A GPU-Based Real Time Method for Simulating Atmospheric Light Scattering

Christoph Keller Christian-A. Bohn

FH Wedel
Feldstraße 143, 22880 Wedel, Germany

0171 2783475 +49(0)4103/8048-40
christoph.kel@gmail.com bo@fh-wedel.de

ABSTRACT

A method to display planetary atmospheres as seen from the inside is proposed. We developed this approach to improve the perceived realism in virtual reality applications with dynamic outdoor scenes. The algorithm uses the underlying physical model with as few simplifications as possible while producing visually convincing results in real time. Based on the effects of light scattering, the sky color is calculated entirely on the graphics processing unit by means of a vertex shader. Computationally costly terms, especially optical depth, are accurately computed without using precalculation. Thus, all relevant parameters (such as viewer height, atmosphere density and planet/atmosphere radius) can be adjusted interactively with immediate feedback. Furthermore, by sampling the light path non-equidistantly, a way of efficiently solving the scattering integral is introduced. The proposed method copes with every time of day situation for planets in single-solar systems and can be easily integrated and extended.

KEY WORDS

Atmospheric scattering, dynamic sky color, planet rendering, virtual reality, vertex shader

1 INTRODUCTION

Realistic image synthesis is a major field of research in computer graphics. Especially the effect of light interacting with materials is focussed on. In this paper, we present a fast, physically accurate real time approach to simulate light passing through an atmosphere. A realistic, dynamic sky simulation - even if in the background - can help to create convincing overall pictures, not to mention its contribution to the credibility of sky-oriented scenes (such as flight simulators). Although sophisticated methods for rendering realistic atmospheric effects exist, they all suffer from the same drawbacks: All previous approaches rely on precomputed lookup-tables and specially tuned algorithms, making it hard to integrate them in existing frameworks. Additionally, precalculated or hard-coded atmospheric conditions limit the algorithm's flexibility. Our idea is to develop a physically based approach for atmosphere rendering, which is flexible

and easy to implement, yet providing interactive frame rates. In the following, we explain basic terms and concepts.

Many planets are enveloped in a mixture of gases which is called atmosphere. This shell is bound to the surface due to gravity. The Earth's atmosphere essentially consists of oxygen (78 %) and nitrogen (21 %), the combination known as air. Another important element is the ozone layer, which absorbs ultraviolet rays. By providing oxygen, protecting us from dangerous radiation and regulating temperature our atmosphere establishes vital conditions.
With increasing altitude, the density of air falls exponentially. 5 km above sea level on Earth, it has decreased by 50 %, at an altitude of 12 km the drop is roughly 75 %. Although the transition between our atmosphere and space is fluent, the commonly used boundary lies at a height of 100 km and is called Kármán line.

While composition and size of atmosphere can vary strongly from planet to planet, sky color calculation - based upon physical laws - always follows the same rules. When a light wave enters the atmosphere, it is likely to collide with molecules and aerosols and change its direction. Some light is absorbed (for example by ozone or dust particles), some is scattered or reflected.
The interaction between radiation and small particles was described by Lord Rayleigh in 1871. Rayleigh found out that the amount of scattering is inversely proportional to the fourth power of the light's wavelength. Light with shorter wavelengths is scattered much more than light with longer wavelengths. This is the main reason why our sky appears blue at day time: Because of its high frequency, blue is scattered more than green and red light. Although violet has an even shorter wavelength, the sky color is blue mainly because our eyes are less sensitive to violet.
However, bigger particles (such as water vapour or dust particles) show a different scattering behaviour which was specified by Gustav Mie in 1908. Here, no strong wavelength dependency exists. For every color the scattering amount is nearly the same. Also, light is scattered most effectively into the forward direction. Due to aerosols and Mie scattering we see a white glare around the sun.

As discussed, our sky appears blue during day time because of atmospheric scattering - blue is spreaded over the sky more than any other color. Yet, approaching the horizon, sky color changes to a pale white. A viewing path through the atmosphere towards the horizon is much longer than a viewing path towards the zenith; accordingly light has to pass through more air. With increasing travel distance, more light is added and at the same time removed to a viewing path by scattering. As it collides with much more air molecules on a path close to the horizon, the inscattered blue light is scattered out again, whitening the overall color. The measurement of a path's transparency which accounts for this complex scattering behaviour is called optical depth.
The same phenomenon causes the impressive orange-red color of earthly sunrise/sunset. Because of the large optical depth, blue is removed from the white light coming directly from the sun, leaving yellow and red fractions.
The sky colors mentioned above primarily apply to Earth-like atmospheres. Generally, skies can take any color depending on atmospheric composition and incoming light properties. The sky on

Mars for instance is brownish red due to strongly scattering and absorbing dust. Still, as said initially the steps for computing sky color are identical for every atmospheric composition.

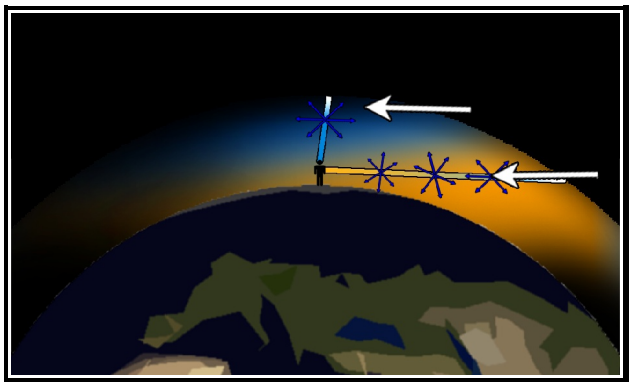

Figure 1: Simplified atmospheric scattering at sunrise/sunset. On a horizontal viewing ray, blue fractions of the white sunlight are attenuated by outscattering over the distance. This is the reason why the sky is reddened at the horizon.

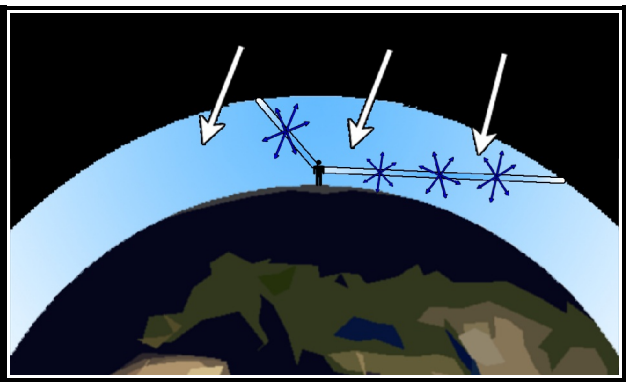

Figure 2: Simplified atmospheric scattering at day time. Close to the horizon, the inscattered blue is removed again by scattering, which results in a pale white. At viewing angles near the zenith the optical depth is small enough for blue to dominate the overall color.

2 RAYLEIGH SCATTERING

In 1871, John William Strutt, third Baron Rayleigh, published his theory on scattering of light by small particles *[Rayleigh1871]*. His model is valid for molecules smaller than 0.1 times the wavelength. Assuming particles to be isotropic (which is adequate for most atmospheric molecules), the total scattering coefficient β is given by

$$\beta(\lambda)=\frac{8\pi^3(n^2-1)^2}{3N\lambda^4} \tag{2.1}$$

where n is the refractive index of air, N is the molecular density and λ is the wavelength *[Preetham2003][Nishita1993]*. The total scattering coefficient specifies the amount of light removed by a scattering event. The most important fact about this equation is the strong dependency on the wavelength.
The Rayleigh phase function describes the scattering strength as to a certain direction:

$$\beta(\theta)=\frac{3}{16\pi}(1+\cos^2\theta) \tag{2.2}$$

with θ being the angle between the viewing path and the light direction.

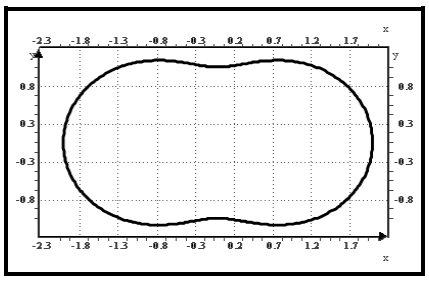

Figure 3: Rayleigh phase function. The light direction is parallel to the x-axis.

It is remarkable that the distribution has a symmetric shape and is strongest in the forward and backward direction. The product between the Rayleigh phase function and the total scattering coefficient results in the angular scattering coefficient, which describes the amount of light scattered in a specific direction:

$$\beta(\lambda,\theta)=\frac{\pi^2(n^2-1)^2}{2N\lambda^4}(1+\cos^2\theta) \tag{2.3}$$

By integrating this equation over the total solid angle 4π, the total scattering coefficient (2.1) can be derived.

As explained in section 1, Rayleigh scattering is responsible for our blue sky.

3 MIE SCATTERING

In the early 20th century, physicist Gustav Mie developed a generalization of Rayleigh's theory, which is applicable to particles of any size *[Mie1908]*. However, as it is more complex, in real time applications Mie scattering is only used for aerosols which are too big for Rayleigh scattering. Mie's total scattering coefficient is given by

$$\beta(\lambda)=0.434c\pi\left(\frac{2\pi}{\lambda}\right)^{v-2}K \tag{3.1}$$

where c is the concentration factor and correlates with the turbidity, v is the Judge's exponent and is commonly set to 4 for sky model and K is a factor that varies slightly with the wavelength *[Preetham2003]*. Note the weak wavelength dependency compared to Rayleigh scattering.
Because of its high complexity, the Mie phase function is usually approximated by applying the Henyey-Greenstein equation *[Henyey1941]*:

$$\beta_{HG}(\theta)=\frac{1-g^2}{4\pi(1-2g\cos\theta+g^2)^{\frac{3}{2}}} \tag{3.2}$$

with g defining the directionality. The function essentially describes an ellipse and has no mathematical relation to the original Mie phase function. Still, it can show the same

characteristics as Mie's function, which is mainly a strong focus on the forward direction (or backward, depending on g).

Because of aerosols in the atmosphere, according to Mie sun light is scattered with a strong directionality, which produces a white glare around the sun (see section 1). Mie scattering is also responsible for the greyish white sky on a misty or cloudy day – due to the high amount of vapour, which is an effective Mie scatterer, all colors are scattered nearly equally.

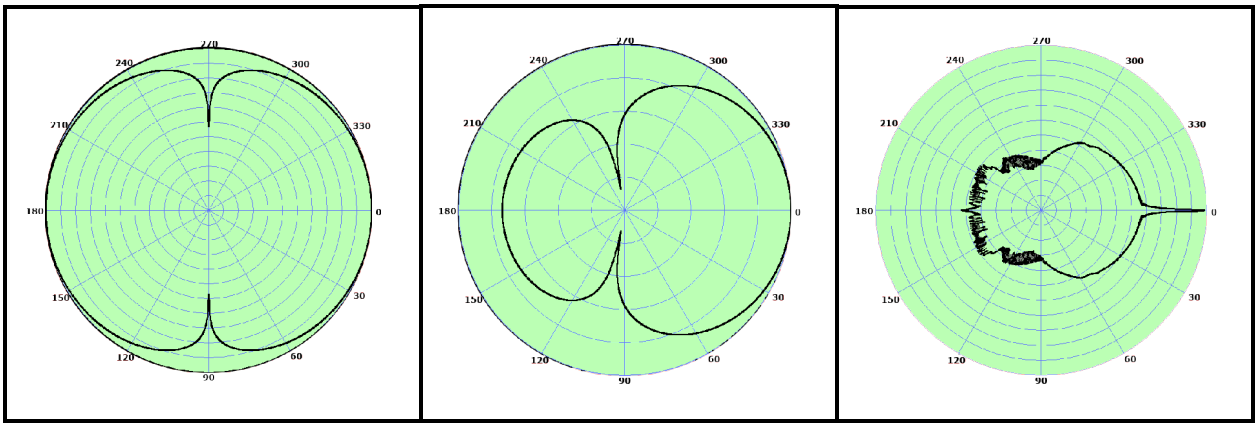

Figure 4: Mie phase function.
The left graph describes the scattering phase for particles with a radius of 10 nm. The function's shape resembles the shape of the Rayleigh phase function. In the second graph, the particle radius is 125 nm, in the third it is 50000 nm. With increasing particle size, the function shape becomes more acuate and develops an antenna like lobe in the forward direction.

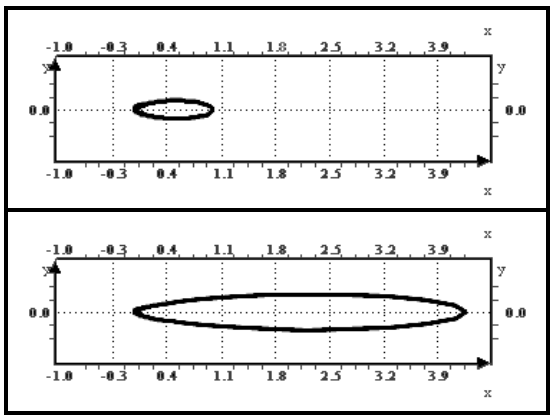

Figure 5: Henyey-Greenstein phase function.
In the top graph, the factor g is 0.8, in the lower it is 0.9. Note the stronger directionality in the bottom picture.

4 OPTICAL DEPTH

Optical depth is a measurement of opacity. It describes the extinction of light passing through a medium. The amount of light added to a path is defined as

$$dI = -\kappa \rho I_0 ds \tag{4.1}$$

where κ is the opacity coefficient which specifies the absorption or scattering efficiency, ρ is the materials density, I_0 is the initial light intensity and ds is the travelling distance. To find the

remaining intensity of a light path, it is necessary to integrate over the entire distance s:

$$\int \frac{dI}{I_0} = \int_0^s -\kappa\rho ds$$

$$\Leftrightarrow \quad \ln\frac{I}{I_0} = -\kappa\rho\int_0^s ds = -\kappa\rho s \quad \Leftrightarrow \quad \frac{I}{I_0} = \mathrm{e}^{-\kappa\rho s} \quad \Leftrightarrow \quad I = I_0\mathrm{e}^{-\kappa\rho s} = I_0\mathrm{e}^{-\tau} \qquad (4.2)$$

The variable τ is called optical depth and determines how much light is removed from a path.

5 SCATTERING EQUATION

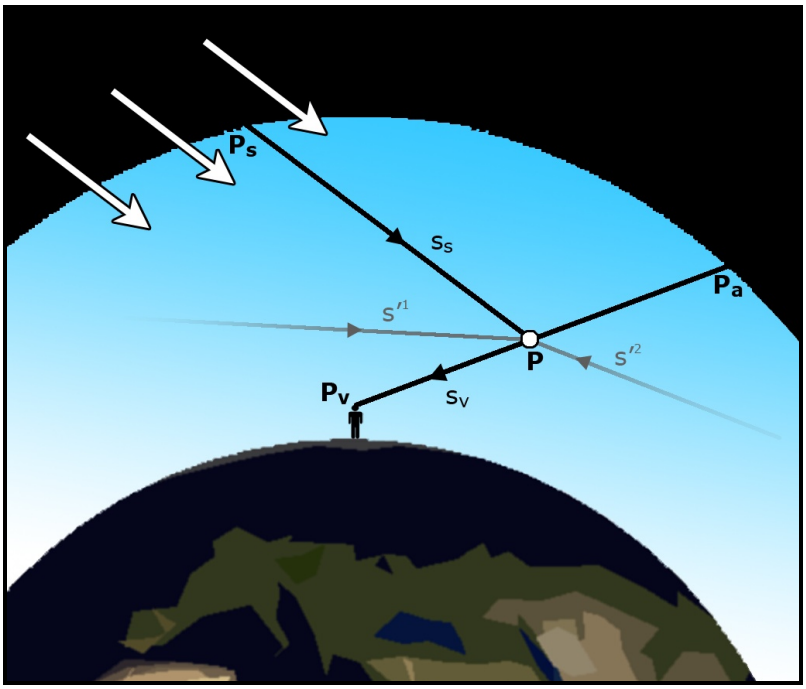

Figure 6: Schematic illustration of a scattering event in the atmosphere

Let us consider the mathematical formula calculating single scattering along a viewing path *[Nielsen2003][Preetham2003][Nishita1993]*.

$$I(\lambda) = \int_{P_v}^{P_a} I_{sun}(\lambda)\ \mathrm{e}^{(-\tau(\lambda, s_s))}\ \mathrm{in}(\lambda, s, \theta)\ \mathrm{e}^{(-\tau(\lambda, s_v))}\ ds$$

$$= I_{sun}(\lambda) \int_{P_v}^{P_a} \mathrm{in}(\lambda, s, \theta)\ \mathrm{e}^{(-\tau(\lambda, s_v)-\tau(\lambda, s_s))}\ ds \qquad (5.1)$$

Light coming from the sun is attenuated along its way through the atmosphere to the scattering point P. Therefore, the initial sun light intensity is multiplied by the extinction factor for the light's path s_s. τ is the optical depth (see equation (4.2)) and is defined as

$$\tau(\lambda, s) = \beta_R(\lambda) \int_0^s \rho_R(l)\, dl + \beta_M(\lambda) \int_0^s \rho_M(l)\, dl \tag{5.2}$$

where $\beta_R(\lambda)$ and $\beta_M(\lambda)$ are the total scattering coefficients for Rayleigh respectively Mie scattering (see equations (2.1) and (3.1)), $\rho_R(l)$ is the molecular density and $\rho_M(l)$ is the density of aerosols. In a nutshell, τ sums up the outscattered light at every point of a path.
The residual intensity is scattered at P into the viewer's direction by this function:

$$\text{in}(\lambda, s, \theta) = \beta_R(\lambda)\, \beta_R(\theta)\, \rho_R(s) + \beta_M(\lambda)\, \beta_M(\theta)\, \rho_M(s) \tag{5.3}$$

with $\beta_R(\lambda)\beta_R(\theta)$ and $\beta_M(\lambda)\beta_M(\theta)$ being the angular scattering coefficient for Rayleigh and Mie scattering and $\rho_R(s)$ and $\rho_M(s)$ being the density of Rayleigh and Mie scatterers.
This inscattered light is again attenuated on its way to the viewer s_v, so it is multiplied by another extinction factor.

As light is scattered at every point of the viewing path $\overline{P_v P_a}$ (with P_a being the intersection point of the viewing ray with the atmosphere boundary), to compute the total intensity integration along the path is necessary. Also, to cover the full spectrum of visible light, another integration over all visible colors must be taken:

$$I = \int_{\lambda_{low}}^{\lambda_{high}} I(\lambda) \tag{5.4}$$

where typically λ_{low} lies at 380 nm and λ_{high} of 750 nm.

Because the final equation contains a double (or even triple) nested integral, it can not be solved analytically; a numerical solution is required.

The calculation method presented above only takes single scattering into account. In reality, light is scattered more than once in the atmosphere, producing very complex circumstances. In figure 6, second order scattering is illustrated by incoming light from s'^1 and s'^2. To compute second order scattering at a point on the viewing path, integration over the total solid angle 4π is required, because inscattered light from all possible directions must be considered.

$$I'(\lambda) = \int_{P_v}^{P_a} \left(e^{-\tau(\lambda, s_v)} \left(\int_0^{4\pi} I(\lambda)\ \text{in}(\lambda, s, \theta')\ ds' \right) ds \right) \tag{5.5}$$

Furthermore, light reflected by the planet's surface plays a role in sky light computation. Although the effects of multiple scattering are not negligible in some situations, they are ignored

in this work for simplicity and performance reasons.

6 IMPLEMENTATION

Full single scattering (based on Nishita's equations) is implemented on the GPU using a vertex shader. This approach has several advantages: Expensive computations like the e- or the square root function are performed very efficiently on graphics hardware. Also, the CPU is left unaffected by costly scattering calculations. In addition, on older graphics hardware the probability of creating a bottleneck is low, since here the vertex shader unit is commonly underutilized and idle.
The input sky mesh can be any geometry centered at the origin; in practice it will be a dome (for optimal performance we advise an adaptive dome mesh which is highly tesselated around the sun). Instead of covering the entire spectrum of colors (see *[Irwin1996]*), three representative samples are taken for red, green and blue light, which the final RGB-color is composed of. Also, the sun direction is considered parallel for all points of the atmosphere.

For numerically solving the scattering integral a special midpoint rule – similar to an approach by *[Nishita1993]* - is applied in order to reduce sample count. To understand its efficiency, the distribution of atmospheric particles and thus atmospheric density must be understood. Atmospheric pressure at a given height is calculated by

$$\rho(height) = \rho_0 e^{(-height/scaleHeight)} \qquad (6.1)$$

where ρ_0 is the pressure at ground level. At the scale height, atmospheric pressure has decreased by the factor e.
Usually the atmospheric density is highest around the point of view, decreasing rapidly along the viewing ray. The area near the point of view is hence most significant for the final scattering result; here, a precise computation is necessary. To benefit from this observation, sampling points are concentrated at the beginning of the viewing ray instead of being distributed equidistantly. This is achieved by exponentiating the sampling point parameter.
Every path through the atmosphere is integrated that way, because very often it will start in thicker regions of the atmosphere, with density getting exponentially thinner along the path.

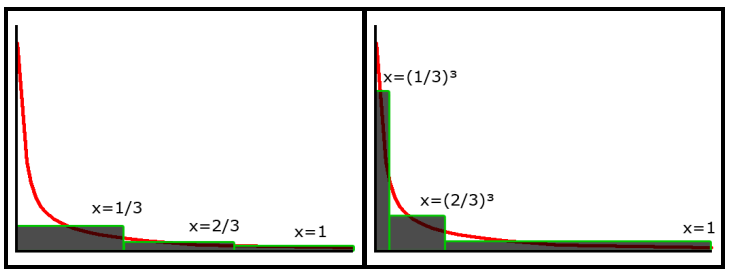

Figure 7: Efficient numerical solution of $1/e^x$ integrals. In the right graph the function area is approximated very accurately with few samples.

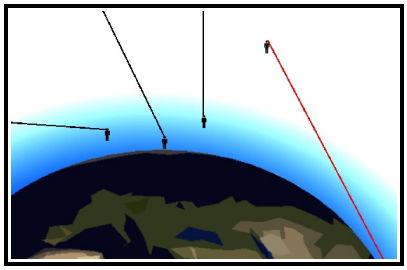

Figure 8: Atmospheric density distribution. On almost every view path towards the sky the density decreases quickly.

The actual implementation of the proposed method is presented in simplified pseudo-code and explained piecewisely in the following.

```
CalculateColor(viewDir)
{
        viewPos         =       vector3(0.0f, planetRadius + viewHeight, 0.0f);
        d               =       GetAtmosphereIntersection(viewPos, view) - viewPos;
        if (!IntersectsPlanet(viewPos, viewDir))
        {
                for (i = 0 to samples - 1)
                {
                        a       =       (i / samples)^3;
                        b       =       ((i + 1) / samples)^3;
                        scatteringPointA        =       viewPos + a * d;
                        scatteringPoint         =       viewPos + (a + (b - a) * 0.5f) * d;
                        scatteringPointB        =       viewPos + b * d;
                        if (!IntersectsPlanet(scatteringPoint, -sunDir))
                        {
                                deltaX          =       length( scatteringPointB - scatteringPointA);
                                finalColor.r    +=      deltaX * GetScatteredIntensity(redWaveLength,
                                                        viewPos, view, scatteringPoint);
                                // repeat for finalColor.g and finalColor.b
                        }
                }
                finalColor      *=      sunColor * absorptionColor;
                finalColor      =       1.0f - exp(-exposure * finalColor);
                return vector4(finalColor, (finalColor.r + finalColor.g + finalColor.b)^
                atmosphereTransparency);
        }
        else
                return BLACK;
}
```

[1] Solves the single scattering integral (see equation (5.1)). Sky color is computed for a specific viewing direction (which is the normalized input vertex). Before this function is executed, a frustum clipping test is performed in the shader to avoid unnecessary calculations.

[2] Check if the sky is visible in viewing direction.

[3] Numerically integrate the scattering function over the viewing path. As described, march the viewing path non-equidistantly. Thus, only about 50 % of the normally required samples are sufficient. Skip the calculation if the sample point lies in the planet's shadow. Finally, multiply the solved integral by the initial sun intensity and an absorption coefficient. Absorption by particles is not considered in a physically correct way because it is often negligible. Instead a constant absorption on every light path is assumed. Then the sky brightness curve is exponentially adjusted to capture the high range of brightness and match human vision. Also, depending on the sky color's intensity transparency is defined, so the mesh can easily be blended with a star background.

```
GetScatteredIntensity(lambda, viewPos, viewDir, scatteringPoint)
{
        inScattering    =       atmosphereTurbidity * (rayleighAngularCoefficient * moleculeDensity +
                                mieAngularCoefficient * aerosolDensity);
        for (i = 0 to densitySamples - 1)
        {
                a                       =       (i / densitySamples)^2;
                b                       =       ((i + 1) / densitySamples)^2;

                densitySampleA          =       viewPos + a * viewIntegrationPath;
                densitySample           =       scatteringPoint + (a + (b - a) * 0.5f) * viewIntegrationPath;
                densitySampleB          =       viewPos + b * viewIntegrationPath;
                opticalLength           =       atmosphereTurbidity * length(densitySampleB -
                                                densitySampleA);
                viewOpticalDepth        +=      opticalLength * (rayleighCoefficient * moleculeDensity +
                                                mieCoefficient * aerosolDensity;

                densitySampleA          =       viewPos + a * sunIntegrationPath;
                densitySample           =       scatteringPoint + (a + (b - a) * 0.5f) * sunIntegrationPath;
                densitySampleB          =       viewPos + b * sunIntegrationPath;
                opticalLength           =       atmosphereTurbidity * length(densitySampleB -
                                                densitySampleA);
                sunOpticalDepth         +=      opticalLength * (rayleighCoefficient * moleculeDensity +
                                                mieCoefficient * aerosolDensity);
        }
        outScattering   =       exp(-viewOpticalDepth – sunOpticalDepth);
        return inScattering * outScattering;
}
```

[1] Calculates the intensity arriving at viewPos after a single scattering event at scatteringPoint.

[2] Compute inscattering (see equation (5.3)). All physical terms are reduced to their essential characteristics. The use of actual units and constants is dispensable, because the final intensity value must be mapped to a displayable range anyway. Our modified terms are listed and explained in table 1.

[3] Solve view and sun optical depth integrals to get outscattering and ultimately attenuation (see equations (4.2), (5.1), (5.2)). Again, the sampling point distance is incremental.

Description	Term	Equation	Comment
Particle density	$\rho_0 e^{(-height\,/\,scaleHeight)}$	(6.1)	-
Rayleigh phase function	$1+0.5cosine^2$	(2.2)	Slightly adjusted for weaker directionality.
Total Rayleigh coefficient	$\lambda_{RED}^4 / \lambda^4$	(2.1)	Factor concentrating on the wavelength dependency, all other units and constants have been eliminated.
Mie phase function	$\frac{1-g^2}{(1-2gcosine+g^2)^{1.5}}$	(3.2)	No division by 4π.
Total Mie coefficient	$\lambda_{RED}^2 / \lambda^2$	(3.1)	See total Rayleigh coefficient.

Table 1: Modified physical terms

7 RESULTS

We have shown that our approach produces visually satisfying results especially for clear Earth-like atmospheres at interactive frame-rates. Using three or four integration samples (which is utterly sufficient thanks to our efficient way of integration), we hardly noticed any performance loss compared to scenes with static atmospheric environments. Even with very accurate settings (five samples for the scattering integral and also five samples for the optical depth integrals), we still achieved 60 frames per second in our test environment (Dual Core AMD Opteron, 2.19 GHz, 2 GB RAM, NVIDIA Quadro FX 4400 at 1600x1200). The implementation on graphics hardware has proven to be extremely efficient - the GPU program was 80-90 times faster than the same algorithm implemented on the CPU.
Also, tests with probands in our VR environment have shown that the application of a realistic and dynamic model of the atmosphere increases the overall acceptance and thus the degree of immersion.

8 DISCUSSION

With physically correct image synthesis becoming more and more important, atmospheric scattering has been discussed copiously over the last few years. Many previous approaches concentrate on Earth's atmosphere, and considerable algorithmic optimizations can be implemented when a relatively static environment like the Earth sky is to be computed. All proposed real time approaches use precomputation and thus are limited with respect to flexibility. Although previous approaches are applicable in many cases, there are applications which require a highly dynamic variability of atmospheric conditions in real time, such as flight/ space simulators, planet rendering engines, computer games, semi-scientific projects and virtual reality applications. Since parameters like viewer height, atmospheric pollution and particle density distributions are adjustable at run time without additional effort, the proposed method is well suited for these areas of application. Furthermore we have proven that a direct implementation of the full scattering equation is possible and performant, rebutting the argumentation of previous papers claiming that precomputation is inevitable for real time applications.

However, our approach merely provides the basis for a universal atmosphere rendering system. Sky color currently is not calculated globally (which would involve multiple scattering), prohibiting effects like twilight or completely overcast skies which are based upon higher order scattering and self illumination. Moreover, several local illumination features are not modelled realistically yet. The planet Mars for example has an extremely thin atmosphere; yet, light scattering is higher compared to our atmosphere due to the dust being present in Martian atmosphere. Red-brown iron oxides in the atmosphere also absorb much blue light. Within our approach, custom particles with individual scattering or absorption behaviour are not included so far; the atmosphere composition is broken down to one type of molecules and one type of aerosols. In the proposed method, atmospheric density and scattering amount depend on each

other, requiring molecular density to be increased in order to achieve stronger scattering. Consequently, although the Martian and many other planet's atmospheres can be visually simulated with our method, the input values need to be tweaked, leaving the path of physical correctness.

Many of these drawbacks can be eliminated as soon as high-performance lookup tables in vertex programs will be available, which is currently not the case with vertex texture fetch. Some expensive computations like the e-function, the root function or Rayleigh and Mie terms can be precalculated without granting overall flexibility, offering more frame time to spend on complex calculations. Nevertheless, section 7 shows that our system already provides good results.

9 SUMMARY

The presented method offers a good balance between visual and physical realism at a high rendering performance while providing a high amount of dynamic and flexibility. The approach of non-equidistantly sampling light paths and the possibilities of modern graphics hardware help to efficiently implement the full single scattering equation with very little compromises, which has not been done in real time so far. Our technique is very easy to integrate by simply applying the shader to a geometry. Although it can not capture all scattering effects present in nature, the proposed method realistically simulates the most significant aspects of atmospheric scattering and is already applicable in many scenarios.

10 FUTURE WORK

Since the developed shader is easy to implement and modify, it provides the foundation for many enhancements. Some potential future improvements are listed below:

- develop a realistic cloud and weather model
- provide a realistic night sky including stars and moon(s)
- properly consider absorbing particles in the atmosphere and account for absorption in the scattering integral (parallel to molecules and aerosols)
- include multiple scattering and self illumination
- use lookup tables (as discussed in section 8)
- implement aerial perspective
- integrate real high dynamic range rendering
- allow the view point to be located outside of the atmosphere
- define more than one light source
- enhance the visual impression by rendering lens flares etc. (see figure 9, 13, 14)

11 PICTURES

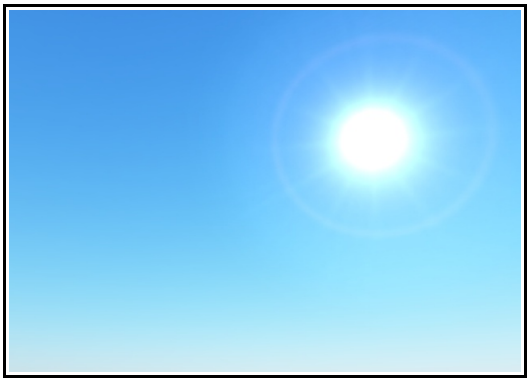

Figure 9: Earth sky at day time

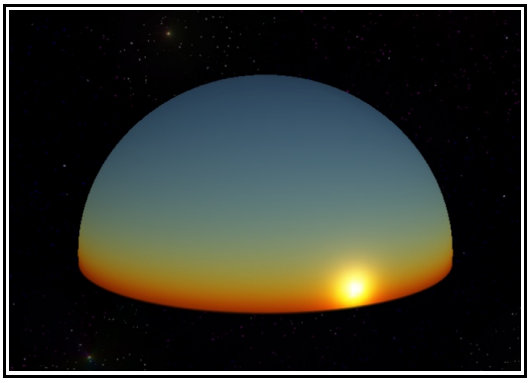

Figure 10: Sky dome from the outside at sunset

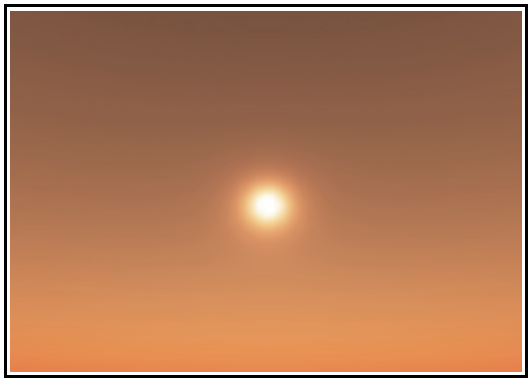

Figure 11: Mars-like atmosphere

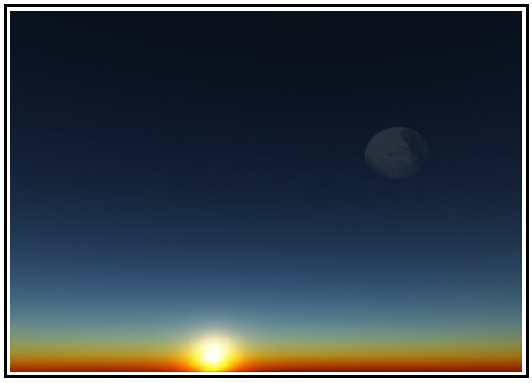

Figure 12: Sunrise at 12 km height

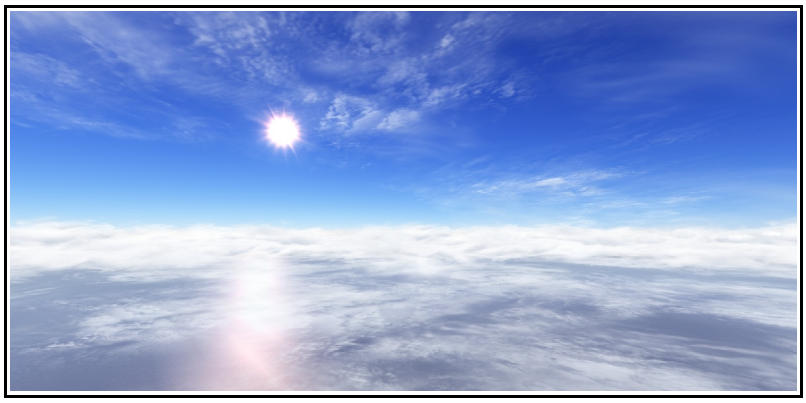

Figure 13: Integration in VR environment 1

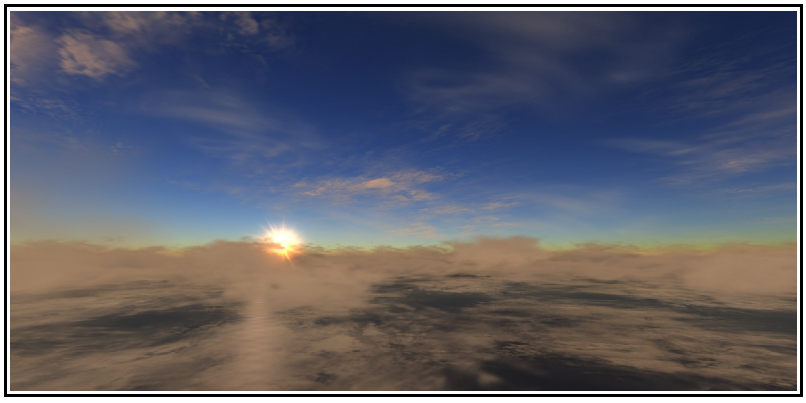

Figure 14: Integration in VR environment 2

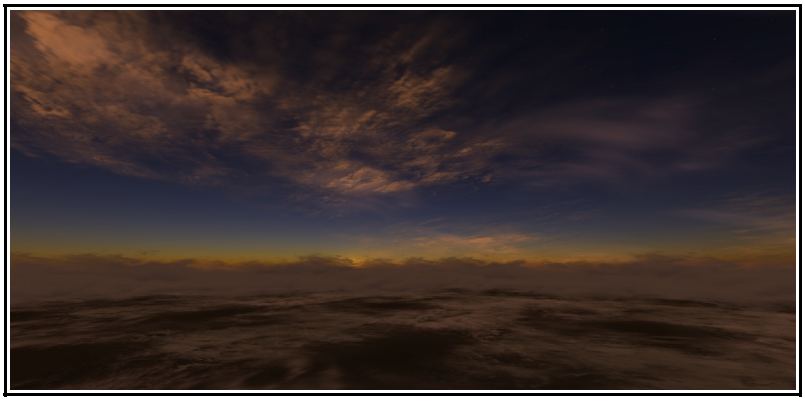

Figure 15: Integration in VR environment 3

REFERENCES

[Dobashi2002] Y. Dobashi, T. Nishita , T. Yamamoto , *Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware*, Proc. Graphics Hardware 2002, pp. 99-108, 2002.

[Henyey1941] L. G. Henyey and J. L. Greenstein. *Diffuse Reflection in the Galaxy.* Astrophys. J. 93, 70, 1941.

[Hoffman2002] Hoffman, N., Preetham, A.J. *Rendering Outdoor Light Scattering in Real Time.* Game Developers Conference, 2002.

[Irwin1996] John Irwin. *Full-spectral Rendering of the Earth's Atmosphere Using a Physical Model of Rayleigh Scattering.* In Proceedings of the 1996 Eurographics UK Conference.

[Mie1908] G. Mie. *Beiträge zur Optik trüber Medien, speziell kolloidaler Metallösungen.* Annalen der Physik 25(3): pp. 377-445, 1908.

[Nielsen2003] R.S. Nielsen. *Real Time Rendering of Atmospheric Scattering Effects for Flight Simulators.* Master's thesis, Technical University of Denmark, DTU, 2003.

[Nishita1993] T. Nishita, T. Sirai, K.Tadamura, E. Nakamae. *Display of the Earth Taking into Account Atmospheric Scattering.* ACM SIGGRAPH 1993, pp.175-182, 1993.

[ONeil2005] S. O'Neil, *Accurate Atmospheric Scattering.* GPU Gems 2, 2005.

[Preetham2003] A.J. Preetham. *Modeling Skylight and Aerial Perspective*. ATI Research, ACM SIGGRAPH 2003, 2003

[Rayleigh1871] J. W. Strutt (Lord Rayleigh). *On the light from the sky, its polarization and colour.* Philos. Mag. 41: pp. 107-120, 274-279, April 1871.

[Schafhitzel2007] T. Schafhitzel, M. Falk, T. Ertl. *Real-Time Rendering of Planets with Atmospheres.* In Journal of WSCG, 2007.

[Wenzel2002] C. Wenzel, *Real-time Atmospheric Effects in Games Revisited.* Game Developers Conference, 2002.