# A visual and performance comparison of atmospheric scattering models

November 1, 2025

*Author*
Dimas Leenman
(0513502)

*Study Programme*
master Computing Science
Utrecht University

*First Supervisor*
Peter Vangorp

*Second Supervisor*
Alex Telea

# Contents

# 1. Abstract

Previous models of atmospheric scattering have either utilized an analytical fitted function, limiting the viewer to the ground, or a number of lookup tables improve runtime performance.

This work introduces 3 new models, one neural network based model that supports both ground and space views, one analytical model that does not require fitting on reference data, and one model using an approximation of the transmittance tables.

The new models are compared to the existing models, as well as a path traced reference, on visual accuracy, runtime performance, and implementation complexity.

# 2. Introduction

As games become increasingly more realistic, more accurate visual effects are needed to render them. In games such as Microsoft Flight Simulator (*Microsoft Flight Simulator 2024*, n.d.), DCS (*Digital Combat Simulator World*, n.d.), and the Scatterer mod for Kerbal Space Program (blackrack, n.d.) the atmosphere is an important element to get right. Other games, such as Gran Turismo 7 (Suzuki & Yasutomi, 2023) make use of a complex sky simulation to ensure both the sky itself and resulting lighting looks correct. This work introduces 3 new models of atmospheric scattering that attempt to improve over existing methods.

Figure 1: Microsoft Flight Simulator, Scatterer and Gran Turismo 7

# 3. Atmospheric scattering

All particles in the atmosphere of the earth affect how light is scattered inside it. When a photon hits a particle, it can either be scattered into a new direction, or absorbed. Which of these events happen, depends on the wavelength of the photon, as well as the type and size of the particle. For every photon, these events can happen 0 times, 1 time, referred to as single scattering, or more than once referred to as multiple scattering.

Before going over previous work, it may prove useful to provide a standard set of symbols, coordinates, as well as a short introduction on the physics behind atmospheric scattering.

## 3.1. Coordinate system

The figure below shows the coordinates used for a viewer $v$ at height $h$ in an atmosphere, with light coming from light source $l$. The coordinate system for this is described below.

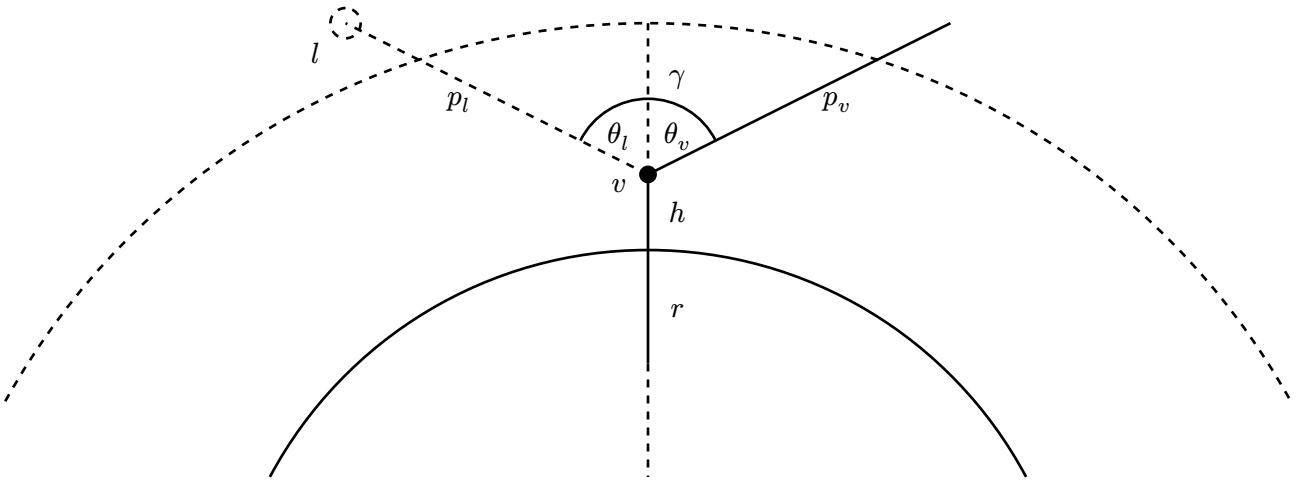



Figure 2: Viewer $v$ in an atmosphere.

| | |
|---|---|
| Radius $r$ | The radius of the planet |
| Height $h$ | The height viewer $v$ is from the surface |
| View zenith $\theta_v$ | The angle between the surface normal, and view direction |
| Sun zenith $\theta_l$ | The angle between the surface normal, and the light direction |
| Angle $\gamma$ | The angle between the view direction, and light direction |

Table 1: Coordinates

## 3.2. Physics of scattering

This section provides a basic overview of the physics behind a photon scattering in an atmosphere. A possible path of a photon scattering from light source $l$ towards viewer $v$ can be seen below.

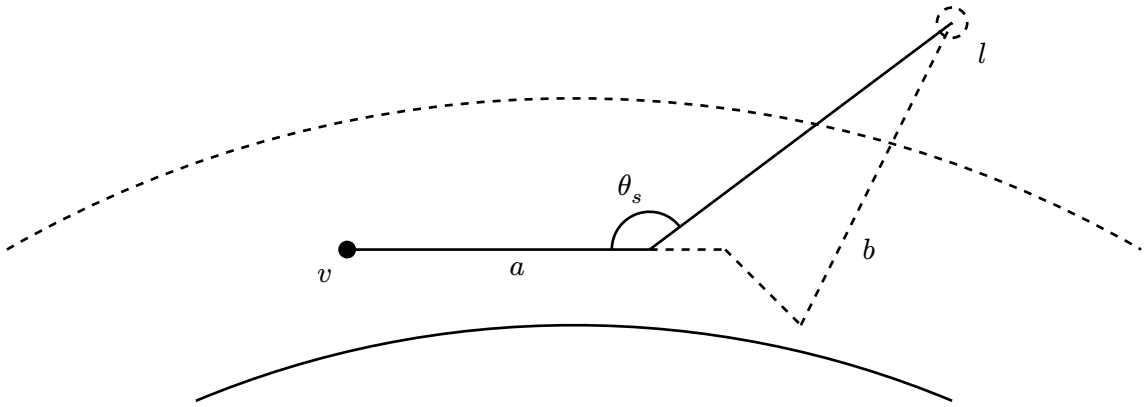



Figure 3: Viewer $v$ in an atmosphere.

Here, path $a$ represents a path with only one scatter event, where the photon scatters at an angle $\theta_s$. Path $b$ represents a path with multiple scattering events.

When light travels through the atmosphere, it is attenuated based on the density of the atmosphere it travels through. Given a density function $\rho(h)$ that depends on height $h$ from the surface, the *optical depth* $\tau$ from viewer $v$, with direction $\theta_v$ can then be calculated with the following integral:

$$\tau = \int_a^b \rho\Big(\sqrt{t^2 + h^2 + 2ht\cos\theta_v}\Big)\,dt \tag{1.}$$

Where $a$ and $b$ are the start and end points of the ray respectively. For $\rho(h)$, most other implementations discussed in chapter 5 use the exponential function for the density of the medium:

$$\rho(h) = \exp\left(-\frac{h}{h_0}\right) \tag{2.}$$

Here, $h_0$ is the *scale height*, which determines how fast the density falls off based on height.

For ozone, (Bruneton, 2017a) and (Hillaire, 2020) use the following density:

$$\rho(h) = \max\left(0, 1 - \frac{|h - 25|}{15}\right) \tag{3.}$$

Where $h$ is in kilometers.

The *transmittance* $T$, which indicates how much light is attenuated, can then be calculated as follows:

$$T = \exp(-\beta_a \tau) \tag{4.}$$

Where $\beta_a$ is the *absorption coefficient*, which determines how much of the light is absorbed.

Transmittance $T$ represents how much light is attenuated for a single path in the atmosphere, but does not take scattering into account. When a photon scatters, it changes direction. The new direction is not uniform, and depends on a phase function $F(\theta_s)$. The phase function used depends on the scattering medium. For Mie scattering, which is used to model aerosols, a common phase function to use is the Cornette-Shanks phase function described by (Cornette & Shanks, 1992). This phase function is used by (Bruneton & Neyret, 2008), (Fong et al., 2017), (Nishita et al., 1993) and others. It is shown below.

$$F(\theta_s) = \frac{3}{8\pi}\frac{(1-g^2)(1+\cos^2\theta_s)}{(2+g^2)(1+g^2-2g\cos\theta_s)^{\frac{3}{2}}} \tag{5.}$$

An alternative phase function used by (Pharr et al., 2016) is the Henyey-Greenstein phase function, described by (Henyey & Greenstein, 1941) can be seen below:

$$F(\theta_s) = \frac{1}{4\pi}\frac{1-g^2}{(1+g^2+2g\cos\theta_s)^{\frac{3}{2}}} \tag{6.}$$

Here, $g$ determines how directional the phase function is, with values near 1 being more directional, and thus photons less likely to significantly change direction upon scattering. For Rayleigh scattering, $g = 0$ is typically used. (Hillaire, 2020) uses an isotropic phase function instead, shown below:

$$F(\theta_s) = \frac{1}{4\pi} \tag{7.}$$

Using the phase function, it is now possible to calculate the transmittance for the entire path that light can take through the scattering medium. For a path with a single scattering event, the total radiance $L_{\text{scatter}}$ reaching the viewer can be calculated as follows:

$$L_{\text{scatter}} = E_{\text{light}} F(\theta_s) \beta_s \rho(h_s) T_s T_l \quad 8.$$

Where $\beta_s$ is the scattering density coefficient, $h_s$ is the height at which the scatter event takes place. $T_s$ is the transmittance from the viewer to the scattering point, $T_l$ is the transmittance from the scattering point to the light source, and $E_{\text{light}}$ is the incident radiance coming from the light source. Note that when the path between the scattering point and the light source is obstructed, $E_{\text{light}} = 0$.

In order to take multiple media types into account, their transmittance must be combined. This is done by multiplying all transmittance from all media types together. $L_{\text{scatter}}$ calculates the resulting radiance from scattering for only one medium type, and thus needs to be calculated for all media types as well. An example of the total radiance $L_{\text{total}}$ due to scattering from two scattering media $m$ and $r$, with respective phase functions, transmittance, scattering densities and density functions can be seen below:

$$L_{\text{total}} = E_{\text{light}} F^m(\theta_s) \beta_s^m \rho^m(h_s) T_s^m T_s^r T_l^m T_l^r + E_{\text{light}} F^r(\theta_s) \beta_s^r \rho^r(h_s) T_s^m T_s^r T_l^m T_l^r \quad 9.$$

When calculating the radiance of a path with multiple scattering events, Equation 8 can be applied recursively, where $E_{\text{light}}$ is replaced by another invocation of $L_{\text{scatter}}$.

# 4. Path tracing

Path tracing is a brute force method that attempts to calculate all possible paths a photon can take when scattering in an atmosphere. For this reason, it is often used as a reference when implementing other models of atmospheric scattering. (Hillaire, 2020), (Bruneton, 2017b) and (Wilkie et al., 2021) use a path tracer as a reference. (Suzuki & Yasutomi, 2023), (Hosek & Wilkie, 2012) and (Preetham et al., 1999) instead use it to fit a model to the result of a path tracer.

The path tracer described here is the implementation provided by (Hillaire, 2020), and was tested against (Pharr et al., 2016) for correctness. The implementation is based on (Fong et al., 2017).

As it is infeasible to compute all possible photon paths, the final image is created using Monte-Carlo integration. This is done by averaging many different, random photon paths into a final color value for the pixel.

## 4.1. Finding the scattering position

In order to find a point at which a scattering or absorption event occurs, delta tracking is used (Neumann, 1951). Delta tracking is an extension of closed-form tracking (Eckhart, 1987), which is used for homogeneous volumes. Tracking allows importance sampling any homogeneous volume with exponential attenuation according to Beer's law

$$T = \exp(-\beta t) \quad 10.$$

Where $t$ is the length of the path the photon takes through the volume, and $\beta$ is the volume density.

Tracking can then be used to determine the distance $t'$ that the photon travels before being scattered, using Equation 11, where $\xi$ is a uniform random number where $0 \leq \xi \leq 1$.

$$t' = -\frac{\ln(1-\xi)}{\beta} \quad 11.$$

As the atmosphere is modeled as an exponentially decaying density based on height from the surface, the tracking method cannot be used there. Instead, it is modified to take this density into account, which results in *delta tracking*. When a photon scatters at distance $t'$ along its path, but the density $\rho$ at this point is 0, no scattering can happen. This is instead treated as a scattering event where the photon continues to travel in the same direction. If the density $\rho > 0$, delta tracking only considers a scatter event to happen if Equation 12 holds, where $\rho_{\max}$ is the highest density that can be found in the volume.

$$\xi < \frac{\beta\rho}{\beta\rho_{\max}} \tag{12.}$$

If this does not hold, delta tracking is performed again, using the predicted scattering position derived from $t'$ as starting point. In case of absorption, the photon path does not scatter any light, and thus no light is added, and no further scattering events are considered. A minimal example of delta tracking can be seen below:

```
vec3 delta_tracking(vec3 start, vec3 dir) {
    while (true) {
        float xi = random();
        float zeta = random();
        float t = -ln(1.0 - xi) / rho_max;
        if (zeta < beta * volume_density(start + dir * t) / (beta * rho_max)) {
            // scatter event, return where
            return start + dir * t;
        } else if (length(start + dir * t) > max_distance) {
            // outside of volume, stop
            return start - dir;
        } else {
            start = start + dir * t;
        }
    }
}
```

Listing 1: Delta tracking

At the found scattering position, it is assumed that both a photon from the light source scatter here, as well as a photon from another scattering event in the atmosphere.

In order to combine multiple types of media with different scattering or absorption properties, $\rho_{\max}$ becomes the sum of all the $\rho$ of the different types of volumes. Then, when testing for a scattering event, each of the $\rho$ is assigned an interval. If $\xi$ is within that interval, there is a scattering event or absorption event for that medium.

## 4.2. Light contribution

The contribution of light to this scattering event can be computed in several ways. It is possible to perform delta tracking towards the light source, and if no scattering event happens, the light contribution is equal to $F(\theta_s)$. It is also possible to compute transmittance $T$ instead, calculating the optical depth using numerical integration directly, using a lookup table, or using ratio tracking, as described by (Novák et al., 2014).

Ratio tracking is a modification of delta tracking, where instead of testing whether there is a scattering event at each predicted scattering event at distance $t`$, the attenuation $T$, initialized with 1, is multiplied by Equation 13

$$1 - \frac{\beta\rho(h)}{\beta\rho_{\max}} \tag{13.}$$

The resulting transmittance is then multiplied by $F(\theta_s)$, and added to the final color of this path. A minimal example of ratio tracking is as follows:

```
float ratio_tracking(vec3 start, vec3 dir) {
    float T = 1.0;
    while (true) {
        float xi = random();
        float t = -ln(1.0 - xi) / rho_max;
        start = start + dir * t;
        if (length(start) > max_distance) {
            // outside of volume, stop
            break;
        }
        T *= 1.0 - beta * volume_density(start) / (beta * rho_max);
    }

    return T;
}
```

Listing 2: Ratio tracking

Performing these steps once results in single scattering.

### 4.3. Phase function

In order to perform multiple scattering, these steps are repeated with a new direction for the photon path, up to $N$ times, where $N$ is the scattering order. This new direction is chosen by importance sampling the phase function. For both Rayleigh and Mie scattering, no importance sampling is done. However, for both the Cornette-Shanks phase function is used, with Rayleigh scattering using $g = 0$.

It is possible to importance sample a phase function, as (Pharr et al., 2016) explains for the Henyey-Greenstein phase function in in *chapter 15.2.3: Sampling phase Functions*.

### 4.4. Final color

As each path is calculated for a single wavelength of light, it needs to be repeated for each wavelength. To then eliminate noise in the image, this needs to be repeated several times as well. The implementation provided by (Hillaire, 2020) simply selects the red, green, or blue color channel at random, by drawing from a uniform random distribution. Based on the color channel that was selected, different values for the scattering density $\beta$ are selected. The other two color channels are then set to 0 for this contribution. This is then repeated many times, and the average of this process is used as final color for the specified pixel.

## 5. Previous work

A number of methods have been developed for rendering atmospheric scattering in real time. This section provides an overview of various methods that have been developed for this purpose, as well as how these methods relate to each other.

### 5.1. Single scattering

The work discussed in this section covers implementations of the physics discussed in Chapter 3, assuming only single scattering occurs. Most of the focus is put on efficiently

evaluating the optical depth $\tau$, from Equation 1, as using numerical integration of it results in a large amount of evaluations of $\rho(h)$. This can quickly become prohibitively expensive.

One of the earlier models for atmospheric scattering was developed in (Nishita et al., 1993), in order to view an atmosphere from space. (O'Neil, 2005) later shows that this technique can be adapted to work from a ground view as well.

In order to keep the computational cost down, (Nishita et al., 1993) assumes that multiple scattering can be ignored as it has a negligible effect. They also assume absorption due to ozone is negligible, and that the density of the air falls off exponentially, as follows:

$$\rho(h) = \exp\left(-\frac{h}{h_0}\right) \tag{14.}$$

(Nishita et al., 1993) assumes only two types of scattering can happen. Rayleigh scattering, for smaller particles, and Mie scattering, for larger particles. For both types of scattering, they opt to use the Cornette-Shanks phase function as described in (Cornette & Shanks, 1992):

$$F(\theta_s) = \frac{3}{8\pi}\frac{(1-g^2)(1+\cos^2\theta_s)}{(2+g^2)(1+g^2-2g\cos\theta_s)^{\frac{3}{2}}} \tag{5.}$$

For Rayleigh scattering, they assume $g = 0$. For Mie scattering, it is assumed that$0.7 \leq g \leq 0.85$. The scale heights $h_0$ are set to $h_0^r = 7994m$ and $h_0^m = 1200m$ for Rayleigh and Mie respectively.

In order to reduce computational complexity, (Nishita et al., 1993) precomputes the optical depth, and stores it in a 2D lookup table. Recall the equation for the optical depth:

$$\tau = \int_a^b \rho\left(\sqrt{t^2 + h^2 + 2ht\cos\theta_v}\right)dt \tag{1.}$$

When assuming that the integral starts at the viewer's position on the view ray, and ends at infinity, and the air density falls of exponentially, it can be rewritten as follows:

$$\tau = \int_0^\infty \exp\left(\frac{\sqrt{t^2 + h^2 + 2ht\cos\theta_v}}{h_0}\right)dt \tag{15.}$$

As the scale height does not change often, it can be assumed that the integral now only depends on the height of the viewer $h$, and the view direction $\theta_v$. This allows these parameters to be used as axes in the 2D lookup table.

For the height, $N$ values of $h_i$ are chosen, with a spacing of

$$h_i = h_0 \log\left(\frac{1-i}{N}\right) + h \tag{16.}$$

Where $h_i$ is the height used to compute the $i$th row of the lookup table. Then, the columns of the lookup table, which depend on the viewing angle, also have to be computed. This is done by taking the heights $h_i$, and imagining them as spheres around the center of the planet. Then, a number of cylinders, aligned to the light direction is swept through the spheres. The index of the sphere, and the index of the cylinder are then used to look up the row and column of the lookup table.

To calculate the optical depth, trapezoidal integration is used.

To arrive at a final color for the red, green, and blue color channels, the equation shown in Equation 9 is evaluated for a number of points along the view path, and added together.

$$L_{\text{total}} = E_{\text{light}} F^m(\theta_s) \beta_s^m \rho^m(h_s) T_s^m T_s^r T_l^m T_l^r + E_{\text{light}} F^r(\theta_s) \beta_s^r \rho^r(h_s) T_s^m T_s^r T_l^m T_l^r \qquad 9.$$

This is then done for every pixel in the output image.

The transmittance for the light direction is calculated using the optical depth stored in the lookup table. The transmittance for the view direction is calculated using trapezoidal integration, as this can be performed at the same time the radiance is accumulated.

From here on, the work of (Nishita et al., 1993) remains used as a base for single scattering. Most of the variations presented next vary from their implementation in the manner of how they implement the numerical integration of the optical depth integral.

(O'Neil, 2004) presents a different formulation of the lookup table that (Nishita et al., 1993) uses. Instead of using the spheres intersecting the cylinders, it is possible to simply divide the height between $h = 0$ and $h = r_a$, where $r_a$ is a defined radius of the atmosphere. Then when looking up the value, (O'Neil, 2004) linearly interpolates between the entries in the lookup table for the specific height. For the view angle, they use the result of $\cos\theta_v$ to map to a column in the lookup table.

Besides storing the optical depth for Rayleigh and Mie, (O'Neil, 2004) also stores the density $\rho(h)$ for Rayleigh and Mie in the lookup table. If the view path for the optical depth intersects the planet, this is set to 0, which emulates the effect of a planet shadow.

Then, in order to compute the transmittance of the view path segment, the lookup table is used again, and the segment optical depth is calculated the same way as (Nishita et al., 1993) does it. However, if the view path intersects the planet, the view direction the lookup table is sampled with is rotated by 180°, and the order in which they are subtracted is swapped. This avoids floating point precision errors due to a very large optical depth.

Later, (O'Neil, 2005) modifies their previous work in (O'Neil, 2004) to run on the GPU. Instead of performing the full calculation for each pixel, they move it to the vertex shader, and let the GPU interpolate between vertices to get the final per pixel result. The phase functions $F(\theta_s)$ are however calculated per pixel, as this resolves some artifacts.

As GPUs at the time did not have the ability to read from a texture from the vertex shader, it is not possible to use a lookup table. Instead, (O'Neil, 2005) finds that it is possible to approximate the optical depth using a different formula. For the height, they find that $\exp(-4h)$ is a common factor for all view angles. For the view angles, they find that taking the logarithm of the view angle results in an unknown curve, and fit a polynomial to this curve. This results in the following equation:

$$\tau = \exp(-s_1 h) s_2 \exp\left(5.25x^4 - 6.80x^3 + 3.83x^2 + 0.459x - 0.00287\right) \qquad 17.$$

Where $s_1$ and $s_2$ are constants that depend on the atmosphere parameters. Of note is that this equation for optical depth only works for one combination of planet radius $r$ and scale height $h_0$. If these are changed, the polynomial and $s_1$ and $s_2$ have to be fit again. Using this, (O'Neil, 2005) implements their single scattering the same way as described in (O'Neil, 2004).

Besides the polynomial approximation (O'Neil, 2005) finds for the optical depth, there has also been work in physics literature to solve a similar problem. The result is the Chapman

function $\mathrm{Ch}(x, \chi)$, first introduced in (Chapman, 1931). As this function cannot be computed exactly, other approximations have been formulated, such as by (Kocifaj, 1996), (Yue, 2024). A more general overview can be found in (Vasylyev, 2021).

(Schuler, 2012) derives a new approximation of the Chapman function, that is simpler and more numerically stable when implemented using 32 bit floating point. (Schuler, 2012) first starts with the Chapman function presented in (Kocifaj, 1996):

$$\mathrm{Ch}(x, \chi) = \frac{1}{2}\left[\cos\chi + \exp\left(\frac{x\cos^2\chi}{2}\right)\mathrm{erfc}\left(\sqrt{\frac{x\cos^2\chi}{2}}\right)\left(\frac{1}{x} + 2 - \cos^2\chi\right)\sqrt{\frac{\pi x}{2}}\right] \tag{18.}$$

Where

$$x = \frac{r + h}{h_0}, \quad \text{and} \quad \chi = \theta_v \tag{19.}$$

The optical depth from viewer $v$ can then be calculated as follows:

$$\tau = \mathrm{Ch}\left(\frac{r + h}{h_0}, \theta_v\right)\exp\left(-\frac{h}{h_0}\right) \tag{20.}$$

In order to simplify Equation 18, (Schuler, 2012) makes the following assumptions:

$$\mathrm{Ch}(x, \chi) = \mathrm{Ch}(x, -\chi), \quad \mathrm{Ch}(x, 0) = 1, \quad \text{and} \quad \lim_{x\to\infty}\mathrm{Ch}(x, \chi) = \frac{1}{\cos\chi} \tag{21.}$$

Using this, they find an approximation that works well when $\chi < 90°$:

$$\mathrm{Ch}'(c, \chi) \approx \frac{c}{1 + (c - 1)\cos\chi} \quad \text{if } \chi < 90° \tag{22.}$$

where $c = \mathrm{Ch}(x, 90°)$. In order to then derive the Chapman function for $\chi > 90°$, it is possible to restructure the problem. The optical depth from the viewer to infinity is the same as the optical depth from infinity behind the viewer to infinity in front of the viewer. Then, by subtracting the optical depth from the viewer to infinity behind the viewer, the optical depth from the viewer to infinity can be calculated. This is expressed as follows:

$$\mathrm{Ch}(x, \chi) = 2\exp(x - x\sin\chi)\ \mathrm{Ch}(x, 90°) - \mathrm{Ch}(x, 180° - \chi) \tag{23.}$$

This means, if $\mathrm{Ch}(x, \chi)$ is known for $\chi < 90°$, it is also known for $\chi \geq 90°$. Rewriting Equation 20, Equation 22, and Equation 23, as it is desired to calculate the optical depth without precision errors, results in the following equation for optical depth:

$$\tau_{\mathrm{Ch}} = \begin{cases} \exp\left(-\dfrac{h}{h_0}\right)\dfrac{c}{1 + c\cos\theta_v} & \text{if } \theta_v < 90° \\ -\exp\left(-\dfrac{h}{h_0}\right)\dfrac{c}{1 - c\cos\theta_v} + 2\sqrt{x_0}\exp\left(\dfrac{r}{h_0} - x_0\right) & \text{otherwise} \end{cases} \tag{24.}$$

Where

$$c = \sqrt{\frac{r + h}{h_0}}, \quad \text{and} \quad x_0 = \sqrt{1 - \cos^2\theta_v}\left(\frac{r + h}{h_0}\right) \tag{25.}$$

(Schuler, 2012) then uses $T_{\text{Ch}}$ for the optical depth to implement a single scattering atmosphere, similar to how (Nishita et al., 1993), and (O'Neil, 2004) have done it.

Of note is another approximation to optical depth by (Bruneton & Neyret, 2008). While not explained is the paper, they implement an approximation of the optical depth in the original implementation, used for the transmittance when calculating the radiance of reflected light of the surface of the planet. Recall the equation for optical depth:

$$\tau = \int_a^b \rho\left(\sqrt{t^2 + h^2 + 2ht\cos\theta_v}\right) dt \tag{1.}$$

Here, the distance from the surface at position $t$ is calculated using:

$$h_t = \sqrt{t^2 + h^2 + 2ht\cos\theta_v} \tag{26.}$$

By assuming that

$$\sqrt{1+u} \approx 1 + \frac{u}{2} \tag{27.}$$

This allows rewriting the optical depth integral, with $\rho(h) = \exp\left(-\frac{h}{h_0}\right)$, and the path going from 0 to maximum distance $d$, to the following:

$$\tau \approx \int_0^d \exp\left(-\frac{\frac{t^2}{2h} + t\cos\theta_v}{h_0}\right) dt \tag{28.}$$

The primitive of this integral is as follows:

$$\mathrm{T}(t) \approx \exp\left(\frac{h\cos^2\theta_v}{2h_0}\right) \operatorname{erfc}\left(\frac{t + h\cos\theta_v}{\sqrt{2h \cdot h_0}}\right) \sqrt{\frac{\pi}{2} \cdot h \cdot h_0} \tag{29.}$$

$\operatorname{erfc}(x)$ can be approximated with

$$\operatorname{erfc}(x) \approx 2\frac{\exp(-x^2)}{2.3192x + \sqrt{1.52x^2 + 4}} \quad \text{if } x \geq 0 \tag{30.}$$

When $x < 0$, $\operatorname{erfc}(-x) = 2 - \operatorname{erfc}(x)$ can be used. Combining this allows the optical depth to be evaluated from the viewer up to distance $d$ with the following code:

```
float optical_depth(float h_0, float h, float cos_theta_v, float d) {
    float a = sqrt((0.5 / h_0) * h);
    vec2 a01 = a * vec2(cos_theta_v, cos_theta_v + d / h);
    vec2 a01s = sign(a01);
    vec2 a01sq = a01 * a01;
    float x = a01s.y > a01s.x ? exp(a01sq.x) : 0.0;
    vec2 y = a01s / (2.3193 * abs(a01) + sqrt(1.52 * a01sq + 4.0))
        * vec2(1.0, exp(-d / h_0 * (d / (2.0 * r) + cos_theta_v)));
    return sqrt((6.2831 * h_0) * h) * exp((r - h) / h_0)
        * (x + dot(y, vec2(1.0, -1.0)));
}
```

Listing 3: (Bruneton & Neyret, 2008)'s optical depth approximation

## 5.2. Multiple scattering

Besides single scattering, multiple scattering adds a non-negligible contribution to the total radiance of the scattered light. On earth, this is best noticed during sunset and sunrise, as it adds an extra blue glow to the atmosphere.

When assuming single scattering, the chosen scattering direction is trivial, as this direction changes from the light directly towards the viewer when a photon is scattered. When assuming two scattering events per photon path, this direction is less trivial. The second scatter, where the photon scatters towards the viewer, now has to integrate over all scattering directions to account for all the incoming radiance due to scattering.

When using numerical integration, this results in having to evaluate many single scattering paths for one multiple scattering event. This becomes prohibitively expensive to compute. In order to still render an atmosphere in real time, (Bruneton & Neyret, 2008) proposes using several lookup tables.

In order to compute the transmittance, (Bruneton & Neyret, 2008) uses a lookup table. Instead of storing the optical depth, and using it to calculate the transmittance, they store the transmittance directly in the table. This allows storing the combined transmittance of all scattering media in a single table, instead of one table for the optical depth of each media type.

Instead of evaluating the single scattering equation directly, (Bruneton & Neyret, 2008) recognizes it can be precomputed as well. Recall the single scattering equation:

$$L_{\text{scatter}} = E_{\text{light}} F(\theta_s) \beta_s \rho(h_s) T_s T_l \tag{8.}$$

Here, $T_s$ depends on the viewer height $h$ and view direction $\theta_v$. $T_l$ depends on the light direction $\theta_l$. The height of the scattering event can be calculated from $h$ and $\theta_v$, as many scattering events happen along one view ray. The last dependency is the scattering angle $\theta_s$. This means single scattering only relies on 4 different variables, and can thus be precomputed. The results are stored in a 4D lookup table.

As 4D textures are not available on the GPU, (Bruneton & Neyret, 2008) instead chooses to use a 3D texture, and for each slice of the texture, divide it into multiple tiles to emulate a 4D texture.

Then, when $L_{\text{scatter}}$ is needed, it can simply be retrieved from the lookup table. Rayleigh and Mie scattering are stored in separate tables.

The contribution of multiple scattering can be precomputed in a similar manner to how it is done for single scattering. Instead of calculating the incoming radiance from the sun due to transmittance, each sample point along the view path integrates the result of single scattering over each direction. For more than two scatter events, the previous result from the multiple scattering table is read instead.

Integrating the scattering for all directions on all sample points along the view path is expensive. To resolve this, (Bruneton & Neyret, 2008) uses another lookup table, that calculates the incoming radiance from multiple scattering for every scattering direction $\theta_s$, viewer height $h$ and light direction $\theta_l$. This lookup table can then be used to evaluate the incoming radiance due to multiple scattering at each sample point.

This process is then repeated for several steps, in order to compute scattering for more than 2 scatter events per path, referred to as *scattering orders* by (Bruneton & Neyret, 2008). This

then results in a final lookup table that incorporates both the single scattering and multiple scattering. This table is then read to determine the radiance.

(Elek, 2009) presents a variation on the model presented by (Bruneton & Neyret, 2008). Instead of using a table with 4 dimensions, (Elek, 2009) chooses to not incorporate the scattering angle $\theta_s$, and use a 3-dimensional table instead. This means that the phase function has to be calculated, instead of reading it from the table when computing radiance. It also means the planet shadow cannot be stored in the table either. (Elek, 2009) argues this effect is negligible enough for this tradeoff to work.

(Hillaire, 2020) proposes a different method of precomputing the scattering. Instead of precomputing single scattering for all viewer positions, they use a 2D lookup table, and recompute it when either light direction $\theta_l$ or viewer height $h$ changes.

For multiple scattering, (Hillaire, 2020) chooses to instead approximate the result, instead of fully calculating each scattering order. To make the approximation work, (Hillaire, 2020) assumes that after single scattering, the phase function becomes isotropic. This eliminates the dependence on the scattering direction $\theta_s$ for multiple scattering.

Second order scattering can then be calculated by, for height $h$, and view direction $\theta_v$, calculate the incoming radiance from scattering for every direction.

Scattering orders higher than 2 can be approximated by multiplying the second order scattering by $F_{\text{ms}}$, which is a geometric series infinite sum:

$$F_{\text{ms}} = 1 + f_{\text{ms}} + f_{\text{ms}}^2 + f_{\text{ms}}^3 + ... = \frac{1}{1 - f_{\text{ms}}} \tag{31.}$$

$f_{\text{ms}}$ represents the radiance that the sample point would receive, if the atmosphere would emit light, with no other scattering contribution. $f_{\text{ms}}$ can thus be calculated by integrating the transmittance over all directions.

## 5.3. Fitted models

The previously discussed methods have all tried computing the scattering equations of Chapter 3 directly. This section introduces a different method, that attempts to find an equation that can be fitted to the output of a path tracer, in order to then render the atmosphere without requiring the scattering equations, or precomputed lookup tables. To keep complexity of the fitted model low, these do assume that the viewer is either on the ground, or close to it, and thus do not support viewing from space.

One of the earlier models to do this was presented by (Preetham et al., 1999). Their model is based on an earlier sky luminance model by (Perez et al., 1993):

$$\mathcal{F}(\theta, \gamma) = \left(1 + A \exp\left(\frac{B}{\cos\theta}\right)\right)(1 + C \exp(D\gamma) + E \cos^2\gamma) \tag{32.}$$

Here, $A$, $B$, $C$, $D$, $E$ are coefficients that are fitted to match a reference. The presented model does not directly give radiance for a specific wavelength, or red, green, and blue channels. Instead, the resulting color space is *CIE xyY*. This can then be converted to the radiance for a specific wavelength as (Preetham et al., 1999) describes.

Radiance $Y$ is calculated as follows:

$$Y = Y_z \frac{\mathcal{F}(\theta_v, \gamma)}{\mathcal{F}(0, \theta_l)} \tag{33.}$$

The chromaticity values $x$ and $y$ are calculated similarly:

$$x = x_z \frac{\mathcal{F}(\theta_v, \gamma)}{\mathcal{F}(0, \theta_l)} \quad \text{and} \quad y = y_z \frac{\mathcal{F}(\theta_v, \gamma)}{\mathcal{F}(0, \theta_l)} \tag{34.}$$

Details for calculating $Y_z$, $x_z$ and $y_z$ are further described in (Preetham et al., 1999).

The coefficients of the model given in (Preetham et al., 1999) are calculated by fitting them to the results of a path tracer. The result is a single formula that can model the color of the atmosphere, given $h = 0$.

(Hosek & Wilkie, 2012) improves upon the findings of (Preetham et al., 1999), by taking the reflection of light from the ground into account. They also improves the aureole, which is a bright halo around the sun. This halo is also darkened when the sun is near the horizon. This results in a new formula:

$$\mathbb{F}(\theta, \gamma) = \left(1 + A \exp\left(\frac{B}{0.01 + \cos\theta}\right)\right)\left(C + D\exp(\gamma E) + F\cos^2\gamma + G\chi(H, \gamma) + I\sqrt{\cos\theta}\right) \tag{35.}$$

Where $A$, $B$, $C$, $D$, $E$, $F$, $G$, $H$, and $I$ are the coefficients to fit. $\chi$ is calculated as follows:

$$\chi(g, \alpha) = \frac{1 + \cos^2\alpha}{1 + g^2 - 2g\cos\alpha} \tag{36.}$$

The final radiance for wavelength, or color channel $\lambda$ is then calculated with:

$$L_\lambda = \mathbb{F}(\theta_v, \gamma) L_{M\lambda} \tag{37.}$$

All coefficients are then again fit on the result of a path tracer. Unlike the work of (Preetham et al., 1999), the coefficients of $\mathbb{F}(\theta, \gamma)$ are interpolated from a larger set of coefficients, based on turbidity, ground albedo, and sun elevation $\theta_s$.

While these models fit the general sky quite well, they are not capable of representing more finer details. To address this, (Wilkie et al., 2021) introduces a new model based on Canonical Polyadic Decomposition (Kolda & Bader, 2009). This method is similar to singular value decomposition, and acts like an image compression technique. Besides outputting the direct radiance, it also outputs the transmittance, and polarization of light. Contrary to (Preetham et al., 1999) and (Hosek & Wilkie, 2012), this model is also able to represent the sky when the sun is below the horizon, as well as $h > 0$, up to a maximum height. The downside is that this requires significantly more coefficients, with the largest version of the model being 2.2 GB in size.

As (Bruneton & Neyret, 2008) and (Hillaire, 2020) have done with precomputing single and multiple scattering, it is also possible to store the results of a path tracer in a lookup table directly, instead of fitting a formula to the results. This is the method (Suzuki & Yasutomi, 2023) takes. For the lookup table, they use 64 2D lookup tables, each table representing a single time of day, of angle $\theta_l$, separated in $\theta_v$ and $\gamma$. In order to fix artifacts near the horizon, more tables are allocated when the sun is close to the horizon. Per table, a special spherical mapping is used that increases the number of samples near the sun.

# 6. Research goals

As seen in Chapter 5, the discussed models of atmospheric scattering can be divided in roughly 3 categories:

- Numerical integration, where all equations are directly integrated. For real-time use, only single scattering is considered.
- Precomputed models, where a large part of the integration is done ahead of time, allowing for the resulting model to be used in real-time. As there is no constraint on the runtime of the precomputation step, these models can take multiple scattering into account.
- Fitted models and approximations, where either part of the calculations or the entire model is replaced using an approximation, or a fitted model. These models make a tradeoff in that they cannot be used in certain circumstances, for example, (Preetham et al., 1999) and (Hosek & Wilkie, 2012) cannot be used for viewers that are viewing the atmosphere from either higher up, or from space.

From these categories it becomes clear that no general fitted model, that can be used for both ground and space views, exists. This work aims to create such a model, and compare it to the previously discussed models. While doing this, this work attempts to answer the following questions:

- To what extent is it possible to create a fitted model for atmospheric scattering that can support both a ground-based viewer, a viewer from higher altitudes, and space?
- How is the runtime performance of this model compared to lookup-table based models?
- How difficult is it to adapt the model to use a different $\rho(h)$ than the exponentially decaying density the discussed models assume?

# 7. Assumptions

Before making the new model, a number of requirements have to be set for what it should be capable of doing. The first requirement is that it has to be able to produce a sky view, as all discussed models in Chapter 5 3 can do so. In order to render sky and space views correctly, both transmittance from the viewer to the surface and scattering from the viewer to the surface need to be provided, in order to attenuate light coming from surfaces inside the atmosphere. (Bruneton & Neyret, 2008) provides a method to calculate transmittance and scattering for a segment of the view path even if there is no way to provide this.

As (Bruneton & Neyret, 2008) and (Elek, 2009) calculate multiple scattering as adding on top of single scattering, it may be possible to calculate single scattering and multiple scattering separately.

## 7.1. Model parts

This means that the model needs to provide at least the following:

- Transmittance from viewer to infinity
- Scattering from viewer to infinity.
- Either combined scattering or separate multiple scattering.

Note that this does not include optical depth. As transmittance is derived from the optical depth, the optical depth approximations provided by (Yue, 2024), (Vasylyev, 2021), (Kocifaj, 1996) and (Schuler, 2012) can still be used here.

As shown in (Bruneton & Neyret, 2008), transmittance and scattering from the viewer to infinity is enough to reconstruct both transmittance and scattering along a segment in the atmosphere. For this reason, it does not have to be modeled directly.

### 7.2. Coordinate space

The coordinate space used in chapter 5 uses viewer height $h$, and viewer direction $\theta_v$ to calculate the height along distance $t$ of the view path. This is illustrated below:

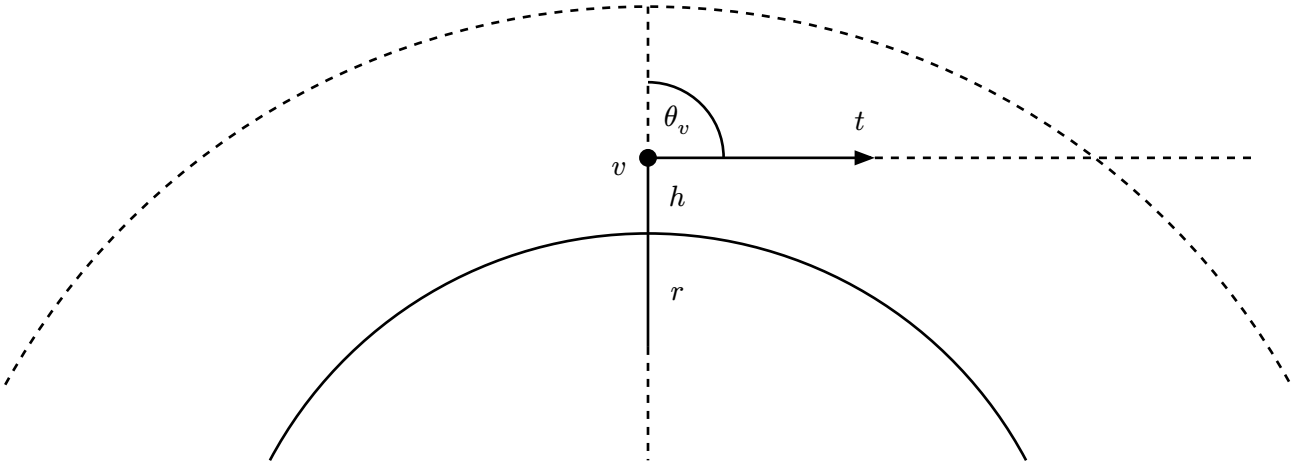



Figure 4: Viewer $v$ in an atmosphere.

The height above the planet surface, along the ray at position $t$, $h(t)$, can then be calculated as follows:

$$h(t) = \sqrt{t^2 + (h+r)^2 + 2t(h+r)\cos\theta_v} - r \qquad 38.$$

For the light direction, $\theta_l$ can be used, as described in Section 5. Only the view and light angles $\theta_v$ and $\theta_l$ do not cover all possible angles that can be used in the phase function. For this reason, $\gamma$ is used. This is the same as described earlier in figure 2:

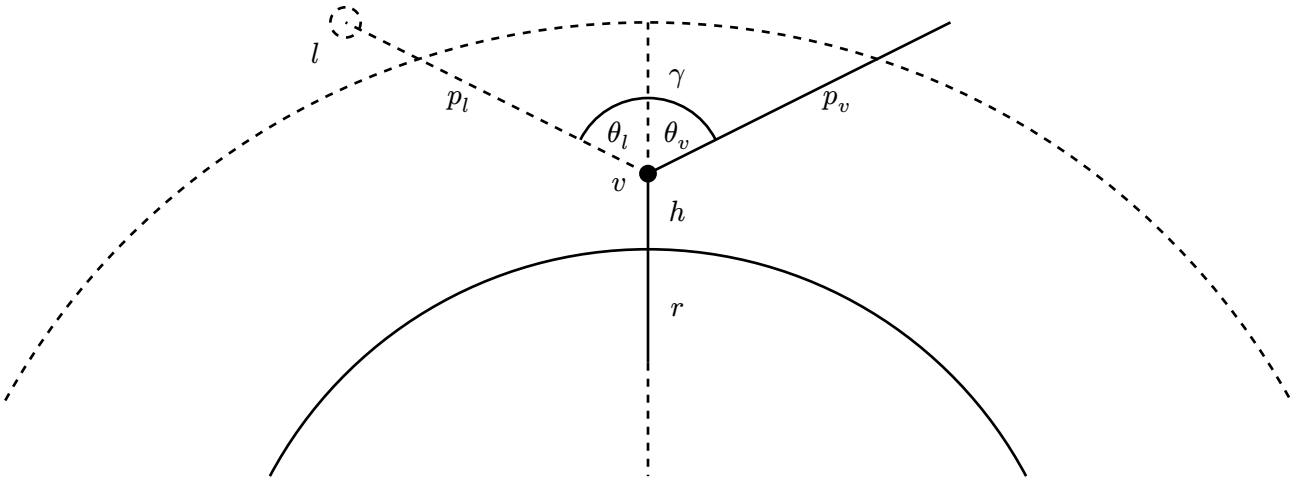



Figure 2: Viewer $v$ in an atmosphere.

## 8. Ground truth

The ground truth rendering is provided by a volumetric path tracer, implemented as described in (Pharr et al., 2016) and (Fong et al., 2017). To match the other models described in chapter 5, the path tracer uses the same atmospheric density as described in (Bruneton & Neyret, 2008). This means Rayleigh and Mie scattering are represented using exponentially decaying densities, depending on altitude, and Ozone density is represented with a tent function.

As correct spectral rendering is not a focus, the path tracer outputs colors in linear sRGB color space.

# 9. Automatic model finding

As manually finding a formula that fits either the transmittance or scattering equations may be very time consuming, as well as complex in case of the scattering equations, it is preferred to find an automated solution for this. For this, it is useful to have some reference data to work with.

## 9.1. Reference data

As a ground truth path tracer is available, it is possible to utilize it to obtain the correct value for transmittance for each $(h, \theta_v)$ pair, as well as the scattering radiance for each $(h, \theta_v, \theta_l, \gamma)$ pair.

The transmittance data is rendered out to a 2D texture. The x axis represents the height $h$, and the y axis is the view angle $\cos\theta_v$. The result is shown below in Figure 5.

For scattering, there are 4 relevant parameters, which are rendered out to a tiled 2D texture. Here, inside each tile, the x, and y coordinates represent the squared height $h^2$, and view angle $\cos\theta_v$. The tile x and y coordinates represent the light angle $\cos\theta_l$, and the phase function angle $\cos\gamma$.

For scattering, a 4D texture is used, but it is rendered as a 2D texture, divided in tiles. Inside each tile, the x axis represents the squared view height $h^2$. The square gives more samples to the lower altitudes, improving results. The y axis then represents the view angle $\theta_v$.

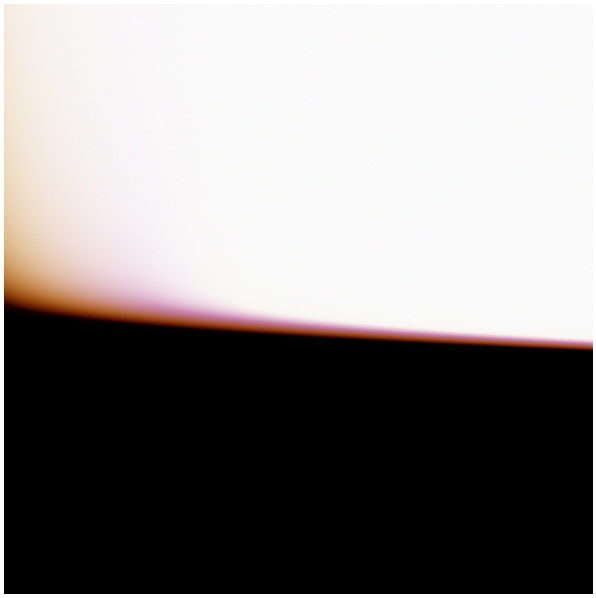


Figure 5: Transmittance data, as image. x axis is the viewer height from the surface, y axis is the viewing angle $\cos\theta_v$

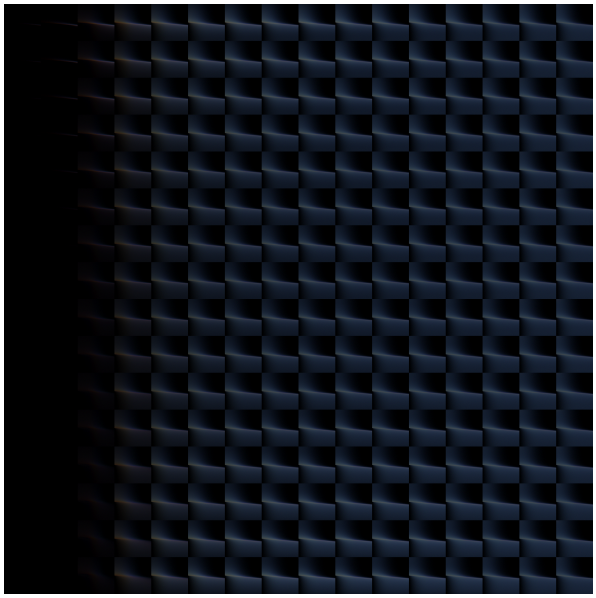


Figure 6: Scattering data, as image.

- x axis is the light angle $\cos\theta_l$,
- y axis is the angle between the light and view rays, $\cos\gamma$.
- Inside each tile, the x axis is the squared viewer height from the surface,
- The y axis is the viewing angle $\cos\theta_v$

As there is no scattering when the sun is obscured by the planet, the light angle $\theta_l$ is limited to be between 0° and 100°. As the phase function angle $\gamma$ is clamped to come from the given combination of $\theta_v$ and $\theta_l$.

For both the transmittance, and scattering, the tables parameterized according to (Bruneton & Neyret, 2008) were also generated, but these didn't give significantly better or worse results when trying to fit functions to these tables.

## 9.2. Observations

When looking at the transmittance data, it becomes clear the transmittance per color channel follows a curve similar to the one defined by the formula $\exp(-\exp(-f(h,\theta_v)))$, where $f(x)$ is some unknown function based on the viewer height $h$ and angle $\theta_v$. When attempting to fit manually,

$$f(h,\theta_v) = h^2[-1+\cos\theta_v] \tag{39.}$$

appears to be a good approximation, but only for low values of $h$. As $\exp(-\exp(-x))$ has the general shape of the sigmoid function:

$$\frac{1}{1+\exp(-x)} \tag{40.}$$

## 9.3. symbolic regression

One method of automatically finding a function that fits given data is symbolic regression. `PySR` (Cranmer, 2023) is a python library that can perform symbolic regression, and claims to better than some other methods and libraries available at finding functions. It however was not capable of finding a suitable function that fit the transmittance data. Even on simplified data, with only one exponentially decaying media, and no ozone layer, it was only able to find the

$$\exp(-\exp(-f(h,\theta_v))) \tag{41.}$$

formula, however, the found formulas for this did not function beyond specific ranges.

As symbolic regression worked badly for transmittance, it was not attempted for scattering, as the scattering data is more complex.

## 9.4. Kolmogorov-Arnold networks

Another method that can potentially produce a function that fits the given data are Kolmogorov-Arnold networks (Liu et al., 2025). Due to limitations in how Kolmogorov-Arnold networks function, they have difficulty representing the steep cutoff needed for the transmittance data. For this reason, they did not fit the data well, and thus weren't tried for scattering.

## 9.5. Polynomial fit

As a polynomial has trouble representing the steep cutoff, it has to be used in conjunction with another method that can. For this reason, the result of the polynomial is put into the sigmoid function. A separate polynomial is used for the height $h$ and $\theta_v$, which are then added together. From experiments it becomes clear that the height does not need more than one degree for the approximation to work. For $\theta_v$, only a second degree polynomial is needed. This results in the following formula:

$$\frac{1}{1+\exp(-a_1-a_2h-a_3\cos\theta_v-a_4\cos^2\theta_v)} \tag{42.}$$

Where $a_n$ are the parameters used. This is then done once for Mie, and once for Rayleigh.

## 9.6. Neural networks

The last automatic method tried are neural networks, specifically multilayer perceptrons, using the python library pytorch (Ansel et al., 2024). For activation functions, both the sigmoid and the rectified linear unit were tried. For training, the Adam optimizer (Kingma & Ba, 2017) was used, as other optimizers did not converge. A learning rate of 0.01 was used.

### 9.6.1. Transmittance

For the transmittance data, a neural network with a single hidden layer of size 4, with the sigmoid activation function was trained. More layers, or layers with more neurons did not significantly improve results. The neural network converged for both the direct transmittance table and the parameterization described in (Bruneton & Neyret, 2008).

### 9.6.2. Scattering

For scattering, a neural network with two hidden layers, of size 8 was trained. For the activation function of the final layer, $\exp(-x)$ is used. Reducing the amount of hidden layers or using smaller hidden layers made the resulting approximation of the scattering data visibly worse. Adding more layers or increasing the layer size did not improve results much.

### 9.6.3. Implementation as atmosphere

As the `GLSL` shading language supports up to `4x4` matrices, implementing the neural network for transmittance is trivial, as the weights for the network can be directly represented using the `mat2x4`, `mat4x4` and `mat4x3` types. The biases can be represented using the `vec4` and `vec3` types. This allows the neural network to be directly implemented into `GLSL`.

As the scattering neural network uses a hidden layer size of 8, it is no longer possible to represent directly in the types available in `GLSL`. Instead, the matrices used for the weights in the hidden layers have to be split up. The hidden layer states also have to be split up into two `vec4`′s.

As the neural network is only trained for viewer positions inside the atmosphere, any viewer that is outside the atmosphere is moved to the top atmosphere boundary.

In order to calculate the transmittance for a given segment, the same method as described in (Bruneton & Neyret, 2008) is used. The neural network is trained on the transmittance of the full ray, and if the ray intersects the surface, the transmittance at this intersection is calculated using the neural network. The full transmittance is then calculated as follows:

$$T = \frac{T_{\text{viewer}}}{T_{\text{intersect}}} \tag{43.}$$

where $T_{\text{viewer}}$ is the transmittance from the viewer, and $T_{\text{intersect}}$ is the transmittance, in the view direction, at the point the view ray intersects the planet.

As the neural network is not accurate enough when the view ray is looking towards the planet surface, the rays are instead reversed, and the intersection position is used as the full ray transmittance. The viewer position is then used as the intersect position.

Scattering is again done using the method described in (Bruneton & Neyret, 2008). The neural network is used to calculate scattering for the full ray. If it hits the planet surface, the scattering from that position onward is then removed from the final result, as follows:

$$S = S_{\text{viewer}} - S_{\text{intersect}} \left( \frac{T_{\text{intersect}}}{T_{\text{viewer}}} \right) \tag{44.}$$

Note that the transmittance of the viewer and intersect point look in the opposite direction as the view ray, as this improves accuracy, as explained earlier.

# 10. Manual modeling

Besides of using automatic methods to find an approximation to the scattering equations, a manual approach may also provide a useful approximation.

## 10.1. Flat homogeneous atmosphere

The simplest possible model for scattering is a flat homogeneous atmosphere, with the viewer $v$ being at a distance of $h_v$ below the top of the atmosphere boundary. Assuming only one scattering medium type, as well as both the viewer and light source $l$ above the horizon, meaning $0° \leq \theta_v \leq 90°$ and $0° \leq \theta_l \leq 90°$. This looks as follows:

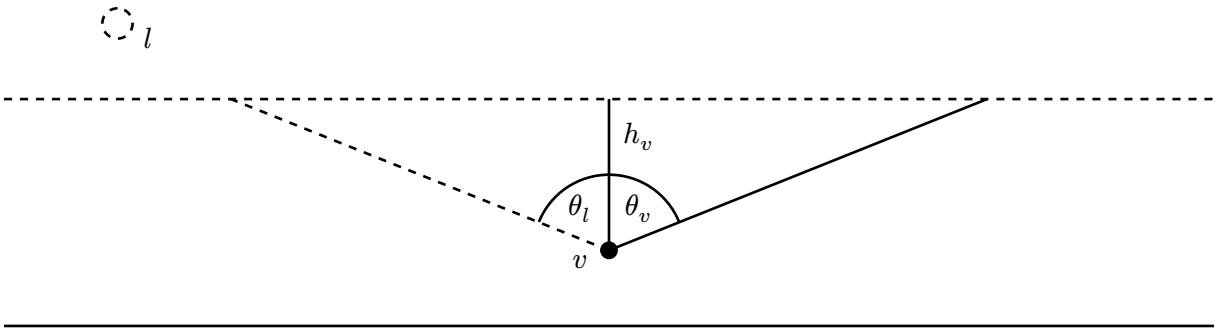



Figure 7: Viewer $v$ in a flat atmosphere.

To then calculate single scattering, the following integral has to be solved, where $\beta_a$ is the absorption coefficient of the atmosphere, and $\beta_s$ the scattering coefficient.

$$L_{\text{scatter}} = \int_0^{h_v} \beta_s \exp\left( -t \frac{\beta_a}{\cos\theta_v} \right) \exp\left( -(h_v - t) \frac{\beta_a}{\cos\theta_l} \right) \mathrm{d}t \tag{45.}$$

To make integration easier, this can then be rewritten as follows:

$$L_{\text{scatter}} = \beta_s h_v \int_0^1 \exp\left( -t \frac{\beta_a}{\cos\theta_v} - (1-t) \frac{\beta_a}{\cos\theta_l} \right) \mathrm{d}t \tag{46.}$$

As this can be integrated exactly, this then becomes:

$$L_{\text{scatter}} = \beta_s h_v \frac{\exp\left( -\frac{\beta_a}{\cos\theta_v} \right) - \exp\left( -\frac{\beta_a}{\cos\theta_l} \right)}{\frac{\beta_a}{\cos\theta_l} - \frac{\beta_a}{\cos\theta_v}} \mathrm{d}t \tag{47.}$$

The shader presented in (Jodie, 2019) utilizes this to then approximate the single scattering equations in a spherical atmosphere.

## 10.2. Spherical atmosphere

When the planet radius is large enough, with a small enough radius for the atmosphere, using a flat atmosphere can serve as a good enough approximation. (Jodie, 2019) does this, and replaces $\frac{\beta_a}{\cos\theta_v}$ and $\frac{\beta_a}{\cos\theta_l}$ with the optical depth of a homogeneous sphere. Under the assumption that the sphere extends up to the scale height of an exponentially decaying atmosphere, with the viewer standing on the planet, at distance $r$ away from the planet center. As the atmosphere remains homogeneous, the optical depth is the distance from the viewer to the edge of the atmosphere. The optical depth can thus be expressed as a line starting from distance $r$ from the origin, intersecting a circle of radius $r + h_0$:

$$\tau_{\text{sphere}} = \beta_a \sqrt{(r + h_0)^2 - r^2 + r^2 \cos^2\theta_v} - \beta_a r \cos\theta_v \tag{48.}$$

When the difference between $r + h_0$ and $r$ is set equal to one, this can then be rewritten to the *DR-1* (shown below) form described in equation 9 in (Rapp-Arrarás & Domingo-Santos, 2011):

$$f(\theta_v, a) = \sqrt{1 + 2a + a^2 \cos^2\theta_v} - a\cos\theta_v \tag{49.}$$

Where $a$ is the radius of the sphere, relative to the scale height $h_0$.

When plotted against the optical depth integral, with $\rho(h) = \exp\left(-\frac{h}{h_0}\right)$, it shows this approximation is reasonable for any view angle above the horizon. Shown below is the approximation compared to the reference integral, as well as the approximation described in (Schuler, 2012), for a planet with a radius of 600, and a scale height of 1. $\tau$ is the optical depth, and $\theta$ is the angle of the view ray with the surface normal of the planet.

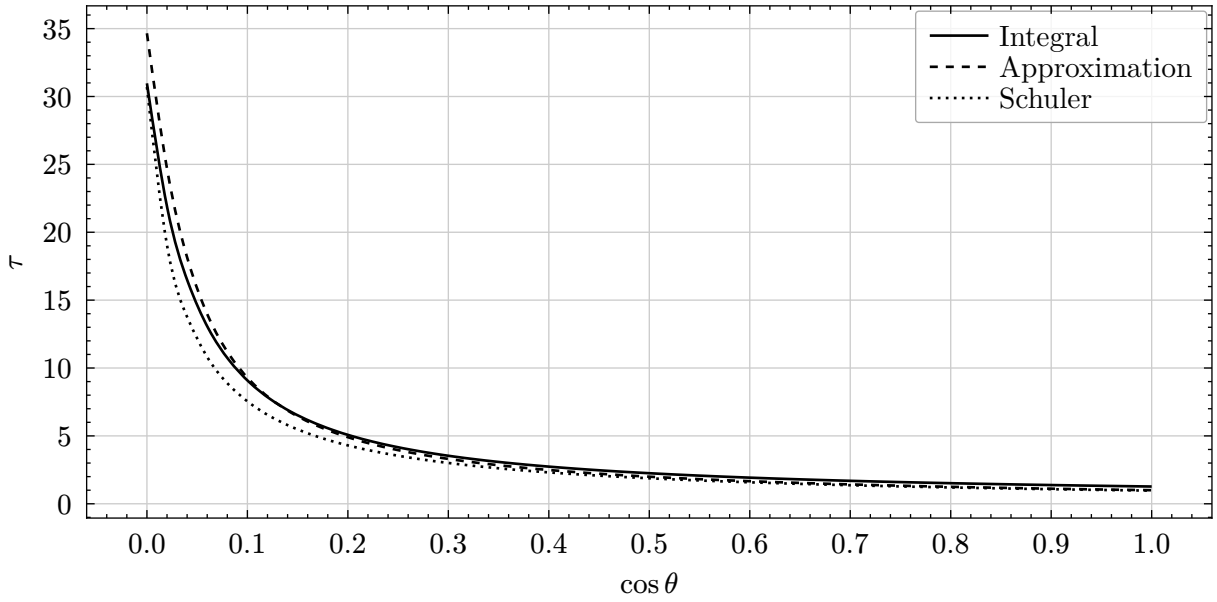



Figure 8: Scaled optical depth

Note that this approximation only works when the viewer is looking up, with $0° \leq \theta_v \leq 90°$.

(Rapp-Arrarás & Domingo-Santos, 2011) show that this approximation can also be applied to other optical depth profiles besides the exponential one.

From here, (Jodie, 2019) uses this to calculate the total amount of single scattering for a path in the atmosphere. With the following optical depths:

$$\tau = \beta_a \sqrt{(r + h_0)^2 - r^2 + r^2 \cos^2\theta} - \beta_a r \cos\theta \tag{50.}$$

Where $\theta$ is $\theta_v$ for a view ray, and $\theta_l$ for a light ray. As this approximates the normalized optical depth for a given altitude, calculating the optical depth at a different altitude $h$ means simply multiplying the optical depth by the medium density at the viewer altitude:

$$\tau = \beta_a \exp\left(-\frac{h}{h_0}\right)\left(\sqrt{(r + h + h_0)^2 - (r + h)^2 + (r + h)^2 \cos^2\theta} - (r + h)\cos\theta\right) \tag{51.}$$

The total single scattering can now be calculated as follows, using the scattering integral for flat homogeneous atmospheres:

$$L_{\text{scatter}} = \tau_{\text{view}} \beta_s F(\theta_s)\left(\frac{\exp(\beta_a \tau_{\text{view}}) - \exp\big(\beta_a \tau_{\text{light}}\big)}{\beta_a \tau_{\text{light}} - \beta_a \tau_{\text{view}}}\right) \tag{52.}$$

This provides a reasonably accurate approximation when the view ray is looking up.

## 10.3. Looking down

The optical depth approximation however does not work well when looking down, below the horizon. The initial approximation provided by (Schuler, 2012) has the same issue. Consider a viewer in the atmosphere looking in a direction where $\theta_v = 90°$:

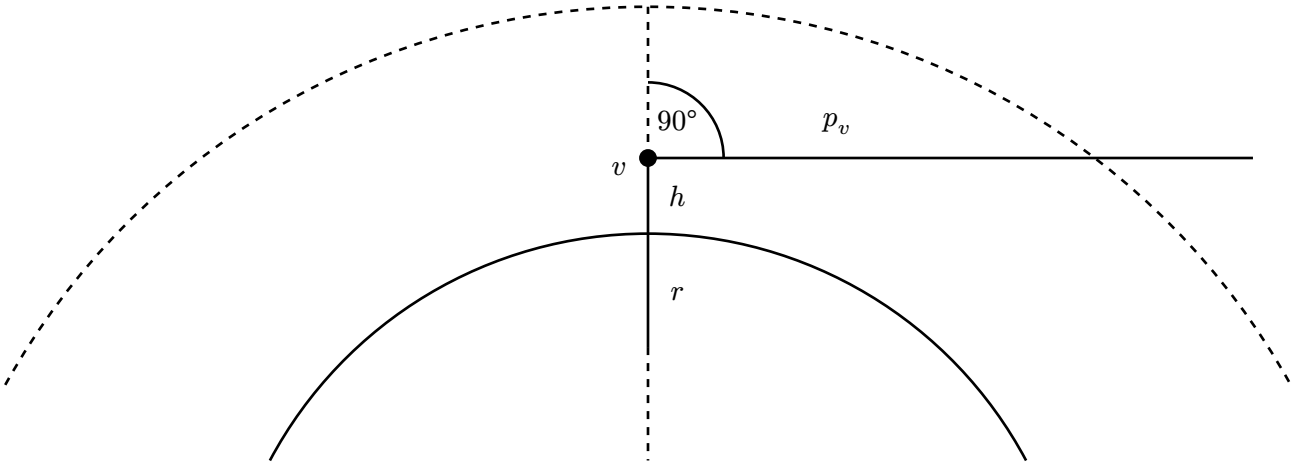



Figure 9: Viewer $v$ in an atmosphere, looking at the horizon

The optical depth of this view path $p_v$ is equal to half the optical depth of a viewer at an infinite distance away from the atmosphere, looking through it. This total optical depth can then be calculated as $2\tau$, as seen in the view ray $p_t$:

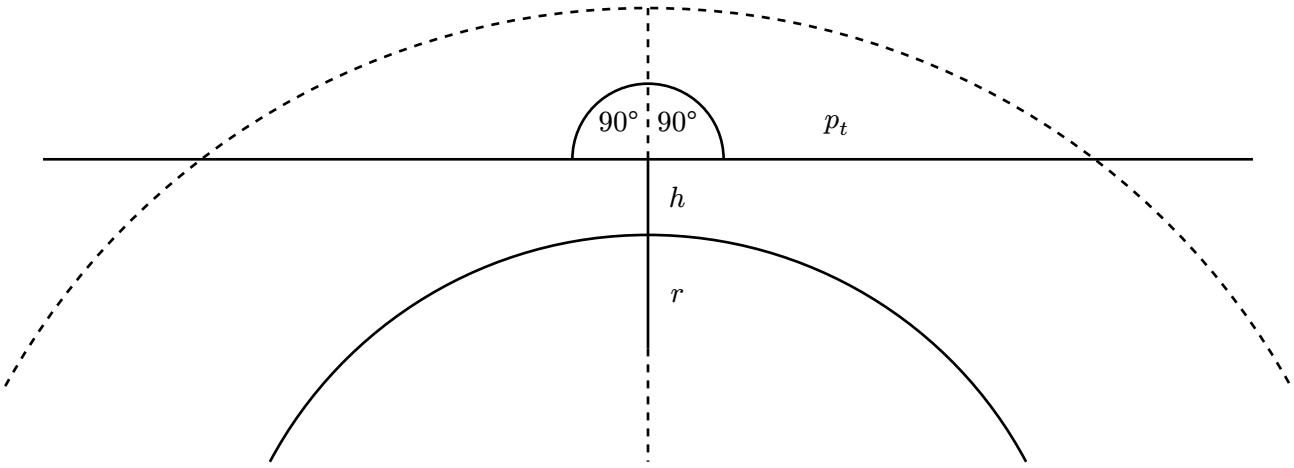



Figure 10: View ray for a viewer at an infinite distance away from the atmosphere

Now consider the viewer looking down at the atmosphere. In that case, the optical depth becomes the optical depth for the full view ray, from the closest point to the planet $c$, without the optical depth of the part of the ray $p_h$ behind the viewer, as seen below:

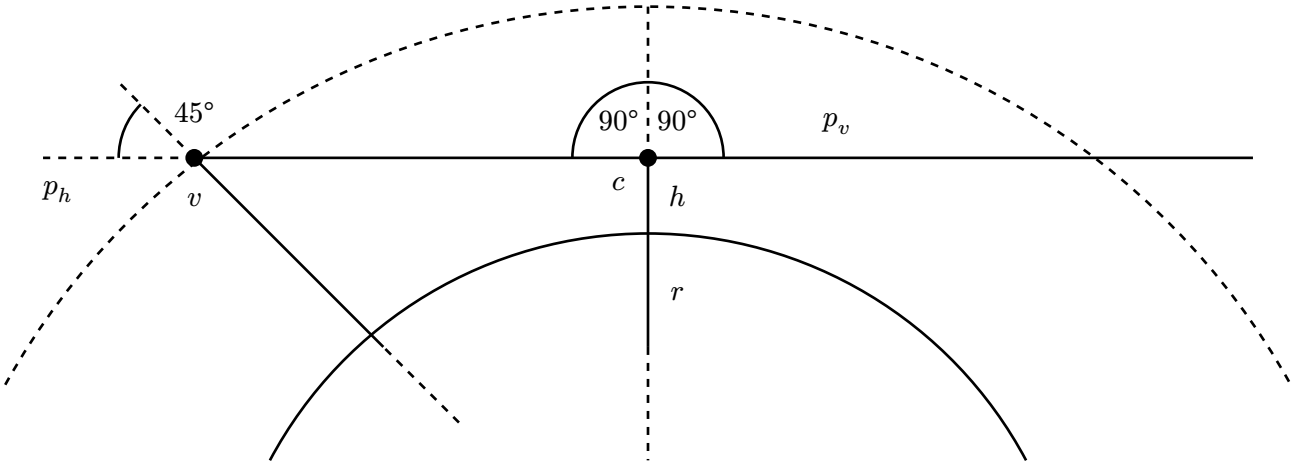



Figure 11: Viewer $v$ looking down

When written as an equation for the optical depth, this becomes:

$$\tau = 2\sqrt{(r+h_0)^2 - r^2} - \sqrt{(r+h_0)^2 - r^2 + r^2\cos^2\theta} - r\cos\theta \tag{53.}$$

Where $\theta$ is $\theta_v$ for a view ray, and $\theta_l$ for a light ray. Note that this equation does not take the density at the viewer $v$ or midpoint $c$'s altitude into account.

While this works for the optical depth used for transmittance, simply using the scattering integral Equation 52 when looking down no longer works, even when using the adjusted optical depth.

It is possible to split the scattering integral into two parts: one from the view ray's closest point to the planet center, which can be calculated as normal, and one from the viewer up to the closest point. This however gives the same result visually as not splitting up the integral. This means another method of calculating the scattering with a viewer from outside the atmosphere is needed.

## 10.4. Views from space

Besides functioning as an approximation to single scattering, the integral for the flat homogeneous atmosphere introduced in Equation 45 can be interpreted differently. Simplifying the integral, it becomes:

$$L_{\text{segment}} = \tau_{\text{segment}} \int_0^1 \exp(-t\tau_a - (1-t)\tau_b) \tag{54.}$$

Here, $\tau_{\text{segment}}$ can be interpreted as the optical depth over a segment in the atmosphere, $\tau_a$ the optical depth to the light at the start of the segment, and $\tau_b$ the optical depth at the end of the segment.

The radiance that then reaches the viewer is $L_{\text{segment}}$ multiplied by the transmittance from the viewer to the start of the segment. However, this can be directly incorporated into the integral, by adding the optical depth from the viewer to the start of the segment to both $\tau_a$ and $\tau_b$. The radiance at the viewer from this segment is then:

$$L_{\text{viewer}} = \tau_{\text{segment}} F(\theta_s) \frac{\exp(-\tau_a - \tau_{\text{view}}) - \exp(-\tau_b - \tau_{\text{view}})}{\tau_b - \tau_a} \tag{55.}$$

When this segment is used to represent the ray from the viewer towards the surface of the planet, it can directly approximate the scattering on that segment. This however does not work well from all viewer positions, especially when the viewer is far above the planet surface, or is near sunrise.

To resolve this, it is instead possible to use the segment scattering to calculate the scattering for a part of the view ray, and add them together. This new integration method resembles the one introduced in (Hillaire, 2015), which is as follows:

$$\int_0^d \exp(-\beta_a t) S \, \mathrm{d}t \tag{56.}$$

Where $d$ is the ray length, and $S$ is the scattered light at this sample position. This can be simplified to:

$$\frac{S - S\exp(-\beta_a d)}{\beta_a} \tag{57.}$$

## 10.5. Ozone layer

For ozone, (Bruneton, 2017a) and (Hillaire, 2020) use the following density:

$$\rho(h) = \max\left(0, 1 - \frac{|h - 25|}{15}\right) \tag{58.}$$

Where $h$ is in kilometers. While an analytical integral is likely possible, a simple approximation for the ozone layer can be set up as a constant density shell, starting at 17.5 kilometers from the planet surface, and ending at 32.5 kilometers from the surface.

This is then expressed as follows:

$$\rho(h) = \begin{cases} 1 \text{ if } 17.5 \leq h \leq 32.5 \\ 0 \text{ otherwise} \end{cases} \tag{59.}$$

The optical depth can then be calculated as the path length inside this shell, which is the difference between the positions along the ray of the spheres representing the bottom and top boundary of the shell.

### 10.6. Multiple scattering

Due to time constraints, multiple scattering is not considered. It may be possible to approximate it in a similar way as (Monzon et al., 2024) describes, as this provides an analytical solution assuming a constant density medium. This however does rely on an extra coefficient, $K_d$, which is a measured value, and would have to be calculated as well.

Another method of implementing multiple scattering is done in (Schuler, 2018). This depends on the delta-Eddington approximation, but not many details are given in the original code.

## 11. Evaluation

In order to see if the new models hold up to the existing ones presented in chapter 5, they have to be compared to these models. This is done by looking at the visual accuracy of the models to a path traced ground truth, as well as looking at their runtime performance and implementation complexity.

### 11.1. setup

As the intention is to run the atmosphere models on the GPU, the evaluation has to take this into account.

For this purpose, a simple tool to run shaders is developed. This shader runner uses the rust library `wgpu` for its GPU functionality. `wgpu` can use multiple graphics API's such as Vulkan, and DirectX.

It can render a provided shader, in `SPIR-V` format, or `WGSL`, to either an output texture, a buffer texture that can later be read by other shaders, or a volume texture that can be read by other shaders. It can then outputs the output textures to a specified image format. To preserve accuracy, the buffer textures are in 32-bit floating point `RGBA` format, (`Rgba32Float` in `wgpu`) and may be a 2D or 3D texture.

The tool can output images to the `png` file format, as well as the `exr` image format, and numpy's `npy` file format (Harris et al., 2020). The `npy` format is used as one of the image comparison tools fails to load the resulting `exr` images.

In order to measure the runtime of shaders, the shader runner can optionally output timings recorded using `wgpu`'s timestamp queries. These report the time difference from before and after issuing the draw call that renders the shader in the render pipeline.

When compiled, the tool includes the shaders to compare directly into the executable, so when the executable is run without any command-line arguments, it runs all the included shaders and copies the performance statistics to the clipboard. This allows for easier performance measurements on multiple computers. The link to the repository containing all relevant code can be found in appendix E.

### 11.2. Implemented models

In order to provide a good set of models to compare to, several of the models mentioned in Chapter 5 are implemented to compare against.

The parameters for the atmosphere were provided by the demo code from (Hillaire, 2020), and, when possible, the models have been set up so that they can use the parameters as provided there. As these parameters provide a planet radius, a planet reflecting no light is rendered as part of the atmosphere.

To keep the comparison simple, no spectral rendering is done, and it is assumed the parameters provided are valid for a linear sRGB color space. If needed, the models could be modified to provide spectral output, as described in (Bruneton, 2017b), as well as (Bruneton, 2017a).

Some of the models have been ported to work in *shadertoy* as well. Links to the shadertoy shaders can be found in appendix F.

#### 11.2.1. Empty shader

This shader only outputs a white image, and does no other calculations. It is used to measure any overhead in running the shader, besides the calculations that the shader has to perform.

#### 11.2.2. Path traced reference

To provide a ground truth to compare against, a path tracer is implemented, according to (Pharr et al., 2016) and (Fong et al., 2017). It implements exponentially decaying density for Mie and Rayleigh scattering, and a tent distribution for ozone absorption. For each color channel, 4096 samples are taken and averaged per pixel. 4 scattering events are considered per pixel, and no denoising is applied.

#### 11.2.3. Bruneton and Neyret

The implementation of the model presented in (Bruneton & Neyret, 2008) is the one provided in (Bruneton, 2017a), translated to work in the shader runner. It calculates 4 scattering orders. While the provided implementation allows calculating spectral radiance as well, this is not used, as the parameters provided are for linear sRGB instead.

#### 11.2.4. Hillaire

This is an implementation of the model presented in (Hillaire, 2020), ported from the code supplied with the paper. It does not implement any of the frustum grid to compute scattering, and instead falls back to raymarching when the camera is outside of the atmosphere.

#### 11.2.5. Preetham, Shirley and Smits

This is an implementation of the model presented in (Preetham et al., 1999). As this is a model already fitted to another atmosphere, it does not use any of the parameters provided, and thus matches the resulting atmospheres less well. It is also only capable of ground views, and thus cannot be used for comparisons to higher altitude and space views.

#### 11.2.6. Naive

The naive atmosphere is an implementation of the atmospheric scattering equations by direct numerical integration. It is a modified version of (Terrell, 2016), adding ozone absorption in order to match the other implementations.

#### 11.2.7. Schuler

This model is a derivation of the naive model, replacing the inner loop that integrates the optical depth from the sample point towards the light, with the Chapman function approximation as described in (Schuler, 2012). As the approximation only works for an

exponentially decaying atmosphere, this version of the model does not incorporate ozone absorption. Note that while an implementation was provided as supplementary material to (Schuler, 2012), this version could not be adapted to work with the given parameters.

#### 11.2.8. Neural network

This is an implementation of atmospheric scattering as described earlier in chapter 9.6. The parameters used are a result of one of several training runs, as some training runs resulted in a less accurate fit.

#### 11.2.9. Flat

This is an implementation of the spherical, homogeneous atmosphere as described earlier in chapter 10.1, as well as (Jodie, 2019). When looking down, and the view ray hits the planet, the scattering is assumed to take place in a segment, as described in chapter 10.4.

Note that the ozone layer is modeled as a constant density shell in this model, instead of using the tent function the other models use. This results in a slight mismatch in some view angles.

#### 11.2.10. Raymarched

This is an implementation using the scattering integral as described in chapter 10.4, as a replacement for the naive integrator used in (Terrell, 2016). For the optical depth, approximation from chapter 10.1 is used. The number of integration steps is set to 5, as this provides a good tradeoff between quality and steps required.

At with the flat model, tho ozone layer for this model is also modeled as a constant density shell, and may thus also appear different.

### 11.3. Transmittance

Besides comparing the scattering results of the models, they can also be used to calculate transmittance. As most models implemented use similar methods to calculate transmittance, only the new neural network, and flat atmosphere models are considered.

#### 11.3.1. Reference

This implements a naive Riemann integrator to calculate the optical depth. The number of steps is intentionally set high, to 128 steps, to improve accuracy.

#### 11.3.2. Neural network

This uses the transmittance neural network as described in chapter 9.6, instead of the scattering neural network. When the view ray hits the planet, the transmittance is calculated for the segment in a similar manner as (Bruneton & Neyret, 2008), as the neural network only provides the transmittance for a full view ray.

#### 11.3.3. Flat

This calculates the optical depth as described in chapter 10.1, with equation 50:

$$\tau = \beta_a \sqrt{\left(r + h_0\right)^2 - r^2 + r^2 \cos^2\theta} - \beta_a r \cos\theta \qquad 50.$$

The optical depth for the ozone layer is calculated as described in chapter 10.5.

If the view ray hits the planet surface, the optical depth for the segment between the viewer and planet surface is computed in a similar manner as the optical depth when looking down.

However, the point on the planet surface is used as midpoint, instead of the point closest to the planet center.

The optical depth is then used with Beer's law to calculate the transmittance:

$$T = \exp(-\tau) \tag{60.}$$

## 11.4. viewer, light, and exposure

When comparing, all models are rendered from different viewer perspectives. The following viewer positions are used:

- ground, directly on the planet surface.
- plane, at 5 kilometers above the planet surface.
- orbit, at 100 kilometers above the planet surface, then moved 1000 kilometers backwards, to give a better overview.
- space, showing the planet and atmosphere in full. The sun is positioned to the top right of the viewer, showing both the illuminated and dark side of the planet and atmosphere.

For the ground, plane and orbit views, the following times of day are used for the viewer:

- Dawn, with the sun towards the viewer, at 6° below the horizon
- Sunrise, with the sun towards the viewer, at 6° above the horizon
- Noon, with the sun directly above the viewer.

A notable feature of the atmosphere is the planet shadow, where the planet itself casts a shadow into the atmosphere. To view this, two extra viewing positions are rendered as well, looking with the sun to the right side of the viewer, 3° below the horizon. One viewer is positioned on the planet surface, and one is positioned 200 kilometers above the surface.

The sun, as light source, uses an irradiance of 6, for the `r`, `g` and `b` color channels. No physical unit is used here.

For all view positions, a different camera exposure is used to map the radiance to a final per-pixel color value that allows viewing the image without adjusting the brightness later on. Exposure is implemented as simply multiplying the radiance by the exposure value. For most view and light combinations, this value is 1. When the time of day is set to "dawn", an exposure of 16 is used. For the "space" viewpoint, an exposure of 2 is used. For the planet shadow views, an exposure of 32 is used.

## 11.5. Visual comparison

For all viewers and light combinations, each model is then rendered to an 1080 by 1920 images, and stored to disk as a `png` file, with $t(x) = 1.0 - \exp(-x)$ applied per color channel as tonemap operator, and converted from linear sRGB to nonlinear sRGB. Besides this, the image is also stored as an `exr` image, as well as numpy's `npy` file format, storing the color channels directly as 32 bit floating point numbers, in sRGB linear, without any other processing applied.

For comparing the results of the scattering models, the path tracing model is used as reference. All other models are then compared to the path tracer, for each viewer and light combination.

Comparison is done using ꟻLIP, in high dynamic range (Andersson et al., 2021). As ꟻLIP is unable to load the `exr` images produced by the shader tool, they are loaded in the `npy` file format. ꟻLIP in high dynamic range also silently fails, and returns a mean error, as well as error map consisting of only the value 0 if there is any black pixel in the input image. For

this reason, a value of $1 \cdot 10^{-9}$ is added to the color channels of every pixel. The mean error that ꟻLIP reports is then written out to a CSV file, and the error map is written out to another image.

The second error metric that is used is the Root-Mean Square Error, or RMSE for short:

$$\text{RMSE} = \sqrt{\sum_{i=0}^{N} \frac{\left(v_{i,m} - v_{i,t}\right)^2}{N}} \tag{61.}$$

Where $v_{i,m}$ is the $i$th sample of the model to test, and $v_{i,t}$ is the $i$th sample of the reference model.

This is applied separately for each color channel, for each pixel in the image. This is then converted into an image, and stored. The mean error for each image is also stored.

This process is repeated for the transmittance models, where the reference transmittance model is compared against the other transmittance models. However, as the transmittance values range between 0 and 1, ꟻLIP is used with low dynamic range instead (Andersson et al., 2020). Note that using ꟻLIP here is not entirely correct, as the transmittance values are never displayed directly. ꟻLIP however does provide a good error map to see where the transmittance differs.

## 11.6. Performance comparison

To measure runtime performance, `wgpu`'s timestamp queries are used. These record the time at the point they are inserted into the render pipeline. In the shader runner, one is inserted before the draw call to render the shader, and one after this draw call. The difference between the two timestamps then corresponds to the time it took the shader to run. To attempt to make the results more consistent, the shader is run 512 times. The last 256 times, the timestamps are recorded for profiling. The average time of these 255 runs is then used as time the model runs in.

For the models presented in (Bruneton & Neyret, 2008) and (Hillaire, 2020), the time needed for precomputation is recorded separately. However, for (Hillaire, 2020), the scattering table is assumed to be part of the final render, as this table needs to be recomputed every time the viewer height, or the angle with the sun changes.

Hillaire's model is not able to use the lookup table when the viewer is outside the atmosphere. This causes it to have different performance characteristics in this case.

## 11.7. Implementation complexity

The last method to compare models with is by implementation complexity. This is a subjective measure of how complex the method is to implement both as a shader, as well as any support code needed for precomputation. As a general rule applied here, more lines of code indicate a more complex implementation. In this case, the significant lines of code are counted. These are any line of code that satisfies the following:

- is not empty.
- is not a comment.
- is not punctuation, such as `}` and `);`, which are common at the end of function calls and function definitions.
- contributes to the final output, and is not setup code, such as uniforms.

Requiring some kind of precomputation also increases complexity, as it requires setting up extra code to render to textures, and then use these textures from the shader. The difficulty of this depends on the renderer that the final model is implemented in.

While the neural network based model does not do any precomputation in the same way as the models presented in (Bruneton & Neyret, 2008) and (Hillaire, 2020), it does require training a neural network, which adds to the complexity needed to make the final model.

# 12. Results

Here, the models are compared as described in the previous chapter.

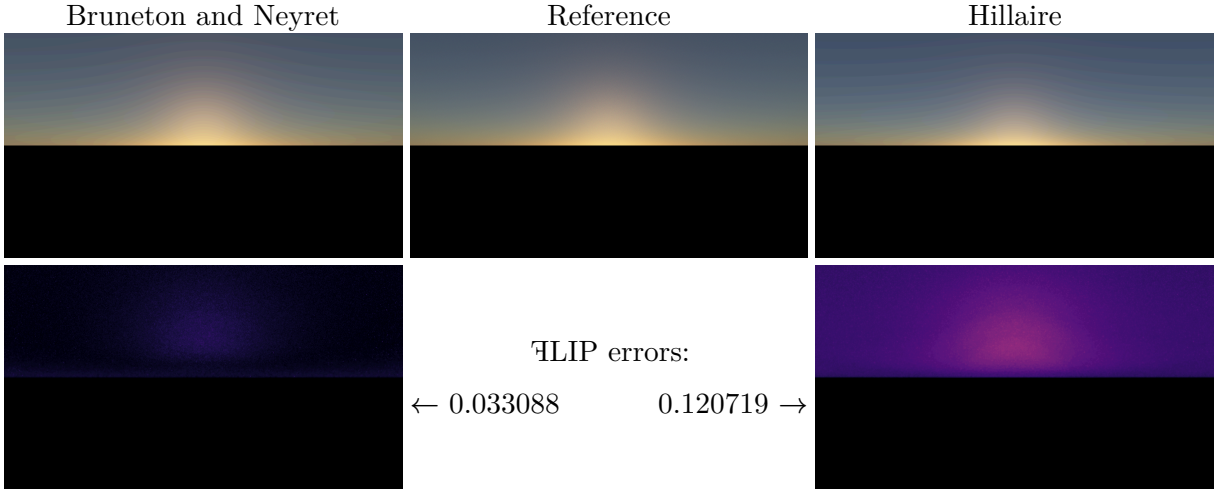
## 12.1. Visual

Here, the models are compared visually. For a full table of how each model compares to the path traced reference, according to the ꟻLIP and RMSE metrics, see Appendix A and Appendix B. Due to space constraints, not all models are compared to the path tracer for each view. Instead, the models are only compared where the results differ more from the path tracer, as well as when there is a failure mode of the model that can be highlighted.

### 12.1.1. Bruneton and Neyret, Hillaire

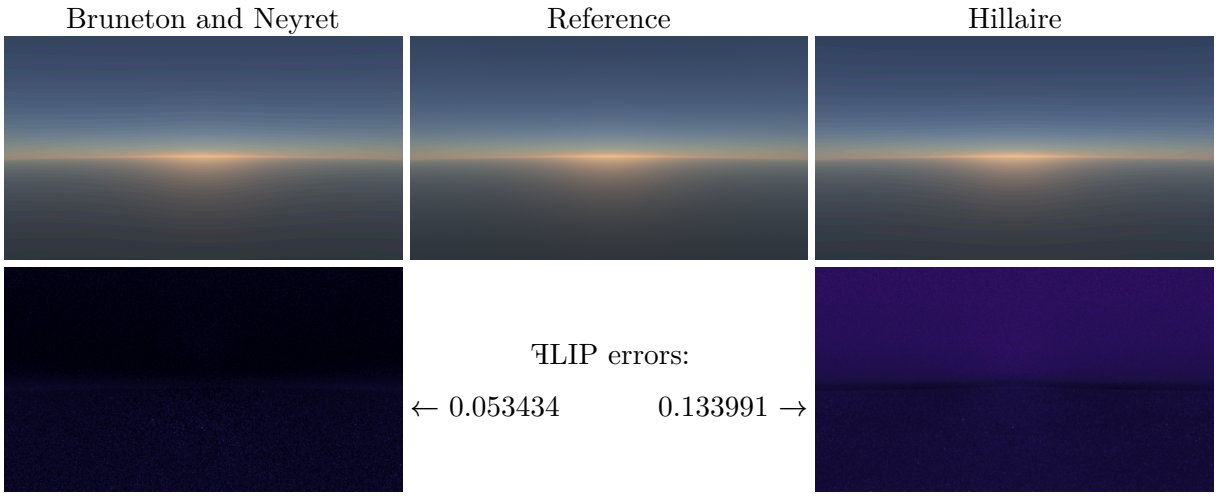
As both Bruneton and Neyret's and Hillaire's model implement multiple scattering, they come very close to the reference path tracer.

Hillaire's model gives a slightly different halo around the sun, caused by Mie scattering. This is likely due to the multiple scattering approximation assuming the phase function is isotropic. This is most noticeable from the ground sunrise view:



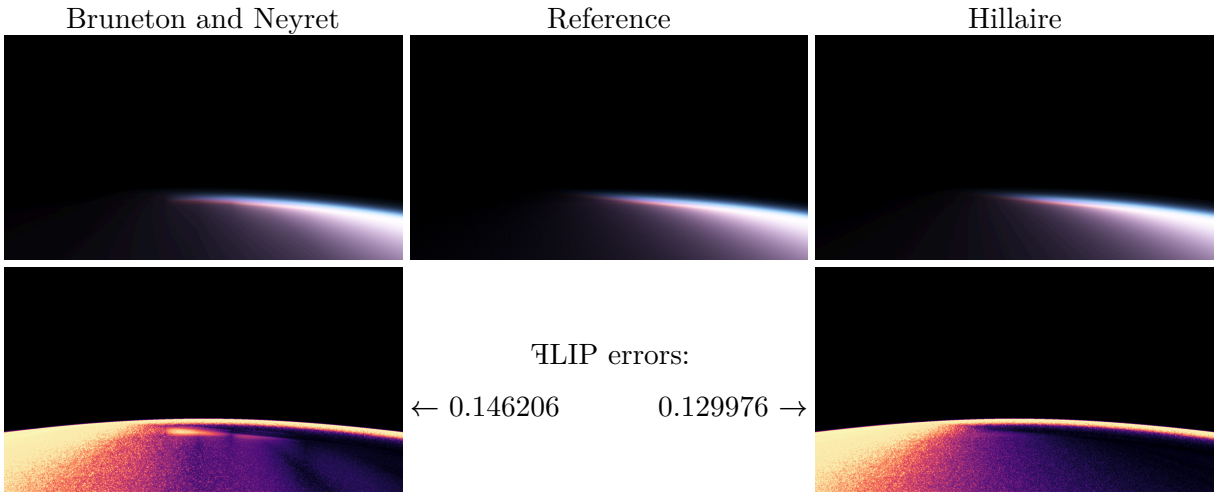

Comparison 12: Ground

This is less evident from a higher altitude:




Comparison 13: Elevated viewer at 5km height from the surface

However, Bruneton and Neyret's model does not handle the planet shadow viewed from orbit very well, as there is a blocky pattern near the terminator. Hillaire's model does not have this issue:

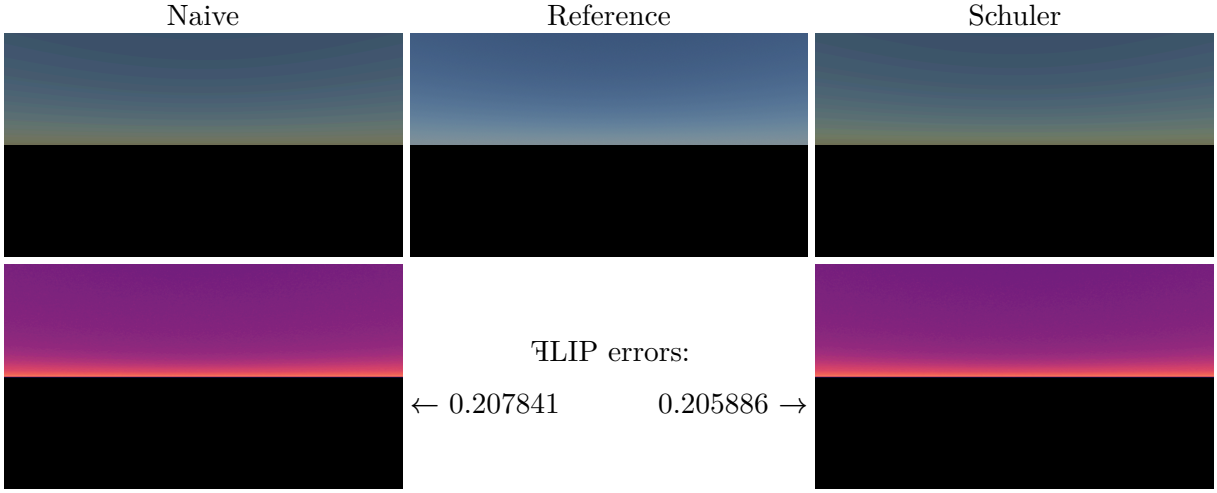



Comparison 14: From orbit

### 12.1.2. Naive and Schuler

The Naive model and Schuler's model do not implement multiple scattering, and will thus look different from the path traced reference. On top of this, Schuler's model does not implement ozone absorption, and thus has different colors as well.

The lack of multiple scattering is evident as an overall darkened atmosphere:

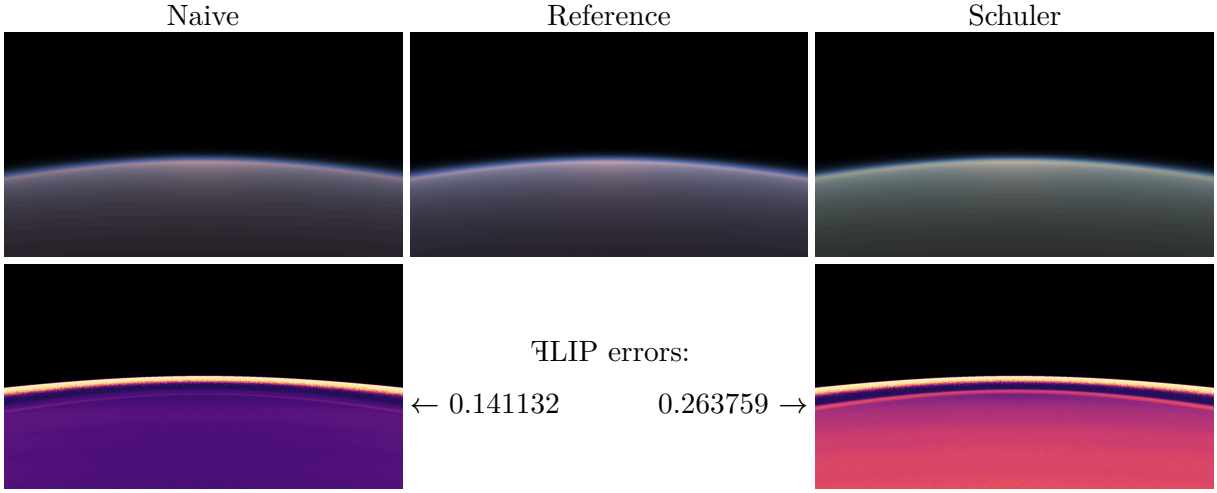



Comparison 15: Ground

The lack of ozone in Schuler's model becomes more apparent during sunrise, resulting in the sky appearing less red:

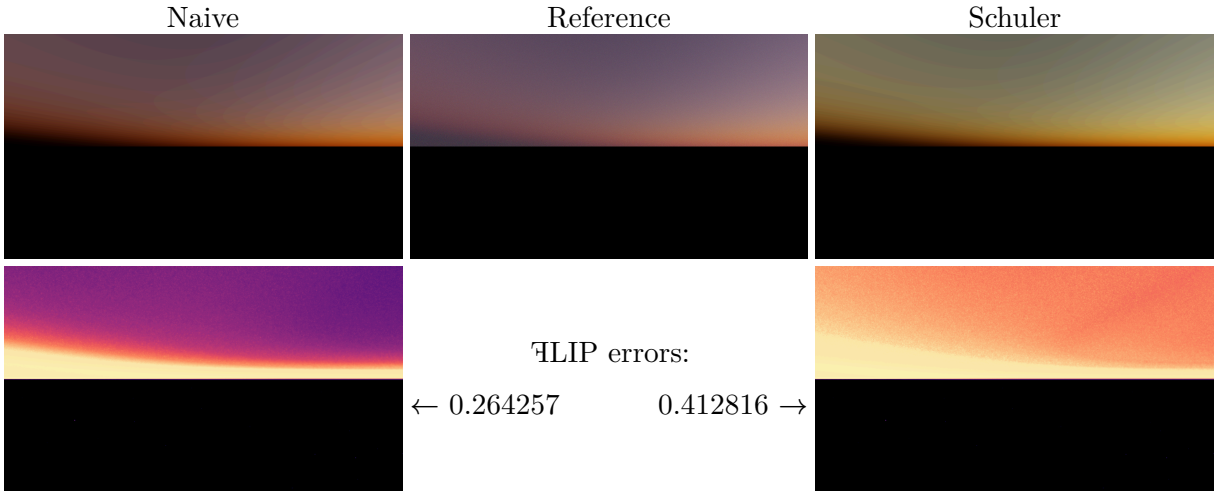

Comparison 16: Ground

The lack of ozone is visible even more when looking from space:




Comparison 17: Ground

The lack of multiple scattering becomes most evident when viewing the planet shadow from the ground, as it illuminates the shadowed part of the atmosphere more, as seen on the left in the images below:

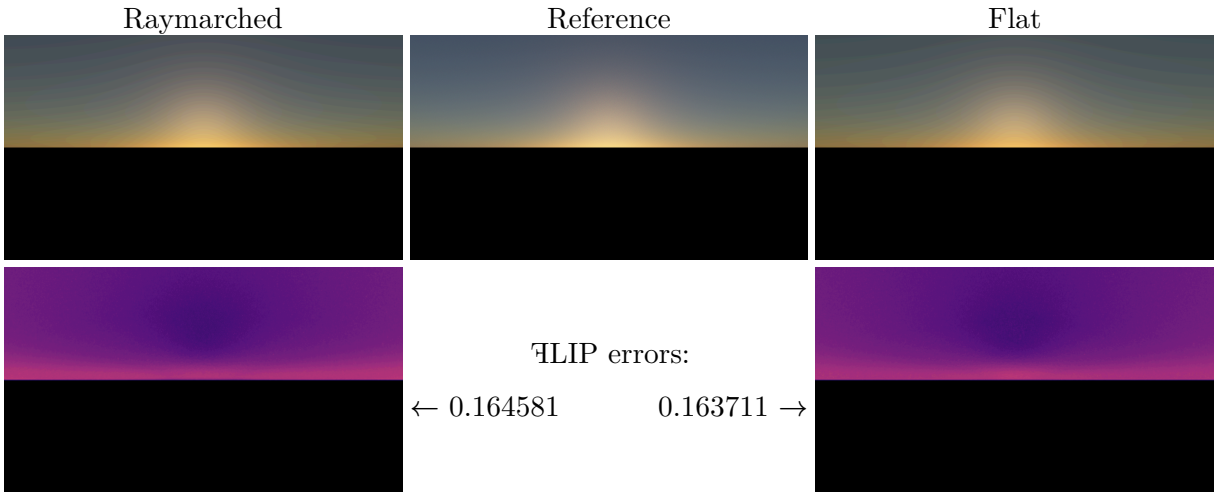



Comparison 18: Planet shadow, viewed from the ground

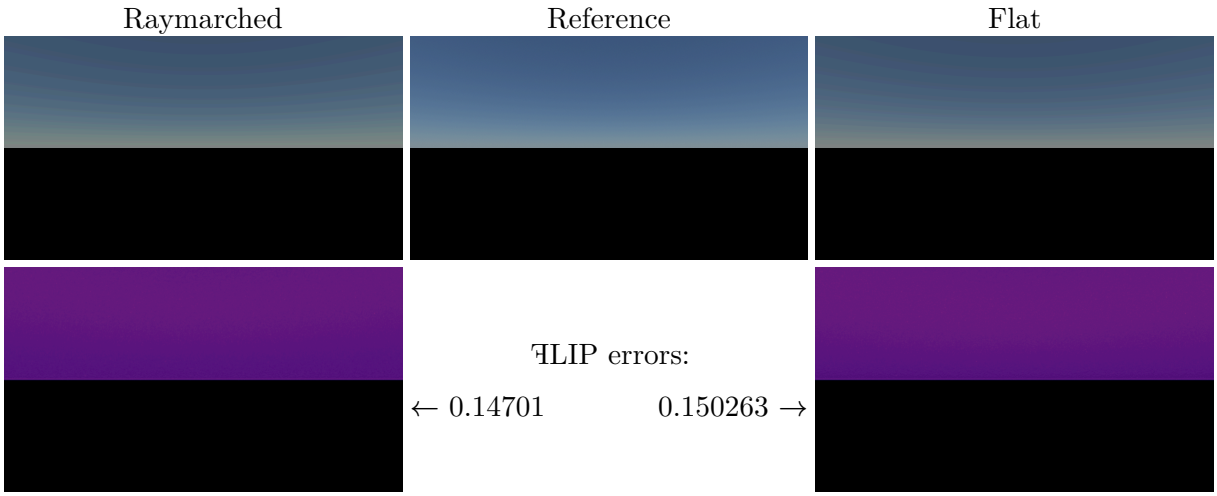
### 12.1.3. Raymarched and Flat

The raymarched and flat models only implement single scattering. As the flat model assumes it starts inside the atmosphere, it does not work well when the viewer is outside the atmosphere.

Inside the atmosphere, they are very similar however:

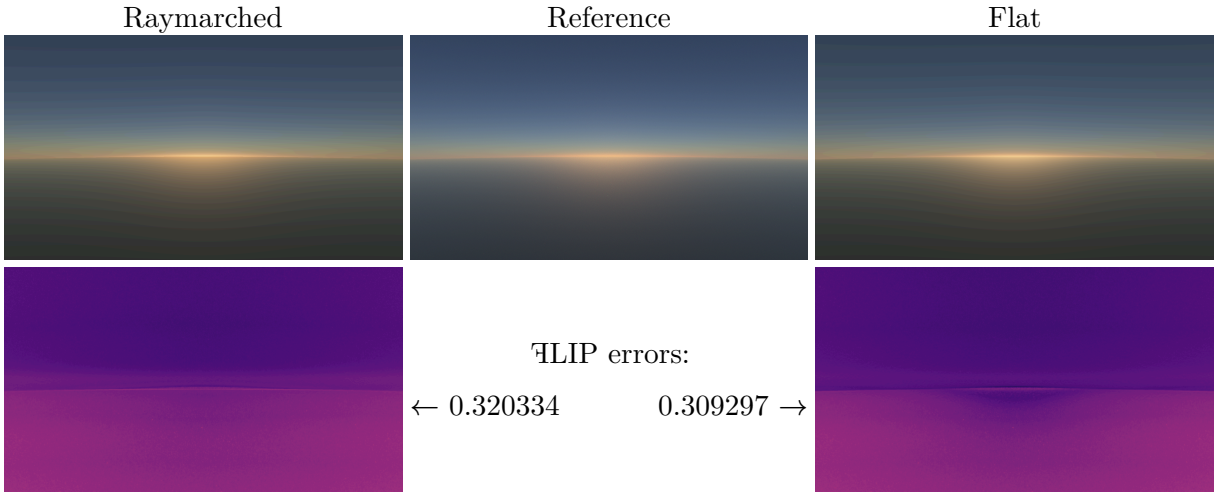



Comparison 19: Sunrise from the ground

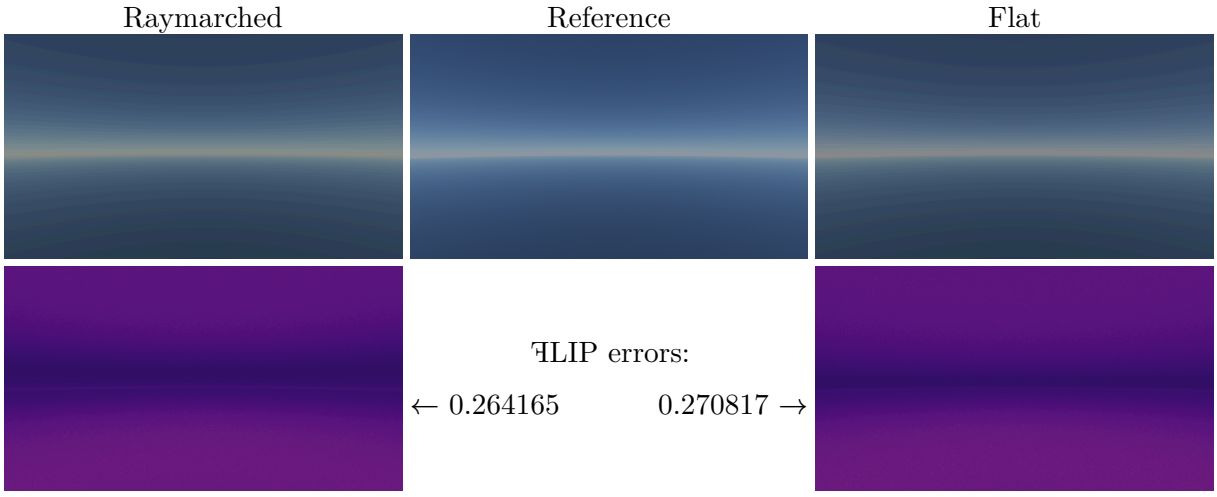



Comparison 20: Noon from the ground

The flat model still remains mostly similar at an elevated viewing position:

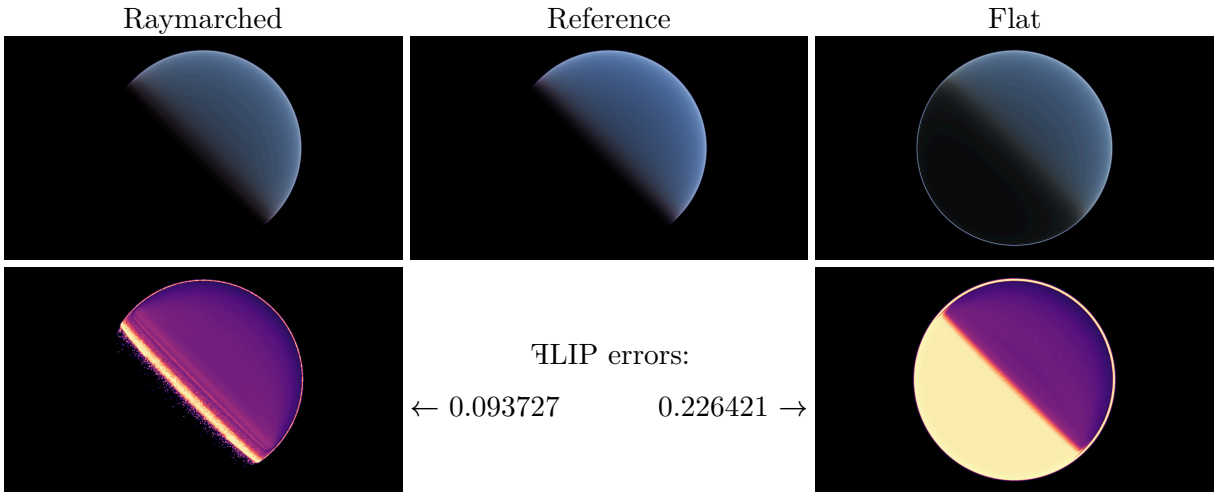



Comparison 21: Sunrise from 5km above the surface



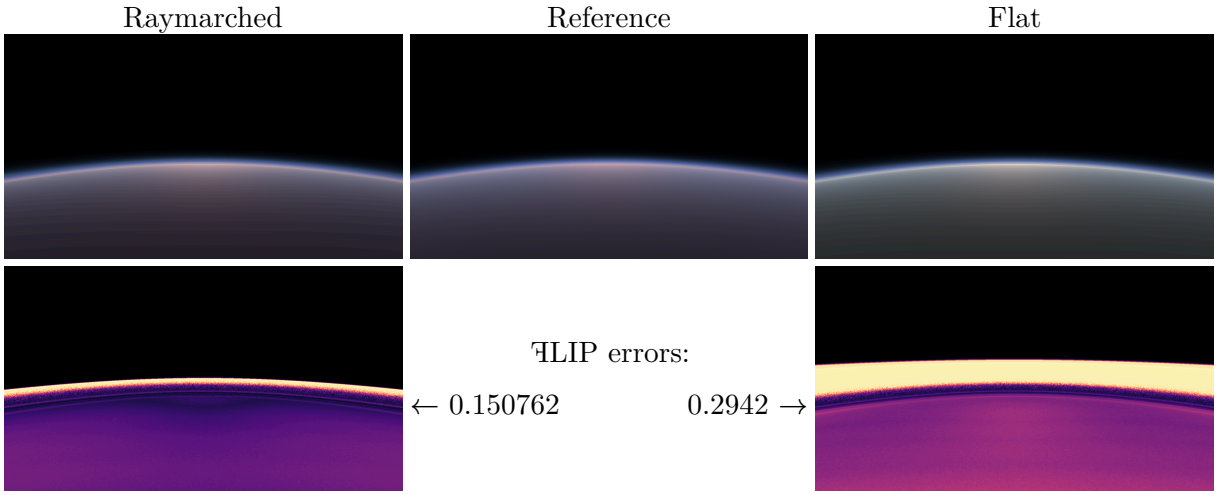

Comparison 22: Noon from 5km above the surface

The flat model however breaks down from space views. Notice that the raymarched model also has difficulty representing the transition into the shaded area of the planet from this view, with the step count of 5 used:



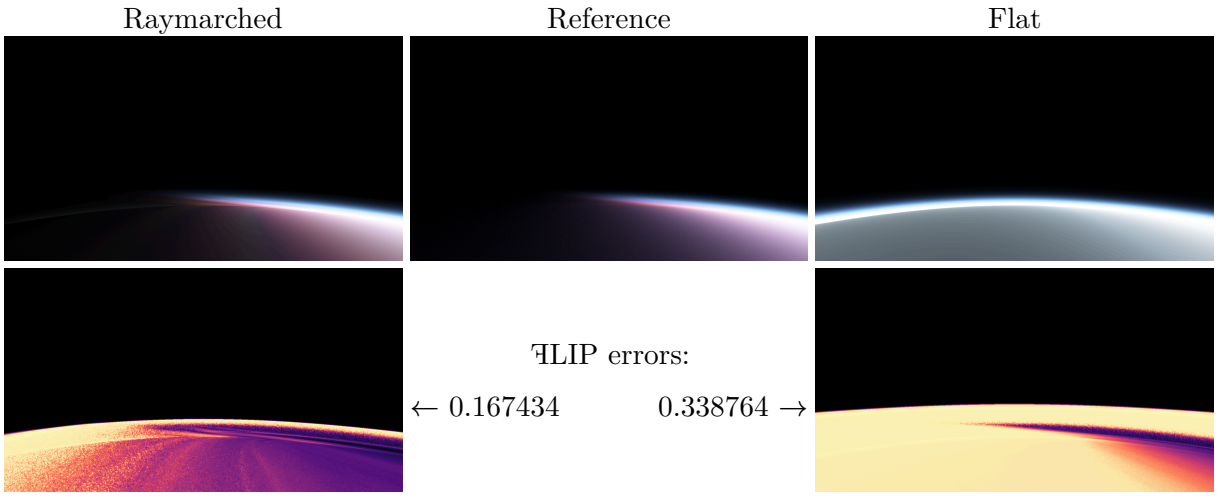

Comparison 23: Far view from space

When used from a viewer with lower elevation, the flat model still fails when viewed from outside the atmosphere, and loses part of the color caused by ozone absorption:




Comparison 24: Sunrise from space

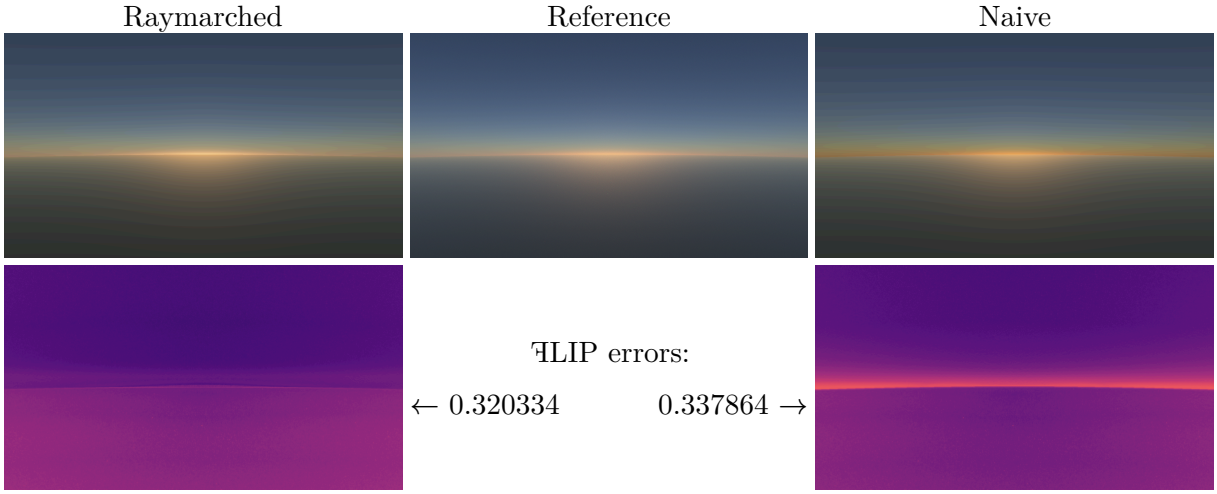
With the low step count used, the raymarched model also does not represent the planet shadow well. Note that the flat model fails completely here, as it does not take the planet shadow into account at all:




Comparison 25: Planet shadow, viewed from 200km above the surface

### 12.1.4. Raymarched and Naive

However, due to the improved integrator, the raymarched model does improve over the naive model in some cases, such as when viewing the sunrise. Note how the horizon is not darkened in the raymarched model, and more closely matched the reference:

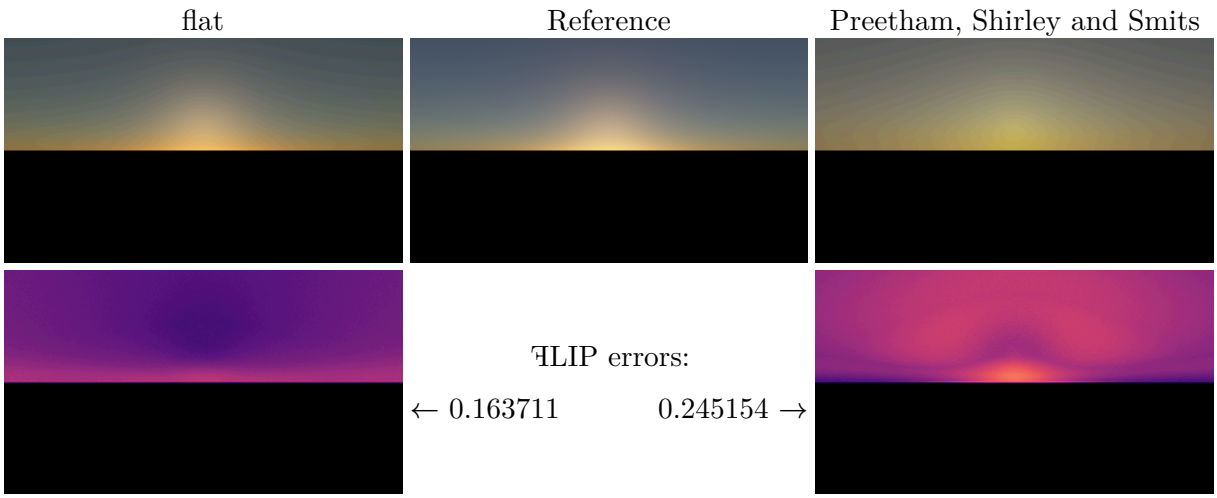



Comparison 26: Sunrise viewed 5km from the surface

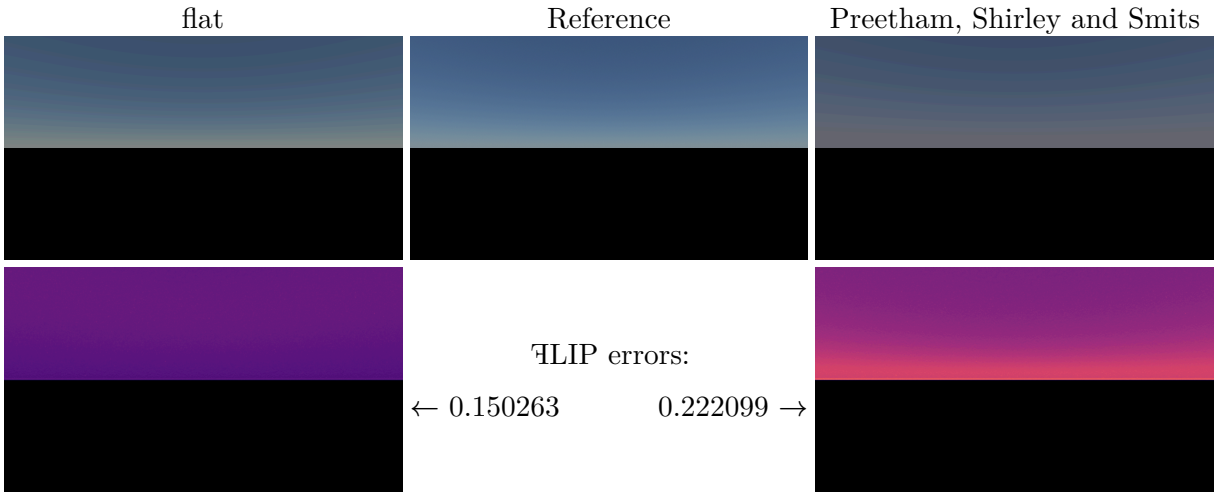
The naive model has a noticeable darkening around the horizon, that both the reference path tracer, and the raymarched model do not have. While this can be resolved by increasing the number of integration steps in the naive model, the improved integrator of the raymarched model does not require this.

### 12.1.5. Flat and Preetham, Shirley and Smits

As Preetham, Shirley and Smits' model is not fitted on the reference path tracer, it will look different. However, it does provide a good baseline for a fitted model that works only close to the ground. It still mostly gives the same image for a ground-bound viewer:

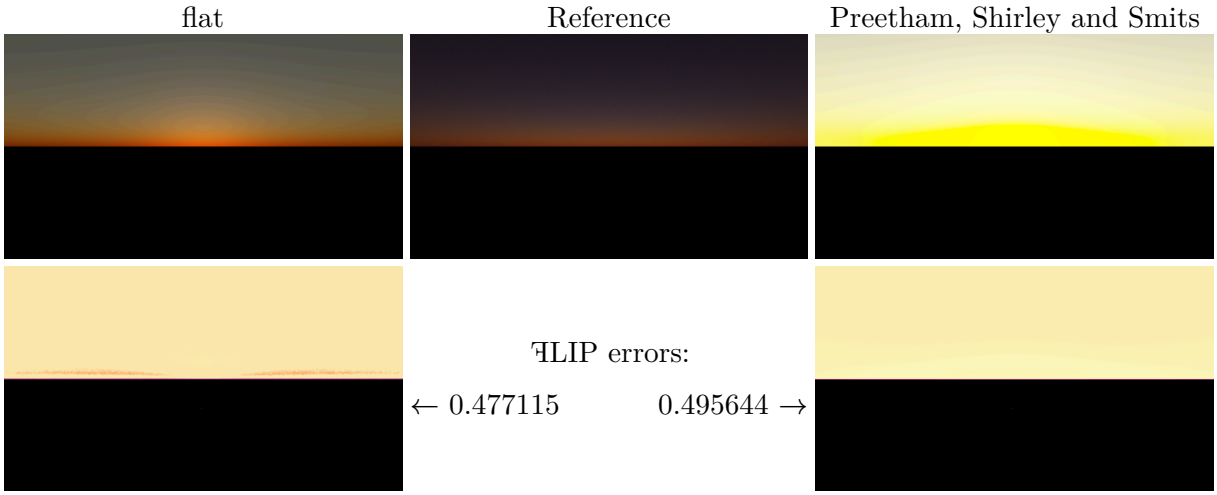



Comparison 27: Sunrise from the ground



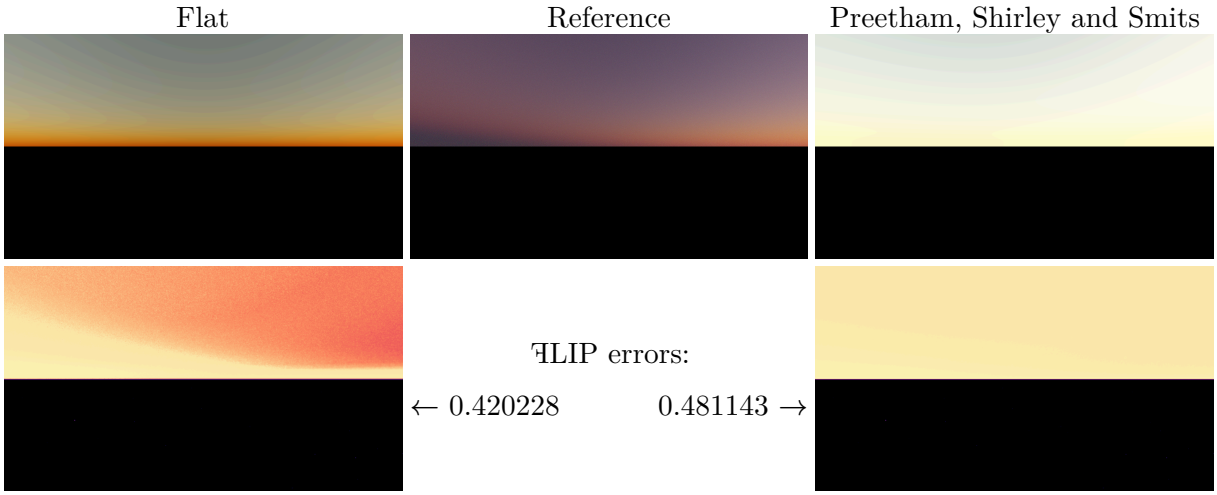

Comparison 28: Noon from the ground

However, as it's not made to work after sunset, it fails here. The flat model also fails in this case, as it does not properly take the planet shadow into account:




Comparison 29: Sun below the horizon, from the ground

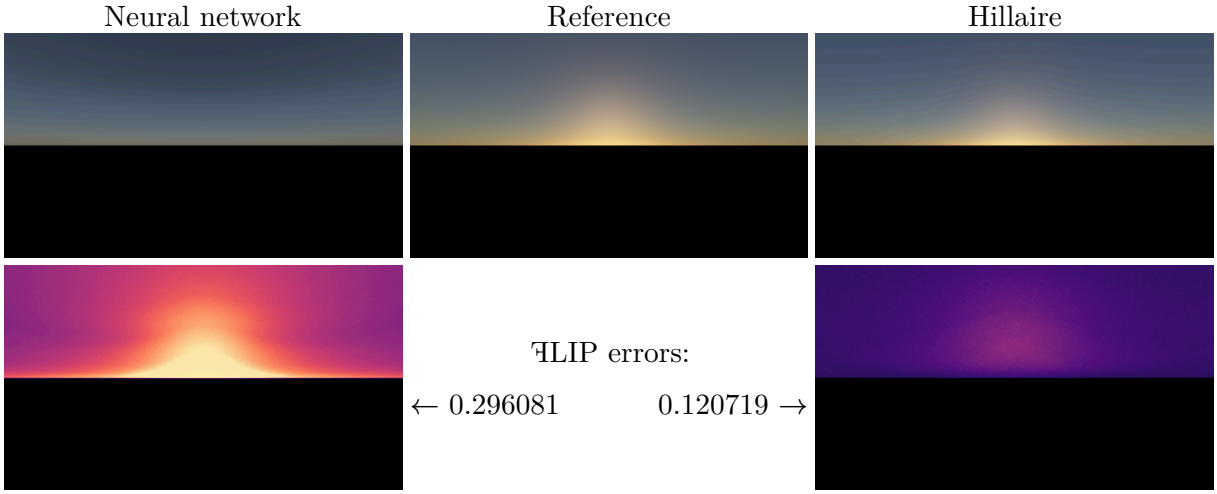
Both also fail to represent the planet shadow:




Comparison 30: Planet shadow viewed from the ground

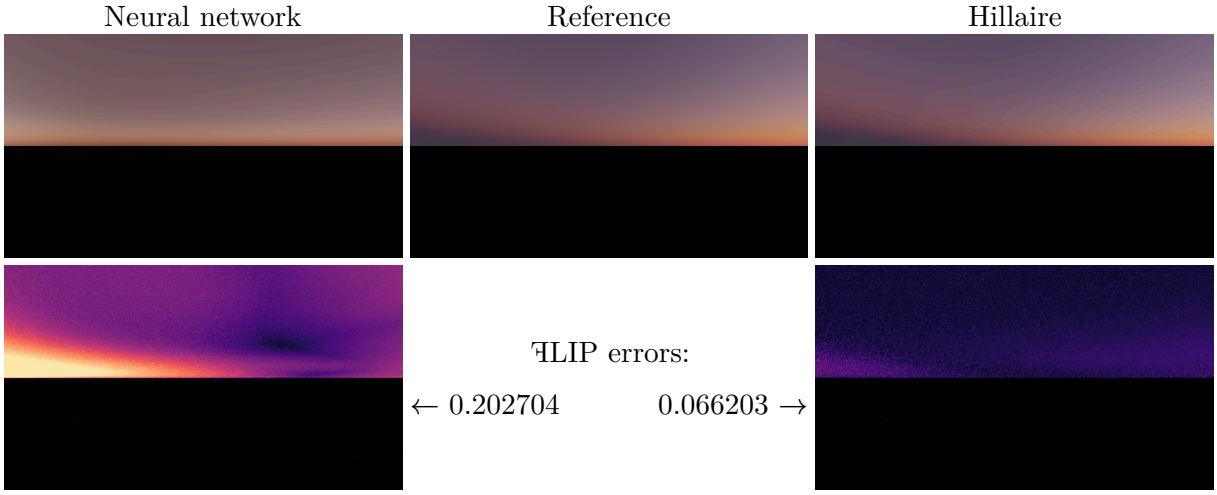
### 12.1.6. Neural network

The new fitted model based on neural networks largely reproduces the right colors of the reference path tracer, but does not correctly reconstruct details, such as the Mie scattering halo:



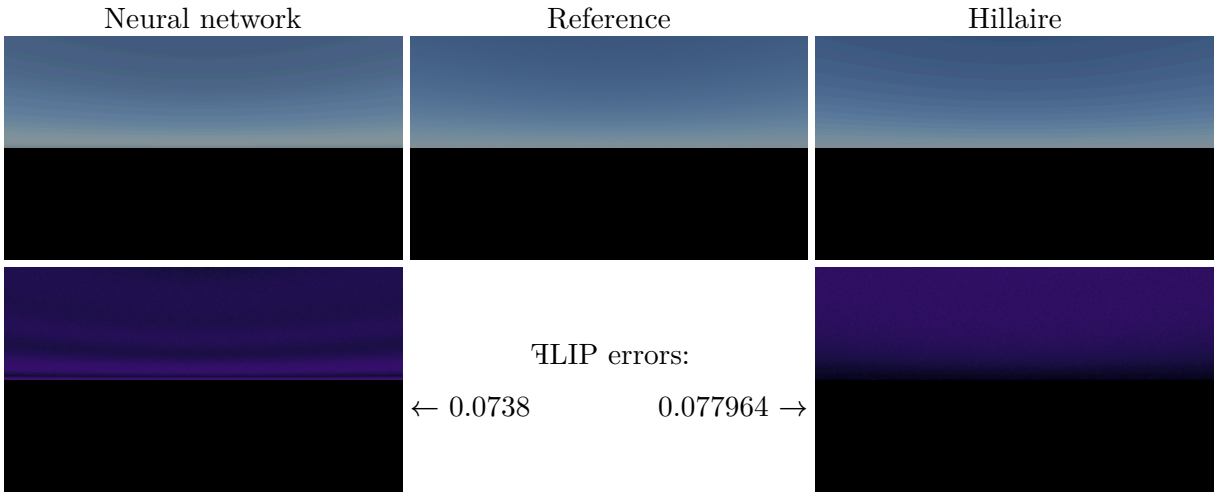

Comparison 31: Sunrise from the ground

Or the planet shadow:



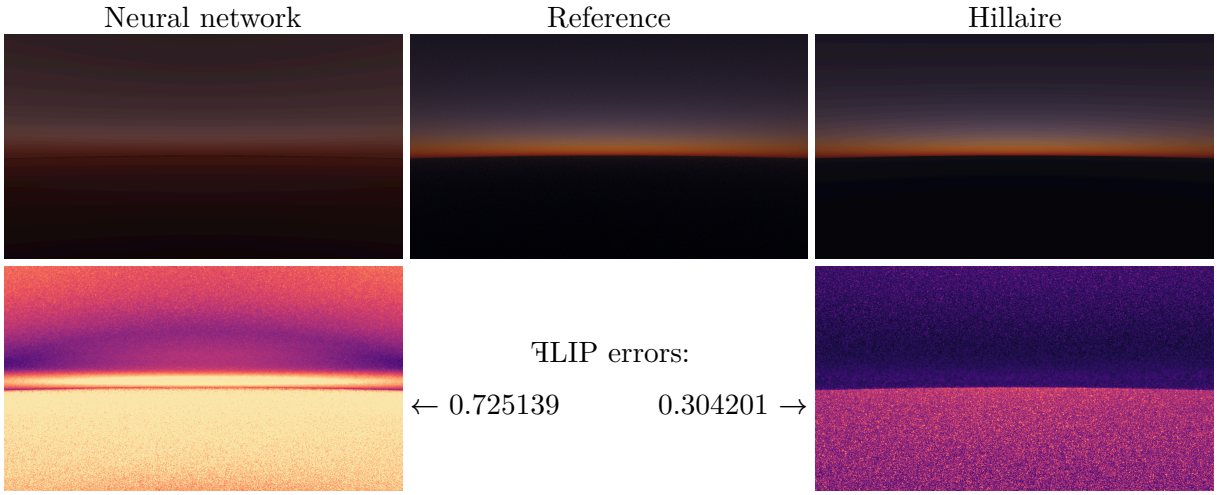

Comparison 32: Planet shadow from the ground

This is less noticeable during daytime:



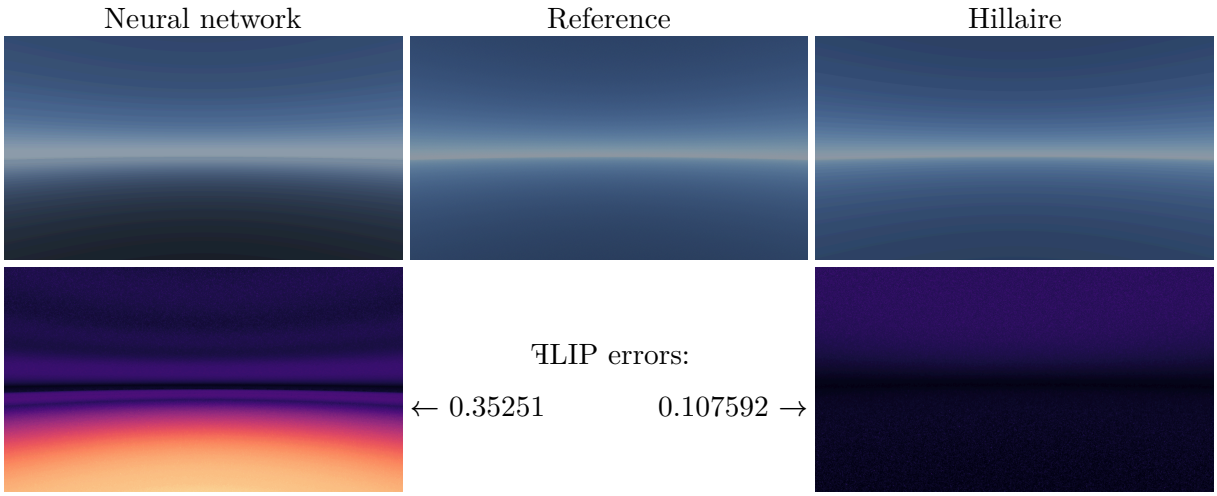

Comparison 33: Noon from the ground

Due to the model not being able to reconstruct finer details, the neural network model does not represent dawn very well:



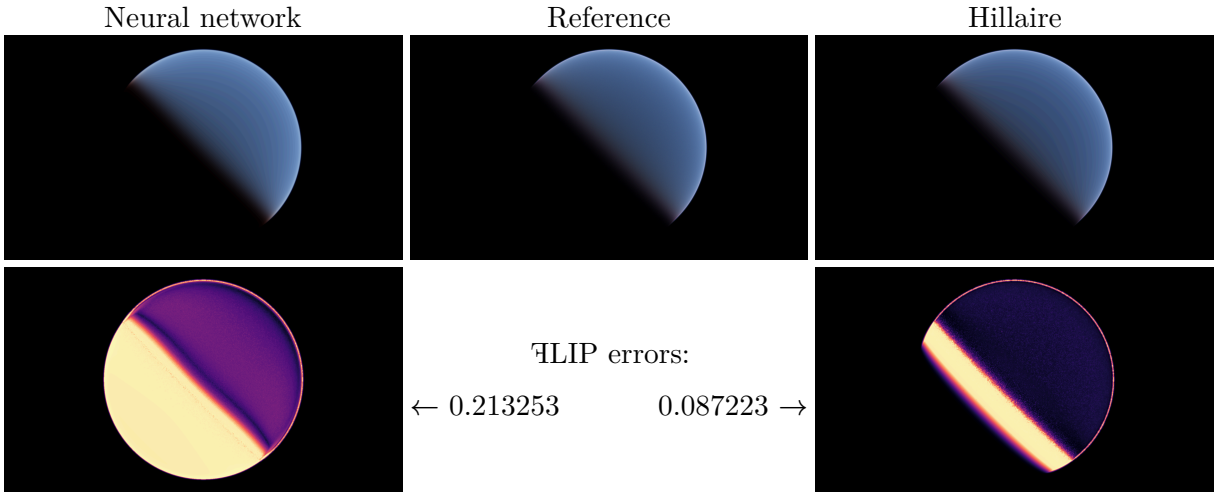

Comparison 34: Sun below the horizon, 5km above the surface

From an elevated viewing position, it also does not represent the light scattered between the planet surface and the viewer well, and makes it too dark:




Comparison 35: Noon, 5km above the surface

When viewed from space, the neural network appears to make the edges of the atmosphere thicker:



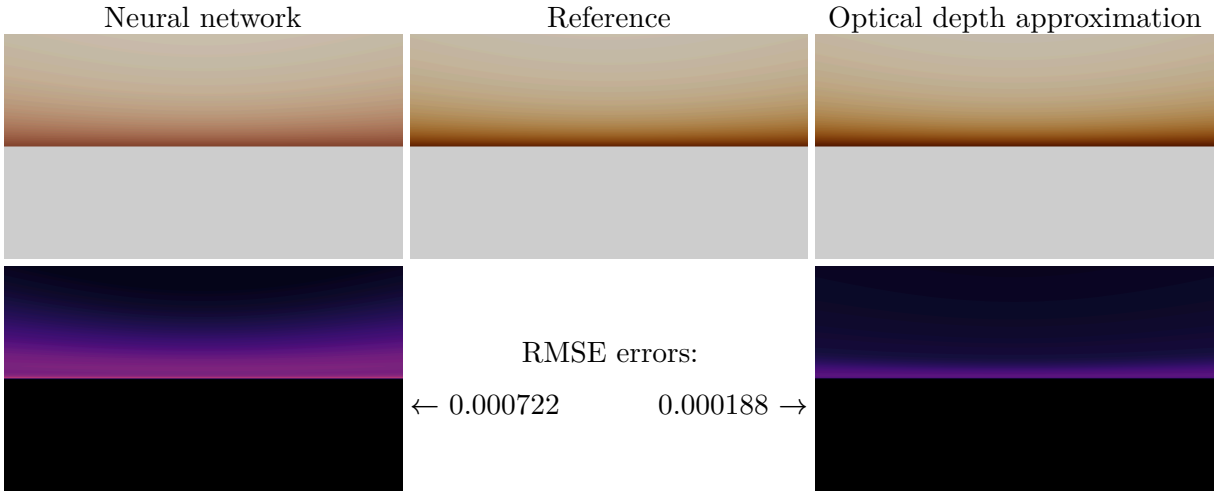

Comparison 36: From space

## 12.2. Transmittance

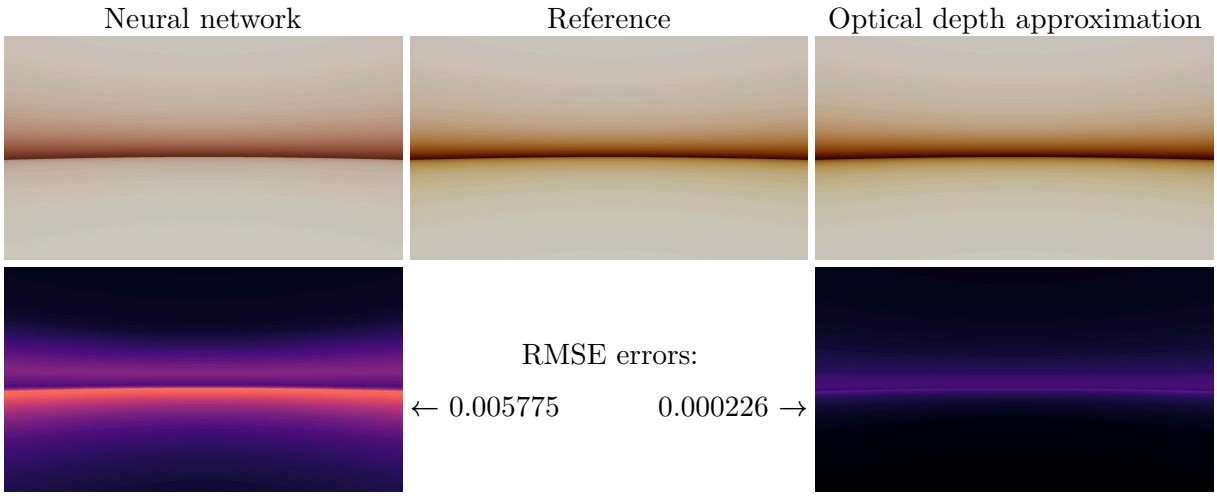
For both the neural network model, as well as the new optical depth approximation, the transmittance can be compared to a naive, raymarched reference, using 128 steps of Riemann integration. The ꟻLIP and RMSE error metrics for each model are found in Appendix C and Appendix D. Note that as the transmittance is not directly used as an output image, using ꟻLIP to judge similarity may not be appropriate. In the comparisons below, ꟻLIP is only used to create a visual difference map.

When visually inspecting the transmittance from the new approximation, it results in slightly more absorption near the horizon. This is expected, as the approximation gives a larger optical depth than the reference, as seen in figure 8.

The neural network does noticeably worse, as it visually fails at giving the same colors. This error is also reflected in a higher RMSE:

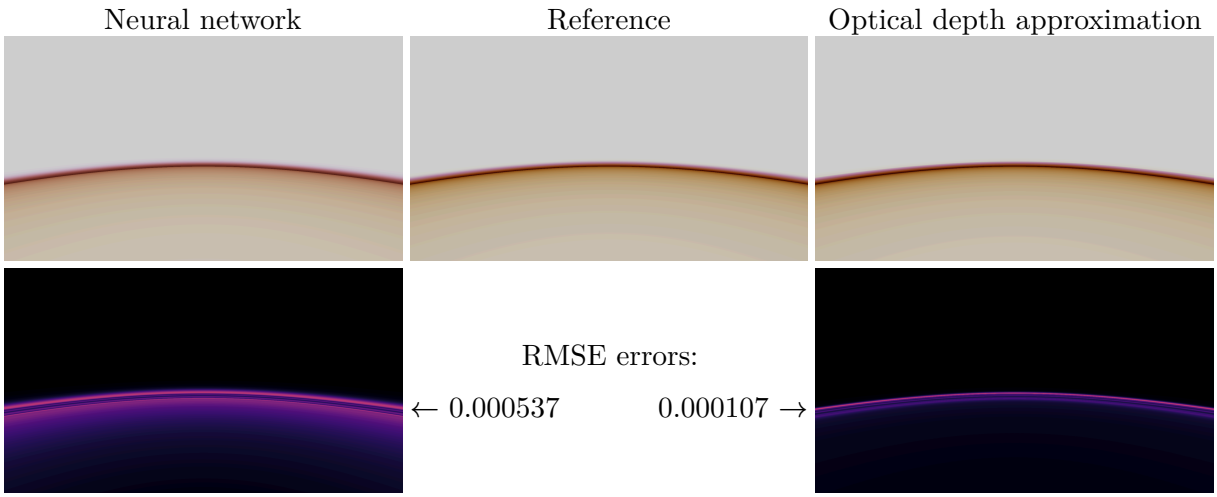



Comparison 37: Ground

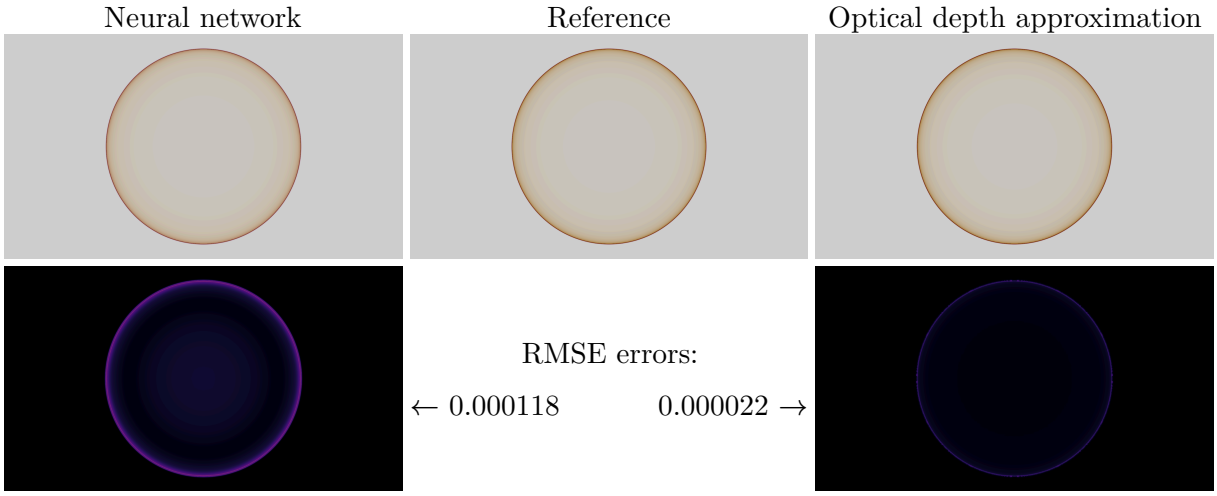



Comparison 38: 5km from the ground

From orbit, it becomes more clear the approximation uses a different way of evaluating the optical depth for the ozone layer, as there is a harsh cutoff that the reference does not have. From this view, it becomes more clear the neural network mixes the ozone layer color into the rest of the atmosphere:




Comparison 39: Ground




Comparison 40: From space

## 12.3. Performance

Here, the performance of the models is compared. As Hillaire's model has different performance characteristics when inside and outside the atmosphere, due to not using the lookup table outside the atmosphere, the performance measurements are split up between inside the atmosphere and outside the atmosphere.

For performance comparisons, many GPU's from different vendors were tested. As different GPU models from the same vendor do not exhibit completely different performance characteristics, and to keep this comparison brief, only one GPU from each vendor is represented here.

The performance data shown below shows the average runtime of each model, in milliseconds, when rendering to a 1920 by 1080 `Rgba32Float` texture, for all views inside, and outside the atmosphere. The viewpoint with the fastest time as well as standard deviation (stdev) are provided as well.

The space view and planet shadow orbit view are not taken into account when plotting the performance graphs. The reason for this is that they calculate the atmosphere for a different amount of pixels than the ground, plane and orbit views, which results in different runtimes.

### 12.3.1. NVIDIA

Below are the tables for an NVIDIA RTX 3090, one with the viewer inside the atmosphere, and one with the viewer outside the atmosphere. When the viewer is outside the atmosphere, Hillaire's model switches to raymarching the atmosphere, and uses a lookup table for its transmittance calculation. It also has a larger step count (40) than Schuler's model (24), and thus becomes the second slowest model.

A somewhat unexpected result is that of Bruneton and Neyret's model being slower than the raymarched model.

Below are the time it takes for precomputation, as well as a bar plot for the average runtime for each view. "out" means the viewer is outside the atmosphere, "in" means the viewer is inside the atmosphere:

| **Model** | **Time (ms)** |
|---|---|
| Bruneton and Neyret | 278.687 |
| Hillaire | 10.216 |

Table 2: NVIDIA RTX 3090, precompute times

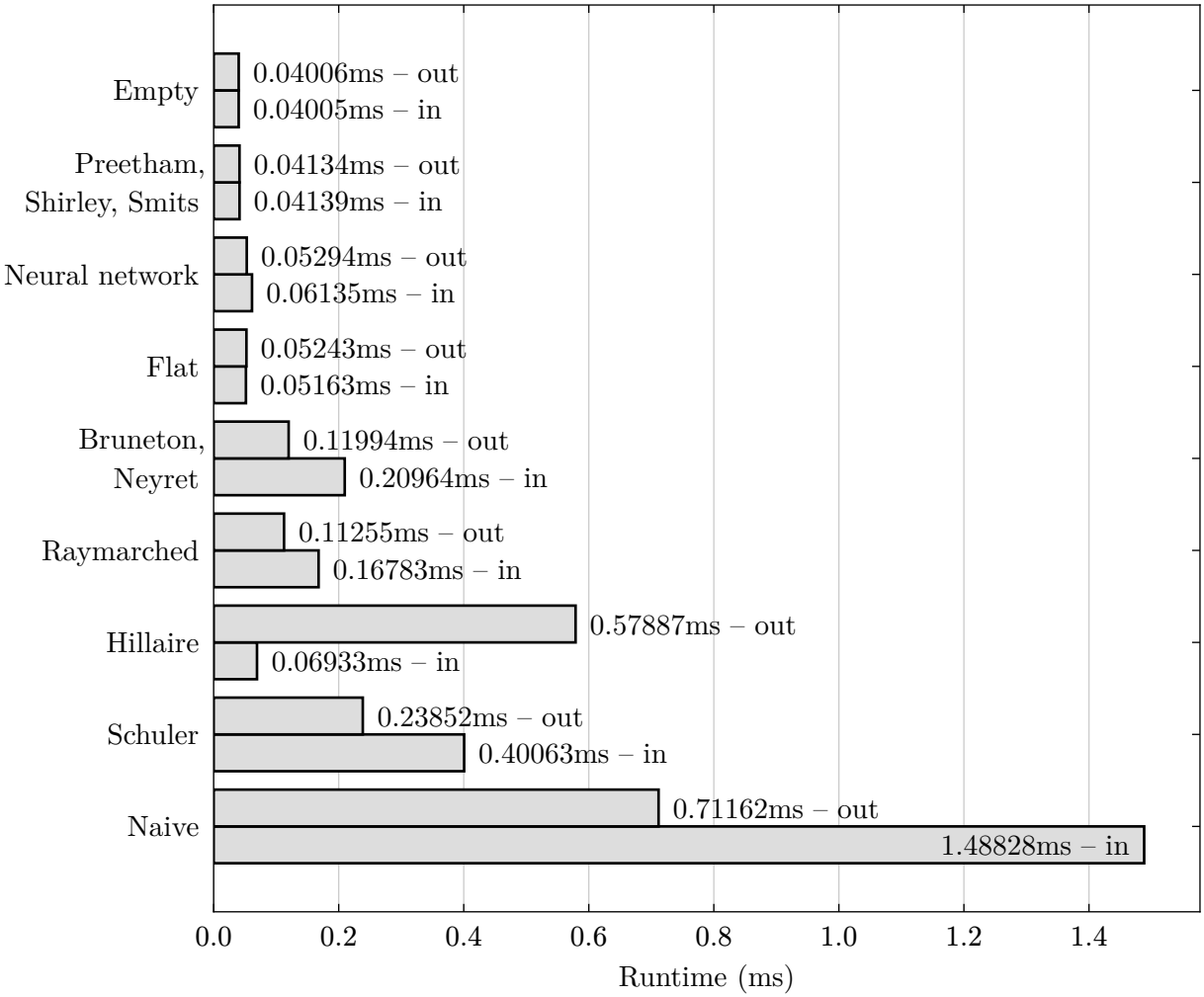


Figure 41: NVIDIA RTX 3090 timings

### 12.3.2. AMD

On AMD GPU's, Bruneton and Neyret's model, as well as Hillaire's model, are consistently faster than both the raymarched and neural network model, which is not the case on NVIDIA GPU's. On an AMD RX 9070 XT, the flat model becomes faster than both Hillaire's and Bruneton and Neyret's model, when the viewer is inside the atmosphere. When the viewer is outside the atmosphere, Hillaire's model becomes the second slowest again.

The timings for an AMD RX 7600 are below:

| **Model** | **Time (ms)** |
|---|---|
| Bruneton and Neyret | 145.339 |
| Hillaire | 21.872 |

Table 3: AMD RX 7600, precompute times

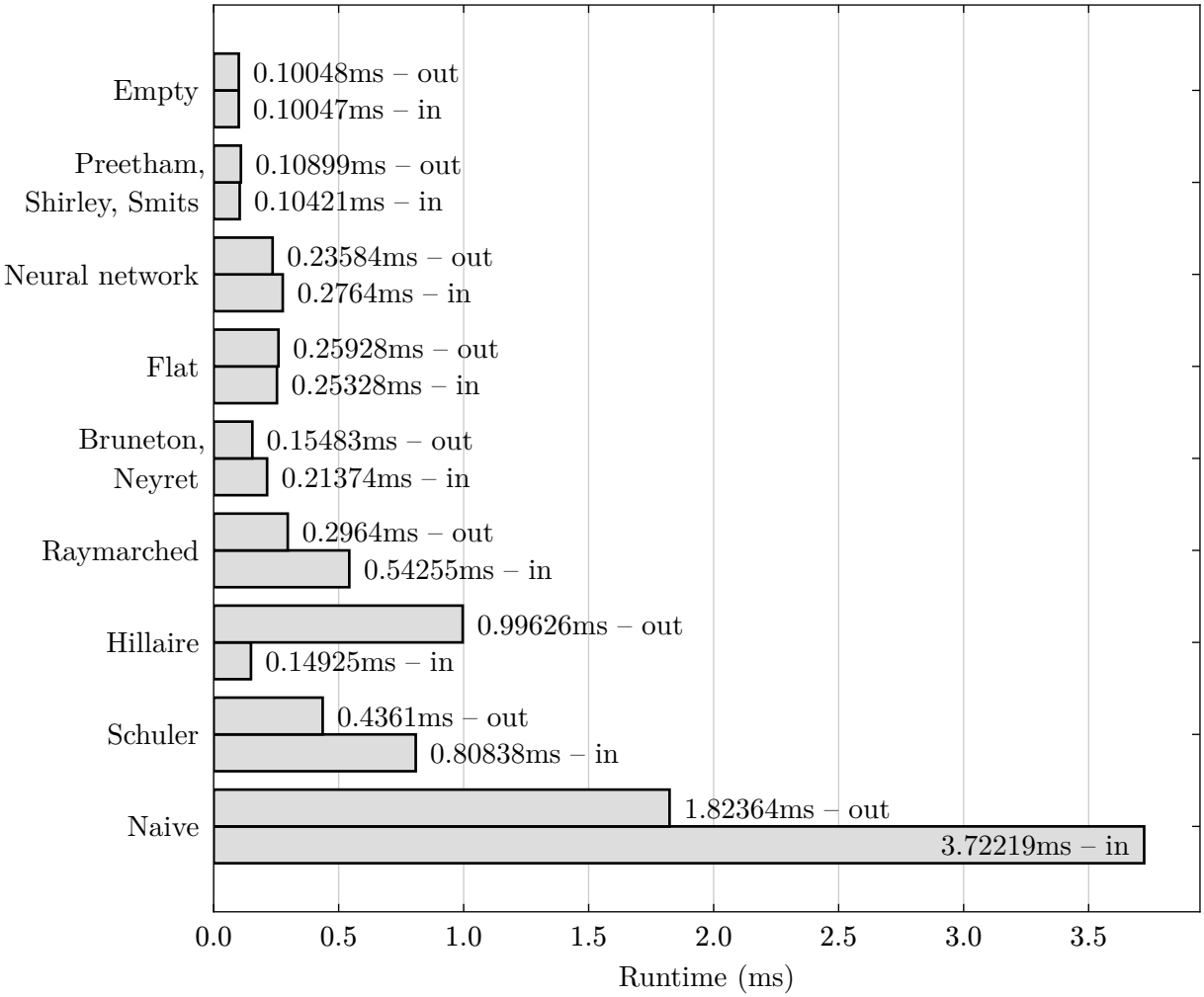


Figure 42: AMD RX 7600 timings

### 12.3.3. Steamdeck

As the steamdeck is a more common device with a lower end AMD GPU, it is included as a separate diagram. It matches the performance characteristics of the other AMD GPU's tested.

| Model | Time (ms) |
|---|---|
| Bruneton and Neyret | 970.708 |
| Hillaire | 227.52 |

Table 4: Steamdeck GPU (AMD), precompute times

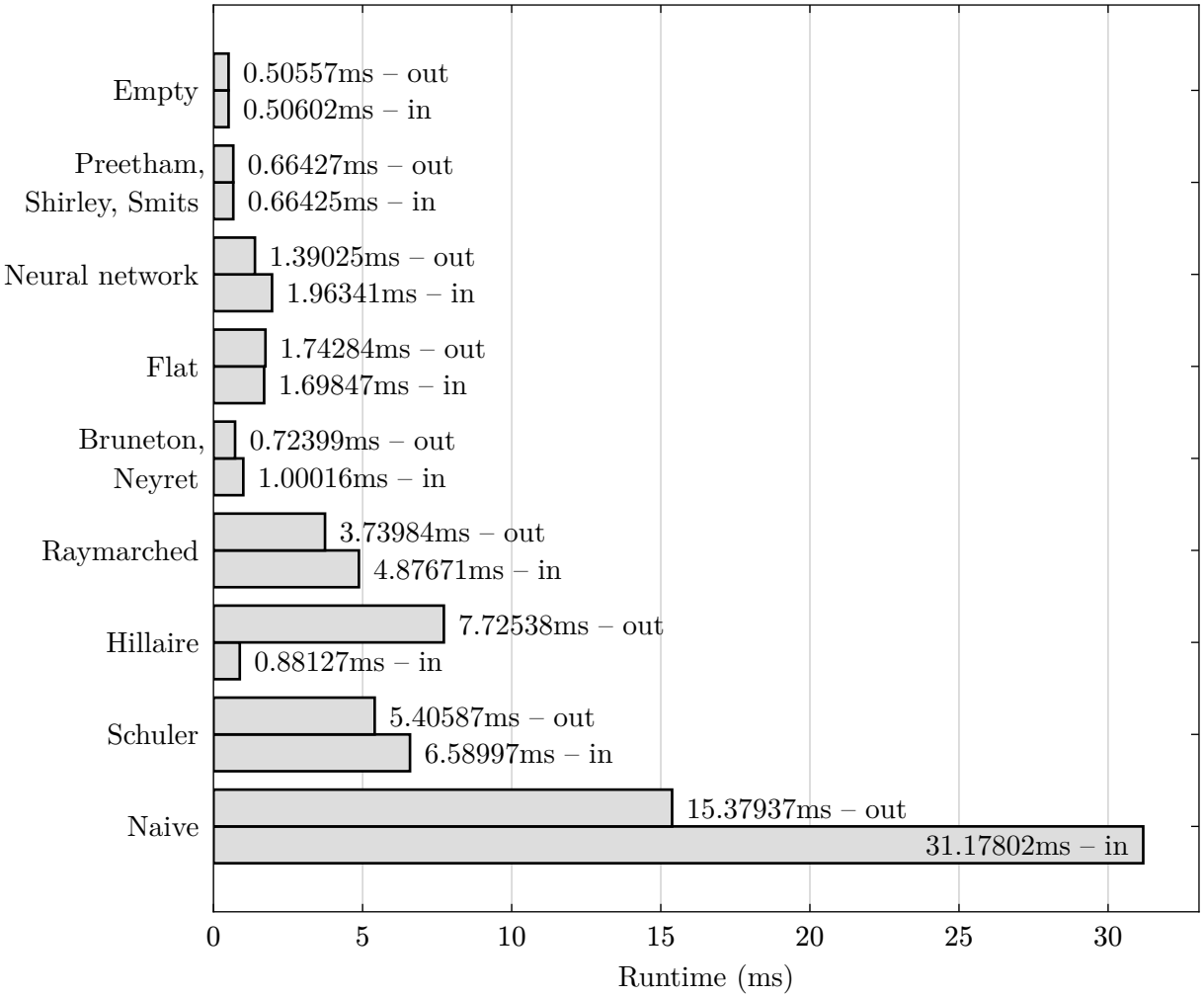


Figure 43: Steamdeck GPU (AMD) timings

#### 12.3.4. Intel

Only one Intel integrated GPU was tested, which is the integrated GPU found in the Intel Ultra 7 258V.

Of note is that when comparing the timings, the model by Preetham, Shirley and Smits runs faster than the empty model, which is counterintuitive, as the empty model should not perform more calculations than any of the actual models.

| Model | Time (ms) |
|---|---|
| Bruneton and Neyret | 535.889 |
| Hillaire | 74.768 |

Table 5: Intel Ultra 7 258V iGPU, precompute times

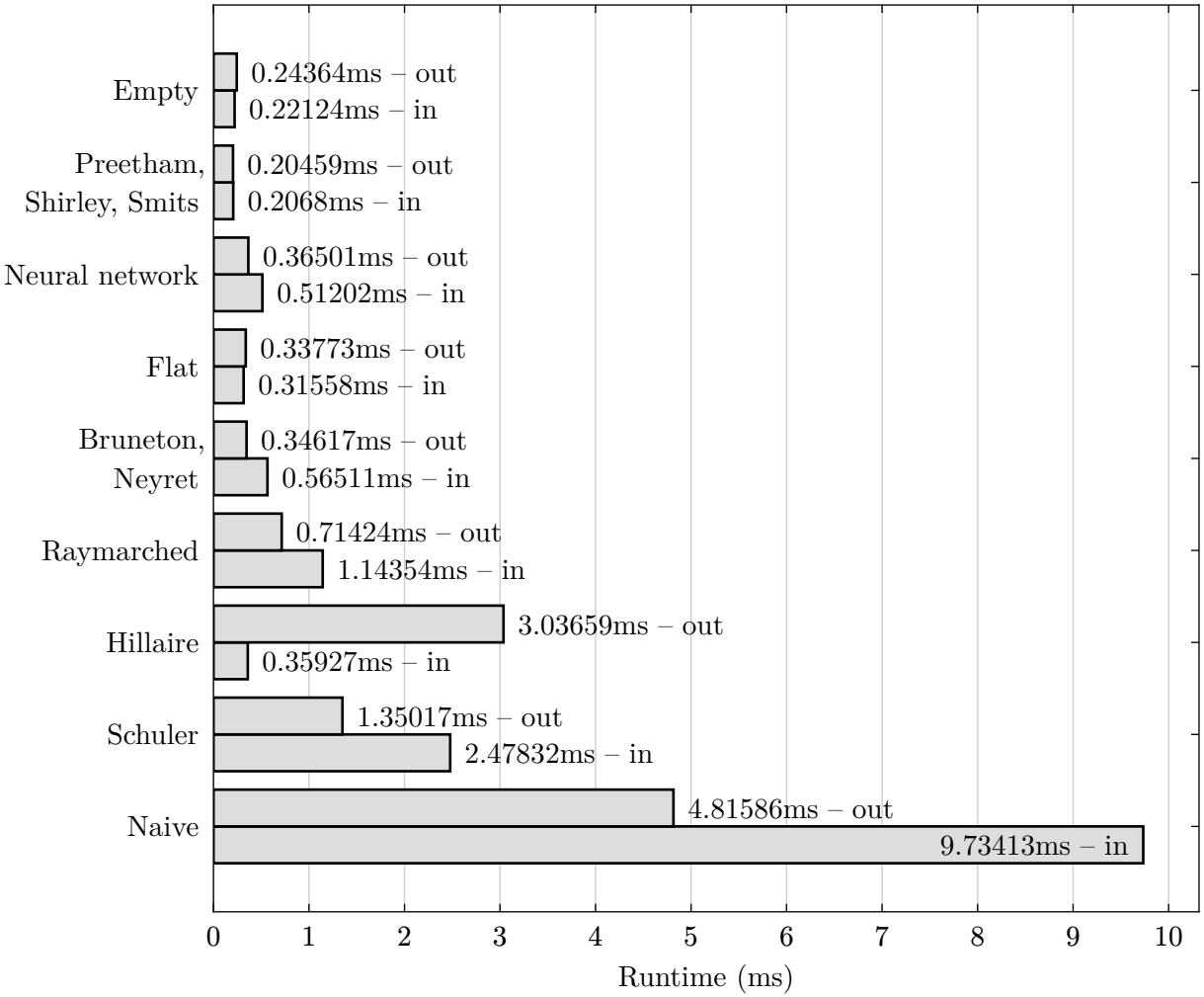


Figure 44: Intel Ultra 7 258V iGPU timings

## 12.4. Implementation complexity

When counting the significant lines of code for all implementation, this is the result, ordered from least to most lines of code:

1. Preetham, Shirley and Smits (58)
2. Flat (58)
3. Raymarched (74)
4. Neural network (94)
5. Schuler (106)
6. Naive (118)
7. Path tracer (147)
8. Hillaire (208)
9. Bruneton and Neyret (832)

Assuming that reference code is available, this code would simply have to be ported to the desired shader language and graphics framework, which is roughly proportional to the lines of code. For both the models by Hillaire, as well as Bruneton and Neyret, extra care has to be taken to properly set up the precomputation steps, as well as required textures.

The precomputation of the model by Hillaire is quite simple, as only three lookup tables are required, and they have a clear order in which they can be computed.

For Bruneton and Neyret's model, this is more difficult. Despite the excellent documentation in (Bruneton, 2017a), the required precomputation steps are still not thoroughly explained, and as it requires more lookup tables, as well as extra intermediate steps to function. This makes setting up the precomputation more error prone.

For the fitted models, both the model by Preetham, Shirley and Smits, as well as the neural network model, the implementation complexity depends on whether the given parameters in the reference code are desired or not. Both models require some reference to fit the model to. This can be another atmosphere model, (Nishita et al., 1993) in case of Preeham, Shirley and Smits' model, and the path tracer for the neural network model.

For both fitted models, extra code is also needed to then fit the coefficients to the reference data. In case of the neural network model, some manual work may be required as well, as not all sets of coefficients fit the data as well as desired.

### 12.5. Overall

Overall, Bruneton and Neyret's model, as well as Hillaire's model, are the closest to the reference visually. While Preetham, Shirley and Smits' model is the fastest of all models runtime wise, it is followed by both Bruneton and Neyret's model, as well as Hillaire's model, when the viewer is inside the atmosphere. The flat model, and the neural network model follow after this.

If multiple scattering is desired, both Bruneton and Neyret's model, as well as Hillaire's model can be used, however, Hillaire's model may come with a performance penalty when used for viewers outside the atmosphere. If no multiple scattering is desired, the raymarched model, or Schuler's model can be used. If the viewer remains in the atmosphere, the flat model provides an increase in performance, at the cost of visually performing worse when the sun is below the horizon. The naive model, using numerical integration, should not be used due to its high performance cost.

According to the raymarched model, Schuler's model, and Hillaire's model when the viewer is outside the atmosphere, using an approximation for the optical depth provides a noticeable speedup over numerical integration. Using a lookup table for transmittance results in a similar speedup.

While the neural network model provides good runtime performance, it does not do well visually. This may be improved with a larger neural network, that would also affect runtime performance.

## 13. Conclusion

After comparing the implemented models, the questions from chapter 6 can now be answered.

The neural network model is a form of fitted model, and it works with both a viewer near the ground, as well as a viewer in space. While performance differs between GPU vendors, it remains close to lookup-table based models. The flat model remains close in performance to the lookup-table based models as well, but the raymarched model is slower, even with the low step count used.

The neural network model has to be retrained to change any of the atmosphere parameters and properties. The flat and raymarched models, while not a fitted model, can work directly with different parameters.

For both the flat and raymarched models, a different $\rho(h)$ than an exponential density is possible, as shown by the ozone layer used in both models. The approximation used for the optical depth is also able to approximate a non-exponential density profiles, according to (Rapp-Arrarás & Domingo-Santos, 2011). This however was not verified in the comparisons done.

# 14. Future work

The comparison done between Bruneton and Neyret's model, as well as Hillaire's model is done using only the `Rgba32Float` texture format, with texture resolutions used as provided in the reference implementations of these models. Of interest may be using different texture formats, as well as resolutions to see if there is a performance tradeoff in doing so. The transmittance lookup table from Hillaire's model should also be compared against the optical dept approximation introduced, as well as the one used in (Schuler, 2012).

The flat and raymarched models also do not model multiple scattering. This can likely be added using a similar method to the one used in (Monzon et al., 2024), or (Schuler, 2018). The approximation used for the optical depth in both these models is also only tested for an exponentially decaying density atmosphere. While (Rapp-Arrarás & Domingo-Santos, 2011) confirms it does work for different density profiles, they only test it with the viewer at a fixed altitude. Extra work is likely required to make it work well with a varying viewer altitude.

# References


Andersson, P., Nilsson, J., Akenine-Möller, T., Oskarsson, M., Åström, K., & Fairchild, M. D. (2020). FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, *3*(2), 1–23.

Andersson, P., Nilsson, J., Shirley, P., & Akenine-Möller, T. (2021, May). Visualizing Errors in Rendered High Dynamic Range Images. *Eurographics Short Papers*. https://doi.org/10.2312/egs.20211015

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., … Chintala, S. (2024). PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 929–947. https://doi.org/10.1145/3620665.3640366

blackrack. *Scatterer - Atmospheric scattering shaders*. https://spacedock.info/mod/141/scatterer

Bruneton, E. (2017b). A Qualitative and Quantitative Evaluation of 8 Clear Sky Models. *IEEE Transactions on Visualization and Computer Graphics*, *23*(12), 2641–2655. https://doi.org/10.1109/tvcg.2016.2622272

Bruneton, E. (2017a, ). *Precomputed Atmospheric Scattering: a New Implementation*. https://inria.hal.science/inria-00288758

Bruneton, E., & Neyret, F. (2008). Precomputed Atmospheric Scattering. *Computer Graphics Forum*, *27*(4), 1079–1086. https://doi.org/10.1111/j.1467-8659.2008.01245.x

Chapman, S. (1931). The absorption and dissociative or ionizing effect of monochromatic radiation in an atmosphere on a rotating earth. *Proceedings of the Physical Society*, *43*(1), 26. https://doi.org/10.1088/0959-5309/43/1/305

Cornette, W., & Shanks, J. (1992). Physically reasonable analytic expression for the single-scattering phase function. *Applied Optics*, *31*, 3152–3160. https://doi.org/10.1364/AO.31.003152

Cranmer, M. (2023, ). *Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl*. https://arxiv.org/abs/2305.01582

*Digital Combat Simulator World*. https://www.digitalcombatsimulator.com/en/products/world/

Eckhart, R. (1987). Stan Ulam, John von Neumann, and the Monte Carlo Method. *Los Alamos Science*, 131–137.

Elek, O. (2009). *Rendering Parametrizable Planetary Atmospheres with Multiple Scattering in Real-Time*.

Fong, J., Wrenninge, M., Kulla, C., & Habel, R. (2017, ). Production volume rendering: SIGGRAPH 2017 course. *ACM SIGGRAPH 2017 Courses*. https://doi.org/10.1145/3084873.3084907

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Henyey, L., & Greenstein, J. (1941). Diffuse radiation in the Galaxy. *\apj*, *93*, 70–83. https://doi.org/10.1086/144246

Hillaire, S. (2015, ). Towards Unified and Physically-Based Volumetric Lighting in Frostbite. *Advances in Real-Time Rendering in Games*.

Hillaire, S. (2020). A Scalable and Production Ready Sky and Atmosphere Rendering Technique. *Computer Graphics Forum*, *39*(4), 13–22. https://doi.org/https://doi.org/10.1111/cgf.14050

Hosek, L., & Wilkie, A. (2012). An Analytic Model for Full Spectral Sky-Dome Radiance. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, *31*(4).

Jodie. (2019, ). *cheap sky simulation*.

Kingma, D. P., & Ba, J. (2017, ). *Adam: A Method for Stochastic Optimization*. https://arxiv.org/abs/1412.6980

Kocifaj, M. (1996). Optical air mass and refraction in a Rayleigh atmosphere. *Contributions of the Astronomical Observatory Skalnate Pleso*, *26*, 23–30.

Kolda, T. G., & Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, *51*(3), 455–500. https://doi.org/10.1137/07070111X

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., & Tegmark, M. (2025, ). *KAN: Kolmogorov-Arnold Networks*. https://arxiv.org/abs/2404.19756

*Microsoft Flight Simulator 2024*. https://www.flightsimulator.com/microsoft-flight-simulator-2024/

Monzon, N., Gutierrez, D., Akkaynak, D., & Muñoz, A. (2024). Real-Time Underwater Spectral Rendering. *Computer Graphics Forum*, e15009. https://doi.org/https://doi.org/10.1111/cgf.15009

Neumann, J. von. (1951). Various Techniques Uesd in Connection with Random Digits. *Journal of Research of the National Bureau of Standards, Applied Math*, 36–38.

Nishita, T., Sirai, T., Tadamura, K., & Nakamae, E. (1993). Display of the earth taking into account atmospheric scattering. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 175–182. https://doi.org/10.1145/166117.166140

Novák, J., Selle, A., & Jarosz, W. (2014). Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.*, *33*(6). https://doi.org/10.1145/2661229.2661292

O'Neil, S. (2004, ). *Real-time atmospheric scattering*. https://archive.gamedev.net/archive/reference/articles/article2093.html

O'Neil, S. (2005). Chapter 16. Accurate Atmospheric Scattering. In M. Pharr & R. Fernando (Eds.), *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems): GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional.

Perez, R., Seals, R., & Michalsky, J. (1993). All-weather model for sky luminance distribution —Preliminary configuration and validation. *Solar Energy*, *50*(3), 235–245. https://doi.org/https://doi.org/10.1016/0038-092X(93)90017-I

Pharr, M., Jakob, W., & Humphreys, G. (2016). *Physically Based Rendering: From Theory to Implementation (3rd ed.)* (3rd ed., p. 1266). Morgan Kaufmann Publishers Inc.

Preetham, A. J., Shirley, P., & Smits, B. E. (1999). A practical analytic model for daylight. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. https://api.semanticscholar.org/CorpusID:7993169

Rapp-Arrarás, Í., & Domingo-Santos, J. M. (2011). Functional forms for approximating the relative optical air mass. *Journal of Geophysical Research: Atmospheres*, *116*(D24), . https://doi.org/https://doi.org/10.1029/2011JD016706

Schuler, C. (2012). An Approximation to the Chapman Grazing-Incidence Function for Atmospheric Scattering. In W. Engel (Ed.), *GPU PRO 3: Advanced Rendering Techniques: GPU PRO 3: Advanced Rendering Techniques* (1st ed., pp. 105–118). A. K. Peters, Ltd.

Schuler, C. (2018, ). *Space Glider*.

Suzuki, K., & Yasutomi, K. (2023, ). *Realistic Real-time Sky Dome Rendering in Gran Turismo 7*. https://www.gdcvault.com/play/1029434/Advanced-Graphics-Summit-Realistic-Real

Terrell, R. (2016, ). *glsl-atmosphere*.

Vasylyev, D. (2021). Accurate analytic approximation for the Chapman grazing incidence function. *Earth Planets and Space*, *73*, 112. https://doi.org/10.1186/s40623-021-01435-y

Wilkie, A., Vevoda, P., Bashford-Rogers, T., Hošek, L., Iser, T., Kolářová, M., Rittig, T., & Křivánek, J. (2021). A fitted radiance and attenuation model for realistic atmospheres. *ACM Trans. Graph.*, *40*(4). https://doi.org/10.1145/3450626.3459758

Yue, D. (2024). Analytical approximation of the definite Chapman integral for arbitrary zenith angles. *Atmospheric Chemistry and Physics*, *24*(8), 5093–5097. https://doi.org/10.5194/acp-24-5093-2024

## A: ꟻLIP errors

| ꟻLIP errors | Preetham, Shirley, and Smits | Empty | Raymarched | Schuler | Naive | Hillaire | Neural Net | Flat | Bruneton and Neyret |
|---|---|---|---|---|---|---|---|---|---|
| ground-dawn | 0.49564359 | 0.97566503 | 0.22362486 | 0.41188592 | 0.16566165 | 0.11073484 | 0.34731606 | 0.47711548 | **0.09809241** |
| ground-sunrise | 0.24515393 | 0.92652833 | 0.16458111 | 0.16321646 | 0.19359379 | 0.12071851 | 0.29608113 | 0.16371098 | **0.03308802** |
| ground-noon | 0.22209883 | 0.90915573 | 0.14701016 | 0.2058856 | 0.20784122 | 0.07796405 | 0.07380003 | 0.15026346 | **0.01935603** |
| plane-dawn | 0.97412103 | 0.97515351 | 0.64763862 | 0.88058525 | 0.57780761 | 0.30420122 | 0.72513878 | 0.93408012 | **0.28129065** |
| plane-sunrise | 0.89782381 | 0.9277581 | 0.32033435 | 0.31173941 | 0.33786449 | 0.13399105 | 0.40059972 | 0.30929685 | **0.05343369** |
| plane-noon | 0.68168336 | 0.84588838 | 0.26416466 | 0.29830816 | 0.30259022 | 0.10759243 | 0.3525098 | 0.27081707 | **0.04459071** |
| orbit-dawn | 0.80826867 | 0.97057861 | 0.27345997 | 0.28710017 | **0.26551905** | 0.40727559 | 0.47434133 | 0.66090101 | 0.41126001 |
| orbit-sunrise | 0.97256225 | 0.95243591 | 0.15076236 | 0.26375878 | 0.14113207 | **0.0641655** | 0.30828747 | 0.29419997 | 0.06427314 |
| orbit-noon | 0.96073997 | 0.88029242 | 0.12195973 | 0.14183527 | 0.14153102 | 0.07187424 | 0.2281246 | 0.20794494 | **0.05222957** |
| space | 0.5143919 | 0.96719503 | 0.09372713 | 0.10406315 | 0.09261382 | **0.08722282** | 0.21325293 | 0.226421 | 0.15413877 |
| planet-shadow-ground | 0.48114309 | 0.96296805 | 0.30747619 | 0.41281596 | 0.26425651 | 0.06620285 | 0.20270388 | 0.42022756 | **0.05699512** |
| planet-shadow-orbit | 0.79023546 | 0.96181011 | 0.16743426 | 0.2336743 | 0.16060859 | **0.12997645** | 0.25569162 | 0.33876443 | 0.14620577 |

## B: RMSE errors

| RMSE errors | Preetham, Shirley, and Smits | Empty | Raymarched | Schuler | Naive | Hillaire | Neural Net | Flat | Bruneton and Neyret |
|---|---|---|---|---|---|---|---|---|---|
| ground-dawn | 3.97865605 | 0.97482044 | 0.00008048 | 0.0002448 | 0.00004764 | 0.00003619 | 0.00030104 | 0.01105675 | **0.00003364** |
| ground-sunrise | 0.00753167 | 0.82744461 | 0.00140716 | 0.00094494 | 0.00308942 | 0.00109592 | 0.02124281 | 0.00176891 | **0.00014694** |
| ground-noon | 0.00638251 | 0.81927776 | 0.00255514 | 0.00589603 | 0.00602361 | 0.00009313 | 0.00015921 | 0.0026469 | **0.00002464** |
| plane-dawn | 3.95045567 | 0.9558543 | 0.00023158 | 0.0008222 | 0.00008869 | 0.0000491 | 0.00108281 | 0.00202921 | **0.0000453** |
| plane-sunrise | 0.03052294 | 0.75808531 | 0.00129824 | 0.00113231 | 0.00245048 | 0.00011849 | 0.00556002 | 0.00116698 | **0.0000506** |
| plane-noon | 0.02289433 | 0.73097199 | 0.00350788 | 0.00615819 | 0.00631668 | 0.00007912 | 0.00254121 | 0.0036586 | **0.00002183** |
| orbit-dawn | 4.08668518 | 0.97941327 | 0.00102501 | 0.00645754 | 0.00020258 | **0.00003532** | 0.01249882 | 0.00992934 | 0.00017324 |
| orbit-sunrise | 0.04049467 | 0.92837852 | 0.00016845 | 0.0005017 | 0.00052833 | 0.00002512 | 0.00371327 | 0.00070512 | **0.00001186** |
| orbit-noon | 0.02664175 | 0.87550873 | 0.00157611 | 0.00226653 | 0.00247161 | 0.00001541 | 0.0032039 | 0.00149308 | **0.00001374** |
| space | 0.03107663 | 0.95405173 | 0.00086303 | 0.00104631 | 0.00118762 | 0.00001662 | 0.00068834 | 0.0013083 | **0.00000733** |
| planet-shadow-ground | 2.65018225 | 0.85214376 | 0.00200995 | 0.0040155 | 0.00188857 | 0.00032487 | 0.00429041 | 0.02545365 | **0.00027963** |
| planet-shadow-orbit | 2.96405864 | 1.12555254 | 0.00552621 | 0.04436417 | 0.01432145 | **0.00023957** | 0.21114185 | 1.40199077 | 0.00067625 |

## C: ꟻLIP errors, transmittance

| ꟻLIP errors | Neural network | Flat |
|---|---|---|
| ground | 0.0906432 | **0.05242816** |
| plane | 0.21796529 | **0.07285634** |
| orbit | 0.06687886 | **0.02894559** |
| space | 0.02778055 | **0.01221722** |
| planet-shadow-orbit | 0.05516684 | **0.02228876** |

## D: RMSE errors, transmittance

| RMSE errors | Neural network | Flat |
|---|---|---|
| ground | 0.00072169 | **0.00018785** |
| plane | 0.00577525 | **0.00022608** |
| orbit | 0.00053684 | **0.00010681** |
| space | 0.00011823 | **0.00002235** |
| planet-shadow-orbit | 0.00046473 | **0.00009816** |

## E: Link to code repository

The public repository of the code used can be found here: https://git.science.uu.nl/vig/mscprojects/a-visual-and-performance-comparison-of-atmospheric-scattering-models

## F: Links to shadertoy versions of shaders

Below are the links to the *shadertoy* versions of the implemented shaders. Only Bruneton and Neyret's atmosphere is not implemented in shadertoy, as it is not possible to use 3D buffers.

- Flat: https://www.shadertoy.com/view/t3XBWH
- Raymarched: https://www.shadertoy.com/view/tXXBWH
- Neural Network: https://www.shadertoy.com/view/tXffD8
- Preetham, Shirley, Smits: https://www.shadertoy.com/view/WX2Bz1
- Naive: https://www.shadertoy.com/view/W3jBz1
- Schuler: https://www.shadertoy.com/view/WX2fR1
- Hillaire's: https://www.shadertoy.com/view/wcdGRX
- Path tracer: https://www.shadertoy.com/view/Wfd3D8