorrit Hoffleit and Carlos Jaschek. *The Bright Star Catalogue*. 1991.

[Fer+96] James A. Ferwerda et al. "A Model of Visual Adaptation for Realistic Image Synthesis." In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 249–258. ISBN: 0897917464. DOI: 10.1145/237170.237262.

[Nec96] Heinz Neckel. "On the wavelength dependency of solar limb darkening ($\lambda\lambda$303 to 1099 nm)." In: *Solar Physics* 167.1 (Aug. 1996), pp. 9–23. ISSN: 1573-093X. DOI: 10.1007/BF00146325.

[ESA97] ESA. *The Hipparcos and Tycho Catalogues*. 1997. URL: https://www.cosmos.esa.int/web/hipparcos/catalogues (visited on 09/21/2022).

[LRP97] G.W. Larson, H. Rushmeier, and C. Piatko. "A visibility matching tone reproduction operator for high dynamic range scenes." In: *IEEE Transactions on Visualization and Computer Graphics* 3.4 (1997), pp. 291–306. DOI: 10.1109/2945.646233.

[HM98] D. Hestroffer and C. Magnan. "Wavelength dependency of the Solar limb darkening." In: *Astronomy and Astrophysics* 333 (Apr. 1998), pp. 338–342.

[DD00] Frédo Durand and Julie Dorsey. "Interactive Tone Mapping." In: *Rendering Techniques 2000*. Ed. by Bernard Péroche and Holly Rushmeier. Vienna: Springer Vienna, 2000, pp. 219–230. ISBN: 978-3-7091-6303-0. DOI: 10.1007/978-3-7091-6303-0_20.

[Jen+00] Henrik Wann Jensen et al. *Night Rendering*. Tech. rep. UUCS-00-016. School of Computing, University of Utah, 2000. URL: https://www.cs.utah.edu/docs/techreports/2000/pdf/UUCS-00-016.pdf.

[Jen+01] Henrik Wann Jensen et al. "A physically-based night sky model." In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001* (2001), pp. 399–408. DOI: 10.1145/383259.383306.

[Ber+02] Emmanuel Bertin et al. “The TERAPIX Pipeline.” In: *Astronomical Data Analysis Software and Systems XI*. Ed. by David A. Bohlender, Daniel Durand, and Thomas H. Handley. Vol. 281. Astronomical Society of the Pacific Conference Series. Jan. 2002, p. 228.

[Rei+02] Erik Reinhard et al. “Photographic Tone Reproduction for Digital Images.” In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 267–276. ISBN: 1581135211. DOI: 10.1145/566570.566575.

[TSF02] William Thompson, Peter Shirley, and James Ferwerda. “A Spatial Post-Processing Algorithm for Images of Night Scenes.” In: vol. 7. Oct. 2002. ISBN: 978-1-56881-246-5. DOI: 10.1080/10867651.2002.10487550.

[Hel07] Jonas Hellsten. “Evaluation of tone mapping operators for use in real time environments.” MA thesis. Linköping University, Department of Science and Technology, 2007. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-9749.

[BN08] Eric Bruneton and Fabrice Neyret. “Precomputed atmospheric scattering.” In: *Computer Graphics Forum* 27.4 (2008), pp. 1079–1086. ISSN: 14678659. DOI: 10.1111/j.1467-8659.2008.01245.x.

[Jar08] Wojciech Jarosz. “Efficient Monte Carlo Methods for Light Transport in Scattering Media.” PhD thesis. UC San Diego, Sept. 2008. URL: https://cs.dartmouth.edu/wjarosz/publications/dissertation/.

[Mel09] Axel Mellinger. “A Color All-Sky Panorama Image of the Milky Way.” In: *Publications of the Astronomical Society of the Pacific* 121.885 (2009), pp. 1180–1187. ISSN: 0004-6280. DOI: 10.1086/648480.

[EK10] Oskar Elek and Petr Kmoch. “Real-time spectral scattering in large-scale natural participating media.” In: *Proceedings - SCCG 2010: 26th Spring Conference on Computer Graphics* May 2010 (2010), pp. 77–84. DOI: 10.1145/1925059.1925074.

[Gen10] Robert Harry van Gent. *Delta T: Terrestrial Time, Universal Time and Algorithms for Historical Periods*. 2010. URL: https://webspace.science.uu.nl/~gent0113/deltat/deltat.htm (visited on 09/21/2022).

[MED12] Daniel Müller, Juri Engel, and Jürgen Döllner. “Single-Pass Rendering of Day and Night Sky Phenomena.” In: *Vision, Modeling and Visualization*. Ed. by Michael Goesele et al. The Eurographics Association, 2012. ISBN: 978-3-905673-95-1. DOI: 10.2312/PE/VMV/VMV12/055-062.

[Agr15] Dulli Agrawal. “Apparent magnitude of earthshine: A simple calculation.” In: *European Journal of Physics* 37 (Aug. 2015). DOI: 10.1088/0143-0807/37/3/035601.

[Goo+16] T.M. Goodman et al. *The Use of Terms and Units in Photometry – Implementation of the CIE System for Mesopic Photometry*. Tech. rep. 004:2016. Commission Internationale de l’Eclairage, 2016. URL: http://files.cie.co.at/841_CIE_TN_004-2016.pdf.

[Hil16] Sebastien Hillaire. "Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite." In: *SIGGRAPH 2016 Course* (2016). URL: https://www.ea.com/frostbite/news/physically-based-sky-atmosphere-and-cloud-rendering.

[Bru17a] Eric Bruneton. "A Qualitative and Quantitative Evaluation of 8 Clear Sky Models." In: *IEEE Transactions on Visualization and Computer Graphics* 23.12 (2017), pp. 2641–2655. ISSN: 10772626. DOI: 10.1109/TVCG.2016.2622272. eprint: 1612.04336.

[Bru17b] Eric Bruneton. *Precomputed Atmospheric Scattering: a New Implementation*. 2017. URL: https://ebruneton.github.io/precomputed_atmospheric_scattering/ (visited on 08/30/2022).

[McA18] Stephen McAuley. "The Challenges of Rendering an Open World in Far Cry 5." In: *SIGGRAPH '18: ACM SIGGRAPH 2018 Courses*. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018. ISBN: 9781450358095. URL: https://advances.realtimerendering.com/s2018/index.htm.

[Bau19] Fabian Bauer. "Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution." In: *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450363075. DOI: 10.1145/3305366.3335036. URL: https://advances.realtimerendering.com/s2019/index.htm.

[Stu19] NASA Scientific Visualization Studio. *CGI Moon Kit*. 2019. URL: https://svs.gsfc.nasa.gov/4720 (visited on 09/19/2022).

[dfr20] Dario "dfranx". *SHADERed - Free and open source shader editor*. 2020. URL: https://shadered.org/ (visited on 09/19/2022).

[Hil20] Sébastien Hillaire. "A Scalable and Production Ready Sky and Atmosphere Rendering Technique." In: *Computer Graphics Forum* 39.4 (2020), pp. 13–22. DOI: https://doi.org/10.1111/cgf.14050.

[Stu20] NASA Scientific Visualization Studio. *Deep Stars Map 2020*. 2020. URL: https://svs.gsfc.nasa.gov/4851 (visited on 09/19/2022).

[NAS21] NASA. *FITS Support Office*. 2021. URL: https://fits.gsfc.nasa.gov/fits_home.html (visited on 08/30/2022).

[Pat21] Jasmin Patry. "Real-Time Samurai Cinema: Lighting, Atmosphere, and Tonemapping in Ghost of Tsushima." In: *SIGGRAPH '21: ACM SIGGRAPH 2021 Courses*. Virtual Event, USA: Association for Computing Machinery, 2021. ISBN: 9781450383615. URL: https://advances.realtimerendering.com/s2021/index.html.

[Wil21] David R. Williams. *NASA Planetary Fact Sheet*. 2021. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/ (visited on 09/19/2022).

[GM22] Stefan Gustavson and Ian McEwan. "Tiling simplex noise and flow noise in two and three dimensions." In: *Journal of Computer Graphics Techniques (JCGT)* 11.1 (Feb. 2022), pp. 17–33. ISSN: 2331-7418. URL: http://jcgt.org/published/0011/01/02/.

[Esp] Fred Espenak. *Eclipse Predictions*. URL: https://eclipse.gsfc.nasa.gov/eclipse.html (visited on 09/21/2022).

Appendix A

Axel Mellinger's Milky Way Panorama 2.0

The *Axel Mellinger's Milky Way Panorama 2.0* (*mwpan2* for short) is another of these HDR panoramic images of the night sky similar to the ones I used in my implementation for rendering the stars, which were provided by the NASA [Stu20]. What makes this panorama stand out among the others is its high resolution, high dynamic range and photometric accuracy [Mel09]. A downloadable version of this HDR panorama is available at its website, unfortunately, it is provided in a non-standard image format named FITS which makes it harder to integrate into computer graphics applications. The FITS (Flexible Image Transport System) file format [NAS21] is a data format used by scientists to store astronomical data, it allows not only to store 2D images but all kinds of multidimensional astronomical data whether it be a 1D spectrum, a 3D data cube or higher dimensional data. Besides the data itself, FITS files also allow to store additional metadata in its headers, in the case of the *mwpan2*, it contains a World Coordinate System (WCS) header that specifies which type of projection and coordinate system has been used to map the data into the 2D image. As specified in this header, the *mwpan2* is provided as an equirectangular projection of galactic coordinates [1]. Using the specialized software SWarp [Ber+02] and thanks to the WCS header, it is possible to reproject the *mwpan2* data into a more familiar combination of projection and coordinate system. In particular, it can be reprojected into equatorial coordinates which makes it then straightforward to use with the same rendering technique described in Section 4.2. Figure A.1 shows both the original data and the reprojected data. As can be noted, the reprojected version contains some missing data which correspond to the north and south poles of the galactic coordinate system.

[1] The galactic coordinate system is yet another astronomical coordinate system similar to the ecliptic and equatorial coordinate systems discussed in Chapter 3.

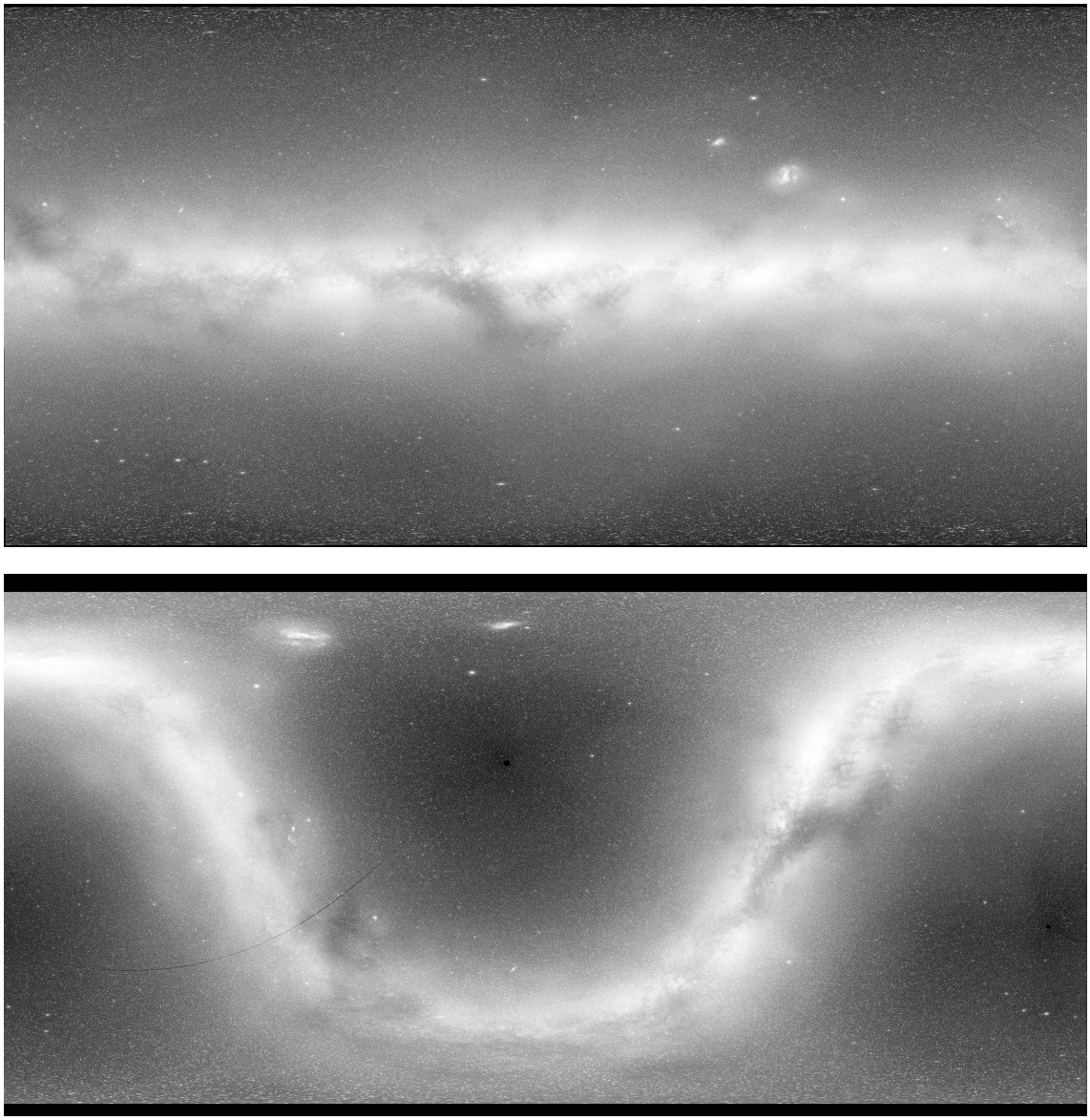

Figure A.1: Tonemapped image of the red channel of *mwpan2*. Top: Original. Bottom: Reprojection to equatorial coordinates performed in SWarp.

Appendix B

Default Atmospheric Parameters

Parameter	**Value**	**Units**	**Source**
Planet Radius	6360	km	[Hil20]
Atmosphere Height	100	km	[Hil20]
Ground Albedo	(0.3, 0.3, 0.3)	-	[Hil20]
Sun Irradiance	See $E_{S\to E}$ in Section 4.1.2	$\mathrm{W}\cdot\mathrm{m}^{-2}$	[Jen+01]
Moon Irradiance	See $E_{M\to E}$ in Section 4.1.2	$\mathrm{W}\cdot\mathrm{m}^{-2}$	[Jen+01]
Rayleigh Scattering	$(5.802\cdot10^{-3}, 13.558\cdot10^{-3}, 33.100\cdot10^{-3})$	km^{-1}	[Hil20]
Rayleigh Density	$e^{\frac{-h}{8000}}$	-	[Hil16]
Mie Scattering	$(3.996\cdot10^{-3}, 3.996\cdot10^{-3}, 3.996\cdot10^{-3})$	km^{-1}	[Hil20]
Mie Absorption	$(0.444\cdot10^{-3}, 0.444\cdot10^{-3}, 0.444\cdot10^{-3})$	km^{-1}	[Hil20]
Mie Phase Function	0.8	-	[Hil20]
Mie Density	$e^{\frac{-h}{1200}}$	-	[Hil16]
Ozone Absorption	$(0.650\cdot10^{-3}, 1.881\cdot10^{-3}, 0.085\cdot10^{-3})$	km^{-1}	[Hil20]
Ozone Density	$\max\left(0, 1-\frac{\lvert h-25\rvert}{15}\right)$	-	[Hil20]