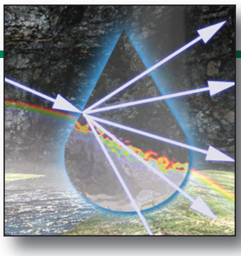

Geometric and Path Tracing Methods for Simulating Light Transport through Volumes of Water Particles

by James Hegarty

Abstract

The visual appearance of volumes of water particles, such as clouds, waterfalls, and fog, depends both on microscopic interactions between light rays and individual droplets of water, and also on macroscopic interactions between multiple droplets and paths of light rays. This paper presents a model that builds upon a typical single-scattering volume renderer to correctly account for these effects. To accurately simulate the visual appearance of a surface or a volume of particles in a computer-generated image, the properties of the material or particle must be specified using a Bidirectional Reflectance Distribution Function (BRDF), which describes how light reflects off of a material, and the Bidirectional Transmittance Distribution Function (BTDF), which describes how light refracts into a material. This paper describes an optimized BRDF and BTDF for volumes of water droplets, which takes their geometry into account in order to produce well-known effects, such as rainbows and halos. It also describes how a multiple-scattering path tracing volume integrator can be used to more accurately simulate macroscopic light transport through a volume of water, creating a more "cloudlike" appearance than a single-scattered volume integrator. This paper focuses on replicating the visual appearance of volumes of water particles, and although it makes use of physical models, the techniques presented are not intended to be physically accurate.

Analysis of Desired Visual Effects

On a macroscopic level, light that a viewer sees when he/she looks at a volume of water particles is light that travels from a light source, interacts with water particles, and then hits the viewer's eye. There is angular and wavelength dependence between the amount of light that enters and exits a water droplet. Vibrant bands of color appear in certain viewing situations, depending on the angles between the viewer, the light source, and the volume of particles. Viewers see rainbows when the sun is behind them. The light from the sun enters droplets at certain angles and then bounces back at the viewer in the distinctive pattern of a colored bow. Because of the large angle at which this occurs, typically only half of the rainbow can be seen. However, the band appears in a conic shape in front of the viewer; theoretically, the whole circle of the rainbow could be seen. Viewers see glories and halos when water particles are between the viewer and the light; the effect seen is a concentration of rays transmitted through the drops towards the viewer. It should be noted that these effects only appear when the sun is the light source, because it acts as an ideal directional light. Typical omnidirectional lights, such as tungsten bulbs, would not exhibit these effects because rainbows, halos, and others would get "blurred out" as the light from multiple directions enters a volume of particles and creates multiple "copies" of the pattern on top of itself, thus obscuring the pattern. Methods of simulating these macroscopic effects with a microscopic model will now be described.

Microscopic Water Droplet Light Transport

Water droplet light transport on a microscopic level can be approximated by tracing rays through a spherical object with a BRDF and BTDF similar to water. Robert Greenler's book *Rainbows, Halos, and Glories* describes the physical interactions between light and water droplets [1]. As a light ray enters a water drop, it bounces around in the drop multiple times, each time reflecting part of the light back into the drop and refracting part of it out of the drop, based on elementary physical models of wave transport. The refracted light obeys Snell's law (Formula 1), and the reflected light obeys the reflection law (Formula 2). The combination of reflected and refracted light depends on Fresnel's reflection law (Formula 3). An example of a single ray traced through the sphere is shown in Figure 1. The black line on the left is the ray that enters the sphere. It is bent when it is inside the sphere, and it is bent again as it leaves the sphere as an exiting ray, the red line. The blue lines are multiple reflections of this ray inside the sphere; the rays resulting from these internal reflections do not have sufficient energy to be visible, so they are not shown.

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

***Formula 1**: Snell's law:* n *is the index of refraction and* theta *is the ray angle relative to the normal* [8].

$$\hat{R} = \hat{I} - 2(\hat{I}\cdot\hat{N})\hat{N}$$

***Formula 2**: Vector reflection law:* I *is the incident ray,* N *is the normal, and* R *is the refracted ray* [2].

$$R_s = \left[\frac{\sin(\theta_t - \theta_i)}{\sin(\theta_t + \theta_i)}\right]^2 \quad R_p = \left[\frac{\tan(\theta_t - \theta_i)}{\tan(\theta_t + \theta_i)}\right]^2 \quad R = \frac{(R_s + R_p)}{2}$$

***Formula 3**: Fresnel's equations:* R *is the strength of the reflected light and* theta *is the angle of incident and transmitted rays relative to the normal* [7].

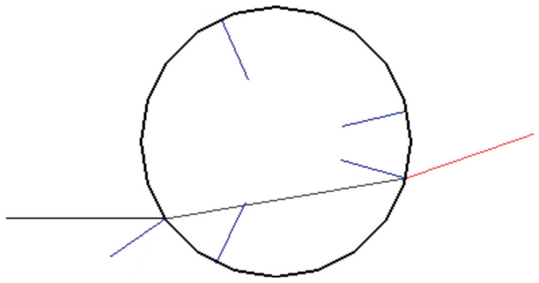

Figure 1: An example of a single ray interacting with a drop.

To use this model to create images, define a "droplet" function that describes how much light exits the sphere, and in which direction, when light enters in a certain direction. This function is the probability that a ray will enter and exit the sphere in another direction, both initially defined as 3-D vectors. Snell's law and the reflection law are used to trace rays through a sphere of radius, ***r***. For simplicity, take the limit as ***r*** approaches 0, and reduce the vector representation of the ingoing and outgoing rays to a directional latitude/longitude on the sphere. If the sphere is at the origin, all rays into the infinitesimal sphere will be in a plane intersecting the upward axis, where the incoming ray lies along the plane. Due to the properties of both Snell's law and the reflection law, all propagated rays will also lie on this plane. The function is identical for all angles in the axis of longitude; therefore, this degree of freedom can be removed. Now, reformulate the function in terms of the angle between the incoming direction and exiting direction. This function is rotationally invariant due to the rotational symmetry of the sphere; therefore, incoming light only needs to be considered from one direction. Outgoing light corresponding to other incoming directions will simply be a rotated version of the original incoming/outgoing pair.

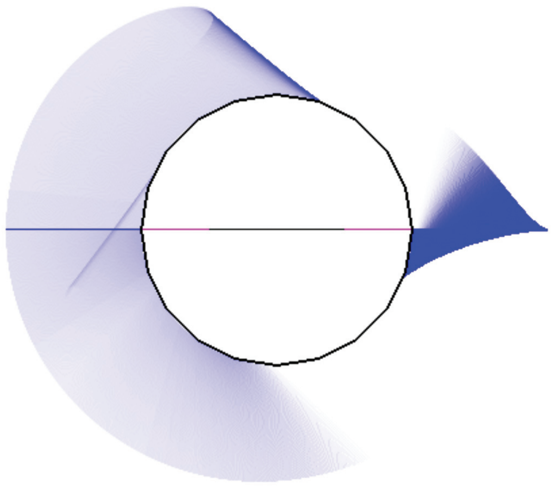

Figure 2: A plot of all of the rays that exit the sphere from one direction.

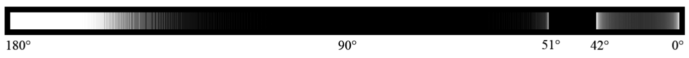

Figure 3: A plot of light intensity leaving the sphere, in terms of the cosine of the angle between the incidence vector and refraction vector.

To apply this theory to create the BRDF and BTDF, use a unit sphere to calculate the pairs of incoming and outgoing rays by randomly shooting rays into one side of the sphere in one direction and tabulating how many rays exit in a certain direction. This data can be precomputed and tabulated into discrete buckets in terms of the cosine of the angle between the entering and exiting ray; with a reasonable number of samples (~1000 rays were used in the sample implementation), the precomputed data produces a visually accurate BRDF and BTDF.

Figure 3 shows that the intensity of light is highly angularly dependent. Particularly, a visually significant amount of light is focused at 42 degrees, and a secondary band appears at 51 degrees. Figure 2 shows the same rays as Figure 3 plotted from the side of the drop, with the light entering from the left. This property of water droplets creates the macroscopic rainbow effect; a rainbow arc around the viewer occurs primarily at 42 degrees, and a secondary arc that is not always visible to the eye occurs at 51 degrees.

Wavelength Dependent Effects

Snell's law depends on the refraction index of the medium that the light is refracted through. In the case of water, the refraction index is slightly different for different wavelengths of light. Due to this effect, the areas of focused light seen above are located at slightly different angles for different wavelengths, producing the colored bow. White light is made up of a range of wavelengths. For the model to be accurate, the pattern of outgoing rays that result from different wavelengths must be accounted for. The physics that relate refraction index to wavelength is quite complicated, but Paul Huibers created a sufficiently accurate approximation of the wavelength dependence of the refraction index based on physical measurements (Formula 4) that produces good results for these purposes [4].

$$n(\lambda)=1.31279+15.76212\,\lambda^{-1}+4382\,\lambda^{-2}\boxplus 1.1455\,x\,10^{6}\,\lambda^{-3}$$

Formula 4: Huibers' index of refraction formula for water.

In order to account for the wavelength dependent effects, the table of angular values discussed previously is computed at various different wavelengths. In the sample implementation, wavelengths are recorded from 400 nm to 700 nm, which is the range of wavelengths of visible light. The wavelengths are recorded in 15 nm intervals; this number was chosen because it was the largest interval that did not produce noticeable banding in test images. A smaller interval could be chosen to reduce banding if necessary. Later, when all of the light rays are gathered, calculations are done separately in each of these wavelength buckets and then converted to XYZ color using the CIE matching functions provided by Pharr and Humphreys [4]. The CIE, or the International Commission on Illumination, is an international committee that sets color and illumination standards, such as defining the conversion from wavelengths to colors on computer screens. CIE XYZ color can easily be converted to the RGB color space typically used in computer-generated images. An extensive discussion of why the CIE matching functions produces correct results in the RGB color space is beyond the scope of this paper; it is covered in detail by Pharr and Humphreys. Essentially, these functions integrate over all wavelengths for X, Y, and Z and weight each wavelength depending on how much it contributes to each of these three color components. This method of accounting for wavelength dependence is not the most efficient method, but it is flexible and easy to implement.

Single-Scattering Volume Rendering Using Ray Tracing

A ray tracing renderer works by gathering rays of light from a scene that correspond to a pixel in the resulting image. Each ray interacts with the scene, determines how much light hits the "virtual film" at a location corresponding to a pixel, and then records this as the color and intensity of light associated with this pixel. For efficiency, almost all ray tracers actually solve the reverse problem; instead of considering rays coming from the light and hitting the camera, as would seem intuitive from physics, they consider rays that originate at the camera and hit the light. This algorithm is considerably faster, as the vast number of paths of light rays that leave the light but never hit the camera are ignored. A basic result from radiometry is that these two methods of tracing rays are equivalent; rays interact with the scene the same in the forward and backward direction [4]. An important characteristic of ray tracing algorithms is that they never produce the "correct" result. A correct, or true-to-life result, is impractical to compute, as it would require an almost infinite number of ray interactions to be calculated. The efficiency of ray tracing algorithms is measured by comparing how many ray interactions need to be calculated versus the difference (called variance) between the true-to-life result and the actual image produced by the algorithm. Ray tracing algorithms that produce low noise with a set number of rays are more efficient than algorithms that produce high noise with the same number of rays.

A single-scattering volume integrator is an extension of the ray tracing algorithm that makes light interact correctly with volumes of particles. Typically, ray tracers make the assumption that there will be no interactions with empty space, only interactions with surfaces, so they ignore volumetric data. In reality, light traveling through a volume of particles will be absorbed, emitted, and scattered. If the particles absorb light, then rays from the light to the camera are decreased in intensity in proportion to the absorption factor of the particles. Some particles may actually emit light; often this is physically incorrect but used because it produces practical and attractive visual results. Scattered light is light that hits a particle and changes direction. A simulator must account for absorption, emission, and scattering to accurately simulate light transport through volumes of particles. In a renderer, a volume integrator serves this purpose and collects the light along rays that interact with volumes of particles based on this physical model.

A single-scattering volume integrator "marches" along a ray that enters a volume of particles at a certain step distance, usually provided by the user. At each step, the integrator sends a ray in the direction of the light, and it once again performs a marching algorithm to determine how much light went from the source to the point that is being considered in this step. The light along this ray then contributes to the total amount of light that hits the virtual camera. Because this algorithm only considers single-scattering, it does not recurse; it only calculates paths directly from the light to the camera. It is not physically correct because, in reality, light can scatter in all directions, not just directory toward the light. The light ray is modulated at every step by the water droplet function described in the previous section. For the sample implementation, the integrator is biased to weight the amount of light contributed by the direct, rainbow-producing light based on a user-given parameter, so that the rainbow's intensity could be controlled for artistic effect.

Sample Implementation

The sample implementation of this algorithm used to produce the images in this paper was based on the Physically Based Ray Tracer (PBRT) renderer [4]. The default single-scattering VolumeIntegrator class was extended to include the techniques described above. Figure 4 shows a full scene rendered using this technique.

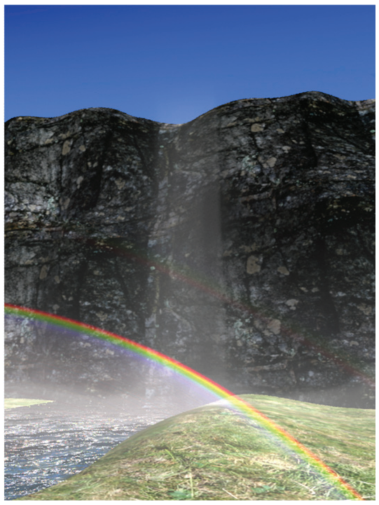

Figure 4: Example image rendered using single-scattering water droplet algorithm.

Extension: Multiple-Scattering Integrator

Although the single-scattering-based integrator was successful at producing rainbows, it is possible to more accurately simulate the effect of light entering a volume of particles and interacting with transparent particles before it reaches the viewer. This should reduce the "flat" appearance of the water spray in Figure 4 and make it appear more voluminous and cloudlike. To accomplish this, a multiple-scattering integrator based on path tracing could be used.

A multiple-scattering integrator differs from a single-scattering integrator in that each ray that enters a volume region is simulated to randomly walk through the region at a step distance provided by the user, instead of light rays marching along in one direction. At each step, the ray randomly continues straight or bounces off in a random direction. This corresponds to the physical concept of a light ray having traveled through space, and either continuing straight or being deflected, based on whether it actually hit a water particle or not. The probability that a ray gets deflected depends on the density of particles at its current location. In areas of zero density, the ray is never deflected, so it continues straight into the scene (never scattering). This is the behavior that would be expected for rays that do not interact with the particles. If the particle is deflected, the color that the ray contributes is modulated using the rainbow transmittance lookup table described earlier, based on the angle between the incident and refracted ray. If a ray ends up hitting a light, then it contributes light to the image.

Extension: Path Tracing Integrator

The integrator just described is a subset of a larger set of techniques known as **path tracing**. With path tracing, rays are sent into the scene from the camera, bounce around the scene and interact with surfaces and volumes in the scene until they reach the light source. This technique is extremely flexible, easy to implement, and produces accurate results. However, in a naïve implementation, it takes an extremely long time to converge on an image with acceptably low noise, because few of the rays sent into the scene ever reach a light in a reasonable amount of time. The result is said to have high variance, i.e., high noise.

To reduce variance, a technique called **importance sampling** is employed, so that rays are more likely to "randomly" bounce into a light. It would appear, however, that this approach would produce incorrect results, as now more rays are hitting the light than should (results are biased). In other words, the resulting image is brighter than it should be. To counter this effect, the light along the rays that hit the light is scaled with a probability factor inversely proportional to the amount of extra rays sent in a direction of the light. Therefore, if twice the amount of rays are being sent in a certain direction (towards a light), then each will be weighted ½ of what they normally would, producing an unbiased image. The result is an image with considerably less variance at basically no cost. Importance sampling can be used here because a large number of the possible angles between the incoming and outgoing ray do not contribute a significant amount of light, as seen in the plot of the droplet function in Figure 3. Therefore, the sample implementation sends fewer rays in the directions that contribute less light.

Similar to the bias issue in importance sampling, the path tracing volume integrator would be biased if rays were naïvely shot into the scene. In order for the path tracing algorithm to be unbiased, each path must be weighted based on the probability that it actually occurs. Logically, because there are more possible paths when the length of the path is long than when it is short, longer paths are less likely to occur. To account for this, each step is calculated to have a certain probability and is multiplied by the weight of all past probabilities, because as a ray proceeds through the volume, it becomes less probable that the specific ray would actually contribute to the scene. Specifically, each step is weighted by a factor of $1/(4\pi)^S$, where **s** is the length of the step size. This is based on the derivation for surface path tracing found on page 746 of Pharr, Humphrys [4]. For path tracing, a ray is weighted by the inverse of all possible rays that could occur. In this case, all rays that could occur at a certain point in the volume come from an area of 1/(surface area of a unit sphere) = $1/(4\pi)$. Because spheres of different sizes may want to be considered, the weight is reformulated to be $1/(4\pi)^S$, based on geometry. For example, a sphere of radius 3 would have the probability of $1/(4\pi)^3$, because it is essentially the probability of three unit spheres in succession. To further verify this formulation, it is also correct in the limit; as the sphere considered becomes infinitely large, the probability of a ray coming from that distance approaches 0. In addition, as the sphere becomes infinitely small, the probability approaches 1.

Sample Implementation: Path Tracing

In the sample implementation, the light source is an imaginary light given as a parameter to the integrator. As an optimization, the implementation does not use the built-in PBRT light sources. The imaginary light is specified using a 3-D origin and direction. This forms an infinite planar light, and any path that crosses this plane is considered terminated. This is not physically correct, but it was useful for practical purposes, and the implementation could be extended to use PBRT's light sources, at the expense of considerably more variance. The implementation has a maximum path size given by the user in order to prevent infinite loops in extreme cases. Paths that reach the maximum path size without hitting the light are ignored. The path tracing algorithm only collects paths that go from a light to the image plane. Including paths that never hit the light would introduce bias; the image would be too dark.

In order to retain the rainbow effect, there is a parameter in the integrator to bias the particles toward bouncing a ray directly at the light. This is not physically correct, but is desirable, because it allows the rainbow effect to be better controlled for artistic reasons.

Figure 5 is an image of a smoke dataset produced by Ron Fedkiw [4] rendered with the light to the side, using the multiple-scattering integrator, to show how it differs from the single-scattering integrator in Figure 6. Note: the angle of view is not wide enough to produce rainbows, so they do not appear.

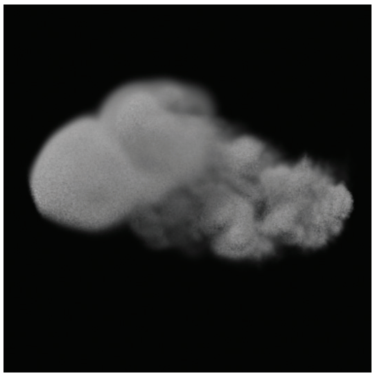

Figure 5: Multiple-scattering cloud rendering.

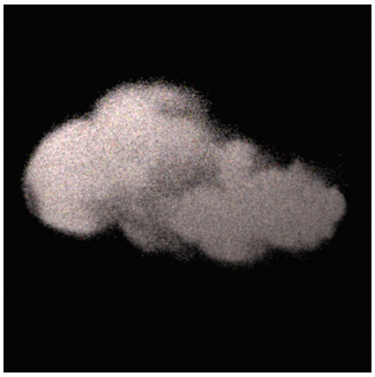

Figure 6: Single-scattering cloud rendering.

Unfortunately, although multiple-scattering does produce a more cloud-like appearance, as seen in Figure 5, it takes several orders of magnitude longer to compute images with this technique. Therefore, it is impractical to use it to render complex scenes, such as the waterfall in Figure 4, without further optimization or advanced hardware.

Future Work

An obvious extension would be to generalize the water rendering technique described so that it produces correct results when water molecules are extremely dense, to the point that they coalesce. Currently, the model assumes that all particles are spherical droplets, which is only the case with the "splashes" surrounding the waterfall image in Figure 4. The waterfall itself and pool at the base of the fall have to be rendered with another model.

The current implementation only deals with spherical water droplets, but the algorithm could be modified to produce effects that result from particles of other shapes, such as halos resulting from crystalline shapes, as described in *Rainbows, Haloes, and Glories* [1]. The directional lookup table described in part 2 would need to be recomputed to account for the different geometry, and the directional lookup would possibly need to be expanded to a higher dimension if the desired effect did not occur in a 1-dimensional angular space, as is the case with rainbows.

Another possibility would be to create a volumetric photon mapping-based version of the multiple-scattering path tracing integrator. Photon mapping is a technique developed by Henrik Jensen in which virtual "photons" are sent from the light into the scene and stored in a tree structure when they reach a certain termination criteria [5]. The renderer literally shines light on the scene and stores where it hits. Then, in the rendering phase, the camera "collects" photons near the rays that it sends into the scene, producing results that, while not physically correct, have low noise. Another technique to try would be a bidirectional path tracer as proposed by Lafortune and Willems [6]. With bidirectional path tracing, sub-paths are traced both from the camera and from the light. Paths from the light or camera side that make small contributions to the resulting image can be ignored; essentially, importance sampling is performed on the sub-paths. Then, the sub-paths can be combined, and the result is an image with considerably less variance.

Acknowledgments

Thanks to Qiqi Wang for providing the volumetric dataset and algorithm for tracing through unstructured meshes. Also, thanks to Pat Hanrahan for providing support and advice on this project.

References

1. Greenler, R. 1990. *Rainbows, Halos, and Glories*. Cambridge University Press, Cambridge, UK.
2. Hanrahan, P. Law of reflection. **https://graphics.stanford.edu/wikis/cs348b-07/Reflection1Lecture/012**.
3. Huibers, P. 1997. Models for the wavelength dependence of the index of refraction of water. *Appl. Optics 36*, 16.
4. Humphreys, G. and Pharr, M. 2004. *Physically Based Rendering*. Elsevier, San Francisco, CA.
5. Jensen, H. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Wellesley, MA.
6. Lafortune, E. and Willems, Y. D. 1993. Bi-directional path-tracing. In *Proceedings of ACM Compugraphics*. Alvor, Portugal. 145-153.
7. Nave, R. Fresnel's Equations. **http://hyperphysics.phy-astr.gsu.edu/Hbase/phyopt/freseq.html**.
8. Weisstein, E. Snell's law. **http://scienceworld.wolfram.com/physics/SnellsLaw.html**.

Biography

*James Hegarty (**jhegarty@stanford.edu**) is a sophomore at Stanford University. He enjoys photography, playing guitar, and conducting computer graphics research.*