

Participating Media and Volume Scattering II

Lecture #19: Thursday, 03 April 2003
Lecturer: Greg Humphreys
Scribe: Dale Beermann
Reviewer: Nolan Goodnight

1 Phase Functions

Shading models for volume scattering are generally based on particle size within the volume. Phase functions, also called scattering functions, are the mathematical models for volume rendering. There are several phase functions, each of which is meant to be used for a slightly different purpose. Since particles in the volume are assumed to be spherical, the average radius of the particles compared to the wavelength of light (555nm, the scotopic peak wavelength, is generally used) determines the appropriate phase function. See Table 1 for the classification. Note that we will use Henyey-Greenstein as an approximation to Mie scattering.

$r \gg \lambda$	Atmospheric absorption
$r < \lambda$	Rayleigh scattering
$r \approx \lambda$	Mie scattering
$r \ll \lambda$	Geometrical optics

Table 1: Selecting a Phase Function

Table 1 tells us that when the radius of the particles is much smaller than the wavelength of light, we have atmospheric absorption where the light is simply absorbed. Conversely, when the radius of the particles is much bigger than the wavelength, we are really just modelling geometrical optics, much as we have been throughout the semester. The other two cases will be discussed below.

The phase function has two important properties that it must uphold. The first is that it is reciprocal, meaning that $\rho(w_i \rightarrow w_o) = \rho(w_o \rightarrow w_i)$. The second property is that the phase function must be normalized, $\int_{S^2} \rho(w_i \rightarrow w_o) dw_i = 1$. Furthermore, the phase functions we will use are usually given in terms of one angle, meaning that they are all isotropic.

1.1 Isotropic Scattering

$$\rho(\cos\theta) = \frac{1}{4\pi}$$

Simple isotropic scattering is just the same as diffuse reflection. This is because it is not dependant on the angle, so the phase function is just a constant. It is equal to $\frac{1}{4\pi}$ because it must be normalized over the sphere.

1.2 Rayleigh Scattering

$$\rho(\cos\theta) = \frac{3}{4} \cdot \frac{1 + \cos^2\theta}{\lambda^4}$$

When the particles smaller than the wavelength of the light, Rayleigh scattering is used. It has been suggested that Rayleigh scattering should be used when $r/\lambda < 0.05$. The above equation includes the wavelength, which is in fact part of σ_s , the scattering coefficient, but it is included here to show how it affects the phase function. Since the contributing term is $1/\lambda^4$, long wavelengths are scattered more than short wavelengths. This is the cause of both the sky being blue and red sunsets on a blue sky. As shown in Figure 1, the Rayleigh phase function is lobed on both sides of the vertical axis. What this means is that light is scattered more towards and away from your eye than in the direction perpendicular to your eye. Since red light is getting scattered more, we only see it during a sunset because as we look towards the horizon, we are looking more directly at the sun, and the red light is getting scattered towards our eye.

1.3 Henyey-Greenstein Scattering

$$\rho(\cos\theta) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g\cos\theta)^{1.5}}$$

$$g = \int_{S^2} \rho(w \rightarrow w') \cos\theta' dw'$$

The Mie phase function is costly to compute since it can involve a term taken to the 32nd power, so the Henyey-Greenstein function can be used instead. The function g included here allows us to determine the type of scattering being simulated. For backscattering, we can take $g > 0$, for uniform scattering $g = 0$, and forward scattering, $g < 0$. It is also possible to combine these phase functions to model other more complicated phase functions.

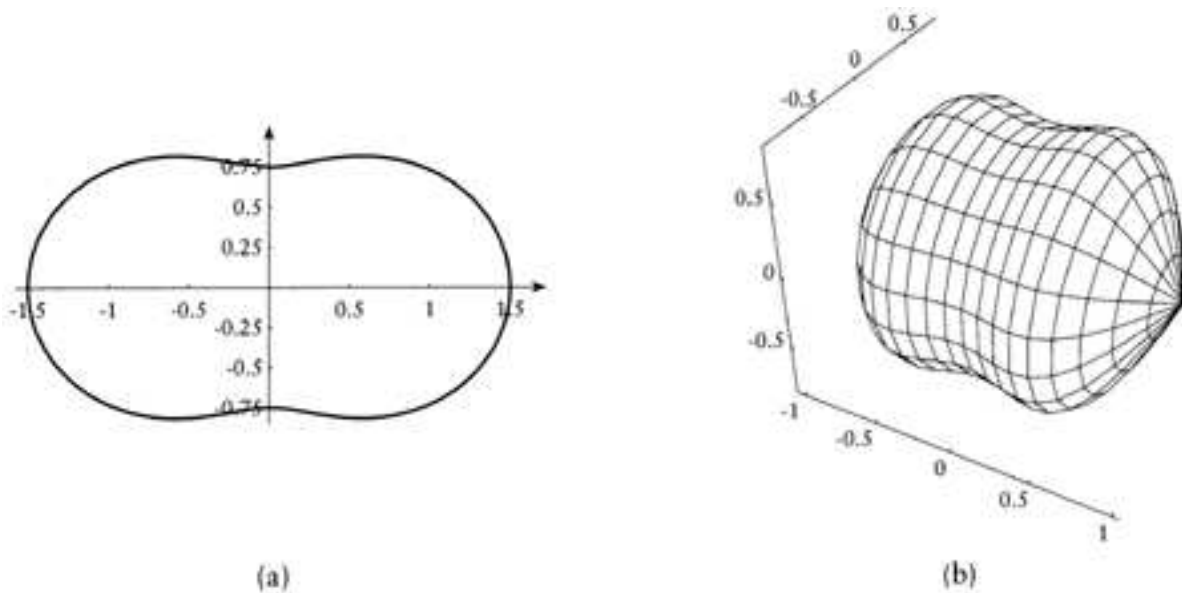


Figure 1: a) The Rayleigh phase function. b) The function in a) rotated around the incident vector

1.4 Schlick Scattering

$$\rho(\cos\theta) = \frac{1 - k^2}{4\pi(1 + k\cos\theta)^2}$$

$$\cos\theta = \frac{2u + g - 1}{2gu - g + 1}$$

The Schlick function can model the other phase functions by using different values for k . Like the Henyey-Greenstein function, it also accounts for all types of scattering, where $k = 1$ is entirely forward scattering, $k = 0$ is isotropic scattering, and $k = -1$ is backscattering. Schlick functions can then be added together to model other phase functions by using the following formula:

$$\rho_{k,k',r}(t) = r\rho_k(t) + (1 - r)\rho_{k'}(t)$$

2 The Volume Rendering Equation

Using these phase functions, we come up with an Integro-differential equation, which is just a little ugly and not so easy to evaluate:

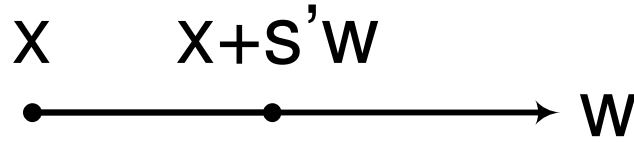


Figure 2: Adding up the amount of light along a ray due to a point.

$$\frac{dL(x, w)}{dS} = -\sigma_t L(x, w) + \sigma_s(x) \int_{S^2} \rho(w' \rightarrow w) L(x, w') dw'$$

We can make this a little prettier however if you begin with the total amount of light due to a point, as in Figure 2. This is equal to $T(s)S(x + s'w)$ where $T(s)$ is the transmittance. The contribution from all points then is $e^{\int_0^{s'} \sigma_t(x+sw') ds'}$. We can then rewrite the volume rendering equation as:

$$L(x, w) = \int_0^\infty e^{\int_0^{s'} \sigma_t(x+sw') ds'} S(x + s'w) ds'$$

Unfortunately, there are only two cases in which this can be solved. The first case is for a homogenous medium, the second is for a semi-infinite homogeneous medium.

2.1 A Homogeneous Medium

$$\frac{dL(s)}{dx} = -\sigma_t L(s) + S$$

$$L(s) = (1 - e^{-\sigma_t s}) + e^{-\sigma_t s} c$$

The second equation above is much easier to read. The first term represents the contribution from the atmosphere or medium and the second term represents the light due to the object. See Figure 3 for a pictorial description. One important thing to note is that the first term makes up the OpenGL light model.

2.2 Semi-infinite Homogeneous Medium

In this case, each ray contributes some radiance times the distance scattered in and the distance scattered out and the probability times the phase function:

$$\cos\theta_o L_o(w_o) = \int L_i(w_i) e^{\frac{-\sigma_t z}{\cos\theta_i}} \sigma_s \rho(w_i \rightarrow w_o) e^{\frac{-\sigma_t z}{\cos\theta_i}} dz$$

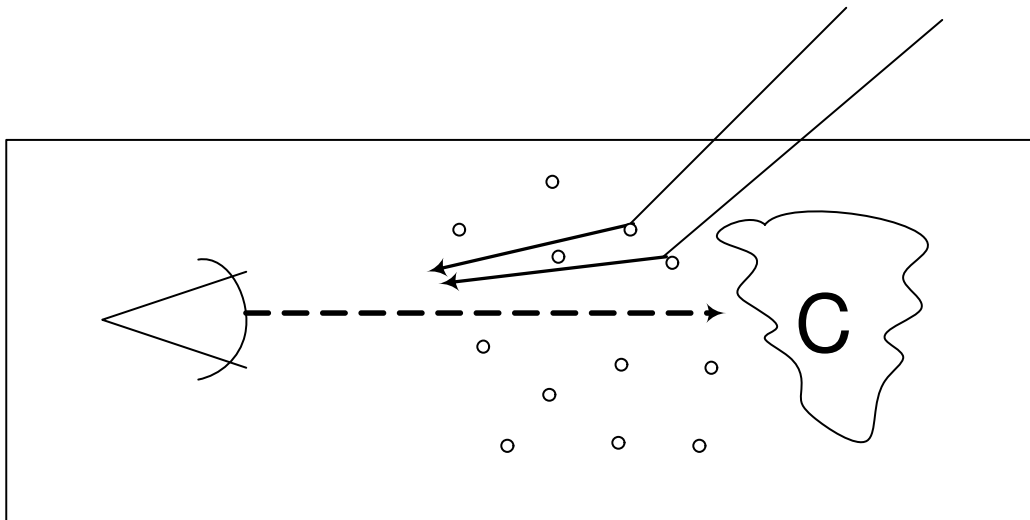


Figure 3: Contribution of light in a homogeneous medium.

This will make sense with respect to Figure ???. We can now rearrange things and try to make this look a little better:

$$= \sigma_s \rho(w_i \rightarrow w_o) L_i(w_i) \int_0^\infty e^{-\sigma [\frac{1}{\cos\theta_i} + \frac{1}{\cos\theta_o}]z} dz$$

$$= \sigma_s \rho(w_i \rightarrow w_o) L_i(w_i) \frac{1}{\sigma_z \frac{1}{\cos\theta_i} + \frac{1}{\cos\theta_o}}$$

$$= W \rho(w_i \rightarrow w_o) L(w_i) \frac{\cos\theta_i}{\cos\theta_i + \cos\theta_o}$$

where $W = \frac{\sigma_s}{\sigma_s + \sigma_a}$ is the albedo. This allows us to create a BRDF now:

$$F_r(w_f \rightarrow w) = \frac{dL}{dE} = \frac{L(w_i \rightarrow w_o)}{L(w_i) \cos\theta_o}$$

$$= W \rho(w_i \rightarrow w_o) \frac{1}{\cos\theta_i + \cos\theta_o}$$

The last equation is known as Seeliger's law.

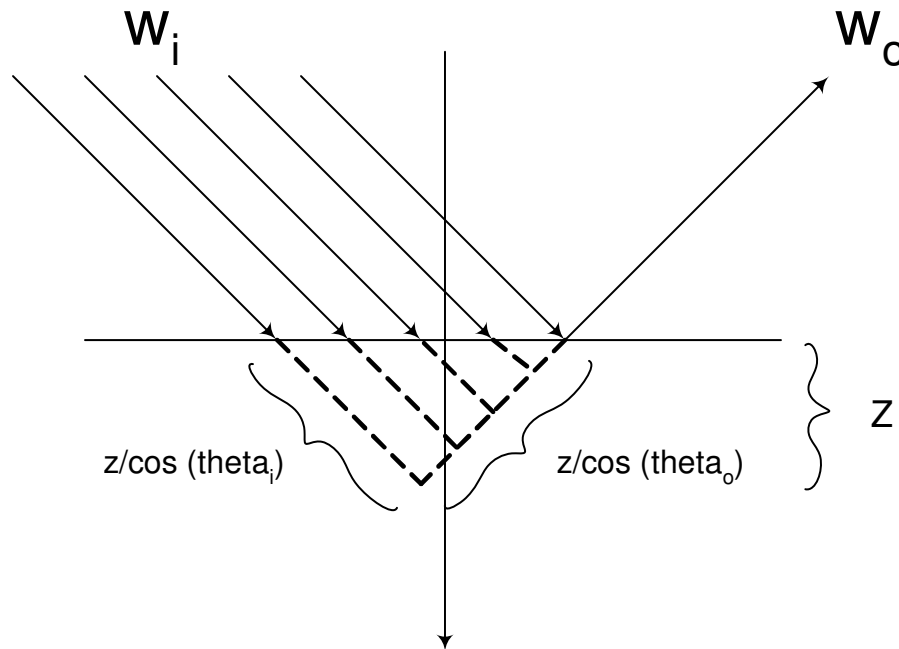


Figure 4: Contribution of light in a semi-infinite medium.

2.3 Ray Marching

The basic idea behind ray marching is accumulating color along a ray when cast through a volume. This is done by stepping at equal intervals through the volume, performing an evaluation at each point on the ray. The evaluations are then composited with each other, from back to front (from the farthest evaluation to the closest) using a simple “over” operation. In this way, it is possible to evaluate a density function in a volume and render such things as fog and smoke. Here’s some pseudocode (make sure you check the next page for a brief explanation):

$T = 1 \leftarrow$ Transmittance

$L = 0 \leftarrow$ Radiance

for ($S = 0, S < 1, s+ = \Delta S$)

$T_S = 1$

 for ($t = 0; t < 1; t+ = \Delta t$)

$T_{S^*} = (1 - \sigma_t(x(t))\Delta t$

$S = \sigma_s(s)\rho(w, w(x(s), x_l))T_S L_i(X_L, w(X_L.x(s)))$

$L+ = T S \Delta S$

$T^* = (1 - \sigma_t(x(s))\Delta S$

The basic idea is that for each point along the incoming ray, you are marching back out of the volume to find the shading (T_S) for that point to find an attenuation factor.