Real Time Rendering of Atmospheric Scattering and Volumetric Shadows

Venceslas Biri, Didier Arquès, Sylvain Michelin

Real Time Rendering of Atmospheric Scattering and Volumetric Shadows

Biri Venceslas
Charles Cros Institute
6 bd du Danube
F-77700 SERRIS
FRANCE
biri@univ-mlv.fr

Arquès Didier
Charles Cros Institute
6 bd du Danube
F-77700 SERRIS
FRANCE
arques@univ-mlv.fr

Michelin Sylvain
Charles Cros Institute
6 bd du Danube
F-77700 SERRIS
FRANCE
michelin@univ-mlv.fr

ABSTRACT

Real time rendering of atmospheric light scattering is one of the most difficult lighting effect to achieve in computer graphics. This paper presents a new real time method which renders these effects including volumetric shadows, which provides a great performance improvement over previous methods. Using an analytical expression of the light transport equation we are able to render directly the contribution of the participating medium on any surface. The rendering of shadow planes, sorted with a spatial coherence technique, and in the same philosophy than the shadow volume algorithm will add the volumetric shadows. Realistic images can be produced in real time for usual graphic scenes and at a high level framerate for complex scenes, allowing animation of lights, objects or even participating media. The method proposed in this paper use neither precomputation depending on light positions, nor texture memory.

Keywords : Real time rendering / Volumetric shadows / Single scattering / Participating media

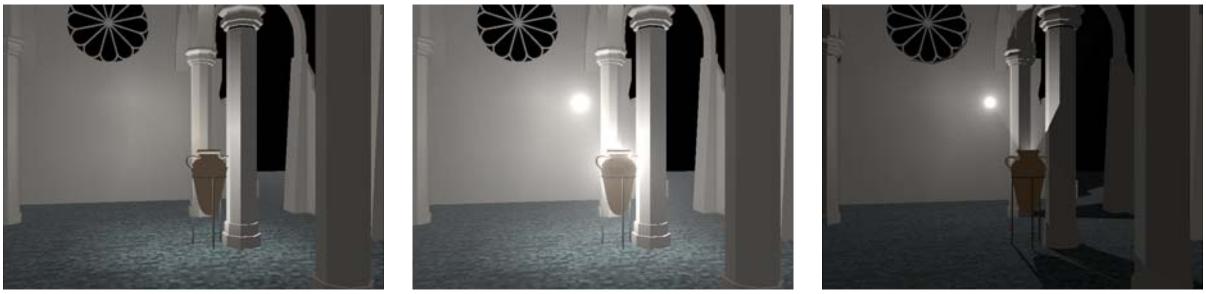


Figure 1: The same scene lit a. (left) classically, b. (center) with single scattering and c. (right) with single scattering and volumetric shadows (right) .

1. INTRODUCTION

The growing capacities of graphic cards enable the rendering of more and more complex physical models in real time, like anisotropic reflection or environment mapping.. Therefore, it is not a surprise if a current challenge in computer graphics is the accurate rendering of atmospheric effects, and especially the light scattering. Atmospheric light scattering is due to little particles -like dust or water - that lay in the air, scattering and absorbing the light they receive. They creates effects such light beams, shafts of light and visibility loss. These phenomena often occur under foggy or smoky conditions but are also visible by clear or cloudy weather in the presence of sunlight.

Unfortunately, rendering such lighting effects in real time remains quite complex since they depend on camera and light positions and since they occur everywhere in the space. Introducing such effect in traditional graphic engine will greatly enhance the

Conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

realism of the virtual scene and have many applications [Ru94]. Considering the particular situation of figure 1, it is clear that rendering the participating medium is not enough. Here, the representation of shadow volumes is necessary to obtain a realistic image. Thus there is a need for a simple algorithm, easily integrated in traditional algorithms, able to render those effects.

In this paper, we present an algorithm that fulfills this goal. It can render accurately participating media, including effects like light beams in foggy or smoky scenes, or any other atmospheric scattering effects. The participating media can be isotropic or anisotropic and are lit by one or several, static or moving, point light sources since no precomputation are done involving either lights or camera. Our technique produces high resolution images and takes into account volumetric shadows, cast by occluders contained in the media. Without any texture memory cost, but using intensively graphics hardware, our method can render images at a high frame rate, and is real time for classical graphic scene. The method is also easy to implement in traditional graphic engines since it follows the same strategy than the shadow volume algorithm. Therefore, it is straightforward to obtain animations where objects, sources and even participating media can move.

2. PREVIOUS WORK

The representation of participating media has been a real challenge for years and the literature about it is abundant. We can easily divide all these studies between the single scattering methods and the multiple scattering ones. Multiple scattering methods try to compute all light reflections and inter-reflections inside a medium, whatever the number of these ones. This complex situation is difficult to handle but is essential in the rendering of clouds for example. Multiple scattering illumination can be obtained by determinist methods [RT87, Ma94, ND96] or by stochastic methods [PM93, LW96, JC98] and sometimes involve a resolution of the flow equations like in [FM97, St99, DK00, FS01]. Despite their realism, they suffer from excessive computation times due to the complexity of light exchanges occurred in these cases. Therefore it is not suitable for our goal and we will focus on single scattering methods.

These techniques [NM87, Ma94, DY00, HP02, DY02] approximate the multiple reflections of light as a constant ambient term and consider only the first scattering of light ray in the direction of camera. This assumption allows a direct rendering of the illumination of the medium which is more suitable for interactive rendering. Visualization is often done by ray tracing or ray marching. View rays are followed to gather the participating media contributions. Unfortunately, these methods [FM97, JC98], are far from being real time on a classical computer. With the growing capacities of graphics hardware, the real time problem has been investigated.

Two approaches can be used to achieve this goal: volume rendering or direct representation. To add the volumetric shadows the first approach will use naturally shadow maps techniques when the second one is oriented to shadow volumes algorithm [He91]. Volume rendering is a classic solution to render participating medium which is a volume *de facto*. Methods like [BR98, WE98, St99, FS01, NM01] represent densities or illumination in voxels encoded into 2D or 3D textures. Accumulation techniques using textured slices or virtual planes are then used to display the result. That kind of methods could produce nice images of clouds or gas. But apart from requiring a lot of texture memory, they are not suitable for shafts of light where sharp edges exist. Special methods are defined to render beams and shafts of light precisely and most of them [DK00, DY00, Ev02, LG02] use volume rendering techniques along with sampling shadows in shadow maps. But they suffer from artifacts due to the sampling. Dobashi et al. [DY02] presents a very elegant solution to solve this problem using specialized adaptive sampling for shadows. They obtain an interactive rendering of participating media without aliasing or artifacts. However the image resolution remains small since the method is expensive in terms of fillrate. Moreover, the method works only with static lights due to the precomputation of shadow maps.

The algorithms belonging to the second approach computes directly, on every point in the scene, the contribution of the participating medium. This is well adapted to classical graphic engines since it consists in one more rendering of the scene. In this case, methods like [Me01, HP02] use participating medium boundaries, or special virtual planes, combined with vertex and fragments shaders. Another method of this group is proposed by Sun et al. [SR05] and is the only one to consider the effect of light scattering on the illumination of objects. Despite it is real time, it does not take into account shadows. Our work belongs also to this group and is the only one of them to integrate realistic lighting effect with volumetric shadows.

3. OVERVIEW OF OUR METHOD

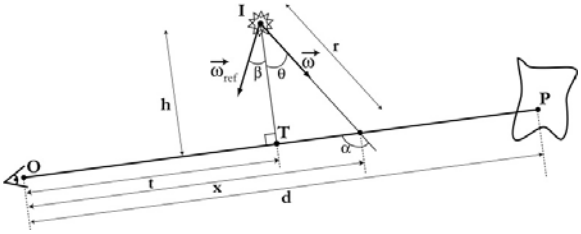
To obtain real time performances, we consider only one scattering of light in the medium. Multiple scattering is approximated by a constant ambient term in the scene and each participating medium is homogeneous.

The algorithm exploits an analytical expression of the total contribution of scattered light along a view ray. This allows the direct computation of this contribution between the camera and any point of the scene. Therefore, we compute the previous expression:

- on scene vertices or on boundaries of participating media.
- on any point of the shadow planes.

As stated before, our method is close to shadow volume techniques [He91] or other algorithms using shadow planes [AA03]. Indeed, after having compute and render light scattering contribution of lit areas, we do the same for the shadow planes of any object that is set to cast shadows.

These shadow planes are classically obtained by using the object silhouettes regarding to the point light position and meshed. After a back to front sorting of the shadow planes, we render them. The participating medium contribution will be added if the shadow plane is frontfacing and subtracted if backfacing to take into account the volumetric shadows.



k_t	: Extinction coefficient	Is	: Light intensity
Ω	: Albedo	p	: phase function
L	: Luminance		

Figure 2 : Single scattering case and notations.

4. THEORETICAL BACKGROUND

As light is progressing through a participating medium, it have four interactions with it : absorption, emission, scattering and in-scattering [SH92]. If we consider only single scattering, the luminance of a point P seen from a point O can be written [SH92]:

$$L(O) = L(P)e^{-k_t d} + \frac{k_t \Omega}{4\pi} \int_0^d \frac{I_s(\vec{\omega}) e^{-k_t x} e^{-k_t r}}{r^2} p(\alpha) dx \qquad (1)$$

The first term takes into account the scattering and the absorption while the second one is the in-scattering which is responsible for the subtle effects of atmospheric scattering. This equation is called the integral transfer equation.

4.1 Angular formulation of the integral transfer equation

The integral transfer equation can be written [LM00] using the angle between the view ray and the direction toward point light. This formulation will be used to obtain an analytical solution of the previous equation. Indeed, instead of integrating the integral transfer equation regarding to the distance *x* along the ray, we choose to use the variation of the angle θ between the vector $\vec{\omega}$ and the vector $\overrightarrow{ST}$ defined by the orthogonal projection of the point light source on the view ray. Using this variable change :

$$r = \sqrt{h^2 + (x-t)^2}$$

we can obtain (see [LM00]) :

$$L(O) = L(P)e^{-k_t d} + L_m(P)$$

where :

$$L_m(P) = \frac{k_t e^{-k_t t}\Omega}{4\pi h} \int_{\theta_0}^{\theta_d} I_s(\theta+\beta) e^{-k_t h \frac{\sin(\theta)+1}{\cos(\theta)}} p(\theta + \frac{\pi}{2}) d\theta$$

Further on, *L_m(P)* will be called the medium contribution of the point P.

The kernel *Λ* of the previous integral is complicated enough to prevent any analytic integration. But we can approximate this kernel to obtain a much more simple expression. The function in the kernel without considering the light intensity, depends only on the angle θ, considering that the extinction coefficient is constant, i.e. that the participating medium is homogeneous. Therefore, we can develop its expression in a polynomial base (we use 4 degree) :

$$e^{-k_t h \frac{\sin(\theta)+1}{\cos(\theta)}} p(\theta + \frac{\pi}{2}) \approx c_0(k_t, h) + c_1(k_t, h)\theta + \dots$$

For traditional phase functions -- isotropic, hazy, murky... -- the formal expressions of coefficients *c* can be obtained in the annex.

We introduce this equation in the expression of *L_m(P)* to obtain :

$$L_m(P) = \frac{k_t e^{-k_t t}\Omega}{4\pi h}\left[c_0(k_t,h)\int_{\theta_0}^{\theta_d} I_s(\theta+\beta)d\theta + c_1(k_t,h)\int_{\theta_0}^{\theta_d} I_s(\theta+\beta)\theta d\theta + \dots \right]$$

Finally, for non directional point light source, these integrals are easily computed :

$$L_m(P) = \frac{k_t e^{-k_t t}\Omega}{4\pi h}\left[c_0(k_t,h)[\theta]_{\theta_0}^{\theta_d} + c_1(k_t,h)\left[\frac{\theta^2}{2}\right]_{\theta_0}^{\theta_d} + \dots \right] (2)$$

and so we can obtain the single scattering contribution created by a point light source along any view ray in constant time.

A study on the quality of these approximations can be found in [Le01]. In general, they are quite good except when the ray passes close to the source, or when the observer is far from the source. In the first case, the contribution is so high, and in the second case, so small, that these errors remain unnoticeable.

Thanks to expression (2), we are now able to compute in “constant time” -- i.e. without any numerical integration -- the contribution of in-scattering light along a ray contained in a participating medium.

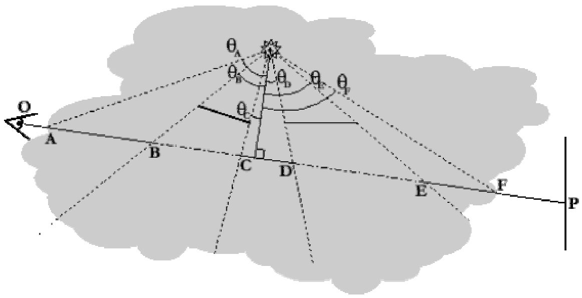


Figure 3 : A view ray partially in shadows.

4.2 Considering shadow volumes

Previous equations describe the particular case where the view ray remains totally lit and lays in the participating medium. To integrate shadow volumes and bounded participating medium, we need to consider more general cases, illustrated in figure 3. Indeed, due to shadows, the part of the ray laying in the medium could be split into lit and shadowed parts. In this example, the medium contribution along the ray is split into three parts on AB, CD and EF. The contribution of the single scattering of the ray OP is then:

$$L_m(P)=\frac{k_t e^{-k_t t}\Omega}{4\pi h}\left[\int_{\theta_A}^{\theta_B}\Lambda(\theta)d\theta+\int_{\theta_C}^{\theta_D}\Lambda(\theta)d\theta+\int_{\theta_E}^{\theta_F}\Lambda(\theta)d\theta\right]$$

The key idea of our approach is to rewrite this equation into a sum of differences. Indeed the light contribution of segment EF for example can be seen as the contribution of segment OF minus the one from segment OE. If we denote $\Gamma_m(P)$ the expression (2) for a lit ray between the camera center O and any point P, then the previous equation can be written :

$$L_m(P)=\left[(\Gamma_m(F)-\Gamma_m(E))+(\Gamma_m(D)-\Gamma_m(C))+(\Gamma_m(B)-\Gamma_m(A))\right]$$

It is also obvious that points B, C, D and E are located on shadow planes, and that the points A and F belong to the boundary of the participating medium. Of course point F and P can merge for object contained in the medium, and if it covers the entire scene, points A and O will also merge.

Finally, when considering a bounded medium, the equations are slightly different. The coefficient r and x in the exponentials of equation (1) must be the distance between the point X and the border of the medium boundary. In our method, we approximate r to the average distance R between a point located in the boundary and a point in the medium. So R is a constant along the ray. The new value xn of x is computed on the fly and is also a constant along the ray. In this case, $L_m(P)$ becomes :

$$L_m(P)=\frac{k_t e^{-k_t(r+R-xn)}\Omega}{4\pi h}\int_{\theta_0}^{\varsigma_d} I_s(\theta+\beta)\, e^{-k_t h\frac{\sin(\theta)+1}{\cos(\theta)}}\, p\left(\theta+\frac{\pi}{2}\right)d\theta$$

what only involves a change of coefficients c.

5. RENDERING ALGORITHM

5.1 Scenes Recovered by a Participating Medium

In this case, every view ray is contained entirely in the participating medium. The method is easy to implement and works as follows:

1. The silhouettes of every moving shadow caster are computed. If light is moving, every silhouette needs to be recomputed.
2. Scene is rendered classically. Surface shadows can be obtained using simple and traditional shadow planes algorithms [He01, EK02]. The stencil buffer now contains lit areas of the scene. An ambient fog is added to take into account both absorption and multiple scattering.
3. Scene is rendered once more and medium contribution is computed for each vertex of the scene. Depth test is set to the equality. Only lit parts of the scene are rendered thanks to the stencil buffer.
4. Shadow planes, determined by the object's silhouettes, are sorted in a back to front order.
5. Shadow planes are rendered in that precise order. The depth test function accepts only planes that are closer to the camera. Front facing planes add their contribution when back facing planes subtract them. Stencil function is set to allow fragments if the stencil is equal to 1 for front facing planes and 0 for back facing ones. Front facing planes always increment the stencil buffer and back facing ones always decrement it.

All stages have to be done for each light source. Each stage is detailed in the following sections.

5.1.1 Computation of silhouettes

In our algorithm, we select some objects to be shadow caster. Their silhouettes are easily computed in determining all edges of their mesh common to a front-facing triangle regarding the light position and one back facing it. Then all these edges are linked together if possible, and stored in a loop list. To obtain correct silhouettes, we need closed triangular meshes (2-manifold) for which connectivity information are available. These conditions for the shadow casters are the same ones that are indicated in [EK02].

Shadow planes are infinite quads formed by a silhouette edge and the light position. They are constituted by the two edge's vertices and two other points, projection of the previous vertices to infinity toward direction : light position - vertex (cf. [He91]). They are oriented toward the unshadowed area of the scene. As we need to compute the medium contribution on all shadow planes, it is necessary to use shadow plane silhouettes rather than the shadow planes of all little triangles. Of course, if the light does not move, only moving shadow caster silhouettes have to be computed. Finally, in case the input geometry is modified by graphics hardware, using displacement mapping for example, a solution to obtain silhouettes of all objects quickly and accurately can be found in [BS03].

5.1.2 Rendering the scene

The scene is rendered classically except for the light attenuation due to absorption and scattering induced by the participating medium. Thus, the luminance of any point P of the scene is :

$$L(P) = \rho_P \frac{I(\omega_P)\cos(\theta_P)}{\pi r_P^2} e^{-k_t r_P}$$

where ρ_P is the reflectivity coefficient of the surface, θ_P the angle between the normal of the surface , *I* the intensity of the light in the direction ω toward *P* and r_P simply the distance to the light point. A simple vertex program can render this equation which differs from the traditional one only in the exponential attenuation.

In this stage we also add a fog effect to take into account both absorption and multiple scattering. We also compute the hard shadows and use the stencil algorithm and its improvements [He91, BS03] to do so. Indeed, they fit perfectly with our application since, we already have the silhouettes. In the end of this stage, the stencil buffer contains the lit areas of the image. Until the end of the image rendering, the lighting is disabled.

5.1.3 Medium contribution of the scene

Still using stencil test, the scene is rendered once more to add, with additive blending, the medium contribution of every surface. This is simply done by computing equation (2) for each vertex. The depth test is set to the equality.

5.1.4 Sorting the shadow planes

Before rendering all shadow planes, we have to make sure that we will not render shadow planes, or part of them, that are themselves in shadow. If we do not care about this problem, it will create artifacts we call shadow in shadows, illustrated in figure 4. In the left image, we can see that the shadow of the top plane is propagated in the shadow of the bottom plane.

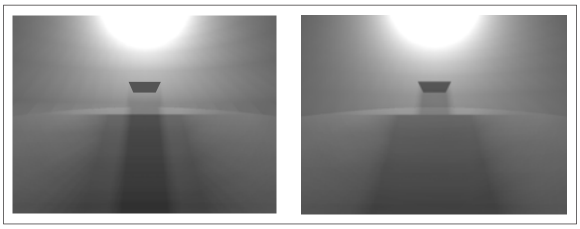


Figure 4: Shadow in shadows artifacts.

To prevent these artifacts we render the shadow planes, back- or front-facing, in a “back to front"' order and use the stencil buffer to avoid the rendering of shadowed shadow planes. The distance we defined for the back to front order depends on both camera and light positions. In two dimension, we can see in figure 5 that the plan (a line in 2D) created by the edge A (a point in 2D) must be rendered before the one created by B. And this one must be rendered before the shadow plane of edge C. This is true whatever the distance between the edge and the camera or between the edge and the light position. A simple realization of such a distance is to compute, for an edge P, the cosine between vectors $\overrightarrow{SO}$ and $\overrightarrow{SP}$ where O is the camera center, S the light position, and P a point belonging to the silhouette.

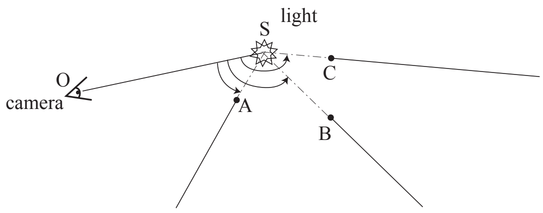


Figure 5: Ordering of shadow planes (in 2D)

We use the same ordering in 3D. In this case, the silhouette edges are segments. Since silhouettes are accurately meshed, these segments can be considered as points (only for ordering). Therefore, we compute the same cosine using as point P the center of the silhouette edge.

5.1.5 *Rendering the shadow planes*

We always keep the stencil we have obtained in the stage 2. Shadow planes are rendered in the order defined in the previous stage with the depth test function admitting only fragments that are closer to the camera.

The color attributed to the shadow planes -- i.e. their contribution -- are computed with exactly the same expression than for lit point of the scene in stage 3, i.e. using equation (2) for homogeneous point light. Front facing planes add their contribution and back facing planes subtract them.

We have to mesh the shadow planes to obtain accurate values of the medium contribution. They will be computed in each vertex of the mesh and the GPU will make the interpolation between them. According to the radial distribution of a point light, it is wise to mesh the shadow planes finely when close to the light and coarsely when far away. It is not necessary to subdivide the silhouette edge which has to be small.

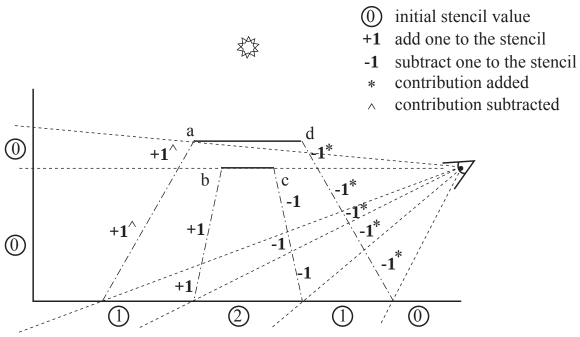


Figure 6: Use of the stencil buffer in the rendering of shadow planes

To take into account correctly the shadow in shadow problem, we use the stencil intensively. Front facing planes pass the stencil test if its value is one (representing shadowed area), and back facing ones passes if it equals 0 (value representing lit area). Ideally the back (resp. front) facing quads should always add (resp. subtract) one to the stencil buffer if it passes depth test. But unfortunately it is not possible to specify two different stencil functions when a fragment fails the stencil test depending of the result of the depth test. It imposes us to render the simple quad of the shadow plane with the stencil function set to always. Such problem will disappear when programmability of graphics card will involve the stencil test. Nevertheless this strategy works in all the case as illustrated in figure 6. The strategy indicated works if the camera is in the light. A slightly different strategy can be used when the camera is in shadow but the philosophy remains the same.

5.2 Rendering Several Bounded Participating Media

Several modifications have to be made to the previous algorithm to take into account boundaries of participating media and to avoid the rendering of each object and each shadow planes for every medium. Indeed, when several participating media exist, stages 3 to 5 need to be computed for each one of them. For simplicity we consider only convex participating media, and that we have a mesh representation of it.

First of all we will compute bounding boxes for each object and each participating medium. This is to avoid the rendering of objects that do not lay in the area of a participating medium in stage 3. We also check for each shadow plane if it is able to cut the participating medium.

In stage 3, we use the equivalent of a shadow volume algorithm to determine shadowed and lit area of the boundary of the participating medium. Then we render the lit areas of objects and of the medium boundary. Objects are rendered only if they belong to the medium bounding box. The front facing triangles of the medium boundary are rendered using the expression seen in section 4.

In stage 4, back-facing triangles of the medium boundary are also sorted and integrated in the order list. Since we use their center for the reference point P in the ordering, these triangles must be small. For each shadow plane, we determine if it is able to cut the medium bounding box. If not, it is removed from the sorted list to avoid unnecessary computation.

Finally the stage 5 remains the same, except that when a back facing triangle of the medium boundary is rendered, we set the stencil to 255 to avoid any further rendering in this area.

6. RESULTS

The previous algorithm has been implemented on a standard computer using a 2.6 GHz processor and a ATI 9800 graphic card. All images that we will present have a 800x600 resolution. We first compare our method with the work of Dobashi et al. [DY02] using their simple scene, a sphere beyond a spot light. In our case, the spot light is obtained by adding a cone above our point light. The silhouette has 32 edges which involves 32 shadow planes. Our rendering time is about 120 frames per second at resolution 800x600. In our case, resolution is not really a problem. For example, the same scene using a 1024*768 resolution is rendered at 107 FPS. For the same test scene, Dobashi's algorithm achieves 12.5 FPS for a 450x300 resolution. This is mainly due to the accumulation of texture rendering inducing a high fill rate.

A drawback exists in our method which is only due to the clamping of the framebuffer. Indeed, when we render the contribution of the medium, it is possible that the final value added to the one present in the framebuffer exceeds 1. In that case, the value is clamped to 1 and if we subtract a medium contribution after that, the final result will be darker than it should be. However, this problem can be avoided in choosing reasonable intensity for the light source, or in the future, using a float texture. Unfortunately, it is impossible for the moment to use blending when rendering in a float buffer.

We also present in table 1 the ratio of work loads for each stage. As expected, we can see that the computations of the shadow plane contributions represent the main cost of the whole process.

Stage	1	2	3	4	5
Fig 7 left	1%	8.7%	21.7%	9.6%	59%
Fig 7 right	1.7%	14%	33.6%	16%	34.7%

Table 1. Work loads for each stage.

We also present some snapshots of our animations. The first image in figure 7.a. is a simple scene, where two pens are bumping in front of a light. It illustrates a classical situation where well design 3D objects are moving and casting shadow. This scene is rendered at more than 35 fps. The image in figure 7.b. represents a simple scene with a box contained into three different participating media, one red, one blue and one green, moving before the light. Here we can clearly see the volumetric shadows of each participating media and how they blend together. As we use exact shadow planes no aliasing occurs. This case illustrates the ability of our algorithm to handle all the position between shadow planes and the boundary of a participating medium. Figure 7.c. is a snapshot from a animation where the light is moving, and its color is also changing. We have chosen this scene because it contains a lot of shadow planes. Finally figure 1.c., in the first page of this paper, is also a snapshot to illustrate the use of our algorithm when light is moving in a complex scene, containing around 100 000 triangles. Table 2 presents the FPS and the number of triangles of those scenes.

Scene	Fig 7a	Fig 7b	Fig 7c	Fig 1
FPS	23	35	25	12
Nb. triangles	34 549	14 785	20 747	107 514

Table 2. FPS and number of triangles of scenes.

7. CONCLUSION

We have presented a new real time algorithm that is able to compute the single scattering of one or several participating media. Our algorithm is fast enough to handle more than 25 frames per second for moderately complex scenes, which is an improvement over other atmospheric scattering algorithms, especially when a medium covers the whole scene. As outlined above, the only computations that we have done in software are the participating medium contributions and the ordering and the computation of shadow planes. Moreover, we plan to design vertex and fragment shaders to make the graphic card computes the participating medium contributions. We also want to point out that our algorithm does not create any sampling aliasing artifact, for both surface and volumetric shadows, thanks to the use of exact shadow planes.

As shadow planes have become more popular recently, we think that our algorithm fit perfectly with this kind of approach and is well adapted to the growing capacities of graphics hardware. For example, the final improvement of the algorithm would be to compute soft surface shadows and soft volumetric shadows. For this goal we can take inspiration of the algorithm [AM03]. Finally, both clustering and culling approaches will greatly speed up this already fast algorithm.

8. REFERENCES

[AM03] Assarson U., Möller T.A., A Geometry-based Soft Shadow Volume Algorithm using Graphics Hardware, In proceedings of SIGGRAPH'03, Computer Graphics, vol. 22 (3), pp. 511-520

[BR98] Behrens U., Ratering R. , Adding Shadows to a Texture-based Volume Renderer. In proceedings of 1998 symposium on Volume Vizualisation , 1998, pp. 39-46

[BS03] Brabec S., Seidel H.P., Shadow Volumes on Programmable Graphics Hardware. In proceedings of Eurographics'03, 2003, vol. 22(3)

[DK00] Dobashi Y., Kaneda K., Yamashita H., Okita T., Nishita T., A Simple, Efficient Method for Realistic Animation of Clouds. In proceedings of SIGGRAPH'00, Computer Graphics, 2000, pp. 19-28

[DY00] Dobashi Y., Yamamoto T., Nishita T., Interactive Rendering Method for Displaying Shafts of Light. In proceedings of Pacific Graphics 2000, pp. 31-37.

[DY02] Dobashi Y., Yamamoto T., Nishita T., Interactive Rendering Method of Atmospheric Scattering Effects Using Graphics Hardware. In proceedings of Graphics Hardware 2002, 2002, pp. 99-107.

[EK03] Everitt C., Kilgard M., Practical and Robust Shadow Volumes, Nvidia white paper, 2003 http://developer.nvidia.com/object/robust_shadow_volumes.html.

[Ev02] Everitt C., A Fast Algorithm for Area Light Source Using Backprojection. In proceedings of SIGGRAPH'94, Computer Graphics, 1994, pp. 223-230

[FM97] Foster N., Metaxas D., Modeling the Motion of a Hot, Turbulent Gas. In proceedings of SIGGRAPH'97, Computer Graphics, 1997, pp. 181-188

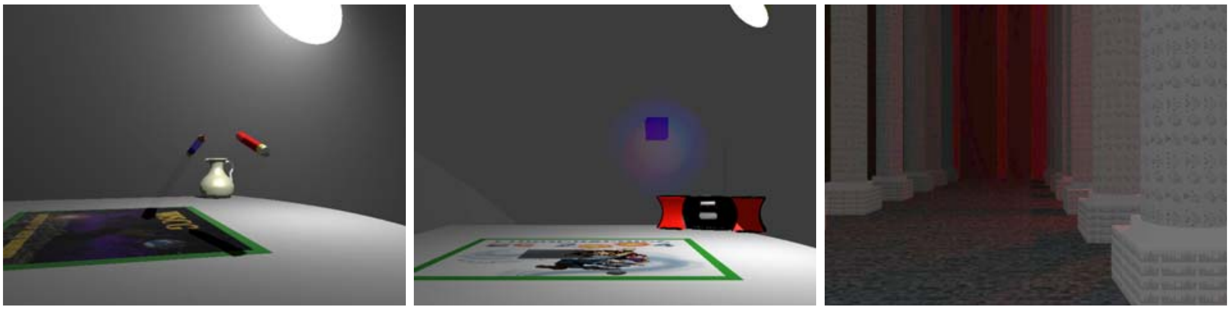


Figure 7. a. (left) Two pen moving b. (center) Three participating media moving c. (right)Light is moving in a complex scene

[FS01] Fedwik R., Stam J., Jensen H., Visual Simulation of Smoke. In proceedings of SIGGRAPH'01, Computer Graphics, 2001, pp. 15-22.

[He91] Heidman T., Real Shadows Real Time. In IRIS Universe (1991), vol. 18, pp 28-31

[HP02] Hoffman N., Preetham A., Rendering Outdoor Light Scattering in Real Time. ATI white paper, 2002. www.ati.com/developer/dx9/ATI-LightScattering.pdf

[JC98] Jensen H., Christensen P., Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps. In proceedings of SIGGRAPH'98, Computer Graphics, pp 311-320

[Le01] Lecocq P., Simulation d'éclairage temps réel par des sources lumineuses mobiles et statiques : outils pour la simulation de conduite. PhD Thesis of the University of Marne-la-Vallée, 2001.

[LG02] Lefebvre S., Guy S., Volumetric Lighting and Shadowing, NV30 Shader, 2002. lefebvre.sylvain.free.fr/cgshaders/vshd/vshd.html

[LM00] Lecocq P, Michelin S., Arquès D., Kemeny A., Mathematical Approximation for Real Time Rendering of Participating Media considering the luminous intensity distribution of light sources. In proceedings of Pacific Graphics 2000. pp 400-401.

[LW96] Lafortune E., Willems Y., Rendering Participating Media with Bidirectional Ray Tracing. In proceedings of 6th Eurographics Workshop on Rendering, june 1996, pp. 92-101.

[Ma94] Max N., Efficient Light Propagation for Multiple Anisotropic Volume Scattering. In proceedings of 5th Eurographics Workshop on Rendering, 1994, pp. 87-104

[Me01] Mech R., Hardware-Accelerated Real Time Rendering of Gaseous Phenomena. In Journal of Graphics Tool, 2001, vol. 6(3), pp. 1-16

[ND96] Nishita N., Dobashi Y, Nakamae E., Display of Clouds Taking into Account Multiple Anistropic Scattering and Skylight. In proceedings of SIGGRAPH'96, june 1996, pp. 379-386

[NM87] Nishita N., Miyawaki Y, Nakamae E., A shading model for atmospheric scattering considering luminous distribution of light sources. In proceedings of SIGGRAPH'97, Computer Graphics, vol. 21(4), pp. 303-310

[NM01] Nulkar M., Mueller K., Splatting with shadows. In proceedings of Volume Graphics 2001, pp. 35-49

[PM93] Pattanaik S., Mudur S., Computation of global illumination in a participating medium by monte carlo simulation. In The journal of Visual and Computer Animation, 1993, vol 4(3), pp. 133-153

[RT87] Rushmeier H., Torrance K., The zonal method for calculating light intensities in the presence of participating medium. In proceedings of SIGGRAPH'87, computer graphics vol 21(4), pp. 293-302.

[Ru94] Rushmeier H., Rendering participating media : problems and solutions from application areas. In proceedings of 5th Eurographics Workshop on Rendering, june 1994, pp. 35-56.

[SH92] Siegel R., Howell J., Thermal Radiation Heat Transfer. 3rd ed. Hemisphere Publishing, 1992.

[SR05] Sun B., Ramamoorthi R., Narasimhan S.G., Nayar S.K., A practical analytic single scattering model for real time rendering. In proceedings of SIGGRAPH'05, Computer Graphics; 2005, vol 24(3), pp. 1040-1049.

[St99] Stam J., Stable fluids. In proceedings of SIGGRAPH'99, Computer Graphics, 1999, pp. 121-128

[WE98] Westermann R., Ertl T., Efficiently using graphics hardware in volume rendering applications. In proceedings of SIGGRAPH'98, Computer Graphics, 1998, pp. 169-177

9. ANNEXES

Expression of coefficients c for classical phase functions. Isotropic phase function :

$$c_0 = e^{-k_t h}$$

$$c_1 = -k_t h \; e^{-k_t h}$$

$$c_2 = \left(k_t^2 h^2 - k_t h\right) e^{-k_t h}$$

$$c_3 = \left(-\frac{(k_t h)^3}{6} + \frac{k_t^2 h^2}{2} - k_t h\right) e^{-k_t h}$$

$$c_4 = \left(\frac{(k_t h)^4}{24} - \frac{(k_t h)^3}{4} + \frac{11\, k_t^2 h^2}{24} - \frac{5 k_t h}{24}\right) e^{-k_t h} \; \ldots$$

Rayleigh phase function :

$$c_0 = \frac{3}{4} e^{-k_t h}$$

$$c_1 = \left(-\frac{3\; k_t h}{4}\right) e^{-k_t h}$$

$$c_2 = \left(\frac{3\, k_t^2 h^2}{8} + \frac{3}{4}\right) e^{-k_t h} \; \ldots$$

Hazzy phase function :

$$c_0 = \frac{265}{256} e^{-k_t h}$$

$$c_1 = \left(-\frac{265 k_t h}{256} + \frac{9}{36}\right) e^{-k_t h}$$

$$c_2 = \left(\frac{265 k_t^2 h^2}{256} - \frac{121\, k_t h}{512} + \frac{63}{64}\right) e^{-k_t h} \; \ldots$$

Murky phase function :

$$c_0 = \frac{2147483673}{2147483648} e^{-k_t h}$$

$$c_1 = \left(\frac{2147483673\; k_t h}{2147483648} - \frac{25}{367108864}\right) e^{-k_t h}$$

$$c_2 = \left(\frac{2147483673\; k_t^2 h^2}{4294967296} - \frac{2147482073\; k_t h}{4294967296} + \frac{775}{134217728}\right) e^{-k_t h} \; \ldots$$