DOI: 10.1111/cgf.70017
EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Physically Based Real-Time Rendering of Eclipses

S. Schneegans[1], J. Gilg[2], V. Ahlers[3], G. Zachmann[1], and A. Gerndt[1,2]

[1]University of Bremen, Germany
[2]German Aerospace Center (DLR)
[3]Hanover University of Applied Sciences and Arts, Germany

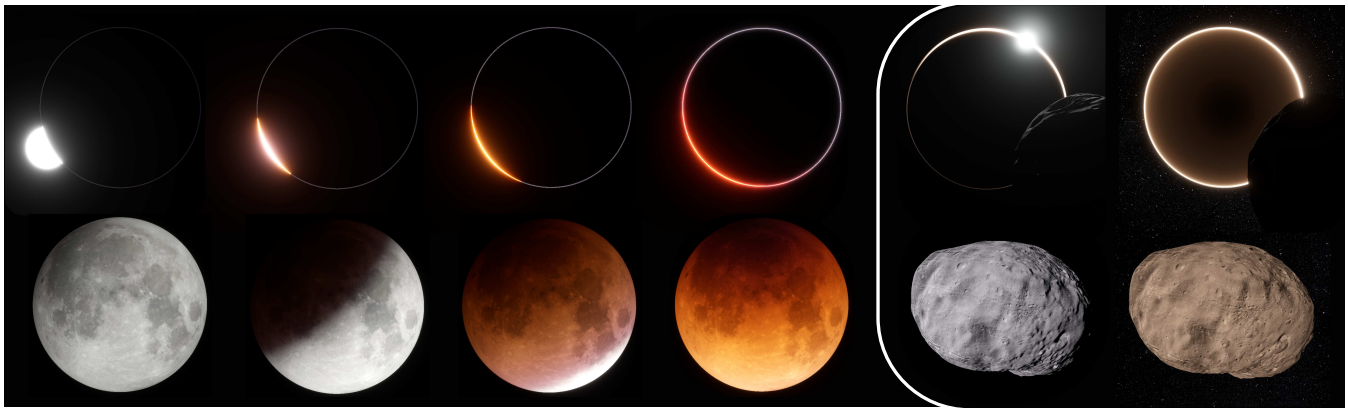


Figure 1: *The first four columns show renderings of the lunar eclipse on 2015-09-28 taken at about 30-minute intervals starting at 1:00 AM UTC. The upper row shows how Earth would look like for an observer on the lunar north pole. Our technique is not limited to the Earth-Moon system, but can also simulate light scattering and refraction in other atmospheres. The last images show a similar scenario for the Martian moon Phobos in the penumbra (last but one column) and in the umbra (last column).*

Abstract

We present a novel approach for simulating eclipses, incorporating effects of light scattering and refraction in the occluder's atmosphere. Our approach not only simulates the eclipse shadow, but also allows for watching the Sun being eclipsed by the occluder. The latter is a spectacular sight which has never been seen by human eyes: For an observer on the lunar surface, the atmosphere around Earth turns into a glowing red ring as sunlight is refracted around the planet. To simulate this, we add three key contributions: First, we extend the Bruneton atmosphere model to simulate refraction. This allows light rays to be bent into the shadow cone. Refraction also adds realism to the atmosphere as it deforms and displaces the Sun during sunrise and sunset. Second, we show how to precompute the eclipse shadow using this extended atmosphere model. Third, we show how to efficiently visualize the glowing atmosphere ring around the occluder. Our approach produces visually accurate results suited for scientific visualizations, science communication, and video games. It is not limited to the Earth-Moon system, but can also be used to simulate the shadow of Mars and potentially other bodies. We demonstrate the physical soundness of our approach by comparing the results to reference data. Because no data is available for eclipses beyond the Earth-Moon system, we predict how an eclipse on a Martian moon will look like. Our implementation is available under the terms of the MIT license.

CCS Concepts

• ***Computing methodologies*** → *Real-time simulation;*

1. Introduction

Lunar eclipses have been awe-inspiring events since the dawn of humankind. There have been attempts to predict the occurrence and appearance of lunar eclipses for thousands of years. Today, we can predict them with high accuracy using computer programs. There are even approximate approaches working in a matter of milliseconds which can be used in real-time applications like scientific visualizations or video games. However, existing real-time rendering approaches are limited to the Earth-Moon system, and the shadow

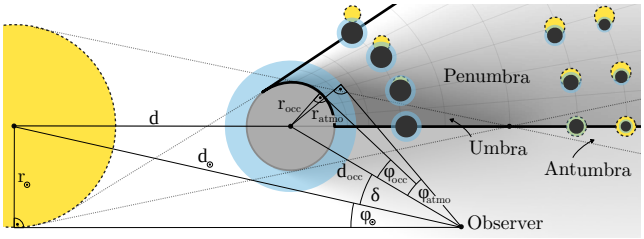


Figure 2: *This figure shows the lengths and angles used in this paper. It extends the nomenclature introduced in Figure 2 of [SGAG22] with the radius r_{atmo}, the angular radius φ_{atmo}, and the average distance between Sun and occluder d. The small glyphs provide an approximate visualization of how Sun and occluder may look from the respective position in the shadow cone. For a more detailed visualization, refer to* Figure 4. *The thick dark line encloses the area for which we will precompute the eclipse shadow.*

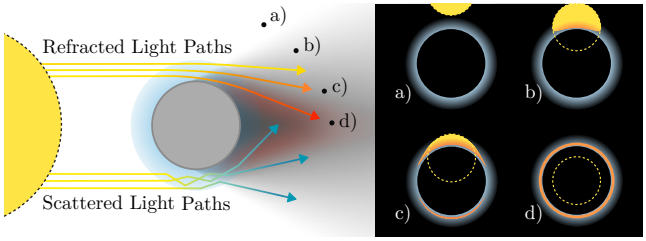


Figure 4: *Due to scattering and refraction, light can travel into the otherwise dark umbra region. In Earth's atmosphere, blue light is scattered more than red light. Hence, the atmosphere around Earth appears bluish, while the refracted image of the Sun turns orange or red when it sets behind the planet. The images on the right show how the Sun may appear from the positions marked in the left image. At some point, a second image of the Sun appears at the bottom (c), and finally it becomes a ring around the planet (d).*

can only be cast onto the lunar surface and thus does not allow visualizing the light conditions on a spacecraft entering the shadow of Earth. Also, existing real-time approaches ignore the contribution of scattered light. As we will see, this can be important on Earth in times of high aerosol loads (e.g. after a volcanic eruption) and is crucial for simulating eclipses on other celestial bodies, such as Mars. Last but not least, we are not aware of any implementation which considers the appearance of the atmosphere around the planet during an eclipse. Only very few images of this exist, but it promises to be a spectacular sight (see *Figure 3*).

Our approach can be used for education, science communication, or scientific visualization. In fact, we have been developing it for yet another purpose: space-mission planning. In the VaMEx3-RGE project, technology is developed which will enable autonomous exploration of vast areas on Mars by a heterogeneous robotic swarm. A key aspect of this project is an immersive mission-control tool which allows planning and observation of the mission under realistic lighting conditions. This tool can potentially be used for future missions to other celestial bodies like our Moon or the Martian moons Phobos and Deimos where eclipses are frequent events.

In this paper, we present a unified approach for rendering eclipses which is not limited to the Earth-Moon system, incorporates scattered light, supports views of the eclipsed Sun from within the shadow cone, and runs in real-time.

Figure 3: *Only very few photographs of Earth during a lunar eclipse exist. The left image is a composite of three monochrome images taken by the Surveyor III lander on the Moon in 1967 [The67]. The right image has been taken by Apollo 12 astronauts on their way back home in 1969 (image credit: NASA/JSC).*

2. Physical Background and Related Work

First, we introduce the physical background and discuss related work in the field of atmosphere and eclipse rendering.

2.1. Geometry of an Eclipse

An eclipse happens if the Sun and at least two other celestial bodies are almost aligned in a straight line. The first body is the occluder which blocks sunlight from reaching the second body, the eclipsed object. The shadow volume of the occluder can be divided into three regions: umbra, penumbra, and antumbra (*Figure 2*). Only if the occluder has an atmosphere, sunlight can reach the otherwise completely dark umbra due to refraction and scattering (*Figure 4*).

2.2. Modelling the Sun

We follow the related work and model the Sun as spherical light source with a radius $r_\odot = 696\,342\,\text{km}$ [SGAG22]. The Sun does not emit light evenly into all directions, the luminance rather decreases towards the edge. This effect is called *limb darkening*. We have shown that this effect has a significant impact on the shadow gradient in the penumbra region [SGAG22]. Hence, we also incorporate limb darkening using the same formula as in [SGAG22]:

$$L(\varphi) = L(0)\left[1 - u\left(1 - \sqrt{\frac{\varphi_\odot^2 - \varphi^2}{\varphi_\odot^2}}\right)\right]. \tag{1}$$

Here, $L(0)$ refers to the luminance at the centre of the solar disc, $\varphi_\odot$ to the angular radius of the solar disc, and φ to the radial distance to the centre. We follow the authors above and use $u = 0.6$ as a constant limb darkening coefficient.

2.3. Scattering and Refraction in an Atmosphere

If the occluder has an atmosphere, light will be scattered and refracted before it reaches the eclipsed object.

Scattering: In Earth's atmosphere, light is scattered by small particles and molecules. Under normal weather conditions, molecular

Figure 5: *On Earth, refraction has a significant impact on the timing and appearance of sunsets and sunrises. The left image shows a sunset on Earth without refraction, the second image shows the same scene at the same simulation time but with refraction. The horizon appears a bit higher, the Sun seems to be significantly higher in the sky, and appears flattened. On Mars (right images), the effect is also present in theory, but the atmosphere is so thin that the effect is close to be invisible.*

scattering is the dominant factor [HM66]. Here, blue light is scattered more than red light, hence some blue light will reach the umbra due to scattering. Overall, the amount of light scattered into the umbra is very small compared to the light refracted into the umbra. However, it has been shown that scattering at larger aerosols can be significant for instance after volcanic eruptions [MP11]. On other planets, such as Mars, light scattering at aerosols is in fact the dominant factor for the appearance of the atmosphere [SMG*24].

Refraction: The refractive index of an atmosphere decreases with increasing altitude. This causes light rays to be bent towards the planet, allowing them to enter the umbra. As blue light is outscattered in Earth's atmosphere, a significant amount of red light will reach the umbra this way.

For an observer on Earth's surface, refraction results in two other effects, which are usually ignored in real-time rendering: first, the apparent position of the Sun is higher above the horizon than it actually is. This effect is most pronounced when the Sun is close to the horizon where it appears approximately 0.5° higher (which corresponds approximately to the apparent size of the Sun itself). Second, the Sun looks more like a vertically flattened ellipse than a circle when it is close to the horizon (*Figure 5*). However, due to atmospheric variability, the actual refraction varies a lot for different locations and times [SL90]. Even for a single location, the refraction has been reported to vary between 0.402° and 2.081° just due to seasonal and weather changes [SLPH03].

On Mars, where the atmosphere is much thinner, the effect of refraction is smaller by several magnitudes. We assumed the refractive index of the Martian atmosphere to be $1 + 3.38 \times 10^{-6}$. This is based on a mixture of 95.1 % carbon dioxide, 2.8 % nitrogen, and 2.1 % argon at a pressure of 610 Pa and a temperature of 215 K.

Ready-to-use algorithms for computing refracted rays due to a continuously changing refractive index have been presented in the literature [SGGC04, vdW08].

2.4. Simulating Eclipse Shadows

In a previous work, we have published an approach for simulating eclipse shadows in real-time *without considering atmospheric effects* [SGAG22]. We build upon this work and extend it to include the effects of refraction and scattering in the atmosphere of the occluder. Other authors have presented approaches for simulating eclipse shadows involving atmospheric effects, but they are either limited to the Earth-Moon system or are not suitable for real-time rendering. No real-time rendering algorithm has been presented for simulating the intricate appearance of the atmosphere around the occluder from the perspective of the eclipsed object.

Müller et al. use a one-dimensional lookup table for the shadow intensity [MED*12]. The lookup table is generated by matching a model to a series of photographs. A single dimension is sufficient because it is only valid for the Earth-Moon distance and due to the rotational symmetry of the eclipse shadow.

A more versatile approach has been proposed by Yapo et al. [YC09]. They trace light rays from the Sun to the Moon through Earth's atmosphere. Rays exiting the atmosphere are used to accumulate an illumination map for the Moon which is then used for real-time rendering. If the simulation time changes, the illumination map has to be recomputed. They do not incorporate outscattering by large particles in lower altitudes, however they introduce a wavelength-independent, optional "dust layer" between 15 km and 20 km to simulate different kinds of eclipses. Also, no in-scattered light is included in their approach. *Figure 4* can be used as an illustration of this limitation: Only light from the Sun is considered, not light coming from the atmosphere around the planet. The authors base this decision on the assumption that the effect of scattered light has a negligible effect on the appearance of lunar eclipses, which does not hold for other celestial bodies.

The prediction of lunar eclipse brightness and the derivation of atmospheric properties (such as for instance ozone content) from eclipse observations is a well-studied field called *Lunar Eclipse Theory* [HSV08,VG08,MP11]. We use results from this field to validate our approach. One important property of Earth's atmosphere is indeed the ozone layer which absorbs predominantly orange light in the visible spectrum [Gio18]. As such, the ozone layer has a significant impact on the appearance of lunar eclipses and has to be considered by the atmosphere model.

2.5. Real-Time Atmospheric Scattering

We do not only want to compute the illuminance at each point in the shadow cone, but also the luminance of the atmosphere around the occluder when observed from within the shadow. Hence, we need a real-time rendering model for atmospheres which simulates refraction and multiple scattering. The latter is especially important for the dusty atmosphere of Mars.

The widely used Hillaire model [Hil20] could potentially be extended to include refraction. However, it is not optimized for vantage points in outer space which are very important for our application. Instead, we extend the older Bruneton model [BN08] for which we recently published and extension which makes it work for Mars and potentially other celestial bodies [SMG*24].

The extended Bruneton model requires several precomputation steps. First, Mie Theory is used to compute tabulated phase functions and scattering coefficients of the individual atmospheric com-

ponents for several wavelengths. Next, multiple-scattering in the atmosphere is precomputed iteratively. This produces several lookup textures which are used at run-time to retrieve the atmosphere colour with just a few texture lookups. The lookup textures contain the transmittance to the end of the atmosphere for every possible ray direction and origin (a 2D texture), the amount of light scattered towards any observer (a 4D texture), and the total indirect illuminance at ground level (a 2D texture).

Conceptually, our approach for rendering the eclipse shadow is not limited to the Bruneton model. The same acceleration structures and algorithms could be used with other models, such as the Hillaire model, if they were extended to include refraction.

3. Our Approach

Here, we want to give a high-level overview before we dive deeper into the details in the following sections.

1. Real-time Atmospheric Refraction: We extend the Bruneton atmosphere model to include the effect of refraction. For this, we apply Snell's law to the light rays and store the deviation of the rays in an additional lookup texture (*Figure 6*). At run time, this deviation can be used for computing the refracted position of astronomical objects such as the Sun or stars.

2. Eclipse-Shadow Lookup-Texture (LuT): We use a two-dimensional texture which contains shadow values for the entire shadow cone as introduced in [SGAG22]. We render a view of the atmosphere for each pixel of the shadow LuT and store how much light reaches the corresponding position by accumulating the pixel values (*Figure 7*). At run-time, we can use this LuT to compute the illuminance at any point in the shadow with a single texture lookup.

3. Limb-Luminance LuT: When seen from outer space, the atmosphere becomes a very thin ring of pixels and the refracted image of the Sun may cover only a small fraction of each pixel. To avoid severe flickering, we precompute this ring for every possible position in the shadow and store it in a four-dimensional lookup texture.

3.1. Limitations

In reality, planets and moons are not perfectly spherical and are usually flattened towards the poles. For Earth, the polar flattening is about 0.3 %. This effectively means that the shadow of the Earth is not a perfect cone but a bit flattened as well. We argue that this error is very small when compared to the brightness difference an eclipse can have only due to changes in the atmospheric composition. As we will see in *Figure 12*, there can be about two magnitudes of brightness difference only due to weather conditions.

Also, it has been shown that regional variations in the atmosphere can lead to a non-symmetric shadow [GV08]. In *Figure 3*, Earth's limb seen from within the shadow appears rather patchy, probably due to high altitude clouds [WS14, p. 66]. However, for our purposes, the spherical symmetry seems acceptable.

Furthermore, our lookup textures are precomputed for a fixed Sun-occluder distance which in reality changes over time. However, our lookup-texture space parametrization is based on angular

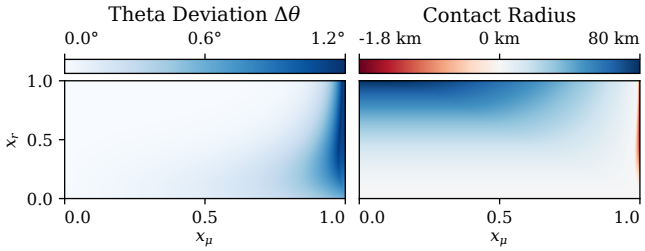


Figure 6: *Our extended atmospheric model precomputes two values which are stored in an additional lookup texture.* $\Delta\theta$ *is the angular deviation a ray encounters when travelling from its origin all the way to the end of the atmosphere (left). The contact radius is the minimum distance the ray had to the planet's surface (right). The texture space* $[x_\mu, x_r]$ *encodes all possible ray origins and directions inside the atmosphere (see* Equation 2*).*

radii which implicitly encodes the distance to Sun and to the occluder. Hence, the errors introduced by this simplification are very small. For very eccentric orbits, multiple lookup textures could be precomputed and blended based on the current distance to the Sun.

We only compute astronomical refraction and no terrestrial refraction. The former describes the apparent displacement of objects in the sky, the latter of objects on the ground. The effect of terrestrial refraction is much more subtle since the rays are much shorter. At the same time, it is much more complex to compute as it can lead to unocclusions where rays are bent behind objects on the ground. Also, for simulating eclipses, terrestrial refraction is not relevant.

Lastly, we chose a constant index of refraction for air. In reality, the index of refraction is wavelength-dependent, but the effect in the visible spectrum is very small. We used our implementation to compute the difference between the maximum refraction for red (750 nm, $n = 1.000275$) and blue light (400 nm, $n = 1.000283$) on Earth. We found it to be less than 0.02° for an observer at sea level.

4. Simulating Atmospheric Refraction

As a first step, we extended an existing atmospheric model to include the effect of refraction. Later, we will use this model to compute the luminance of the sky and the refracted image of the Sun for positions within the shadow cone of the occluder.

4.1. Extending the Bruneton Atmospheric Model

As a basis, we chose the generalized the Bruneton model which is applicable to extraterrestrial atmospheres [SMG*24]. In this model, atmospheric density, scattering, and absorption are integrated by sampling at regular intervals along straight rays. We replaced this with a more general integration scheme using the path length as the integration variable. Using the path length as integration variable does not suffer from numerical instabilities for rays which are either vertical or close to the horizon [vdW08]. For the actual refraction computation, we use ray marching with a fixed step length and the Runge-Kutta integration scheme. For the individual integration steps, we implemented the methods described in the last column of Table 1 of [vdW08] and Equation 7 of [SGGC04]. Both produce

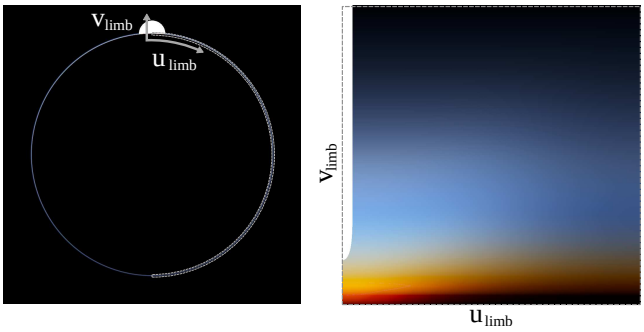


Figure 7: *The left image shows a true-to-scale view of Earth from a distance of about* 150 000 km*. The atmosphere appears as a very thin line around the planet. To compute the total illuminance at this point, we propose to render a rectangular view of half of this limb. The Y-axis extends from the lower end to the upper end of the atmosphere, while the X-axis is mapped to the circumference of the planet. The resulting limb view is shown on the right. The bright elongated spot on the left side is the refracted image of the Sun. Due to refraction, the image of the Sun does not touch the horizon even if half of the Sun is geometrically behind the horizon.*

about the same results, however, the method by Seron et al. is better suited for a double-precision GPU implementation as it requires no trigonometric functions. As suggested by other authors, a step size of 10 km produces good results [SGGC04, MP11]. We use a refractive index of $n = 1.000277$ at sea level which decreases vertically in the same way as the density of the atmosphere. Overall, refraction computation adds a significant computational overhead to the original model. Yet all this is done during preprocessing, and so it does not add to the runtime cost of the model.

With the above modifications, the precomputed transmittance, sky-luminance, and illuminance textures contain values for refracted light paths. This means, without much modifications to the runtime code, the atmosphere model now supports refraction for the sky colour. The only change required is to offset the color lookup position in the frame buffer according to the deviation of the ray. This is described in more detail in *Section 4.2*.

Yet, some simplifications are made by this approach. As we ignore terrestrial refraction, rays are not bent before they hit the ground. The transmittance and inscattering between the camera and the ground is computed by the difference of the precomputed values for both points. As the ground point would be a bit different for the refracted ray, this introduces a small error. Another simplification is that we use the angle between the ray direction and the geometric direction towards the Sun as input for the phase function. Due to refraction, both the direction of the ray and the direction towards the Sun change slightly along the ray. Yet also this effect is very small as rays are only bent by a small amount.

Most importantly, for the purpose of this paper, the key properties are the transmittance to the end of the atmosphere and the maximum ray deviation for refracted but unscattered rays. The transmittance gives us the color of the refracted image of the Sun, and the ray deviation describes the apparent position of the Sun in the sky. These properties are not affected by the simplifications made.

Precomputing a Ray's Deviation

To compute astronomical refraction, we store information about the angular deviation a ray encounters when passing through the atmosphere during the precomputation phase. Theoretically, we would need to precompute for every possible ray the position and direction where the refracted ray exits the atmosphere. However, in a spherically symmetric atmosphere, every ray start position and direction can be described by two parameters only: the origin's altitude $d_{occ} - r_{occ}$ and the ray-zenith angle θ. Furthermore, rays are always bent towards the planet's surface. This means, that one angle $\Delta\theta$ is sufficient to describe the ray's deviation. Lastly, we do not store the exit position of the ray, as we assume that any celestial body visible in the sky is extremely far away. Hence, the exit position does not matter, only the direction of the ray is important. Consequently, a single one-channel two-dimensional texture is sufficient to store the angular deviation $\Delta\theta$ of all possible rays.

It is convenient to store $\Delta\theta$ as a by-product of the transmittance computation. Computing the transmittance to the end of the atmosphere is the first preprocessing step in the Bruneton model. In this step, a ray is traced for each possible observer altitude / view-zenith angle. Once the ray leaves the atmosphere, we compute $\Delta\theta$ as the angular difference between the initial ray direction and the final ray direction. We store this as a 32 bit floating point value using the same texture space as the transmittance. The implementation by Schneegans et al. stores the transmittance in a two-dimensional texture using a parametrization which has been used by Eric Bruneton for his 2016 paper [Bru16]. The texture coordinates x_μ and x_r are based on the observer's distance to the planet's centre d_{occ} and the distance d_{exit} to the atmosphere exit point in ray direction. *Figure 6* shows the resulting textures for Earth's atmosphere.

$$\begin{aligned} x_\mu &= \frac{d_{exit} - d_{min}}{d_{max} - d_{min}} \quad , \quad x_r = \frac{\rho}{H} \quad \text{with} \\ \rho &= \sqrt{d_{occ}^2 - r_{occ}^2} \quad , \quad H = \sqrt{r_{atmo}^2 - r_{occ}^2} \quad , \quad \text{and} \\ d_{min} &= r_{atmo} - d_{occ} \quad , \quad d_{max} = \rho + H \end{aligned} \tag{2}$$

Figure 6 suggests that the mapping of x_μ could be improved for this use case. The "interesting" part of the texture is on the right-hand side, for values of x_μ close to 1. However, we rather chose to keep the parametrization as it is to avoid an additional precomputation step. The maximum computed deviation of about 0.6° for an observer at sea level on Earth is stored at $x_\mu = 1$ and $x_r = 0$. This value is in good agreement with the measured values in the literature [SL90, SLPH03]. We found a resolution of $512x256$ to be sufficient for the ray-deviation texture.

Precomputing the Impact Radius

Due to refraction, some rays are bent downwards and hit the planet. To avoid any discontinuities in the data, we continue tracing them assuming a constant index of refraction in the underground. However, we need to know which rays are blocked by the planet at runtime in order to not render astronomical objects which are below the horizon. For this, we store the minimum distance a ray had to the planet's surface in a second channel of the lookup texture. We call this quantity the *contact radius* of the ray. It is shown in the right graph of *Figure 6*. Rays which are blocked by the planet

are indicated by a negative distance. We will use this information not only for rendering the sky, but also for computing the eclipse shadow LuT where we can compare a ray's contact radius to the planet's average terrain height in order to attenuate rays passing very close to the surface.

4.2. Real-time Rendering of the Refracted Atmosphere

We have integrated the extended atmosphere model into CosmoScout VR, a virtual reality application for the exploration of our Solar System [SFGG25]. In this software, the atmosphere is drawn as a post-processing effect. Before our changes, the luminance of the atmosphere was combined with the frame-buffer colour via a simple alpha blending operation.

To account for astronomical refraction, we now need to shift the colour lookup position in the frame buffer according to $\Delta\theta$. For this, we first retrieve the depth value from the frame buffer to decide whether something is blocking the view. If so, we use the frame-buffer colour at that position, since we do not compute terrestrial refraction. If a ray through the pixel potentially leaves the atmosphere unblocked, we retrieve the ray's contact radius from the lookup texture (right chart of *Figure 6*). If the contact radius is negative, the refracted ray hits the planet. As it is non-trivial to compute the position where the ray hits the planet (it can also be behind the geometric horizon) we assume a black surface colour. This is usually not a problem, as this happens only for a few pixels above the geometric horizon where the atmosphere is very dense and the planet's surface would not be visible anyway. If the contact radius is positive, we retrieve the ray's angular deviation $\Delta\theta$ from the lookup texture (left chart of *Figure 6*) and use it to compute the exit ray by rotating the original ray towards the planet's centre by $\Delta\theta$. We sample depth and colour from the frame buffer at the vanishing point of the exit ray. If there is something in the depth buffer at that position, the ray was actually bent to a point behind the planet. To draw the refracted image of the Sun in this case, we compute the angle between the refracted ray and the direction to the Sun. If this angle is less than the angular radius of the Sun, we assume a background colour equal to the luminance of the Sun (incorporating the effect of limb darkening), else we assume a black background. For all other rays we can use the colour information from the frame buffer as a background colour to draw the atmosphere on top of it.

The first image of *Figure 5* shows a scene without refraction. The bottom part of the Sun is geometrically behind the horizon and therefore not visible in the frame buffer. To compute the second image (with refraction), the lower part of the Sun has been reconstructed by the technique described above. We only reconstruct the Sun for rays which are bent behind the horizon. Other celestial objects which potentially could become visible (like stars) will not be visible in this small region above the horizon.

5. Precomputation of Eclipse Shadows

Using the atmospheric model described in *Section 4*, we can now precompute the eclipse shadow by rendering the atmosphere from each position in the shadow.

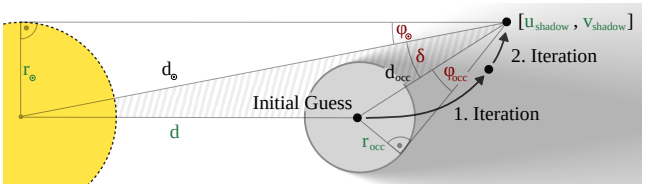


Figure 8: *This shows a schematic overview of the iterative search required to reconstruct $\varphi_\odot$, δ, and φ_{occ} (red) for some given u_{shadow}, v_{shadow}, d, $r_\odot$, and r_{occ} (green). d_{occ} and $d_\odot$ are iteratively updated to find the correct position in space. The law of cosines is used in the hatched triangle to compute $d_\odot$ based on the other two sides and δ (see* Equation 6*).*

5.1. Parametrization of the Shadow LuT

We use the shadow LuT parametrization from [SGAG22]: Based on the apparent radii of Sun and occluder,

$$\varphi_\odot = \arcsin(r_\odot/d_\odot) \quad \text{and} \quad \varphi_{occ} = \arcsin(r_{occ}/d_{occ}) \tag{3}$$

and the angular distance between the Sun and the occluder, δ, we compute the LuT coordinates u_{shadow} and v_{shadow} as follows:

$$u_{shadow} = \frac{1}{\varphi_{occ}/\varphi_\odot + 1} \qquad v_{shadow} = \frac{\delta}{\varphi_{occ} + \varphi_\odot}. \tag{4}$$

See *Figure 8* for an illustration of the involved angles and distances. A LuT using this parametrization is shown in *Figure 9*.

To compute the shadow intensity for each $[u_{shadow}, v_{shadow}]$ position, we also need the reverse mapping of *Equation 4*. This means, for a given u_{shadow} and v_{shadow}, we need to find the corresponding $\varphi_\odot$, φ_{occ}, and δ in order to be able to render the planet and the atmosphere from the corresponding position in space. Interestingly, this is a non-trivial problem which cannot be solved analytically but requires an iterative approach. Ultimately, we are looking for a $\varphi_\odot$, φ_{occ}, and δ for which *Equation 3* and *Equation 4* hold under the given u_{shadow}, v_{shadow}, d, $r_\odot$, and r_{occ}. To do this, we use the following reformulation of *Equation 4*:

$$\varphi_{occ} = \frac{\varphi_\odot}{u_{shadow}} - \varphi_\odot \qquad \delta = v_{shadow} \cdot (\varphi_{occ} + \varphi_\odot) \tag{5}$$

As an initial guess for the searched position in space, we use the origin of the occluder ($d_{occ} = 0$, $d_\odot = d$, see *Figure 8*). We compute the apparent angular size of the Sun $\varphi_\odot$ for this position according to *Equation 3*. Together with u_{shadow} and v_{shadow}, we can now use *Equation 5* to compute a φ_{occ} and a δ for this $\varphi_\odot$. With some trigonometry and the law of cosines, we can then update the values for d_{occ} and $d_\odot$:

$$\begin{aligned} d_{occ} &= \frac{r_{occ}}{\sin(\varphi_{occ})} \\ d_\odot &= d_{occ} \cdot \cos(\delta) + \sqrt{d_{occ}^2 \cdot \cos^2(\delta) - d_{occ}^2 + d^2} \end{aligned} \tag{6}$$

This improves our guess for the distances. Each iteration brings us closer to the actual values for d_{occ} and $d_\odot$. We can stop, if the values converge to enough precision. In practice, all relevant positions can be found with at most four iterations to a sufficient precision.

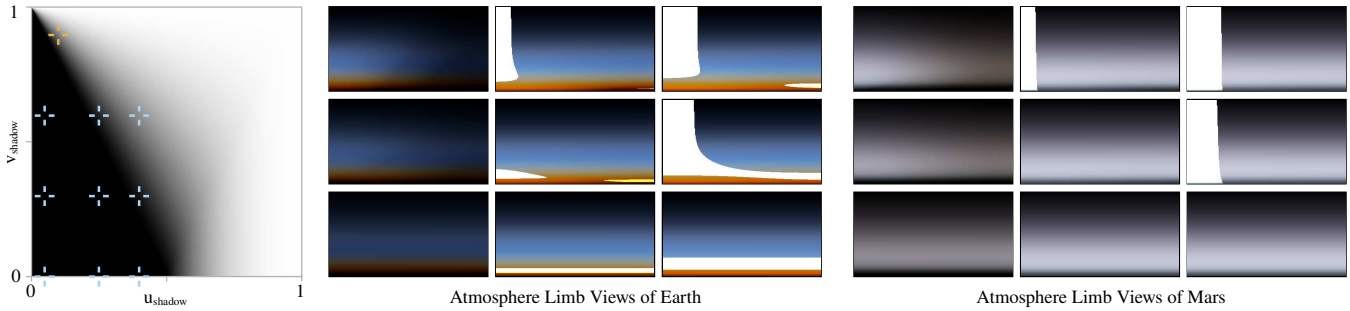


Figure 9: *The left graph shows an eclipse-shadow LuT. The u-axis corresponds to the ratio of the angular radii of Sun and occluder which correlates with the distance to the observer. The occluder is at the left edge of the LuT, the end of the umbra is in the middle, and the right edge is infinitely far away from the occluder. The v-axis is mapped to the relative angular distance between Sun and occluder: The umbra core is at the bottom, the outer edge of the penumbra at the top. The extends of this LuT are marked with a thick black line in* Figure 2*. Next to the LuT, several limb views of the atmosphere around Earth and Mars are shown. The limb views have been captured from positions corresponding to the blue positions marked in the left graph. The orange position is the view from* Figure 7*.*

5.2. Computing the Shadow LuT

For each $[u_{shadow}, v_{shadow}]$ position in the LuT, we reconstruct $\varphi_{\odot}$, φ_{occ}, and δ as outlined above. We use CUDA to render an image of the atmosphere from the corresponding position in space. By rendering the atmosphere from the position corresponding to the shadow-map pixel, we achieve the same sampling quality everywhere in the shadow. This is an advantage over the approach by Yapo et al. where rays are traced starting from the Sun [YC09]. With their approach, a lot of rays have to be traced to achieve a sufficient amount of samples in the dark parts of the shadow.

We use the same atmosphere-rendering implementation as for the real-time rendering, but ported to CUDA for improved double precision support. As only a thin limb of the atmosphere is visible, most of the image would be black. To improve this situation, we use a rendering technique based on spherical coordinates: We map the thin limb to the texture coordinates $[u_{limb}, v_{limb}]$ according to this parametrization:

$$u_{limb} = \frac{\beta}{\pi} \qquad v_{limb} = \frac{\gamma}{\varphi_{atmo} - \varphi_{occ}}. \tag{7}$$

See *Figure 7* for an illustration and *Figure 10* for an explanation of the involved angles. We call this type of images *limb views*. *Figure 9* shows several limb views for our and the Martian atmosphere.

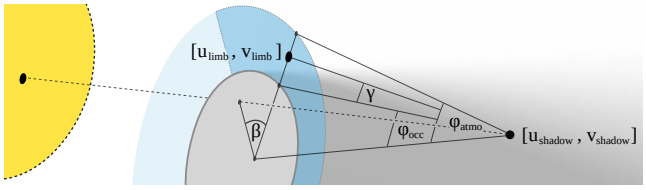


Figure 10: *To compute the total illuminance at a position $[u_{shadow}, v_{shadow}]$ in the shadow LuT, an atmosphere-limb view is rendered. This contains half of the visible atmosphere ring around the planet (darker blue area). The texture coordinates u_{limb} and v_{limb} of the limb view are based on the angles γ, φ_{occ}, and φ_{atmo} as depicted above (see* Equation 7*).*

We use a simple model to account for rough terrain on Earth. During rendering of the intermediate atmosphere images, we retrieve the impact radius for each ray from the texture generated during the precomputation (see *Section 4*). We assume a mean land elevation of 240 m for Earth and attenuate the illuminance of rays linearly between 0 m and 480 m above sea level. A more precise approach would be to use a hypsometric curve which models the altitude distribution of a planet. Yet we found that the terrain only has an impact if the atmosphere is unrealistically clear.

We used a resolution of 256^2 both for the intermediate limb views and for the shadow LuT. Once a limb view is rendered for a position in the shadow LuT, we accumulate the luminance of all pixels to get the total illuminance $I_{indirect}$ at this position. For this, we compute the contribution of each pixel of the atmosphere-limb view separately. Based on the solid angle subtended by each individual pixel, we compute the illuminance a planar surface would receive with the normal pointing towards the Sun. We add the illuminance of the direct sunlight I_{direct} which passed above the upper atmosphere boundary to get the total illuminance $I = I_{indirect} + I_{direct}$. The direct illuminance contribution is computed assuming an opaque occluder with radius r_{atmo} according to circle-intersection method described in [SGAG22]. In the shadow LuT, we store $I/I_{\odot}$, with $I_{\odot}$ being the illuminance at the position in space if the Sun was not eclipsed. In total, 256^2 limb views with a resolution of 256^2 are rendered for a single LuT. The LuT is stored as a 32 bit floating point RGB texture and requires less than 1 MB of memory. The shadow LuT for Earth is shown in the left chart of *Figure 9*. Note that no red light is visible in the umbra as it is several times darker than the penumbra or antumbra.

5.3. Computing the Limb Luminance LuT

A naive attempt in real-time rendering the atmosphere limb around the occluder when observed from within the shadow would result in severe graphical artefacts. These are both visible in the image in *Table 2* and in the auxiliary material provided with this paper. This is because the atmosphere is only a few pixels wide, and only in a small fraction of some pixels light rays are bent exactly towards

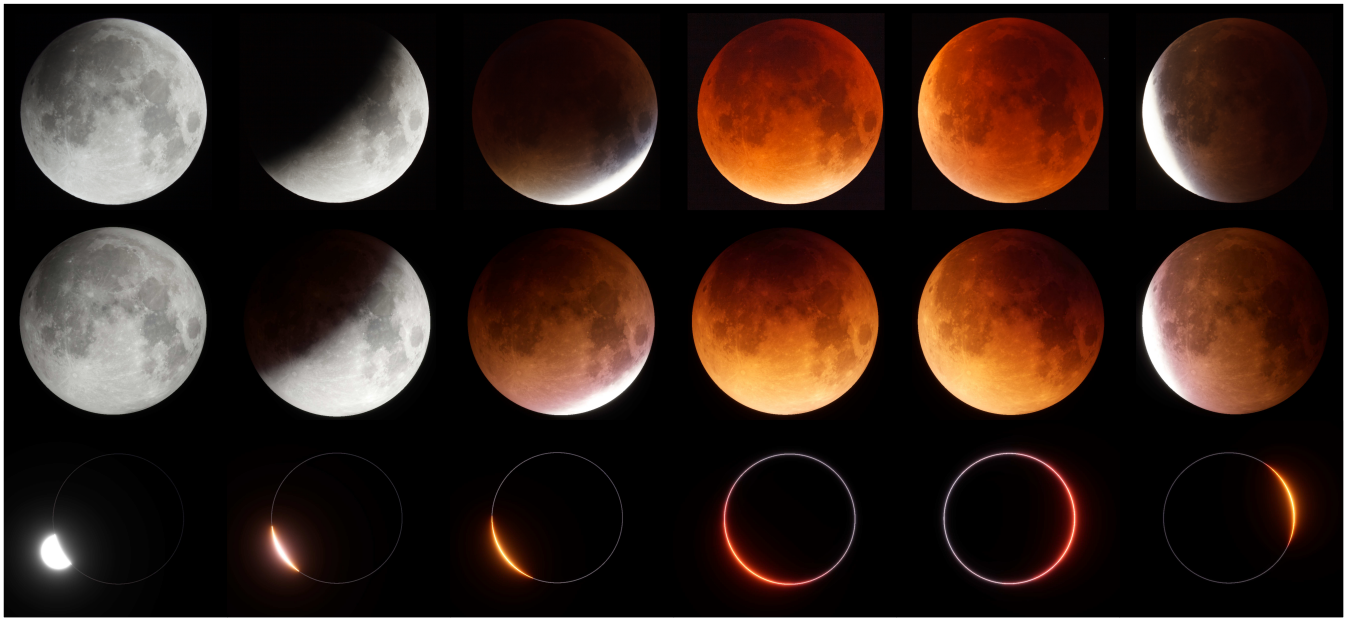


Figure 11: *The upper row shows photographs of the lunar eclipse on 2015-09-28 taken at about 30-minute intervals starting at 1:00 AM UTC. All images use the same white balance and relative exposure values of 0 EV, +4 EV, +9 EV, +13 EV, +13 EV, and +9 EV respectively. The centre row shows renderings using our system. If there was no atmosphere, the third and sixth image would be almost completely black (except for the bright rim). The fourth and fifth image would be completely black. The bottom row shows how Earth would look like for an observer on the lunar north pole. If there was no atmosphere, the Sun would only be visible in the first image.*

the observer. We experimented with multi-sampling and jittering the rays, but found that an unreasonable amount of samples would be needed to get a smooth image. Also, precision issues arise when trying to vary the ray direction slightly.

Hence, we precompute limb views for all positions in the shadow. This is a four-dimensional problem: For each $[u_{shadow}, v_{shadow}]$ position in the shadow, a two-dimensional limb image as shown in *Figure 7* is required. This seems to be infeasible to store in a reasonable amount of memory at first. However, one can observe that we only need this information if the atmosphere is very thin. This means, we can drastically reduce the vertical resolution of the limb view. In our implementation, this is a parameter which can be chosen for the precomputation. All images in this paper and the auxiliary material were generated with a vertical resolution of just a single pixel. For close-up shots of the limb with a very narrow field of view, a higher resolution might be necessary. However, we found that even two pixels can reproduce the atmospheric gradient along the planet's normal quite well.

The precomputation is done similarly as for the shadow LuT. For each $[u_{shadow}, v_{shadow}]$ position, we render a limb image from the corresponding position in space with a user-defined resolution (64x256 in our case). These images are then vertically subsampled to the given amount of pixels. The data for each image is stored in a 3D texture: the first two dimensions are $[u_{shadow}, v_{shadow}]$ and the individual pixel rows of the image are stored consecutively in the third dimension. We found that a resolution of $64 \times 64 \times (64 \times$ verical resolution$)$ is sufficient. Using 32 bit floating point values, this texture requires about $(3 \times$ vertical resolution$)$ MB of memory.

5.4. Real-Time Eclipse Rendering

Computing $\varphi_{\odot}$, φ_{occ}, and δ for a given position in space is trivial. *Equation 4* can then be used to compute the $[u, v]$ position in the shadow LuT. This allows for a straight-forward integration into an existing rendering engine as the incoming light can simply be multiplied by the value retrieved from the shadow LuT. *Figure 11* shows images of the Moon during an eclipse rendered this way.

For the limb luminance, the atmosphere shader first checks the width of the atmosphere ring around the planet. If it is less than a threshold (we used 5 pixels), no atmosphere is rendered but the precomputed value from the limb luminance texture is used. The u and v lookup coordinates are the same as for the shadow LuT. The third dimension is u_{limb} as illustrated in *Figure 7* and *Equation 7*.

If limb views with more than one pixel vertical resolution were precomputed, the pixel rows are stored consecutively in the third dimension of the texture. In this case, the lookup position has to be adjusted according to the angle γ in *Figure 10*. If $\gamma = 0$, the first pixel row is used, if $\gamma = \varphi_{atmo} - \varphi_{occ}$, the last pixel row is used. For all values in between, linear interpolation has to done manually between neighbouring pixel rows. The bottom row of *Figure 11* shows images generated using the limb luminance texture.

It is important to note that we use the glare effect from CosmoScout VR for the renderings in *Figure 11*. This is a luminance-preserving filter which only redistributes the luminance of the frame buffer between neighbouring pixels. It is necessary to add this filter to the images because else the high dynamic range and the luminance of overexposed areas would not be perceivable. For

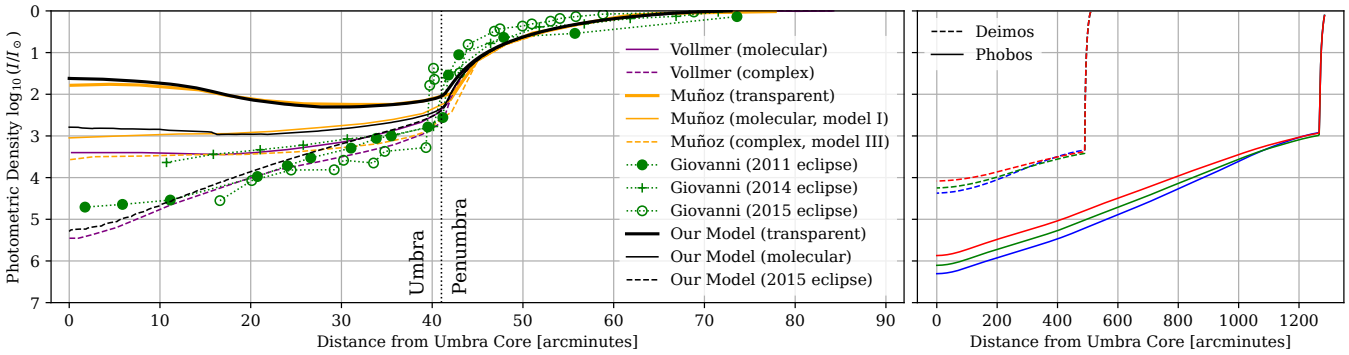


Figure 12: *The left graph shows a cross-section of Earth's shadow at the average distance to the Moon for red light. In the penumbra, our implementation agrees well with models from the domain of lunar Eclipse Theory. The umbra region illustrates the variability in brightness of lunar eclipses. However, our approach produces similar results here as well. The glyphs show measured data for past eclipses. The data for the graphs were extracted from figures from [VG08, MP11, Gio18]. The right graph shows the brightness of an eclipse on the Martian moons Phobos and Deimos as predicted by our system. The RGB colour channels show that in most parts of the shadow, red light is dominant. Only close to the penumbral boundary, the shadow gets grey. An example of how Phobos could look like during an eclipse is given in* Figure 1.

instance, the bottom figures of *Figure 11* would only show a white one-pixel ring. If the exposure was reduced, the part of the ring where the Sun is visible would turn red indeed, but the rest of the ring would become invisible.

6. Evaluation

We evaluated our system in terms of visual quality and performance. The quality evaluation was done by comparing the results of our system to photographs of a past lunar eclipse, to measured data, and to models from the domain of Lunar Eclipse Theory. For the performance evaluation, we measured the precomputation time and the real-time rendering speed.

6.1. Quality

Depending on the atmospheric properties, the brightness of lunar eclipses can vary by several magnitudes [Gio18]. This makes a direct comparison to photographs of past eclipses difficult. Nevertheless, we attempted to simulate the eclipse of 2015-09-28 which was exceptionally dark due to the eruption of the Calbuco volcano in Chile a few months prior. To simulate this, we doubled the amount of aerosols in the atmosphere and increased their scale height from 1200 m to 3000 m. A more sophisticated modelling of the atmospheric conditions would be possible, but the results are already close to photographs of the event (see *Figure 11*).

A more robust way to evaluate the physical soundness of our technique is to compare it to models from the domain of Lunar Eclipse Theory. The thick upper lines in *Figure 12* show the hypothetical brightness of the shadow if no light was scattered or absorbed in the atmosphere. Here, our simulation is close to the corresponding model by Muñoz et al. [MP11]. If molecular scattering is simulated, the shadow gets darker (thin solid lines). This can be assumed to be a theoretical upper brightness limit for a lunar eclipse. Here, our system produces similar results as the model by Muñoz et al., the corresponding model by Vollmer et al. produces slightly darker shadows [VG08].

If components like ozone, clouds, and other aerosols are added to the atmosphere, the shadow can get darker by several magnitudes (dashed lines). Vollmer et al. modelled a wide variety of atmospheric conditions. Their most complex model includes compo-

Figure 13: *Our system allows for a fine-grained control of the atmospheric properties. For the left images, we simulated an atmosphere without ozone. The centre images have been generated using a tent-shaped ozone distribution with a peak altitude at* 25 km*, a width of* 15 km*, and the peak molecular density number shown in the caption. The right image shows two reference photographs of the 2015 lunar eclipse. The upper images show the Moon in the umbra, the lower images at the edge of the penumbra.*

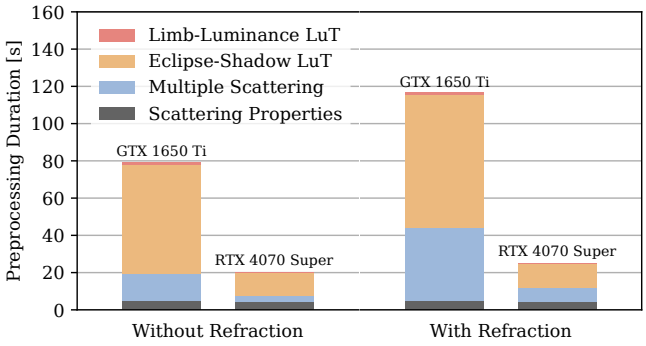


Figure 14: *This graph shows the time it takes to precompute everything needed for an eclipse. First, the atmospheric scattering properties are precomputed. Thereafter, multiple scattering is precomputed iteratively. Finally, the eclipse-shadow LuT and the limb-luminance LuT are generated. The multiple-scattering computation is about two times slower if refraction is enabled.*

nents like ozone, clouds, dust, other aerosols, and topology. Despite being much simpler, our atmosphere parameterization for the 2015 eclipse produces a similar shadow. If we fine-tuned the vertical distribution of the aerosols to historic weather conditions, we could probably match their results even better. With high-density low-altitude aerosols, we could even mimic the effect of clouds.

Visualizing the effect of ozone on the colour of the eclipse shadow is an interesting additional approach for validating the physical soundness of our technique. *Figure 13* shows how our system predicts the Moon to look like with varying ozone concentrations in the atmosphere. The reference images fit well to measured ozone concentrations which are typically between 4×10^{18} $1/\mathrm{m}^3$ and 6×10^{18} $1/\mathrm{m}^3$ [MWR*21].

6.2. Performance

We did a performance evaluation in terms of precomputation time and real-time rendering speed. Both were measured on two different hardware configurations: a laptop equipped with an NVIDIA GTX 1650 Ti and a desktop PC with an NVIDIA RTX 4070 Super.

lunar Surface	GTX 1650 Ti	RTX 4070 Super
a) without shadow	0.28 ms	0.048 ms
b) with shadow	0.33 ms	0.052 ms

Table 1: *The scenes above were rendered at Full HD resolution on two different hardware configurations. Using GPU timer queries, we measured the time it took on average to render the lunar surface with and without the shadow.*

Preprocessing Time

Preprocessing time is not very important for our system, as it only needs to be done once for each celestial object. However, it is still interesting to see the overhead introduced by the refraction computation. *Figure 14* shows the time it took to precompute everything needed for a lunar eclipse on both hardware configurations.

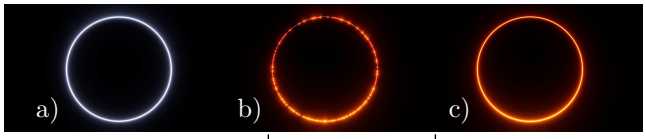
First, we compute the scattering properties (phase functions, scattering, and absorption coefficients) of the individual atmospheric components for 15 different wavelengths using Mie Theory according to [SMG*24]. Next, we precompute multiple-scattering in the atmosphere according to the original implementation by Bruneton et al. and the extended version with refraction. Our implementation with refraction is about two times slower than the original implementation without refraction. Finally, the eclipse-shadow LuT and the limb-luminance LuT are generated. Due to the lower resolution, the limb-luminance generation is much faster than the shadow LuT generation. The entire process takes about two minutes on lower-end consumer hardware.

Real-Time Rendering

To evaluate the real-time performance, we rendered two different scenes at full HD resolution on both hardware configurations.

The first scene shows the lunar surface with and without the shadow. The Moon is drawn as a simple textured ellipsoid with a few triangles only. *Table 1* shows the average time it took to render the scene. It can be seen that the shadow has a very small impact on the rendering time.

Computationally more intense is the rendering of the atmosphere including refraction around the planet. For this, we chose a viewpoint inside the shadow of Earth looking towards the Sun. The results are shown in *Table 2*. If refraction is disabled (a), predominantly blue light reaches the observer due to scattering and the iconic red ring is missing. If refraction is enabled (b), the red ring becomes visible and the scattered light becomes invisible because of the decreased exposure time. Also, rendering the atmosphere becomes about 20 % slower due to the additional texture lookups and computations. The apparent artefacts in (b) are mitigated by pre-integrating the limb luminance (c), which also makes the rendering a bit faster than the original implementation without refraction.



Sun behind Earth	GTX 1650 Ti	RTX 4070 Super
a) without refraction	1.38 ms	0.72 ms
b) with refraction	1.74 ms	0.76 ms
c) with pre-integrated limb luminance	1.25 ms	0.75 ms

Table 2: *This table shows the average time it took to render the atmospheres seen above at Full HD resolution.*

7. Summary and Future Work

We presented a novel approach for rendering eclipses in real-time. To the best of our knowledge, it is the first which can simulate the shadow intensity at any point in the shadow cone of a celestial body. Also, it is the first not being limited to the Earth-Moon system. It can simulate the shadow of other celestial bodies like, for instance, the shadow of Mars. Last but not least, it is the first approach which can simulate astronomical refraction in the atmosphere of the occluder in real-time, which allows for spectacular visualizations of the Sun being eclipsed. All this is made possible by using three additional lookup textures which are precomputed using an extended version of the Bruneton atmosphere model.

We have validated our approach by comparing the results to reference data for lunar eclipses. As there seems to be no data available for eclipses beyond the Earth-Moon system, we have predicted how an eclipse on a Martian moon would look like. In future, with missions like MMX [CYT*18], we will be able to validate our predictions. We could also use our approach to simulate the shadows of other bodies, like for instance some Jovian moons for which data is available [Mal91]. Also, it will be promising to use the information from the limb-luminance texture for advanced lighting effects such as specular highlights on satellites or for ambient lighting. This could increase the realism of scenes close to the occluder as for instance a spacecraft flying through the shadow. Last but not least, one could explore the possibilities of incorporating clouds when rendering the atmosphere limb to reproduce the patchy limb seen in photographs (*Figure 3*).

We have integrated the atmospheric refraction and the eclipse shadows into CosmoScout VR [SFGG25], an open-source virtual-reality simulation of our solar system. We use this software frequently for public outreach events, and the new features help in explaining celestial mechanics in general and the physics of eclipses in particular. Especially the moment when the visitors land on the lunar surface and watch the Sun being eclipsed by the Earth is always an inspiring experience.

Acknowledgements

We thank Rolf Hempel for his reference images of the 2015 lunar eclipse. We also thank the reviewers who significantly helped to improve the conciseness of this paper. This work is supported by the German Aerospace Center (DLR) Space Administration with financial means of the German Federal Ministry of Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag as part of the VaMEx3-RGE project (50NA2204A). Open Access funding enabled and organized by Projekt DEAL.

References

[BN08] BRUNETON E., NEYRET F.: Precomputed atmospheric scattering. In *Computer graphics forum* (2008), vol. 27, Wiley Online Library, pp. 1079–1086. 3

[Bru16] BRUNETON E.: A qualitative and quantitative evaluation of 8 clear sky models. *IEEE transactions on visualization and computer graphics 23* (2016), 2641–2655. 5

[CYT*18] CAMPAGNOLA S., YAM C. H., TSUDA Y., OGAWA N., KAWAKATSU Y.: Mission analysis for the martian moons explorer (mmx) mission. *Acta Astronautica 146* (2018), 409–417. 11

[Gio18] GIOVANNI G. D.: Lunar eclipse brightness and the terrestrial atmosphere. *Journal of the British Astronomical Association 128* (Feb. 2018), 10–17. 3, 9

[GV08] GEDZELMAN S. D., VOLLMER M.: Simulating irradiance and color during lunar eclipses using satellite data. *Applied optics 47*, 34 (2008), H149–H156. 4

[Hil20] HILLAIRE S.: A scalable and production ready sky and atmosphere rendering technique. *Computer Graphics Forum 39* (2020), 13–22. 3

[HM66] HANSEN J. E., MATSUSHIMA S.: Light illuminance and color in the Earth's shadow. *Journal of Geophysical Research 71*, 4 (1966), 1073–1081. 3

[HSV08] HERNITSCHEK N., SCHMIDT E., VOLLMER M.: Lunar eclipse photometry: absolute luminance measurements and modeling. *Applied Optics 47*, 34 (2008), H62–H71. 3

[Mal91] MALLAMA A.: Light curve model for the Galilean satellites during Jovian eclipse. *Icarus 92*, 2 (1991), 324–331. 11

[MED*12] MÜLLER D., ENGEL J., DÖLLNER J., MÜLLER D., ENGEL J., DÖLLNER J.: *Single-Pass Rendering of Day and Night Sky Phenomena*. The Eurographics Association, 2012. 3

[MP11] MUÑOZ A. G., PALLÉ E.: Lunar eclipse theory revisited: Scattered sunlight in both the quiescent and the volcanically perturbed atmosphere. *Journal of Quantitative Spectroscopy and Radiative Transfer 112*, 10 (jul 2011), 1609–1621. 3, 5, 9

[MWR*21] METTIG N., WEBER M., ROZANOV A., AROSIO C., BURROWS J. P., VEEFKIND P., THOMPSON A. M., QUEREL R., LEBLANC T., GODIN-BEEKMANN S., KIVI R., TULLY M. B.: Ozone profile retrieval from nadir TROPOMI measurements in the UV range. *Atmospheric Measurement Techniques 14*, 9 (2021), 6057–6082. 10

[SFGG25] SCHNEEGANS S., FLATKEN M., GILG J., GERNDT A.: CosmoScout VR 1.10.0, Jan. 2025. URL: https://doi.org/10.5281/zenodo.14748678. 6, 11

[SGAG22] SCHNEEGANS S., GILG J., AHLERS V., GERNDT A.: Real-time rendering of eclipses without incorporation of atmospheric effects. *Computer Graphics Forum 41*, 7 (2022), 279–289. 2, 3, 4, 6, 7

[SGGC04] SERON F. J., GUTIERREZ D., GUTIÉRREZ G., CEREZO E.: Visualizing sunsets through inhomogeneous atmospheres. In *Proceedings Computer Graphics International, 2004.* (2004), IEEE, pp. 349–356. 3, 4, 5

[SL90] SCHAEFER B. E., LILLER W.: Refraction near the horizon. *Publications of the Astronomical Society of the Pacific 102*, 653 (1990), 796. 3, 5

[SLPH03] SAMPSON R. D., LOZOWSKI E. P., PETERSON A. E., HUBE D. P.: Variability in the astronomical refraction of the rising and setting sun. *Publications of the Astronomical Society of the Pacific 115*, 812 (2003), 1256. 3, 5

[SMG*24] SCHNEEGANS S., MEYRAN T., GINKEL I., ZACHMANN G., GERNDT A.: Physically based real-time rendering of atmospheres using mie theory. *Computer Graphics Forum 43*, 2 (2024), e15010. 3, 4, 10

[The67] THE SURVEYOR INVESTIGATOR TEAMS: *Surveyor III Mission Report, Part II: Scientific Results*. Tech. rep., Jet Propulsion Laboratory, 1967. 2

[vdW08] VAN DER WERF S. Y.: Comment on "Improved ray tracing air mass numbers model". *Applied Optics 47* (2008). 3, 4

[VG08] VOLLMER M., GEDZELMAN S. D.: Simulating irradiance during lunar eclipses: the spherically symmetric case. *Applied Optics 47*, 34 (2008). 3, 9

[WS14] WESTFALL J., SHEEHAN W.: *Celestial shadows: Eclipses, transits, and occultations*, vol. 410. Springer, 2014. 4

[YC09] YAPO T. C., CUTLER B.: Rendering lunar eclipses. In *Proceedings of Graphics Interface 2009* (CAN, 2009), GI '09, Canadian Information Processing Society, p. 63–69. 3, 7