Diffusion and Multiple Anisotropic Scattering for Global Illumination in Clouds

Nelson Max
Lawrence Livermore National Lab.
Mail Stop L-560
7000 East Avenue
USA 94550 Livermore CA
max2@llnl.gov

Greg Schussman
Stanford Linear Accelerator Ctr.
Mail Stop 26
2575 Sand Hill Road
USA 94025 Menlo Park CA
schussman@SLAC.stanford.edu

Ryo Miyazaki, Kei Iwasaki, and Tomoyuki Nishita
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Japan 113-0033 Tokyo
{ryomiya, kei-i, nis}@nis-lab.is.s.u-tokyo.ac.jp

ABSTRACT

The diffusion method is a good approximation inside the dense core of a cloud, but not at the more tenuous boundary regions. Also, it breaks down in regions where the density of scattering droplets is zero. We have enhanced it by using hardware cell projection volume rendering at cloud border voxels to account for the straight line light transport across these empty regions. We have also used this hardware volume rendering at key voxels in the low-density boundary regions to account for the multiple anisotropic scattering of the environment.

Keywords

Diffusion approximation, multiple anisotropic scattering, global illumination, participating media, clouds.

1. INTRODUCTION

The appearance of a cloud is produced by multiple scattering of the incident illumination. The phase function for the result of two scattering events is the convolution of their two phase functions, and the convolution of any non-delta-function phase function with itself enough times approaches a uniform distribution, as shown in [Stam95]. In the interior of a cloud, the water droplets are dense, and the mean free path of a photon is short, so the large number of scattering events in a voxel will cause it to radiate scattered light diffusely. In the diffusion approximation, explained briefly in section 2 below, the equation for radiance transport is approximated by an equation for diffuse light transport, which can be more easily solved by multigrid relaxation. Kajiya and von Herzen [Kaj84] first proposed it to the computer graphics community, and Stam [Stam95] gave the first practical implementation for radiance transport in clouds. The diffusion calculations in [Stam95] assume a rectangular solid of non-zero cloud droplet density, which is unrealistic for actual clouds. Furthermore, Mie scattering from small water droplets is concentrated in the forward direction, and this directional scattering is evident in the subtle color and intensity effects at the edges of a cloud, where the expected number of scattering events is not large enough to produce anisotropy. The goal of our work is photorealistic rendering of clouds using the diffusion approximation on the dense core, with modifications to account for regions of zero droplet density, and multiple anisotropic scattering at the edges.

Older work on global illumination in participating media is well summarized in [Per97] and the references cited there. The basic Monte Carlo method of tracing random rays from the light source or the viewpoint suffers from the problem that few paths from the viewpoint reach the light source, and vice-versa. By tracing photons from the light source, accumulating scattering events into a voxel data structure called a photon map, and then doing a final gather from the viewpoint by ray tracing the photon map, Wann Jensen and Christensen [Jen98] solve this problem. However, in realistically large and dense clouds, the photons will basically make a random

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

walk through the dense medium, and progress a straight line distance O($\sqrt{n}$) in n bounces, total path length n, or computation time n. Thus illumination on one side of a dense cloud will take a very long time to reach the other side. Recently, Premoze, Ashikmin, and Shirley [Prem03] have applied the path integral method from quantum mechanics to make a more efficient approximation, but it can be biased because it takes into account mainly the forward scattering, and does not account for all light paths.

Jensen *et al*. [Jens01] proposed an approximate solution to the diffusion equations, using a dipole of two point sources, one above and one below a planar object surface, and a simple analytic solution for the diffusion equation for point sources in a homogeneous medium. Jensen and Buehler [Jens02] applied this to objects represented by surface point clouds, using an octree hierarchy to cluster the effects of irradiance at points far from the position being rendered. Mertens *et al*. [Mer03] used a similar hierarchical method on polygonal surfaces, with a semi-analytic integration for the effect of the irradiance on a triangle. Lensch *et al*. [Len02] used triangle-based piecewise linear basis functions for global transport, and texel-based piecewise constant basis functions for local transport. Dachsbacher and Stamminger [Dach03] combine this local/global transport separation with the hierarchical evaluation into a texture-based approximation, which can be evaluated in real time using programmable vertex and fragment shaders. The dipole approximation used in all these methods assumes that the surface is planar, and that the medium is uniform and optically thick. These methods give plausible realistic renderings for more complex geometry, but cannot deal with the non-diffuse scattering discussed above at the tenuous edges of clouds, where the mean free path of a photon is comparable to the geometry feature size, and the diffusion approximation is invalid.

2. DIFFUSION APPROXIMATION

The following is a brief summary of the derivation of the diffusion approximation. For details, see [Ish78]. (Note that the derivation in [Stam95] is incorrect, and Stam's equations differ from Ishimaru's by factors of 3 in several terms.)

The equation for light propagation in a participating medium is

$$\frac{dI(X,\omega)}{ds} = -\kappa(X)I(X,\omega) + \kappa(X)\int_{4\pi} a \cdot p(\omega,\omega')I(X,\omega')d\omega', \qquad (1)$$

where X is 3D position, ω and ω' are ray directions on the unit sphere, $I(X,\omega)$ is radiance at X flowing in direction ω, s is length along the ray in direction ω, $\kappa(X)$ is the extinction coefficient, a is the albedo, the scattered fraction of the extinction, and $p(\omega,\omega')$ is the phase function giving the probability density that scattered light from ω' scatters to ω. (Note that in [Ish78], $p(\omega,\omega')$ also includes the albedo a.)

In a cloud, $\kappa(X) = \rho(X)\ \sigma_t$, where $\rho(X)$ is the number density of water droplet particles, and σ_t is the extinction "cross section" per particle. Also let $\sigma_s = a\ \sigma_t$ be the scattering cross section per particle, and $\sigma_a = \sigma_t - \sigma_s$ be the absorption cross section per particle.

Assume that the phase function $p(\omega,\omega')$ depends only on the angle θ between the unit vectors ω and ω', and let

$$p_1 = \int_{4\pi} p(\omega,\omega')(\omega\cdot\omega')d\omega$$

be the average cosine of the scattering angle θ. For the Henyey-Greenstein scattering phase function

$$p(\theta) = \frac{1-g^2}{4\pi(1-2g\cos\theta+g^2)^{\frac{3}{2}}}$$

which we use, $p_1 = g$.

Let $I_{ri}(X, \omega)$ be the reduced incident radiance, which can be computed from the sun and sky radiance $I_{sky}(\omega)$ by the attenuation formula

$$I_{ri}(X,\omega) = I_{sky}(\omega)\exp\left(-\int_0^\infty \sigma_t\rho(X-s\omega)ds\right). \qquad (2)$$

Let

$$I_d(X,\omega) = I(X,\omega) - I_{ri}(X,\omega)$$

be the "diffuse" radiance that has been scattered at least once. Approximate $I_d(X, \omega)$ with the first order dependence on the unit direction vector $\boldsymbol{\omega}$:

$$I_d(X,\omega) = I_0(X) + \boldsymbol{I}_1(X)\cdot\boldsymbol{\omega}.$$

Substituting

$$I(X,\omega) = I_{ri}(X,\omega) + I_0(X) + \boldsymbol{I}_1(X)\cdot\boldsymbol{\omega} \qquad (3)$$

into equation (1), and integrating ω over the unit sphere, we get a scalar equation. Then multiplying by the unit vector $\boldsymbol{\omega}$ and integrating over the unit sphere, we get a vector equation. We solve this vector equation for $\boldsymbol{I}_1(X)$ in terms of $I_0(X)$ and its derivatives, and substitute the result into the scalar equation, which also involves the divergence of $\boldsymbol{I}_1(X)$. By doing some vector algebra, vector calculus, and mathematical manipulations on these equations, we end up with the single "diffusion" equation

$$\nabla \cdot \left(\frac{1}{3\rho(X)\sigma_{tr}} \nabla I_0(X) \right) - \rho(X)\sigma_a I_0(X) = -Q_0(X) + \nabla \cdot \left(\frac{1}{4\pi\rho(x)\sigma_{tr}} \boldsymbol{Q}_1(X) \right) \quad (4)$$

where

$$\sigma_{tr} = \sigma_t(1 - ap_1)$$

is the "transport cross section",

$$Q(X,\omega) = \rho(X)\sigma_s \int_{4\pi} p(\omega,\omega') I_{ri}(X,\omega') d\omega'$$

is the first scattered external illumination,

$$Q_0(X) = \frac{1}{4\pi} \int_{4\pi} Q(X,\omega) d\omega \quad (5)$$

is the first scattered external illumination averaged over the unit sphere, and

$$\boldsymbol{Q}_1(X) = \frac{1}{4\pi} \int_{4\pi} Q(X,\omega)\ \boldsymbol{\omega}\, d\omega \quad (6)$$

is its directionally weighted average, a vector.

The incident sun and sky illumination are assigned to direction bins on the subdivided surface of a unit cube, as shown in Figure 1*b*). The power in bin i is $I_{sky}(\omega_i)\Delta\omega_i$, where $\Delta\omega_i$ is the solid angle of bin i. This power is propagated into the voxels of the cloud volume to compute $I_{ri}(X,\omega_i)$ by tracing a dense collection of rays in the direction ω_i , starting from the external faces of the box enclosing the cloud. Each ray intersects the voxels in a collection of segments S_j, along which the attenuation integral in (2) can be accumulated. The lengths $l(S_j)$ of these segments weight the contributions of $I_{ri}(X,\omega_i)$ to the integrals in (5) and (6) at the voxel X. So the contribution to the integral (5) of a ray segment S_j is $\rho(X)\ \sigma_s\ I_{ri}(X,\omega_i)\ \Delta\omega_i\ l(S_j)$. The phase function $p(\omega,\omega')$ depends only on the angle θ between ω and ω'. Thus the contribution of light flowing in direction $\omega' = \omega_i$ to the integral (6) lies only along direction ω_i , since the other components cancel by symmetry. It therefore equals $\rho(X)\ \sigma_s\ I_{ri}(X,\omega_i)\ \Delta\omega_i\ l(S_j)\ p_1\ \boldsymbol{\omega}_i$. After these contributions are summed for all rays in direction bin i, the values of the sums for (5) and (6) in each voxel are normalized by dividing by the sum of the lengths $l(S_j)$ of the segments it contains. The scalar $Q_0(X)$ and the vector $\boldsymbol{Q}_1(X)$ are saved at each voxel X.

The sky and sun colors were computed taking into account Rayleigh scattering from the molecules in the atmosphere, using the methods of [Nis96], [Dob97], and [Pre99], and Mie scattering from aerosols, using the methods of [Slo02].

Boundary Conditions

For the boundary conditions at the surface of the voxel regions containing non-zero droplet density, we really want the inward diffuse radiance $I_d(X, \omega)$ to be exactly zero for all directions ω with $\boldsymbol{\omega} \cdot \boldsymbol{n} > 0$, where $\boldsymbol{n}$ is the inward surface normal. But this is impossible for a function of the form $I_0(X,\omega) + \boldsymbol{I}_1(X, \omega)\cdot\boldsymbol{\omega}$ unless the function is zero for all ω. So an approximate condition of no total inward flow is used:

$$\int_{2\pi} I_d(X,\omega)\ (\boldsymbol{\omega} \cdot \boldsymbol{n})\, d\omega = 0,$$

where the integral is over the inward hemisphere. After some mathematical manipulations, this results in the boundary condition:

$$I_0(X) - \frac{2}{3\rho(X)\sigma_{tr}} n \cdot \nabla I_0(X) + \frac{1}{2\pi\rho(X)\sigma_{tr}} n \cdot Q_1 = 0 \quad (7)$$

For more details on the derivation of this equation and of equation (4), see [Ish78].

Finite Difference Solution

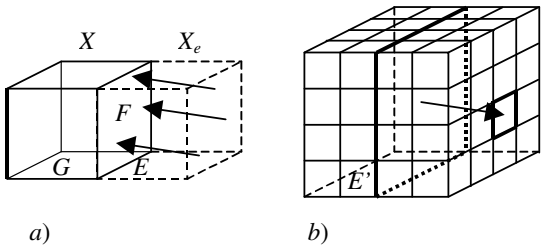
To solve the partial differential equation (4) we use finite differences to estimate the derivatives. The difference for the gradient of I_0 is evaluated at the face centers, and for the value of ρ in the first term of equation (4), the average of the $\rho(X)$ for the two cells sharing the face is used, unless one of them has $\rho = 0$, in which case the value at the other cell is used. Central differences are used to evaluate the gradient for the last term of equation (4). All this results in a set of linear equations relating each value $I_0(X)$ to the values at its six adjacent neighbors. The boundary condition (7) is enforced by assuming a temporary value of $I_0(X_e)$ at an adjacent empty cell X_e where the droplet density is zero, set to satisfy (7), and using it to compute the differences for equation (4). This affects the coefficients of the linear equations, which end up not involving $I_0(X_e)$, so the temporary value of $I_0(X_e)$ can be different for different full cells adjacent to the same empty cell X_e.

The resulting set of linear equations can be solved iteratively by the Gauss-Seidel or Jacobi method, by setting each $I_0(X)$ to a weighted sum of its old value and the old values of its six neighbors, plus a constant term involving Q_0 and $\boldsymbol{Q}_1$ that comes from the constant right hand side of the linear equation. This will take a long time for a large grid, since a bright spot caused by sunlight hitting one side of the cloud can only propagate across one adjacent cell per iteration, so it will take many iterations for it to affect the opposite side of the cloud. Therefore a multigrid method is used. This is like mip mapping. The data

volume is considered as a multi-resolution octree hierarchy, and coarser versions of the linear equations are written for lower levels of the hierarchy. Their solutions are used to correct the solutions at finer levels of the hierarchy to allow for faster propagation, and the solution method iterates up and down the levels in the hierarchy until convergence. For details, see [Bri00]. We used the Hypre system API and multigrid solver from Lawrence Livermore National Laboratory [Hypre03].

3. PROPAGATION IN CLEAR AIR

If X_e is an empty border voxel with $\rho(X_e) = 0$, adjacent across face F to a full voxel X, with $\rho(X) > 0$, as shown in Figure 1*a*), we place a viewing cube, shown in Figure 1*b*), at the center of X_e. Then, as in the hemicube algorithm for radiosity, we render the scene, including the clouds, onto the six faces of this viewing cube. We used 64 by 64 resolution images on each cube face, and then averaged the colors into the 4 by 4 direction bins for each face, as shown in Figure 1*b*), to give a total of 96 direction bins for the six faces.



a) *b*)

Figure 1. *a*) Light propagating from an empty cell to a full cell. *b*) A viewing cube divided into bins.

Figure 1*a*) shows several rays in the direction ω_i at the center of direction bin i, crossing face F. Instead of continuing to propagate and attenuate $I_{ri}(X,\omega_i)$ to accumulate the values of Q_0 and $\boldsymbol{Q}_1$ as described above, we replace the propagating value by the average color seen in direction bin $-\omega_i$, which is indicated in bold lines in Figure 1*b*), in order to include the inscattered light from the clouds. If F is the only face across which X_e is adjacent to a full cell, all directions ω_i crossing F from X_e to X have flow towards the left, so we need only render the hemicube to the right of the bold line in Figure 1b). Similarly, if X_e is adjacent to only two full voxels, across faces F and G that share a common edge E, then the needed incoming directions fill two overlapping hemicubes, and we do not render the two half faces adjacent to the dashed edge E' shown in Figure 1*b)*.

The diffusion approximation assumes that the clouds are dense, so that enough multiple scattering takes place to make the radiance almost independent of direction. As described above and in [Ish78], $\boldsymbol{I}_1(X)$ can be computed from $I_0(X)$ and its derivatives. However the linear approximation (3) for $\boldsymbol{I}(X)$ is not valid at the edges of the cloud, where single scattering, and orders of multiple scattering too low to insure directional independence, are important. Therefore the approximation (3) cannot be used to render an image of the cloud surface.

Instead, hierarchical volume rendering is used to render the volume with opacities $\rho(X)\sigma_t$ and voxel colors $I_0(X)$. (The solution described above is done for the red, green, and blue wavelength bands.) An octree hierarchy of least squares approximations to the color and opacity values on the various level cubes is precomputed, together with the RMS error of each approximation. Then, in a view dependent manner which weights the error by the projected area of the cube on the image, appropriate cubes are selected, and composited in back to front order, using the hardware cell-projection method. The details are given in [Schu01]. For this application, the algorithm in [Schu01] was modified to handle color, and to use either piecewise linear or piecewise constant functions in the approximation hierarchy.

The revised values for Q_0 and $\boldsymbol{Q}_1$ obtained by propagating the incident radiance from this volume rendering are used again in obtaining a new solution to the partial differential equation (4), and the sequence of PDE solution, volume rendering, and ray propagation is iterated until I_0 stops changing significantly. The illumination and shadowing of the ground by the sky and clouds is also found using hardware volume rendering onto hemicubes placed on a ground sample grid, and the diffusely reflecting ground is included as a background to the volume rendered radiance images of the cloud. The sky hemisphere, colored as described above, is also part of the background. However the radiance of the sun exceeds the dynamic range of the fixed-point frame buffer, so the sun is not added to the sky background. Instead, the direct sunlight, $I_{sun}(X_e,\omega_{sun})$ is propagated and attenuated separately, and saved as $I_{risun}(X_e,\omega_{sun})$ at every voxel. Its effects are added separately to Q_0 and $\boldsymbol{Q}_1$.

4. FINAL GATHER

The final rendering is a "final gather" ray tracing of the volume solution to equation (4). The reduced sun flux $I_{risun}(X_e,\omega_{sun})$ is used, together with the exact phase function $p(\omega_{sun}, \omega_{viewing})$ to give an exact final single scattering of the attenuated sunlight, similar to two pass volume rendering for shadow effects. This can give the brightening effect on the edges of a cloud when the sun is directly behind it. However, it cannot account for the multiple directional scattering

of the sunlight, nor of the skylight, which contributes to the darker and bluer color of the edges of the cloud, when seen against other non-edge cloud regions as a background.

To include a more accurate multiple directional scattering effect for the regions near the edges of the cloud, where the particle density is below a threshold T necessary for the diffusion approximation to be adequate, we again use volume rendering. When doing the final gather ray tracing per pixel, we determine low-density ray segments S_j where the ray stays within the edge region where $0 < \rho(X) < T$. For each such segment, we integrate $\rho(X)\sigma_t$ along the segment to get the optical depth D_j, put a cube at the "center" point c_j where the integral reaches half this value D_j, and compute the radiance on the faces of the cube using volume rendering, as above. The x axis of the cube is aligned with the direction of the viewing ray, as shown in Figure 2.

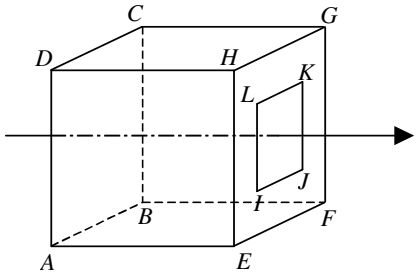


Figure 2. Faces and subface of the viewing cube.

Knowing D_j, the Poisson distribution [Poi97] can tell us the probability q_{ij} of scattering of scattering exactly i times along this segment:

$$q_{ij} = \frac{\exp(-D_j)D_j^i}{i!} . \tag{8}$$

This is only an approximation, because the scattering will change the direction of the ray. However, for small water droplets inside clouds, the scattering is highly peaked in the forward direction, so for the first few orders of scattering, it is a good approximation.

To get the phase function for the result of i scattering events, we must convolve the phase function $p(\omega,\omega')$ with itself i - 1 times, using integration over the unit sphere. This is particularly simple for the Henyey-Greenstein phase function, because the result is the same formula, with g replaced by g^i, as can be proved by expanding $p(\omega,\omega')$ as a Legendra series in $\cos\theta$. These multiple scattering phase functions $p_i(\omega,\omega')$ are precomputed, and saved for each pixel on the faces of the viewing cube shown in Figure 2. Then for each ray segment S_j, they are multiplied pixelwise by the radiance images on the cube and summed to give a convolution sample K_{ij}. Finally the K_{ij} are multiplied by the probabilities q_{ij} and summed over i to get the contribution E_j of the segment S_j to the viewing ray through a pixel:

$$E_j = \sum_i q_{ij} K_{ij} . \tag{9}$$

If the centers c_{jkl} of a collection of segments S_{jkl} for a region of pixels (k, l) form a continuous surface, there is a possible speed-up. The higher order (large i) scattering convolutions K_{ijkl} of the radiance images are blurred, and thus change slowly with the pixel index (k, l). Therefore they can be interpolated from a sparser sampling of the pixels in the region. Only the probabilities q_{ijkl} needs to be computed at every pixel. On the other hand, for smaller i, only the front face of the cube containing the viewing direction need be accurate, because the scattering is mainly forward. For even smaller i, only a central section of the front face is required. This saves rendering time because the clipping to the view frustum is performed while traversing the octree hierarchy, so clipping eliminates large octree cells, and their children are not even considered.

We used three different viewing cube sampling patterns. The first is for the rear five faces of the viewing cube in Figure 2: *ABCD*, *AEFB*, *BFGC*, *CGHD*, and *DHEA*, which are sampled initially along rays at every 8 output image pixels in x and y, as indicated by the * symbols in Figure 3 below. This figure shows the method of computing the convolutions for these five faces, for the first low-density segment, if any, along the viewing rays for part of a low resolution image.

```
   2 1     2 - 4 1                                    
 1 - 1 1 1 - - - 1 1                                1
 1 2 - 2 - 2 - - - 2 1                        2 - 2 -
 1 - - - - - - 1 - - 1 1     1 1 1 - - - -
4 - - - 4 - - - 4 - - - 4 - - - 4 - - - 4 - - - 4 -
- - - - - - - - - - - - 1 - - - - - - - - - - -
- - - - - - - - - - - - 2 - - - - - 2 - 2 -
- - - - - - - - - - - - - - - - - - - - 1 1 1
4 - - - * - - - - - - - * - - - 4 - 2 1
- - - - - - - - - - - - - - - - - - - 1
2 - 2 - - - - - - - - - - - - - - - - 2 - 2
1 1 1 - - - 1 - - - - - - - - - - - - - -
  1 2 - - - - - - - - - - - - - 4 - - - 4
  1 - - - - 1 - - - - - - - - - - - - - -
  1 2 - - - - - - - - - - - - - - - - - - -
  1 - - - - - - - - - - - - - - - - - - - -
    2 - * - - - - - - - * - - - - - - - * -
```

Figure 3. Convolution computation locations.

The convolutions K_{ijkl} are bilinearly interpolated to the positions indicated by the symbol "–" if the values of the distances along the ray at which the viewing cubes were placed are close enough. If not, a new viewing cube is placed at the appropriate distance along the ray, and new values of K_{ijkl} are calculated. These new calculations are done hierarchically at the vertices of a quadtree, in order to find values that may be satisfactory for interpolation in one or more sub-squares. Thus a `4` in Figure 3 indicates a new viewing cube along a ray at an output image pixel with x and y coordinates divisible by 4, a `2` indicates a new viewing cube along a ray at a pixel with x and y coordinates divisible by 2, and a `1` indicates a new viewing cube for a pixel whose x or y coordinate is an odd number. A blank indicates that the viewing ray had no low-density segments.

The second sampling pattern was for the front face *EFGH* of the viewing cube, which was initially sampled every 4 pixels in x and y, with doubled resolution, in order to give more accurate interpolated convolutions near the peak of the forward scattering phase function. The final pattern uses only the center quarter *IJKL* of the front face of the viewing cube, initially sampled every 2 pixels in x and y. The total pixels were the same as on the front face in the previous case, so the pixel spacing is half the spacing on the whole front face, and a quarter the spacing on the other five cube faces. This central section is disregarded when computing the convolution for the whole front face. Thus, every direction is counted exactly once in the three methods, and the convolutions which have been interpolated or computed for each of them can be added to get the total convolution for each low-density ray segment.

The convolutions are computed hierarchically on 9 by 9 blocks of pixels, so that when interpolation is allowed, the convolutions to be interpolated are already known. The blocks are computed in horizontal sweeps across the image, so that the left hand column of pixels is known from the previous block, and the top row of pixels is known from the previous sweep of blocks. Thus the convolutions need only be saved for the right hand column of the previous block, and for one full image row, representing the bottom row of the previous sweep's blocks. Once the convolutions are known for the low-density segments, equations (8) and (9) are evaluated per pixel, and the voxel segments on the viewing ray are composited from front to back, using either the low-density convolution solution (appropriately weighted by the optical depth of the voxel segment), the high-density diffusion solution, or a density dependent interpolation of the two (in order to eliminate aliasing artifacts from switching solutions).

5. RESULTS

We used the final time step of a cloud simulation produced by a hierarchical octree enhancement of the simulation method of [Miy02], using the stable semi-Lagrangian advection scheme of [Stam99] and the vorticity confinement method described in [Fed01]. The simulation was at resolution 200 x 160 x 120, and was averaged down to 100 x 80 x 60 cells for rendering. A rectangular solid enclosing the cells with non-zero droplet density had size 91 x 51 x 25.

Figure 4 shows a cloud with only single scattering of the sun illumination. Since the phase function for the droplets has $p_1 = g = 0.9$, and the scattering is strongly peaked in the forward direction, with little scattering towards the viewpoint, the cloud is very dark and has been artificially brightened. Figure 5 show the final gather of the diffusion approximation solution with both sun and sky illumination, but no correction for the low-density boundary regions, and no propagation across the clear air. Figure 6 shows the added effects of the volume rendering method of accounting for the low-density regions at the edge of the cloud, using only the initial sun, sky and ground illumination in the diffusion solution. Figure 7 shows the result of one iteration of the clear air propagation, using volume rendering at the border voxels to image the clouds. If one flips between Figures 6 and 7 in a screen image viewer, one can see that some regions of the cloud look brighter in Figure 7. An image from two iterations of the clear air propagation is indistinguishable from Figure 7.

Figure 7 took 21 minutes and 33 seconds to compute, on a 1.7 GHz Pentium 4 with nVidia GeForce3 graphics. Of this time, 46 seconds was spent setting up and solving the linear equations for diffusing the incident sunlight, 2 seconds for setting up the octree hierarchy, 227 seconds to render the 12460 viewing cubes at the border cells around the clouds, at resolution 64 by 64 per face, and the 32 by 32 array of hemicubes on the ground sample grid, 94 seconds to propagate the flux from the ground, sky, and cloud in 96 directions though the cloud, another 46 seconds to solve the revised linear equations taking into account this extra flux, and 854 seconds to render the image, including the time to render the viewing cubes at the low-density segments, compute the convolutions, interpolate and weight them, and ray trace the 480 by 360 image.

Figure 8, at a time near sunset, took 24 minutes and 45 seconds. The final rendering time took longer, 1034 seconds, because there were more cloud pixels in this image, due to different viewing parameters.

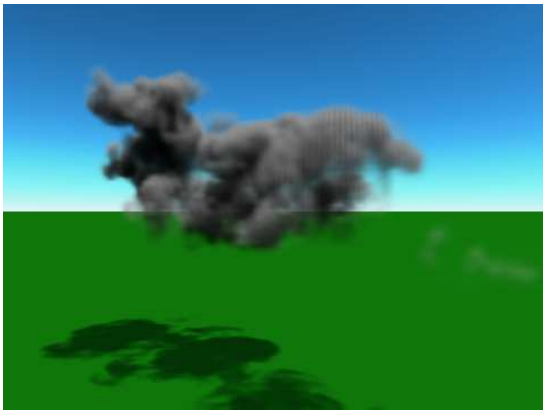


Figure 4. Single scattering only.

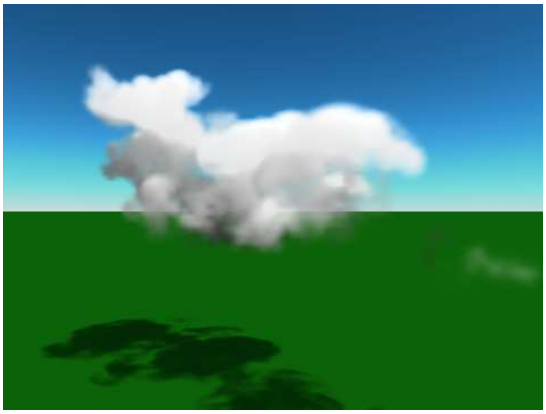


Figure 5. Initial diffusion solution.

Figure 6. Initial diffusion solution with correction for low-density regions.

6. DISCUSSION AND FUTURE WORK

The hardware rendering component of our work suffered from the dynamic range limitations of the 8 bit fixed point format of our graphics hardware pipeline. We hope to produce more accurate results using floating-point graphics pipelines and frame buffers. The computing time and memory sizes limit the data resolution for our diffusion and light propagation passes, but we hope to add extra resolution to the final gather pass, by ray tracing the original volume instead of the averaged version, and interpolating the radiance solution. We also hope to add a subvoxel-resolution Perlin noise texture and to advect the texture coordinates using the velocity from the weather simulation.

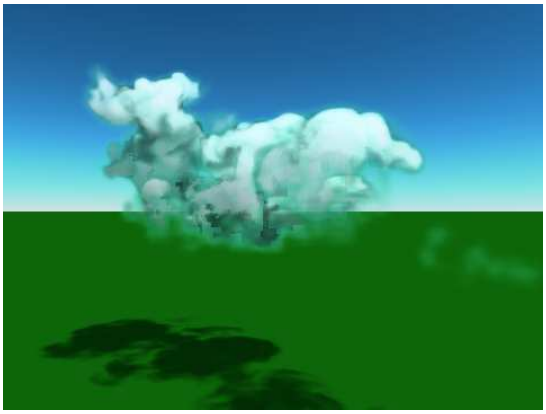


Figure 7. Diffusion solution with correction for low-density regions, after one iteration of propagation through clear air.

Figure 8. Image at sunset.

The artifacts near the bottom of Figures 6 and 7 come from the transitions where two low-density segments join into one. The probabilities q_{ij} for larger numbers of scattering events are higher for the larger optical depth of the single low-density segment, so that more illumination is included from the sides, and less from the front. We should add some kind of transition to alleviate the edges caused by this abrupt switch. Currently for the volume rendering in the final gather, all cells except the one containing the viewpoint c_j are rendered. We should also remove any cell intersecting the low-density segment S_j, in order to avoid double counting the scattering along this segment.

7. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy *a*) by the University of California, Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, and *b*) at the University of California, Davis under Memorandum Agreement No. DE-FC02-01ER41202 through the SciDAC Program. We thank the WSCG reviewers, Kwan-Liu Ma, and Fabrice Neyret for suggestions that improved the exposition, Fabrice Neyret for the conversations that inspired this work, Jos Stam and Phil Collela for advice, Jim Jones and Rob Falgout for help with Hypre, Jos Stam for his multigrid code which we used before switching to Hypre, and Robin Bing-Yu Chen, Henry Johan, and Kenichi Amou for system help at the University of Tokyo, where the first author did much of this work as a visiting professor.

8. REFERENCES

[Bri00] Briggs, W., Henson, V. E., and Steve McCormick, S., "A multigrid tutorial: Second edition", SIAM, Philadelphia, 2000.

[Dach03] Dachsbacher, C., and Stamminger, M., "Translucent shadow maps", Eurographics Symposium on Rendering 2003, pp. 197 – 201.

[Dob97] Dobashi, Y., Nishita T., Kaneda, K., and Yamashita, H., "A fast display method of sky color using basis functions", The journal of visualization and computer animation, Vol. 8 No. 2, pp 115 – 127, 1997.

[Fed01] Fedkiw, R., Stam, J., and Jensen, H. V., "Visual simulation of smoke", Proceedings of Siggraph '01, pp. 15 – 22, 200.

[Ish78] Ishimaru, A., "Wave propagation and scattering in random media. Volume 1: Single scattering and transport theory", Academic Press, New York, 1978.

[Jen98] Jensen, H. W., and Christensen, P., "Efficient simulation of light transport in scenes with participating media using photon maps", Proceedings of Siggraph 1998, pp. 311 – 329.

[Jen01] Jensen, H. W., Marchner, S., R., Levoy, M., and Hanrahan, P., "A practical model for subsurface light transport", Proceedings of Siggraph 2001, pp. 511 – 518.

[Jen02] Jensen, H. W. and Buhler, J., "A rapid hierarchical rendering technique for translucent materials, ACM Transactions on Graphics Vol. 21 No. 3 pp. 576 – 581, 2002.

[Kaj84] Kajiya, J., and von Herzen, B., "Ray tracing volume densities", ACM Computer Graphics (Proceedings of Siggraph 1984) Vol.18 No. 3, pp. 165 – 174.

[Hyp03] http://www.llnl.gov/CASC/hypre

[Len02] Lensch, H., Goesele, M., Bekaert, Ph., Kautz, J., Magnor, M., Lang, J., and Seidel, H-P., "Interactive rendering of translucent objects", Proceedings of Pacific Graphics 2002, pp. 214 – 224.

[Mer03] Mertens, T., Kautz, J., Bekaert, Ph., Seidel, H-P., and Reeth, F. V., "Interactive rendering of translucent deformable objects", Eurographics Symposium on Rendering 2003, pp. 130 – 140.

[Miy02] Miyazaki, R., Dobashi, Y., and Nishita, T., "Simulation of cumuliform clouds based on computational fluid dynamics", Proceedings of Eurographics 2002 Short Presentations, pp. 405-410, 2002.

[Nis96] Nishita T., Dobashi, Y., Kaneda, K., and Yamashita, H., "Display method of sky color taking into account multiple scattering", Pacific Graphics '96, pp. 117 – 132, 1996.

[Per97] Perez, F., Pueyo, X., and Sillion, F., "Global illumination techniques for the simulation of participating media", in "Rendering Techniques '97", Dorsey and Slusallek, editors, Springer, Vienna, pp. 309 – 320, 1997.

[Poi97]http://info.bio.cmu.edu/Courses/03438/PBC97 Poisson/PoissonPage.html

[Pre99] Preetham, A. J., Shirley, P., and Smits, B., "A practical analytic model for daylight", Proceedings of Siggraph 1999, pp. 91 – 100.

[Prem03] Premoze, S., Ashikmin, M., and Shirley, P., "Path integration for light transport in volumes", Proceedings of the Eurographics Symposium on Rendering, pp. 52 – 63, 2003.

[Schu01] Schussman, G., and Max, N., "Hierarchical perspective volume rendering using triangle fans", Volume Graphics 2001, Stony Brook, NY, pp. 195 – 200.

[Slo02] Sloup, J., "A survey of the modelling and rendering of the earth's atmosphere", SCCG 2002, pp. 141 – 149, 2002 .

[Stam95] Stam, J., "Multiple scattering as a diffusion process", in "Rendering Techniques '95", Hanrahan and Purgathofer, editors, pp. 41 – 50, Springer, Vienna, 1995.

[Stam99] Stam, J., "Stable fluids", Proceedings of Siggraph '99, pp. 121 – 128, 1999.