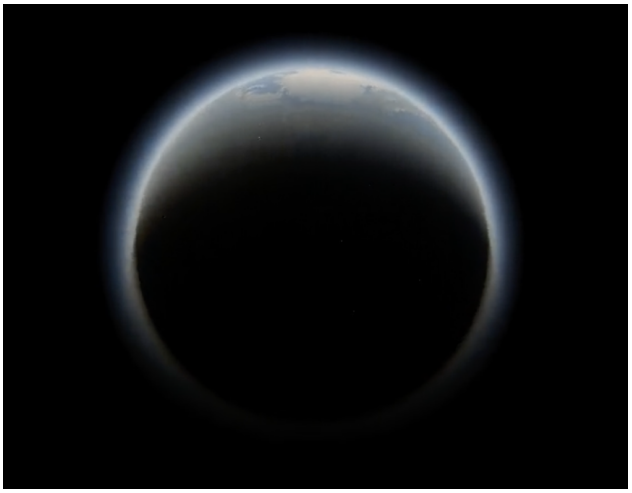
Implementation of Atmospheric Scattering

CS 284A Final Project Writeup

Wayne Chu

March 2026



1 Introduction

Atmospheric scattering [2, 5] plays an important role in determining the appearance of the sky. When sunlight interacts with molecules and particles in the atmosphere, it is scattered in different directions. Therefore, phenomena such as the blue sky and haze around the sun can be produced.

In this final project, I implemented a simple atmospheric scattering model using Rayleigh scattering for small molecules and Mie scattering for larger particles. They make it possible for us to simulate the directional scattering of light and reproduce key visual effects of the Earth's atmosphere or other planet's atmosphere.

2 Preliminary

2.1 Rayleigh Scattering

Rayleigh scattering is caused by molecules in the atmosphere, such as oxygen and nitrogen. It is sensitive to the light with shorter wavelengths. This is the main reason why the sky is blue. To implement Rayleigh scattering, we have to first derive Rayleigh Phase [3] in terms of the angle between the incident light direction and the viewing direction.

For Rayleigh scattering, the angular dependence is proportional to

$$1 + \cos^2 \theta.$$

We therefore assume the phase function has the form

$$P_R(\theta) = k\left(1 + \cos^2\theta\right),$$

where k is a normalization constant.

To make $P_R(\theta)$ a properly normalized phase function over the sphere, we require

$$\int_\Omega P_R(\theta)\, d\Omega = 1.$$

Using spherical coordinates,

$$d\Omega = \sin\theta\, d\theta\, d\phi,$$

so

$$\int_\Omega P_R(\theta)\, d\Omega = \int_0^{2\pi}\int_0^{\pi} k(1+\cos^2\theta)\sin\theta\, d\theta\, d\phi.$$

First integrate over ϕ:

$$= 2\pi k \int_0^{\pi} (1+\cos^2\theta)\sin\theta\, d\theta.$$

Let

$$u = \cos\theta, \qquad du = -\sin\theta\, d\theta.$$

When $\theta = 0$, $u = 1$; when $\theta = \pi$, $u = -1$. Thus,

$$\int_0^{\pi} (1+\cos^2\theta)\sin\theta\, d\theta = \int_1^{-1} (1+u^2)(-du) = \int_{-1}^{1} (1+u^2)\, du.$$

Now compute the integral:

$$\int_{-1}^{1} (1+u^2)\, du = \left[u + \frac{u^3}{3}\right]_{-1}^{1} = \left(1+\frac{1}{3}\right) - \left(-1-\frac{1}{3}\right) = \frac{8}{3}.$$

Therefore,

$$\int_\Omega P_R(\theta)\, d\Omega = 2\pi k \cdot \frac{8}{3}.$$

Imposing the normalization condition,

$$2\pi k \cdot \frac{8}{3} = 1.$$

Solving for k,

$$k = \frac{3}{16\pi}.$$

Hence the normalized Rayleigh phase function is

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

2.2 Mie Scattering

Larger atmospheric particles (aerosols, dust, haze, and small water droplets) give rise to Mie scattering. Mie scattering is less selective for incident wavelengths than Rayleigh scattering and tends to scatter almost exclusively in the forward direction. The result is that, on hazy days, the sky will have a light and bright appearance surrounding the sun.

Mie scattering is achieved theoretically by numerically solving Maxwell's equations for a sphere, resulting in an equation that depends on the size, refractive index, and wavelength of the sphere, resulting in an infinitely large sum. Since this full solution is too expensive for real-time rendering, I use the Henyey–Greenstein [1] phase function as a practical approximation.

Let θ denote the angle between the incident light direction and the viewing direction. The Henyey–Greenstein phase function is defined as

$$P_{HG}(\theta) = \frac{1 - g^2}{4\pi \left(1 + g^2 - 2g\cos\theta\right)^{3/2}},$$

where g is the asymmetry parameter.

The parameter g controls the preferred scattering direction:

$$g = \langle \cos\theta \rangle.$$

Its physical meaning can be interpreted as follows:

$$-1 \leq g \leq 1.$$

In particular,

$$g = 0 \quad \Rightarrow \quad P_{HG}(\theta) = \frac{1}{4\pi},$$

which corresponds to isotropic scattering,

$$g > 0$$

produces predominantly forward scattering, and

$$g < 0$$

produces predominantly backward scattering.

For atmospheric haze and aerosol rendering, we usually choose a positive value of g, since these particles scatter most strongly in the forward direction. As g approaches 1, the phase function becomes increasingly peaked around $\theta = 0$, which matches the visual appearance of Mie-like scattering.

Therefore, in the implementation, the Mie phase function is approximated as

$$P_M(\theta) = \frac{1 - g^2}{4\pi \left(1 + g^2 - 2g\cos\theta\right)^{3/2}}.$$

In shader code, I evaluate it using $\cos\theta$ directly:

$$c = \cos\theta,$$

so the expression becomes

$$P_M(c) = \frac{1 - g^2}{4\pi \left(1 + g^2 - 2gc\right)^{3/2}}.$$

3 Implementation Overview

I implement several functions that estimate light transmittance and scattering contributions along a ray to simulate how light propagates through the atmosphere. These functions model how incident light is attenuated and redistributed when it travels through the atmosphere.

Height-Dependent Density Profile In the implementation, the Rayleigh and Mie terms are given different density profiles depending on height. Let

$$h = \frac{r - R_p}{R_a - R_p},$$

where r is the distance from the sample point to the center of the planet, R_p is the radius of the planet, and R_a is the outer radius of the atmospheric shell. The height h is normalized to the interval $[0, 1]$.

The Rayleigh density is modeled with exponential falloff as

$$\rho_R(h) = e^{-k_R h},$$

with k_R governing the rate of decay of molecular density with height.

Similarly, the Mie density is modeled with exponential falloff, but biased toward the lower atmosphere:

$$\rho_M(h) = e^{-k_M h}(1-h)^3,$$

with k_M governing the aerosol falloff rate. The extra term in the Mie density function causes the density of the Mie component to be stronger near the planet's surface, creating thicker haze near the horizon.

Single Scattering Model To calculate the scattering along a given viewing ray, I use an approximation of single scattering contributions from both Rayleigh and Mie particles. I estimate how much light is scattered towards the camera at a given point along a ray by computing

$$L_{scat} = \beta_R \, \rho_R \, P_R(\theta) + \beta_M \, \rho_M \, P_M(\theta),$$

where ρ_R and ρ_M are the local Rayleigh and Mie densities, P_R and P_M are the corresponding phase functions, and β_R and β_M denote the wavelength-dependent Rayleigh and Mie scattering coefficients used in the shader. In the implementation, the coefficients are represented by per-channel scattering colors scaled by user-controlled Rayleigh and Mie strengths. As a result, the scattered radiance is computed per color channel, instead of a single scalar value.

The angle θ is defined as the angle between the light direction and the viewing ray direction. We compute

$$\cos\theta = \omega_{light} \cdot \omega_{view}.$$

The scattered light is further reduced by the atmospheric transmittance along the path of the light and the path of the view. Hence, the additional component that is added to the ray is given by

$$L = T_{view} \, T_{light} \, L_{scat},$$

where T_{light} is the transmittance from the sample point to the light source and T_{view} is the total transmittance from the camera to the sample point.

Beer–Lambert Extinction Light attenuation in the atmosphere is given by the Beer-Lambert law [4]. During the passage of light through the atmosphere, the amount of scattering and absorption is given by

$$T = e^{-\tau},$$

where T is the transmittance and τ is the accumulated optical depth along the ray.

The optical depth is computed by integrating the extinction coefficient over the path of the ray,

$$\tau = \int \beta_{ext}(s) \, ds,$$

where β_{ext} represents the total extinction coefficient, including the effects of Rayleigh scattering, Mie scattering, and atmospheric absorption.

In the implementation, the integral is computed by ray marching through the atmosphere, and the computed optical depth is used to compute the transmittance by applying the Beer-Lambert law.

Atmospheric Transmittance The function `getAtmosphereLightTransmittance` calculates how much light is left after traveling from the light source to a sample point. If there is no occlusion to the straight path of light, it is intersected with the outer shell of the atmosphere. Then, the optical depth is integrated over this part of the path. The extinction is then converted to transmittance using the Beer-Lambert law. If the planet blocks the straight path, the algorithm falls back to an approximation that estimates refracted light near the horizon.

Algorithm 1 `getAtmosphereLightTransmittance`

Cast a ray from sample position $\mathbf{x}$ toward light direction ω_s
if the ray does not intersect the atmospheric shell **then**
    return 1
end if
Compute the valid segment $[t_{enter}, t_{exit}]$ inside the atmosphere
if the planet blocks the straight light path **then**
    return `getApproximateRefractedLightTransmittance`$(\mathbf{x}, \omega_s)$
end if
$\tau \leftarrow 0$
for each integration sample between t_{enter} and t_{exit} **do**
    Evaluate local Rayleigh, Mie, and absorption extinction
    Accumulate optical depth: $\tau \leftarrow \tau + \beta_{ext}\Delta s$
end for
return $\exp(-\tau)$

Approximate Refracted Light The function `getApproximateRefractedLightTransmittance` gives a cheap approximation of refracted sunlight near the planetary limb. Rather than computing a refracted ray path as a curve through the atmosphere, we estimate how much light might skim around the planet.

Firstly, I verify if the light ray crosses behind the planet from the sample's point of view. If it does, I then find the depth to which the ray crosses the silhouette of the planet. This value is then used to locate a virtual tangent path to the atmosphere, which is a likely refracted trajectory around the planet. The optical depth is integrated along this virtual path and then converted to transmittance.

Algorithm 2 `getApproximateRefractedLightTransmittance`

Cast a ray from $\mathbf{x}$ along ω_s
if the ray does not pass behind the planetary disk **then**
    return 0
end if
Compute silhouette penetration depth d
Map d to a virtual tangent altitude h_t
Construct a virtual grazing path through the atmosphere using h_t
$\tau \leftarrow 0$
for each integration sample along the virtual path **do**
    Evaluate atmospheric extinction σ_t
    $\tau \leftarrow \tau + \sigma_t \Delta s$
end for
return $\exp(-\tau)$

Approximate Multiple Scattering I also provide a heuristic approximation to weak multiple scattering using the function `getApproximateMultipleScattering`. Instead of tracing additional bounce paths, it adds a soft ambient contribution when the following are true:

- There is sufficient Rayleigh or Mie density present in the local marching step to scatter the light.
- There is some degree of incident light already attenuated, which suggests a transfer of energy within the medium.
- The sample is lower in the atmosphere where multiple scattering is stronger.

This approximation uses empirically tuned coefficients instead of a physically derived multiple scattering model and enables the renderer to produce the visual appearance of atmospheric glow at a relatively low computational cost.

Algorithm 3 Approximate Multiple Scattering

Evaluate local Rayleigh and Mie densities
if local scattering density is too low **then**
    return 0
end if
Evaluate attenuated incident light
if incident light attenuation is too small **then**
    return 0
end if
Compute a height-dependent weight that increases toward the lower atmosphere
Combine density, attenuated light, and height using empirically chosen coefficients
Form a soft ambient multiple-scattering term L_{ms}
return L_{ms}

Filtering Light Through Atmospheres Finally, the function `filterLightThroughAtmospheres` assesses how a light ray is filtered through all atmosphere-enabled spherical media. For each atmosphere, the algorithm checks whether the ray crosses its atmospheric shell. If it is blocked by the planet, the refracted light approximation is used. Otherwise, optical depth is integrated along its valid portion and converted to Beer-Lambert transmittance. All atmospheres' contributions are summed to determine final light attenuation.

References

[1] Ocean Optics Web Book. *The Henyey-Greenstein Phase Function*. Accessed: 2026-03-14. 2023. URL: https://www.oceanopticsbook.info/view/scattering/level-2/the-henyey-greenstein-phase-function.

[2] Ken Nishita et al. *Accurate Atmospheric Scattering*. https://developer.nvidia.com/gpugems/gpugems2/part-ii-shading-lighting-and-shadows/chapter-16-accurate-atmospheric-scattering. GPU Gems 2, Chapter 16. 2005.

[3] Planetary Data System Atmospheres Node. *Rayleigh Scattering Phase Function*. https://pds-atmospheres.nmsu.edu/education_and_outreach/encyclopedia/rayleigh_phase.htm. Accessed: 2026-03-14. 2025.

[4] Wikipedia contributors. *Beer–Lambert law*. Accessed: 2026-03-14. 2026. URL: https://en.wikipedia.org/wiki/Beer%E2%80%93Lambert_law.

[5] Wikipedia contributors. *Scattering — Wikipedia, The Free Encyclopedia*. Accessed: 2026-03-16. 2026. URL: https://en.wikipedia.org/wiki/Scattering.