Physically Based Rendering of the Martian Atmosphere

Peter Collienne[*†], Robin Wolff[*], Andreas Gerndt[*], Torsten Kuhlen [†]

[*] DLR Simulations und Softwaretechnik
Lilienthalplatz 7
38108 Braunschweig
Tel.: +49 / 531 / 295-2782
Fax : +49 / 531 / 295-2767
E-Mail: Andreas.Gerndt@dlr.de

[†] RWTH-Aachen Virtual Reality Group
Kopernikusstraße 6
52074 Aachen
Tel.: +49 / 241 / 80 24783
Fax : +49 / 241 / 80 22134
E-Mail: peter.collienne@rwth-aachen.de

Abstract: With the introduction of complex precomputed scattering tables by Bruneton in 2008, the quality of visualizing atmospheric scattering vastly improved. The presented algorithms allowed for the rendering of complex atmospheric features such as multiple-scattering or light shafts in real-time and at interactive framerates. While their published implementation corresponding to the publication was merely a proof of concept, we present a more practical approach by applying their scattering theory to an already existing planetary rendering engine. Because the commonly used set of parameters only describes the atmosphere of the Earth, we further extend the scattering formulation to visualize the atmosphere of the planet Mars. Validating the modified scattering and resulting parameters is then done by comparison with available imagery from the Martian atmosphere.

Keywords: Atmosphere Rendering, Light Scattering, Mars

1 Introduction

Atmospheric visualization is widely used in computer graphic applications such as videogames, special effects in movies or in scientific visualization. Especially when used in a real-time application, rendering a realistic looking atmosphere is a trade-off between quality and performance. With the algorithms presented by Bruneton [BN08] in 2008, a realistic and plausible atmosphere can be rendered in real-time, including effects such as multiple-scattering and lightshafts. The available implementation published by Bruneton corresponding to the publication is limited to spherical surfaces only and is rather designed as a proof-of-concept rather than as a plugin for existing renderers. Thus the here presented implementation extends this approach by creating a plugin-like structure to work with an arbitrary planetary renderer. This allows for an increased amount of realism for any planet rendering and improved presence inside a virtual reality environment. While the Earth's atmosphere is, due to its convenience of viable reference, the commonly used example case for atmospheric rendering, the visualization of other planet's atmospheres have become increasingly desirable, e.g. for the planet Mars. Visualizing the Martian atmosphere is challenging because of its difference in scattering properties but still can be done in a plausible manner with using the same atmospheric rendering technique as used for rendering the Earth's atmosphere.

2 Related Work

Previous to the already mentioned work by Bruneton, several approaches have been made to visualize an atmosphere in real-time. Research on the scattering theory inside the atmosphere and its implementation has been done by Nishita et al. [NSTN93]. Their approach is based on a clear sky atmospheric model, assuming the atmosphere consists only of two types of particles, namely gas-molecules and aerosols. Despite using a precomputed 2D lookup table to improve performance on calculating the scattering-integral, their implementation is not suitable for real-time rendering. Hoffmann and Preetham [HP02] further analysed the light scattering inside the atmosphere by describing absorption, in- and out-scattering and the concept of the aerial perspective. Based on the physical model presented by Nishita [NSTN93], ONeil [ONe05] improved the rendering speed using low sampling of the scattering integral in the vertex shader. His usage of a polynomial to approximate the transmittance results in a real-time implementation. However to ensure interactive framerates, the number of samples is limited. Still the visual quality of the outcome is high while being able to render in real-time and thus this approach is very suitable for the usage in e.g. video-games. Schafhitzel [SFE07] et al. present a method based heavily on precomputation. Their approach is using a 3D texture parameterized by observer height, view-direction and angle of incident sunlight. During runtime the scattering integral is substituted by one texture-lookup per pixel. While they provide images of a Martian atmosphere, they do not describe the parameters used and if this approach differs from rendering Earth's atmosphere.
In 2008, Bruneton [BN08] presented a method to precompute all light scattering for any observer position, view angle and angle of incident light. Using a four-dimensional texture, all needed information can be accessed at runtime by performing texture lookups for the current view-configuration. This approach includes atmospheric effects like multiple scattering and light-shafts, which no other implementation was able to render in real-time. Additionally, Sperlhofer [Spe11] further discussed Bruneton's approach by applying it to a custom build planetary renderer to visualize the Earth.
All approaches are using a set of parameters resembling the atmosphere of the Earth. The reason the following work is based on the approach by Bruneton is the high physicality of their implementation in addition to the support of multiple scattering. While multiple scattering not being included in this implementation, the possibility of adding it with only increasing the amount of calculations in the precomputation step is an important benefit to allow a scientific visualization which is close to the real scattering inside the atmosphere. Using Bruneton's approach, we thus present an implementation for the visualization of the Martian atmosphere which is designed as a plugin for existing planetary rendering engines such as the renderer provided by the *German Aerospace Center* (DLR), developed by Westerteiger [WGH$^+$11].

3 Introduction to Atmospheric scattering

As previously mentioned, there have been a number of approaches for the visualization of atmospheric scattering effects. To give a short introduction on the scattering theory used in state of the art atmosphere rendering, we first present the basic scattering equations by Rayleigh and Mie, fit to atmospheric rendering by Nishita [NSTN93].

3.1 Rayleigh-Scattering

The so-called *Rayleigh-Scattering* explains computation of light-scattering due to small particles (smaller than 10% of the incident lights wavelength [Ray99]). Because these particles have a very small cross-section, the wavelength λ has a strong impact on the resulting scattering. This is due to the higher probability of short wavelengths to collide with a given particle, thus resulting in increased scattering for lower wavelengths. When computing the scattering-coefficients which describe the amount of reflection from incident light, the scattering parameter is divided in reflection coefficient $\beta_{R/M}$ and the phase function $\gamma_{R/M}$, indicating the directional characteristic.
The corresponding equations for the Rayleigh-Scattering are described by the equations

$$\beta_R(h,\lambda) = \frac{8\pi^3(n^3-1)^2}{3N_s\lambda^4}\rho(h) \tag{1}$$

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

with

$$\rho(h) = e^{-\frac{h}{H_R}} \tag{3}$$

and h being defined as the observer height over ground $h = r - R_{Ground}$ with r denoting the distance of the observer to the planet's origin and R_{Ground} the altitude of the surface. H_R is the scale height or thickness of the atmosphere if it had uniform density and N_s represents the molecular number density of the standard atmosphere. n is the refraction index of the air, θ is the angle of the incident light and ρ is called the density ratio.

3.2 Mie-Scattering

Mie-Scattering accounts for particles which are of equal or greater size than λ. These particles are commonly referred to as aerosols, which cause a dusty blur on objects far away from the observer. Since their size is greater than the wavelength mixture of visible light emitted from the sun, all light gets scattered equally. This results in omitting the wavelength dependency in equation 1, resulting in the equation

$$\beta_M(h) = \frac{8\pi^3(n^3-1)^2}{3N_s}\rho(h). \tag{4}$$

The phase function

$$\gamma_M(\theta) = \frac{3}{8\pi}\frac{(1-g^2)(1+cos^2(\theta))}{(2+g^2)(1+g^2-2gcos(\theta))^{\frac{3}{2}}} \tag{5}$$

is a result from the Henyey-Greenstein phase function and was improved by Cornette [CS92]. The asymmetry factor g is a constant and depends on conditions like haze or dusty air. If $g = 0$, this function is equal to *Rayleigh-Scattering.*

4 Precomputing the Light Contribution

Based on the presented scattering theories, the light intensity reaching an observer inside the atmosphere can now be computed. The formulation of incident light for an observer inside the atmosphere is split into direct light, inscattered light and reflected light. While direct light is the circle of the sun visible on the sky which is rendered later as an additional post-processing effect, the inscattered light, being the global illumination inside the atmosphere, and the reflected light, when looking at the planet's surface, contribute to the overall atmosphere brightness.

4.1 Transmittance

Independent of the direction of incident light, we need a formulation on how much light is reaching the observer when emitted from any arbitrary point inside the atmosphere. With $\beta_{R,M}$ representing the probability of light being scattered away at a certain height, we can calculate this attenuation. Computing a for an arbitrary ray inside the atmosphere is done by tracing the view-vector and summing the respective values of $\beta_{R,M}$ along the ray using

$$a(\lambda, r) = \int_0^d \beta_R(\lambda, h(r))\beta_M(\lambda, h(r))\mathrm{dr}. \tag{6}$$

The following equation

$$t(\lambda, r) = e^{-a(\lambda,r)} \tag{7}$$

then converts the attenuation coefficient a into the extinction factor, here referred as the transmittance t. This transmittance is defined as a percentage of incident light from the start point of the ray r, reaching the end of the ray.
As seen in equation 6, the transmittance only depends on the height h and the distance d to the edge of the atmosphere. Because the distance d from any point inside the atmosphere is connected to the view-angle, we have a description for the transmittance at any height h and view-angle α. Since the transmittance does not change during rendering, it can be entirely precomputed and stored inside a texture (Figure 1)

4.2 Inscattered Light

Without accounting for inscattered light, the atmosphere would appear black whenever the view-ray is not directed at the sun. But because light is scattered into the view ray at each point along the ray, the atmosphere has a high overall brightness.
A physical representation of inscattered light is given by equation

$$I_{inscatter} = \int_x^p t(\lambda, r')j(y, v)dy \tag{8}$$

and

$$j(y, v) = \int_0^{4\pi} (\beta_R(y)\gamma_R(v \bullet \theta) + \beta_M(y)\gamma_M(v \bullet \theta)) \cdot I_{total}(y, v', d')\mathrm{d}\theta. \tag{9}$$

The equations represent integration along the view-ray and evaluate the inscattered light at each point along the ray. With r' being the vector from the observer at x to an arbitrary point y. Performing the spherical integration $j(y, v)$ for the precomputation is called *multiple scattering* and was, as previously mentioned, first presented by Bruneton in a real-time implementation. For the sake of simplicity we only take single-scattering into account. While the loss of general brightness is tolerable, multiple scattering can later be added by including iteration over the inscattered light [BN08]. Single-scattering is the result of omitting the spherical integration at each point along the view-ray while only taking the direct light into account.
The parameters of the inscattered light are, in addition to the already used height above ground and view-angle, the angle of the incident light and the angle between the view and the light-direction. Those are needed to be able to compute the incident light to every point along the view-ray. For a resulting precomputation, we solve the integral along the view-ray by sampling and trapezoidal integration. At each sample point, the intensity of incoming light is computed using the transmittance table we previously generated. The thus computed intensity is then attenuated from the sample point up until it reaches the observer. To account for all possible daytimes, the precomputation is performed for all light directions and for all angles between view-direction and light direction, resulting in the need of a four-dimensional texture to store all needed data for a real-time visualization.
The resulting four-dimensional inscatter-table, which is rendered into a 3D texture, is shown in figure 1.

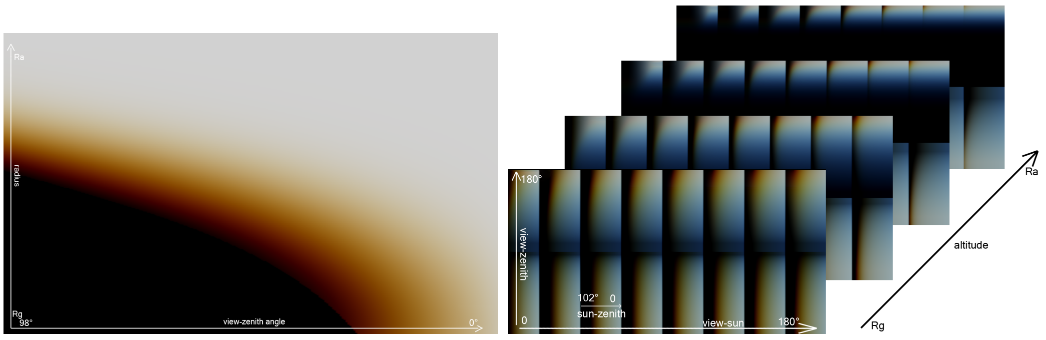

Figure 1: Left: Transmittance Texture; Right: Inscatter Texture

4.3 Reflected light

With now having a formulation to compute the light intensity at any position inside the atmosphere, still a formulation for a view-ray intersecting with the planet is missing. The light incident to any point on the planet's surface can be computed by integrating over the hemisphere at said point, calculating the overall light intensity from all directions. This is again called multiple-scattering but is, as with the inscattering, omitted in this implementation for the sake of simplicity. To still account for the reflected light, again only the direct light reaching the planet's surface is taken into account. In the case of an already shaded planet, e.g. because of the planetary renderer already applying a phong lighting, we are able to directly use this lighting information and only need to attenuate it twice.

5 Implementation

While the previous chapters briefly described the theoretical approach to atmospheric scattering, we are now able to use this approach to add this atmosphere to an planetary rendering engine using the methods introduced by Nishita and used by Bruneton. Precomputing both the transmittance and inscatter texture can be performed once before rendering the atmosphere. With this data at hand, only the parameters for acessing the textures are yet to be computed before the atmosphere can be rendered around the planet. Obtaining these parameters without having detailed acess to the scene itself is described in the next section.

5.1 Accessing the Textures

For the following computations we assume a scenario in which the atmosphere is to be applied to an existing, arbitrary renderer with no influence on the already rendered planet and little information about the scene itself. The previously generated inscatter-texture needs four parameters to be accessed correctly during runtime.
Starting with the observer's height above ground, the current position of the camera relative to the planet's radius is either given or can be obtained by extracting the modelview matrix from the currently used rendering context. Is the planet in the origin of the coordinate-system, the height above ground is the length of the camera's position vector. Otherwise the planet and the observer's position first have to be moved so that the planet's origin lies on the coordinate-system's origin.
Next up is the computation of the view-angle parameter. This angle is unique for each pixel on the screen and is thus most efficiently computed on the GPU. Using the hardware-accelerated interpolation of vertex-attributes, we can get each pixel's world position by creating a vertex at each edge of the far-plane.
By setting the viewport to an artificial unit-viewport and placing four vertices to each edge of back-plane of the the view-frustum, we obtain the quad representing the far-plane. Using the inverted projection matrix, we can now project this far-plane back into world-coordinates and thus have both the edges of the view-frustum and, by using linear interpolation, all position vectors for each pixel on the far-plane in world-coordinates. Subtracting these values for each pixel from the camera position, we obtain the view-direction per pixel which can directly be converted into the view-zenith angle by using the dot-product.
The two missing parameters for accessing the inscatter texture are both depending on the direction of incident light. In the event that the planet is already shaded, that light direction

has to be used. Is the planet not shaded, the normals on the surface of the planet have to either be given or computed. The angle between the view-direction and the light direction can again be computed using the dot-product.
Since the light reaching the observer has to be computed for each pixel on the screen, the whole computation is, as already mentioned, performed on the GPU inside a fragment-shader.

5.2 Accounting for Rough Terrain

Adding the atmosphere to the rendered planet yields incorrect results. This is due to the simplified representation of the planet as only a radius without features such as mountains or valleys. Every surface feature which lies above or under the parameterized planet radius is not accounted for and thus does not block the atmosphere as it should (figure 2). Bruneton

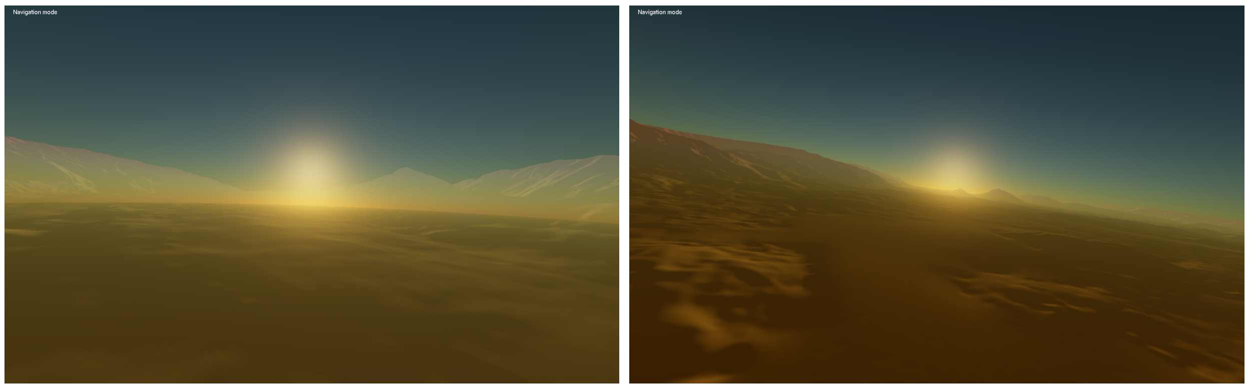

Figure 2: Left: Not accounting for rough terrain; Right: Using the corrected light intensity

proposed a solution to this problem by computing the atmosphere color at both the camera position and the surface point with the same view-direction. Because the light intensity is additive, these can be subtracted and thus yield the correct atmosphere color from the observer to the point on the planet's surface.
To apply this in a practical sense, the coordinates of each intersection of the view-rays with the planet's surface have to be known. Assuming the atmosphere only being an post-processing effect, this results in the need of accessing the depth buffer after the planet was rendered.
After linearizing the depth, we can use the previously per-pixel computed view-direction in world coordinates to reconstruct each pixel's world position. Is the computed world coordinate of the pixel differing from either the far-plane or the parameterized radius of the planet, we then can perform an additional texture lookup to correct the light intensity.

6 Parameter Estimation

With the theory and implementation of state of the art atmospheric visualization described in the previous sections, the visualized atmosphere is still based on the Earth's atmosphere and its physical properties. We will now first discuss the parameters resembling the atmosphere of the Earth and then develop a method to use the implemented visualization to visualize a Martian atmosphere.

6.1 Earth Atmosphere Parameterization

Recall the scattering parameters $\beta_{R,M}$ and their dependencies (eq. 1 and eq. 4). While Nishita [NSTN93] discusses these equations in terms of atmospheric scattering, he does not explicitly give the parameters used for the presented example images. Bruneton, who based his work on the presented equations for $\beta_{R,M}$ and $\gamma_{R,M}$, uses values presented by Riley [REK$^+$04] for his implementation [BN08].

$$\beta_R = (5.8, 13.5, 33.1)10^{-6} \quad \textit{for wavelengths} \quad \lambda = (680, 550, 440)10^{-9}m \tag{10}$$

Riley uses a molecular number density at sea level $N_s = 2.55 * 10^{25}$ and a set of wavelengths resembling red, green and blue light (eq. 10) for each scattering parameter. Information missing however is the used refraction index of air n. Reconstructing the used refraction index based on the given scattering coefficients β_R yields the used refraction index $n = 1.000206102$ for all three wavelengths. This value however differs from the refraction of dry air at $15°C$ defined as $n = 1.000276$ for $\lambda = 680nm$ [1]. Also a phyiscal reason for choosing the wavelengths is given in neither publication.
While all parameters are based on physical data available for the standard atmosphere of Earth, the choice of both refraction index and wavelengths is influenced by the visual outcome of the rendering. Due to the convenient way of comparing a rendering with the Earth's atmosphere on photographs, adjusting these parameters to fit the atmosphere yields a very high image quality.

6.2 Martian Atmosphere Parameterization

However, for a Martian atmosphere, estimating parameters is limited by the availability of both physical data and viable visual reference. Starting with a physical approach, the Martian atmosphere weights about 0.5% of the Earth's atmosphere and has a scale height of $11km$ compared to the Earth's $7km$. It consists to >95% of CO_2 and has, although being very thin compared to the Earth's atmosphere, a high amount of aerosols, which let the atmosphere appear very dusty [Rob06]. The dust gets the characteristic red-yellowish color from iron oxide which also partially covers the surface of the planet.
Based on these observations, the direct use of Rayleigh- and Mie-Scattering would yield incorrect results. While Mie-Scattering is simulating the light scattering at larger particles and is thus independent of the light's wavelength, the dust particles in the Martian atmosphere

[1] refractiveindex.org

however are scattering red light while absorbing blue light. The Rayleigh-Scattering on the other hand is barely visible due to the very thin atmosphere compared to e.g. the Earth's atmosphere. Thus even though the atmosphere would appear in a darker blue when the sky would be clear of the iron oxide dust, the Rayleigh-Scattering impact on the visual outcome is negligible.
Since the scattering theory by Nishita [NSTN93] is hardly applicable in the context of the Martian atmosphere, we still want to get a visual convincing visualization using the provided tools. Usually the scattering coefficients behave in a way that with increasing wavelength, the probability of scattering decreases (eq. 1). On Earth, this results in a blue sky when the sun is high at the sky, because blue light is scattered widely across the atmosphere. When the sun sets or rises, the distance the light travels through the atmosphere is longer than during the day, resulting in increased scattering of the blue light. With the blue light being almost completely scattered away, the sky appears yellow or red. On Mars on the other hand, the sky has, due to the iron oxide in the dust, a yellow or red-ish color during daylight. This is again because the red light is scattered widely inside the atmosphere just like the blue light in the Earth's sky. When the sun is setting or rising on Mars, the red light is, just like the blue light on Earth, scattered over a longer distance inside the atmosphere and thus loses intensity. Because of this effect, a Martian sunset shows a blue hue over the horizon in comparison to the Earth where the sky appears yellow or red (figure 3).

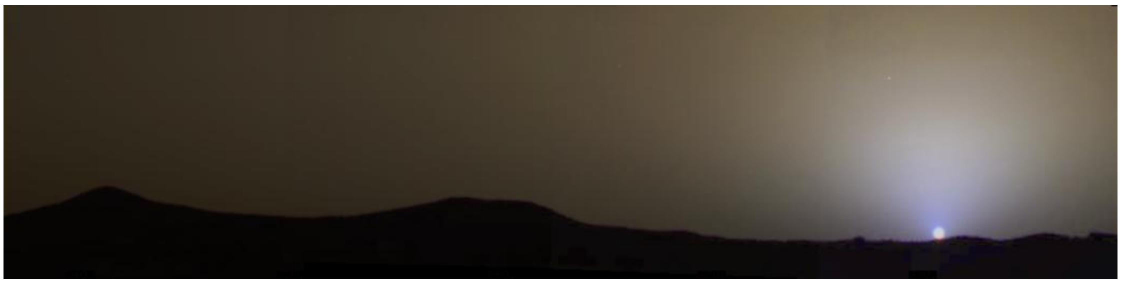

Figure 3: Sunset as seen by the Pathfinder Mars Lander

Using this characteristic, we can adjust the scattering coefficients to simulate an equivalent scattering formulation. Instead of having increased scattering for short wavelengths we want the inverse effect, namely the increased scattering of red light, thus of longer wavelengths. The other parameters for the scattering coefficient β_R can be adjusted for an atmosphere consisting of mainly CO_2. Because the Martian sky has a variety of sky colors due to different concentrations of dust and occasional, potential planet-wide, sandstorms, these parameters are merely estimations and are in this case designed to yield a plausible outlook of the Martian sky. For reference on the Martian sky, we use imagery published by the NASA and taken by several Mars-Rovers or -Landers. Figure 3 was taken by the Mars-Lander *Pathfinder* and shows the blue hue at sunset as well as a pale, red-ish atmosphere color. While this picture is not necessary how human eyes would perceive the atmosphere, NASA published photos taken by the *Curiosity* Rover with a corrected set of colors (figure 4). To resemble this yellow-ish dusty atmosphere, we choose the scattering parameters for the

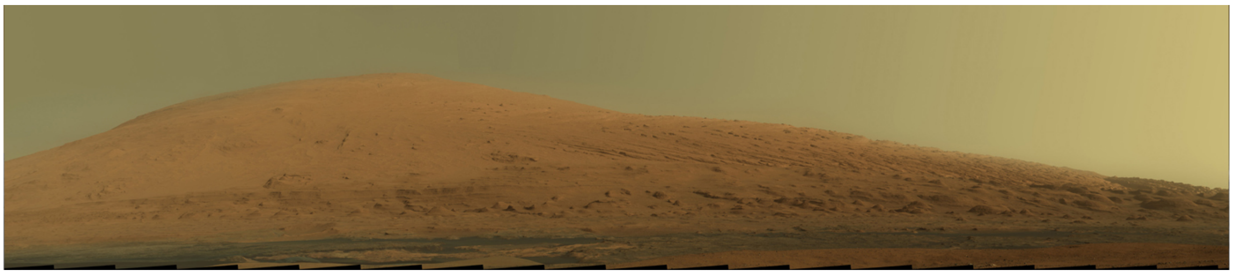

Figure 4: Color-corrected image from mount sharp taken by the curiosity rover

Rayleigh-Scattering as seen in equation 11.

$$\beta_R = (19.918, 13.57, 5.75)10^{-3}; \quad \lambda = (680, 510, 440)10^{-9}m \tag{11}$$

As already mentioned, longer wavelengths are now scattered with a higher probability than shorter wavelengths. While this results in a formulation different from the Rayleigh Scattering as presented by [NSTN93], it is a physically accurate model for the Martian atmosphere because it takes the reflected color of the particles into account. The effect of the Rayleigh Scattering on the other hand is neglectable because of the low density of the Martian atmosphere.
Rendering the same scene as photographed by the *Curiosity* Rover yields a convincing rendering of the Martian atmosphere (figure 5). When rendering a sunset using the proposed

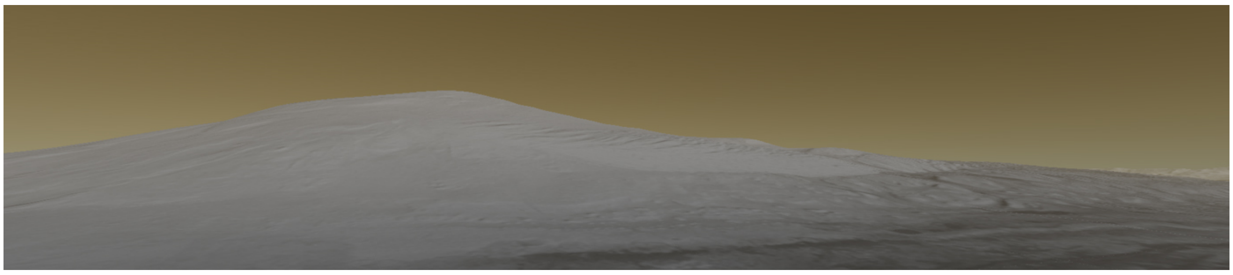

Figure 5: Rendered Atmosphere

parameters above, the sky above the horizon changes to a blue hue just as seen on figure 3. An example render for a sunset on Mars is shown in figure 6. The size and light intensity of the sun are not accurate in this image but since the sun is also parameterized, the parameters can be set to fit the actual size and brightness of the sun as seen from an arbitrary distance. Additional information on how the post-processing of the planet and the sun are performed can be found in [Col13].

7 Conclusion

We proposed a way of computing and rendering a Martian atmosphere by extending existing rendering algorithms to support an arbitrary planetary renderer by only using the previously rendered image and depth map. Rendering the atmosphere is performed during

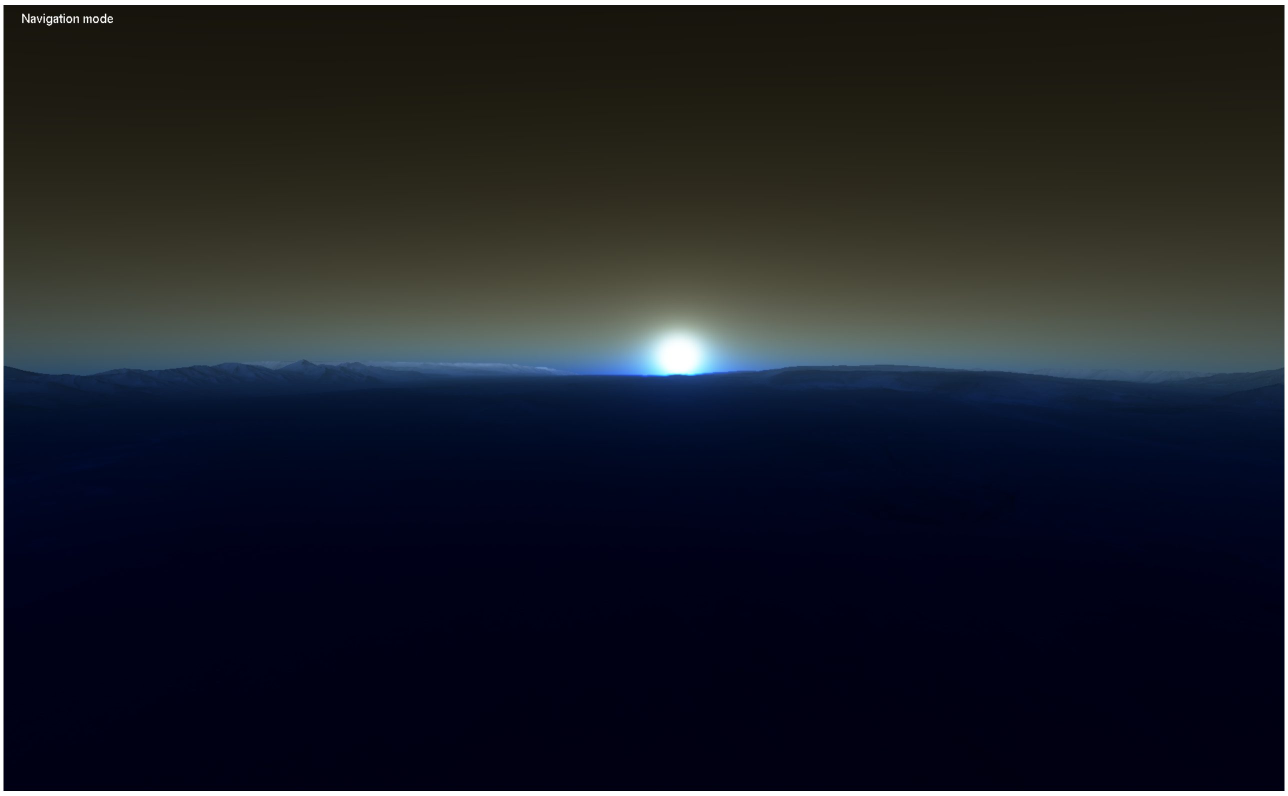

Figure 6: Sunset visualized by the atmosphere renderer

runtime while draining about 10% of the planetary rendering engine's framerate. With the additionally introduced set of parameters, we invert the properties of Rayleigh scattering to visualize the characteristic dusty red atmosphere of Mars while still using the benefits of the precomputation, thus allowing for real-time rendering in a virtual environment.
While this gives a plausible and realistic look for a Martian atmosphere, the implementation still can only render a uniform atmosphere. Different densities of the dusty atmosphere can not be visualized using only one transmittance- and one inscatter-texture. Also the representation is limited to spherical planets, because of the simplifications in parameterization. Improving these approximations can yield to an even more accurate physical representation for the Martian atmosphere, even though more viable sources for the actual outlook of the atmosphere on Mars is needed, in order to appropriately match reality with a rendered image.

References

[BN08] Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. In *Computer Graphics Forum*, volume 27, pages 1079–1086. Wiley Online Library, 2008.

[Col13] Peter Collienne. Interactive visualization of parameterized atmospheres in virtual reality. Bachelor's thesis, Rheinisch-Westfaelische Technische Hochschule

Aachen, 2013.

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

[HP02] Nathaniel Hoffman and Arcot J Preetham. Rendering outdoor light scattering in real time. In *Game developers conference*, page 499, 2002.

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

[ONe05] Sean ONeil. Accurate atmospheric scattering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.

[Ray99] Lord Rayleigh. Xxxiv. on the transmission of light through an atmosphere containing small particles in suspension, and on the origin of the blue of the sky. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 47(287):375–384, 1899.

[REK$^+$04] Kirk Riley, David S Ebert, Martin Kraus, Jerry Tessendorf, and Charles Hansen. Efficient rendering of atmospheric phenomena. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 375–386. Eurographics Association, 2004.

[Rob06] Stuart Robbins. Elemental composition of mars atmosphere. http://burro.astr.cwru.edu/stu/advanced/mars.html, 2006.

[SFE07] Tobias Schafhitzel, Martin Falk, and Thomas Ertl. Real-time rendering of planets with atmospheres. In *Journal of WSCG*, volume 15, pages 91–98. Vaclav Skala-UNION Agency, 2007.

[Spe11] Stefan Sperlhofer. Deferred rendering of planetary terrains with accurate atmospheres. Master's thesis, University of Applied Sciences Wien, 2011.

[WGH$^+$11] Rolf Westerteiger, Andreas Gerndt, Bernd Hamann, Christoph Garth, Ariane Middel, and Hans Hagen. Spherical terrain rendering using the hierarchical healpix grid. In *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering-Proceedings of IRTG 1131 Workshop 2011*, volume 27, pages 13–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011.