Eurographics Symposium on Rendering (2005)
Kavita Bala, Philip Dutré (Editors)

Real-Time Multiple Scattering in Participating Media with Illumination Networks

László Szirmay-Kalos, Mateu Sbert, and Tamás Umenhoffer

Budapest University of Technology and University of Girona
Emails: szirmay@iit.bme.hu, mateu@ima.udg.es

Abstract

This paper proposes a real-time method to compute multiple scattering in non-homogeneous participating media having general phase functions. The volume represented by a particle system is supposed to be static, but the lights and the camera may move. Lights can be arbitrarily close to the volume and can even be inside. Real-time performance is achieved by reusing light scattering paths that are generated with global line bundles traced in sample directions in a preprocessing phase. For each particle we obtain those other particles which can be seen in one of the sample directions, and their radiances toward the given particle. This information is stored in an illumination network that allows the fast iteration of the volumetric rendering equation. The illumination network can be stored in two-dimensional arrays indexed by the particles and the directions, respectively. Interpreting these two-dimensional arrays as texture maps, the iteration of the scattering steps can be efficiently executed by the graphics hardware, and the illumination can spread over the media in real-time.

1. Introduction

Physically plausible rendering of participating media simulates multiple scattering effects [Max94, NDN96, LBC95]. Multiple scattering algorithms can be considered as particular ways to generate light paths connecting the light sources to the eye. The computation time can be reduced if the steps of paths are reused and are not generated again when they are needed.

Path reuse is a common technique of advanced global illumination methods developed for surface and volume rendering. Steps of the path can be reused in bi-directional and even in classical *path tracing* [LW96, BSH02]. *Metropolis light transport* reuses path parts that are left unchanged by the current perturbation [PKK00]. *Instant radiosity* [Kel97] and *photon mapping* [JC98] reuse a shooting path for all gathering paths. *Precomputed radiance transfer* [SKS02] and finite-element based iteration methods [Neu95, SK99, DMK00] can also be interpreted as path reuse techniques. Whenever the radiance of a patch or a voxel is shot or gathered, all paths ending here are simultaneously extended. Path reuse is more explicit in *global line* global illumination methods that exchange radiance along globally sampled lines [BF89, Sbe96, Pre65]. Global lines are worth organizing in parallel or perspective bundles since this allows the exploitation of the rendering hardware and the application of incremental scan conversion algorithms [BF89, Neu95, SK99, DMK00]. Global lines form an *illumination network*, which can replace the geometry of the surfaces or the density of the volume in illumination computations [HDKS00, SMKY04].

In this paper we extend the global line bundle illumination network concept to participating media represented by a particle system, and obtain the particle radiance with iteration. Since the iteration works on the illumination network, it does not require ray casting or queries of the particle system. Encoding the illumination networks by textures, the GPU can calculate a scattering step on all particles and in all sampled directions rendering a single textured quadrilateral.

2. Multiple scattering in volumes

Let us consider how the light goes through participating media. The change of radiance L on path of length ds and of direction $\vec{\omega}$ depends on different phenomena:

- *Absorption* and *outscattering*: the light is absorbed or scattered out from its path when photons collide with the particles. If the probability of collision in a unit distance is τ, then the radiance changes by $-\tau \cdot L \cdot ds$ due to the collisions. After collision a particle may be either absorbed or reflected with the probability of *albedo* a.

- *Emission*: the radiance may be increased by the photons emitted by the participating media (e.g. fire). This increase is $L^e \cdot ds$ where L^e is the emission density.
- *Inscattering*: photons originally flying in a different direction may be scattered into the considered direction. The expected number of scattered photons from differential solid angle $d\omega'$ equals to the product of the number of incoming photons and the probability that the photon is scattered from $d\omega'$ to $\vec{\omega}$. The scattering probability is the product of the collision probability (τ), the probability of not absorbing the photon (a), and the probability density of the reflection direction, called *phase function* P. We use the Henyey-Greenstein phase function [HG40, CS92]:

$$P(\vec{\omega}', \vec{\omega}) = \frac{1}{4\pi} \cdot \frac{3(1-g^2) \cdot (1 + (\vec{\omega}' \cdot \vec{\omega})^2)}{2(2+g^2) \cdot (1 + g^2 - 2g(\vec{\omega}' \cdot \vec{\omega}))^{3/2}},$$

where $g \in (-1, 1)$ is a material property describing how strongly the material scatters forward or backward.
Taking into account all incoming directions Ω', we obtain the following radiance increase due to inscattering:

$$ds \cdot \tau(s) \cdot a(s) \cdot \left(\int_{\Omega'} L(s, \vec{\omega}') \cdot P(\vec{\omega}', \vec{\omega}) \, d\omega' \right).$$

Adding the discussed changes, we obtain the following *volumetric rendering equation* for radiance L of the ray at $s + ds$:

$$L(s+ds, \vec{\omega}) = (1 - \tau(s) \cdot ds) \cdot L(s, \vec{\omega}) + ds \cdot L^e(s, \vec{\omega}) +$$
$$ds \cdot \tau(s) \cdot a(s) \cdot \int_{\Omega'} L(s, \vec{\omega}') \cdot P(\vec{\omega}', \vec{\omega}) \, d\omega'. \qquad (1)$$

2.1. Particle system model

The particle system model of the volume corresponds to a discretization, when we assume that scattering can happen only at N discrete points called particles. We assume that particles are sampled randomly, preferably from a distribution proportional to collision density τ, and we do not require them to be placed at grid points [GRWS04]. As demonstrated in [HL01] such particle systems can generate acceptable clouds with a few hundred particles. Let us assume that particle p represents its spherical neighborhood of diameter Δs_p, and introduce its *opacity* as $\alpha_p = 1 - e^{-\tau_p \cdot \Delta s_p}$, its *emission* as $E_p = L^e \cdot \Delta s_p$, its *incoming radiance* by I_p, and ist *outgoing radiance* by L_p. The *discretized volumetric rendering equation* at particle p is then:

$$L_p(\vec{\omega}) = (1 - \alpha_p) \cdot I_p(\vec{\omega}) + E_p(\vec{\omega}) +$$
$$\alpha_p \cdot a_p \cdot \int_{\Omega'} I_p(\vec{\omega}') \cdot P_p(\vec{\omega}', \vec{\omega}) \, d\omega'.$$

In homogeneous media, albedo a and phase function P are the same for all particles. In non-homogeneous media, these parameters are particle attributes [REK*04].

3. The proposed solution method

We solve the discretized volumetric rendering equation by iteration. The volume is represented by a set of randomly sampled particle positions. Suppose that we have an estimate of particle radiance values (and consequently, of incoming radiance values) at iteration step $n-1$. The new particle radiance in iteration step n is obtained by substituting these values to the right side of the discretized volumetric rendering equation:

$$L_p^n(\vec{\omega}) = (1 - \alpha_p) \cdot I_p^{n-1}(\vec{\omega}) + E_p(\vec{\omega}) +$$
$$\alpha_p \cdot a_p \cdot \int_{\Omega'} I_p^{n-1}(\vec{\omega}') \cdot P_p(\vec{\omega}', \vec{\omega}) \, d\omega'. \qquad (2)$$

This iteration is convergent if the opacity is in $[0, 1]$ and the albedo is positive and less than 1, which is always the case for physically plausible materials.

In order to calculate the directional integral representing the inscattering term of equation 2, we suppose that D random directions $\vec{\omega}_1, \ldots, \vec{\omega}_D$ are obtained from uniform distribution of density $1/(4\pi)$, and the integral is estimated by Monte Carlo quadrature:

$$\int_{\Omega'} I_p(\vec{\omega}') \cdot P_p(\vec{\omega}', \vec{\omega}) \, d\omega' \approx \frac{1}{D} \cdot \sum_{d=1}^{D} I_p(\vec{\omega}'_d) \cdot P_p(\vec{\omega}'_d, \vec{\omega}) \cdot 4\pi.$$

Note that replacing the original integral by its approximating quadrature introduces some error in each iteration step, which accumulates during the iteration. This error can be controlled by setting D according to the albedo of the participating media since if the error of a single step is ε and the albedo is a, then the error is bound by $\varepsilon/(1-a)$.

3.1. Building the illumination network

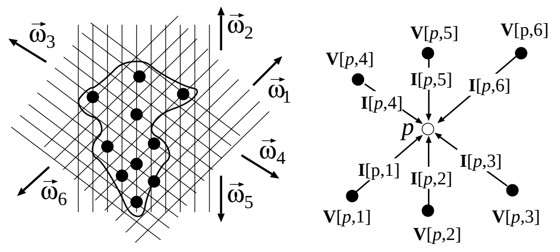

Figure 1: *Illumination and visibility networks*

If we use the same set of sample directions for all particles, then the incoming radiance and therefore the outgoing radiance are needed only at these directions during iteration.

For a single particle p, we need D incoming radiance values $I_p(\vec{\omega}_d)$ in $\vec{\omega}_1,\ldots,\vec{\omega}_D$, and the reflected radiance needs to be computed exactly in these directions. In order to update the radiance of a particle, we should know the indices of the particles visible in sample directions, and also the distances of these particles to compute the opacity. This information can be stored in two-dimensional arrays **I** and **V** of size $N \times D$, indexed by particles and directions respectively (figure 1). Array **I** is called the *illumination network* and stores the incoming radiance values of the particles on the wavelengths of red, green, and blue. Array **V** is the *visibility network* and stores index of visible particle *vp* and opacity α for each particle and incoming direction, that is, it identifies from where the given particle can receive illumination (figure 2).

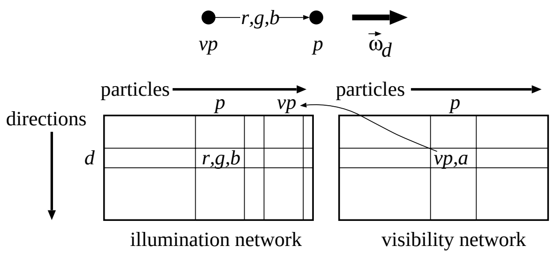

Figure 2: *Storing the networks in arrays*

In order to handle emissions and the direct illumination of light sources, we use a third array **E** that stores the sum of the emission and the reflection of the direct illumination for each particle and discrete direction. This array can be initialized by rendering the volume from the point of view of the light source and identifying those particles that are directly visible. At a particular particle, the discrete direction closest to the illumination direction is found, and the reflection of the light source is computed from the incoming discrete direction for each outgoing discrete direction.

Visibility network **V** expressing the visibility between particles is constructed during a preprocessing phase (figure 3). The bounding sphere of the volume is constructed and then D uniformly distributed points are sampled on its surface. Each point on the sphere defines a direction aiming at the center of the sphere, and also a plane perpendicular to the direction. A square window is set on this plane to include the projection of the volume, and the window is discretized to $M \times M$ pixels.

When a particular direction is processed, particles are orthographically projected onto the window, and rendered using a standard z-buffer algorithm, having set the color of particle p equal to its index p (figure 3). The contents of the image and depth buffers are read back to the CPU memory, and the indices and depths of the visible particles are stored together with the pixel coordinates. The particles that were visible in the preceding rendering step are ignored in the subsequent rendering steps. Repeating the rendering for the

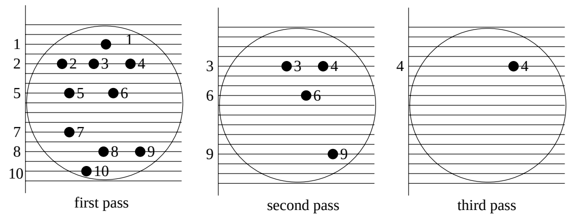

Figure 3: *Constructing the visibility network*

remaining particles and reading back the image and depth buffers again, we can obtain the indices of those particles which were occluded in the previous rendering step. Pairing these indices to those previously obtained ones which have the same pixel coordinates, we can get the pairs of particles that occlude each other in the given direction. On the other hand, the difference of the depth values is the distance of the particles, from which the opacity can be computed. Repeating the same step until the image is empty, we can build lists of particles that are projected onto the same pixel. Subsequent pairs of these lists define a single row of array **V** corresponding to this direction (figure 2). Executing this algorithm for all predefined directions, the complete array can be filled up.

Note that multiple z-buffer steps carry out a *depth-peeling* procedure. Since this happens in the preprocessing step, its performance is not critical. However, if we intend to modify the illumination network during rendering in order to cope with animated volumes, then the performance should be improved. Fortunately, the depth peeling process can also be realized on the GPU as suggested by [Eve01, Hac04].

3.2. Setting the parameters of the illumination network

The illumination network depends on radius R of the bounding sphere, number of discrete directions D, and on resolution of the windows $M \times M$. These parameters are not independent, but must be set appropriately taking into account the density of the medium as well. Since point rendering is used during the projection onto the window of size $2R \times 2R$ and of resolution $M \times M$, the projected area of a particle is implicitly set to $A = 4R^2/M^2$. The solid angle in which a particle at point $\vec{x}$ is seen from another particle at $\vec{y}$ is $\Delta\omega = A/|\vec{x}-\vec{y}|^2$. Approximating the maximum distance by the expected free run length $1/\tau$, we get $\Delta\omega \geq A\tau^2$.

If we do not want to miss particle interactions due to the insufficient number of sample directions, solid angle $\Delta\omega$ should not be smaller than the solid angle assigned to a single directional sample, which is $4\pi/D$. Substituting the implicit projected area of a particle, we obtain that the number of sample directions and the resolution of the window should meet $D \cdot (R\tau)^2/\pi \geq M^2$. The number of sample directions is typically 128, factor $R\tau$ is the expected number of collisions while the light travels through half of the volume, which usually ranges in 10–100, thus maximum resolution M is in 60–600, i.e. it is usually quite small.

3.3. Iterating the illumination network

The solution of the global illumination problem requires the iteration of the illumination network. A single step of the iteration evaluates the following formula for each particle $p = 1,\dots,N$ and for each incoming direction $i = 1,\dots,D$:

$$\mathbf{I}[p,i] = (1-\alpha_{\mathbf{V}[p,i]})\cdot \mathbf{I}[\mathbf{V}[p,i],i] + \mathbf{E}[\mathbf{V}[p,i],i] +$$

$$\frac{4\pi\cdot\alpha_{\mathbf{V}[p,i]}\cdot a_{\mathbf{V}[p,i]}}{D}\cdot\sum_{d=1}^{D}\mathbf{I}[\mathbf{V}[p,i],d]\cdot P_{\mathbf{V}[p,i]}(\vec{\omega}'_d,\vec{\omega}_i).$$

Interpreting the two-dimensional arrays of the emission, visibility and illumination maps as textures, the graphics hardware can also be exploited to update the illumination network. The first texture is visibility network $\mathbf{V}$ storing the visible particle in red and the opacity in green channels, the second stores emission array $\mathbf{E}$ in the red, green, and blue channels, and the third texture is the illumination map, which also has red, green and blue channels. Note that in practical cases number of particles N is about a thousand, while number of sample direction D is typically 128, and radiance values are half precision floating point numbers, thus the total size of these textures is quite small ($1024\times128\times8\times2$ bytes $=2$ Mbyte).

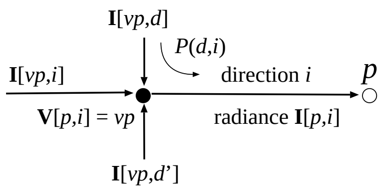

Figure 4: *Notations in the pixel shader code*

In the GPU implementation a single iteration step is the rendering of a viewport sized, textured rectangle, having set the viewpoint resolution to $N\times D$ and the render target to the texture map representing the updated illumination network. A pixel corresponds to a single particle and single direction, which are also identified by input variable `texcoord`. The pixel shader obtains the visibility network from texture `Vmap`, the emission array from texture `Emap`, and the illumination map from texture `Imap`. Function `P` is responsible for the phase function evaluation, which is implemented by a texture lookup of prepared values allowing other phase functions to be also easily incorporated [REK*04]. In this simple implementation we assume that opacity `alpha` is precomputed and stored in the visibility texture, but albedo `alb` are constant for all particles. Should it not be the case, the albedo could also be looked up in a texture.

When no other particle is seen in the input direction, then the incoming illumination is taken from the sky radiance (`sky`). In this way not only a single sky color, but sky illumination textures can also be used [PSS99, REK*04].

For particle `p` and direction `i`, the pixel shader finds opacity `alpha` and visible particle `vp` in direction `i`, takes its emission or direct illumination `Evp`, and computes radiance `Ip` as the sum of the direct illumination and the reflected radiance values for its input directions `d = 1...D` (figure 4):

```
float  p  = texcoord.x; // particle
float  i = texcoord.y;  // input direction
float  vp = tex2d(Vmap, float2(p,i)).r;
if (vp >= 0) { // another particle is seen
   float alpha = tex2d(Vmap, float2(p,i)).g;
   float3 Evp = tex2d(Emap, float2(vp,i)).rgb;
   float3 Ivp = tex2d(Imap, float2(vp,i));
   float3 Ip = (1 - alpha) * Ivp + Evp;
   for(int d = 0; d < 1; d += 1.0/D) {
      Ivp = tex2d(Imap, float2(vp, d));
      float3 BRDF = alb * alpha * P(d,i);
      Ip +=  BRDF * Ivp * 4 * PI / D;
   }
   return Ip;
} else return sky; // no particle is seen
```

The illumination network provides a view independent radiance representation. When the final image is needed, we can use a traditional participating media rendering method, which sorts the particles according to their distance from the camera, splats them, and adds their contributions with alpha blending.

When the outgoing reflected radiance of a particle is needed, we compute the reflection from the sampled incoming directions to the viewing direction. Finally the sum of particle emission and direct illumination of the external lights is interpolated from the sample directions, and is added to the reflected radiance.

4. Results

The proposed algorithm has been implemented in OpenGL/Cg environment and run on an NV6800GT graphics card. We compute one iteration in each frame, and when the light sources move, we take the solution of the previous light position as the initial value of the iteration, which results in fast convergence.

The results are shown in figures 5, 6, and 7. Where it is not explicitly stated, the cloud model consists of 1024 particles, and 128 discrete directions are sampled. With these settings the typical rendering speed is about 26 frames per second, and is almost independent of the number of light sources and of the existence of sky illumination. The albedo is 0.9, and the expected number of photon–particle collisions ($2R\tau$) is 20, and material parameter g is 0. Figures 5 shows how two

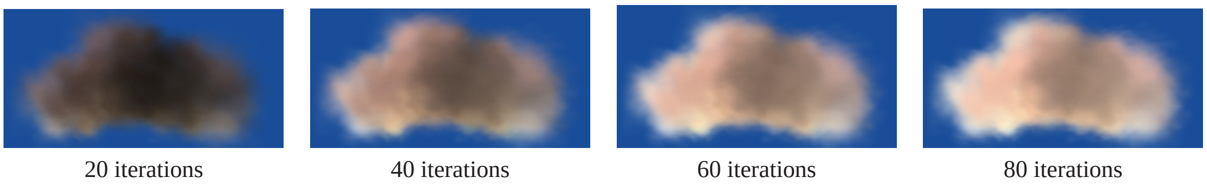

Figure 6: *A cloud illuminated by two directional lights rendered with different iteration steps*

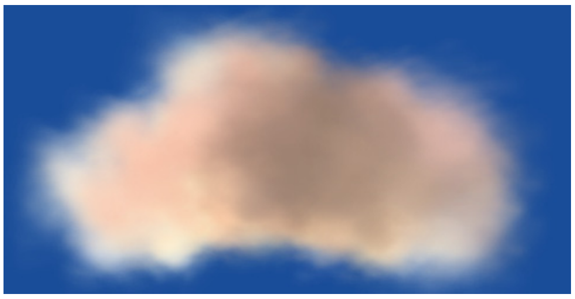

Figure 5: *Cloud illuminated by two dynamic directional light sources (the first is left-up, the second is bottom-right) and sky illumination, and rendered by an animated camera at 26 FPS.*

external light sources illuminate the cloud. In figure 6 we can follow the evolution of the image of the same cloud after different iteration steps, where we can observe the speed of convergence. Figure 7 describes how the number of sample directions and the number of particles affect the image quality and the rendering speed. Note that using 128 directions and 512 particles we can obtain believable clouds at interactive frame rates, and the method is not too sensitive to the number of discrete directions. Changing the number of particles, however, has more significant impact on the image.

5. Conclusions

This paper presented a global illumination algorithm for participating media, which works with a prepared visibility network, and maintains a similar illumination network. These networks are two-dimensional arrays that can be stored as textures and managed by the graphics hardware. The resulting algorithm can render multiple scattering effects at high frame rates.

The current implementation assumes that the volume is static. To cope with evolving volumes, such as clouds in wind, fire, or smoke, we plan to gradually update the illumination network, re-evaluating the visibility just in a single direction at a time, and thus amortizing the cost of the building the data structure during animation. We also plan to extend the method for hierarchical particle systems to handle complex phenomena.

6. Acknowledgement

This work has been supported by OTKA (T042735), GameTools FP6 (IST-2-004363) project, by TIC 2001-2416-C03-01, and by the Spanish-Hungarian Fund (E-26/04).

References

[BF89] BUCKALEW C., FUSSELL D.: Illumination networks: Fast realistic rendering with general reflectance functions. *SIGGRAPH '89 Proceedings 23*, 3 (July 1989), 89–98. 1

[BSH02] BEKAERT P., SBERT M., HALTON J.: Accelerating path tracing by re-using paths. In *Proceedings of Workshop on Rendering* (2002), pp. 125–134. 1

[CS92] CORNETTE W., SHANKS J.: Physical reasonable analytic expression for single-scattering phase function. *Applied Optics 31*, 16 (1992), 31–52. 2

[DMK00] DACHILLE F., MUELLER K., KAUFMAN A.: Volumetric global illumination and reconstruction via energy backprojection. In *Symposium on Volume Rendering* (2000). 1

[Eve01] EVERITT C.: *Interactive order-independent transparency.* Tech. rep., NVIDIA Corporation, 2001. 3

[GRWS04] GEIST R., RASCHE K., WESTALL J., SCHALKOFF R.: Lattice-boltzmann lighting. In *Eurographics Symposium on Rendering* (2004). 2

[Hac04] HACHISUKA T.: Final gathering on GPU. In *ACM Workshop on General Purpose Computing on Graphics Processors* (2004). 3

[HDKS00] HEIDRICH W., DAUBERT K., KAUTZ J., SEIDEL H.-P.: Illuminating micro geometry based on precomputed visibility. In *SIGGRAPH 2000 Proceedings* (2000), pp. 455–464. 1

[HG40] HENYEY G., GREENSTEIN J.: Diffuse radiation in the galaxy. *Astrophysical Journal 88* (1940), 70–73. 2

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud

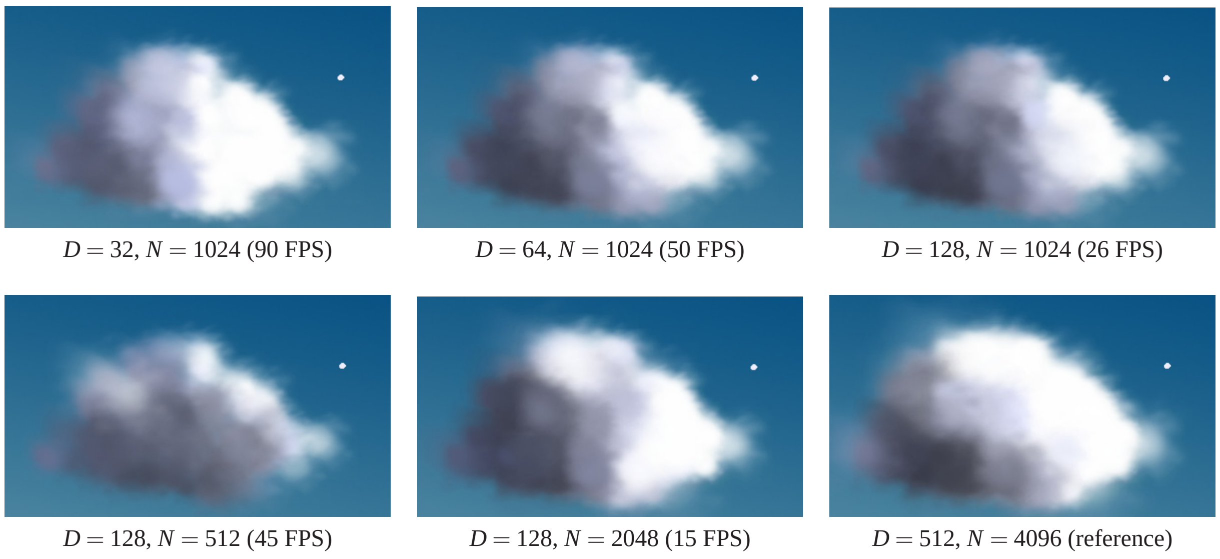
$D = 32$, $N = 1024$ (90 FPS)
$D = 64$, $N = 1024$ (50 FPS)
$D = 128$, $N = 1024$ (26 FPS)
$D = 128$, $N = 512$ (45 FPS)
$D = 128$, $N = 2048$ (15 FPS)
$D = 512$, $N = 4096$ (reference)

Figure 7: *Effect of number of discrete directions D and number of particles N on the image quality and on rendering speed. The light source is in the direction of the white point in the upper right part of the image.*

rendering. *Computer Graphics Forum 20*, 3 (2001). 2

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH '98 Proceedings* (1998), pp. 311–320. 1

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97 Proceedings* (1997), pp. 49–55. 1

[LBC95] LANGUENOU E., BOUATOUCH K., CHELLE M.: Global illumination in presence of participating media with general properties. In *Eurographics Workshop on Rendering* (1995), pp. 69–85. 1

[LW96] LAFORTUNE E. P., WILLEMS Y. D.: Rendering participating media with bidirectional path tracing. In *Rendering Techniques '96* (1996), pp. 91–100. 1

[Max94] MAX N. L.: Efficient light propagation for multiple anisotropic volume scattering. In *Eurographics Workshop on Rendering* (1994), pp. 87–104. 1

[NDN96] NISHITA T., DOBASHI Y., NAKAMAE E.: Displaying of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH '96 Proceedings* (1996), pp. 379–386. 1

[Neu95] NEUMANN L.: Monte Carlo radiosity. *Computing 55* (1995), 23–42. 1

[PKK00] PAULY M., KOLLIG T., KELLER A.: Metropolis light transport for participating media. In *Rendering Techniques* (2000), pp. 11–22. 1

[Pre65] PREISENDORFER R. W.: *Radiative Transfer on Discrete Spaces*. Pergamon Press, 1965. 1

[PSS99] PREETHAM A., SHIRLEY P., SMITS B.: A practical analytic model for daylight. In *SIGGRAPH '99 Proccedings* (1999), pp. 91–100. 4

[REK*04] RILEY K., EBERT D., KRAUS M., TESSENDORF J., HANSEN C.: Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering* (2004), pp. 374–386. 2, 4

[Sbe96] SBERT M.: *The Use of Global Directions to Compute Radiosity*. PhD thesis, Catalan Technical University, Barcelona, 1996. 1

[SK99] SZIRMAY-KALOS L.: Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum 18*, 3 (1999), 233–244. 1

[SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH 2002 Proceedings* (2002), pp. 527–536. 1

[SMKY04] SLATER M., MORTENSEN J., KHANNA P., YU I.: A virtual light field approach to global illumination. In *Computer Graphics International* (2004), pp. 102–109. 1

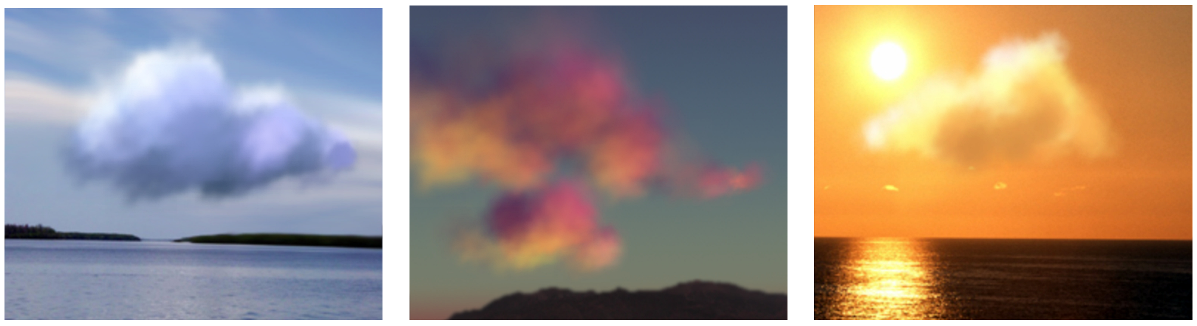

Figure 8: *Globally illuminated clouds of 512 particles rendered with 128 directions at 45 FPS.*