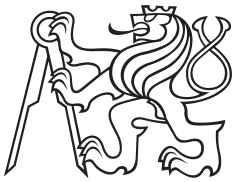
Master's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Night sky rendering

Bc. Michal Pozník

Supervisor: Ing. Jaroslav Sloup
Field of study: Open Informatics
Subfield: Computer Graphics
May 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pozník** Jméno: **Michal** Osobní číslo: **474541**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vykreslování noční oblohy

Název diplomové práce anglicky:

Night Sky Rendering

Pokyny pro vypracování:

Poveďte rešerši existujících metod vykreslování noční oblohy [1-3]. Analyzujte důležité složky modelu noční oblohy (vícenásobný rozptyl světla v atmosféře, hvězdy, měsíc, posun barev do modra) včetně souvisejících metod mapování tónů [4-5].
Navrhněte a implementujte interaktivní aplikaci, která bude vykreslovat noční oblohu v libovolnou roční dobu. Do aplikace zahrňte taktéž simulaci západu slunce. Pro zrychlení simulace se pokuste co možná nejvíce dat předpočítat, aby vykreslování probíhalo v reálném čase. Inspiraci lze čerpat z článků [6-7].
Vygenerované obrázky porovnejte s reálnými fotografiemi noční oblohy. Vyhodnoťte rychlost implementované metody a její paměťovou složitost.
Implementaci proveďte v C/C++ s využitím OpenGL.

Seznam doporučené literatury:

[1] Henrik Wann Jensen, Fredo Durand, Julie Dorsey, Michael M. Stark, Peter Shirley, Simon Premože: A Physically-based Night Sky Model. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 399-408, 2001.
[2] Daniel Müller, Juri Engel and Jürgen Döllner: Single-Pass Rendering of Day and Night Sky Phenomena. 17th International Workshop on Vision, Modeling and Visualization, 2012.
[3] Jorg Haber, Marcus Magnor, Hans-Peter Seidel: Physically-based Simulation of Twilight Phenomena. ACM Transactions on Graphics, vol.24, no.4, pp. 1353-1373, 2005.
[4] Robert Wanat, Rafal K Mantiuk: Simulating and Compensating Changes in Appearance Between Day and Night Vision. ACM Transactions on Graphics, vol.33, no.4, pp.1-12, 2014.
[5] Saad Masood Khan, Sumanta N. Pattanaik: Modeling Blue Shift in Moonlit Scenes by Rod Cone Interaction. Journal of Vision, vol.4, no.8, p.316a, 2004.
[6] Sébastien Hillaire: A Scalable and Production Ready Sky and Atmosphere Rendering Technique. Computer Graphics Forum, vol.39, no.4, p.13-22, 2020.
[7] Eric Bruneton, Fabrice Neyret: Precomputed Atmospheric Scattering. Computer Graphics Forum, vol.27, no.4, p.1079-1086, 2008.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jaroslav Sloup Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **01.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Jaroslav Sloup podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Acknowledgements

I would like to express gratitude to the supervisor of this thesis, Ing. Sloup, for his guidance through it. Despite the number of theses under his supervision and the workload outside that, he regularly found time to review the progress and offer advice on difficulties. I would further like to thank RNDr. Ingrid Nagyová PhD. for her suggestion that saved me some time and my classmate with a similar thesis topic Matěj Sakmary, with whom we have discussed problems and obstacles along the way. Last but certainly not least, I would like to thank my family, who helped fine-tune some aspects of the application.

Declaration

I declare to have prepared the submitted work independently and listed all the used literature and data sources.

Prague, 20. May 2022

Abstract

The matter of this thesis is a rendering OpenGL application written in C++. The application aims to render the sky in real-time, including all its essential features, and can also render the sky at any given moment and from any given location on Earth. The application renders the Sun, the Moon, the stars, the background of the stars and maps the color tones so that the scene would feel more like at night. Additionally, the scattering of light in the atmosphere is approximated, so several further phenomena can be observed in the app, like dusk, dawn, and the blue color of the sky during the day. The user can change the parameters influencing the location of the rendered objects on run-time. The application uses freeglut and glew.

Keywords: night, sky, Moon, stars, twilight, rendering

Supervisor: Ing. Jaroslav Sloup

Abstrakt

Předmětem práce je vykreslovací OpenGL aplikace napsaná v C++. Účelem aplikace je vykreslování oblohy v reálném čase včetně všech jejích důležitých prvků. Aplikace je schopna vykreslit oblohu v libovolný okamžik a z libovolné pozice na Zemi. Aplikace vykresluje Slunce, Měsíc, hvězdy, hvězdné pozadí a mapuje barevné tóny, aby scéna více evokovala noc. K tomu se aproximuje také rozptyl světla v atmosféře, v aplikaci lze tedy pozorovat soumrak, rozbřesk, ale i modré zbarvení oblohy během dne. Uživatel může za běhu aplikace měnit parametry ovlivňující pozice těles na obloze a výsledný vzhled oblohy. Aplikace používá freeglut a glew.

Klíčová slova: noc, obloha, Měsíc, hvězdy, soumrak, vykreslování

Překlad názvu: Vykreslování noční oblohy

Contents

Figures

Chapter 1

Introduction

The matter of this thesis is an OpenGL application (hereinafter referred to as the app). The app renders the sky with all the features the average person would expect on a clear day or night. Accurate sky rendering is essential in many applications that wish to simulate believable and immersive environments. From games, which build immersive worlds and give artists the freedom to create environments beyond imagination, to educational material that wishes to show accurate information and use it for teaching, a good and effective rendering technique is required for the sky to appear believable and not break immersion. Furthermore, the ability to render the night sky and twilight phenomena are worth the effort just for their perceived beauty.

Recent research has mainly focused on specific features, like stars and Moon[JDD+01, MED12], light scattering in the atmosphere[BN08, Hil20], or the blue shift of human vision in nighttime conditions[JPS+00, KP04]. However, not much previous work has combined all features of the day and night sky in one application. This thesis aims to compose these techniques to create a more comprehensive simulation of the sky phenomena.

The app renders the Sun, the Moon, the stars, the background light of the night sky, with most of the contribution collectively coming from stars in the Milky Way that are too dim to be seen individually, simulates the scattering of light in the atmosphere, and maps the color tones toward blue when the Sun is below the horizon to add more night-like feel to the scene. All of these features are rendered separately and are mixed together to form the final image. The tone mapping is applied to the whole image at the end after everything has been rendered.

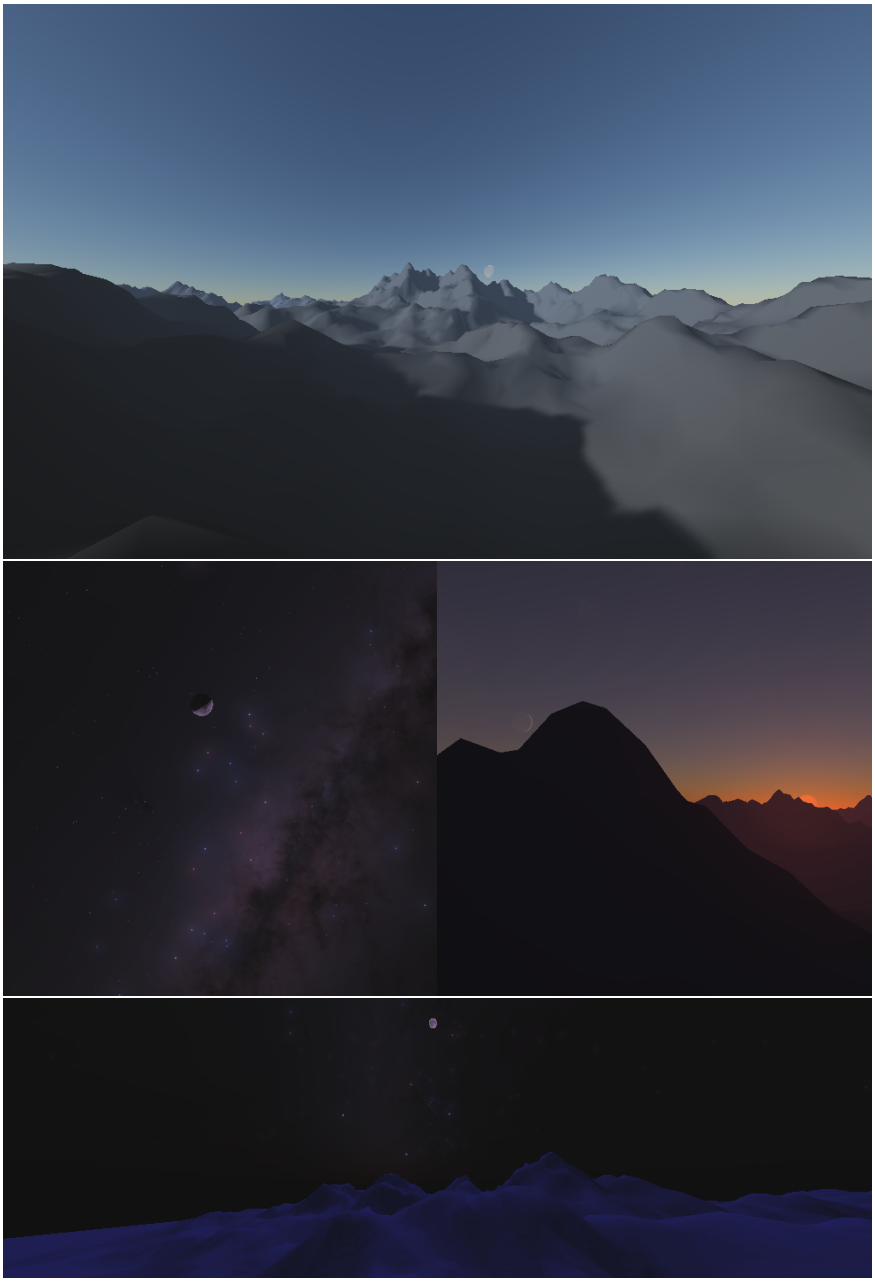


Figure 1.1: Several renders of the app in differing times of day and year and on different locations on Earth.

The app is written in C++ and utilizes freeglut and glew.

1.1 Overview

Theoretical description of the focus of this thesis will be described in Chapter 2. Previous work done in the field on individual features will be reviewed in Chapter 3. Following that, the proposed solution of individual rendered features will be presented in Chapter 4, mostly put together from previous work. Chapter 5 will then describe the final implementation of the app. The results will be discussed in Chapter 6. The thesis concludes in Chapter 7.

There are several appendices attached to this thesis, most notably Appendix B, containing the astronomical algorithms. The sheer volume of the equations warrants a description of its own. The text of the thesis is less disrupted as a result and the calculations are more compact, which helps avoid duplicates, since some terms are used throughout.

Chapter 2

Theoretical description

This chapter will go over the observable features the app will focus on. Namely the Sun, the Moon, the stars, tonemapping, and the atmosphere. These will be described in theory, along with the features within them that we see, and what about them is important to include so as to not break the impression. Some of the sections will for completeness go into more detail than what the app renders.

2.1 Sun

The Sun is by far the brightest object we observe. Life as we know it owes its entire existence to the heat and light the Sun provides. The Sun is really just a star like many others we can see at night, but its proximity means the power received is much higher compared to the others. So much in fact that we can feel the incoming heat on our skin when standing out in the sun.

The light from the Sun drowns out most other sources of light, and so during the day, it remains the only visible object outside of our atmosphere, other than the Moon. This is because even the light scattered in Earth's atmosphere, which is a fraction of the incoming power of the Sun, is much brighter than the stars and rivals the light reflected off the Moon. The inclusion of the Sun is vital for any simulation that hopes to present a realistic view of the sky.

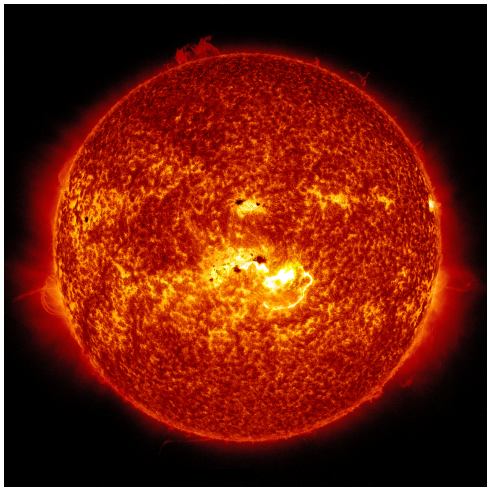


Figure 2.1: Image of the Sun by NASA[1]. Notice the nontrivial surface features with varying brightness.

The Sun does not appear as a sphere with a flat color across its entire surface. Through filters that block enough light to allow us to study it more closely, we can observe dark-colored marks running throughout the surface. In some places, these become large enough to stand out and create much larger areas. These areas are called sunspots and, if large enough, can even be observed without any magnification aides, only with the human eye through a light blocking filter. These dark regions are caused by lower temperatures of the solar surface, usually due to the higher intensity of the Sun's magnetic field. In Figure 2.1, the sunspots can be seen as dark regions in the middle and on the left side of the image.

There is another phenomenon that changes the perceived color of the surface, solar flares, seen in Figure 2.1 as noticeably bright spots in the middle and on the far right side of the solar disk. Flares appear as bright flashes of light and release much radiation energy. Flares happen as a consequence of the Sun's ever-changing magnetic field when the field's lines quickly reorganize and "shuffle". Another manifestation of the Sun's complex magnetic field is the corona. Streams of charged particles create streamlines that follow the field's lines. The corona is among the most visually exciting phenomena of the Sun. However, its brightness is much lower than the solar surface, so under usual conditions, it is unobservable by the human eye due to its proximity to the Sun. In Figure 2.1, the corona can partially be seen as a faint glow around the disk, but it quickly loses brightness with distance from the Sun's surface. Additionally, the local brightness of the solar disc is not constant but falls off toward the edge.

[1]Article by NASA: https://solarsystem.nasa.gov/news/268/10-things-june-12-nasas-first-mission-to-touch-the-sun/ [cit. 2022-05-05]

(a) : Image of the Moon by NASA[3]. Notice the difference in crater density between the maria (dark spots) and poles.

(b) : Image of the Moon by NASA[4]. Earthshine's effect can be seen well, due to adjusted exposure of the camera.

Figure 2.2: Images showing the Moon's appearance at different phase angles.

Another factor influencing the perceived appearance of the Sun is the atmosphere. Depending on the conditions, the atmosphere can only change the color of the Sun's disk or distort its shape. The amount of light scattered depends on the wavelength, but mostly blue light is scattered (as will be explained later), so the color is shifted toward red, most noticeably at low elevations. Different temperatures and densities have different refraction indexes, so the light coming from the Sun does not follow a straight line. The more subtle consequence of this is that the Sun disk appears shifted in place, the other extreme side of the spectrum are effects more akin to a mirage that warps the shape, for example red flash, green flash, and omega Sun[HM05]. These effects are much more noticeable near the horizon, where the light has to travel longer distances through the denser parts of the atmosphere.

For more information, see NASA's in-depth article about the Sun[2].

2.2 Moon

The Moon is a natural satellite of the Earth, and if it is visible during the night, it is the dominant feature of the sky. The Moon is also bright enough to be visible during the day, but due to the overpowering brightness of the

[2]Sun in depth by NASA: https://solarsystem.nasa.gov/solar-system/sun/in-depth/ [cit. 2022-05-05]

[3]Article by NASA: https://solarsystem.nasa.gov/news/411/nasas-exploration-campaign-back-to-the-moon-and-on-to-mars/ [cit. 2022-05-05]

[4]Article by NASA: https://apod.nasa.gov/apod/ap120324.html [cit. 2022-05-05]

Sun, it does not stand out as much as during nighttime observations.

The Moon does not emit any light of its own; what we see is reflected sunlight. This reflected light, however, is much weaker than direct illumination, so it does not drown out the light of other objects in the sky. Therefore, even at the brightest during a full moon, other stars and planets are still visible. This fact allows us to see more features of the Moon with our own unaided eyes, and it is easier to notice how much the Moon changes with time.

Since sunlight is coming only from one direction, only one side of the Moon is lit, while the other receives no illumination from the Sun at all. An example can be seen in Figure 2.2a. Additionally, the Moon orbits the Earth at about 27 days per orbit. The combination of both creates what we know as the phases of the Moon. The phases are a cycle in which the illuminated part seems to "travel" along the surface from right to left. The period of this cycle is one synodic month or about 29.5 days.

The lunar surface has a very interesting property. Due to the granularity of the particles and the lack of erosion that would smoothen it out, there is a sharp peak of brightness at full moon, when the direction toward the Sun and Earth coincide. Furthermore, not only does the brightness suddenly peak, but it is at that moment independent of position on the lunar disk. This is called the opposition effect (also opposition brightening), and the principle behind it is still not entirely understood[Hap66].

Another curiosity is the very faint illumination at phase angles close to the new moon. The side facing away from the Sun is not completely black but can still be vaguely made out, as in Figure 2.2b. The Sun is not the only source of light on the lunar surface. Light reflected off the Earth reaches the Moon and allows us to see the surface even though it is in shadow. This light is called earthshine. It does not take much to realize that the Earth and Moon are in the exact opposite phases in relation to each other. Therefore, earthshine is strongest during the new moon and practically nil at the full moon.

The Moon follows an elliptical orbit with about 6.6° obliquity. The sidereal period of this orbit is approximately 27 days, which happens to be the exact same period of the Moon's own rotation. Therefore only one side of the Moon (called the near side) faces the Earth, so the Moon's appearance (other than illumination) remains roughly the same. The orbit, however, is not perfectly circular, which causes the Moon to apparently "wobble." This effect is called librations and results in around 59% of the Moon's surface being visible.

We differentiate three types of librations. Optical librations are the most obvious and are the wobble, as mentioned above. Optical librations are further separated into longitudinal and latitudinal. Longitudinal librations are caused by the eccentricity of the orbit. As the Moon speeds up and slows down, it "comes ahead of" and "falls behind" its own rotation. Latitudinal librations are caused by the obliquity, the angle between the axis of rotation and the orbital plane. The principle is similar to the cycle of seasons on Earth.

The second type of libration is diurnal. As the Earth rotates, the observer's position changes relative to the Moon, and so does the angle at which they are looking at the Moon, which makes it seem to rotate. The effect is strongest at the equator, just as the Moon is rising and setting above the horizon, and is one of the consequences of parallax. The last and least intense type of libration is physical. The Moon does not rotate uniformly, and due to various factors, tiny oscillations occur. These oscillations, which amount to only about $100''$, seen as less than $1''$ from the Earth, are called physical librations.

Another consequence of the ellipticity is oscillations of the apparent size. At its closest, the Moon is at about 89% of the maximum distance, resulting in noticeable optical size changes. Interestingly, the angular size of the Moon is very similar to that of the Sun, which allows it to block out only the solar disk and nothing around it, when the two align (also called a total solar eclipse). Solar eclipses are ideal for observing the solar corona.

The surface of the Moon is not smooth. Rather, it is littered with craters of various sizes. These craters create unevenness that can cast shadows. These shadows can be observed by the human eye, mostly around the terminator (the divide between illuminated and shadowed sides), where the angle of the incoming light is greatest. Interesting to note is that the near side has noticeably fewer craters, especially in the large dark regions called lunar maria.

The Moon sees the same atmospheric effects as the Sun. The lunar disk is shifted away from the horizon, its color is shifted toward red, and in extreme cases, the shape of the disk can change. Moreover, the Moon may seem more blue during the day. The atmosphere's own color is in a sense "overlayed" over objects outside it, so in addition to the light coming from the Moon, we observe the scattered sunlight coming from the same direction.

For more information, see NASA's in-depth article about the Moon[5].

[5]Earth's Moon in depth by NASA: https://solarsystem.nasa.gov/moons/earths-moon/in-depth/ [cit. 2022-05-05]

Figure 2.3: Image of the center of the Milky Way by NASA[6]. Notice how the galaxy is not comprised of individual stars, but seems to be a continuous source of light.

2.3 Stars

Most stars we see as points of light in the sky are actually massive balls of plasma-powered nuclear fusion, similar to our Sun. While many can be much more powerful, they are such distances away that we see them as no more than dots in the night sky. However, not all bright dots we see are really stars. Some among them are objects that produce light in different ways than fusion, like quasars, and some of them produce no light of their own and only reflect light, like planets and nebulas. Depending on the distance and intensity of the emitted/reflected light, we see the stars with varying brightness. Many we cannot even see with unaided eyes and need some method of magnification for their observations.

The most notable non-star point lights of the night sky are planets of our solar system. As the planets and the Sun are all moving within a dynamic system alongside the Earth, their apparent position relative to the stars behind them changes as they travel along the sky. Furthermore, Mercury and Venus, the two planets closest to the Sun, can never "get behind" the Earth, so their path seems to be bound to that of the Sun. On the other hand, planets that are further away seem to draw circles as they follow their paths in the sky. These circles appear as a result of the Earth's own orbit

[6]Article by NASA: https://exoplanets.nasa.gov/milky-way-overlay/ [cit. 2022-05-05]

around the Sun. The planets also go through phases of their own, similarly to the Moon, but since we cannot see them as disks, only as points, this only has the effect of change in brightness. There are other objects in our solar system that we can see at night, from asteroids and satellites to more exciting objects like comets.

Other stars, objects outside our solar system, are also not static and all perform a complex system of movements in space. As a result, they appear to slowly move in relation to each other. Our ancestors have seen the stars in different places, and humans of the future will see a different sky again. The magnitude of this movement is too small to notice in one night or even a year, but over millions of years, the stars will have moved enough for the change to be noticeable, even if we do not consider stars being born and dying off. For example, Barnard's star has the largest observed effect of this at $10.28''$ per year, meaning it travels about one apparent radius of the Moon in a century.

Positions are not the only attribute of the stars that can change. A star can experience processes that affect its brightness in the long term or the short term. Shadowing from its own planets and eruptions on the surface will quickly fade off, but a star reaching the end of its lifespan will affect it long term. There are, however, even more interesting sources of variance. Many stars we see in the sky are actually multi-star systems, where the stars orbiting each other can be several lightyears away from each other. As they move, they get closer and farther away from us, occlude each other, and the point light we see them as seems to change brightness.

There are many more objects we cannot see individually with the unaided eye than those we can. However, that does not mean they do not affect the appearance of the night sky. While their brightness might not be enough on their own, we can observe them collectively as a color background of the stars we do see. The best example is the Milky Way, the galaxy containing our solar system. Comprised of many point sources of light, like stars and nebulae, we only see it as a band of light across the sky with relatively few objects that we can pinpoint. An example of a view of the Milky Way is in Figure 2.3.

Given the varying brightnesses of the stars, it is easy to see them as arranged in the sky into shapes, as if they were dots connected by lines. These shapes are called constellations and are vital for orientation in what might at first seem like a mess of colored dots. Our ancestors saw in them great heroes and monsters; eventually, they became navigation aides. Now they serve to separate the sky into sectors for better orientation. Most people know at least some constellations or individual stars, most notably Ursa Major (also called the Big Dipper or the Plough), Orion and Polaris.

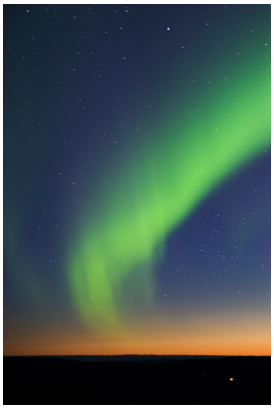
Figure 2.4: Camera image suffering from glare effects by Malaya Optical[7]. Starburst, glare, and lenticular halo can be seen around the headlights of the car. Human eye experiences very similar effects.

Just as a satellite may encounter a source of gravity and fly by on a curved path (e.g., the effect of a gravitational slingshot), so too does light bend around strong enough gravitational sources. Due to the high speed of electromagnetic rays, of which light is a subset, the gravitational pull required to bend their path is so big that only the most massive objects are capable of this effect. This is not as easy to observe even with telescopes, so it is not an effect worth including in simulations of the sky.

The color of the stars is not expressed in rgb values, as would be convenient for the purposes of rendering. Instead, brightnesses are recorded through several filters, most notably B for blue and V for visible (green-ish yellow). From there, the $B - V$ color index is used to quantify the star's color. A value of 0 means that both brightnesses are equal, which is typical for hot stars. Cool stars have a color index of around 2.0.

Even though the stars are far enough to be virtually indistinguishable from points, we see them with some sort of glare or halo. This glare appears due to the scattering of incoming light in the atmosphere and in the eye. In fact, this glow can be seen around any bright object with a comparatively dark

[7]article by Malaya Optical: https://www.malayaoptical.com/what-is-glare/ [cit. 2022-05-06]



(a) : Image of aurora borealis by aurora-borealis.us[9].

(b) : Image of the zodiacal light by Cristoph Stopka[10]. The light appears as a column of light rising from the horizon. Notice the angle to the horizon as well.

Figure 2.5: Images showing the Moon's appearance at different phase angles.

background, like the Sun or streetlamps at night. The effect appears to some degree even in digital cameras, as can be seen in Figure 2.3 around the bright object above the center of the galaxy. A much better example is in Figure 2.4, where all major scattering and diffraction effects can be seen: starburst, lenticular halo, and bloom.

Stars are affected by the atmosphere in the same way as was mentioned with the Sun and Moon, except for the shape distortion since the apparent angular size is so negligible. However, small chaotic changes of the density and temperature have the effect of changing the apparent brightness and color. As a result stars appear to flicker both in intensity and color. This phenomenon is called scintillations and is the strongest close to the horizon, since the light has to travel a long distance through the atmosphere.

For more information see Stars, formation, classification and constellation article at space.com[8], where several aspects are described in more datail, like the lifecycle or classification of stars.

[8]Stars, formation, classification and constellation at space.com: https://www.space.com/57-stars-formation-classification-and-constellations.html [cited 2022-05-05]

[9]Title page of aurora-borealis.us: https://www.aurora-borealis.us/ [cit. 2022-05-06]

[10]Article by earthsky.org: https://earthsky.org/astronomy-essentials/zodiacal-light-false-dusk-how-to-see-explanation/ [cit. 2022-05-06]

2.4 Other lights in the sky

There are several other sources of light in the sky, though they are either very faint or there is only a limited time when they are visible. The most notable of these lights is aurora borealis (aurora australis in the southern hemisphere), also referred to as the northern lights (southern lights). The Sun's activity is not constant and during coronal storms, such as a coronal mass ejection, a huge amount of ionized particles and energy is released out into space. If the Earth is in the way of such storm, the magnetic field deflects some particles away and some others toward the poles. These particles then collide with the Earth's atmosphere and energy is released in the form of visible light. The color of the light depends on the gas the particle has collided with, nitrogen produces blue and purple, whereas oxygen glows red and green.

A great explanation with image visualizations has been created by NASA in the form of a multipage poster[11].

Another of these lights is the so-called zodiacal light. Planets, moons, and asteroids are not the only objects orbiting the Sun. The solar system contains disks of dust particles of sizes from several molecules to 0.1mm. These particles reflect sunlight that we can observe as a column of light when the Sun is just under the horizon. For this reason it is also called the false dawn. Since the dust orbits in the ecliptic plane, the light follows the zodiac, the path the Sun takes among the stars. This can be seen much better during a new moon, as there is less light interference, possibly spanning the entire sky.

An in-depth explanation can be found at earthsky.org[12].

Airglow is the last source of light mentioned here. It is essentially the northern/southern lights outside their expected latitudes, but with much less intensity. Not all particles are deflected by the magnetic field, some hit the atmosphere and light is emitted even far away from the poles. The intensity is, however, much weaker and only appears as a faint, nearly uniform illumination.

[11] Aurora, fabled glowing lights of the Sun-Earth connection by NASA: https://pwg.gsfc.nasa.gov/polar/EPO/auroral_poster/aurora_all.pdf [cit. 2022-05-06]

[12] Zodiacal light: All you need to know article at earthsky.org: https://earthsky.org/astronomy-essentials/everything-you-need-to-know-zodiacal-light-or-false-dawn/ [cit. 2022-05-06]

Figure 2.6: Image of the atmosphere with Sun low above the horizon by John Fowler[13] under CC2.0. High above the horizon, the sky appears blue, near the horizon, it goes through light blue to desaturated orange.

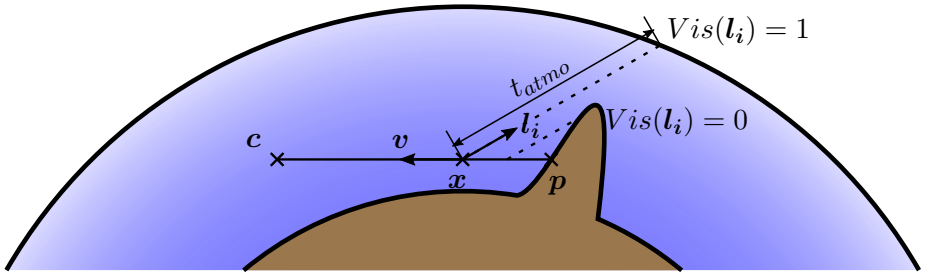


Figure 2.7: Diagram illustrating the scattering equations.

2.5 Atmospheric scattering

The Earth's atmosphere is a major factor in our perception of the sky. It only has minor effects at night, like the scintillations of the stars, as mentioned earlier. During the day, however, the incoming light is powerful enough that the scattering becomes significant enough to be seen by the naked eye. As the light travels through the atmosphere, some is absorbed, some scattered away, and some continues unchanged. This scattered light then travels in a different direction. The light gets scattered throughout the atmosphere like this into many scattering orders. Scattered and absorbed amounts depend on several factors, like wavelength, density of particles, and the type of particle encountered. The blue color we see is the result of these scattering events.

For simulations, three types of particles are considered. The first type is aerosols scattered in the atmosphere. Interaction with them is independent of the wavelength, and both scattering and absorption events happen. Scattered

[13]Sky panorama by John Fowler: https://pxhere.com/en/photo/473481 [cit. 2022-05-05]

directions are very forward-oriented, so the direction of the light ray deviates only slightly.

The second particle type is molecules of the air. The molecules do not absorb any light, only scatter it. The scattered amount of light depends on the wavelength. Mostly blue light is scattered, so the sky appears blue during the day. The direction distribution is close to uniform, with soft peaks in the forward and backward directions. The density also does not fall off as quickly with altitude as for aerosols.

The last considered contributor is ozone particles. Ozone only absorbs light and does not contribute to scattering. Similar to regular air particles, the wavelength is a factor for the amount of absorbed light. The peak wavelength seems to be in the middle of the visible spectrum and the amount absorbed falls off toward both sides of the spectrum. That means that mostly green light is absorbed. Ozone is not distributed in the same way as other particles of the atmosphere. It is mostly concentrated in what is called an ozone layer, and outside it, the density is nigh negligible.

The atmospheric scattering reaching the camera at point $\boldsymbol{c}$ from the direction $\boldsymbol{v}$ (direction of the light travel) can be described with the following set of equations[Hil20]:

$$L(\boldsymbol{c}, \boldsymbol{v}) = T(\boldsymbol{c}, \boldsymbol{p}) L_O(\boldsymbol{p}, \boldsymbol{v}) + \int_{t=0}^{\|\boldsymbol{p}-\boldsymbol{c}\|} L_{scat}(\boldsymbol{c}, \boldsymbol{c} - t\boldsymbol{v}, \boldsymbol{v}) \mathrm{d}t \tag{2.1}$$

$$T(\boldsymbol{x_a}, \boldsymbol{x_b}) = e^{-\int_{\boldsymbol{x}=\boldsymbol{x_a}}^{\boldsymbol{x_b}} \sigma_t(\boldsymbol{x}) \|\boldsymbol{dx}\|} \tag{2.2}$$

$$L_{scat}(\boldsymbol{c}, \boldsymbol{x}, \boldsymbol{v}) = \sigma_s(\boldsymbol{x}) \sum_{i=1}^{N_{light}} T(\boldsymbol{c}, \boldsymbol{x}) S(\boldsymbol{x}, \boldsymbol{l_i}) p(\boldsymbol{v}, \boldsymbol{l_i}) E_i \tag{2.3}$$

$$S(\boldsymbol{x}, \boldsymbol{l_i}) = Vis(\boldsymbol{l_i}) T(\boldsymbol{x}, \boldsymbol{x} + t_{atmo} \boldsymbol{l_i}). \tag{2.4}$$

$T(\boldsymbol{x}_a, \boldsymbol{x}_b)$ is the transmittance between two given points in the atmosphere. $T(\boldsymbol{x}_a, \boldsymbol{x}_b)$ in the interval $[0, 1]$ expresses the atmospheric extinction along the light path, the higher, the less light is scattered away or absorbed. $S(\boldsymbol{x}, \boldsymbol{l_i})$ is the shadow term. If the point $\boldsymbol{x}$ is not in the shadow of light i, it has the value of transmittance to the top of the atmosphere toward the light, otherwise 0. L_{scat} determines the amount of inscattered light along the path. It is the main source of the sky's color. Finally $L(\boldsymbol{c}, \boldsymbol{v})$ combines the inscattered light with the decayed original luminance to calculate the incoming luminance to the camera from the given direction. Also important to note is that the behavior of these equations changes with wavelength. Mostly the terms σ_t (extinction), σ_s (scattering), and p (scattering phase function) change drastically for different wavelengths of light. The equation variables are illustrated in Figure 2.7.

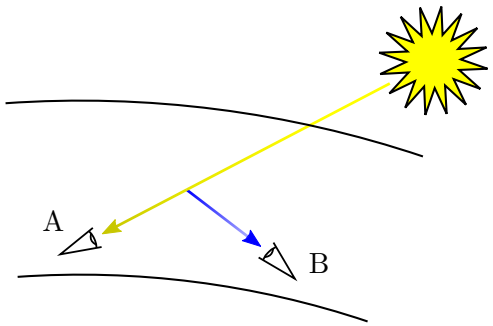


Figure 2.8: Visualization of light scattering. Observer A sees the light as coming straight from the Sun. Observer B see the light as the blue color of the sky. Note the loss of intensity along the way.

Figure 2.8 shows a simplified version of the scattering principle. The Earth shadowing is omitted and only the transmittance of the atmosphere that decreases the intensity, and blue light scattering are shown. The same ray sent from the Sun gives different perceptions to each observer.

These equations do not take multiple scattering into consideration, as L_{scat} only adds up the primary scattering. As there is no theoretical limit to the scattering orders, the equations would become recursive. These four equations will be revisited and explained in more detail in Chapter 5, when describing the implementation used for the app.

An example photo of the atmosphere can be seen in figure 2.6. As can be deduced from the shadows, the Sun is barely above the horizon, yet the sky retains a lot of its blue color.

The color of the sky is not the only effect the atmosphere has. As some of the incoming light gets scattered and absorbed, what remains is less than what arrived at the top levels of the atmosphere. Therefore the apparent brightness has decreased. This effect can be easily observed during the morning and evening when the Sun is just above the horizon. Taking a direct look at the Sun then is much easier than during the day, when even with squinted eyes, one can only bear the intensity for a while.

Another side effect of the scattering and absorption is the color shift toward red. As mentioned before, air molecules scatter primarily blue light, ozone absorbs mostly green light, and aerosols interact independently of the wavelength. Wavelengths least affected by this extinction are those at the red end of the spectrum. Therefore, light passing through the atmosphere appears to have a slight red tint. This effect is again exaggerated at low elevations when the color of the solar and lunar disks turns red. This shift is

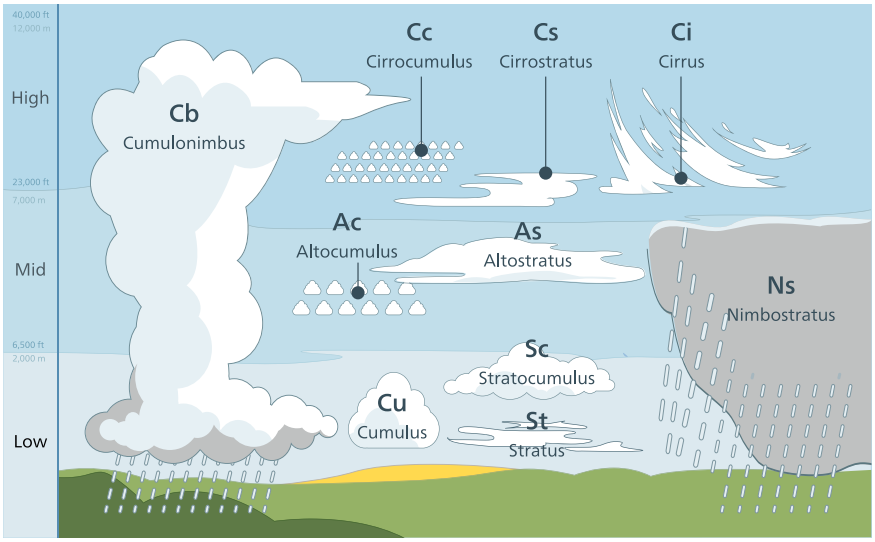


Figure 2.9: Illustration of various cloud types by Valentin de Bruyn[14]. The ten most common cloud types with their abbreviations and expected shapes.

also the reason why the sky starts turning red as well. Despite mostly blue light being scattered, there is such an imbalance of the wavelengths amounts that even then, the most scattered wavelengths are the long ones.

Lastly, clouds are an essential feature of the sky. Clouds are made of water drops or ice crystals suspended in the air. They are not transparent, only somewhat translucent, so their color appears to be white, sometimes gray for denser clouds. As such, they can occlude other objects in the sky, like the stars, Moon, and the Sun. The clouds scatter and absorb light similarly to the atmosphere but with a higher intensity, which gives the clouds their translucent look.

Furthermore, there are several types of clouds classified by their location in the atmosphere and shape. Cirrus clouds usually appear in the highest layers and look like feathers or light haze. Cumulus are the stereotypical cloud shape. They appear as white balls of cotton with relatively flat and darker bottoms. Stratus are clouds in the lowest layer above the ground. They tend to spread wide and cover the bottom of the layer like bedsheets. Stratus can fall all the way to the surface, at which point we call it fog. These types are not exclusive but can move to other layers and combine, which is then reflected in their name. E.g., stratocumulus are flatter cumulus clouds in the lower cloud layer, almost covering the sky with visible gaps in between them. Stratus and cumulus can also appear in their rain variants, nimbostratus and

[14]Illustration for the app Coton by Valentin de Bruyn: https://pxhere.com/en/photo/473481 [cit. 2022-05-07]

Figure 2.10: Artistic illustration of the night by Deborah Ann Good[16]. Notice the liberal use of shades of blue, even for the terrain.

cumulonimbus resp. Nimbostratus has a very characteristic look of uniform dark gray. All three main types of clouds can be seen in Figure 2.6. Cirrus clouds appear at the top of the image as a stringy white substance. Cumulus clouds can be see at the bottom right as huge white masses. Stratus clouds are the thin, wide, dark clouds at the bottom right and far bottom right.

For a better explanation of individual types of clouds see an excellent article by Thought Co.[15]

2.6 Tonemapping

When lit by moonlight, our environment seems to have a blue tint, as if the Moon's light was blue. This effect is often called blue shift because it is not the light that is blue (in fact, it is reddish in hue) but our perception of it. Blue shift is often used in visual art, e.g., palettes of cooler colors are used for paintings of night scenes[JPS+00].

[15]The 10 Basic Types of Clouds by Thought Co.: https://www.thoughtco.com/types-of-clouds-recognize-in-the-sky-4025569 [cit. 2022-05-07]

[16]Moon, Mountain, Starry Night by Deborah Ann Good: https://pixels.com/featured/moon-mountain-starry-night-deborah-ann-good.html [cit. 2022-05-06]

Human vision is realized with two types of light-sensitive cells: rods and cones. Rods are best suited for low-light conditions, less than 10^{-1}cd/m^2. They provide no color information and are the sole source of dark vision. At around 10cd/m^2 and beyond, the rod vision is completely saturated from the amount of light, and the information becomes useless. Unlike rods, which have only one type, cones are cells of three types, often called L, M, and S. These labels are derived from the long (red), medium (green), and short (blue) wavelengths the cells are most sensitive to. Combining the information from these cells gives us the perception of color. They react to light in the range 10^{-1}cd/m^2 to 10^6cd/m^2[KP04].

At 10^{-1}cd/m^2 and lower, only the rods react to light, and we see in night vision, also referred to as scotopic vision. In the range 10^{-1}cd/m^2 to 10cd/m^2 both types of cells are active and make a significant contribution to our vision. This is also known as mesopic vision. For completeness' sake, beyond 10cd/m^2, the rods are blinded and make no further contribution to visual stimulation. Only the cones provide relevant information. This is also called photopic vision.

The rod and cone information is processed by a series of neural circuits, the input layer of which is formed by the bipolar cells. The rod bipolar massively magnifies the incoming signal, which is excellent under conditions of barely any light but can quickly become overwhelmed. Therefore it is only valid at the lower end of scotopic vision. The cones are connected to two bipolar cells: on and off cone bipolars. Rods are, however, connected not only to their corresponding bipolar, but connections have been found to some of the cone cells themselves and to their cone bipolars. This is called rod intrusion.

The principle behind blue shift is not entirely understood, but the aforementioned rod intrusion is a good candidate as it directly sends signals from the more sensitive achromatic rods to the cones responsible for color vision. In Figure 2.10, you can see that the chosen palette for the night scene is exclusively blue. This is a very common pattern in art, where cold colors like blue and purple are used to give the scenery a night feeling.

Hue shift is not the only effect we see during twilight and moonlit nights. Another noticeable change in our vision is the loss of contrast perception, thus loss of ability to see finer details. That is the reason why we can still make out the shapes of objects, but not their texture or detail, since our ability to recognize silhouettes is hindered less. A good demonstration of how we use local and global detail to distinguish objects is the hybrid illusion, which combines both types of detail from different objects.

Chapter 3

Related work

Several papers have attempted to create a physically-based simulation of several sky phenomena. This thesis is not the first publication to compose them into a single application. However, it uses some new research and uses the volume a thesis permits for a more comprehensive description. For various reasons, the explanations will not be too detailed. Mostly to prevent duplicities with Chapter 5 and avoid complex descriptions and equations. Some approaches are notable but not explained well in their original papers, often due to the limited size of the publication.

3.1 Sun

Rendering of the Sun disk seems to be omitted in much related research. Some papers[JPS+00, JDD+01, MED12] focus only on the night sky, so the Sun is only considered as an illuminator, not as a rendered object. Haber et al.[HMS05] focus specifically on twilight, but the Sun is missing from the sky renders. Some research[BN08, Hil20] focuses on the day sky, but only on the light scattering in the atmosphere and color of the atmosphere itself. The Sun, again, is considered only as an illuminator.

As described in Section 2.1, even at this distance, mostly due to the effects of the atmosphere, some visual features can be observed on the solar disk. The Sun is therefore still a viable target of study. Haber et al.[HM05] focus specifically on that. They assume the atmosphere to consist of hundreds to

thousands of geocentric layers, each containing the same amount of particles, but having different heights, temperatures, humidities, and other attributes. The upper boundary of the atmosphere is set to 35km, since the upper layers do not have the density to affect the light much. Each ray is traced through the layers and follows a parabolic path within each. Under correct circumstances, it is possible for the ray to return back down to Earth; this effect is usually called a mirage.

The escaped ray is then checked for an intersection with the Sun and colored accordingly. The Sun is assumed to behave like a flat-colored disk with a darkened limb. This gives the disk more of a spacious feel, but also follows real images. The incoming irradiance is multiplied by the extinction calculated during the ray's path through the atmosphere. This shifts its color closer to red and decreases its brightness when close above the horizon. Tweaking the attributes of the atmosphere produces various visual effects, like split Sun, omega Sun, double Sun, and many more.

Another approach is curved photon mapping[GSAM04]. The path the light takes can be traced in the other direction, from the light to the surface. This approach is called photon mapping and helps calculate paths through non-homogeneous media, a good example being caustics. As the atmosphere is an inhomogeneous medium and the light takes a curved path, it is possible to precompute the incoming photons following these curves into a photon map. Sampling of the map can then also be used for global illumination.

Since it is in shape similar to the Moon, almost a sphere that appears as a disk, even having a very similar apparent angular size, similar rendering techniques can be used for their shapes. The Sun can be modeled as a sphere or as a disk, but for today's real-time rendering purposes, it is often simplified to a simple flare due to its overwhelming brightness.

3.2 Moon

Rendering the Moon is a much better explored topic, compared to the Sun, as it is a visually more interesting object in the sky. As was described in Section 2.2, the Moon has many different features to be replicated: phase, varying albedo among others. Phase is simple to calculate if we know the directions to the Sun and to the Moon if we assume the direction to the Sun is the same on Earth as on the Moon. In other words, the Earth-Moon distance is negligible in comparison to that to the Sun. Without this simplification,

the process becomes more complicated. To get the directions themselves, astronomical algorithms or some approximations can be utilized. Jean Meeus seems to be the go-to author for these sorts of calculations[Mee91].

The Moon itself can be drawn in several ways. The obvious way is spherical geometry with textures[JPS+00, JDD+01]. This approach has several benefits, like having simpler visualization of lunar topology, as it can be directly modeled, or the simplicity of a drawn object without the need of manually considering parallax, or the distance influencing the viewed percentage of the surface. The roughness is the easiest to notice in shadows in and around craters on the lunar surface. However, it can also break the spherical shape at extreme zoom levels, as the silhouette will include this unevenness. This approach can be further optimized with tesselation. The detailed geometry does not need to be stored directly, but only in height textures that are then used when adding more detail to the simplified Moon mesh. This approach works really well with ray tracing, as shadows are solved naturally with the usual process of shadow rays.

Another method is shading a quad to look like the Moon[MED12]. A simple square quad can, by discarding some of its fragments, be made to resemble a disk. This disk is then drawn onto as an orthographic projection of a virtual sphere. Each fragment then calculates its position on the sphere, which is conveniently also its normal. This position can be rotated as if rotating the whole Moon in the opposite direction and, along with the normal vector, can be used for texturing and calculations of illumination. This method achieves a good-looking illusion of a sphere. However, shadows become hard to compute and can be approximated by altering the surface normals with a normal map. This looks good at first glance but it can introduce artifacts like illuminated areas with extreme normal changes. On the other hand, breaking the shape of a perfect sphere is nigh impossible, as to get height information from a texture for the purpose of silhouette alteration, texturing coordinates have to be found, which requires the assumption of a perfect sphere.

The reflective properties of the lunar surface have been an object of study for a long time. The best fitting photometric function has been theorized[Hap63] and later improved[Hap66] by Hapke. He assumed the surface has a profile with a strong self-shadowing tendency. The function consists of three terms in multiplication: the average angular scattering function of a single particle, the retrodirective function describing the backscattering, and a function describing reflection of a porous surface at the microscopic level, but a flat plane at the macro scope. The third term is the primary target of improvement in the latter paper. As the model of the surface changes, the parameter space is separated into several regions. As this was the chosen approach, it will be described in more detail in Chapter 5.

Earthshine has a very soft effect and as such small changes are not visible by the human eye. It is possible to omit it completely, as it only is noticeable during a new moon[JPS+00]. However, at these large phase angles, when only a thin sliver is lit by the Sun, the Moon may feel too dark. A simple way is to estimate the full strength of earthshine and multiply it by the percentage of lit portion of the Earth's disk[JDD+01]. As the Earth's and Moon's phase angles are exactly opposite, the lit surface percentages add up to one. Jean Meeus has shown the calculation of illuminated fraction of the disk to be simple if the phase angle is known[Mee91].

The Earth's albedo is not uniform, however, and so the strength of earthshine changes slightly during the day. The albedo change is mostly due to cloud and ice coverage, but also the difference between ocean and ground albedo or the seasons. It has been shown it is possible to estimate an average albedo to different orientation of the Earth with a function fitted to measured data[GQY+01]. This approach, while more accurate, seems to be overkill for realtime rendering applications, like games, but also for most offline renderings, like animations.

Librations are another easy-to-miss effect. It is therefore possible to ignore them, as they do not change our perception of the Moon. Only under special conditions, like a timelapse, can we notice the Moon being "too still" and missing the wobble we might have seen visualized or explained somewhere else.

A straightforward solution is to represent the Moon as geometry and use astronomical algorithms to calculate its rotation in space[JPS+00, JDD+01, WTB+10]. This will naturally simulate optical librations and allows the simple incorporation of parallax and thus diurnal librations as well. If the Moon's position is shifted by one Earth's radius (observer is moved from the center of the Earth to its surface), diurnal librations will be a natural consequence. Physical librations are nothing more than small changes to the rotation, so this approach allows their inclusion as well. As the shaded quad method simulates the Moon sphere, only with less triangles drawn, physical orientation in space can also be used to determine optical rotation here, though its use is more complicated[MED12].

Another theoretical way to calculate optical librations is to find the path the selenographic center takes across the Moon disk. As this would leave one freedom of rotation, another piece of information would be needed, such as the direction to the north pole. This method is potentially more computationally expensive and I am aware of no research done on the topic.

3.3 Stars

The stars appear as point lights, which is not suitable for raytracing. We could even assume them to be very distant spheres, but the probability of an escaped ray hitting one is practically zero. One possible way of modeling the stars for raytracing is using a background image with the stars and then drawing it behind the raytraced image[JPS+00, JDD+01]. Each ray escaping the atmosphere then also records its optical depth and, transitively, transmittance, which is then stored in an alpha image, used when blending the background with the stars into the raytraced foreground. This simulates the partial decrease in brightness due to some of the incoming light being absorbed or scattered away. However, Jensen et al. do not explain this approach in their paper in too much detail.

The rasterization approach can be realized with drawing a simple quad for each star, on which a small circle is drawn[MED12]. This circle has a minimum pixel size of $2\sqrt{2}$ to prevent flickering and depends only on the fov and resolution, not on the star's magnitude. The brightnesses are then distinguished by the intensity of the star's color. This intensity is not constant across the entire circle, but follows a point spread function (PSF), scaled using the apparent magnitude of the star. This gives the stars soft edges and thus less alias as they move across the sky. The star intensity is then decreased by extinction along the light's path. For this, the optical path length Φ is calculated for altitude h, angle to zenith Θ, and length in zenith direction t_e using the law of sines as:

$$\Phi(\Theta) = -\sin(\arcsin(\frac{r_e + h}{r_e + t_e}\sin(\Theta)) - \Theta)\frac{r_e + t_e}{\sin(\Theta)} \tag{3.1}$$

$$O(\Theta) = c_r(1 + \beta_r)(t_e - h)^{-1}\Phi(\Theta). \tag{3.2}$$

The color and intensity are attenuated by O. $c_r \approx 6$ is a scaling constant and β_r is scattering coefficient for Rayleigh scattering, set to $(0.16, 0.37, 0.91)$ at sea level. For Rayleigh scattering, $t_e = 8\text{km}$ is common and the mean Earth radius is set to $r_e = 6371\text{km}$.

It is further possible to highlight the stars with a glare. For the raytracing method mentioned above, a more physically based approach has been chosen[JPS+00, JDD+01]. During tonemapping, the final pass of the image, bright spots are brightened with a physically based flare that attempts to model the scattering in a human eye[SIS+95]. The flare simulates several effects. Scattering of light within the eye in scotopic vision causes bloom and flare lines. Here, the bloom is light bleeding from bright sources to dark areas. As a result, some contrast information around bright light sources is lost. Flare lines, also called starburst, are those distinct lines coming out of

sources of light we see at night. Diffraction within the eye causes lenticular halo, concentric colored rings of decomposed light. This flare is applied to all bright points in the image, most notably stars.

Müller et al.[MED12] have chosen a much simpler star glare for their rasterization approach. They use a similar principle as for the star itself, a quad with a circle colored by a PSF. However, where the star's size depends on magnitude, and the intensity is dependent on the magnitude and the camera's fov, both the glare's intensity and size depend solely on the magnitude. Only some stars are given this glare, which, along with the glares' sizes, is controlled with a control magnitude. This glare method has visible borders between concentric rings of brightness, as the quantization snaps the calculated intensities to the nearest representable value. However, when the glares are not too large for the quantization to be visible, they are a good approximation of the aforementioned physical approach.

Müller et al.[MED12] have further added scintillations to the stars. For this, the optical depth Φ is utilized again. Each frame a random number n is generated in the range $[0, 1]$ for every star. This number is then transformed to $[0, 1]$ by $N = \frac{0.02}{n}$. N is used to attenuate the star's brightness as:

$$S = c_s \beta_r \Phi N, \tag{3.3}$$

where $c_s \approx 20$ is a scaling coefficient. This ensures that the flickering is not too chaotic but still occasionally oscillates severely enough to be noticed more easily.

Star data is universally taken from the Yale Bright Star Catalogue[HW91], since it contains only those stars we can see with the naked eye. It contains over 9000 stars in the J2000.0 equinox, with position and apparent motion in equatorial coordinates. The catalogue includes information about color either with star classification or with the $B - V$ value, so these have to be converted to a format more useful for the purposes of rendering. Furthermore, some entries seem to be missing some information, like apparent magnitude or color information, despite it being available in other, more comprehensive catalogues.

To get the color of the star in rgb space, the $B - V$ value is first used to estimate the star's temperature. The star is assumed to be an ideal black body, so the color of emitted radiation is determined from its temperature. This is realized with the Planckian locus, a piece-wise polynomial curve in the xyY space[YsBhBsDi06]. This color can then be converted to the rgb values for realtime rendering.

There are many stars we cannot see individually. We could therefore assume we can simply omit them[JPS+00]. This, however, results in an unnatural-looking sky full of bright points and eerie emptiness among them. Alternatively, we can draw the background starlight as an environment map around the planet[JDD+01, MED12]. For both papers, the environment map has had the bright stars removed beforehand, as drawing them on top of the background would duplicate them, and if we assume the stars to be moving in relation to each other, one copy would stay in place, while the other would slowly drift away with time. Müller et al. have further made the map's brightness depend on the fov, resolution, and control magnitude.

3.4 Other sources of light

Rendering the aurora is a rewarding task due to the inherent beauty of the phenomenon. Lawlor et al. devised a multi-step approach for their raytracing application[LG11]. The vertical space above the planet is divided into the atmosphere at and below 80km and the aurora layer above that but below 200km. First, the footprint is calculated. This cannot be a simple curve, as auroras tend to have noticeable width to them. Fluid dynamics is thus utilized to distort a long rectangle and give it a more chaotic look. A distance field is calculated for the footprint texture to speed up sampling during rendering. The curtain is then stretched into 3D using an atmospheric electron deposition function. These textures are then sampled during raytracing; the distance field is used to determine the length of the next step, and the auroral emission is the product of curtain footprint and vertical deposition function.

Tao and Wenlong[TW14] have changed the footprint generation. They initialize the fluid generation with a sine wave, instead of a straight line, before simulating a few frames. After the simulation, some low-frequency noise is added to remap the density field and create a more complex profile shape. The rendering itself is done by photon mapping with a volumetric photon map, the generation and sampling of which is altered with multiple perlin noises to add more interesting features.

Zodiacal light is a much less visually intense feature, and the principle behind it is still not entirely understood. However, since its intensity changes very little with direction, measured values in a texture can be used as an environment map[JPS+00, JDD+01]. Each escaping ray then samples this texture with a bilinear lookup.

Finally, due to the uniform nature of airglow, it can be simply simulated as an actively emissive layer in the upper atmosphere[JPS+00, JDD+01].

3.5 Atmospheric scattering

The scattering of light in the atmosphere, resulting in the blue color during the day and orange color during twilight, is a thoroughly researched topic. Since a physical simulation would take too long to compute, look-up tables for faster color acquisition are heavily utilized for the purposes of real time rendering.

As can be seen from Equations 2.1-2.4, the single scattering of light in the atmosphere is fairly simple. Adding multiscattering, however, brings high complexity since it is recursive, though its impact is fairly small and ignoring it still gives good results. The straightforward approach is iteratively calculating the scatters up to a certain depth[BN08]. Several calculations are precomputed and stored in look-up tables (LUTs).

The simplest is transmittance $\mathbb{T}$, which, due to spherical symmetry, depends only on altitude and angle to the zenith so that it can be stored in a 2D LUT. It is computed simply by raymarching the atmosphere to the top boundary and calculating the optical depth of the ray. Further parametrization is utilized to not waste space with rays that would intersect the ground. Three-colored texels are stored, representing the transmittance of three selected wavelengths: red, green, and blue (more specifically $(680, 550, 440)$nm).

The second LUT is the total illuminance $\mathbb{E}$ by scattered light at a given point for a given Sun direction. This value is not as straightforward to compute as it requires multiscattering to be evaluated. The table is therefore filled iteratively over individual scattering orders. Each order adds its contribution to the table. As the intensity of light scattered into additional orders quickly decreases, no more than 10 iterations are needed, and 5 were chosen in the paper. Similarly to transmittance, the data is spherically symmetrical, so with smart parametrization, 2 dimensions are again enough for storing the table.

The third and last LUT is the amount of inscattered light $\mathbb{S}$ at a point from a particular direction with a given Sun vector. This, again, has to consider multiple scattering orders, so it is computed iteratively in the same loop as the second LUT. This yields the benefit of allowing data to be interchanged each

step since the final LUTs only store the sum over all steps. As mentioned, this LUT is indexed with position, view direction, and Sun direction. Symmetry can help reduce the dimensionality again, though the Sun direction makes it more complicated. The result is a four dimensional table, which is realized with a 3D texture and the data about the fourth dimension is stacked in the third. This also means manual interpolation is needed for the fourth dimension.

Three auxiliary tables are used during precomputation: change in light scattered from a given point in a given direction $\Delta\mathbb{J}$, change in the second LUT $\Delta\mathbb{E}$, and change in the third LUT $\Delta\mathbb{S}$. A rough outline of the precomputation algorithm is then as such:

```
precompute 𝕋
initialize Δ𝔼 with light from the Sun
initialize Δ𝕊 with light from the Sun
initialize 𝔼 with 0
initialize 𝕊 with Δ𝕊
for (i in range(N_Orders)):
        compute Δ𝕁 from Δ𝔼 and Δ𝕊
        compute Δ𝔼 from Δ𝕊
        compute Δ𝕊 from Δ𝕁
        add Δ𝔼 to 𝔼
        add Δ𝕊 to 𝕊
```

This approach gives good results for atmospheres of densities similar to that of the Earth. However, it has several major drawbacks. For one, it is fairly slow. Five scattering orders took five seconds on the tested hardware, which was an NVidia 8800 GTS. Even though the LUTs do not have to be recomputed every frame, it still is a lot of time added to the application's loading time. Secondly, denser atmospheres require more scattering orders, which takes even longer, but the computation starts to diverge. As was shown by Hillaire[Hil20], at 40 iterations, the colors and intensities are way off the reference, giving strange results. Thirdly, the four-dimensionality of the third LUT is not very practical and makes working with it somewhat clumsy.

Hillaire has shown that these drawbacks can be mitigated with more assumptions: the second-and-higher-order scattering happen according to a uniform phase function, so no direction is preferred, and the illuminance reaching a point in space is the same for all points within a large area around it. This will allow us to skip iterating the scattering orders and calculate multiscattering as a function of the second order with the infinite sum of a geometric series.

This simplifies the multiscattering LUTs to just one two-dimensional table, parametrized by altitude and angle between the direction to the Sun and zenith. Furthermore, high resolutions can dramatically impact performace if every pixel raymarches the atmosphere to get the inscattered light. Some measures have to be taken to make the rendering independent of the resolution. To this end, two more LUTs are introduced for a total of four.

The first is the sky-view LUT. A pre-rendered view of the sky without obstacles and terrain, assuming the surface is a smooth sphere. This ensures that the sky is rendered with a set resolution and sampled as a texture during the rendering of the scene. Another benefit of this decision is that it can be used as an environment map. Since the frequency of change in the sky is not great, the resolution of the sky-view LUT can also be relatively modest.

One exception to this assumption is the horizon, especially during twilight, as the Earth's own shadow can be observed. Colors just above the horizon can change quickly, so higher resolution is needed to prevent artifacts. Therefore, instead of increasing the resolution of the texture, a better parametrization is chosen to cluster texels around the horizon. The solar disk is not added to this texture, as it is too small for the texels, which would introduce an alias and a blocky-looking Sun with changing size.

The other added table is the aerial perspective LUT. The color effect of the atmosphere can be observed even between the camera and the drawn geometry, not necessarily only when looking out to space. To prevent per-pixel integration of the atmosphere, this effect is precomputed into a 3D table. The texture space is treated as the camera's frustum, and each texel has the color of scattered light and transmittance along the path stored. The third dimension serves as slices of the space in front, and as such, the transmittance should decrease with distance. Finally, to get the color and transmittance to the geometry, trilinear interpolation is used.

As this was the chosen method, it will be described further in Chapter 5.

3.6 Tonemapping

Tone mapping is more complicated since the principle behind the blue shift (often called "rod intrusion") is not entirely understood. Attempts have been made to model the shift empirically by shifting the hue toward a blue point estimated from several paintings[JPS+00]. The point chosen is

$(x_b, y_b) = (0.25, 0.25)$ in the xyY color space. A scalar s in the range $[0, 1]$ is used to interpolate between fully scotopic and fully photopic visions, so for values inside that range, the viewer is in their mesopic state, about 0.1cdm^{-2} to 4cdm^{-2}. The colors are then shifted toward the blue point according to the parameter s. The overall computation is:

$$
\begin{aligned}
s &= \begin{cases} 0 & -2 \le \log_{10} Y \\ 3\left(\frac{\log_{10} Y + 2}{2.6}\right)^2 - 2\left(\frac{\log_{10} Y + 2}{2.6}\right)^3 & -2 < \log_{10} Y \le 0.6 \\ 1 & \log_{10} Y < 0.6 \end{cases} \\
W &= X + Y + Z \\
x &= X/W \\
y &= Y/W \\
x &= (1 - s)x_b + sx \\
y &= (1 - s)y_b + sy \\
Y &= 0.4468(1 - s)V + sY \\
X &= \frac{xY}{y} \\
Z &= \frac{X}{x} - X - Y
\end{aligned}
$$

While this achieves good-looking results, it does not accurately represent our perception and is only approximated from paintings, which is not a reliable way to model human vision. A different approach to blue shift is attempting to model the rod-cone interaction directly. Khan and Pattanaik have devised a procedure that stays more true to the physiological findings[KP04]. The intensity of the light has to first be calculated. From this intensity, the response of the rods is then derived. For this, the fast neural adaptation and the much slower reaction of photopigment, like saturation and regeneration, are simulated.

With the rod response known comes the second step of the procedure, the rod-cone interaction. Fully scotopic vision is assumed, so the cone responses are inherently zero. However, as about 20% of rods have been found to have a connection to a neighboring cone, that amount of rod response is transferred to the cones. Since the tone shift is toward blue, this connection is assumed to be purely to the S-cone cells (S for short wavelengths).

Now the cone responses (two of which are zero) have to be converted back to a color to show on the screen. This is done with the inverse of the function from the first step. The result of this transformation is intensities of individual wavelengths, which are further transformed with a matrix introduced by Hunt[Hun05]. Hunt's matrix converts the radiation intensities to the

XYZ tristimulus values, which are then transformed to the xy chromaticity coordinates. That is because this calculation only serves to determine the hue of the final pixel color. The intensity for the final color is the initial intensity of the color pre-mapping.

Despite being closer to the physiological principle of the blue shift of our scotopic and photopic vision, the provided images demonstrating the results of Khan et al.[KP04] are almost unsettlingly blue. One potential reason for that could be the incorrect assumption that rods only form connections to the S-cones, and the response might be transferred to other types of cone cells as well. The is no direct physiological evidence that would either support or refute this assumption.

Hue shift is not the only effect we see during twilight and moonlit nights. Another noticeable change in our vision is the loss of contrast perception, thus loss of ability to see finer details. That is the reason why we can still make out the shapes of objects, but not their texture or detail, since our ability to recognize silhouettes is hindered less.

The loss of detail has been traditionally done with a low-pass filter. This, however, does not represent the natural effect very well since blurring disrupts the edges, which we can still see sharply, as is often visualized in art, e.g., in Figure 2.10. Artists often keep the silhouettes sharp but omit much of the finer detail within the objects themselves.

Jensen et al. use an informed low-pass filter. Firstly, the gradient magnitude $grad_{mag}$ is computed for each pixel. It seems they may have used a smoothening filter for the purposes of gradient detection first. The image is then blurred with a Gaussian filter with standard deviation σ specified by

$$\sigma = \sigma_0(grad_{strong} - grad_{mag}/grad_{strong}), \tag{3.4}$$

where σ_0 is the standard deviation of the maximum blurring that should be applied and $grad_{strong}$ is the smallest gradient for which no blurring should be done at all. This will completely avoid filtering areas with strong enough edges, just like the silhouette preservation in scotopic vision.

Chapter 4

Proposed solution

This chapter will list all features to be implemented by the app and briefly describe the chosen approach for each. Rasterization is the method selected for the app itself for its simplicity and better real-time capabilities. Furthermore, rasterization has much better support from common graphical libraries, like OpenGL. The major features implemented are the Sun, the Moon, the stars with their background, atmospheric scattering, and tonemapping. Lights of the night sky like the zodiacal light and aurora lights were omitted because they are only visible on specific occasions and would bring additional complexity without bringing much of a benefit. Clouds were also not implemented because they are complex and could further hinder the app's performance. Most of the chosen methods have been adopted from certain previous research, but some have been changed, and the occasional mistake has been corrected. A more in-depth description will follow in the next chapter.

The shaded quad from Müller et al.[MED12] has been chosen to render the Moon disk purely for the simplicity of its geometry. A very simple high dynamic range transformation has been added to make the Moon stand out more in larger phase angles. The Sun is rendered in the same way – as a single quad – but a flat color is used, and a darkened limb is added as in Haber et al.[HM05]. The Sun's shape is kept unchanged and circular since raytracing many layers of the atmosphere is required to implement the visual distortion.

The stars will be rendered as simple quads generated on the geometry processor in a very similar manner to Müller et al.[MED12]. One difference is their size, which I have decided to keep constant in pixel size. This will

give them even more of a point-like feeling regardless of the zoom level. The glare will be implemented the same way, except for the scintillations. The nonlinear randomness approach does not feel like what the stars' flickering looks like, so I went with a more uniform-looking Perlin noise.

The atmosphere precomputations are adopted from Hillaire's work[Hil20], though several mistakes have been corrected. The biggest ones were probably the height for Rayleigh and Mie particles and the Mie extinction coefficient. Another minor change was the sky-view texture, which Hillaire parametrized to always have the Sun at the very left. Instead, it will be oriented relative to the north direction.

Tonemapping theorized by Khan et al.[KP04] will be implemented since it tries to stay closer to the physiological principle. However, since their results look too blue, some weighing will be used. Furthermore, the stars would lose too much of their color and would look washed out, so their blue shift will be weighted much less.

Some rendering attributes will be changeable both on initialization and during runtime. Especially the time values, like date and instant of the day, but also the speed at which time flows. Furthermore, geographical location, altitude above sea level, and timezone can also be changed. These attributes can be set during the app's startup as parameters in the command line. A separate console window will open during runtime, where these attributes can be retrieved or set in a text interface. Additionally, help messages can be printed either for the program parameters or for control of the app on runtime.

The app will contain a scene with some simple terrain and a movable camera, the fov of which can be increased and decreased. The camera's orientation can be changed with the mouse, and the position moved at varying speeds with the keyboard. The general overview of the rendering steps can be seen in Figure 4.1 and two specific examples are shown in Figure 4.2. Keep in mind that the night renders are heavily lightened to make the changes better visible (especially the stars).

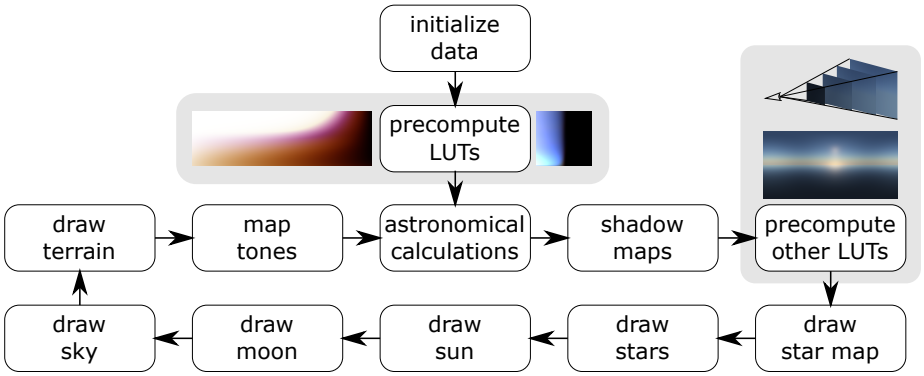


Figure 4.1: Order of operations as a flow chart.

(a) : Some features cannot be seen at all during the day.

(b) : The images have been severely lightened for the features to stand out more.

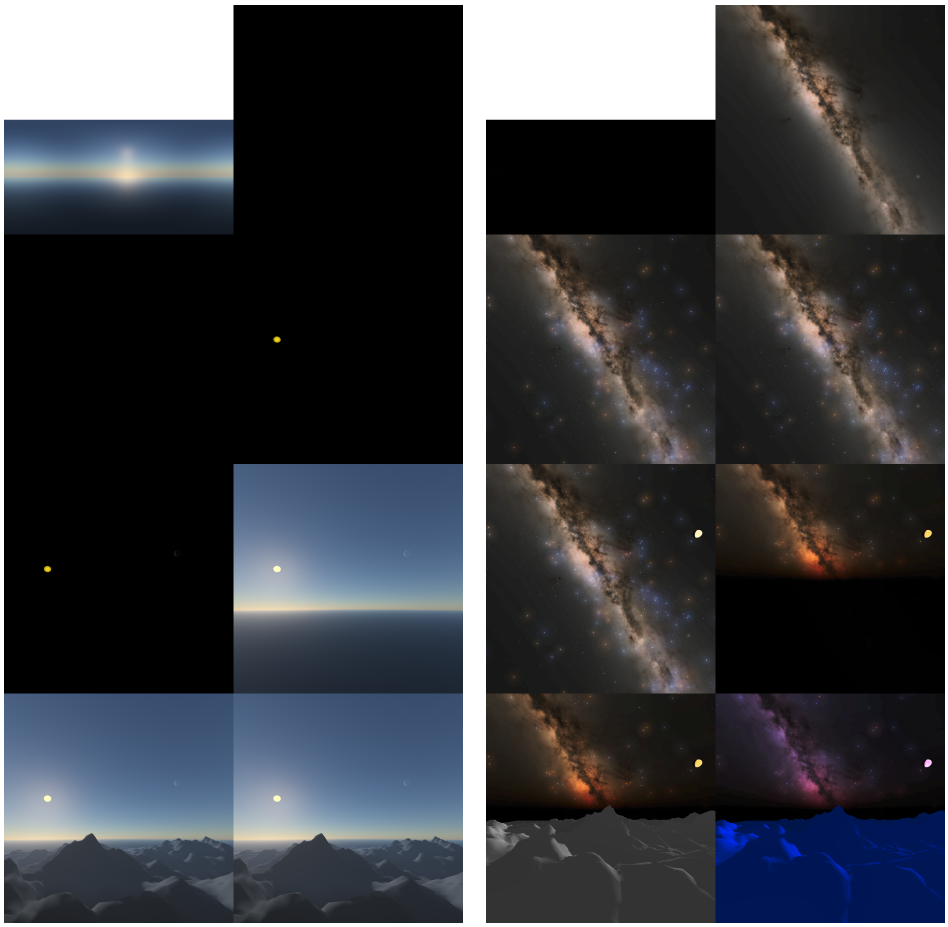


Figure 4.2: Notable steps of the rendering process for day (left) and night (right) in order: sky LUT, star map, stars, Sun, Moon, sky, terrain, tone mapping.

Chapter 5

Implementation

This chapter will describe the implemented method in detail in sections on the specific parts of the rendering method shown briefly in Figure 4.1. The intermediate steps of the process can be found and referenced in Figure 4.2. The bulk of the numerical algorithms has, for the sheer size, been moved to Appendix B. These computations are only limited to the more standard formulas, like astronomy and coordinate conversions, whereas the more specific calculations are kept in their individual sections in this chapter.

5.1 Sun disk

The Sun is rendered as a flat-colored disk with a darkened limb. The disk is on a view-oriented billboard represented by a quad. The geometry is calculated on the GPU from the Sun direction $\boldsymbol{d_S}$, which can be calculated with the result of Equation B.10 transformed to local horizon coordinates. The billboard is oriented so that its normal aims toward the camera and rotated with regards to the world's up vector so that its horizontal sides are horizontal in world space. It is in a tangential plane to a unit sphere centered at the camera, so the mid-point's distance to the camera is one. A drawing of this approach is shown in Figure 5.1.

The quad is scaled according to the average Sun-Earth distance to about $9.355 \cdot 10^{-3}$, corresponding to about 0.536° angular diameter. Perspective projection is not considered, but at these distances, it can be safely omitted

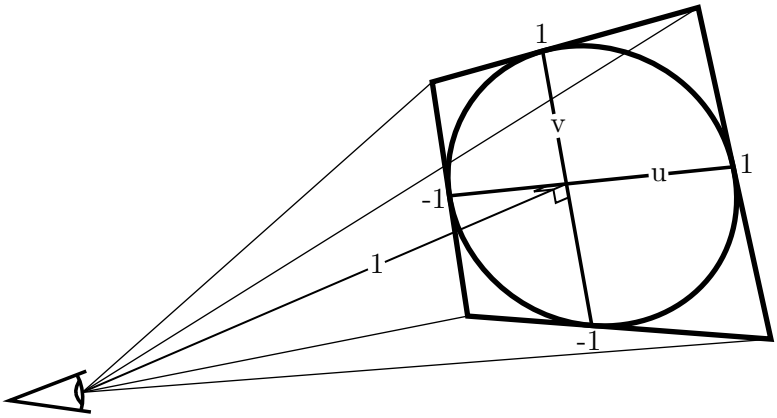


Figure 5.1: Sun disk realized with a single quad at a unit distance from the observer.

since it only amounts to about three millionths of a degree. Both the Sun and the Moon are much smaller than what we perceive them as. Therefore, to make them stand out more, they are scaled up to prevent the perception of being "too small" by 5.

On this quad, the Sun disk is drawn. Each fragment has uv coordinates in the range $[-1, 1]^2$. On the fragment processor, the distance d to the center is calculated first as $d = \sqrt{u^2 + v^2}$. Those fragments that lie outside the disk, so if $d > 1$, are discarded. This leaves us with as perfect of a circle as the rasterizer permits. Note that the coordinates are not in their usual $[0, 1]$ range. This is to help simplify the calculations.

The color of the Sun has been chosen as $(0.98, 0.86, 0.12)$[CH]. The Sun actually emits light colored much closer to white than yellow, and we perceive it as yellow due to the scattering and absorption in the atmosphere. Even though the app attempts to recreate this effect of loss of blue color, manual tweaking is necessary to achieve a look closer to our perception of the Sun's color. The color is multiplied by the limb darkening multiplier $1 - c(1 - \sqrt{1 - d^2})$, where c is the limb darkening coefficient for the Sun, which is set to 0.6[HM05]. Limb darkening gives the disk more depth and makes it look more like a spherical object. The resulting render can be seen in Figure 5.5a.

5.2 Moon disk

The Moon is rendered in a similar manner as the Sun, using the principle in Figure 5.1. In fact, the same shaders are used but initialized in a specific way

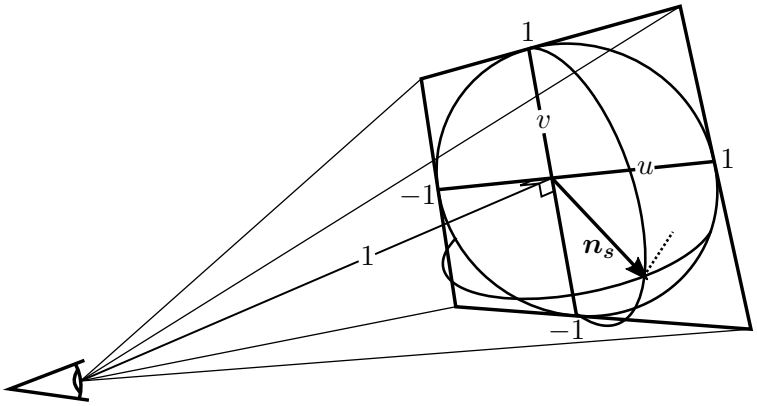


Figure 5.2: Moon sphere realized similarly to the Sun disk.

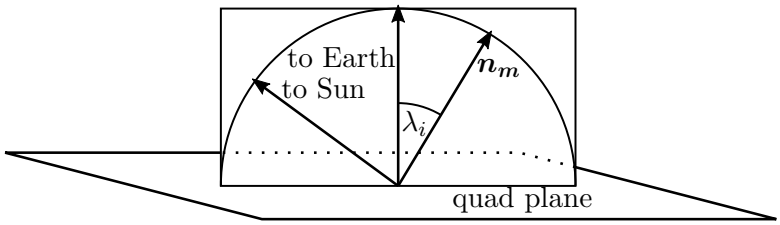


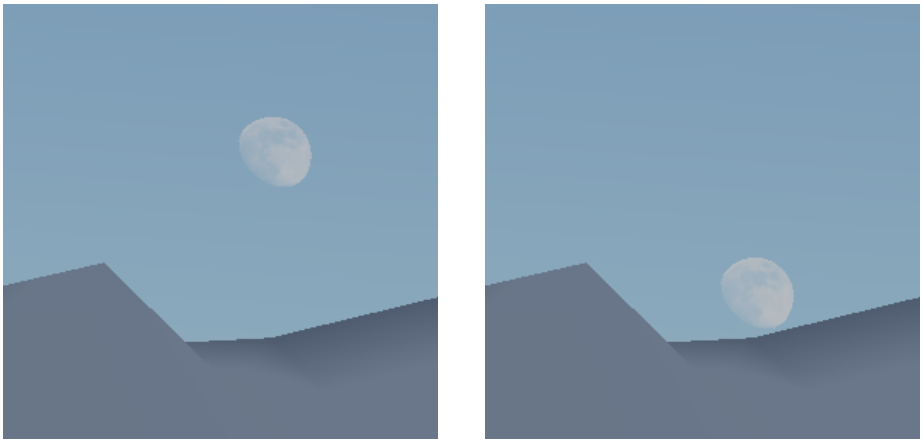
Figure 5.3: The illumination longitude is calculated in local space of the quad. In this case, the value would be negative.

to render the Sun. The zero vector is sent to the GPU as the Sun direction, and the Sun direction $\boldsymbol{d_S}$ is sent to the location of the Moon direction. This is then recognized on the fragment processor, and the Sun-drawing branch is taken.

The size of the Moon quad is determined from the calculated distance Δ in Equation B.10. When the Moon position is transformed to rectangular coordinates, it is multiplied with Δ, the vector $(0, 1, 0)$ is subtracted to correct for the radius of the Earth, and so the real distance l to the Moon can be computed. The angular size of the Moon s_M then follows from $s_M = 2 \arcsin(\frac{1737.5}{6378.14l})$, where the constant 6378.14 is the average radius of the Earth in kilometers, and 1737.5 is that of the Moon.

The offset done to move the observer to the surface of the Earth has a significant parallax effect (and is also responsible for the diurnal librations). It is at its strongest just above the horizon when the offset is nearly perpendicular to the direction. An example of this can be seen in Figure 5.4. Please note that the size of the Moon has not been scaled up so that the effect is better visible.

As mentioned above, the fragment processor is identical to the Sun's but



(a) : An unfixed position. **(b) :** After the offset.

Figure 5.4: The Moon's apparent position is very different from the center and surface of the Earth.

differs in the active branch. Where the Sun is colored with one single color multiplied by a simple function, the Moon has varying albedo and surface normals. With the way the Moon orbits the Earth, these features change their apparent location on the Moon disk. The Moon also goes through phases that need to be considered, as they are an integral part of Moon observations.

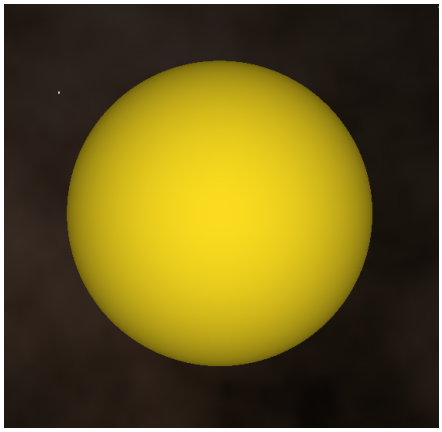
To color the Moon disk, the fragment's lunar latitude γ and longitude λ have to be calculated first. This is achieved through a series of transformations. The sphere normal vector is retrieved as $\boldsymbol{n_s} = (u, v, \sqrt{1-d^2})$, where d is the normalized distance to the center of the disk as with the Sun in Section 5.1. The rotation matrix $R_{g\to M}$, which transforms directions from the global horizon space to the Moon disk plane, given by the uv axes and the direction to the camera, has to be found next as

$$R_{g\to M} = \left[\frac{\boldsymbol{d_M}\times(0,1,0)}{|\boldsymbol{d_M}\times(0,1,0)|}, \frac{(\boldsymbol{d_M}\times(0,1,0))\times\boldsymbol{d_M}}{|(\boldsymbol{d_M}\times(0,1,0))\times\boldsymbol{d_M}|}, -\boldsymbol{d_M}\right]^{-1}, \tag{5.1}$$

where $\boldsymbol{d_M}$ is the direction to the Moon from the Earth. The vector $\boldsymbol{n_s}$ is then rotated by the Moon's rotation R_M in global coordinates from Equation B.12. This rotates the Moon to the correct orientation (here, the matrix in the middle inverts the first row to convert from left-handed to right-handed space):

$$\boldsymbol{n_s}' = (R_{g\to M} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} R_M)^{-1}\boldsymbol{n_s}. \tag{5.2}$$

This approach naturally implements optical librations, as the Moon is rendered with its actual position and orientation. Diurnal librations are a



(a) : Close up of the Sun with a darkened limb.

(b) : Close up of the Moon with shadows in the craters near the terminator.

Figure 5.5: The Sun and Moon are realized in the same way, only colored differently. The atmospheric effects have been removed so the views are clearer.

consequence of moving the observer from the center of the Earth, and physical librations are considered too small to be observable.

Now the selenographical coordinates can be found with trigonometry as

$$\lambda = \arctan 2(n'_{sz}, n'_{sx}), \gamma = \arcsin(n'_{sy}). \tag{5.3}$$

γ and λ can then be used for sampling the Moon textures, namely albedo and normal maps.

Next, the illumination has to be evaluated, for which the normal-map-altered normal $\boldsymbol{n_m}$ is required. For illumination, the Hapke-Lommel-Seeliger photometric function $\boldsymbol{F}(\lambda_i, \phi)$ is utilized[Hap66]. λ_i here denotes the illumination longitude, and ϕ is the Moon's phase angle. Notice that the Moon's appearance is independent of the illumination latitude, as is explained in the paper by Hapke. The phase angle is calculated simply as $\phi = \arccos(-\boldsymbol{d_M} \cdot \boldsymbol{d_S})$. The direction to the Sun is simplified to be the same from the Earth as from the Moon, which saves us more complex calculations. λ_i is a more complicated attribute to compute. The normal $\boldsymbol{n_m}$ is first projected to the plane given by the directions to the Sun and the Earth. The Sun direction is as above, and the Earth direction is trivial in the local quad's space, and it splits the plane into two halfplanes. λ_i is then the angle between this projected vector and the direction toward the Earth, as seen in figure 5.3. If the projected normal lies in the same halfplane as the Sun direction, the value is positive, and it is negative otherwise.

Figure 5.6: A grossly exaggerated intensity of the Moon at a high phase angle. Notice, how the surface in shadow is lit by earthshine.

The Moon fragment color $\boldsymbol{C_m}$ is then given as

$$\boldsymbol{C_m} = (\boldsymbol{H}(\boldsymbol{F}(\lambda_i, \phi)) + \boldsymbol{E_{em}}(\phi)) \cdot \boldsymbol{A_m} \cdot a_m(\gamma, \lambda) \cdot c_m \cdot i_m(d_{Sy}) \tag{5.4}$$

$$\boldsymbol{H}(x) = (1 - e^{-x \cdot 10})^{\frac{1}{2.2}} \tag{5.5}$$

$$\boldsymbol{E_{em}}(\phi) = c_e(-0.0061\phi^3 + 0.0289\phi^2 - 0.0105\sin(\phi)) \tag{5.6}$$

$$i_m(c) = 0.5 + (2 + 2(c + 1.05)^{32})^{-0.5}, \tag{5.7}$$

where $\boldsymbol{H}(x)$ is a simple transformation function to low dynamic range, $\boldsymbol{E_{em}}(\phi)$ is the strength of earthshine, $\boldsymbol{A_m}$ is the general albedo of the lunar surface, $a_m(\gamma, \lambda)$ is the local albedo value fetched from a texture, and $c_m = 3$ is an amplification coefficient to counter balance the fairly low albedo values in the texture the app uses.

Even though the Moon's intensity does not decrease during the day, we perceive it as such due to the amount of light coming from the Sun illuminating the entire sky. To simulate this, an intensity factor $i_m(c)$ dependent on the vertical coordinate of the Sun's direction $\boldsymbol{d_S}$[MED12] is used.

The light coming from the Moon is slightly reddish in hue, so the general surface albedo is set to $\boldsymbol{A_m} = (0.92, 0.79, 0.64)$ to reflect that fact[MED12]. The grayscale local albedo texture a_m is taken from the LRO-LOLA dataset[God]. The normal map of the lunar surface is taken from the Celestia Motherload addon[Mot]. The map has been calculated from the LRO-LOLA data.

Earthshine is an important part of the Moon's illumination. Light reflected off the Earth has a very noticeable effect at small phase angles, as can be seen in Figure 5.6. It doesn't take much to realize that the Earth is in exactly

the opposite phase from the Moon as the Moon is from the Earth. Therefore, earthshine is strongest at a new moon and practically nil during a full moon. The strength of earthshine $\boldsymbol{E_{em}}$ is calculated by Equation 5.6 and the color of the light is set to $c_e = (0.88, 0.96, 1)$[MED12].

An example of the Moon rendered with about 70% of its surface lit can be seen in Figure 5.5b. Notice the shadows near the poles and terminator. Another example can be seen in Figure 5.6, where the Moon is in its waxing crescent phase. Earthshine is strong enough to make some features on the unlit side visible.

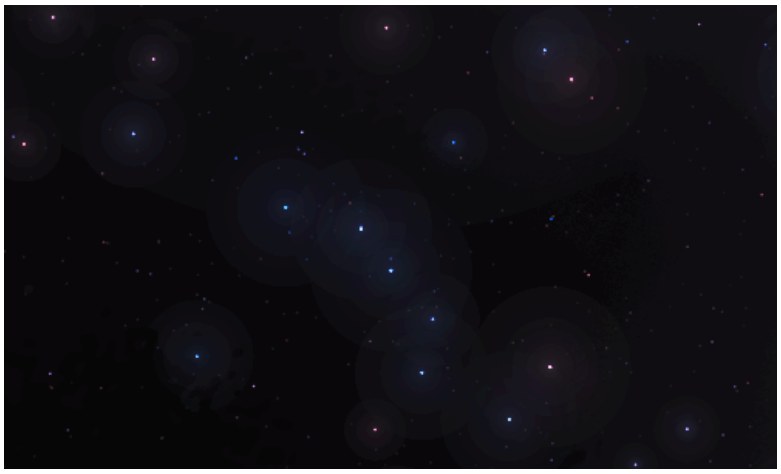
5.3 Stars

There are 9110 stars in the Yale Bright Star Catalogue[HW91], which was utilized for this app, but over one hundred of them have some information missing, so they could not be rendered. The approach chosen for the app is to render the most visible stars independently. The stars are loaded with their right ascension and declination in regards to JD2000, so for each, the direction vector has to be derived from these. Each star also has a magnitude and $B - V$ value. The $B - V$ is crucial for determining the color, but the procedure is not straightforward, as can be seen in Section B.5. The rgb values are rescaled so that the maximum among them is equal to 1. The star data, namely direction in equatorial coordinates, magnitude, and color, are then passed to the GPU as one vertex per star.

Each frame, these vertices are transformed on the vertex processor from equatorial to local horizontal coordinates by the matrix $R_{eq \to ho}$. The movement of the stars in relation to each other is not considered. Each transformed vertex is then passed to the geometry shader to generate a square quad. The size of this quad is $2\sqrt{2}$ pixels high and wide to prevent flickering as the stars move across the screen. This quad has the same size in pixels, independently of the fov or aspect ratio.

Like drawing the Sun and Moon, all fragments outside the circle inscribed to the quad are discarded. In other words, if the calculated distance $d = \sqrt{u^2 + v^2}$ is greater than 1. The color of the fragment is set to the color of

(a) : Zoomed in, the fainter stars can be seen.



(b) : Zoomed out, the fainter stars disappear and the bright loose brightness.

Figure 5.7: Several stars in and around Ursa Major of varying magnitude and the dependency of their appearance on the fov. Note that most of the stars in Figure 5.7a were not visible at all in Figure 5.7b.

the star, and the intensity w of the fragment is[MED12]:

$$w = \min(1, T(d) \cdot I_T) \cdot i_s(d_{Sy}) \tag{5.8}$$

$$T(d) = 1 + 2d^3 - 3d^2 \tag{5.9}$$

$$I_T = \frac{1}{1.167} \cdot B \cdot \frac{c_q}{q^2} \tag{5.10}$$

$$q = 2\sqrt{2} \cdot \tan(\frac{1}{2} fov_y)/res_y \tag{5.11}$$

$$B = 2.512^{m_a - m} \tag{5.12}$$

$$i_s(c) = (1 + (c + 1.14)^{32})^{-0.5}, \tag{5.13}$$

where m is the star's magnitude, fov_y is the vertical fov, res_y is the vertical resolution, and c_q is a scaling constant, estimated to around $c_q \approx 4 \cdot 10^{-6}$. $T(d)$ is the point spread function (PSF) and controls the intensity falloff from

the center to the edge of the star disk. $m_a = 4$ is the control magnitude that globally controls the brightness of the stars. q can be considered the size of the star if it changed with the fov, just like Müller et al. implemented[MED12].

I_T is a scale for the PSF. Through relative brightness of the star B, it increases as magnitude decreases (remember that lower magnitude means higher intensity). Through q, the scale is inversely proportional to vertical fov and directly proportional to vertical resolution. The purpose of this is to prevent having too many bright stars on the screen at once. As the camera zooms in and fov decreases, the stars appear brighter as if focusing on them. Similarly, as the vertical resolution is lowered and the aspect ratio thus increases, and the stars are compressed toward the center of the screen, their intensity decreases to prevent overcrowding. The 1.167 constant is the disk integration of the PSF function, which is thus normalized when divided by the constant.

w is used as alpha for blending all stars' contributions to the sky. The stars are usually far away enough from each other to where not even their glares have any significant overlap. There is, therefore, no particular ordering of star fragments, and it is left entirely up to the GPU.

As mentioned, the brightness is inversely related to the fov. This relation results in the dimming of stars as the camera zooms out and brightening with zooming in. This creates an interesting effect, where one can inspect individual stars if one wants to, but the constellations remain easily recognizable at certain zoom levels. An example of the impact the fov dependency has is in Figures 5.7. In Figure 5.7a the blue star with high intensity is the second to last star of Ursa Major's tail. Ursa Major can be seen in Figure 5.7b at the center to the bottom right.

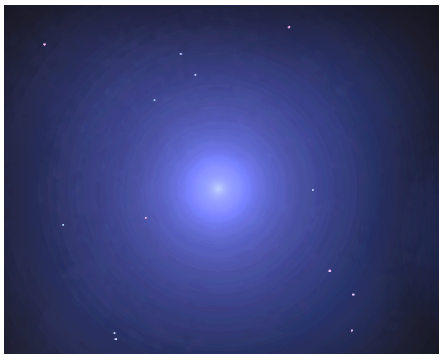
The stars' intensities (along with their glares and background, explained further below) are additionally scaled with an intensity factor, similar to the Moon. This factor i_s ensures the stars fade in/out during twilight as their light is drowned out by that of the Sun.

5.3.1 Star glare

Since the stars do not change their size, only their intensity, they can only get so bright, and at some point, their brightness becomes somewhat indistinguishable from the other stars. An example can be seen in Figure 5.8,

Figure 5.8: At high zoom levels, constellations become unrecognizable.

(a) : Orion's belt, three bright stars next to each other.



(b) : Sirius' glare coves many stars around it.

Figure 5.9: Some stars can have a very strong glare when zoomed in.

a very close zoom of the Pleiades. Since all the stars become visually very similar, we lose our ability to distinguish the bright ones. Instead of the "seven sisters" as they are also called, we see an assortment of bright dots without any information about their brightness differences (please note that to make the colors of the stars stand out more on paper, the colors have been heavily altered in post-process throughout this thesis).

To help mitigate this problem, stars are highlighted with a glare from a certain magnitude. Like the star itself, the glare is rendered on a quad generated on the geometry processor. If the star has a glare, there is, in fact, only one quad, and the star and glare are both drawn onto it and blended together. Where the stars have the same size but different PSF scales, their glares are all different sizes with the same PSF scale.

The glare is applied only for stars with a magnitude less than 4. The diameter Q of the glare quad is set to:

$$Q = 2\frac{\sqrt[4]{I_G}}{8} \tag{5.14}$$

$$I_G = (2.512^{m_a - mag - 0.167} - 1) \cdot \frac{c_q}{q^2}, \tag{5.15}$$

where q is as above in Equation 5.11. Q is the size of the quad in the normalized coordinates $[-1, 1]^3$. The geometry processor then passes the ratio of the star's and glare's diameters to the fragment processor, where the distance to the center for the purposes of drawing the star itself is determined as $d' = d \cdot ratio_{dist}$.

The color of the fragment from Equation 5.8 is modified as such:

$$w = (w_s(1 - w_g) + w_g) \cdot i_s(d_{Sy}) \tag{5.16}$$

$$w_s = \min(1, T(d') \cdot I_T) \tag{5.17}$$

$$w_g = \min\left(1, \frac{G(d)}{\tan(\max(0.2, fov_y) \cdot 0.5)}\right) \tag{5.18}$$

$$G(d) = 1 - \sqrt[64]{d}. \tag{5.19}$$

Notice that the fov is limited to a maximum of 0.2 radians. This prevents the glare quad from getting too big and separates the stars a little better when zooming in. It also prevents the star's glare from overpowering the image with a solid color since it visibly shrinks. Some examples of strong glares can be seen in Figure 5.9. G is a very basic PSF that only lightly increases the brightness of the pixels, but it gives a decent highlight to the rendered stars.

Notice the two ways the glare is decreased in intensity in equation 5.14: the root and the division. Dividing the size by a number has the simple effect of scaling glares of all sizes in the same way. However, changing the root index increases the size of smaller glares and decreases the size of larger ones. In this way, larger and smaller glares can be tweaked more independently.

The glares are then blended onto the resulting image using the star's own color and the calculated w as alpha, just as the stars were. The ordering is left up to the GPU, but that does not appear to be a problem when looking at the results, probably because of the low alphas and low amount of overlap.

5.3.2 Star scintillations

Scintillations are realized with Perlin noise affecting the resulting color weight w multiplicatively. At the app's start, 1 octave of 1-d Perlin noise texture is generated. The noise has 1000 samples in the range $[0, 1]$, but that is later rescaled in the shader. The vertex processor then hashes the id of each star and uses that as an offset for sampling the texture. This offset is added to the time offset from the app's start and is used as the sample coordinate for the noise. The value that then passes through the geometry shader into the fragment processor is

$$noise = 1 - P(H(id) + t) \cdot (1 - 0.6 \cdot p_y), \tag{5.20}$$

where P is the Perlin noise (sampled with a float in the range $[0, 1)$), H is the mentioned hash function that takes the star's id, t is a float number of seconds since the start of the app, and d_y is the y coordinate of the star's direction vector. The second factor in the equation is the strength of the flickering given by the star's height above the horizon. The scaled noise value is subtracted from one to keep the maximum value so that it can be simply multiplied by the star's intensity later.

The passed value is used on the fragment processor to decrease the intensity of the color drawn to the screen. This is achieved by multiplication with the color weight w. The two calculations then change as such:

$$w_s = \min(1, \min(1, T(d') \cdot I_T) \cdot (noise \cdot 0.3 + 0.7)) \tag{5.21}$$

$$w = (w_s(1 - w_g) + w_g) \cdot (noise \cdot 0.15 + 0.85) \cdot i_s(d_{Sy}). \tag{5.22}$$

In both cases, the noise is rescaled to a different interval. For the star itself, the $[0, 1]$ interval is rescaled to $[0.7, 1]$ to limit the magnitude of scintillations and make them less noticeable. For the star after the glare was applied, the value is rescaled to $[0.85, 1]$ for the same reason. This interval is smaller because the glare is much bigger than the star itself and the intensities are much lower. Using the same interval as for the star dot itself yielded a very noticeable oscillations in intensity, and the glare was visibly getting smaller and larger by a significant margin.

5.3.3 Starry background

There are only 9110 stars in the catalog, but there are billions just in our galaxy. The Milky Way is a significant feature visible to the naked eye at night. Without it, drawing only the stars, Moon, and Sun, the sky feels

Figure 5.10: Galactic background. The brightest star is Alpha Centauri. Notice how faint many other stars are, which is a testament to Alpha Centauri's brightness.

empty. While many of the stars are not visible individually, their sheer number produces enough light to be visible.

An equirectangular texture with the most bright stars pre-removed was chosen as the background to be rendered behind the stars[Wri20]. The texture, however, features noise even in areas other than the galaxy plane. While this noise could be true to some actual light coming to Earth from these directions, it turned out to be very distracting in the app. The noise is not of very high intensity on its own, but the stars' glare would amplify it, and thus it warrants removal. Figure 5.10 shows the texture used as the background.

The texture has been converted to a cubemap to fix the distortion of the poles. For each fragment, the world direction is calculated and transformed to equatorial coordinates with the inverse of the matrix $R_{eq \to ho}$. The transformed direction is used for sampling the cubemap. The color retrieved is then divided by 4 to lower the intensity of the galaxy to where it does not overpower the stars but only serves as their background.

To help align the stars with the galactic background, the constellation map, provided from the same source, has been used [Wri20]. The texture connects the appropriate stars and highlights misalignments, where the constellation lines are missing bright stars. The same intensity factor from Equation 5.13 dims the texture's color as the stars themselves. This gives it a fluid transition between being fully visible at night and invisible during the day.

5.4 Atmospheric scattering

The app implements Equations 2.1–2.4 with four precomputed lookup tables. One of those tables extends the equations by the multiscattering of light. The textures will be explained in the following subsections, and a description of how they are integrated into the app will follow.

5.4.1 Atmospheric model

The atmosphere is simulated as a constant-thickness layer around a spherical Earth. The radius of the Earth is set to $R_g = 6360$km and the radius of the top atmospheric boundary is set to $R_a = 6460$km. This leaves us with a 100km thick atmosphere. Throughout this section, the value h will represent the altitude above the Earth's surface. Since only views from within the atmosphere are assumed, this means h will always be constricted to the interval $[0, 100]$km. The ground is considered to have a uniform albedo $\rho = 0.3$ for all wavelengths. However, no multi-spectral representation is used. Only the usual rgb-based representation will be used throughout.

The atmosphere consists of three main components that will be considered in simulating the atmosphere's interaction with the light, each with a different phase function, density distribution, and scattering and extinction coefficients.

Rayleigh particles are molecules of the air. The theory assumes that light is never absorbed, only scattered. Thus the scattering and extinction coefficients are equal. Its phase function is a simple-looking curve with a peak at the front and back. Their density follows a simple geometric curve.

Mie particles are the aerosols scattered in the air, like dust or pollution. These particles can both scatter and absorb light, so the coefficients are different, but the phase function is much more complex. The density again falls exponentially but much faster compared to the Rayleigh particles.

Ozone particles are the third and final considered type of particle. These are ozone molecules and are assumed only to absorb light, so the scattering coefficient is zero, and it has no phase function. What is interesting about ozone particles is that they mostly appear in a continuous layer above the ground. Above and below this layer, the density is almost negligible.

Figure 5.11: The transmittance texture characterizing the optical depth of the atmosphere.

These are the values for various attributes of the atmosphere (all σ values have units km^{-1}):

$$\begin{aligned} \sigma_s^r &= (5.802, 13.558, 33.1)10^{-3} & \sigma_t^r &= \sigma_s^r \\ \sigma_s^m &= 3.996 \cdot 10^{-3} & \sigma_t^m &= 4.4 \cdot 10^{-3} \\ \sigma_s^o &= 0 & \sigma_t^o &= (0.650, 1.881, 0.085)10^{-3} \end{aligned} \tag{5.23}$$

$$\begin{aligned} p^r(\theta) &= \frac{3(1+\cos(\theta)^2)}{16\pi} \\ p^m(\theta, g) &= \frac{3}{8\pi}\frac{(1-g^2)(1+\cos(\theta)^2)}{(2+g^2)(1+g^2-2g\cos(\theta))^{3/2}} \\ p^u &= \frac{1}{4\pi} \end{aligned} \tag{5.24}$$

$$\begin{aligned} d^r(h) &= e^{\frac{h}{H^r}} \\ d^m(h) &= e^{\frac{h}{H^m}} \\ d^o(h) &= \max(0, 1 - \frac{|h-25|}{15}). \end{aligned} \tag{5.25}$$

Superscripts ^{r}, ^{m}, and ^{o} represent the Rayleigh, Mie, and ozone particles, respectively. Subscripts $_s$ and $_t$ are for scattering and extinction. p are phase functions of individual scattering models, and d are the densities of various particle types. Notice the uniform function p^u, which will be useful later for multiscattering. θ is the angle between incoming and scattered directions, and $g = 0.6$ is the asymmetry parameter. Finally, $H^r = 8\text{km}$ and $H^m = 1.2\text{km}$ are heights of the atmosphere if the densities were uniform.

5.4.2 Transmittance LUT

The first of the textures, transmittance, precomputes Equation 2.2. This is the simplest of the equations, but its values are used throughout the computations. It is, therefore, essential to precompute it, as raymarching the atmosphere is expensive and can quickly add up. As it only changes when

the attributes of the atmosphere change, it is enough to precompute it once at the start of the app.

The texture has to store information about pairs of points around the Earth $T(\boldsymbol{x_a}, \boldsymbol{x_b})$, which is 6 variables. However, because the of the nature of the computation being geometric, we can use this to store the value of transmittance all the way to the atmospheric boundary $T(\boldsymbol{x_a}, \boldsymbol{v})$. To get the desired transmittance between two given points, we use the identity $T(\boldsymbol{x_a}, \boldsymbol{x_b}) = T(\boldsymbol{x_a}, \boldsymbol{v})/T(\boldsymbol{x_b}, \boldsymbol{v})$ where $\boldsymbol{v} = \frac{\boldsymbol{x_b}-\boldsymbol{x_a}}{|\boldsymbol{x_b}-\boldsymbol{x_a}|}$.

The values stored in this texture represent the relative amount of light passing through the atmosphere. The theory achieves this with a geometric curve and integrating the extinction coefficients along the ray's path. The approximation raymarches the atmosphere and imitates this process with a simple numerical integration:

$$T(\boldsymbol{x_a}, \boldsymbol{v}) = e^{-\sum_{i=1}^{N_{max}} \sigma_t(\boldsymbol{x_a}+i\frac{t_{atmo}}{N_{max}}\boldsymbol{v})\cdot\frac{t_{atmo}}{N_{max}}}, \tag{5.26}$$

where $N_{max} = 40$ is the number of integration steps and t_{atmo} is the length of the ray.

To avoid high dimensionality of the texture, spherical symmetry is used. The point $\boldsymbol{x_a}$ is assumed to be on the vertical axis with a distance to the center of the Earth r, and the direction $\boldsymbol{v}$ is given only as the cosine of the angle to the zenith μ. This loses no information but lowers the dimensionality to two. Furthermore, as the same information would be essentially stored twice, only those rays that do not intersect the Earth are stored, which is achieved with the following parametrization of uv coordinates:

$$\begin{aligned} H &= \sqrt{R_a^2 - R_g^2} \\ \rho &= H \cdot v \\ r &= \sqrt{\rho^2 + R_g^2} \\ d &= R_a - r + u(\rho + H - R_a + r) \\ \mu &= \frac{H^2 - \rho^2 - d^2}{2 \cdot r \cdot d}. \end{aligned} \tag{5.27}$$

The texture's appearance for the atmospheric model described above is in Figure 5.11. The precomputation was sped up with a compute shader on the GPU, with a thread-per-texel distribution.

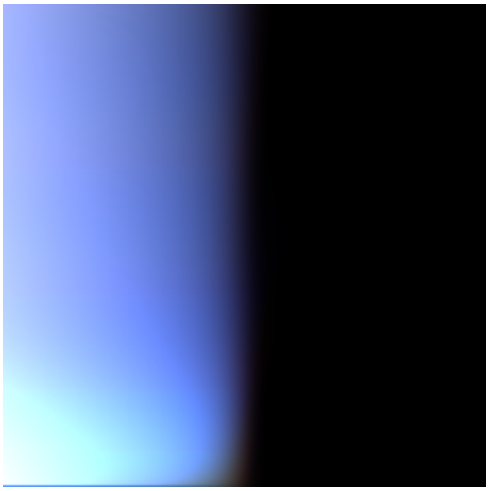


Figure 5.12: The multiscattering texture is not large and the values stored are very small. In this picture, the values have been multiplied by 50 and the resolution upscaled.

5.4.3 Multiscattering LUT

The second LUT stores information about the total scattered light of order 2 and above and is not connected directly to any equation mentioned above. This texture, again, is independent of time and only changes with the atmospheric properties. It is, therefore, enough to precompute this texture once at the start of the app, as can be seen in Figure 4.1. The values stored are for a given point $\boldsymbol{x_s}$ in the atmosphere with a given Sun direction $\boldsymbol{\omega_s}$. The parametrization can help reduce the dimensionality in this case as well:

$$\begin{aligned} r &= 100v + R_g \\ \mu_s &= 1 - 2u. \end{aligned} \tag{5.28}$$

Here, μ_s is the cosine of the angle between the Sun's direction and the zenith.

To find the multiscattered light, several attributes have to be found. The first of them is the second-order scattered luminance $\boldsymbol{L_{2nd}}$ as

$$\boldsymbol{L_{2nd}} = \int_{\Omega_{4\pi}} L'(\boldsymbol{x_s}, -\boldsymbol{\omega}) p^u d\boldsymbol{\omega} \tag{5.29}$$

$$L'(\boldsymbol{x}, \boldsymbol{v}) = T(\boldsymbol{x}, \boldsymbol{p}) L_o(\boldsymbol{p}, \boldsymbol{v}) + \tag{5.30}$$

$$\int_{t=0}^{|\boldsymbol{p}-\boldsymbol{x}|} \sigma_s(\boldsymbol{x} - t\boldsymbol{v}) T(\boldsymbol{x}, \boldsymbol{x} - t\boldsymbol{v}) S(\boldsymbol{x} - t\boldsymbol{v}, \boldsymbol{\omega_s}) p^u dt \tag{5.31}$$

$$\boldsymbol{L_{2nd}} = \sum_{i=1}^{N_{max}} L'(\boldsymbol{x_s}, -\boldsymbol{\omega_i}) \frac{1}{N_{max}}. \tag{5.32}$$

L' is the incoming luminance along one ray, and L_o is the reflected luminance from the Earth's surface if the ray intersects it. The spherical integration

in Equation 5.29 is done with $N_{max} = 64$ samples spread by the Fibonacci lattice. This gives something very close to an even distribution of points. Equation 5.32 shows how the numerical integration of the sphere changes the calculation. L' is integrated in a similar manner as the transmittance above.

The second property is the factor $\boldsymbol{f_{ms}}$ representing the transfer of energy from around the sample point. This value can be found as

$$\boldsymbol{f_{ms}} = \int_{\Omega_{4\pi}} L_f(\boldsymbol{x_s}, -\boldsymbol{\omega}) p^u d\boldsymbol{\omega} \tag{5.33}$$

$$L_f(\boldsymbol{x}, \boldsymbol{v}) = \int_{t=0}^{|\boldsymbol{p}-\boldsymbol{x}|} \sigma_s(\boldsymbol{x} - t\boldsymbol{v}) T(\boldsymbol{x}, \boldsymbol{x} - t\boldsymbol{v}) dt \tag{5.34}$$

$$\boldsymbol{f_{ms}} = \sum_{i=1}^{N_{max}} L_f(\boldsymbol{x_s}, -\boldsymbol{\omega_i}) \frac{1}{N_{max}}. \tag{5.35}$$

Notice the similarity to the previous equations. The transfer factor does not consider Sun visibility or the phase function, as it does not integrate the scattered light per se. sphere integration in Equation 5.33 is approximated the same way as before, resulting in Equation 5.35. L_f is again integrated in the same way as the transmittance above. f_{ms} represents only one transfer; to get the infinite scattering factor, we calculate the infinite geometric series sum

$$\boldsymbol{F_{ms}} = 1 + \boldsymbol{f_{ms}} + \boldsymbol{f_{ms}}^2 + \cdots = \frac{1}{1 - \boldsymbol{f_{ms}}}. \tag{5.36}$$

The total infinite scattering luminance, which is stored directly in the LUT, can be then calculated as the second-order scattering magnified by the infinite energy transform factor:

$$\boldsymbol{\psi_{ms}} = \boldsymbol{L_{2nd}} \boldsymbol{F_{ms}}. \tag{5.37}$$

Equation 2.3 can then be modified with the precomputed multiscattering approximation values as

$$L_{scat}(\boldsymbol{c}, \boldsymbol{x}, \boldsymbol{v}) = \sigma_s(\boldsymbol{x}) \sum_{i=1}^{N_{light}} (T(\boldsymbol{c}, \boldsymbol{x}) S(\boldsymbol{x}, \boldsymbol{l_i}) p(\boldsymbol{v}, \boldsymbol{l_i}) + \boldsymbol{\psi_{ms}}(\boldsymbol{x}, \boldsymbol{l_i})) E_i. \tag{5.38}$$

This effectively removes the need for an iterative method but requires more assumptions that somewhat simplify the model. The precomputation was sped up with a compute shader on the GPU with 64 threads per texel. Each thread integrates the atmosphere in one spherical sample, and a parallel reduction is used to sum the contributions from all threads. 40 samples along each ray are taken. The texture can be seen at a much higher resolution in Figure 5.12. The colors in the texture were multiplied by 50 to be visible. The flash of light at the bottom is primarily due to the light reflected off the Earth's surface L_o in Equation 5.31. There is also a row of dark blue pixels at the bottom as a result of the proximity of the ground, where not much light can scatter in from the bottom.

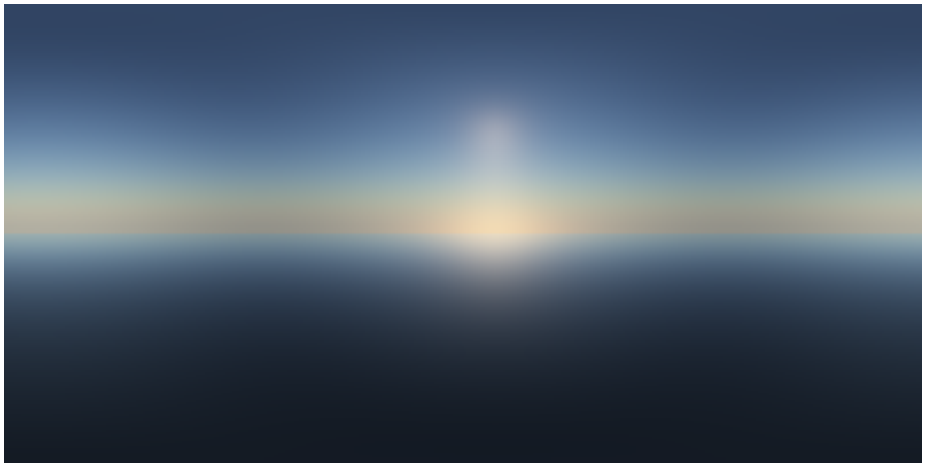


Figure 5.13: The prerendered sky at a low sample rate. Notice the distorted area around the horizon.

5.4.4 Sky view LUT

The sky view LUT precomputes the sky's appearance according to Equation 2.1. However, since the frequency of change is so small, it can be rendered at a much lower resolution, thus requiring fewer ray marches. This has the further benefit of being independent of the resolution of the final image.

The only area with a high frequency of change is the horizon. Especially during twilight, there are not enough texels to represent the horizon's appearance properly. Therefore, a better parametrization is used, so more texels are utilized around the horizon. For uv coordinates, the azimuth λ and elevation β are as follows:

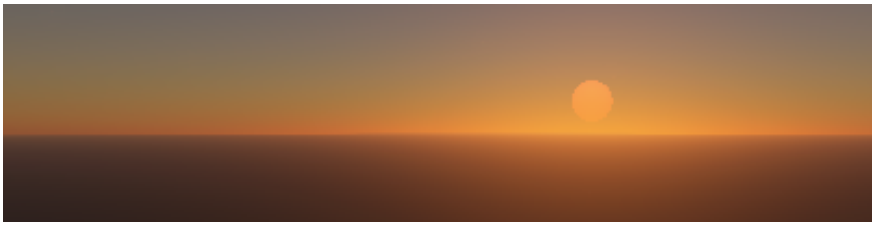
$$\lambda = u \cdot 2\pi \tag{5.39}$$

$$\beta = \begin{cases} \beta_H(1-(2v-1)^2) - \frac{\pi}{2}(2v-1)^2 & v \leq 0.5 \\ \beta_H(1-(2v-1)^2) + \frac{\pi}{2}(2v-1)^2 & v > 0.5 \end{cases} \tag{5.40}$$

$$\beta_H = \arcsin(R_g/r) - \frac{\pi}{2},$$

where β_H is the elevation of the true horizon. Notice the dependence on the point's distance from the Earth's center r. β is calculated as the linear interpolation of the true horizon's elevation and the zenith (or the opposite to the zenith for texels below the horizon). The weight of the interpolation is a simple parabolic curve. The effect this parametrization has can be seen in Figure 5.14. This parametrization prevents artifacts around the horizon both in the form of the noticeable bands of color and another form of temporal artifact, where the color within the bands changes from side to side as the Sun slowly sets.

(a) : Naive parametrization yields severe artifacts.



(b) : Better parametrization improves the look of the horizon.

Figure 5.14: The difference the vertical parametrization makes is significant. Notice the visible layers in Figure 5.14a.

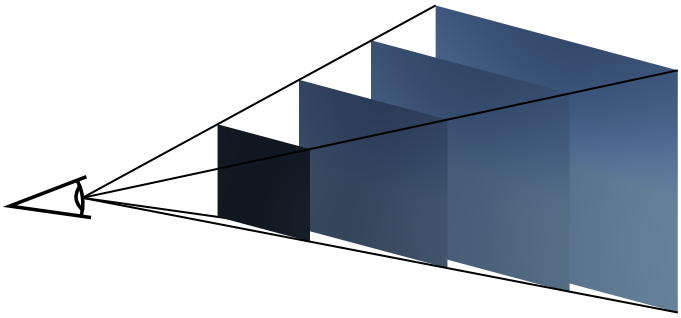


Figure 5.15: The texture can be interpreted as slices for various depths.

This LUT needs an update every time the illumination parameters change. This can be the illuminator's direction (the Sun does not have to necessarily be the only illuminator of the sky), but even the atmospheric properties or the observer's altitude. The precomputation was sped up with a compute shader on the GPU on a thread-per-texel basis. Before the value is stored in the texture, it goes through the same basic $HDR \rightarrow LDR$ function H from Equation 5.5 for rendering the Moon.

5.4.5 Aerial view LUT

The last of the four LUTs is the 3D aerial view texture. This texture recreates the light scattering that happens in between the camera and the terrain, whereas the sky view texture only precomputes for rays that exit the atmosphere. Each slice along the third axis behaves practically the same as

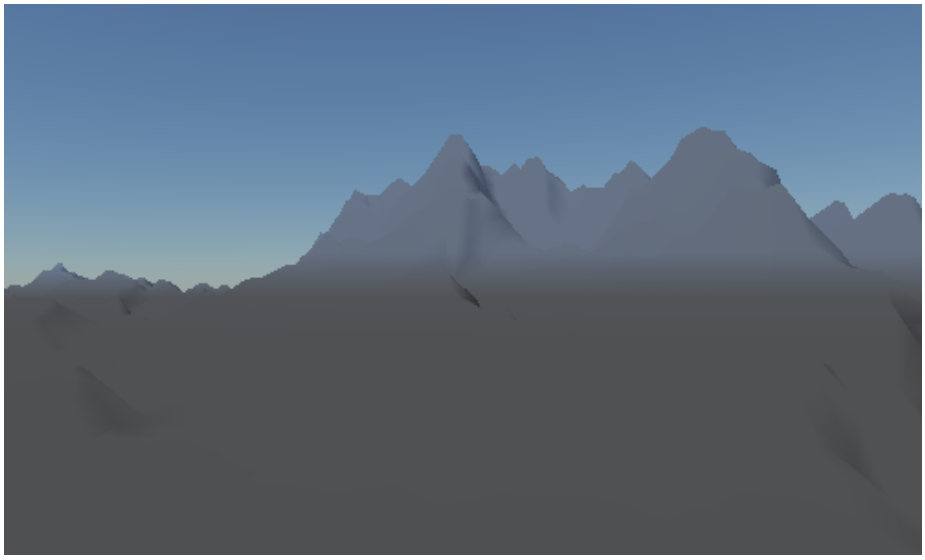


Figure 5.16: Naive integration of the paths. Notice how the top half with rays that reach the terrain is much more blue than the bottom, where rays stop immediately when intersecting the Earth.

the sky view LUT. The depth of the integration is limited, and the texture is fixed to the view frustum of the camera as in Figure 5.15.

Each slice has its own cutoff distance. These distances were chosen to be evenly distributed up to 32km, so for 32 slices, that is 1km per slice. The slice for depth 0 is trivially black, so that we can start at a distance of 1km, but we have to take that into account when sampling.

The integration process is very similar to that of the sky view LUT, but one difference must be considered. As the cutoff is strictly given by the slice index, looking down at the Earth's surface has to be handled as a special case. When not addressed, it can have extreme consequences, as seen in Figure 5.16. This is solved by clamping the densities to altitude $h = 0$. For the purposes of Sun shadowing, all underground samples are supposed to be on the surface of a sphere concentric with the Earth. This gives good and seamless-looking results.

The fourth value of the texture is used for storing the average transmittance across all three rgb components. This is done for several reasons. Firstly, combining them together does not change the color, only the intensity, which is the more desired of the two options. Secondly, it saves us time either looking up the transmittance texture or integrating manually. Thirdly, as some rays may go underground, transmittance texture does not always have the correct data, but this approach prepares it specifically for this use.

As the values change with time, atmospheric properties, and camera movement, the texture is recalculated every frame. The precomputation is again sped up with a compute shader on the GPU with a thread-per-texel distribution. Before the value is stored in the texture, it goes through the same basic $HDR \rightarrow LDR$ function H from Equation 5.5 for rendering the Moon.

5.4.6 Atmospheric rendering

When the LUTs are ready, and everything beyond the atmosphere has been drawn, the sky itself is rendered. A quad across the whole screen is drawn, and to color it, the render of the stars is blended with the sky view LUT.

The world direction $\boldsymbol{d_f}$ of the fragment has to be determined first. From it, we can get the azimuth and elevation to sample the sky view LUT. The color C'_f of the fragment is then given by

$$C'_f = C_f T(\boldsymbol{c}, -\boldsymbol{d_f}) + S_t(\lambda, \beta), \tag{5.41}$$

where C_f is the background color and S_t is the sky view texture sampled by the azimuth λ and elevation β. The background color is multiplied by the transmittance to the edge of the atmosphere. This gives the background a red hue shift near the horizon.

When the sky is drawn, the terrain comes next. The distance of the fragment is converted to the slice index to be sampled (not necessarily integer), and trilinear interpolation is used to get the desired value. The retrieved color is added to the color of the terrain multiplied by the all-spectrum transmittance stored in the fourth component. A more detailed description of the terrain rendering follows below.

5.5 Terrain

The scene rendered in the app is a simple procedurally generated terrain. The scale of the terrain is 6000 units in both horizontal directions and 372 in the vertical. The kilometer-to-unit ratio has been chosen as 1/300, so in kilometers, the size is $20 \times 20 \times 1.24$km. The size really only matters for the aerial view LUT application.

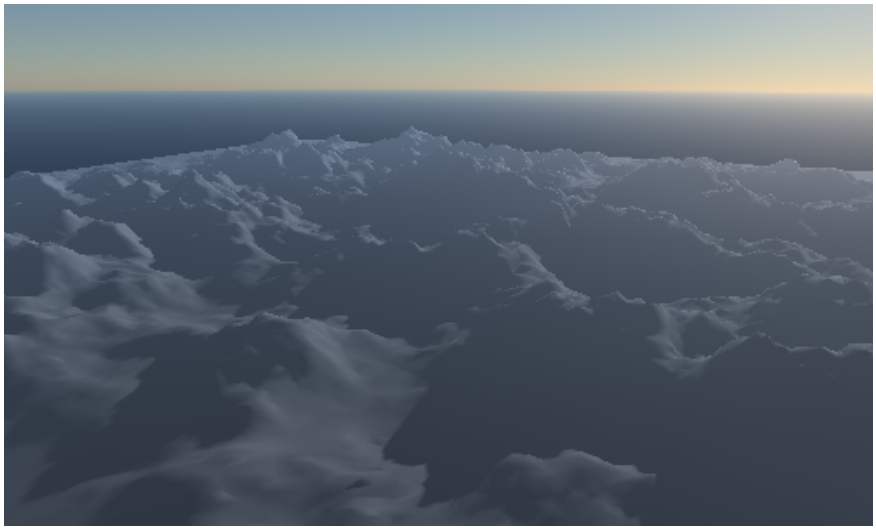


Figure 5.17: The scene is a single subdivided quad with procedural terrain.

The terrain is illuminated by two sources of light: Sun and Moon. Their intensities I_s and I_m are calculated on the CPU as ratios of their optical area above the horizon to their total area:

$$I_s = \frac{\phi_v - \sin(\phi_v)}{2\pi} \tag{5.42}$$

$$\phi_v = \begin{cases} 2\arccos(\min(1, \frac{\arcsin(|d_{Sy}|)}{s_S})) & d_{Sy} \leq 0 \\ 2\pi - 2\arccos(\min(1, \frac{\arcsin(|d_{Sy}|)}{s_S})) & d_{Sy} > 0, \end{cases} \tag{5.43}$$

where ϕ_v is the visible angle of the Sun disk. I_m is calculated in the same manner, only with the height d_{My}, and it is further multiplied with the illuminated fraction of the lunar disk $\frac{1-\boldsymbol{d_S}\cdot\boldsymbol{d_M}}{2}$. s_S is the angular size of the Sun increased tenfold. The illumination contributions L_S and L_M from both light sources are calculated for each fragment of the terrain independent of each other and multiplied by their appropriate intensity. they are then blended as such:

$$(1 - I_S)^2 L_M + L_S. \tag{5.44}$$

This gives a smooth transition from sunlight to moonlight illumination.

For shadows, a shadow map is used. The size of the terrain would need a huge texture to be able to draw nice shadows, so percentage closer filtering is applied. The kernel has the size 3×3 and utilizes a Gaussian blur kernel for sampling the maps. Both the Sun and the Moon cast shadows, and the illumination calculation uses them even during blending.

After the illumination is evaluated, the aerial view texture is sampled to get the transmittance and inscattered light.

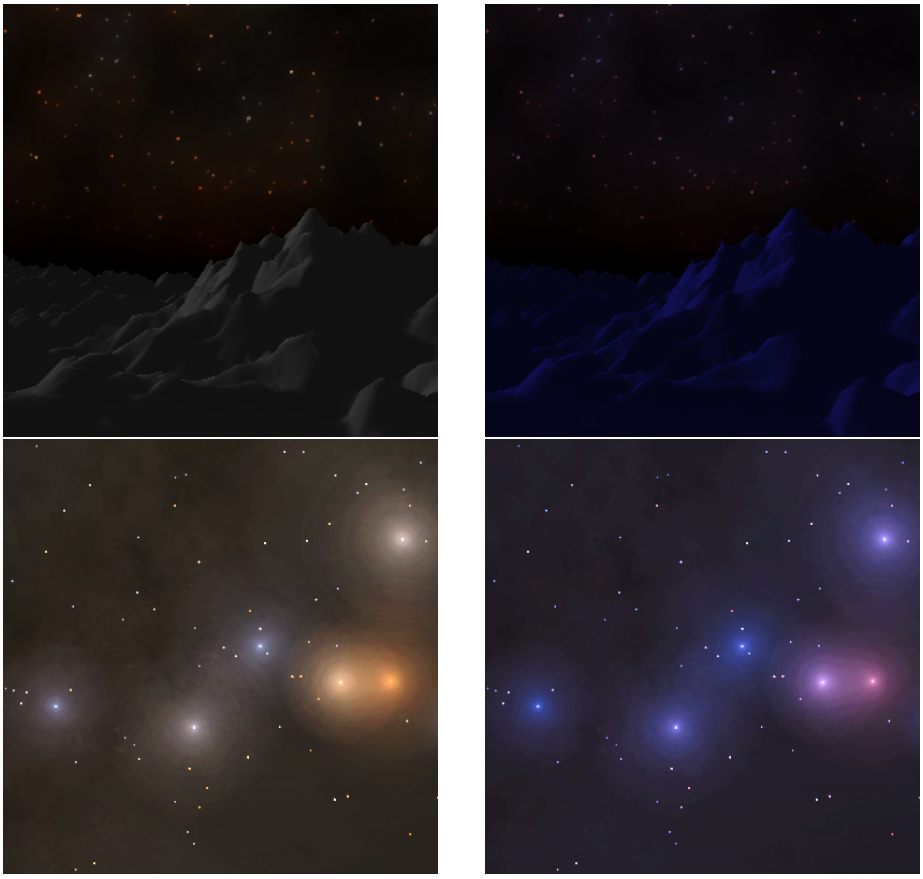


Figure 5.18: Illustration of the difference between mapping the scene (top row) and the stars (bottom row). In the left images, the tonemapping is not applied and you can see that it does not resemble night much.

5.6 Tonemapping

Everything that has been mentioned before is drawn into one texture in a frame buffer. This texture is then drawn across the screen so that each pixel can be processed for the last operation, tone mapping.

The blue shift is modeled by calculating the response of the rods, transferring it to the s cones (cones reacting to short wavelengths), and transforming back into sRGB. Firstly, the rod response R_r is calculated from the pixel color as:

$$R_r(I) = B \cdot f(I \cdot F) \tag{5.45}$$
$$f(I) = 40\frac{I^{0.73}}{I^{0.73} + 2} \tag{5.46}$$
$$I = 0.2126r + 0.7152g + 0.0722b \tag{5.47}$$
$$B = \frac{0.04}{0.04 + I_A} \tag{5.48}$$
$$F = 3800j^2\frac{5I_A}{2.26} + 0.2(1 - j^2)^4(\frac{5I_A}{2.26})^{\frac{1}{6}} \tag{5.49}$$
$$j = \frac{1}{5.1^5 I_A + 1}, \tag{5.50}$$

where r, g, and b are the three color components, I is the intensity of the pixel, and I_A is the adaptation intensity. The two factors F and B simulate the neural adaptation and the bleaching and recovery of photopigment. In the sense of the returned values, they shift the response curve to the right and decrease the amplitude, respectively, although they have minimal effect for the small range of values in the app.

A part of this response is then carried over to the short-wavelength cones, while the others remain unstimulated, so $R_s = 0.2 \cdot R_r, R_m = 0, R_l = 0$. These responses are then run through the inverse of the function R_r, as the cones and cells are assumed to behave the same in this regard. The inverse is

$$I(R) = \frac{2^{1/0.73}}{F(40\frac{B}{R} - 1)^{1/0.73}}. \tag{5.51}$$

From this, we get the intensity I_s of the light that would have to stimulate the s cones to get the same response as 20% of the rods' response. The intensities I_m and I_l are zero.

Next, the calculated intensities have to be converted to the XYZ tristimulus values. For this purpose, a transformation matrix has been devised by Hun[Hun05]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1.9102 & 0.371 & 0 \\ -1.1121 & 0.6291 & 0 \\ 0.2019 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_l \\ I_m \\ I_s \end{bmatrix}. \tag{5.52}$$

From here, the tristimulus is converted to xyY and then back to XYZ, but as intensity Y in xyY space, the intensity I as described above is used. From XYZ, we can finally transform to sRGB space.

If these colors were drawn directly, the night would feel unnaturally blue, so some blending takes place to rectify that. For the stars, starry background,

Sun, and Moon, the weight of the tone-mapped colors is given as $0.3 \cdot (1 - \sqrt[4]{I_S})$. For the scene, it is given as $0.7 \cdot (1 - \sqrt[4]{I_S})$, where I_S is the intensity of sunlight calculated in the previous section. The weight of tone mapping for stars is significantly lower to keep their colors from changing too much. Since the colors shift toward blue, the stars all start looking the same color, which is not what we see at night. They consequently start being harder to recognize since some stars are connected to their color (like Polaris being white or Betelgeuse being red). The difference between the two weights of tonemapping can be seen in Figure 5.18.

The various features to be given different tonemapping treatments are recognized through their fourth color component w. The star background, the Sun, and the Moon are all drawn with $w = 1$. The stars are blended into the image with the help of their alpha channel, but they never have $w = 0$. The terrain is the only rendered object to be rendered with $w = 0$. Thus, when tonemapping, the non-zero alpha channel triggers the lower intensity of the hue shift.

5.7 Interaction

The user can interact with the app in two major ways. The first one is manipulation with the camera, and the other is changing the time and location data used for rendering the sky.

The camera can be manipulated in several ways. By dragging the left mouse button across the window, the camera rotates around its global x and y axes and, as such, looks around the scene and sky. The camera can also zoom in and out with the scroll wheel. Scrolling up will decrease the fov, concentrating the camera on finer detail, whereas scrolling down will zoom out to show more of the scene and sky. Pressing the middle mouse button will reset the fov back to the default 90°.

The camera can be moved around using the keyboard. Holding `W` moves the camera in the direction it is looking in, and holding `S` moves it in the opposite direction. Holding `A` moves the camera in the direction of the local left vector, and holding `D` moves it to the right. Holding the space bar will move it in the direction of the local up vector, and holding the left `Ctrl` key will move it down. Holding any two keys for the opposite movement will cancel them out. The movement speed is 5 units per frame, so 150 units every second (0.5km/s). When the left `Shift` key is held in addition, the movement is slowed down to 10%.

The time and location data are manipulated through the console. The console opens as another window next to the main window with the 3d view and is processed by another thread on blocking read. The thread expects one command on each line. There are three commands that can be executed: `help`, `set`, and `get`. The `help` command prints some information about how to interact with the app. The remaining two commands expect an argument. The argument has to be one of: `year`, `month`, `day`, `hour`, `minute`, `second`, `timezone`, `speed`, `longitude`, `latitude`, `altitude`. The first eight are time data, the last three pertain to location.

The `get` command will print the current value of the requested attribute. For example, typing `get speed` will print the speed at which the time passes. The `set` command, on the other hand, expects another argument to set the value of the given attribute to. For example `set timezone 1` will set the timezone to +1 UTC. When setting time, it has to be set one attribute at a time, but the date has to be left in a valid state. When the date is January 31st, for example, `set month 2` is impossible as such date does not exist, and the `day` attribute has to be changed first.

Time calculations use the Gregorian calendar. The switch from Julian in the year 1582 is not considered, so dates before that will be inaccurate. A warning is displayed when switching to a date before the switch to the Gregorian calendar was made. If time-correct renderings are needed, the date must be manually converted and entered in that form.

The values can further be inputted with console parameters during the app's start. Using the parameter `-h` or `-help` will display a basic message and end the application. Additional optional parameters can be used to set the same parameters as in the console window that opens with the app. Every parameter expects an argument. `-z` sets the timezone, `-lo` sets the longitude, `-la` sets the latitude, `-a` sets the altitude, `-s` sets the time speed. Up to two triplets can be added to the front to set the date and time. The first triplet is considered the date, and the second triplet is considered the time instant within the day. An example that uses all parameters can be:

```
2022 12 0 0 -la 14.4181 -z 2 -a 12 -s 72 -lo 50.0767
```

Chapter 6

Results

Figure 6.1: Comparison of the sky in the app to real photos. Photos have been taken from Jooinn[1].

Many features rendered by the app have been shown throughout the thesis. The correctness of astronomical position calculations and the appearance of the Moon were cross-referenced with online tools[2][3][4]. These tools allow seeing the apparent positions of different objects in the sky and what the Moon's appearance is for a given time instant. They can also show the positions

[1] Free photo website Jooinn.com: https://jooinn.com/clear-sky-2.html [cit. 20-05-2022]

[2] Online Planetarium by The Sky LIVE: https://theskylive.com/planetarium [cit. 17-05-2022]

[3] Moon Phase and Libration, 2022 by NASA: https://svs.gsfc.nasa.gov/4955 [cit. 17-05-2022]

[4] A calculator on PlanetCalc made by the user Timur: https://planetcalc.com/318/

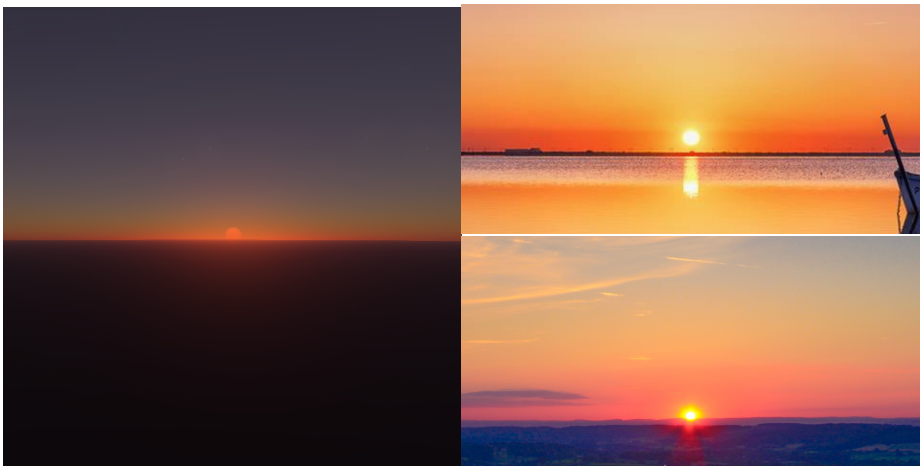


Figure 6.2: Comparison of the sky during sunset, left is in the app, on the right are two real photos. Photos have been taken from Wallpaper Flare (top)[5] and Jooinn (bottom)[6].

in the equatorial spherical coordinates, which was useful as another way of validation.

In Figure 6.1, you can see the sky rendered by the app on the left and two example photos on the right. As you can see, one of them seems to match pretty well, whereas the other is a much deeper shade of blue. The daytime sky is very bright, and some measures to handle the brightness have to be taken when taking a photo of it. The difference could be just the camera's settings when taking the picture, but other factors, like humidity, temperature, and others, could play a part. Personally, though, I prefer the top image, as it reflects closer the color of the sky we see.

Figure 6.2 shows a comparison of the twilight sky to actual photos of a sunset (the sun appears partially under the horizon in the app because it is enlarged five times). Notice the difference in intensity, where the real photos appear much brighter than the render from the app. Again, the author probably tweaked exposure and sensitivity to make them appear better. The app does not react to the change of brightness and renders everything in low dynamic range. A more responsive model could be used to better approximate the brightness management both in cameras and in the human eye.

You can also see that the colors in the top photo in the up direction from

[5]Site collecting wallpapers Wallpaper Flare: https://www.wallpaperflare.com/panoramic-photography-of-ocean-sunset-with-clear-sky-wallpaper-16519 [cit. 20-05-2022]

[6]Free photo website Jooinn.com: https://jooinn.com/sunset-under-blue-sky.html [cit. 20-05-2022]

Figure 6.3: Comparison of the Moon in the app to a real photo. The photo has been taken on November 29th, 2017 at 19:20:21 in Ann Arbor, Michigan. Photo has been taken from Pentax Forums[7].

the horizon go from red through orange to purple, whereas the app seems to include a green/yellowish strip. However, the bottom photo resembles the app's result slightly better color-wise. Atmospheric properties may play a role again, but the bottom photo still has a much softer yellow tint than the app. Multiscattering does not seem to be the cause, so a wrong model is one possibility since other papers in the field also seem to have it.

In Figure 6.3, you can see the Moon as rendered by the app on the left and as captured by a photo on the right, both at the same time and roughly in the same location. You can see that many of the same features are there: phase angle, rotation, and shadows. The most significant difference is the color, but that is primarily due to the tonemapping, simulating viewing by the human eye, whereas the camera does not attempt to emulate it. Another difference is the quality of shadows. Notice that the shadows around the poles look much cleaner in the photo compared to the render. This issue with the app is perhaps caused by the normal approach to shadowing or the albedo texture that has static shadows around the poles. The shadows also seem to span a wider area around the terminator in the photo on the right, the cause of which is probably again the shadows being rendered by normals and not considering accurate shadowing by the terrain.

The stars are much more difficult to compare since their brightness is small, they appear only as small dots, and the black background is not well suited for their visualization. The rendered images have had to be lightened a lot

[7]User dave2k on the amateur photography forum Pentax Forums: https://www.pentaxforums.com/gallery/photo-moon-shot-54525/ [cit. 20-05-2022]

Figure 6.4: Comparison of the stars in the app to a real photo. The photo has been taken on April 8th, 2019 at around 20:21:36 in Lake Angelus, South Island, New Zealand. Photo has been taken from Pentax Forums[8].

throughout the thesis specifically, so they are better visible on paper. Digital versions of the images are not impacted as much.

That said, Figure 6.4 shows a comparison of two images of the starry sky. Yet again, the night scene in the app appears much darker than its real counterpart. Notice how the sky in the real photo also has a sort of "glow" to it. As the photo's exposure has been set to 30 seconds, each pixel has captured a lot of light, even from areas where we cannot see any star. For the same reason, the stars we can see are much more distinguishable, and many more are visible than what the human eye can see. Therefore, the photo appears full of stars, whereas the render only has several well-visible stars.

Another difference to note is the colors. There is the obvious effect of tonemapping that shifts the colors closer to blue, again, since the camera does not attempt to replicate the human vision. However, the stars themselves appear almost colorless in the photo. This is probably a consequence of the used camera since we can distinctly see stars of various colors when we look at the night sky. The final color discrepancy is the transmittance effect near the horizon. You can see in the rendered image that the Milky Way trails off to red and then into black, whereas the photo does not seem to show this effect at all. This could be due to the exposure of the camera, atmospheric properties, or simply that the actual effect is not as strong as the app's.

[8]User Focusrite on the amateur photography forum Pentax Forums: https://www.pentaxforums.com/gallery/photo-lake-angelus-milky-way-59743/ [cit. 20-05-2022]

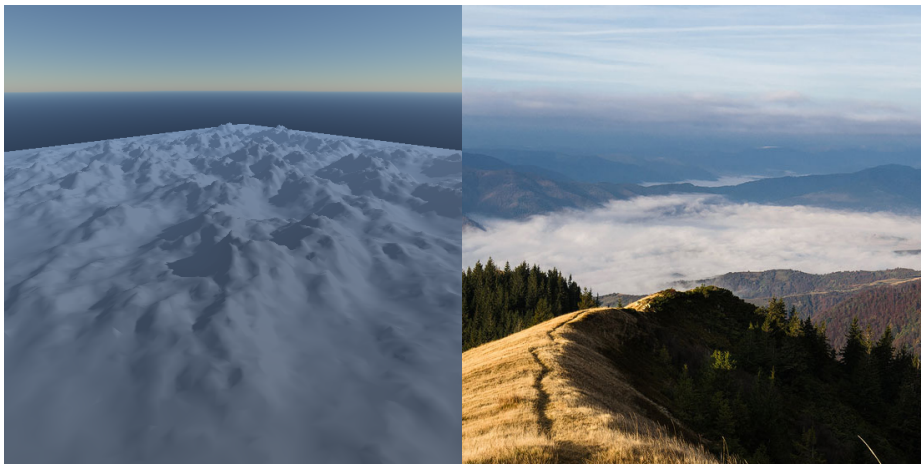


Figure 6.5: Comparison of the terrain as rendered in the app to that in a real photo. the photo has been taken from Anton Gorlin's personal gallery[9].

Finally, Figure 6.5 shows a comparison of the atmospheric effect on the terrain. You can see that the blue shift is present in both, but the light in the real photo seems to lose much more intensity over traveled distance. There can be various reasons, from simply different atmospheric conditions, most notably humidity and altitude, to more severe like a wrong simulation model. The amount of blue light reaching the camera also seems to be much smaller in the photo, which could be caused by overcast, and thus, much of the path would be in the shade.

Not all features can be seen at once. A simple example can be tonemapping and the Sun. However, there are also some consequences of the chosen approach that printed images cannot do enough justice to. These include the paths of the objects in the sky, Moon librations, and star scintillations. The fluidity of the transition is also difficult to capture in text or images.

Table 6.1 shows the time required to create each texture. The measurements were done on an Nvidia RTX 3060. As you can see, the times are not insignificant, and, surprisingly, the multiscattering LUT with 64 raymarches per texel was computed comparatively quickly. All LUT-creation times are much higher than what Hillaire has achieved with his method[Hil20]. The reason is not entirely clear, but it could be optimization issues or a different measurement method. The app uses more samples in the raymarching, but the time difference is much greater than one would expect purely from this change.

[9]Anton Gorlin's landscape gallery: https://antongorlin.com/galleries/mountain-landscape/ [cit. 20-05-2022]

Texture	Resolution	Time (ms)
Shadowmap	4096×4096	0.57
Transmittance LUT	256×64	0.13
Multicattering LUT	32×32	0.15
Sky-view LUT	200×100	0.29
Aerial-view LUT	$32 \times 32 \times 32$	0.32
1 frame	1920×1080	4.97

Table 6.1: Creation time for precomputed textures. Rendering time for one frame has been added as a frame of reference. The resolutions have been added for context. Keep in mind that two shadow maps have to be rendered, one for the Sun and one for the Moon.

The table further includes the time to render a shadowmap. The time shown is only for one texture, so the two shadowmaps account for about one ninth of the rendering time. The main reason for this is their sheer size. Since they have to encompass the entire scene, their resolution has to fit enough detail not to make the shadows too blocky. However, higher resolutions mean more rendering time and more storage space.

The resolutions were chosen to align with Hillaire's work. Each texture has four channels, each in the 32-bit float format. The only exception is the shadowmaps, which only have one 32-bit depth component. The resolution was brought as high as the app allowed without any significant impact on performance. As it is, each shadowmap takes 64MB of space in VRAM. Even larger sizes were tested, up to $32,768^2$ (4GB per texture), but though the shadows looked much better when successful, the app would often not have enough memory to initialize correctly. The size of the window was chosen to be the common display resolution of modern PC monitors.

Chapter 7

Conclusion

An app using several techniques to render various different features of both the night and day skies has been implemented and described. These are all combined into a single scene and merged to create a believable simulation of the day, night, and transition between them. This simulation goes beyond what an ordinary observer might consciously notice or even know of. The rendering process works in real time and is therefore suitable for a variety of applications like games or education.

Figure 7.1: The Moon close to the center of the Milky Way. Rendered using the app. Antares and the tail of Scorpius can be seen to the right. The image has been heavily lightened to be easier to see.

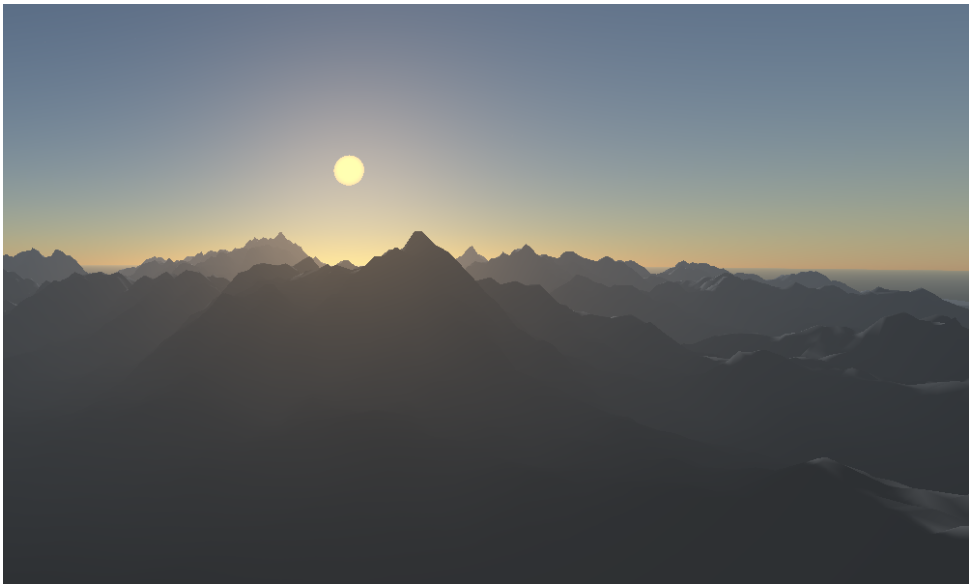


Figure 7.2: Terrain in the foreground with the Sun rising in the morning in the background.

7.1 Future work

The highest potential for improvement is the unification of the low dynamic range used when rendering the Sun, terrain, and stars and the high dynamic range used for calculating the intensities of the sky and Moon. Tonemapping is probably the best example since only a small interval of input intensity is used for the function. With the wide range of visible intensities, a better-looking transition between day and night could be achieved purely with a better human eye perception model.

The star glare appears as concentric rings around the stars, which is easily noticeable since minimal values are used. A good example is Figure 5.7a. Only so many colors can be represented, so the point spread function values are clamped to the nearest. A way to break this perception of regular circles would be to introduce dithering, but more research can be done to make the stars more believable.

A good addition would be the simulation of clouds, as they can fundamentally change the impression of the sky. A well-implemented cloud model can really add to the immersion of a virtual environment. Having the ability to change atmospheric properties locally can also be a desired addition, which would simulate the difference we see when moving between a city center and the countryside.

Appendix A

Bibliography

[BN08] Eric Bruneton and Fabrice Neyret, *Precomputed Atmospheric Scattering*, Computer Graphics Forum (2008).

[CH] Color-Hex.com, *Sun spots color palette*, https://www.color-hex.com/color-palette/9190, [cit. 2022-05-05].

[God] Space Flight Center Goddard, *Lro-l-lola-4-gdr*, https://pds-geosciences.wustl.edu/missions/lro/lola.htm, [cit. 2022-05-05].

[GQY+01] P. R. Goode, J. Qiu, V. Yurchyshyn, J. Hickey, M.-C. Chu, E. Kolbe, C. T. Brown, and S. E. Koonin, *Earthshine observations of the earth's reflectance*, Geophysical Research Letters **28** (2001), no. 9.

[GSAM04] Diego Gutiérrez, Francisco Serón, O Anson, and Adolfo Muñoz, *Chasing the green flash: A global illumination solution for inhomogeneous media*, Spring Conference on Computer Graphics, SCCG 2004 - Conference Proceedings (2004).

[Hap63] Bruce W. Hapke, *A theoretical photometric function for the lunar surface*, Journal of Geophysical Research (1896-1977) **68** (1963), no. 15.

[Hap66] Bruce W. Hapke, *An improved theoretical lunar photometric function.*, The Astronomical Journal **71** (1966), 333.

[Hil20] Sébastien Hillaire, *A scalable and production ready sky and atmosphere rendering technique*, Computer Graphics Forum **39** (2020), no. 4, 13–22.

[HM05] Jörg Haber and Marcus Magnor, *Realistic solar disc rendering*, In WSCG'2005 Full Papers Conference Proceedings. 79–86, 2005, pp. 79–86.

[HMS05] Jörg Haber, Marcus Magnor, and Hans-Peter Seidel, *Physically-based simulation of twilight phenomena*, ACM Trans. Graph. **24** (2005), no. 4, 1353–1373.

[Hun05] R.W.G. Hunt, *Subtractive methods in colour photography*, The Wiley-IS&T Series in Imaging Science and Technology, 2005.

[HW91] D. Hoffleit and W.H. Warren, Jr., *Yale bright star catalogue, 5th revised*, http://tdc-www.harvard.edu/catalogs/bsc5.html (1991), [cit. 2022-05-05].

[JDD+01] Henrik Wann Jensen, Frédo Durand, Julie Dorsey, Michael M. Stark, Peter Shirley, and Simon Premože, *A physically-based night sky model*, Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), SIGGRAPH '01, Association for Computing Machinery, 2001, p. 399–408.

[JPS+00] Henrik Wann Jensen, Simon Premoze, Peter Shirley, William B. Thompson, James A. Ferwerda, and Michael M. Stark, *Night rendering*, 2000.

[KP04] Saad M. Khan and Sumanta N. Pattanaik, *Modelling blue shift in moonlit scenes using rod cone interaction*, Journal of Vision **4** (2004), 316–316.

[LG11] Orion Sky Lawlor and Joe Genetti, *Interactive volume rendering aurora on the gpu*.

[MED12] Daniel Müller, Juri Engel, and Jürgen Döllner, *Single-pass rendering of day and night sky phenomena*, VMV, 2012.

[Mee91] J. Meeus, *Astronomical algorithms*, Willmann-Bell, 1991.

[Mot] Celestia Motherload, *Lro-l-lola-4-gdr*, http://www.celestiamotherlode.net/catalog/moon.html, [cit. 2022-05-05].

[SIS+95] Greg Spencer, Taligent Inc, Peter Shirley, Kurt Zimmerman, and Donald Greenberg, *Physically-based glare effects for digital images*, Computer Graphics **29** (1995).

[TW14] Du Tao and Lu Wenlong, *Rendering aurora*.

[Wri20] Ernie Wright, *Deep star maps*, https://svs.gsfc.nasa.gov/4851 (2020), [cit. 2022-05-05].

[WTB+10] Julia Weratschnig, D Taylor, S Bell, J Hilton, and A Sinclair, *Computation of the quantities describing lunar librations in the astronomical almanac*.

[YsBhBsDi06] Kim Young-sun, Cho Bong-hwan, Kang Bong-soon, and Hong Doo-il, *Color temperature conversion system and method using the same*, 2006.

Appendix B

Astronomical calculations

In this appendix, various formulas will be presented for time and position calculations, coordinate conversions, and color space transformations. Many of the astronomical formulas were taken from Meeus[Mee91], and some from Jensen et al.[JDD+01]. The color formulas are mostly standard, though some parts were taken from Hunt's book about color calculations[Hun05].

All angles are in radians, and distances are generally in kilometers unless stated otherwise. All values were also truncated to up to four decimal places. For the exact values, see the original literature or the source code for the app. All vectors are column vectors to align with OpenGL.

The space the astronomical formulas are in **is left-handed**, so after the directions are calculated, they have to be **converted to the right-handed** OpenGL coordinates by negating the x coordinate.

Rotational matrices R_x, R_y, R_z are the standard counterclockwise rotations in left-handed rectangular space. To avoid confusion, the rotation matrix R_x used for the astronomy calculations is as follows (others should be easy to determine):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}.$$

B.1 Time

Instead of using the date as several values, Julian dates are used in astronomy, representing the date and time instant. For the date and time in format $Y/M/D/h/m/s$, the Julian date JD is given by:

$$\begin{aligned} JD =& 1720996.5 - \left\lfloor \frac{Y'}{100} \right\rfloor + \left\lfloor \frac{Y'}{400} \right\rfloor + \lfloor 365.25Y' \rfloor \\ &+ \lfloor 30.6001(M'+1) \rfloor + D + (h + (m + s/60)/60)/24, \end{aligned} \tag{B.1}$$

where Y' and M' are the adjusted year and month. If $M \leq 2$, then $Y' = Y - 1$ and $M' = M + 12$, otherwise $Y' = Y$ and $M' = M$. The time is GMT, so zone correction is therefore necessary before the calculation. This formula also does not account for the slowing of the Earth's rotation. The time difference denoted ΔT is set to 72 seconds, the value for 2022, estimated from the extrapolation of historical data. This value should be added to the date value.

For the purpose of astronomical algorithms, a specific time value T is used. T is the number of centuries since a given equinox. In this thesis, the chosen equinox is J2000.0, and T is given by:

$$T = (JD - 2451545.0)/36525. \tag{B.2}$$

Local mean sidereal time $LMST$ expresses the Earth's rotation relative to the stars. However, since it is bound to the rotation, the ΔT correction should not be used when determining $LMST$. The value is then calculated as:

$$LMST = 1.7534 + 628.3320T \tag{B.3}$$

This equation only holds for values of T at midnight Universal Time. To get $LMST$ for any instant time, convert that instant from time to angle, multiply it by 1.0027, and add it to the result of Equation B.3.

Please note that these formulas offer many optimization opportunities and may require special treatment to avoid temporal incoherence due to big number calculations. A good example is the constant 1720996.5 in JD and subtracting 2451545. As JD is only ever used for T, those two constants can simply be combined, etc. Another thing to notice is that $LMST$ is an angle, as will be the case with many values in these calculations. For best precision, these angles should be wrapped to avoid dealing with large numbers and losing precision.

B.2 Coordinate conversion

The position calculations are done in spherical coordinates, but coordinate conversion is most easily done in rectangular coordinates with transformation matrices. To transform from spherical to rectangular, we use:

$$\begin{aligned} x &= r\cos(\lambda)\cos(\beta) \\ y &= r\sin(\beta) \\ z &= r\sin(\lambda)\cos(\beta), \end{aligned} \tag{B.4}$$

where λ and β represent longitude and latitude respectively. Notice that y has been chosen as the vertical axis to align more with the OpenGL standard. It should also be mentioned that the $-x$ axis has been chosen as the geographical north direction and $+z$ as the geographical west.

For the transformations, we only need one more value calculated. The Earth's axis of rotation is not perpendicular to the ecliptic, the Earth's orbital plane. This deviation is called obliquity and amounts to about 23°. However, a more precise calculation of obliquity ϵ is as follows:

$$\epsilon = 0.4093 - 2.27 \cdot 10^{-4} T. \tag{B.5}$$

With this, we can finally represent the transformation from the ecliptic space to local horizon coordinates with the matrix

$$R_{ec\to ho} = R_{eq\to ho} \cdot R_x(\epsilon) = R_z(-\beta_o + \frac{\pi}{2}) \cdot R_y(-LMST - \lambda_o + \pi) \cdot R_x(\epsilon), \tag{B.6}$$

where $R_{eq\to ho}$ is the rotation from the equatorial plane to the local horizon, which will be helpful for transforming the stars. λ_o and β_o are the geographical longitude and latitude of the observer.

B.3 Sun calculations

The Sun's position is a simple calculation because, firstly, there are not many factors influencing the Earth's orbit. Secondly, the ecliptic latitude β is zero from the definition of the ecliptic plane. Any minor deviations that occur in β are considered unobservably small. First, compute the Sun's mean anomaly

$$M = 6.2401 + 628.3020T + 2.721 \cdot 10^{-6} T^2. \tag{B.7}$$

From that, the geocentric ecliptical longitude λ of the Sun follows

$$\begin{aligned}\lambda = \quad & 4.8951 && + 628.332T && + 5.2918 \cdot 10^{-6}T^2 && \\ & +(3.3416 \cdot 10^{-2} && - 8.40723 \cdot 10^{-5}T && - 2.4435 \cdot 10^{-7}T^2) && \sin(M) \\ & +(3.4894 \cdot 10^{-4} && - 1.7628 \cdot 10^{-6}T) && && \sin(2M) \\ & +5.0615 \cdot 10^{-6} && && && \sin(3M)\end{aligned} \tag{B.8}$$

To get the direction to the Sun in horizontal coordinates, first convert to rectangular coordinates with Equation B.4, then rotate by matrix $R_{ec \to ho}$ in Equation B.6.

B.4 Moon calculations

Unlike the Sun's, the Moon's ecliptic latitude is non-trivial as the Moon oscillates up and down. Additionally, the Moon's distance from the Earth is not negligible because it can move the Moon's disk by several diameters.

To get the position of the Moon, we first need to calculate the mean longitude of the Moon L', mean elongation of the Moon D, Sun's mean anomaly M, Moon's mean anomaly M', Moon's argument of latitude F, and the eccentricity of the Earth's orbit E:

$$\begin{aligned} L' =& 3.8103 && + 8399.7091T && - 2.3157 \cdot 10^{-5}T^2 \\ M' =& 2.3556 && + 8328.6914T && + 1.5703 \cdot 10^{-4}T^2 \\ M =& 6.2401 && + 628.302T && + 2.6808 \cdot 10^{-6}T^2 \\ D =& 5.1985 && + 7771.3771T && - 2.8449 \cdot 10^{-5}T^2 \\ F =& 1.6279 && + 8433.4662T && - 5.9392 \cdot 10^{-5}T^2 \\ E =& 1 && - 2.516 \cdot 10^{3}T && - 7.4 \cdot 10^{-6}T^2. \end{aligned} \tag{B.9}$$

With these, the position can be calculated as

$$
\begin{aligned}
\lambda = \quad & L' \\
+ \quad & 1.0976 \cdot 10^{-1} \sin(M') \\
+ \quad & 2.2236 \cdot 10^{-2} \sin(2D - M') \\
+ \quad & 1.1490 \cdot 10^{-2} \sin(2D) \\
+ \quad & 3.7283 \cdot 10^{-3} \sin(2M') \\
-E \quad & 3.2309 \cdot 10^{-3} \sin(M) \\
- \quad & 1.9955 \cdot 10^{-3} \sin(2F) \\
+ \quad & 1.0261 \cdot 10^{-3} \sin(2D - 2M') \\
+E \quad & 9.9599 \cdot 10^{-4} \sin(2D - M - M') \\
+ \quad & 9.3064 \cdot 10^{-4} \sin(2D + M') \\
+E \quad & 7.9863 \cdot 10^{-4} \sin(2D - M) \\
+E \quad & 7.1424 \cdot 10^{-4} \sin(M - M') \\
- \quad & 6.0598 \cdot 10^{-4} \sin(D) \\
-E \quad & 5.3028 \cdot 10^{-4} \sin(M + M')
\end{aligned}
$$

$$
\begin{aligned}
\Delta = \quad & 60.3638 \\
- \quad & 3.2777 \cdot 10^{-0} \cos(M') \\
- \quad & 5.7997 \cdot 10^{-1} \cos(2D - M') \\
- \quad & 4.6346 \cdot 10^{-1} \cos(2D) \\
- \quad & 8.9358 \cdot 10^{-2} \cos(2M') \\
+ \quad & 3.8595 \cdot 10^{-2} \cos(2D - 2M') \\
-E \quad & 3.2077 \cdot 10^{-2} \cos(2D - M)
\end{aligned}
$$

$$
\begin{aligned}
\beta = & 8.9503 \cdot 10^{-2} \sin(F) \\
& +4.8974 \cdot 10^{-3} \sin(M' + F) \\
& +4.8467 \cdot 10^{-3} \sin(M' - F) \\
& +3.0236 \cdot 10^{-3} \sin(2D - F) \\
& +9.6714 \cdot 10^{-4} \sin(2D - M' + F) \\
& +8.0758 \cdot 10^{-4} \sin(2D - M' - F) \\
& +5.6851 \cdot 10^{-4} \sin(2D + F),
\end{aligned}
\tag{B.10}
$$

where Δ is the Earth-Moon distance in Earth radii. After the Moon's position is transformed to local horizon coordinates, the vector $(0, 1, 0)$ is subtracted to move the observer from the Earth's center to the surface. Then the angular size θ is calculated as

$$
\theta = \arcsin\left(\frac{6375.14}{1737.5 \cdot l}\right), \tag{B.11}
$$

where l is the distance to the offset position of the Moon as above.

Lastly, the rotation of the Moon is given by the matrix

$$
R_M = R_{ec \to ho} R_y(F + \pi) R_x(0.02692) R_y(L' - F). \tag{B.12}
$$

B.5 Star calculations

The star positions are given in spherical equatorial coordinates, so they have to be converted to rectangular, and then every frame rotated with the $R_{eq \to ho}$ matrix. The temperature of a star is estimated by

$$T_{eff} = \frac{7000}{B - V + 0.56}, \tag{B.13}$$

where $B - V$ is given for every star from the catalogue. This temperature is clamped to $[1650, 25000]$. Next, the xy coordinates in the chromaticity diagram are found by

$$x = \begin{cases} \frac{-266123900}{T_{eff}^3} - \frac{234358.9}{T_{eff}^2} + \frac{877.6956}{T_{eff}} + 0.17991 & T_{eff} < 4000 \\ \frac{-3025846900}{T_{eff}^3} + \frac{2107037.9}{T_{eff}^2} + \frac{222.6347}{T_{eff}} + 0.24039 & T_{eff} >= 4000, \end{cases} \tag{B.14}$$

and

$$\begin{aligned} &\text{If } T_{eff} < 2222: \\ &\quad y = -1.1063814x^3 - 1.3481102x^2 + 2.18555832x - 0.20219683 \\ &\text{If } 2222 <= T_{eff} < 4000: \\ &\quad y = -0.9549476x^3 - 1.37418593x^2 + 2.09137015x - 0.16748867 \\ &\text{If } 4000 <= T_{eff}: \\ &\quad y = 3.081758x^3 - 5.8733867x^2 + 3.75112997x - 0.37001483. \end{aligned} \tag{B.15}$$

These are then converted to rgb (described below), supposing the luminance $Y = 1$.

B.6 Color transformations

To get xyY coordinates from XYZ, the following identities are used:

$$\begin{aligned} x &= \frac{X}{X + Y + Z} \\ y &= \frac{Y}{X + Y + Z} \\ Y &= Y, \end{aligned} \tag{B.16}$$

and the inverse is

$$\begin{aligned} X &= \frac{xY}{y} \\ Y &= Y \\ Z &= \frac{(1 - x - y)Y}{y}. \end{aligned} \tag{B.17}$$

Converting XYZ to rgb values is more complicated since the conversion depends on the target space of the device. The PC industry has defined a color space for PC monitors (sRGB), which was chosen as the target space in the app. The matrix for conversion to sRGB is

$$\begin{bmatrix} 3.2404542 & -1.5371385 & -0.4985314 \\ -0.9692660 & 1.8760108 & 0.0415560 \\ 0.0556434 & -0.2040259 & 1.0572252 \end{bmatrix}, \tag{B.18}$$

and for completeness, the inverse is

$$\begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix}. \tag{B.19}$$

Appendix C

Contents of the attachment

- **imgs.zip** 6 example images from the app. A good illustration of the app's capabilities.
- **exe.src** Compiled app in an executable state. The folder `example bat files` contains several batch files to launch the app in different configurations.
- **src.zip** Microsoft Visual Studio 19 project ready to be built and run. Both `Debug` and `Release` configurations have the dynamically linked libraries prepared in the appropriate folders.