Interactive multiple anisotropic scattering in clouds

Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, Cyril Crassin

Interactive multiple anisotropic scattering in clouds

Antoine Bouthors[1] Fabrice Neyret[1] Nelson Max[2] Eric Bruneton[1] Cyril Crassin[1]
[1]EVASION - LJK / Grenoble Universités - INRIA [2]UC Davis

Abstract

We propose an algorithm for the real time realistic simulation of multiple anisotropic scattering of light in a volume. Contrary to previous real-time methods we account for all kinds of light paths through the medium and preserve their anisotropic behavior.

Our approach consists of estimating the energy transport from the illuminated cloud surface to the rendered cloud pixel for each separate order of multiple scattering. We represent the distribution of light paths reaching a given viewed cloud pixel with the mean and standard deviation of their entry points on the lit surface, which we call the collector area. At rendering time for each pixel we determine the collector area on the lit cloud surface for different sets of scattering orders, then we infer the associated light transport. The fast computation of the collector area and light transport is possible thanks to a preliminary analysis of multiple scattering in plane-parallel slabs and does not require slicing or marching through the volume.

Rendering is done efficiently in a shader on the GPU, relying on a cloud surface mesh augmented with a Hypertexture to enrich the shape and silhouette. We demonstrate our model with the interactive rendering of detailed animated cumulus and cloudy sky at 2-10 frames per second.

1 Introduction

The realistic rendering of clouds is still a challenging problem. Multiple anisotropic scattering must be simulated in a volume, the lack of absorption makes the convergence very slow, and detailed clouds require high resolution volumes. In the scope of real-time rendering this is even worse: volume radiosity or Monte Carlo methods cannot converge in real-time, and volume rendering cannot be achieved with enough resolution for sharp clouds such as cumulus. Precomputations forbid the animation of clouds and light source, and most real-time models do not account for view-dependent effects or miss important visual features such as backscattering.

To solve these problems, our approach represents clouds as surface-bounded volumes and optimizes the calculation of light transport inside the cloud from the illuminated cloud *surface* to the rendered cloud pixels. For this, we study and characterize the light transport for each order of scattering.

Our contributions are:

- a new model characterizing the light transport (*i.e.,* amount of transmitted energy) between the flat surface of a slab and a point $\mathbf{p}$ anywhere in the slab at any given order of scattering (see Figure 1(a)) ;
- a new model characterizing the distribution on a slab surface of the entry points of the light rays reaching any given point $\mathbf{p}$ in the slab for any given order of scattering. We call this entry area the *collector area* (see Figure 1(a)) ;
- an iterative algorithm to determine this collector area on an arbitrary cloud surface for any given rendered point $\mathbf{p}$, and to compute the associated light transport (see Figure 1(b)) ;
- An efficient representation of detailed cloud shape using a Hypertexture [Perlin and Hoffert 1989] on a surface mesh (see Figure 1(c)) ;
- A GPU implementation of these contributions, resulting in highly detailed rendering of animatable clouds interactively with all-orders multiple Mie scattering.

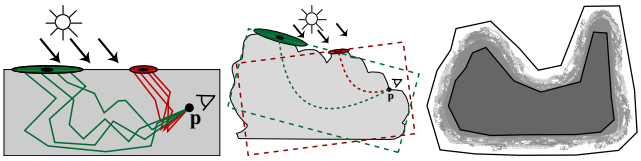

Figure 1: *Overview of our contributions.* (a): *In an preliminary analysis, we characterize light transport in a slab via a collector area representing location of incoming light for each scattering order.* (b): *We use this characterization to find collectors on an arbitrary cloud shape and compute the light transport.* (c): *Our cloud model is represented by a Hypertexture, with procedural details on the boundary and a homogeneous core.*

Contrary to most previous real-time models our multiple scattering simulation:

- does consider the high orders of scattering which are responsible for diffusion and backscatter, as well as the low orders of

scattering which are responsible for the glory, the silver lining on the silhouette, and the appearance of thin parts;
- uses the physically based strongly anisotropic Mie phase function (not Rayleigh, Gaussian or isotropic) ;
- does not need to walk through the volume ;
- does not rely on any shape-dependent precomputation.

Representing the cloud shape with a Hypertexture on a surface mesh allows us to efficiently render inhomogeneous boundaries as well as sharp detailed clouds such as cumulus, contrary to methods based on sliced volumes. We can thus render sharp, fluffy or complex wispy clouds at little cost.

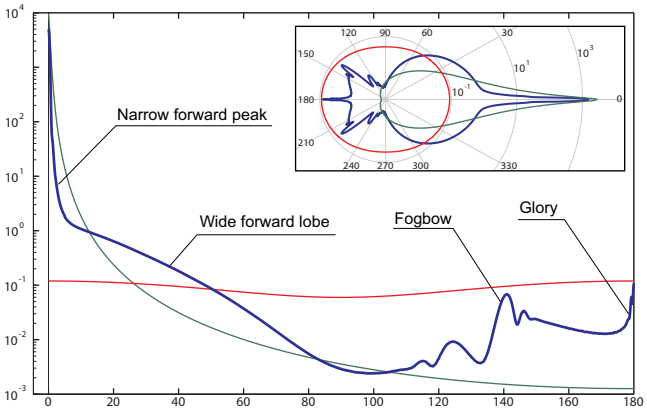

Figure 2: *Log plots (inset: polar log plots) of commonly used phase functions.* Red: *Rayleigh.* Green: *Henyey-Greenstein with* $g = .99$. Blue: *Mie.*

2 Cloud physics

2.1 Density and size of droplets in clouds

Real convective clouds are not *blurry* on boundaries, they are usually sharp or *wispy* (see Figures 5(c), 5(d)). Collapsing cloud turrets or weak clouds may have a larger wispy layer. These inhomogeneities in the cloud liquid water content (*i.e.,* mass density) are a strong visual feature of clouds. Inside the cloud, this density can also vary – especially near rain condition – due to coalescence of droplets.

The size of droplets is characterized at any location by a *droplet size distribution* (DSD), which is generally modeled in literature by a lognormal or a modified Gamma function [Levin 1958]. The optical properties of a cloud depend highly on the droplet size. Accounting for the DSD radically changes the resulting phase function, thus the visual appearance.

Still, these data within a cloud is not often available and its physics (*e.g.,* the coalescence mechanism or the evolution of the DSD) is not fully understood. Since computer graphics applications require primarily visual plausibility, approximations of these values are common. As an example, density variations are visually more important on the clouds boundaries –where they are directly visible– than in the cloud core –where they only influence the appearance indirectly, through high-order multiple scattering. Usually, the DSD over a cloud (if using a DSD at all) is assumed constant.

2.2 Phase function

The phase function of a cloud droplet is given by the Mie theory. Common approximations are Gaussian or Henyey-Greenstein functions. As seen in Figure 2 the Mie function combines a strong narrow forward peak (51% of energy), a wide forward lobe (48% of energy), and a complex backward lobe with peaks. These three features all yield very specific and visible effects at the scale of the

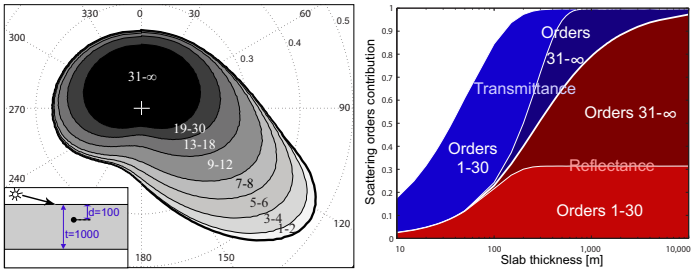

Figure 3: *Some results of our light transport analysis.* **Left:** *BSDF for a point of view at a depth* $d = 100m$ *inside a cloud slab of thickness* $t = 1000m$*, with an incident illumination angle* $\phi_L = 75°$*. Areas represent the contribution of different orders of scattering. Even high orders (up to* 30 *scattering events) show an anisotropic behavior, while orders* > 30 *show an isotropic behavior.* **Right:** *Contribution of different orders of scattering for the reflectance (in red) and transmittance (in blue) of a slab of varying thickness. Isotropic orders (31-*∞*) begin to appear at thicknesses* $> 100m$*. Anisotropic orders (1-30) play a role up to* $1000m$ *in transmittance and contribute to* $30\% - 100\%$ *of the reflectance.*

cloud. The absence of backward peaks means no glory or fogbow. No narrow forward peak means huge underestimation of global transmittance and anisotropy. Gaussian and Henyey-Greenstein functions do encode a lobe and ease calculation but they give far from accurate visual effects. These approximations miss cloud features such as the glory and fogbow, and blur out the narrow forward scattering peak. Rayleigh scattering is even less appropriate since it is symmetrical (50% backward) and it physically corresponds to molecular scattering (giving its blue color to the sky), as opposed to scattering due to droplets.

In the visible spectrum, water cloud albedo can be considered as 1 since there is no absorption (all light is either reflected or transmitted). Note that some atmospheric phenomena commonly credited to clouds are actually caused by other elements (*e.g.,* rainbows are caused by rain, and sundogs by ice crystals in the atmosphere).

2.3 Anisotropy

Multiple scattering is strong in clouds and often shows anisotropic behavior. Clouds span hundreds to thousands of meters and the mean free path of a light ray is about 20m. As a result, most rays will be scattered multiple times before exiting the cloud. Because of the highly anisotropic nature of the Mie phase function (99% of the light is scattered in the forward direction), even multiple scattering can be anisotropic. From our analysis of light transport in a slab, we estimate that the light behavior is isotropic only after about 30 scattering events. By isotropic, we mean that multiple scattering behaves as if the phase function of the medium was isotropic, not that the resulting Bidirectional Scattering Distribution Function (BSDF) is isotropic.

3 Previous Work

Simulating multiple anisotropic Mie scattering through Monte-Carlo integration is not practical in real-time. Previous optimization approaches rely on various simplifications, which we discuss below.

3.1 Phase function

Since the Mie phase function is complex and expensive to compute, it is often approximated using other functions such as Henyey-Greenstein [Max 1994], Gaussian [Premoze et al. 2004] or even Rayleigh [Harris and Lastra 2001] (see comments in Section 2.2). [Riley et al. 2004; Bouthors et al. 2006] precompute the Mie

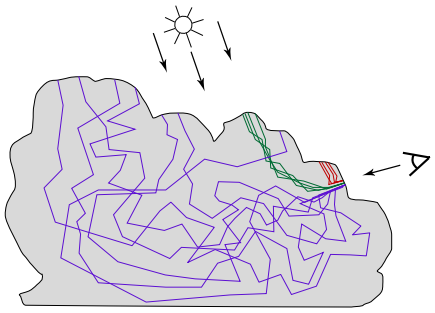

Figure 4: *Various kinds of light paths in participating media, brought by different orders of scattering. Low orders of scattering bring short, steep-turning paths (red). Higher orders bring long, slow-turning paths (green). Highest orders bring complex, spread, diffusive paths (blue). The* Most Probable Paths *approach [Premože et al. 2003] reproduces the green ones, but underestimates red and blue ones.*

phase function for a given DSD, which reproduces real-life features (glory, fogbow, etc.). We use the same approach for our phase function.

3.2 Light transport

[Kajiya and von Herzen 1984] and [Blinn 1982] consider either low albedo or low density: only single scattering is considered. This neglects all multiple scattering effects. Assuming the transport is mostly forward allows for real-time single-pass algorithms such as slice accumulation [Dobashi et al. 2000; Harris and Lastra 2001; Riley et al. 2004] but this neglects some of the multiple scattering effects such as backscattering. Computing multiple scattering [Nishita et al. 1996] for only the lower orders only has the same effect. The diffusion approximation for multiple scattering [Stam 1995; Jensen et al. 2001] allows efficient computations of high orders but neglects anisotropy in multiple scattering.

[Premože et al. 2003] introduced the idea of *Most Probable Paths* (MPP) in participating media. The root idea is that most of the photons arriving at one point in one direction roughly followed the same path. Thus, integrating light transport only along this path is sufficient to account for most of the energy. In addition, [Premoze et al. 2004] speed up this technique and account for the spatial spreading of light around this mean path through an analytical formulation. This approach has been brought to real-time [Hegeman et al. 2005] using graphics hardware for volume slicing in a manner similar to [Harris and Lastra 2001; Riley et al. 2004]. These methods based on two slicing passes (accumulating the flux from the light source in the slices, then from the slices to the eye) restrict the variety of light paths accounted for.

As mentioned in [Premože et al. 2003; Hegeman et al. 2005] the main limitation of the MPP approach is that the paths it computes are mainly of high order, with a small angle change per scattering event. Thus, paths of low order as well as diffusive paths are underestimated. These paths contribute to most of the lighting in thin cloud parts and in the backscattering, and thus should not be neglected (see Figure 4).

We address these limitations by treating light paths of all orders. We also propose a new, faster light transport computation approach that does not rely on slicing or marching through a volume. Our approach is also inspired by the radiative transfer studies on basic shapes such as slabs [Chandrasekhar 1960; Krim and El Wakil 1986; Mobley 1989; Max et al. 1997]. We use hardware-friendly representations such as depth maps [Dachsbacher and Stamminger 2003] to implement our rendering algorithm on the GPU.

(a) [Harris and Lastra 2001] (b) [Riley et al. 2004]

(c) Photograph (d) Photograph

(e) [Schpok et al. 2003] (f) [Gardner 1985]

Figure 5: Top: *real-time **(a)** and interactive **(b)** CG cumulus clouds using billboard or slices.* Middle: *wispy **(c)** and sharp **(d)** real cumulus clouds.* Bottom: *CG clouds using 3D noise **(e)** and surfaces with procedural details **(f)**. Adding procedural details gives a less blurry and more contrasted appearance, increasing realism.*

3.3 Cloud densities representation

Since real cloud density data is difficult to measure or simulate, early work used procedural models [Gardner 1985; Dobashi et al. 2000]. Recent approaches have been using fluid simulation techniques [Harris and Lastra 2001] or atmospheric data [Trembilski and Broßler 2002; Riley et al. 2004]. However, fluid simulation can only be computed at coarse level for real-time applications, and atmospheric data have low resolution. As a result, adding high frequency details is necessary to avoid a blurry appearance (see Figure 5). [Ebert 1997] combines Perlin solid noise [Perlin 1985] and implicit surfaces. [Schpok et al. 2003] advect a noise texture [Neyret 2003], but do not account for the noise in the light transport computation, which gives them a uniform appearance (see Figure 5(e)). These ideas inspired us to combine two representations –meshes and 3D textures– to model the clouds at two different scales.

Various rendering primitives have been considered to render clouds. Since clouds are a 3D distribution of droplets, volume grids have often been used to describe them and volume rendering techniques to render them. Methods using billboards or volume slices [Harris and Lastra 2001; Premoze et al. 2004; Riley et al. 2004; Dobashi et al. 2000] result in a lot of overdraw[1], which is computationally expensive. While textured slices are an efficient way of rendering gaseous phenomena, they are not the best option for dense clouds,

[1] *i.e.,* a given pixel is rasterized many time by different rendered primitives.

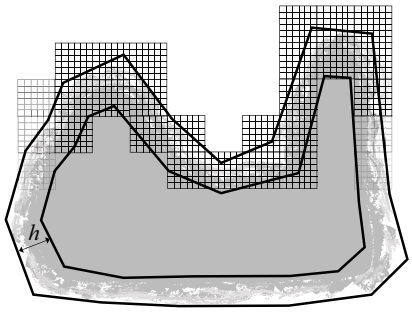

Figure 6: *Our cloud representation. A mesh is used to describe the outer cloud boundaries at low resolution. A procedural volumetric hypertexture adds details under the boundary up to a certain depth h inside the cloud. The core is considered homogeneous.*

where most pixels of the back slices are hidden by those in front and therefore wasteful to render. Moreover, 3D texture memory limits the resolution of such models, resulting in blurry silhouettes and a lack of details (see Figures 5(a), 5(b)).

Since cumulus clouds are dense and often have a sharp interface, they also have been represented as surface-bounded volumes such as sets of ellipsoids [Gardner 1985; Elinas and Stürzlinger 2000] or meshes [Trembilski and Broßler 2002]. To avoid the "hard" appearance of polygonal surfaces, they use a procedural shader simulating the detailed silhouette and giving a volumetric impression (see Figure 5(f)). Light transport through the volume is not simulated. [Bouthors et al. 2006] do simulate light transport inside a mesh, but they consider no enrichment at all on silhouettes which thus appear polygonal and opaque.

We take advantage of both volumes and surfaces by representing the high-scale cloud boundary with a mesh and the high-frequency density variations at cloud borders with a Hypertexture [Perlin and Hoffert 1989]. The rest of the cloud interior is considered homogeneous, so only a thin layer of voxels is necessary (see Figure 6).

4 Overview of our method

Our shading approach is based on the analysis of light transport for each scattering order. Observing that paths of different orders have different anisotropy and spreading behaviors, we treat them separately:

- order 1 (single scattering), which is the most anisotropic and depends on the finest details, is computed using an analytical form of the scattering equation (see Section 6.2) ;
- orders 2-∞ (multiple scattering) are computed in 8 separate sets (2, 3-4, 5-6, 7-8, 9-12, 13-18, 19-30, 31-∞) using our collector-based algorithm (see Section 6.1) ;
- opacity is computed by integrating the extinction function through the cloud volume (see Section 6.3).

Our collector-based approach follows and extends the idea of *Most Probable Paths* (MPP) [Premože et al. 2003]. We consider one most probable path and spreading per set of scattering orders. More specifically, we consider a *collector area*, which is the piece of surface through which enters 95% of the light that reaches the current rendered pixel in the view direction (see Figure 7(b)). This collector is defined by its center $\mathbf{c}$ on the cloud surface and width σ. For a given pixel, for each set of scattering orders, we look for this collector, and compute the corresponding light transport.

To find this collector area on the lit cloud surface, we rely on an algorithm that iteratively matches the cloud surface with the most probable collector area. This algorithm is described in Section 6.1.

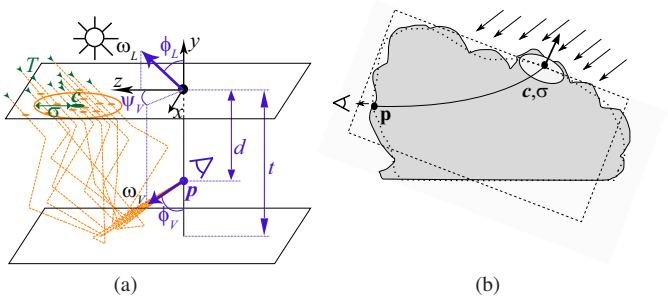

Figure 7: **(a)**: *Setup and notation for the canonical light transport T through a collector area $(\mathbf{c}, \sigma)$ between a lit slab surface and a point $\mathbf{p}$ in the slab, depending on the input parameters $(\phi_V, \psi_V, \phi_L, d, t)$ for each set of scattering orders. Note that ψ_L is always zero (the reference frame is aligned with the light direction).* **(b)**: *Light transport in the cloud between its lit surface and a point $\mathbf{p}$. For each set of scattering orders the incoming light at $\mathbf{p}$ is assumed to come from a collector area. We characterize this transport by looking for a fitting slab, then relying on the canonical transport function.*

By definition, the role of the cloud surface outside the collector is negligible, thus we can locally approximate the cloud shape as a plane-parallel slab aligned on the collector in order to simplify the computation of light transport (see Figure 7(b)).

Computing anisotropic light transport even for a simple shape like a slab is still very complex [Chandrasekhar 1960]. To accelerate the light transport computation, we characterize the radiative transfer in a slab through a *canonical transport function*. This is described in Section 5. We obtain it by analyzing numerous Monte-Carlo simulations for different slab parameters (see Appendix A).

We implement our whole rendering approach on the GPU, using depth maps to represent the cloud surface. This is described in Section 7. In Section 8 we introduce our enrichment of the cloud model and especially of its silhouette in a Hypertexture upon the surface. The gathering of all steps within a single shader is synthesized in Section 8.1. Then we present our results and performance in Section 9.

5 Characterizing canonical light transport

5.1 Simulation

In this section we describe our experimental setup in the canonical case of a plane-parallel slab of thickness t (see Figure 7(a)). For each set of scattering orders we compute the light transport T (*i.e.,* the proportion of transmitted energy), the collector center $\mathbf{c}$ and the collector standard deviation σ.

These values $\langle T, \mathbf{c}, \sigma \rangle$ are computed against 5 input parameters: slab thickness t, viewpoint depth d, viewing angles (ϕ_V, ψ_V) and lighting elevation angle ϕ_L, in the reference frame shown in Figure 7(a). We define $\mu_V = \cos(\phi_V)$, $\mu_L = \cos(\phi_L)$, $\vec{\omega}_V$ = view direction, $\vec{\omega}_L$ = light direction, $\cos\theta = \vec{\omega}_V \cdot \vec{\omega}_L$, $\mathbf{p} = (0, -d, 0)$ = viewpoint location. Since we want to characterize light transport up to any point $\mathbf{p}$ within the volume, we consider values of the viewpoint depth d within the slab ($0 \leq d \leq t$). Viewpoints outside the slab correspond to $d = 0$ or $d = t$.

We ran numerous Monte-Carlo simulation of light paths for various values of these 5 parameters (see Appendix A for the details). We store the results $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ in a 5D table for each set of scattering orders. We call $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ the *canonical transport function*.

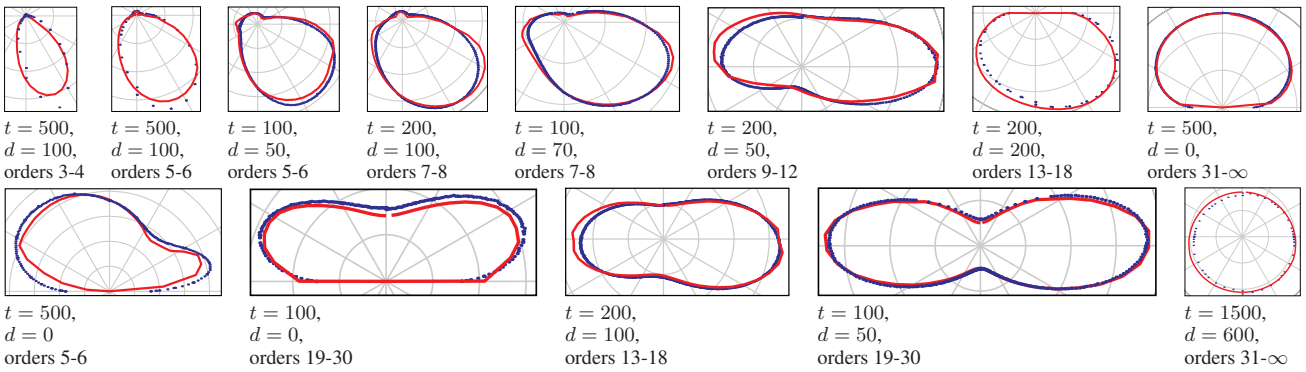

Figure 8: *Some resulting BSDF of our light transport analysis. T is plotted against ϕ_V (ψ_V fixed at $0°$) for various input parameters ($\phi_L = 25°$).* Blue: *Monte-Carlo simulations.* Red: *our fitting.*

5.2 Data compression

These 5D tables (1 per set of scattering orders) encode the *canonical transport function*, that is, the macroscopic light behavior in a slab of cloud depending on the input parameters $(\phi_V, \psi_V, \phi_L, d, t)$. Such a set of 5D tables cannot be stored in GPU memory. We compress them by approximating the computed results with empirical functions. Appendix A describes our experimental and fitting setup.

• For the light transport T, this fitting yields

$$T = P \cdot A \cdot X \cdot \mu_L \frac{\log(d+D)}{\log(B+D)} e^{-\frac{(d-B)^2}{2C^2}}$$

with $A = \mathbf{A}(t, \mu_V)$, $B = \mathbf{B_1}(t, \mu_V) - \mathbf{B_2}(t, \mu_L)$, $C = \mathbf{C}(t, \mu_V)$, $D = \mathbf{D}(t, \mu_V)$, $X = \mathbf{X}(t, \mu_L)$, $P = \mathbf{P}(\theta)$, *i.e.,* a 5D table for T reduces to an analytical function and seven 2D tables. $\mathbf{P}$ encodes the anisotropy of the result and equals 1 if the light behavior is isotropic (orders 31-∞). $\mathbf{X}$ modulates the result according to the lighting angle and is inspired from Chandrasekhar's X-function [Chandrasekhar 1960]. $\mathbf{A}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{C}$, $\mathbf{D}$ are the parameters of a "skewed" Gaussian function encoding the light behavior according to the depth of the viewpoint. Figure 8 shows the result of this fitting.

• For the collector center $\mathbf{c}$, our compression results in $\mathbf{c} = (c_x, 0, c_z)$ with

$$\begin{aligned} c_x &= A_x \log(1 + E \cdot d) + B_x \\ c_z &= A_z \log(1 + E \cdot d) + B_z \\ A_x &= F \sin\psi_V \sin(G \cdot \phi_L) \\ B_x &= H \sin\psi_V \sin(\phi_L) \\ A_z &= I + J[\cos\psi_V \sin(K \cdot \phi_L) + L\phi_L] \\ B_z &= M + N \cos\psi_V \end{aligned}$$

with $E = \mathbf{E}(\mu_V)$, $F = \mathbf{F}(\mu_V)$, $G = \mathbf{G}(\mu_V)$, $H = \mathbf{H}(\mu_V)$, $I = \mathbf{I}(\mu_V)$, $J = \mathbf{J}(\mu_V)$, $K = \mathbf{K}(\mu_V)$, $L = \mathbf{L}(\mu_V)$, $M = \mathbf{M}(\mu_V)$, $N = \mathbf{N}(\mu_V, \mu_L)$.

• Our compression of the collector size σ gives

$$\sigma = \mathbf{O} + \mathbf{Q} \cdot t \log(1 + \mathbf{R} \cdot d) + \mathbf{S} \log(1 + \mathbf{T} \cdot t)$$

where $\mathbf{O}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$, $\mathbf{T}$ are constant for a given set of scattering orders.

6 Estimating light transport in clouds

6.1 Multiple scattering

Considering a given cloud pixel to be rendered (corresponding to a location $\mathbf{p}$ in the cloud), for each scattering order we look for the collector area $(\hat{\mathbf{c}}, \hat{\sigma})$, *i.e.,* the origin of dispersed light paths on the lit cloud surface which ended at $\mathbf{p}$ in eye direction (see Figure 7(b)). Since a cloud is a volume of inhomogeneous densities, defining its surface is difficult. We choose to define it as the boundary where the density ρ crosses a user-defined threshold value ρ_0. As explained in the overview, we assume that the cloud behaves locally like a slab tangent to the surface at the collector location. Here, surface orientation is a scale-dependent notion. Since we are interested in the surface of incoming dispersed light paths, our local surface orientation is obtained through filtering the cloud surface according to the dispersion standard deviation σ (dashed cloud shape in Figures 7(b) and 9). Sections 7 and 8 give more details on this filtering.

To find this collector location, we iterate as shown on Figure 9. We start at step 1 with a first collector of size σ_0 at location $\mathbf{c}_0$ which we project in step 2 along the light direction on the cloud surface at $\mathbf{c}'_0$. The corresponding slab is tangent to the surface (filtered according to σ_0) at $\mathbf{c}'_0$. View and light parameters $(\phi_V, \psi_V, \phi_L, d, t)$ according to this slab are obtained by simple geometric transformations. In step 3, the canonical transport function provides us with the collector location $\mathbf{c}_1(\phi_V, \psi_V, \phi_L, d, t)$ and size $\sigma_1(\phi_V, \psi_V, \phi_L, d, t)$ corresponding to such a configuration, which is likely to be different from $\mathbf{c}'_0$. We project $\mathbf{c}_1$ on the cloud surface filtered by σ_1 at $\mathbf{c}'_1$ in step 4, and we iterate up to convergence, *i.e.,* $\mathbf{c}_n \simeq \mathbf{c}'_n$. We then take $(\hat{\mathbf{c}}, \hat{\sigma}) = (\mathbf{c}_n, \sigma_n)$. Once we found the collector, we can obtain the corresponding light transport T (*i.e.,* the amount of energy transmitted from this collector to the eye) with the same input parameters. We run this algorithm for each set of scattering orders on the GPU (see Section 7).

This iterative algorithm is in fact a fixed-point method applied on $\mathbf{c}$, *i.e.,* we are looking for value x that satisfies $f(x) = x$, where x here is our collector parameters $(\mathbf{c}, \sigma)$ and f represents our canonical transport function. Since the search space of this collector is restricted to the cloud lit surface, this algorithm cannot diverge. However, like any basic fixed point method, it can reach a state where it loops between two values without converging (*i.e.,* $\mathbf{c}_i \neq \mathbf{c}_{i-1}$ and $\mathbf{c}_i = \mathbf{c}_{i-2}$). It can also take too large steps and miss the solution. To avoid these cases, we limit the size of each step (we constraint $\mathbf{c}_i$ such that $\|\mathbf{c}_i - \mathbf{c}_{i-1}\| < \sigma_i$). We start the algorithm with an initial position $\mathbf{c}_0 = \mathbf{p}$. We use a large initial size σ_0. Indeed, if the initial collector spans the whole cloud, the projected result at step 2 will be a first-order approximation of the whole lit cloud surface, which

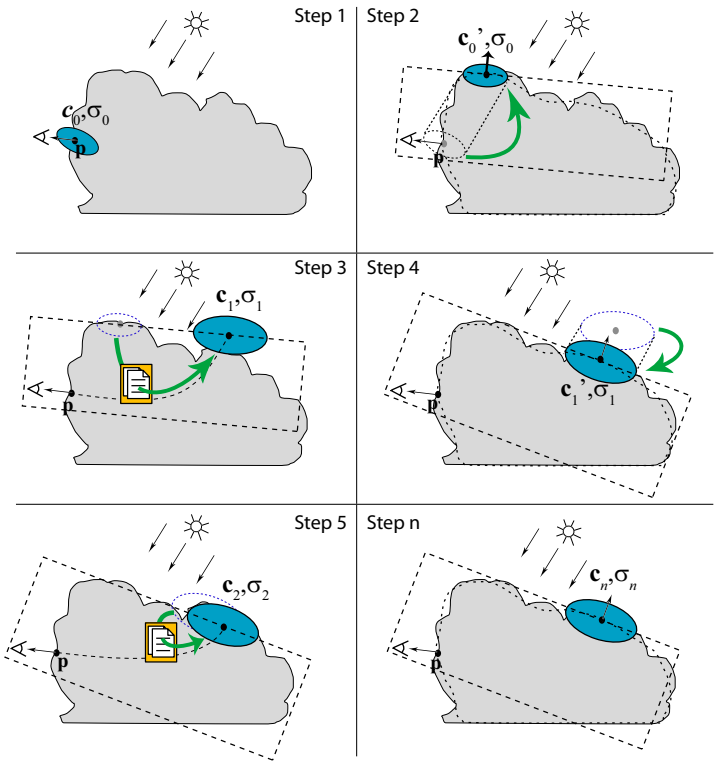

Figure 9: *Summary of our iterative algorithm to locate the collector (blue area). Solid cloud shape: unfiltered cloud surface. Dashed cloud shape: surface filtered according to σ. Dashed box: slab matching the filtered surface at collector position* **c**. *Orange boxes represent the use of our canonical transport function.*

is a good starting point for our algorithm in terms of convergence speed. Subsequent steps will then "refine" the lit cloud surface at the most probable collector location.

6.2 Single Scattering

For the first scattering order, which constitutes a degenerate case for our collector-based algorithm, we simply integrate analytically the Mie single scattering along the eye direction. This allows us to account for the inhomogeneous densities and the fine details along the view direction.

This integration is done by considering piecewise linear segments with samples taken on the cloud surface as shown in Figure 10 (*i.e.,* exponentially spaced samples are read along the view direction). For one segment i, the single scattering formula gives

$$\begin{aligned} T_i &= \mathcal{M}ie(\theta) \int_{x_i}^{x_{i+1}} \kappa e^{-\kappa(x+l(x))} dx \\ &= \mathcal{M}ie(\theta)(e^{-\kappa(x_{i+1}+l_{i+1})} - e^{-\kappa(x_i+l_i)}) \end{aligned} \quad (1)$$

where $\mathcal{M}ie$ is the Mie phase function and κ the extinction coefficient defined in Equation 4, Appendix A. Note that since the Mie phase function is wavelength-dependent, we encode and use $\mathcal{M}ie$ as an RGB function. In multiple scattering, this dependence is blurred out, thus T needs only be a scalar.

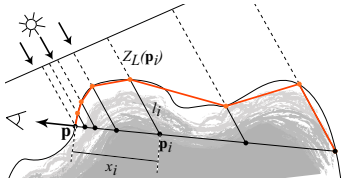

Figure 10: *Piecewise linear integration of the single scattering paths.*

6.3 Opacity

The opacity α is computed by integrating the extinction along the view direction. This results in $\alpha = \int_0^l e^{-\kappa x} dx$. Since the extinction coefficient κ is varying in space, we discretize this integration by sampling segments along the view direction as for single scattering. For each segment i, we read κ in the cloud volume and accumulate the opacity

$$\alpha_i = \int_{x_i}^{x_{i+1}} e^{-\kappa x} dx = \frac{e^{-\kappa x_{i+1}} - e^{-\kappa x_i}}{\kappa}. \quad (2)$$

This is a standard ray marching procedure.

7 Implementation on the GPU

At rendering time, the cloud mesh is rendered using a fragment shader that computes the light transport from the lit surface as explained in Section 6. Since this process is fairly involved we rely heavily on GPU capabilities to make it efficient, especially for computing distances to the filtered cloud surface, which we explain in this section.

The 22 intermediate functions **A** through **X** described in Section 5.2 necessary to compute the canonical transport function are discretized and stored in textures, as well as the $\mathcal{M}ie$ phase function used for the single scattering. Overall this gives a few textures occupying less than 2MB in video memory.

To manage efficiently the surface processing (*e.g.,* filtering, computing distances) on the GPU we rely on depth maps. We create two depth maps $Zmin_L$, $Zmax_L$ and a normal map N_L from the light point of view. This is done at each frame to allow for the animation of the cloud shape and of the light.

A convenient way to approximate the filtering of the surface according to a kernel of size σ is to rely on the MIP-mapping of the Z values in the depth map. To compute the MIP-map correctly (*i.e.,* without taking into account pixels where there is no cloud), we add an alpha channel A and we follow the same scheme as for an alpha-premultiplied texture. In fact the premultiplication is done automatically when setting the default Z value to 0. After the MIP-map pyramid $(Z^0, ..., Z^n)$ is computed, a filtered Z value is obtained by $\frac{Z^i(P)}{A^i(P)}$, as in [Dachsbacher and Stamminger 2003].

Computing the distances d (Section 6.1) and l_i (Equation 1) to the filtered surface in the light direction is done by simply accessing the associated MIP-mapped depth maps. To compute the thickness t, we subtract $Zmax_L - Zmin_L$. The orientation of the filtered surface is obtained by reading the MIP-mapped normal map N_L. Thanks to this representation, all the parameters $(\phi_V, \psi_V, \phi_L, d, t)$ can be computed and the whole algorithm described in Section 6 can be implemented in a fragment shader. For each set of scattering orders the shader first finds the collector $(\hat{\mathbf{c}}, \hat{\sigma})$ using our algorithm described in Section 6.1, then computes the associated intensity T using the compressed canonical transport function described in Section 5.

8 Volumetric enrichment of the cloud model

Surface-based cloud approaches generally rely on a transparency shader to fake the effect of inhomogeneities on the silhouette. Volume-based methods are limited in resolution due to the huge memory requirement, which usually makes the silhouette lack details (see Figure 5). Our approach combines the best of both: we define a volumetric Hypertexture in a layer below the surface in order to have high-resolution 3D effects on the cloud edges with low memory requirements (see Figure 6). Thus, the volume representation handles all the frequencies of cloud details that are not handled by the cloud mesh.

The Hypertexture layer is defined through a distance function $D(\mathbf{p})$ to the surface (which equals 0 on the surface and increases inside the cloud). The procedural details are modulating the density N_0 (Equation 3) and the extinction coefficient κ (Equation 4) with $\rho(\mathbf{p}) = S(D(\mathbf{p}) + noise(\mathbf{p}))$ where S is a sigmoid function and $noise$ is a scalar 3D Perlin noise [Perlin 1985]. Due to the cost of evaluating distance fields, $D(\mathbf{p})$ is computed and stored in a volumetric texture. $noise()$ is evaluated on the fly in the shader. We rely on [Crassin and Neyret 2007] to compute and voxelize quickly the distance field at high resolution using only the minimal necessary memory usage.

This volumetric enrichment is used in three parts of the shader. When computing the cloud opacity (Section 6.3), we perform a ray marching on the GPU [Crassin and Neyret 2007] through this layer to integrate the cloud density in this part of the cloud. When computing single scattering (Section 6.2), the extinction coefficient κ (Equation 4) is also modulated for each segment according to $\rho(\mathbf{p})$. Finally, to account for these details in the multiple scattering computations (Section 6.1), the depth map $Zmin_L$ is modulated: we store in $Zmin_L$ the depth of the first voxel having $\rho > \rho_0$.

Note that for most pixels except on the silhouette and thin parts, strong opacity will stop the ray close to the entry point, so the average marching length is limited.

8.1 Final rendering

The precomputation for the current frame is limited to rebuilding the depth and normal maps. This allows us to change the viewpoint, lighting conditions, and cloud shape.

Terrain and opaque objects are drawn first. Then the cloud mesh is drawn using our pixel shader through deferred shading. This pixel shader:

- iteratively finds the collectors $(\hat{\mathbf{c}}, \hat{\sigma})$ for the 8 sets of scattering orders corresponding to sun illumination (Section 6.1) ;
- computes the light transport T associated to each set (as explained in Section 5), and multiplies each by the lighting conditions (intensity, color and visibility) ;
- computes the analytical single scattering (Section 6.2) ;
- computes the opacity (Section 6.3).

We handle environment illumination like sun illumination. A blue source is added above the cloud and a brown one below the cloud. We use our multiple scattering algorithm with both of these additional sources. Outdoor scenes require accounting for aerial perspective. We rely on the model of [Hoffman and Preetham 2003]. Rendering very bright objects such as clouds also require some tone mapping management. We use a simplified, unblurred version of [Goodnight et al. 2003].

Note that our representation allows us to easily account for points of view inside the clouds, since our canonical transport function accounts for this case.

9 Results

Our tests were conducted on a Pentium 4 at 1.86 GHz with a nVidia 8800 GTS graphics board. All benchmarks were done at resolution 800×600.

We tested our method against different cloud shapes: a cloud slab, an animated stratocumulus layer without procedural noise, and a cumulus cloud with procedural noise. The slab allows us to test our algorithm and validate directly its results against the canonical transport function. It also allows us to see features such as anisotropic reflectance, anisotropic transmittance in thin slabs and isotropic transmittance in thick slabs. The stratocumulus layer allows us to test our method against a shape close to the canonical one and to validate the handling of animations. It is composed of 130K triangles, and the framerate is 10fps. We can see on this example all the wanted cloud features: anisotropic scattering, diffusion, backscattering, glory, fogbow. The cumulus cloud model allows us to test our algorithm on an arbitrary shape. It is composed of 5K triangles with a 512^3 Hypertexture. We obtain a framerate of 2fps on this model. Note that most of the time is spent in evaluating the noise on the fly in the hypertexture. Without the procedural noise, the framerate rises to 10fps. This model also displays all the sought cloud features. 10 iterations of our collector-finding algorithm are sufficient in all cases to reach convergence. Figure 12 and the teaser show the result of our method on various cases.

10 Discussion and Future Work

We address the limitations of other real-time approaches [Harris and Lastra 2001; Riley et al. 2004; Hegeman et al. 2005] by treating accurately light paths of all orders and by using a more detailed cloud shape. We use the Mie phase function for cloud droplets, which provides us with the right anisotropy and enables visual effects such as glory and fogbow. Contrary to [Schpok et al. 2003], we account for the procedural details in the light transport calculation. Although similar details could be attained by other real-time approaches [Harris and Lastra 2001; Riley et al. 2004; Hegeman et al. 2005] by increasing the volumetric resolution of their models (at the expense of speed and within memory limits), our method brings features that are not handled by these approaches such as accurate backscattering and diffusion. We allow for the animation of viewpoint, light direction and cloud shape. In case of cloud animation, we rely on the algorithm described in [Crassin and Neyret 2007] to recompute in real-time the distance field used by our procedural noise.

Like the other methods using a depth-map to solve scattering problems [Dachsbacher and Stamminger 2003], this representation has some issues regarding resolution and non-convex shapes. The precision of this representation is limited by the resolution of the depth map. As a result, full scalability of the method would probably require alternative depth maps techniques [Stamminger and Drettakis 2002; Arvo 2007]. In case of non-convex shapes, light transport is slightly underestimated in shadowed regions (see Figure 11(a)): the algorithm incorrectly assumes that there is matter between the shadowed surfaces and the lit surface. In consequence, light is considered more attenuated than it should be. In practice, this error happens on shadowed parts of the cloud, where environment (*i.e.,* sky and ground) illumination is predominant. As mentioned in [Dachsbacher and Stamminger 2003] who suffer from the same issues, this issue can be solved by using a depth peeling technique to treat non-convex shapes as a collection of convex shapes. Using deep shadow maps [Lokovic and Veach 2000] would also help addressing this issue and increase the accuracy of single scattering.

One limitation of our approach is the assumption that light arriving

to the viewer goes through only one connected collector (*i.e.,* one most probable path) per set of scattering orders. As shown on figure 11(b) it is not always the case. In these configurations, our algorithm accounts for only one collector, thus underestimates the amount of light transmitted. Note that since we look for several collectors (one for each of the 8 sets of scattering orders) per pixel, this error concerns only a fraction of the pixel color. One solution might be to look for several collectors per set of scattering orders with different initial values.

Searching for a collector $\hat{\mathbf{c}}$ corresponds to searching for the entry point $\hat{\mathbf{c}}$ on the lit surface of a local maximum of the light transport T. Our iterative algorithm corresponds to a fixed-point method: we look for the fixed point $f(\mathbf{c}) = \mathbf{c}$ where f represents our canonical transport function. As future work, better techniques from the field of optimization could be used to ensure faster convergence and to handle the case of several maxima of T.

Our light transport computation on GPU using our collector-finding algorithm (*i.e.,* the main contribution of this paper) is fast enough for interactive applications (10fps). The speed of this method can be still be improved. Moreover, the computation of the procedural noise and the GPU ray marching in our implementation is an unoptimized version of [Crassin and Neyret 2007] and is computationally expensive (80% of the rendering cost).

As future work, we would like to take into account higher-scale light effects such as interreflections between clouds and cloud lobes. Also, the method described in [Bouthors et al. 2006] can be applied to this work to compute the interreflections between the clouds and the ground, which have been shown to be of visual importance.

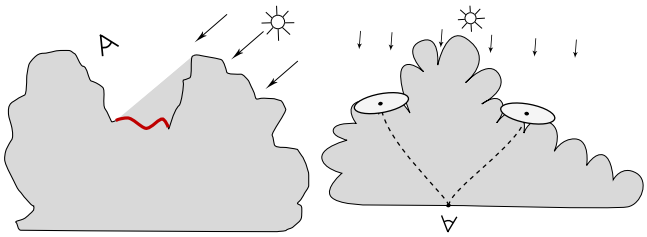

Figure 11: ***(a):*** *Limitation of using depth maps. The radiance of the red surface is slightly underestimated. However, this error is of low visual importance.* ***(b):*** *Limitation of searching for only one collector per set of scattering orders. In this example two collectors of equal importance exist, but our method will find only one.*

11 Conclusion

We have proposed a study of light transport yielding a new formulation for the macroscopic behavior of light in participating media that is suited for computer graphics. We have proposed a new way of computing radiative transfer through a volume of homogeneous participating media with a dedicated optional method to handle inhomogeneous boundaries. Contrary to previous approaches, our method only needs to walk the cloud boundaries rather than walking through the whole volume. We have demonstrated this technique on detailed cumulus-type clouds. As shown on our results, we correctly reproduce visual features such as back-scattering, anisotropic multiple scattering, glory, etc.

We believe this approach can be used in other applications where multiple scattering in well-bounded participating media is important such as sub-surface scattering, *e.g.,* as an alternative to the dipole approximation [Jensen et al. 2001]. As future research, our approach could also be used in offline rendering, where other representations (*e.g.,* meshes) can be used for the collector-finding algorithm, which would remove the limitations due to depth maps. It could also be used to find starting paths in a volumetric Metropolis light transport simulation.

12 Acknowledgments

We thank Guillaume Piolat for the days and night of coding, Florent Moulin and Laurence Boissieux for the week-ends spent modeling the clouds, Sébastien Barbier, Paul Kry, and the reviewers for the comments on the paper. INRIA provided the clusters and databases, managed by Eric Ragusi and Frédéric Saint-Marcel. The Mie phase function was generated by Philip Laven's MiePlot. This work was partially funded by the *Exploradoc* fellowship and the France-Berkeley Fund.

A Measuring light transport in a cloud slab

As explained in Section 2, the appropriate phase function for clouds is Mie scattering. It is a complex oscillating function depending of droplet size and wavelength. Like [Bouthors et al. 2006], we preintegrate the effect of the DSD on the Mie phase function and we rely on the *modified Mie* approximation in which the narrow (5°) strong forward peak is suppressed out of the phase function and converted into 50% lowered density [Lenoble 1985]. This proved to yield equivalent results while needing to consider only half of the scattering events and gave a better conditioned phase function.

For the DSD, we draw on the modified Gamma distribution [Levin 1958] (adapted to cloud droplets) defined by

$$N(r) = \frac{\rho N_0}{\Gamma(\gamma) r_n} \left(\frac{r}{r_n}\right)^{\gamma-1} e^{-\frac{r}{r_n}}. \qquad (3)$$

This function describes the density $N(r)$ of droplets of radius r, with r_n the characteristic radius of the distribution, γ representing its broadness, and N_0 is the total density of droplets. Γ is the gamma function. ρ is a modulation factor for the total density brought by our procedural enrichment described in Section 8. A cloud droplet size distribution is then only described by the three parameters N_0, r_n and γ. In terms of optical properties, the effective radius corresponding to this distribution is $r_e = (\gamma + 2) r_n$. We consider the following typical parameters for cumulus clouds: $r_e = 6\mu\text{m}$, $\gamma = 2$, $N_0 = 4.10^8\text{m}^{-3}$.

The *extinction coefficient* κ is defined by

$$\kappa = \rho N_0 \pi r_e^2. \qquad (4)$$

It yields the extinction function $e^{-\kappa x}$ which is the probability to traverse the cloud along a path of length x without hitting a droplet. It is used in all scattering equations, including opacity (Equation 2), single scattering (Equation 1) and multiple scattering (Equation 5).

The multiple scattering equation has been extensively discussed in the literature [Chandrasekhar 1960; Kajiya and von Herzen 1984]. It can be written in the form

$$-\frac{\vec{\omega}}{\kappa} \cdot \frac{dT(\mathbf{p}, \vec{\omega})}{d\mathbf{p}} = T(\mathbf{p}, \vec{\omega}) + \frac{1}{4\pi} \int_{\vec{\omega}'} P(\vec{\omega} \cdot \vec{\omega}') T(\mathbf{p}, \vec{\omega}') d\vec{\omega}' \quad (5)$$

for the intensity T reaching a point $\mathbf{p}$ in direction $\vec{\omega}$, with $P(\theta)$ the phase function of the media (here, Mie). This integro-differential equation can be solved computationally through Monte-Carlo integration. The contribution of each order of scattering in the intensity T, as well as the collector information $(\mathbf{c}, \sigma)$ can be easily tracked during the integration. For our light transport characterization step

(Section 5), we computed the solution of Equation 5 in a plane-parallel slab for various values of $(\phi_V, \psi_V, \phi_L, d, t)$ by ray tracing multiple scattering from the eye. We stored the contribution T of each scattering order into a database, along with the collector data $(\mathbf{c}, \sigma)$. These computations were done with an error margin of 5% at the 95% level. We computed them against 10 million different sets of values for $(\phi_V, \psi_V, \phi_L, d, t)$, resulting in a raw database size of 25GB. These computations took several weeks on a 100-nodes, dual-core 900MHz Itanium-2 cluster. The database and fitting results will be made accessible online.

The analysis of these results (*i.e.,* finding lower-dimensional functions fitting the results) was done empirically by plotting $\langle T, \mathbf{c}, \sigma \rangle$ against the input parameters and looking for remarkable behaviors. It was inspired by previous approaches such as Chandrasekhar's X- and Y-functions. The fitting itself was done using classical non-linear least square optimization with MATLAB.

References

ARVO, J. 2007. Alias-free shadow maps using graphics hardware. *journal of graphics tools 12*, 1, 47–59.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH'82*, 21–29.

BOUTHORS, A., NEYRET, F., AND LEFEBVRE, S. 2006. Real-time realistic illumination and shading of stratiform clouds. In *Eurographics Workshop on Natural Phenomena*.

CHANDRASEKHAR, S. 1960. *Radiative transfer*. New York: Dover, 1960.

CRASSIN, C., AND NEYRET, F. 2007. *Représentation et algorithmes pour l'exploration interactive de volumes procéduraux étendus et détaillés*. Master's thesis, INPG/UJF M2R - IVR.

DACHSBACHER, C., AND STAMMINGER, M. 2003. Translucent shadow maps. In *Eurographics Workshop on Rendering (EGWR)*, 197–201.

DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. 2000. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH'00*, 19–28.

EBERT, D. S. 1997. A cloud is born. In *SIGGRAPH'97*, 245.

ELINAS, P., AND STÜRZLINGER, W. 2000. Real-time rendering of 3D clouds. *J. Graph. Tools 5*, 4, 33–45.

GARDNER, G. Y. 1985. Visual simulation of clouds. In *SIGGRAPH'85*, ACM Press, 297–304.

GOODNIGHT, N., WANG, R., WOOLLEY, C., AND HUMPHREYS, G. 2003. Interactive time-dependent tone mapping using programmable graphics hardware. In *Eurographics Workshop on Rendering (EGRW)*, 26–37.

HARRIS, M. J., AND LASTRA, A. 2001. Real-time cloud rendering. *Computer Graphics Forum 20*, 3, 76–84.

HEGEMAN, K., ASHIKHMIN, M., AND PREMOŽE, S. 2005. A lighting model for general participating media. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)*, 117–124.

HOFFMAN, N., AND PREETHAM, A. J. 2003. *in Graphics programming methods*. Charles River Media, Inc., ch. Real-time light-atmosphere interactions for outdoor scenes, 337–352.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *SIGGRAPH'01*, 511–518.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray tracing volume densities. In *SIGGRAPH'84*, 165–174.

KRIM, M. S. A., AND EL WAKIL, S. A. 1986. Angular distribution of radiation in an isotropically scattering slab. *Astrophysics and Space Science 121* (Apr.), 137–145.

LENOBLE, J. 1985. *Radiative transfer in scattering and absorbing atmospheres: standard computational procedures*. A. Deepak Publishing.

LEVIN, L. M. 1958. Functions to represent drop size distribution in clouds. the optical density of clouds. *Izv. Akad. Nauk. SSSR, Ser. Geofiz. 10*, 198–702.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *SIGGRAPH'00*.

MAX, N. L., MOBLEY, D., KEATING, B., AND WU, E. 1997. Plane-parallel radiance transport for global illumination in vegetation. In *Eurographics Workshop on Rendering (EGWR)*, 239–250.

MAX, N. L. 1994. Efficient light propagation for multiple anisotropic volume scattering. In *Eurographics Workshop on Rendering (EGWR)*, 87–104.

MOBLEY, C. 1989. A numerical model for the computation of radiance distributions in natural waters with wind-roughened surfaces. *Limnol. Oceanogr. 34*, 1473–1483.

NEYRET, F. 2003. Advected textures. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA)*.

NISHITA, T., NAKAMAE, E., AND DOBASHI, Y. 1996. Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH'96*, 379–386.

PERLIN, K., AND HOFFERT, E. M. 1989. Hypertexture. In *SIGGRAPH'89*, 253–262.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH'85*, 287–296.

PREMOŽE, S., ASHIKHMIN, M., AND SHIRLEY, P. 2003. Path integration for light transport in volumes. In *Eurographics Symposium on Rendering (EGSR)*, 52–63.

PREMOZE, S., ASHIKHMIN, M., TESSENDORF, J., RAMAMOORTHI, R., AND NAYAR, S. 2004. Practical rendering of multiple scattering effects in participating media. In *Eurographics Symposium on Rendering (EGSR)*, 363–374.

RILEY, K., EBERT, D. S., KRAUS, M., TESSENDORF, J., AND HANSEN, C. 2004. Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering (EGSR)*, 375–386.

SCHPOK, J., SIMONS, J., EBERT, D. S., AND HANSEN, C. 2003. A real-time cloud modeling, rendering, and animation system. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 160–166.

STAM, J. 1995. Multiple Scattering as a Diffusion Process. In *Eurographics Workshop on Rendering (EGWR)*, 41–50.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *SIGGRAPH'02*.

TREMBILSKI, A., AND BROSSLER, A. 2002. Surface-based efficient cloud visualisation for animation applications. In *WSCG*, 453–460.

Figure 12: *Some results of our method.*