A Flexible Framework for Volume Tracing

Stefan Maierhofer
sm@cg.tuwien.ac.at

Institute of Computer Graphics
Vienna University of Technology
Vienna / Austria

Abstract

In recent years, it has become inevitable to use volume rendering techniques to create highly realistic, state-of-the-art images of a variety of different phenomena. Especially for the rendering of natural phenomena (i.e. clouds or fire), and the depiction of medical datasets (i.e. computed tomographies), volume tracing is very well suited. This paper discusses the simulation of light interacting with participating media and presents a flexible framework, which allows for the incorporation of many different volume rendering properties into general purpose rendering systems.

KEYWORDS: rendering, volume tracing, volume rendering, participating media, natural phenomena, medical visualization, density functions, transfer functions.

1 Introduction

"... in 10 years, all rendering will be volume rendering."

Jim Kajiya at SIGGRAPH '91 [Elv92]

Only time will show if in 2001 all rendering will be volume rendering. But even today one thing can be said for sure: at least a significant part of all rendering already is volume rendering and will be much more so in 2001. Back in 1991, when Kajiya made his statement, it took at least minutes, if not hours to produce volume rendered images. Today it takes only seconds to produce images of comparable quality and as stated in Moore's Law, computer capacity will increase further in an exponential manner. Of course that does not necessarily mean that in some years all volume rendering will be done in real-time, because additional computing capacity is not only used to shorten rendering times, but also to render more complex and more convincing scenes.

In volume rendering, images of radiatively participating media are created. Participating media consist of a large amount of small particles, like water droplets, soot or other suspended solids or individual molecules. Light passing through them may be distracted in many different ways. It may be attenuated by absorption, like light passing through smoke. Participating media may also emit light like fire, or light may be scattered by small particles. In clouds, for example, light is scattered at a myriad of small water droplets.

Most rendering toolkits or frameworks have evolved over years and rendering systems have been around for almost decades. Because volume tracing has not been in widespread application for such a long time, only few general ideas exist on how to integrate all the different specialized volume tracing applications into one general purpose rendering system or into existing rendering systems. The main purpose of this paper is to outline a general, flexible framework for volume tracing.

In the next section some important concepts of volume tracing and its mathematical foundations will be presented. In the 3rd section, the framework, allowing for a flexible combination of different volume data sources, density functions and transfer functions, is outlined. The 4th section briefly describes an implementation of the framework as an integral part of the Advanced Rendering Toolkit (ART for short) and finally results are presented in the 5th section.

2 Fundamentals of Volume Tracing

2.1 Density Functions

The basic structure of participating media is defined by density functions $f(x)$ which return a scalar density value for each point x in space. Different kinds of density functions exist.

Procedural density functions are based on arithmetic expressions. Arbitrary expressions can be used - constant, linear, quadratic and so on, as well as turbulence, noise or other solid texturing functions. Excellent results can be achieved by using procedural density functions to render natural phenomena.

Density functions may also be based on volume data sets, which are created by a large number of different methods and applications. Just to name a few: Computed Tomography (CT), Magnetic Resonance Imaging (MRI), finite elements simulations, sampled geological or meteorological quantities, ...

2.2 Transfer Functions

In order to render participating media, renderable properties like absorption, emission and scattering have to be available. A problem however is, that in general, absorption-, emission- and scattering-functions can not use scalar density values directly. Therefore, function specific parameters, like the color of emitted light for emission-functions has to be derived from scalar density functions. For this reason, so-called transfer functions are used to map scalar density values to values that can be used for rendering. I.e. in medical volume rendering systems, the density function often depends on the type of tissue - transfer functions are used to color different types of tissue differently.

2.3 Participating media

Participating media may absorb, emit and/or scatter light. The simplest participating medium only absorbs light. That means that light passing through the medium is attenuated depending on the density of the medium. Max [Max95] derives the following equation for a light absorbing participating medium

$$I(s) = I_0 \exp\left(-\int_0^s \tau(t)dt\right) \tag{1}$$

which gives the light intensity $I(s)$ at distance s. I_0 is the light intensity at $s = 0$ where the ray enters the volume. $\tau(t)$ denotes the extinction coefficient at position t in the medium, which gives the fraction of light that is absorbed rather than let through.

A good example for light-emitting media are hot particles in a flame. The amount of light which is emitted along a ray can be described by

$$I\left(s\right) = I_0 + \int_0^s g\left(t\right) dt \tag{2}$$

where $g(t)$ is called the source term. I_0 is the amount of light which enters the medium. The integrated emission along the ray is simply added to the light that entered the medium from the outside.

Real particles both absorb and emit light, so the equations for absorption and emission have to be combined. Max derives an equation which gives the intensity at the eye

$$I\left(D\right) = I_0 \exp\left(-\int_0^D \tau\left(t\right) dt\right) + \int_0^D g\left(s\right) \exp\left(-\int_s^D \tau\left(t\right) dt\right) ds \tag{3}$$

Because participating media consist of small particles, light is not just reflected or refracted by the medium, but scattered. That means that at arbitrary points, light is scattered in different directions. The way light is scattered is defined by so-called phase-functions. In order to understand phase-functions, another term has to be defined. The particle albedo of a participating medium gives the fraction of the extinction which represents scattering rather than absorption. Clouds or snow, for example, have a very high albedo and therefore appear very bright. Soot, in contrast, has a very low albedo and therefore it appears very dark.

Phase functions describe the way light is scattered by a participating medium. They return the fraction of light which is scattered from the lightsource into the eye. Two different classes of phase functions can be distinguished - isotropic and anisotropic phase functions. In an isotropic medium, light is scattered uniformely in all directions, whereas in an anisotropic medium, scattering depends on the angle between the incident and outgoing direction of light. I.e. certain kinds of fog tend to scatter more light back to the lightsource than in the forward direction. This phenomenon is called backward-scattering.

In a medium with low albedo and low density it is unlikely that a ray of light is scattered more than once before leaving the medium. Therefore it is sufficient to consider only light that is scattered from the light source directly into the eye.

In the simplest approach it is assumed that light reaches the particles from a distant lightsource (or lightsources) and is not blocked by objects or absorbed by the participating medium. Max gives a general shading rule for this approach:

$$S(X, \omega) = r(X, \omega, \omega')i(X, \omega') \tag{4}$$

where $i(X, \omega')$ is the incoming light reaching X flowing in direction ω'. $r(X, \omega, \omega')$ is the BRDF (bidirectional reflection distribution function) which describes which fraction of the light coming in from direction ω' to point X is reflected in the direction of ω. A rule especially suited for volume rendering is

$$r(X, \omega, \omega') = a(X)\tau(X)p(\omega, \omega') \tag{5}$$

where $a(X)$ is the particle albedo, $\tau(X)$ the extinction coefficient and p is the phase function describing the directionality of the scattering. The term $S(X, \omega)$ can simply be added to the source term g

$$g(X) = e(X) + S(x) \tag{6}$$

where $e(X)$ is the direct emission at position X and $S(X)$ the in-scattered light at position X. If the source term is defined this way, equation 3 can be used to handle direct emission as well as scattering.

The above approach is quite simple, but does not account for shadows. Clouds, for example, often appear darker on the side which is opposite to the sun, because the clouds itself absorb light and shadow themselves from the sun. In order to handle shaded scattering, equation 3 has to be refined. Max [Max95] presents a solution, where a shadow-feeler is sent to the lightsource for each point X along the primary ray. Then, the amount of incoming light at each of these points along the primary ray is diminished using the absorption value along the shadow feeler.

To render even more accurate images, multiple scattering effects have to be taken into account. This means, that light is scattered more than once before it reaches the eye. In participating media with high albedo, like clouds, the influence of multiple scattering cannot be ignored. Modelling multiple scattering is a very demanding task - the problem is comparable to the radiosity problem, but instead of surfaces which can receive light from all other surfaces, volume elements receive light from all other volume elements. In order to calculate multiple scattering effects, different methods have been presented to calculate approximate solutions [RT87], [KV84], [Max95], [Sta95].

2.4 Calculation Methods

In the course of implementing a volume tracing algorithm, it is necessary to evaluate the integral equations for absorption, emission and scattering. The problem is, that for all but the most trivial scenes, this can not be done analytically. The only possible solution is to evaluate the equations by means of numerical methods. The simplest numerical approximation to an integral $\int_0^D h(x)dx$ is the Riemann sum $\sum_{i=0}^{n} h(x_i)\Delta x$. The interval $[0, D]$ is divided up into n equal segments and for each segment a sample x_i is choosen. The length of a segment is $\Delta x = D/n$.

If shaded rendering is used, it has to be considered, that the sourceterm g will also include a Riemann sum to approximate the absorption and emission properties of the "shadow feeler" between the sample point and the lightsource.

If the number of segments is chosen too low, aliasing effects may occur due to undersampling of the underlying density functions. Undersampling will result in striped images (similar to Mach-Bands) and loss of detail.

Participating media may also be incorporated into rendering systems using global illumination, yielding some of the most impressive computer generated images produced so far. Important work has been presented by Jensen [JC98], [Jen96], as well as Lafortune [LW96].

3 A Flexible Framework

A framework for volume tracing [Mai99] can be split into a number of distinct building blocks. It is essential that the interfaces between these blocks are accurately defined, so that they can interact smoothly and efficiently. The basic building blocks can be identified as

- density functions
- transfer functions
- phase functions
- participating material

Additionally, the framework should also meet the following design criteria:

- Seamless integration into an existing rendering system
- Assignment of participating media to objects of arbitrary topology
- Participating materials have to handle arbitrary combinations of absorption, emission and scattering functions
- Absorption-, emission- and scattering-functions have to handle arbitrary combinations of transfer functions
- Transfer functions may be based on arbitrary combinations of density functions
- Density functions may be based on arbitrary types of data
- Standardized interfaces (i.e. procedural density functions and volume data sets should have the same interface)
- Modularity (adding new types of functions without interfering with the rest of the framework).

3.1 Density Functions

The interface of density functions consists of a single function $getDensity(X)$, which simply returns a scalar density value for each point X in space. Procedural density functions [Per85] (i.e. noise, turbulence, checker, ...), volume data sets (different interpolation filters can be used [MN88] [ML94]) and density emitters [Ebe93] will be mapped to this single interface function. Because it would be rather boring if only one density function at a time could be used, it has to be possible to combine different density functions. Therefore, a set of arithmetic functions has to be provided to be able to add, substract, multiply and so on on, different density functions. Of course these arithmetic operators are implemented as density functions themselves. With this approach, expressions of arbitrary complexity can be constructed.

3.2 Transfer Functions

Transfer functions in our framework use color-maps as well as special algorithms like iso-surface or region boundary algorithms to map scalar density values to renderable properties. Again, only a single interface function is needed - $getColor(X)$, which returns a color for each point X in space, using the underlying density function. In order to combine different transfer functions, 2 additional functions, taking transfer functions as arguments and presenting themselves again as transfer functions, are used.

The first transfer-function-combination-function is called $GeneralTransfer$. It is quite powerful and allows for the combination of an arbitrary number of transfer functions with associated weights. The values at each point X are scaled by their associated weight and then summed up. The result is a new transfer function which is a weighted sum of other transfer functions:

$$GeneralTransfer.getColor(X) = \sum_{i=1}^{n} w_i(f_i.getColor(X))$$

where w_i is the weight associated with the i-th density function and f_i is the i-th transfer function.

The second functions is called $MappedTransfer$ and allows for the selection of a single transfer function out of a set of transfer functions, depending on the value of a density function. This sounds quite complicated, so here is an example: if a density function *cloud* is used to model a cloud, then perhaps it would be a nice effect to use different emission or scattering functions for the cloud depending on the density at a point X. MappedTransfer functions are defined similar to color maps. Given a density function $d(X)$ and a number of $(density_i, transferfunction_i)$-pairs, the mapped transfer functions return an interpolated color value:

$$MappedTransfer_{d(X)=d_i}(X) = color_i$$

or, for in-between values, colors are again linearly interpolated:

$$MappedTransfer_{d_i<d(X)<d_{i+1}}(X) = \frac{d_{i+1} - d(X)}{d_{i+1} - d_i} color_i + \frac{d(X) - d_i}{d_{i+1} - d_i} color_{i+1}$$

3.3 Phase Functions

In the framework, phase functions are embedded in so-called scattering functions. Scattering functions are based on a density function and a phase function, whereas the phase function defines the amount of light that is in-scattered into the viewing direction at a point X and the density function defines the albedo at this point X. Different phase functions can be used: isotropic scattering, Lambert scattering, Henyey-Greenstein scattering, Mie scattering, ... The simple interface to phase functions is $getColor(X)$. Because in our framework phase functions act the same way as transfer functions (returning a color) they also can be combined the same way, using $GeneralTransfer$ and $MappedTransfer$ functions.

3.4 Participating Material

What kind of interface is needed to seamlessly incorporate a participating material into a general purpose rendering system. Of course this depends largely on the rendering system. However, some general guidelines can be outlined. Usually, a rendering system will already possess some kind of general material class, which has some basic properties like colour and refraction index. In addition, it will make available some kind of function used to communicate with the rendering algorithm. Given a segment of a ray, the basic functionality of such a function is usually to return a value which represents the illumination along the given segment, as well as a filter value, which is similar to the extinction in volume rendering and determines how much illumination is absorbed along the ray segment. In order to succesfully integrate a participating material into a rendering system, the material class "simply" has to implement all methods necessary to communicate with the rest of the rendering system.

The core of the participating material class is an algorithm which is able to evaluate all material properties along a given ray. That means it has to determine the amount of absorption and emission (direct emission as well as in-scattered light). This can be done, for example, by ray-marching, which is simply an algorithm which integrates material properties along the ray using a Riemann approximation.

The Ray Marching algorithm subdivides the ray into equal segments and takes a sample of the participating materials properties for each segment. The segments correspond to the segments used in the Riemann approximation. As a result, the integral of the participating materials properties along the viewing ray can be approximated by the sum of the sample values multiplied by the length of one segment (see Figure 1).

4 Implementation

The flexible framework for volume tracing has been implemented as a part of the "Advanced Rendering Toolkit" (ART for short), which is being developed at the

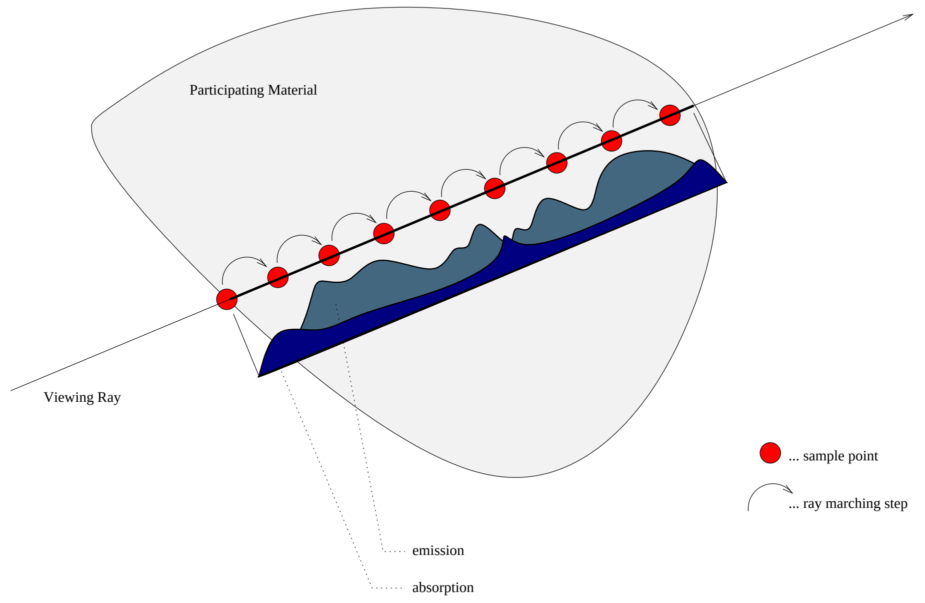

Figure 1: Ray-Marching

Institute of Computer Graphics in Vienna. ART is a set of Objective C Libraries that provide a wide range of functionality suitable for graphics applications. The ART libraries do not deal with the user interface, they provide classes and methods starting with primitive graphics objects like vectors, points and matrices up to classes that make it possible to define complete three dimensional scenes and a number of different methods to manipulate and render these scenes.

5 Summary and Results

In this paper, a flexible framework for volume tracing, which allows for the arbitrary combination of different density functions, transfer functions and phase functions in a participating material, has been presented. Participating media have absorption, emission and scattering properties and are rendered using a ray-marching algorithm. The framework has been implemented as an add-on to the Advanced Rendering Toolkit. Participating media can be integrated seamlessly into “conventional” scenes. No restrictions whatever apply. Density functions can be defined by means of procedural density functions (like in solid texturing), density emitters and volume data sets. Different kinds of filters may be applied to volume data sets (Mean, Gaussian, Laplace, . . .) to improve data quality before rendering. Furthermore, isosurfaces and region boundaries of volume data sets may be rendered. Finally, different types of volume data interpolation can be used. Transfer functions can be designed by combining arbitrary numbers of transfer function “primitives” like constant-, colormap-, general- and mapped transfer functions. Constant transfer functions may be used to model homogenous-atmosphere-like participating media. Colormap transfer functions are used to map density values to arbitrary color maps. General- and mapped transfer functions can be used to combine arbitrary

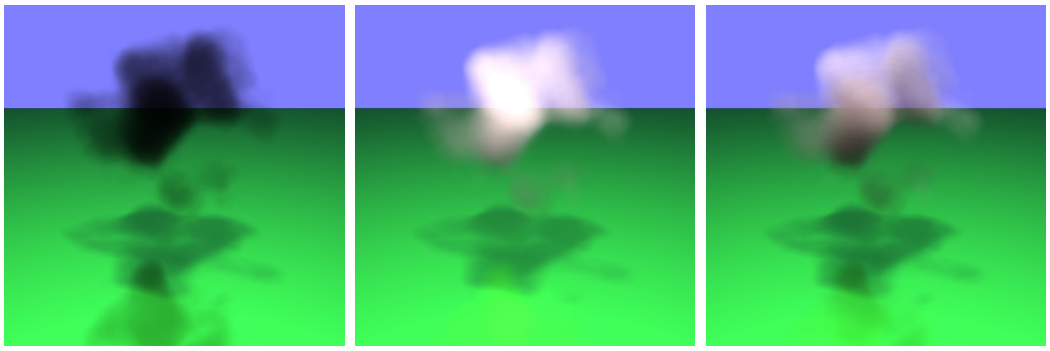

Figure 2: Absorption, Scattering and Shaded Scattering: a cloud, composed out of noise, tubulence and a spherical density emitter is shown. The left cloud only absorbs light, which results in a black-smoke-like appearance. The second cloud has been rendered using absorption and scattering, yielding a more natural appearance. Finally, in the right image, shaded scattering has been used.

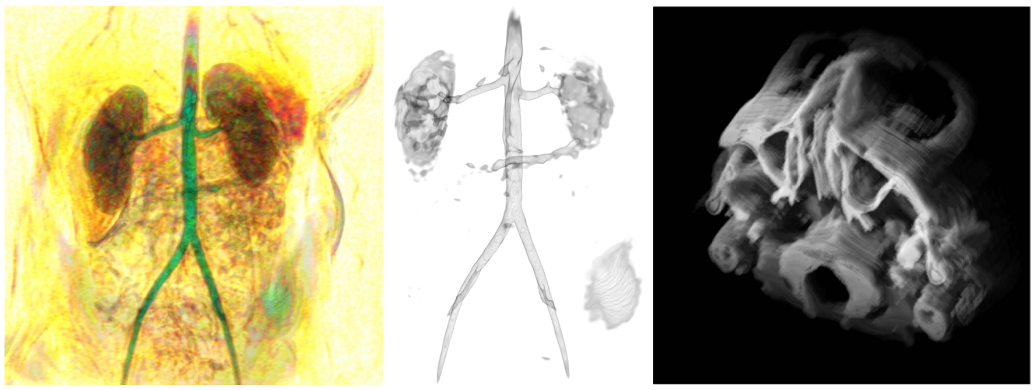

Figure 3: Volume Data Sets (kidneys, section of a human head, Courtesy VisMed-Project): the left image has been rendered using trilinear interpolation, as a result, no distinct voxels can be seen. The second image showns the same data-set using Levoys isosurface algorithm. The third image has been rendered using shaded scattering.

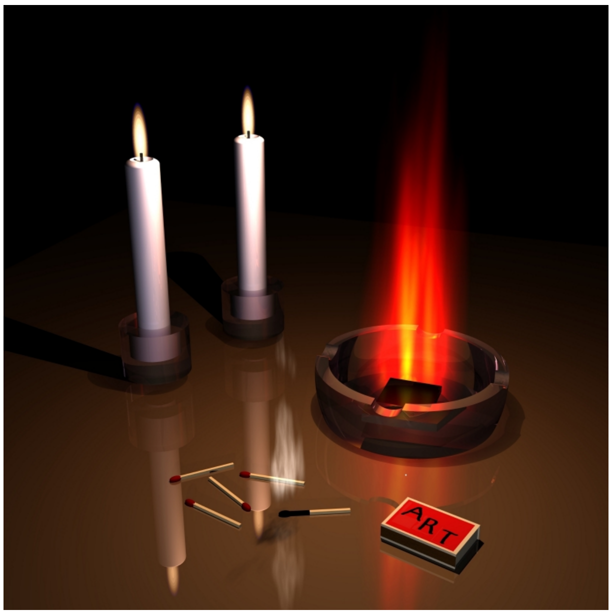

Figure 4: Natural Phenomena: Two candlelights, one quite turbulent flame and some smoke rising from a burned match. Although both the candlelights and the flame are rendered using emission, these media do not act as light sources, which results in a somewhat unrealistic lighting of the scene.

numbers of transfer functions. Different kinds of phase functions are used to define scattering characteristics of participating media. Isotropic, anisotropic, Lambert, Henyey-Greenstein and Mie scattering have been implemented. Although some very realistic images of natural phenomena and different types of volume data sets can be rendered, some ideas would deserve further investigation in future work: More sophisticated density functions may be examined, for example, physically based models to create more convincing images of natural phenomena. Wavelength dependent scattering functions should be implemented quite easily because ART itself already is able to perform wavelength-dependent rendering. Furthermore, right now, participating media do not act as light sources even if they possess an emission property. This shortfall could be circumvented by means of participating media as volume light sources, or by means of global illumination modells. Global illumination models also would be useful in rendering effects like volume caustics and indirect lighting by participating media.

References

[Ebe93] D. S. Ebert. Design and animation of volume density functions. *The Journal of Visualization and Computer Animation*, 4(4):213–232, October–December 1993.

[Elv92] T. Todd Elvins. A survey of algorithms for volume visualization. In *Computer Graphics*, volume 15, pages 194–201, August 1992.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 311–320. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[KV84] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 165–174, July 1984.

[LW96] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 91–100, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[Mai99] Stefan Maierhofer. A flexible framework for volumetracing. Master's thesis, University of Technology Vienna, 1999.

[Max95] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995. ISSN 1077-2626.

[ML94] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of the Conference on Visualization '94*, pages 100–107, October 1994.

[MN88] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer graphics. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 221–228, August 1988.

[Per85] Ken Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 287–296, July 1985.

[RT87] Holly E. Rushmeier and Kenneth E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. In *ACM Computer Graphics*, volume 21, pages 293–302, July 1987.

[Sta95] Jos Stam. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.