IADIS International Journal on Computer Science and Information Systems
Vol. 9, No. 2, pp. 96-114
ISSN: 1646-3692

PARTICLE-BASED PARTICIPATING MEDIA RENDERING USING DENSITY OCTREES

Richard Monette. *Carleton University, 1125 Colonel By Drive, Ottawa, ON, CANADA, K1S 5B6*

Anthony Whitehead. *Carleton University, 1125 Colonel By Drive, Ottawa, ON, CANADA, K1S 5B6.*

ABSTRACT

In order for computer generated imagery to recreate the characteristic visual appearance of phenomena such as smoke and fog it is necessary to compute the way in which light interacts with participating media. In this work we present a novel technique for computing volumetric single scattering lighting solutions for particle-based inhomogeneous participating media data sets. We seek to calculate volumetric lighting solutions for particle-based data sets as such data sets have the advantage of being spatially unbounded and relatively unrestricted with regard to memory as compared to uniform grids. In order to perform the calculations which are required for computing such a lighting solution, we introduce a novel octree based data structure. We refer to this new data structure as a density octree. The design of the density octree allows for efficiently computing light attenuation throughout the spatial extent. Using our data structure, we are able to produce high quality output imagery of arbitrary particle data sets in the presence of arbitrary numbers of lights.

KEYWORDS

Rendering, Graphics, Participating Media, Octree, Efficiency

1. INTRODUCTION

One of the most compelling aspects of computer graphics is the rendering of so called participating media phenomena such as smoke, fog and clouds. These phenomena are referred to as participating media in that their presence in a virtual scene to be rendered affects the transfer of light between the solid surfaces and the camera. Aesthetically pleasing treatment of these phenomena is critical to achieving high quality, realistic computer graphics imagery that are comparable to the photograph in Figure 1. This paper presents a high quality, temporally coherent and computationally efficient technique for rendering such participating media. Specifically, we introduce a new method for computing volumetric lighting solutions for participating media which is modeled by particle data sets, as opposed to traditional grid based representations. We seek to render particle-based participating media because modern fluid dynamics solutions use particle based data sets, instead of traditional grid-based methods.

Figure 1. Particle media example from real photography

The primary contribution of this paper is a novel technique for computing single scattering volumetric lighting solutions for non-homogeneous particle-based participating media data sets consisting of arbitrary numbers of particles for an arbitrary number of lighting types in an arbitrary volume. The technique presented exhibits logarithmic complexity with respect to the number of particles and linear complexity with regard to the number of lights. This performance characteristic is achieved by modelling the particle density within the spatial extent using an octree.

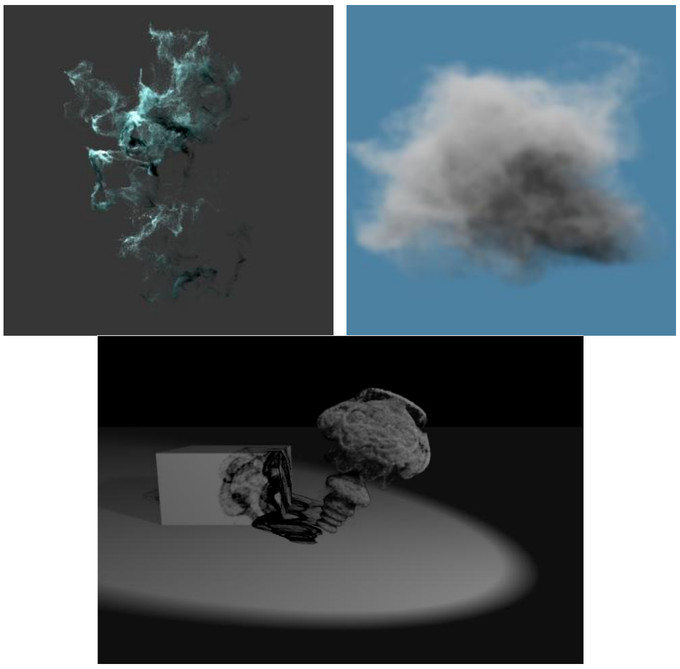


Figure 2. Images rendered using the algorithm described in this paper.

The density octree allows for computationally efficient evaluation of attenuation along each light path between the particles and the lights in the data set. Our work extends that conducted by Knoll, who uses a similar octree method, however only to represent solid geometry (Knoll 2008). The volumetric lighting software routines developed for this paper have been included in the Exocortex Fury renderer(Madill, Houston 2010), a production ready product and has been used in productions.

In Section 2, we begin by establishing a theoretical basis which describes the various types of interactions which light may have with participating media and a short survey of relevant research is provided. In Section 3 we discuss the system architecture and the features it allows. In Section 4, we present our performance results and a selection of example output images. We conclude with a discussion of directions for future work and conclusions in Sections 5 and 6, respectively.

2. BACKGROUND

In general, computer graphics solutions rely upon the assumption that the scenes to be rendered are made up of a collection of solid surfaces existing in a vacuum. This assumption greatly simplifies rendering since, in a vacuum, radiance (the measure of brightness and color of a single ray of light (Akenine-Moller et al, 2008)) is constant along any path between the surfaces, lights and virtual cameras. However, this assumption, while beneficial in terms of reducing computational complexity, has the shortcoming of preventing the rendering of effects such as attenuation and scattering of light due to participating media such as fog, smoke and other atmospheric phenomena.

To capture the appearance of such phenomena, we must use those techniques which calculate the interaction of the light with the participating media present in the scene. Participating media is matter, such as water droplets (in the case of fog) or dust particles (in the case of smoke), which will affect the behavior of light by changing the direction in which it travels and the amount of energy it carries due to processes such as scattering and absorption. For a complete treatment of the topic we refer the reader to the comprehensive resource, Physically Based Ray Tracing (Pharr & Humphreys, 2004). Next, we introduce the main ideas of Participating Media.

2.1 Participating Media Interaction Processes

Participating media is characterized as being either homogeneous or non-homogeneous in nature. A homogeneous media is uniform throughout the scene whereas a non-homogeneous media spatially varies in properties such as density, color or other attributes. Four main processes affect the distribution of radiance in a scene in the presence of some participating media (Chandrasekhar, 1960):

Absorption - Absorption is the process by which radiance interacts with participating media reducing the total transmitted radiance. Absorption is measured using the absorption coefficient α which is the probability that light is absorbed per unit distance traveled in the medium. In the case of a non-homogeneous medium this absorption coefficient may be spatially varying, whereas it will be uniform in a homogeneous medium

Emission - Increases the amount of radiance traveling along some path through a participating medium due to the conversion of energy into visible light by chemical, thermal or other means. For example, consider a candle that releases energy, some of which will be in the form of visible light as wax particles burn and emit light.

Scattering - As a beam of light passes through some participating media, it may intersect with the particles that constitute that media. This interaction can cause the light to change the direction in which it is traveling. The effects of this scattering are twofold. The first outcome is that the total radiance will be reduced due to the *out-scattering* of light energy. However, it is also possible that light will enter the medium and be scattered such that it increases the total radiance which reaches the virtual camera. This outcome, where the total radiance is increased, is referred to as *in-scattering*

2.2 The Equation of Transfer

The equation of transfer is the fundamental equation that governs the behavior of light in a medium that absorbs, emits and scatters radiation (Chandrasekhar, 1960). It accounts for all of the volume scattering processes, which are; absorption, emission and in- and out-scattering. These factors provide an equation that describes the distribution of radiance in an environment. In fact, the light transport equation is a specialized case of the equation of transfer, which has been simplified by the removal of consideration of participating media and specialized only to scattering from solid surfaces. (Arvo, 1993)

The equation of transfer is an integro-differential equation that describes how the radiance along a beam changes at a point in space. It can be transformed into a pure integral equation that describes the effect of participating media from the infinite number of points along a line. It can be derived in a straightforward manner by subtracting the effects of the scattering processes that reduce energy along a beam (absorption and out-scattering) from the processes that increase energy along it (emission and in-scattering).

If a ray (p, ω) intersects a surface at some point p0 at a distance t along the ray, then the integral equation of transfer is:

$$L_i(p,\omega) = T_r(p0 \to p)L_0(p0,-\omega) + \int_0^t T_r(p' \to p)S(p',-\omega)dt' \qquad (1)$$

where p0 = p + tω is the point on the surface and p' = p + t'ω are points along the ray. This equation describes the two effects that contribute to radiance along the ray. First, reflected radiance back along the ray from the surface is given by the term L_0, which gives the emitted radiance and reflected radiance from the surface. This radiance may be attenuated (T_r) by the participating media; the transmittance from the ray origin to the point p0 accounts for this. The second term accounts for the added radiance along the ray due to volume scattering and emission, but only up to the point where the ray intersects the surface; points beyond the surface don't affect the radiance along the ray.

2.3 Volume Representation

In order to render participating media, its properties must be tracked throughout the spatial extent of the volume in which it resides. The two common means by which this is achieved are the traditional, and popular, use of Eulerian (grid-based) methods (Stam, 2003, Bridson, 2008). As well, Lagrangian (particle-based) methods are used, and they can be further categorized into vortex (Yaeger et. al, 1986)(Gamito et. al. 1995)(Angelidis & Neyret 2005)(Park & Kim 2005) and hydrodynamic (Desbrun 1996)(Miller et. al. 2003)(Premoze et. al, 2003) based methods.

Particle based methods have the advantage of not needing to use computer memory in order to store information about zero density grid cells. For example, consider a large room through which a cloud of smoke is traveling (say from a lit cigarette). Using a uniform grid, it would be necessary to have memory tracking that there is no smoke present in the vast majority of the spatial extent. It is possible that a situation may arise in which it is not possible to use a fine enough uniform grid to resolve the fine details of the smoke without exhausting available memory. In contrast, a particle based system only needs to store the information about the location of the particles that are actually representing the smoke.

It should be noted, that in the case of a uniform volume of participating media, it may not be ideal to use a particle representation. However this scenario rarely arises in practice and in the extreme case of a fully homogenous volume, there exist more efficient methods for rendering such homogenous participating media data sets.

2.4 Ray Marching

Except for cases where participating media is homogeneous and has a uniform isotropic scattering function, the volume rendering function is solved using numerical integration. This integration can be performed using a technique called ray marching (Levoy 1990)(Perlin & Hoffert, 1989)(Ebert et. al. 2002) which takes steps through the participating media and evaluates at each step. The amount of light reaching the camera is the summation of each segment evaluation. Introduced by Bridson (Bridson 2003), ray marching using the sparse block grid is a two level data structure which balances the benefits of spatial subdivision against the costs of the potentially deep hierarchy of an octree. A first top level coarse grid is allocated to encompass the scene extents. Then, if some coarse region is necessary (because there is some data in the region) the second level high resolution grid is allocated. In his thesis Bridson details how the sparse block grid achieves O(n2.25) memory utilization as compared to an octree which is O(n2) and a uniform grid which is O(n3). While this is slightly higher than the octree, the sparse block grid does maintain the O(1) access time characteristic as compared to O(logn) for the octree. Thus, based upon its relatively more efficient memory utilization, the sparse block grid is a popular choice for data storage when using traditional ray marching techniques. However, although using the sparse block grid does ameliorate the memory issue associated with dense grids, traditional ray marching still has its own set of limitations. Notably, interpolations must be used at sample points and further care is required to efficiently distribute samples across the rendering volume, not always possible with particle simulations.

3. THE SYSTEM

Our goal is to create a system that can render volumetric lighting effects in a time efficient, physically plausible manner. To be clear, we say those volumetric lighting effects are:

- Light attenuates as a result of interaction with inhomogeneous participating media
- Solid objects block light thereby forming volumetric shadows

- Support for an arbitrary number of lights
- Support for conventional computer graphics light types, such as point and spot lights

Our rendering system models non-homogeneous participating media using particle data sets. We elect to use particles for modeling participating media so that our simulations are spatially unbounded and grid free. By not using a grid-based system, we significantly reduce our memory requirements. The above requirements, with regard to lighting and particle data sets, present a unique rendering challenge. In order to achieve our lighting objectives, we must solve two density integrations for each particle:

- Find the attenuation of the light reaching the camera from a given particle
- Find the attenuation of the light reaching a particle from a given light source

We make the observation that use of GPU hardware is an excellent solution to Problem 1. Using the alpha blending functionality built into GPU hardware, we can solve the forward density integration per particle for millions of particles at interactive rates. Even including the computation required to achieve the particle depth sorting which is required to obtain correct alpha-blending, the forward density integration process is a relatively computationally inexpensive task.

As a result of using the standard GPU pipeline for our forward density integration, it is possible to integrate other GPU based rendering techniques into our volumetric lighting and rendering system. For example, we implement a pre-existing GPU based technique [35] to include motion blur and depth of field into our system. We explain in greater detail the means by which these effects are achieved in section 3.6.

The rendering process begins with the lighting computation. Presently, the lighting calculation is conducted using the CPU for ease of programmatic implementation. Once computed, the lighting data is added to the vertex buffers in memory which represent the particles. Once the lighting data is added to the vertex buffers, they are sent to the GPU for the final rendering as outlined in Figure 3.

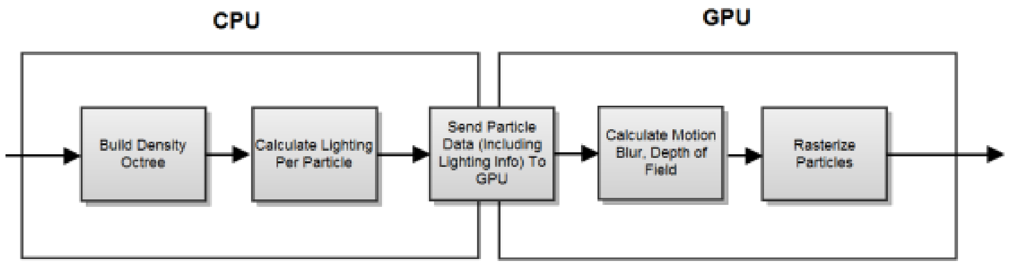

CPU
GPU
Build Density Octree
Calculate Lighting Per Particle
Send Particle Data (Including Lighting Info) To GPU
Calculate Motion Blur, Depth of Field
Rasterize Particles

Figure 3. Particle rendering pipeline

We continue by explaining the lighting process and then outline the process used to create the final image using the GPU.

3.1 Single Scattering Volumetric Lighting Calculation

We determine how much light reaches a particle from a light source by integrating the density along a ray constructed between the particle and the light source. Performing density integrations on particle based participating media is a challenge as there is no immediately available grid representation of the density upon which a ray marching density integration can be performed. Instead, we use a novel octree based data structure which contains the particles as our density integration acceleration structure. We refer to this data structure as the density octree.

3.1.1 Density Octree

In order to accelerate the volumetric lighting calculation, we use a novel octree construction approach in order to reduce the number of ray-particle intersection tests that are required. The density octree is a loose octree (Figure 4) which is created using a subdivide as needed rule (Wilhelms and VanGelder, 1992). Our goal in setting up the subdivision rule is to equally balance the amount of time spent performing tree traversal bounding-box intersection and particle intersection tests. If too deep a hierarchy is created then too much time will be spent in traversal, conversely too shallow a tree results unnecessary particle intersection tests.

The density octree is created by inserting the constituent particles of our data set into an octree. In the standard manner, the particles are placed into the lowest octree node which they fit. However, in order to ensure that particles are concentrated in the leaves, we consider the particles to be a single point in space, as opposed taking their radius into account when determining which node they reside within.

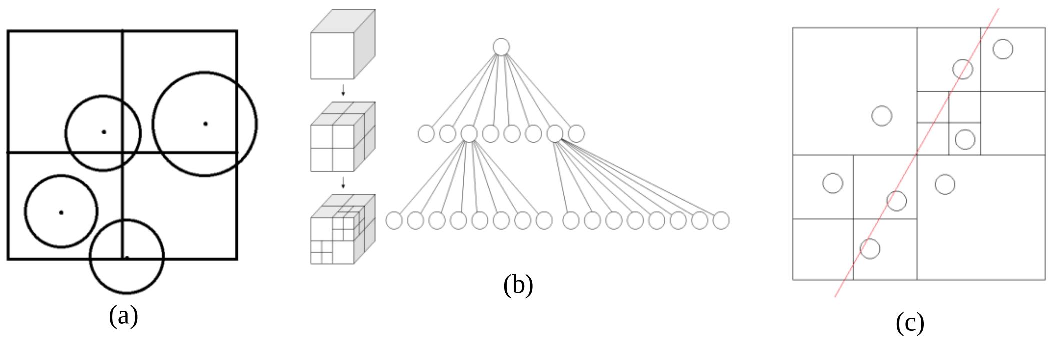


Figure 4. Loose octree allows shapes to overlap cells (b) Octree formed through three subdivision iterations (https://en.wikipedia.org/wiki/File:Octree2.svg) (c) Example of a light ray, shown in red, intersecting a cross section of a density octree

This creates a loose octree in which the stored particles actual radius may overlap the boundaries of a given octree node. We ensure this concentration of particles in the leaves for performance reasons. As it is computationally cheap to traverse the tree compared to performing ray-particle intersection tests, we wish to minimize the number of potential ray-particle intersection tests that will be required.

The density octree begins as a single cell which is then subdivided only when a user selected density is reached, as measured by the number of particles residing in a given cell (See Figure 4). Unlike a conventional grid, the density octree has the desirable characteristic of only requiring memory in those areas of the scene where density information exists in the original particle data set. In the case of an animated sequence, the density octree is only built once per frame in which particles or other scene objects change their position. The same density octree is used for each light. In this way, the small setup time associated with creating the density octree is amortized across the number of lighting passes required.

3.1.2 Lighting Evaluation

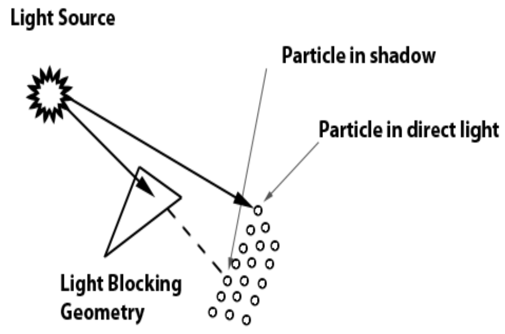
Integration of the density octree is performed via ray casting. A light ray is constructed between some particle and the light source that is currently being evaluated. This light ray is then intersected with the nodes of the octree in a recursive, top-down manner. This analytical ray casting approach achieves computational efficiency in that Ray-Axis Aligned Bounding Box (AABB) tests are only required at the boundaries of the density octree nodes. This is in contrast to ray marching, where it is often the case that multiple steps might be performed within a given uniform grid cell. By performing the intersection tests only as necessary, performance is increased. At each density octree node that the light ray intersects, the light ray is then intersection tested against those particles residing in that node. When the ray intersects a particle, the attenuation of that particle is added to the accumulated attenuation. Finally, once the total attenuation along the light ray has been accumulated, the total density is used to compute the amount of light reaching the particle.

3.2 Intersection Testing Routines

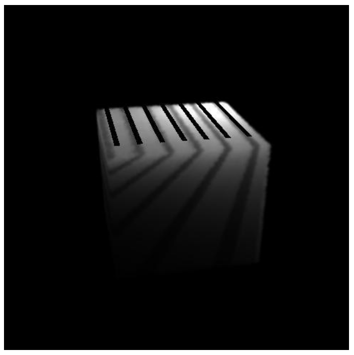
We use the optimized ray-AABB intersection presented by (Williams et al., 1992) We use the following optimized ray-sphere intersection presented in (Akenine-Moller et. al. 2008) Our ray-sphere intersection routine calculates the distance between the closest point on the line segment and the particle center. Using this distance, we are able to apply a scaling to the particle density based upon the extent to which the light ray intersects the particle radius. By having a gradual relationship based upon distance, we eliminate visual glitches that result from otherwise binary intersect/not intersect calculations. This is modeled as a Gaussian distribution.

3.3 Temporal Coherence

In order for our system to be used for computing volumetric lighting solutions for animations we must ensure the results that are coherent between frames. This entails that there is no flickering or visual artifacts between frames of animation. As we are computing an exact solution on a per particle level and scaling our particle density based upon the extent to which a light ray intersects a particle, temporal coherency is an inherent property of our algorithm.



(a)



(b)

Figure 5. Shadows that scene geometry cast onto volumetric media (a) concept (b) actual render.

3.4 Solid Geometry Volumetric Shadowing

So far our discussion has pertained primarily to calculating the interaction of light with participating media. However, we also want to take into account the presence of solid surfaces within the scene. In order to achieve this aim, we build a bounding volume hierarchy spatial data structure on the solid geometry. Our bounding volume hierarchy implementation is based on the code provided by Shirley et al. in his ray tracing book (Shirley et. al. 2003). Prior to conducting a full attenuation calculation using the density octree, the given light ray in question is tested against the solid geometry bounding volume hierarchy. If the light ray intersects a solid surface then no light will be reaching that point and it can be concluded that the illumination for that particle is zero without further computation as Figure 5 illustrates.

3.5 Performance Enhancements

The primary means by which our volumetric lighting algorithm achieves its computational efficiency is the use of the density octree which ensures that only the minimum of required ray/particle intersection test calculations are performed. However, we also implement our solution using early termination when transmission has attenuated to zero on top of a multi-threaded (Reinders, 2007) implementation. Additional user configurable parameters for the system are presented in Table 1.

Table 1. A range of parameters are provided which allow the user to tweak the output of our system.

Parameter	Effect
Shadowing Density	Increasing this value results in light reaching zero intensity in fewer ray-particle interactions.
Solid Geometry Shadow Intensity	This allows the user to set the shadow intensity to values greater than zero in order to achieve a more subtle shadowing effect that is closer in appearance to the results achieved with a multiple scattering solution.
Particle Sub Sampling	Allows the user to select what percentage of the total particle data set to be included in the density octree. The sub-sampling is performed by taking a uniform random distribution sampling of the input data set.
Particle Intersection Test Size	By adjusting this size to be larger, it is more likely that a ray will intersect with the particles resulting in light reaching zero intensity in fewer steps

3.6 Forward Density Integration using GPU Hardware

In order to determine the amount of light reaching the camera along some viewing ray we must evaluate the out-going radiance from each particle along that ray taking into account the cumulative attenuation due to the proceeding particles along the viewing ray. This cumulative attenuation is calculated using the GPU hardware alpha blending functionality. In order to render correctly alpha blended particles using the GPU, we must sort the vertices that represent those particles. Sorting is performed based upon the vertices distance to the camera. Once the particles are sorted they are uploaded to the GPU in order to be rendered. A custom geometry shader is used which expands each single vertex into a set of four vertices which represent a quad facing the camera centered around the original vertex position. By varying the size of the generated quad it is possible to have user selectable particle sizes. Furthermore, because UV coordinates for the quad are generated by geometry shader it is possible to map arbitrary particle color and opacity maps to the quad. Having adjustable particles sizes is convenient in that it allows for smoothing to be performed by using larger than a single pixel, low opacity particles.

We further detail the methods by which we produce two characteristic cinematic effects, motion blur and depth of field, which are essential for compositing particle renderings into special effects shots for film and television. By varying characteristics such as size, opacity falloff towards edges and shape of the quads created in the geometry shader we obtain these effects, as detailed in the following sections. The ease with which our algorithm is added to existing particle rendering solutions is amongst its strengths.

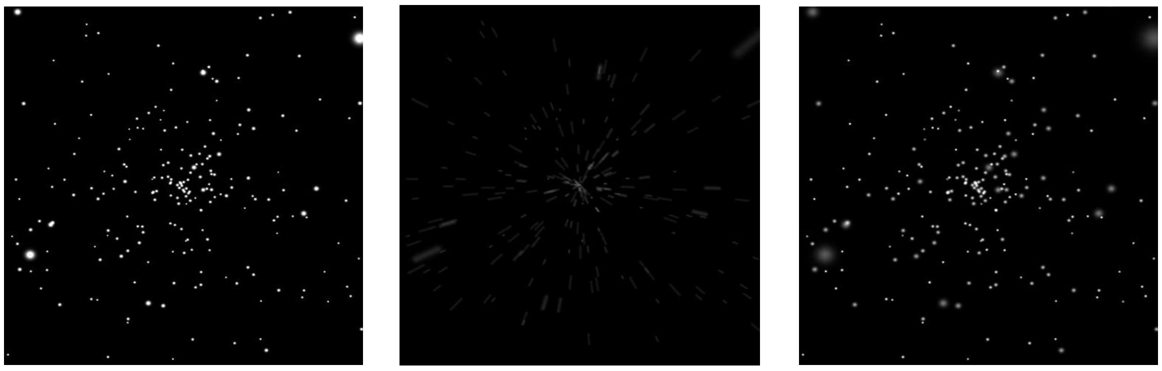


Figure 6. A basic scene being rendered (left), with motion blur (center) and depth of field (right).

3.6.1 Motion Blur

Motion blur results from the rapid movement of the camera or the scene being viewed such that during a single exposure multiple views of the scene occur. Using the geometry shader stage it is possible to efficiently simulate a motion blur effect. This is accomplished by modifying the shape of the quads that are produced. By creating rectangular quads of varying dimensions, where elongation correlates to greater particle speed, a motion blur effect is achieved.

3.6.2 Depth of Field

Depth of field is the distance between the nearest and furthest objects in a scene that appear acceptably in focus in a rendered image. By adjusting the depth of field it is possible to direct the viewers attention to that potion of the scene which is in focus. In a manner similar to that which is used for rendering motion blur, it is possible to use the geometry shader to create a depth of field effect. This is accomplished by varying the size of the quad based upon its distance from the focal point. Additionally, the drop off curve which controls the particle opacity is modified to simulate the effects of depth of field.

4. RESULTS

In order to calculate the lighting for a given particle it is necessary to perform a density octree traversal which is a logarithmic time complexity operation on average. If multiple lights are in the scene then the lighting solution is computed sequentially for each light and the amount of illumination is aggregated. We have found that the results of our performance evaluations are congruent with the theoretical bounds of our system. We assess our new rendering algorithm using a range of quantitative and qualitative metrics. Our quantitative metric is processing time. The purpose of these quantitative experiments is to examine the performance characteristics for the primary rendering time factors; the number of particles, the number of lights and the output resolution.

As our point of comparison for both our quantitative and qualitative evaluations, we have selected Krakatoa (Thinkbox 2014), an industry standard particle rendering system produced by Thinkbox Software. Krakatoa performs lighting calculations using a deep shadow maps based approach. All of our tests have been con-ducted on the same hardware consisting of a standard desktop personal computer with a single Intel i7 quad core hyperthreaded processor running at 2.67GHz and 12GB of RAM. Our system uses the GPU for final display, but like Krakatoa, the CPU is used for the lighting calculations making our comparisons relevant and fair.

4.1 Quantitative Performance

Our first performance evaluation explores raw particle throughput in terms of how much time is required to light a given number of particles and for a number of lights with a consistent number of particles. We observe that our system is characterized by a logarithmic time increase with respect to the number of particles. This is expected, as our density evaluations are performed on an octree based data structure.

Both our system and Krakatoa exhibit a linear performance relationship between the number of lights present in the scene and the time required to compute a lighting solution. These results are clear in Figure 7.

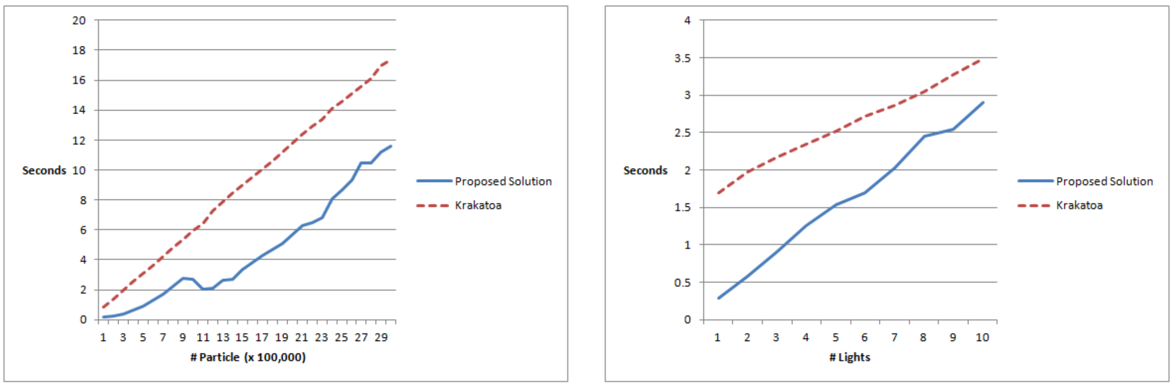
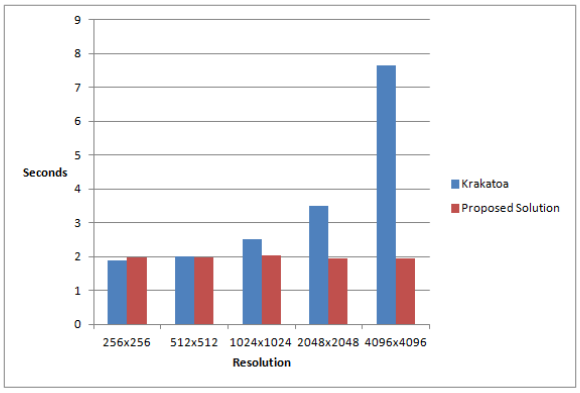


Figure 7. Time to calculate (a) a lighting solution for a single light given some number of particles. (b) a lighting solution for a given number of lights on a scene of 250,000 particles.



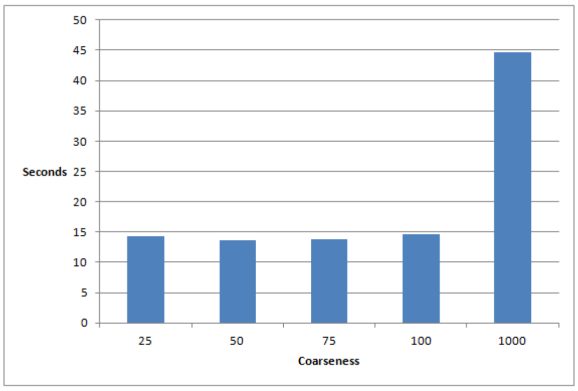


Figure 8. Time required for (a) rendering a given output frame size for a scene consisting of 250,000 particles and one light. (b) to compute lighting for a single light in a scene of 1,000,000 particles for various density tree coarseness settings.

Our volumetric lighting, motion blur and depth of field calculations are independent of output image size, as shown in Figure 8. This is because all our lighting computation is performed using the output resolution independent density octree. As such, no additional computation time is required for higher resolution output. In contrast, we observe that Krakatoa appears to exhibit a quadratic increase in time required to render as output frame size increases. Given that the performance of our algorithm is independent of resolution, it is a good choice in situations where modern high resolution frames are required.

As our system provides better results with fewer particles, we demonstrate in Table 3 that we also provide a tangible improvement in speed While a rendering at 100% of particles using the proposed solution provides a marked improvement in speed, there is also a significant qualitative improvement. This allows our system to be able to use fewer particles for the same qualitative output compared to Krakatoa. In essence we could use 50% of the particles in the proposed system and get similar qualitative results to using 100% of the particles yet gain a threefold improvement in processing time.

Table 3. Performance results for various percentages of total particles included in density tree. Krakatoa numbers were obtained by reducing the number of particles in the source data set.

# particles	Proposed Solution	Krakatoa
100%	10.4 s	17.8 s
50%	5.5 s	9.0 s
10%	3.5 s	2.2 s

4.2 Qualitative Performance

Although raw performance statistics are an important metric of the utility of a rendering system, ultimately the final criterion is the quality of the output imagery. To this end, we examine several real world scenes and compare the output of our system against Krakatoa. The smoke plume data set (rendered in Figure 9) consists of 350,000 particles. Note how Krakatoa is unable to produce a smooth lighting solution with the number of particles provided. In contrast, our system, which calculates the lighting at each particle and is able to scale particles sizes while maintaining a correct lighting solution, is able to produce a smooth lighting solution. Using our algorithm, it is possible to achieve higher quality results using fewer particles. In practice, requiring fewer particles to achieve acceptable visual quality means that our system out performs Krakatoa on typical rendering tasks quantitatively. Although a full evaluation with human subjects could not be presented, we believe the proposed system is qualitatively better as well. (See Figure 9).

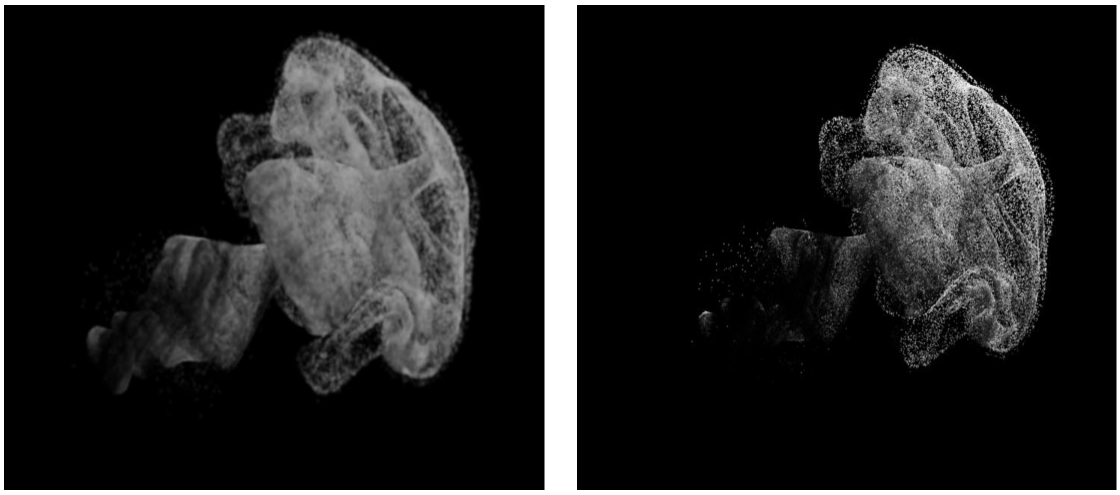


Figure 9. Smoke plume data set (350,000 particles) rendered with proposed (left) and Krakatoa (right).

Additionally, our system is able to calculate the shadows that would be cast by participating media onto arbitrary scene geometry, as shown in Figure 12. The shadow map is output as a separate floating point image buffer which can be composited into original frame buffer using standard image compositing tools.

4.3 Manipulations to allow Better Qualitative/Quantitative Performance

As the results show, our rendering system, based around our novel particle based participating lighting algorithm, is able to achieve high quality output in an efficient and time competitive manner as compared to a current industry particle rendering tool. As explained in the introduction of this section, in the default configuration, our system is logarithmic in time complexity with regard to the number of particles. However, based upon our observation that it is unnecessary to use the entire data set in creating the density octree we can continuously reduce the amount of particles included in a subsampled set such that our run time can be adjusted so that it is closer to linear in characteristic (by sub sampling particles by factor of log N)/

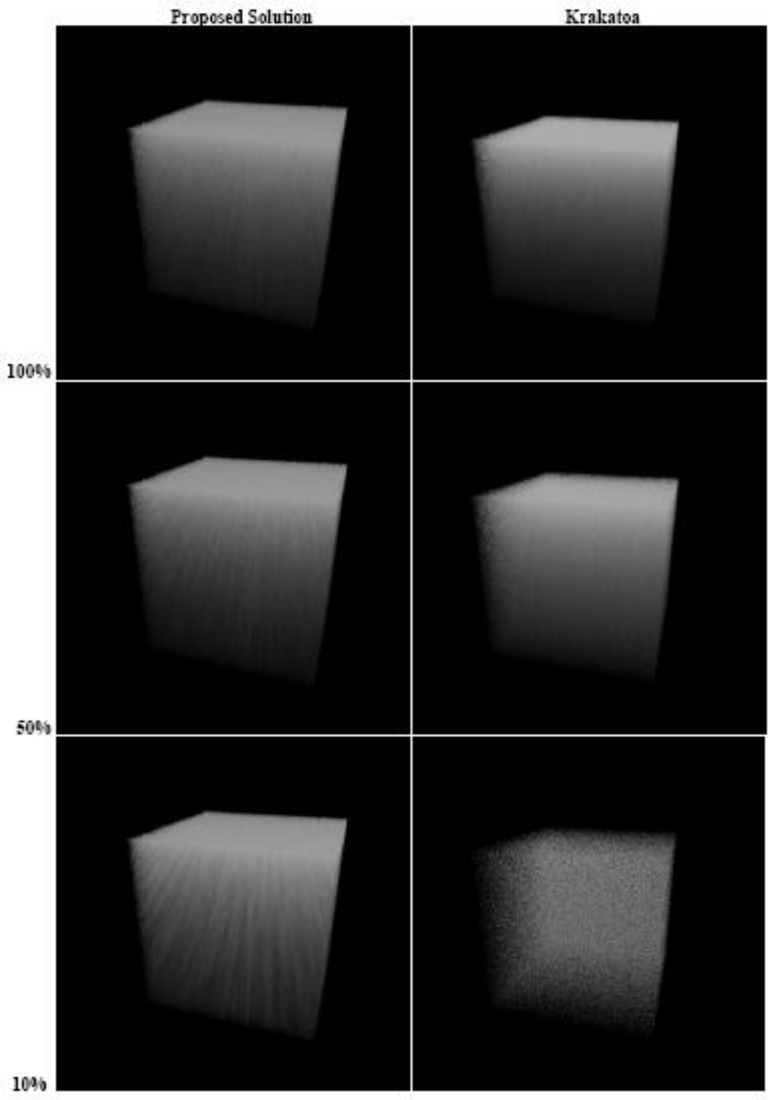


Figure 10. Effects of changing percentage of total particles included in the density tree for a 3,000,000 particle data set (left: proposed system, right: Krakatoa). Runtimes for renders are outlined in Table 3.

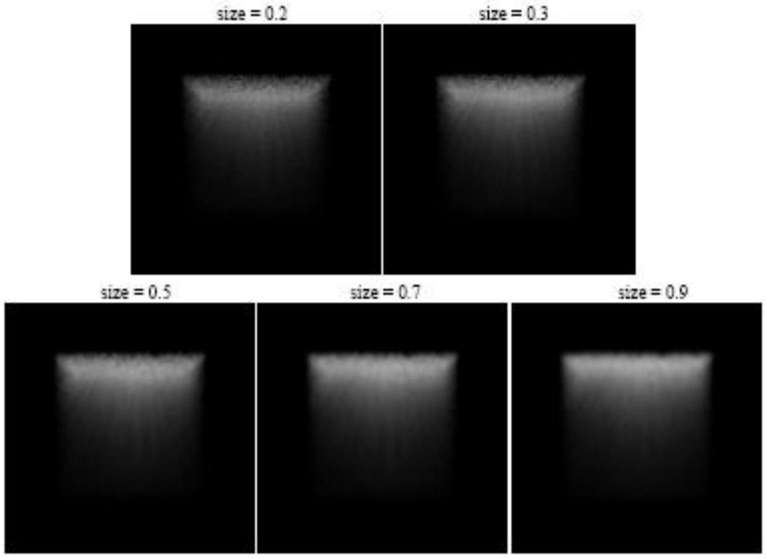


Figure 11. Effects of increasing output particle size on the same lighting solution. (render time is constant)

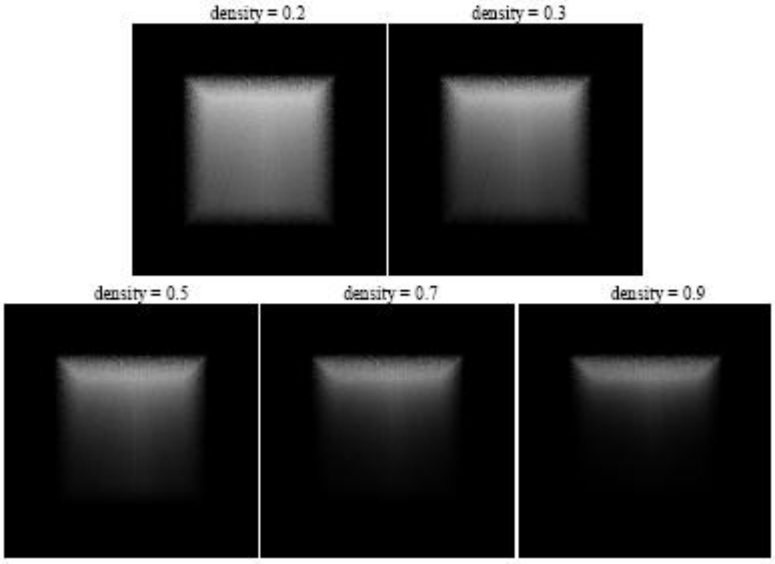


Figure 12. Effects of increasing the particle density. In all cases, render time remains constant.

5. FUTURE WORK

The following sections present a selection of areas which we identify as future avenues of development to continue improving our existing system.

5.1 Level of Detail

In regions further from the camera, it is not necessary that the lighting calculations be as accurate since the viewers will not be able to as readily see detail in those areas. Using our algorithm it would be possible to automatically adjust the level of detail by making the percentage of total particles used in ray-particle checks a function of the distance to the camera. For example, consider some density octree node located near the camera which contains one hundred particles. In this case, since the density octree node is located close to the camera we would want the most accurate lighting calculation possible, so all one hundred particles would be ray-sphere tested. However, consider some density octree node which is a significant distance away from the camera. In this case, we might only perform ray-sphere tests for a representative sub-portion of the particles in the density octree node. In this way, full computation would only performed in those regions which it is required.

5.2 Multiple Scattering

At present, our system only calculates the single scattering component of volumetric lighting. A further extension would be to develop a method by which it would be possible to efficiently utilize the spatial density information provided by the density octree in order efficiently calculate the multiple scattering component of volumetric lighting.

5.3 Ambient Occlusion

Given the spatial locality information provided by the density octree it should be possible to create a fast ambient occlusion calculation process. This would be useful for creating various artistic effects as well as fast approximations of multiple scattering when a fully calculated result is not required.

5.4 Particle Replication

One of the issues when working with particle based simulations is that it often takes more time to calculate the particle dynamics than it does to render those particles. As a result, rendering systems can be underutilized because it is not possible to simulate enough particles to fully load the particle rendering systems. In order to alleviate this problem, it would desirable that a particle rendering system could take a base set of particles which represent the structure, movement and other characteristics of the original simulation and then add additional particles at render time which further enhance the final output.

6. CONCLUSION

As the results show, our rendering system, based around our novel particle based participating lighting algorithm, is able to achieve high quality output in an efficient and time competitive manner as compared to the current industry standard particle rendering tool. In the default configuration our system is logarithmic in time complexity with regard to the number of particles. However, based upon our observation that it is unnecessary to use the entire data set in creating the density octree we can continuously reduce the amount of particles included in a sub sampled set such that our run time can be adjusted so that it is closer to linear in characteristic (this can be done by sub sampling particles by factor of log N)

In our qualitative evaluations, we found that our system produces higher quality outputs with less particles than Krakatoa. As a result of requiring fewer particles for the same output quality, our system achieves comparable or better performance in practical usage scenarios. Requiring less input particles is a major advantage in that it reduces the amount of disk space required to store the data sets. Perhaps even more importantly, because less particles are required for rendering, it is not necessary to simulate as many particles at the outset of the production process. As this simulation step is routinely amongst the most expensive in terms of computation time, our system with its lower particle requirements, is an attractive option for production rendering pipelines.

The effect of sub-sampling was examined. Even with only 10% of the original data set being used in the density octree, our system still provides meaningful, albeit coarse information about the lighting in the scene. Krakatoa on the other hand fails to provide much meaningful lighting data once the input data set is reduced. This makes our system viable for quick visualization and allows artists many more development iterations. By varying characteristics such as size, opacity falloff towards edges and shape of the quads created in the geometry shader we obtain effects such as motion blur and depth of field. The ease with which our algorithm is added to existing particle rendering solutions is amongst its strengths.

Furthermore, our system seamlessly integrates the calculation of volumetric particle shadows and the shadows that are cast by participating media onto any arbitrary scene geometry. As our lighting calculation is independent of output resolution, there is no additional expense for higher resolution output images. The user parameters of system which control the particle size and density can also be adjusted without affecting computation time. Although not real-time, the reduction of particles that maintain realistic rendering of participating media suggests that an optimized implementation may make the proposed system useful for next generation gaming platforms.

REFERENCES

Akenine-Moller, T, et al. Real-Time Rendering 3rd Edition. A. K. Peters, Ltd., Natick, MA, USA. (2008).

Angelidis, A And Neyret, F. "Simulation of smoke based on vortex filament primitives." In Proc. of the ACM SIGGRAPH-Eurographics symposium on Computer animation, pages 87-96. ACM, New York, NY, USA. (2005).

Arvo, J. Transfer Equations in Global Illumination (1993).

Bridson, R. Fluid simulation for computer graphics. A K Peters, Wellesley, Mass. ISBN 9781568813264 (2008).

Bridson, R, S. U. P. In Scientific Computing, And C. Mathematics. Computational aspects of dynamic surfaces. Stanford University (2003).

Chandrasekhar, S. Radiative Transfer. New York: Dover Publications. (1960).

Desbrun, M And Paule Gascuel, M. “Smoothed particles: A new paradigm for animating highly deformable bodies." In Computer Animation and Simulation 96 (Proceedings of EG Workshop on Animation and Simulation, pages 61-76. Springer-Verlag (1996).

Ebert, D, Musgrave, F, Peachey, D, Perlin, K, And Worley, S. Texturing and Modeling: A Procedural Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edi-tion. ISBN 1558608486 (2002).

Gamito, M.N., Lopes, P.F., and Gomes, M.R. "Two-dimensional simulation of gaseous phenomena using vortex particles." In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation," pages 3-15. Springer-Verlag (1995).

Knoll, A. “Ray traversal of octree point clouds on the gpu." In “Interactive RayTracing”, 2008. RT 2008. pg. 182 (2008).

Levoy, M. "Efficient ray tracing of volume data." ACM Trans. Graph. 9, 245-261. ISSN 0730-0301 (1990).

Madill, H.E., Houston, B. "Fury renderer." Exocortex Tech-nologies, Inc. Ottawa, Canada. (2010).

Miller, M., Charypar, D., And Gross, M. "Particle-based uid simulation for interactive applications." (2003).

Park, S.I. And Kim, M.J.. "Vortex fluid for gaseous phenomena." In Proceedings of the 2005 ACM SIGGRAPH-Eurographics symposium on Computer animation," SCA '05, pages 261-270. ACM, New York, NY, USA. ISBN 1-59593-198-8 (2005).

Perlin, K And Hoffert, E. "Hypertexture." SIGGRAPH Comput. Graph. 23, 253-262. ISSN 0097-8930 (1989).

Pharr, M And Humphreys, G. Physically Based Rendering: From Theory to Implementation (The Interactive 3d Technology Series). Morgan Kaufmann. ISBN 012553180X (2004).

Premoze, S, Tasdizen, T, Bigler, J, Lefohn, A, And Whitaker, R.T.. "Particle based simulation of fluids." (2003).

Reinders. J. Intel threading building blocks. O'Reilly & Associ-ates, Inc., Sebastopol, CA, USA, first edition. ISBN 9780596514808 (2007).

Shirley, P. And Morley, R. Realistic ray tracing. Ak Peters Se-ries. AK Peters. ISBN 9781568811987 (2003).

Stam, J. "Real-time fluid dynamics for games." (2003).

Thinkbox Software Krakatoa http://www.thinkboxsoftware.com/krakatoa/. Accesssed March 2014.

Wilhelms, J And Van Gelder, A. "Octrees for faster isosurface generation." ACM Trans. Graph. 11, 201-227. ISSN 0730-0301 (1992).

Williams, A, Barrus, S, Morley, R, And Shirley, P. "An effi-cient and robust ray-box intersection algorithm." Journal of graphics, gpu, and game tools 10(1), 49-54 (2005).

Yaeger, L., Upson, C, And Myers, R. "Combining physical and visual simulation creation of the planet Jupiter for the film 2010." In Annual Conference on Computer Graphics, volume 20, pages 85-93 (1986).

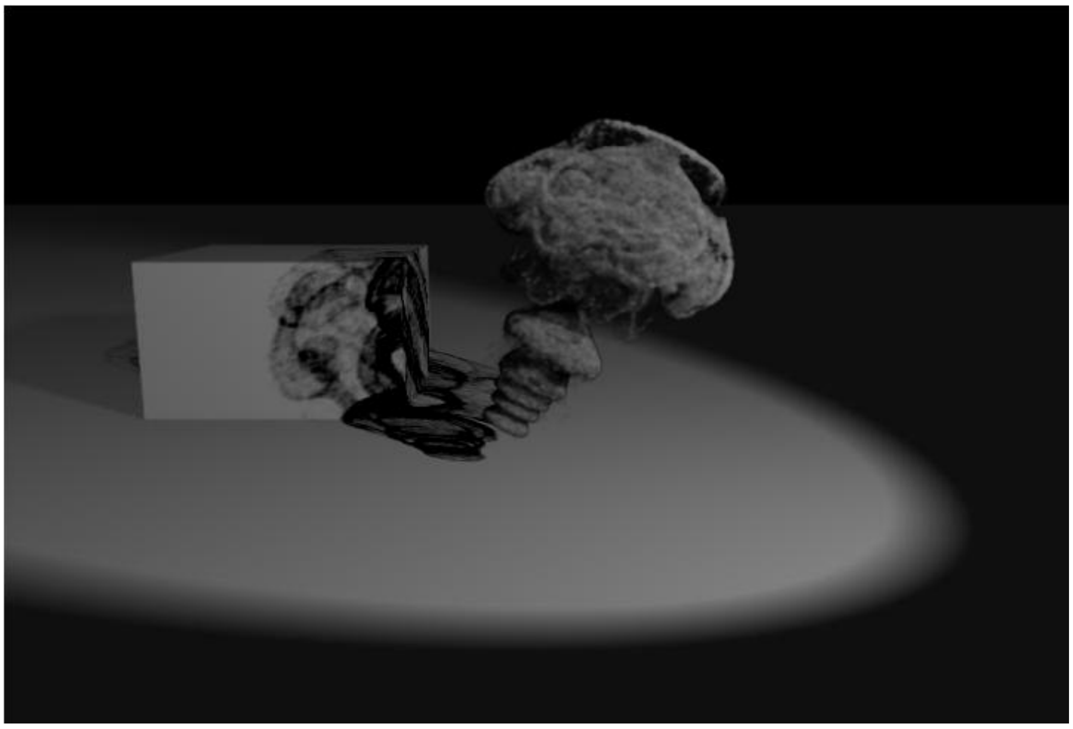


Figure 13. High resolution render exhibiting volumetric media casting shadows no solid scene geometry.