A Coherent Grid Traversal Approach to Visualizing Particle-Based Simulation Data

Christiaan P. Gribble, Thiago Ize, Andrew Kensler, Ingo Wald, and Steven G. Parker

Abstract— **We present an approach to visualizing particle-based simulation data using interactive ray tracing, and describe an algorithmic enhancement that exploits the properties of these datasets to provide highly interactive performance and reduced storage requirements. This algorithm for fast packet-based ray tracing of multi-level grids enables interactive visualization of large, time-varying datasets with millions of particles and incorporates advanced features like soft shadows. We compare the performance of our approach with two recent particle visualization systems: one based on an optimized single ray grid traversal algorithm, the other on programmable graphics hardware. This comparison demonstrates that the new algorithm offers an attractive alternative for interactive particle visualization.**

Index Terms— **Particle visualization, interactive ray tracing, coherent grid traversal, large and time-varying particle datasets**

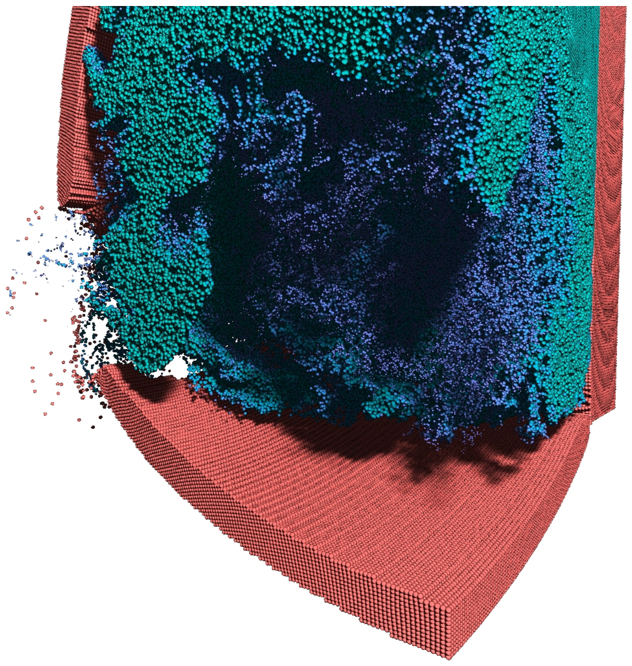


Fig. 1. *Visualizing particle-based simulation data with efficient ray tracing*. We describe a new algorithm based on coherent grid traversal that efficiently renders large, time-varying particle-based simulation data at highly interactive rates. The performance of this approach compares favorably with systems that represent the current state-of-the-art in particle visualization.

I. Introduction

PARTICLE methods are commonly used to simulate complex phenomena in many scientific domains, including astronomy, biology, chemistry, and physics. Using these techniques, computational scientists model such phenomena as a system of discrete particles that obey physical laws and possess certain properties. Particle-based simulation methods are particularly attractive for problems with high deformations or complex geometries, and are used to solve time-dependent problems on scales from the atomic to the cosmological. Frequently, millions of particles are required to capture the behavior of a system accurately, leading to large, complex datasets such as those depicted in Figs. 1 and 2. An effective visualization method will communicate subtle changes in the three-dimensional structure, spatial organization, and qualitative trends within the simulation as it evolves, as well as enable easier navigation and exploration of the data through interactivity.

Interactive visualization of particle datasets typically serves one of three purposes: data analysis, code development, or generation of publication quality images. First, an interactive visualization process enables users to identify and explore the salient features of their data more effectively. Second, the ability to debug ill-behaved solutions is an obvious, but important, consequence of highly accessible interactive visualization. Finally, an interactive environment allows a user to quickly identify optimal views in which each image or frame of an animation conveys the most pertinent information.

Unfortunately, the size and complexity of typical particle datasets make interactive visualization a difficult task. Particle values can be projected to a three-dimensional grid, and the transformed data can then be visualized using standard techniques such as direct volume rendering [1] or isosurface rendering [2]. Grid-based representations of the data are suitable for some, but not all, particle visualization tasks. For example, the need to simultaneously visualize both the large- and small-scale features within the data often make grid-based representations problematic. Additionally, interpolation may hide features or problems present in the original particle data [3], while isosurface extraction can be very time-consuming, particularly for large datasets.

Particle data can also be represented directly by simple iconic shapes called glyphs. For many applications, a sphere or an ellipsoid is a natural representation of an individual particle. Using graphics hardware, particle data can be visualized directly by rendering either highly tessellated spheres or high quality spherical impostors (textured billboards). Unfortunately, tessellating millions of particles often results in too many triangles to be rendered at interactive rates, and implementing advanced visualization features such as soft shadows with impostor-based geometry is non-trivial.

In this paper, we investigate the use of interactive ray tracing for visualizing large, time-varying particle-based simulation datasets. We present an efficient algorithm using fast packet-based ray tracing and multi-level grids. Currently, there are

The first author is with the Department of Computer Science at Grove City College. Mailing address: 100 Campus Drive, Grove City, PA 16127. E-mail address: cpgribble@gcc.edu. The remaining authors are with the Scientific Computing and Imaging Institute in the School of Computing, University of Utah. Mailing address: 50 S Central Campus Drive, MEB 3490, Salt Lake City, UT 84112. E-mail addresses: {thiago|aek|ingo|sparker}@cs.utah.edu.

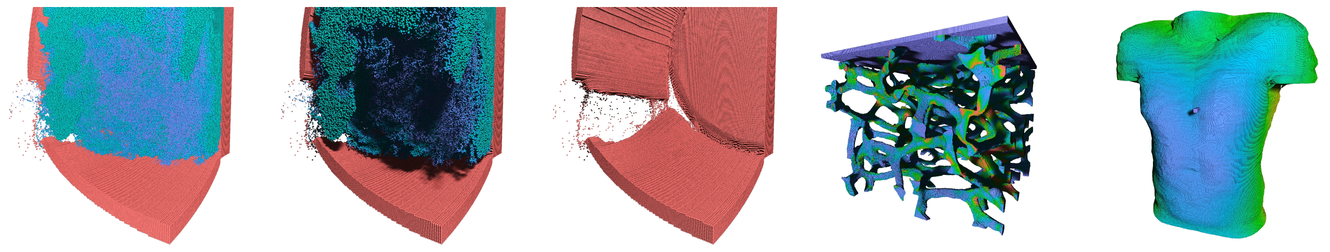


Fig. 2. *Advanced visualization features for particle datasets*. Our approach renders datasets with millions of particles at highly interactive rates. The system also provides run time control of several advanced visualization features, including color mapping, soft shadows, and parameter range culling.

three acceleration structures that support packet-based ray tracing: kd-trees [4], bounding volume hierarchies (BVHs) [5], and multi-level grids [6]. We explore the latter for three reasons: First, the radii of particles within these datasets are often uniform in size or fall within a well-defined range, and grids typically perform well for such uniformly sized primitives. Second, the particles are typically either uniformly distributed throughout the environment or densely packed with large regions of empty space between them. While hierarchical data structures like kd-trees or BVHs often provide superior performance for scenes with varying primitive density, a grid can skip completely empty space as fast as these structures using a macrocell hierarchy [7]. In addition, grids can be advantageous for densely packed regions and often provide the best performance. Third, parallel grid construction algorithms [8] enable these structures to be built on-the-fly for time-varying datasets, and thus offer the potential for computational steering within integrated problem solving environments. In this paper, we introduce optimizations that exploit the properties of particle-based simulation data to tailor the coherent grid traversal algorithm [6] for particle datasets, achieving both improved performance and reduced storage requirements. While we demonstrate the effectiveness of our approach using the results of material point method [9], [10] simulations of structural mechanics problems, our approach is applicable to particle data from other simulation methods and other application domains as well.

II. Background and Related Work

Our approach builds upon several existing techniques from various fields, which we briefly review below.

A. Interactive Particle Visualization

Interactivity encompasses a wide range of activities in the context of particle visualization. For example, interactive viewing and lighting enable investigators to identify and interrogate specific features within the data more easily. Interactivity also provides important visual cues from relative motion [11], [12] and environmental frame of reference [13], while advanced features such as parameter range culling (discussed in Section III-D) and color mapping [14] provide opportunities for additional insights. Using the algorithm we describe in Section III, each of these activities is under the full control of a user at run time and can be changed at interactive rates.

In addition to these interaction features, important perceptual cues from non-local shading effects are easily integrated into our algorithm because it is based on ray tracing. For example, shadows are a well-studied visual cue that provide important information about shape and relative position [13], [15]. We use soft shadows from area light sources because they typically exhibit a smooth transition from shadowed to unshadowed regions, are less likely to be misinterpreted, and provide additional visual cues about the relative position of objects in complex datasets [16], [17]. Using our approach, investigators are able to interactively control both the size and position of the light source, as well as the shadow quality. Recent research has shown that visual cues from advanced shading models such as physically based diffuse interreflection can also be beneficial in the context of particle visualization [18]. Although advanced shading models have not yet been implemented in this system, they are, in principle, easily integrated as well.

Finally, parallel grid construction algorithms [8] accommodate the time-varying nature of particle datasets in a straightforward manner. These datasets are quite large, containing many millions of particles across tens or hundreds of time steps. In our approach, grids for time-varying datasets are constructed on-the-fly, so the user can easily cycle through all of the time steps at run time. This process accommodates the changing structure of the data as the simulation evolves, enabling interaction with millions of particles across the entire simulation. Moreover, because scientists can interact with the whole dataset, a clear understanding of the physical state of each particle, as well as its relationship to the full computational domain, can be achieved.

Many efforts have explored techniques to render large numbers of spheres efficiently, from rasterization on massively parallel processors [19], visualization clusters [20], custom hardware [21], [22], and programmable graphics hardware [23] to interactive ray tracing on tightly coupled supercomputers [24]. Additional aspects of particle visualization, including silhouette enhancement and advanced shading models [18], [24], [25], have also been investigated.

Two recent systems represent the current state-of-the-art in interactive particle visualization. On the one hand, interactive ray tracing on tightly-coupled supercomputing platforms is used to visualize large, time-varying particle datasets at interactive rates [24]. Unfortunately, the hardware costs of such a system are often prohibitive and impede accessibility. At the other extreme, programmable graphics hardware brings interactive particle visualization to the desktop [23]. Though such hardware is widely accessible, interactive performance is constrained by the fill rates of current GPUs, which limits interaction to datasets with at most a few hundred thousand particles. In addition, advanced visualization features such as soft shadows are difficult to implement with impostor-based rendering. In Section III, we present an efficient algorithm for ray tracing large, time-varying particle datasets at interactive rates. This approach not only satisfies the requirements of an effective particle visualization method as outlined in Section I, but it is more accessible than previous systems that require expensive hardware and easily incorporates advanced features that are difficult to implement using current graphics hardware.

B. Interactive Ray Tracing

This work also builds upon recent research concerning interactive ray tracing, specifically efficient acceleration structures for packet-based ray tracing. We extend the coherent grid traversal (CGT) algorithm [6] to efficiently visualize large numbers of particles represented by spherical glyphs. We combine elements of offset surfaces [26], which are commonly used for collision detection in a wide variety interactive applications, with the frustum based ray traversal algorithm by exploiting the properties of particle-based simulation datasets.

In CGT, ray packets are traversed through the grid by considering vertical slices rather than individual cells. Multiple cells in each slice are traversed by all of the rays in a packet, and each ray is tested against all of the objects within a given cell. Although this approach implies that some rays will traverse cells they would not have otherwise considered, the packet is traced as a coherent whole in each step, and no splitting or merging operations are required (Fig. 3).

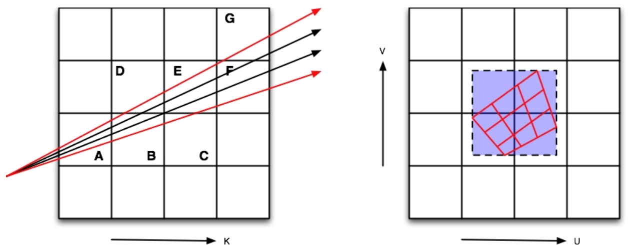


Fig. 3. *Packet traversal in a grid.* In coherent grid traversal, rays step along vertical slices in the major traversal direction. Rays traverse the grid as a coherent whole, so no splitting or merging operations are required (left). The bounding frustum overlaps the same cells as the individual rays, and this frustum can be used to guide ray traversal (right).

III. Coherent Grid Traversal for Particle Data

Our approach to particle visualization is inspired by the CGT algorithm described above. We use packet-based ray tracing and multi-level grids to achieve interactive performance for large, time-varying particle datasets. Frustum based traversal achieves lower per-ray cost by amortizing traversal operations over multiple rays in a packet, and the algorithm is well-suited to SIMD implementation. Further, advanced visualization features such as soft shadows are integrated easily because they can be implemented naturally in a ray tracing framework. We thus extend the original CGT algorithm to efficiently visualize large, time-varying particle datasets by exploiting the properties of particle-based simulation data to improve performance and reduce storage requirements.

A. The Sphere-Center Method

The CGT algorithm can be optimized for glyph-based particle visualization by leveraging the fact that all primitives are spheres. In particular, several observations permit optimizations beyond those employed by the original CGT algorithm. First, a sphere S with center C and radius r is symmetric, so determining whether S overlaps a frustum F is analogous to testing whether C is in the r-neighborhood of F. Second, testing whether the distance from C to the planes of F is less than r is the same as testing whether C is inside another frustum F_r that has been enlarged by r. Thus, if we traverse the grid using an enlarged frustum, we only need to intersect those spheres whose centers lie inside that frustum, and therefore only have to store each sphere at exactly one location: the cell in which its center is located. We call this approach the *sphere-center* method.

Using the enlarged frustum F_r for traversal requires a priori knowledge of r, a value that (potentially) varies with each sphere. However, for our application, the radii are either uniform or lie within some small range, so the maximum radius r_{max} across all particles can be used to generate the enlarged frustum F_r. Using this value, the distance each plane must be shifted is given by:

$$s = r_{max}\sqrt{1+dU^2}$$

where dU is direction vector of the given plane's normal (Fig. 4). Thus, the enlarged frustum can be computed in just five SIMD operations: three additions, two multiplies, and a single square root. The near and far planes of the frustum must be shifted by r_{max} as well, and the early ray termination criteria must be adjusted to accommodate potentially intersecting spheres whose centers lie in the next slice.

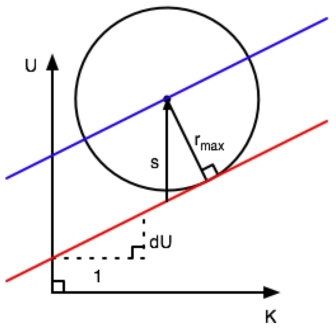


Fig. 4. *Shifting the bounding planes.* We observe that testing whether a sphere of radius r intersects a bounding frustum (red) is equivalent to testing whether a frustum enlarged by r_{max} (blue) contains the center of the sphere.

In a traditional grid, primitives often overlap multiple cells, which results in redundant computation and storage, as well as additional levels of indirection. A mailbox structure [27] can be used to avoid redundant intersection computation, but requires additional storage and computation and may not be effective for primitives with inexpensive ray intersection routines such as spheres or triangles. While reordering techniques [28] can be used to minimize the overhead introduced by pointer indirection, the cost of this process means that reordering is not well-suited to acceleration structures that are built on-the-fly during rendering.

The sphere-center method avoids these problems. The sphere-center method obviates the need for mailboxes: spheres are stored in exactly one grid cell, and will be intersected no more than once during traversal. In addition, data duplication is never required because the primitives are stored directly in the grid, and the center of each sphere is guaranteed to lie in exactly one cell. As a consequence, locality of reference for spatially local primitives is improved without an explicit sorting or data reorganization process.

The sphere-center method is motivated by observations similar to those motivating so-called offset surfaces, which are often used for collision detection in interactive applications such as video games [26]. In this case, the collision of an object with its environment can be computed quickly by offsetting the geometry of the environment according to the dimensions of the object. The object is then represented by a point, and a line captures the motion of the object through the environment. A simple line segment check can then be used to detect a collision between the object and the environment. In a similar manner, the sphere-center method permits a simplified representation of the objects (in our case, spheres) by offsetting the frustum that guides ray traversal through the grid. However, the sphere-center method is used as a ray tracing acceleration technique; our algorithm combines these observations with fast, packet-based grid traversal to reduce memory requirements and improve interactive performance.

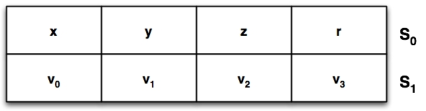
B. Grid Organization

The organization and design of our multi-level grid follows that of a typical hierarchical structure. Primitives are stored at the finest level of the grid, the resolution of which is determined such that the number of cells is a multiple of the total number of particles N, denoted by λ. Cubically shaped cells minimize surface area with respect to volume, and thus reduce the expected cost of traversal, so the resolution of the grid is given by:

$$N_x = d_x \sqrt[3]{\frac{\lambda N}{V}}, N_y = d_y \sqrt[3]{\frac{\lambda N}{V}}, N_z = d_z \sqrt[3]{\frac{\lambda N}{V}},$$

where $\vec{d}$ is the diagonal and V the volume of the grid.

Once the grid resolution has been determined, the data associated with each particle are inserted directly into the appropriate grid cells. To facilitate efficient insertion, culling, and ray/sphere intersection tests, these values are stored in two consecutive 16-byte aligned SIMD variables (Fig. 5). The position and radius (x, y, z, and r) are stored in the first SIMD variable, while up to four scalar properties from the simulation (v_0, v_1, v_2, and v_3) are stored in the second. As will be discussed in Section III-D, these values allow investigators to isolate particles with properties falling within some range of interest, a technique we call parameter range culling.



x	y	z	r	S_0
v_0	v_1	v_2	v_3	S_1

Fig. 5. *Data layout for the new CGT algorithm.* The data associated with each sphere are stored in two consecutive SIMD variables, which facilitates efficient insertion, culling, and ray/sphere intersection tests.

To facilitate a more efficient traversal, the grid is organized hierarchically. Hierarchical grids typically divide densely populated regions of space more finely than empty regions. There are several ways to accomplish this task [7], [29]–[31], and we leverage the macrocell hierarchy described by Parker et al. [7]. Each level in this hierarchy imposes a coarser grid over the previous level, and each macrocell corresponds to an $M \times M \times M$ block of cells in the underlying level. In the current implementation, we use a simple two-level hierarchy: one level of macrocells imposed on top of the actual grid.

To support run time parameter range culling, the macrocell hierarchy differs from the one used in the original CGT algorithm and more closely resembles one used in interactive volume visualization applications [7]. In particular, a macrocell must store the minimum and maximum values of each parameter across all of the particles it contains, rather than a Boolean flag indicating indicating an empty or non-empty condition.

We use the sort-middle construction algorithm described by Ize et al. [8] to quickly rebuild the grid in each frame for time-varying datasets. In this approach, the construction-related tasks corresponding to disjoint sections of the grid are statically distributed among all of the threads in the system. The sort-middle insertion essentially performs a coarse parallel bucket sort of the particles by their cell locations, and each thread inserts the particles in its set of buckets into the appropriate grid cells. The regions corresponding to each bucket are disjoint, so each thread inserts its particles into different parts of the grid. Write conflicts are thus avoided, and mutexes or other synchronization primitives are not necessary.

C. SIMD Sphere/Frustum Culling

Mailboxes and fast SIMD frustum culling are critical components of the original CGT algorithm. These operations are typically much faster than packet/primitive intersection tests, the cost of which is linear in the number of rays in each packet. While the sphere-center method described in Section III-A prevents redundant intersection tests and alleviates the need for mailboxes, we use frustum culling to prevent unnecessary ray/sphere intersection tests and further improve performance.

The two sources of potentially unnecessary intersection tests with our approach are illustrated in Fig. 6. First, a sphere may lie within a cell through which the enlarged frustum passes, but the sphere does not overlap either the enlarged or the original frustum. Second, the radius of a given sphere may be smaller than the maximum radius r_{max} used to compute the enlarged frustum, again implying a potential intersection when there is actually no overlap. We use frustum culling to efficiently reject non-overlapping spheres.

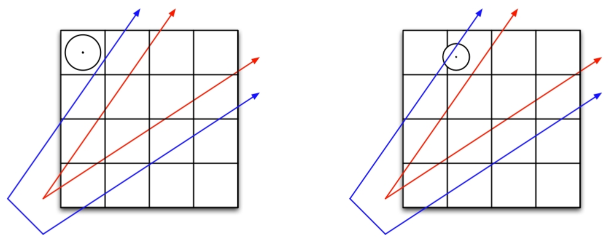


Fig. 6. *Avoiding unnecessary ray/sphere intersection tests.* Even if a sphere lies within a cell through which the enlarged frustum passes, the sphere may not overlap either the original or the enlarged frustum (left). Additionally, a sphere whose radius is less than r_{max} may overlap the enlarged frustum, but not the original one (right). SIMD sphere/frustum culling detects these situations and discards the spheres.

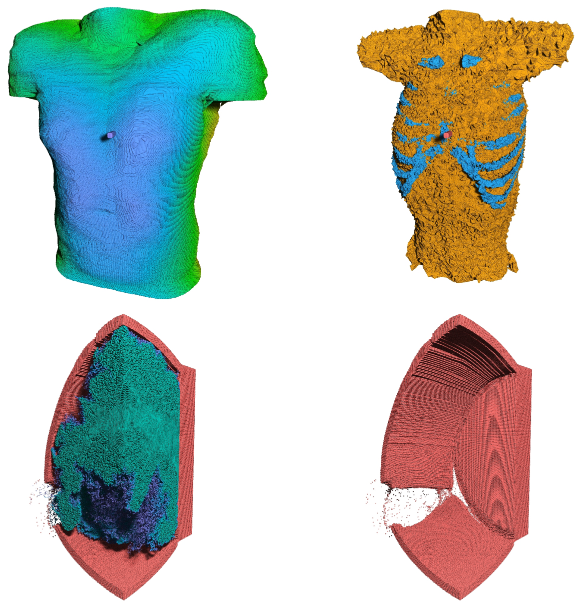


Fig. 7. *Run time parameter range culling.* Using parameter range culling, particles representing the bone and internal tissues within the BulletTorso dataset (top) and only those representing the alloy container in the Thunder dataset (bottom) have been isolated. Parameter range culling puts the range of valid parameter values used during visualization under the full control of the user at run time, and these values can be changed interactively.

The original CGT algorithm employs SIMD shaft culling [32] to prevent unnecessary intersection tests by quickly discarding triangles that lie outside the current bounding frustum, but this technique works only for primitives that possess planar edges. Fortunately, a much simpler test can be used to quickly cull spheres and avoid unnecessary intersection tests: if the signed distance from the center of a given sphere to any of the planes of the bounding frustum is greater than the radius of the sphere, the rays bounded by the frustum cannot intersect the sphere.

D. Run Time Parameter Range Culling

To gain additional insight into the behavior of a simulation, investigators may isolate particle subsets with parameters that take on a particular value or that lie within some range of values (Fig. 7). Particles whose range of values do not overlap the currently valid range must be culled.

Parameter range culling is first applied to large groups of particles via the macrocell hierarchy. The minimum and maximum parameter values stored in each macrocell are used during traversal to determine whether or not any spheres within a macrocell will potentially produce a valid intersection. The cost of the range checking operations becomes trivial when amortized over the number of particles contained within a typical macrocell, and is significantly less than the cost of the ray/sphere intersection tests that would otherwise be required for each of the particles within a macrocell.

When the values of at least one particle lie within the currently valid range, the macrocell cannot be skipped and parameter range culling must be applied at the level of individual particles. We ensure that each particle lies within the currently valid range before actually performing the intersection test.

E. Soft Shadows

As discussed in Section II-A, shadows provide important visual cues about the relative position of objects in complex datasets. Soft shadows are preferable to hard shadows because the smooth transition from shadowed to unshadowed regions is less likely to be misinterpreted as a discontinuity in the underlying data. Although shadows and other global effects are difficult to implement in rasterization systems using impostor-based geometry, these effects are easily integrated into our approach because it is based on ray tracing.

In particular, packets of coherent shadow rays can be generated by connecting the hit point of each primary ray to multiple samples on an area light source. Thus, the shadow rays share a common origin and can be traversed in a manner identical to that used for primary ray packets. With this approach, both the number of shadow rays and the size of the light source can be controlled interactively by the user, enabling performance-for-quality trade-offs.

IV. RESULTS

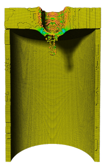
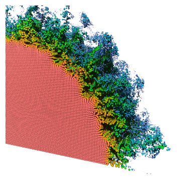
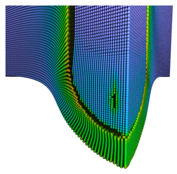
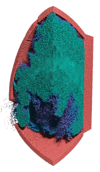
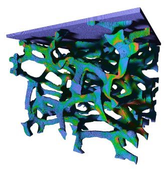
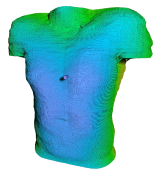
We evaluate the performance of the new CGT algorithm using several particle datasets of varying sizes and complexity with a working implementation. The pertinent characteristics of these datasets and the viewpoints used during testing are given in Table I. We first discuss the impact of the various parameters and optimizations in the new algorithm, and then compare the performance of our approach with other state-of-the-art particle visualization systems. Unless stated otherwise, the results were gathered by rendering 1024×1024 images using an Opteron machine with eight 2.4 GHz dual-core processors and 64 GB of physical memory.

A. Impact of Grid and Packet Resolution

Like the original CGT algorithm, the performance of our approach is governed by four parameters: grid resolution, macrocell resolution, ray packet size, and image resolution. As described in Section III-B, the grid resolution is determined using λ, a parameter that relates the number of cells in the grid to the total number of particles. However, unlike the original CGT algorithm, in which most scenes were largely insensitive to the value of λ, the performance of the new algorithm varies widely with different values of λ. This difference is a result of the extremely large number of particles in the test datasets. Testing several values in the range $[0.2, 5]$ shows that $\lambda = 1$ provides the best performance for all of the datasets we use. Further testing shows that a macrocell resolution of $6 \times 6 \times 6$ yields reasonable performance for these datasets. Although tuning the parameters for each dataset may yield slight performance gains, we use these parameter values for all of the tests reported in this section.

Ray packet size has a significant impact on interactive performance. For a given packet size, the cost of a traversal step is constant while the cost of intersecting the cells in a given slice increases with the number of cells the frustum overlaps. The frustum bounding a small ray packet will overlap fewer cells than that of a larger packet, but large packets amortize the costs over more rays, so there is an obvious trade-off between packet size and performance.

TABLE I
PARTICLE DATASETS USED TO EVALUATE OUR APPROACH

						
	Cylinder	**JP8**	**Bullet**	**Thunder**	**Foam**	**BulletTorso**
# particles	212980	815345	2.1 M	2.8 M	7.2 M	34.9 M
Data size	6.50 MB	21.77 MB	47.75 MB	86.33 MB	136.52 MB	1.04 GB
Frame rate	100.20 fps	99.88 fps	126.93 fps	40.86 fps	14.34 fps	18.31 fps

These datasets exhibit a wide variety of sizes and geometric complexity, and each represents a single time step of the full simulation. We evaluate a working implementation of our algorithm using the viewpoints and time steps shown above. The frame rates reported in this table were achieved by rendering 1024×1024 images using 16 threads, 8×8 ray packets, and Lambertian shading. (Performance with soft shadows is reported below.)

Table II gives the frame rates achieved when rendering each of the test datasets with a single thread using various packet sizes. The number of particles in these datasets ranges from a few hundred thousand to tens of millions, so the resulting grids are often several hundred cells in each dimension. As can be seen, 4×4 and 8×8 packets typically provide the best performance by balancing traversal cost with the number of overlapped cells. Unless stated otherwise, we use 8×8 ray packets for the remainder of our tests.

TABLE III
IMPACT OF IMAGE RESOLUTION

	1024^2		**2048^2**		
Dataset	**4×4**	**8×8**	**8×8**	**16×16**	**Ratio**
Cylinder	6.83	**7.25**	**2.95**	2.32	0.41
JP8	5.84	**8.06**	**3.04**	2.91	0.38
Bullet	8.01	**9.37**	**3.32**	2.97	0.35
Thunder	**3.56**	2.79	**1.74**	0.98	0.49
Foam	**1.57**	0.95	**0.74**	0.31	0.47
BulletTorso	**1.79**	1.30	**0.90**	0.46	0.50

Frame rates using a single thread for various packet sizes and image resolutions. The new CGT algorithm scales sublinearly with image resolution.

B. Impact of Image Resolution

In addition to grid resolution (and thus the number of particles), the optimal packet size is also influenced by the image resolution: high resolution images result in higher ray density and permit larger packet sizes. As noted, a resolution of 1024×1024 pixels, which is suitable for current displays, is used as the default value for these experiments. However, the aliasing problem, which is particularly acute for the large numbers of particles and complex geometries, increases the demand for oversampling (which is equivalent to higher image resolutions).

Ray tracing cost is typically linear in the number of pixels, but because higher resolution images allow larger ray packets, the new algorithm scales sublinearly with image resolution, as demonstrated by the data in Table III.

C. Impact of the Sphere-Center Method

The sphere-center method introduced in Section III-A alleviates many problems associated with grids. In this method, each particle is stored in exactly one grid cell so data is not duplicated, locality of reference is improved, and schemes to prevent redundant intersection tests become unnecessary.

It is not immediately clear that traversing the enlarged frustum required by the sphere-center method would not simply cancel these benefits or actually degrade performance. As the data in Table V demonstrates, however, frame rates improve by a factor of 1.02–1.27 over a standard grid that stores references to the particle data in (possibly) many cells.

TABLE II
IMPACT OF PRIMARY RAY PACKET SIZE

Dataset	**2×2**	**4×4**	**8×8**	**16×16**
Cylinder	3.99	6.94	**7.29**	4.32
JP8	2.78	5.92	**8.10**	5.93
Bullet	4.26	8.04	**9.42**	6.41
Thunder	2.96	**3.80**	2.95	1.32
Foam	1.37	**1.65**	0.98	0.25
BulletTorso	1.42	**1.83**	1.32	0.43

Frame rates achieved using a single thread for various packet sizes. In general, 4×4 and 8×8 packets provide the best performance for the datasets tested.

TABLE V
IMPACT OF THE SPHERE-CENTER METHOD

Dataset	**Standard**	**Sphere-Center**	**Speed-up**
Cylinder	6.58	7.29	1.11
JP8	7.91	8.10	1.02
Bullet	9.21	9.42	1.02
Thunder	2.37	2.95	1.24
Foam	0.81	0.98	1.21
BulletTorso	1.04	1.32	1.27

Frame rates achieved using a single thread for standard and sphere-center grids. Though originally designed to reduce storage overhead and simplify data access, the rendering performance improves by a factor of 1.02–1.27.

In addition, the sphere-center method reduces the memory footprint of our application by a factor of 2. Primitive data is stored directly in the finest level of the grid, and is neither duplicated nor referenced by pointers. For example, the BulletTorso dataset, which consists of nearly 35 million particles and consumes just over 1 GB of storage, results in a $420 \times 198 \times 432$ grid. The average particle overlaps 16.19 grid cells in this case, and a standard grid implementation

TABLE IV
COMPARISON OF GRID CONSTRUCTION TIMES

	Standard Grid				Sphere-Center Grid				
Dataset	**Insert**	**Merge**	**Mcell**	**Total**	**Insert**	**Merge**	**Mcell**	**Total**	**Speed-up**
Cylinder	3.42	11.33	15.11	32.88	1.67	8.09	6.86	18.79	1.75
JP8	9.16	43.23	54.12	116.77	5.35	25.57	23.88	62.10	1.88
Bullet	34.76	308.99	166.88	546.11	11.77	80.77	44.81	159.75	3.42
Thunder	31.90	187.61	178.64	428.69	15.79	131.13	98.66	271.69	1.58
Foam	77.28	364.08	491.54	1023.89	39.34	233.06	235.41	566.44	1.81
BulletTorso	758.85	6541.34	3041.65	10961.80	191.29	1254.40	830.86	2560.25	4.28

Time (in milliseconds) for the major stages of the grid construction process, including total build time, using eight threads for the standard and sphere-center grids. The sphere-center method allows a more efficient construction process, improving total build times by a factor of 1.58–4.28.

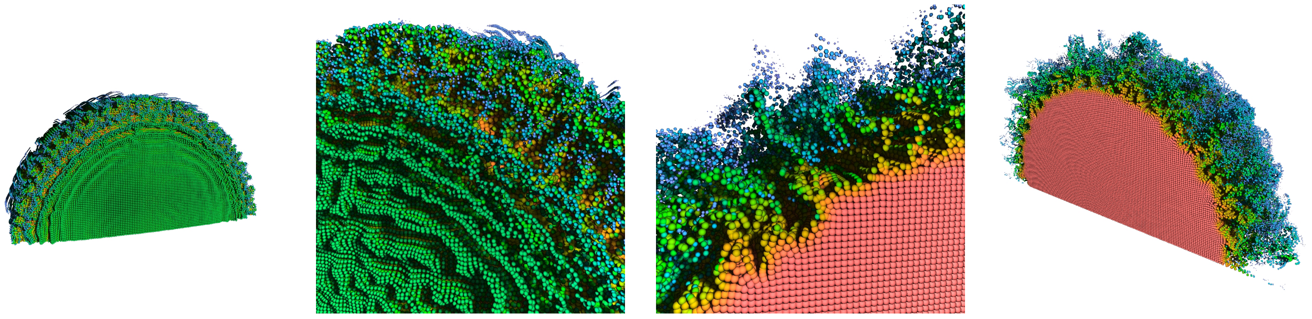


Fig. 8. *Time-varying datasets.* Using the sphere-center method, grid construction is more efficient, so grids can be built on-the-fly during rendering. Shown here are several images from an interactive session with the entire JP8 dataset, which consists of more than 140 million particles over 173 time steps.

that stores particle identifiers (as 4-byte integers) in each cell adds an additional 2.11 GB. However, using the sphere-center method, only 1.04 GB of storage is required: 32 bytes (8 data values $\times$ 4-byte floating point numbers) for each of 34.9 million spheres. Using this method, the data consumes less than half of the memory required by a standard grid.

Similarly, the results in Table IV indicate that the sphere-center method also improves grid construction times by a factor of 1.58–4.28. Spheres are placed in exactly one cell by simply truncating the floating point values expressing their centers in the grid coordinate space to integers, which requires only one SIMD operation on modern CPUs. In addition, an efficient construction process enables grids for time-varying datasets to be built on-the-fly. Constructing the grid during rendering saves the memory overhead associated with separate data structures for each time step and thus allows more time steps to be loaded. For example, Fig. 8 depicts an interactive session with the entire JP8 dataset, which consists of more than 140 million particles across 173 time steps.

D. Impact of Frustum and Parameter Range Culling

Efficient frustum culling plays an important role in the original CGT algorithm, and the same holds true for our approach. Uniform grids do not adapt to the local variations in primitive density as well as structures like kd-trees or BVHs. As a result, more primitive intersection tests are typically required during traversal of a grid than for other structures. Frustum culling cancels this behavior and reduces the number of ray/primitive intersection tests actually performed, as demonstrated by the results in Table VI. The efficient SIMD sphere/frustum culling procedure described in Section III-C reduces the number of ray/sphere intersection tests performed to 38–81% of the total potential tests. This reduction improves performance by a factor of 1.17–1.68 as shown in Table VII.

TABLE VI
STATISTICS FOR SIMD SPHERE/FRUSTUM CULLING

Dataset	**# potential tests**	**# skipped**	**% culled**
Cylinder	219470	135722	61.84%
JP8	284990	231832	81.34%
Bullet	161938	108858	67.22%
Thunder	593191	346076	58.34%
Foam	2120452	1179271	55.61%
BulletTorso	925835	355538	38.40%

Number of potential ray/sphere intersection tests and number of tests skipped by frustum culling. Efficient SIMD frustum culling reduces the number of ray/sphere intersection tests required during grid traversal.

TABLE VII
IMPACT OF SIMD SPHERE/FRUSTUM CULLING

Dataset	**No culling**	**Culling**	**Speed-up**
Cylinder	71.04	99.16	1.40
JP8	59.13	99.79	1.68
Bullet	87.20	123.37	1.41
Thunder	29.22	40.78	1.40
Foam	9.64	14.21	1.47
BulletTorso	15.69	18.31	1.17

Frame rates achieved with and without frustum culling. Interactive performance improves by a factor of 1.17–1.68 for the test datasets.

As described in Section III-D, parameter range culling is applied at the level of both the macrocells and the individual particles. The results in Table VIII, which correspond to the images in Fig. 7, indicate that this feature adds some additional overhead. However, efficient SIMD implementation of the range checking operations decreases performance by only a factor of 1.38–2.28 over preprocessed versions of the data. Some of this performance difference can be attributed to the slight difference in the grid bounds (with smaller bounds leading to fewer packet traversals), but the 16 SIMD operations implementing macrocell and particle range checking also adds some computational overhead. Nevertheless, parameter range culling provides additional flexibility during the data analysis process, a benefit that clearly outweighs the relative impact on interactive performance.

TABLE VIII
IMPACT OF PARAMETER RANGE CULLING

Dataset	**% total**	**PR culling (w/o mcells)**	**PR culling (w/ mcells)**	**Preprocessed**
Thunder	43.00%	14.23	43.06	59.55
BulletTorso	34.17%	3.17	11.18	25.49

Frame rates achieved for run time parameter range culling and preprocessed data. Parameter range culling adds some additional overhead, but interactive performance degrades only slightly when compared to preprocessed datasets composed of the same particles.

E. Impact of Soft Shadows

To this point, we have only considered simple ray casting and local shading; non-local effects such as shadows have not been considered. However, using the approach described in Section III-E, we can also support soft shadows quite easily.

Although interactive performance with soft shadows depends heavily on the coherence exhibited by secondary ray packets, the impact is sublinear in the number of shadow rays traced, as demonstrated by the data in Table IX. Interactive performance varies widely for the datasets and lighting configurations tested, with the impact ranging from a factor of 2.42 (for 2×2 packets, or 4 shadow rays per primary ray) to as much as 19.35 (for 8×8 packets, or 64 shadow rays).

TABLE IX
IMPACT OF SOFT SHADOWS

Dataset	**No shadows**	**2×2**	**4×4**	**8×8**
Cylinder	100.20	14.91	10.47	5.34
JP8	99.88	16.31	12.21	6.65
Bullet	126.93	19.55	13.69	6.56
Thunder	40.86	14.07	10.74	6.45
Foam	14.34	4.38	2.51	1.17
BulletTorso	18.30	7.39	6.09	3.82

Frame rates achieved for various shadow settings. In these tests, the light source area is rather large, at 5% of the solid angle subtended by the bounding box of the scene.

The flexibility of an interactive visualization environment puts these parameters under the full control of the user at run time, allowing trade-offs between image quality and interactive performance. For example, Fig. 9 shows the results of using 2×2, 4×4, and 8×8 shadow rays per primary ray. Quality can be traded for performance by simply using fewer shadow rays or by disabling shadows during periods of interaction.

F. Comparison with Other Approaches

Finally, we compare the performance of our approach with two recent systems that, to our knowledge, represent the current state-of-the-art in interactive visualization of large

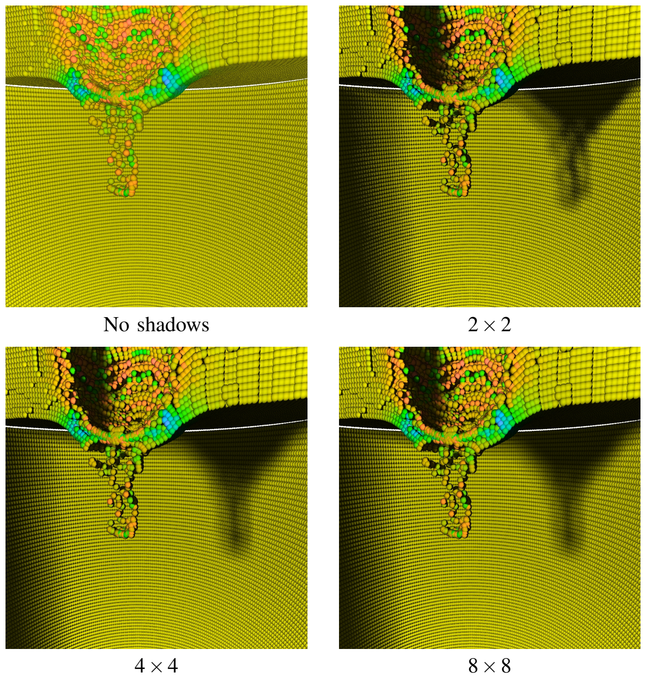


Fig. 9. *Rendering with soft shadows.* Soft shadows from area light sources provide important visual cues about the relative position of objects in complex datasets. Shadows and other global effects are easily integrated into an approach based on ray tracing such as the one described here.

particle datasets. The first is based on an optimized single ray grid traversal algorithm [24], [33], while the second leverages programmable graphics hardware and software-based acceleration techniques [23].

The results reported in Table X that correspond to optimized single ray traversal have been gathered on the test machine described above; those corresponding to the approach based on programmable graphics hardware were gathered using a dual processor workstation with 8 GB of physical memory and an NVIDIA GeForce 7800 GT graphics card. As can be seen, our CGT algorithm compares favorably with these systems for the test datasets. The benefits of packet-based traversal become evident when compared to single ray traversal: interactive performance improves by a factor of 1.35–14.48 for the test datasets. Though some of the improvement results from our use of SIMD instructions that are not easily employed by a single ray scheme, such an implementation usually provides an improvement of only a factor of 2–3; the remainder is a result of the cost amortization and algorithmic improvements inherent to a packet-based traversal method. The overall improvement is consistent with that obtained by the original CGT algorithm for polygonal scenes, despite the differences in primitive type and density exhibited by our application.

Surprisingly, our approach also outperforms the system based on programmable graphics hardware. This system uses view-aligned, textured billboards to represent each particle. Vertex and fragment programs manipulate this data to provide a high-quality representation of each particle that is consistent with the results of an approach based on ray tracing. In addition, software-based acceleration techniques (including basic frustum culling and more sophisticated occlusion culling algorithms) are used to reduce the rendering workload in each frame. Nevertheless, and despite the fact that our test machine actually provides less raw FLOPS than the NVIDIA GeForce 7800 GT used to test the GPU-based system, our approach outperforms this system by a factor of about 5 to almost 50 for the datasets tested.

TABLE X
COMPARISON OF PARTICLE VISUALIZATION METHODS

Dataset	**Our CGT**	**RTRT**	**GPU-based**
		Bigler et al. [24]	Gribble et al. [23]
Cylinder	100.20	12.60	5.78
JP8	99.88	6.90	17.40
Bullet	126.93	11.90	2.56
Thunder	40.86	14.90	8.10
Foam	14.33	6.40	2.04
BulletTorso	18.31	13.50	1.56

Frame rates achieved using our CGT algorithm and two state-of-the-art particle visualization systems. The benefits of packet-based traversal become evident when compared to single ray traversal, and our approach also outperforms an approach leveraging programmable graphics hardware.

V. CONCLUSIONS

We have presented a new algorithm for interactive particle visualization that is based on efficient ray tracing using coherent grid traversal. We employ fast ray tracing methods for multi-level grids, including ray packets, frustum based traversal, frustum culling, and SIMD operations. The algorithm exploits the properties of particle-based simulation data to improve performance and reduce storage requirements using the sphere-center method. This approach addresses some problems traditionally associated with grids, namely duplicate data, little locality of reference, and redundant ray/primitive intersection tests. The sphere-center method not only reduces the memory required when rendering large, time-varying particle datasets, but also leads to improved performance and facilitates valuable data exploration tasks such run time parameter range culling. In addition, the grid construction process is made more efficient with this method by replacing primitive/cell or bounding box/cell overlap tests with a simple float-to-int truncation. Additional optimizations based on efficient sphere/frustum culling further improve the interactive performance of the algorithm.

A. Discussion

We have evaluated the performance of our approach using a system with eight 2.4 GHz dual-core processors (16 processing cores total). The theoretical peak available on this machine is less than 155 GFLOPS, which is an order of magnitude less than the terascale performance of, for example, the ATI X1900 graphics processing unit [34]. The evaluation of our algorithm shows highly interactive frame rates on reasonably priced multi-core platforms (a system with compute power similar to the test machine would cost less than $35,000 at the time of this writing). Moreover, it is only a matter of time before compute power of this magnitude is available on commodity desktop systems.

Our approach also compares favorably with recent systems that represent the current state-of-the-art in interactive visualization of large particle datasets. A previous approach based on interactive ray tracing [24] utilizes single ray traversal, but the new CGT algorithm has demonstrated a 35% to 700% performance improvement over this system (Table X), and reduces the hardware costs necessary to achieve interactivity. Systems using programmable graphics hardware [23] also offer a way to visualize large, time-varying particle datasets at interactive rates. This hardware is widely available, and a desktop system so equipped is considerably less expensive (roughly a factor of 7) than the system used to evaluate our algorithm. However, GPU-based approaches are not easily extended to include visualization features like soft shadows or advanced shading models, while an algorithm based on ray tracing can be extended to include these features naturally. We have achieved reasonably interactive performance with a naive implementation of soft shadows, and advanced shading models such as ambient occlusion or physically based diffuse interreflection will become feasible with continued improvements in both algorithmic design and CPU performance.

B. Future Work

Several areas require further attention. First, techniques similar to the sphere-center method may be applicable to other types of primitives such as triangles, and exploring these methods is of interest. In addition, the current implementation of soft shadows treats secondary rays in a manner identical to primary rays. Additional improvements in performance may result from optimizations specific to secondary ray packets. Accelerating performance of secondary rays is also important if the visual cues from advanced shading models like ambient occlusion and physically based diffuse interreflection are to be used during interactive rendering. Exploring efficient methods to include these effects is of particular interest. Finally, multi-modal visualization of particle and volumetric data, such as a container (particle-based simulation) in a pool fire (computational fluid dynamics simulation), would also be useful. Efficient techniques for packet-based volume rendering are required to combine this visualization modality with the particle visualization method we have described.

ACKNOWLEDGMENTS

We thank Jerry Seidler and Erin Miller, both of the University of Washington, and Scott Bardenhagen, of the Los Alamos National Laboratory, for providing the Foam dataset. Parts of this work were funded by the DOE ASC program.

REFERENCES

[1] M. Levoy, “Display of Surfaces from Volume Data,” *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1988.
[2] W. E. Lorensen and H. E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm,” in *International Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 163–169.
[3] J. E. Guilkey, J. A. Hoying, and J. A. Weiss, “Computational Modeling of Multicellular Constructs with the Material Point Method,” *Journal of Biomechanics*, vol. 39, no. 11, pp. 2047–2086, August 2006.
[4] A. Reshetov, A. Soupikov, and J. Hurley, “Multi-Level Ray Tracing Algorithm,” *ACM Transacions on Graphics*, vol. 24, no. 3, pp. 1176–1185, 2005, (ACM SIGGRAPH ’05).
[5] I. Wald, S. Boulos, and P. Shirley, “Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies,” *ACM Transactions on Graphics*, vol. 26, no. 1, 2007, (to appear).
[6] I. Wald, T. Ize, A. Kensler, A. Knoll, and S. G. Parker, “Ray Tracing Animated Scenes using Coherent Grid Traversal,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 485–493, 2006.

[7] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley, "Interactive Ray Tracing for Volume Visualization," *IEEE Trans. on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 238–250, 1999.

[8] T. Ize, I. Wald, C. Robertson, and S. G. Parker, "An Evaluation of Parallel Grid Construction for Ray Tracing Dynamic Scenes," in *2006 IEEE Symposium on Interactive Ray Tracing*, 2006, pp. 47–55.

[9] D. Sulsky, S. Zhou, and H. L. Schreyer, "A Particle Method for History Dependent Materials," *Computer Methods in Applied Mechanical Engineering*, vol. 118, pp. 179–196, 1994.

[10] ——, "Application of a Particle-in-Cell Method to Solid Mechanics," *Computer Physics Communications*, vol. 87, pp. 236–252, 1995.

[11] S. Ullman, *The Interpretation of Visual Motion*. Cambridge, Massachusetts: MIT Press, 1979.

[12] H. von Helmholtz, *Handbook of Physiological Optics*. New York: Optical Society of America, 1925.

[13] L. R. Wanger, J. A. Ferwerda, and D. P. Greenberg, "Perceiving Spatial Relationships in Computer-Generated Images," *IEEE Computer Graphics and Applications*, vol. 12, no. 3, pp. 44–58, 1992.

[14] E. R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, Connecticut: Graphics Press, 2001.

[15] P. Mamassian, D. C. Knill, and D. Kersten, "The Perception of Cast Shadows," *Trends in Cognitive Sciences*, vol. 2, no. 8, pp. 288–295, 1998.

[16] P. Rheingans and C. Landreth, "Perceptual Principles for Effective Visualizations," *Perceptual Issues in Visualization*, pp. 59–74, 1995.

[17] M. Sattler, R. Sarlette, T. Mücken, and R. Klein, "Exploitation of Human Shadow Perception for Fast Shadow Rendering," in *Proceedings of the Second Symposium on Applied Perception in Graphics and Visualization*, 2005, pp. 131–134.

[18] C. P. Gribble and S. G. Parker, "Enhancing Interactive Particle Visualization with Advanced Shading Models," in *Proceedings of the Third Symposium on Applied Perception in Graphics and Visualization*, July 2006, pp. 111–118.

[19] M. Krogh, J. Painter, and C. Hansen, "Parallel Sphere Rendering," *Parallel Computing*, vol. 23, no. 7, pp. 961–974, 1997.

[20] K. Liang, P. Monger, and H. Couchman, "Interactive Parallel Visulization of Large Particle Datasets," in *Eurographics Symposium on Parallel Graphics and Visualization*, 2004, pp. 111–118.

[21] P. Zemcik, P. Tisnovsky, and A. Herout, "Particle Rendering Pipeline," in *19th Spring Conference on Computer Graphics*, 2003, pp. 165–170.

[22] P. Zemcik, A. Herout, L. Crha, O. Fucik, and P. Tupec, "Particle Rendering Engine in DSP and FPGA," in *11th International Conference and Workshop on the Engineering of Computer-based Systems (ECBS '04)*, 2004, p. 361.

[23] C. P. Gribble, A. J. Stephens, J. E. Guilkey, and S. G. Parker, "Visualizing Material Point Method Datasets on the Desktop," in *British HCI 2006 Workshop on Combining Visualization and Interaction to Facilitate Scientific Exploration and Discovery*, September 2006, pp. 1–8.

[24] J. Bigler, J. Guilkey, C. Gribble, S. Parker, and C. Hansen, "A Case Study: Visualizing Material Point Method Data," in *Proceedings of the Eurographics/IEEE Symposium on Visualization*, 2006, pp. 299–306.

[25] M. Tarini, P. Cignoni, and C. Montani, "Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1237–1244, 2006.

[26] S. Melax, "Dynamic Plane Shifting BSP Traversal," in *Proceedings of Graphics Interface '00*, 2000, pp. 213–220.

[27] D. Kirk and J. Arvo, "Improved Ray Tagging for Voxel-Based Ray Tracing," in *Graphics Gems II*. Academic Press, 1991, pp. 264–266.

[28] D. E. DeMarle, C. Gribble, and S. Parker, "Memory-Savvy Distributed Interactive Ray Tracing," in *Eurographics Symposium on Parallel Graphics and Visualization*, 2004, pp. 93–100.

[29] F. Cazals, G. Drettakis, and C. Puech, "Filtering, Clustering and Hierarchy Construction: a New Solution for Ray Tracing Very Complex Environments," *Computer Graphics Forum*, vol. 14, no. 3, 1995.

[30] D. Jevans and B. Wyvill, "Adaptive Voxel Subdivision for Ray Tracing," in *Proceedings of Graphics Interface '89*, 1989, pp. 164–172.

[31] K. S. Klimaszewski and T. W. Sederberg, "Faster Ray Tracing using Adaptive Grids," *IEEE Computer Graphics and Applications*, vol. 17, no. 1, pp. 42–51, January/February 1997.

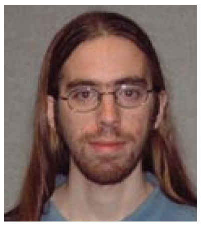
[32] K. Dmitriev, V. Havran, and H.-P. Seidel, "Faster Ray Tracing with SIMD Shaft Culling," Max-Planck Institut für Informatik, Tech. Rep. MPI-I-2004-4-006, 2004.

[33] S. Parker, W. Martin, P.-P. Sloan, P. Shirley, B. Smits, and C. Hansen, "Interactive Ray Tracing," in *Symposium on Interactive 3D Graphics*, 1999, pp. 119–126.

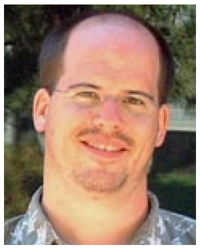
[34] D. E. Polkowski, "ATI's Radeon X1900 Heats Up With 48 Shader Units," http://www.tomshardware.com/2006/01/24/, January 2006.

Christiaan P. Gribble is an Assistant Professor in the Department of Computer Science at Grove City College. His research focuses on global illumination algorithms, interactive and realistic rendering, scientific visualization, and high-performance computing. Gribble has served as a post-doctoral research fellow and research assistant for the Scientific Computing and Imaging (SCI) Institute at the University of Utah, and as a research assistant at the Pittsburgh Supercomputing Center. In 2005, he received the Graduate Research Fellowship from the University of Utah. Gribble received the BS degree in mathematics from Grove City College in 2000, the MS degree in information networking from Carnegie Mellon University in 2002, and the PhD degree in computer science from the University of Utah in 2006.

Thiago Ize received the BS degree in both mathematics and computer science from the University of Virginia in 2004. He is currently a PhD student in the School of Computing and Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research interests are in rendering, with a current focus on interactive ray tracing.



Andrew Kensler is a PhD student in the School of Computing at the University of Utah and a research assistant with the Scientific Computing and Imaging (SCI) Institute. He received the BA degree in computer science from Grinnell College in 2001. His research focuses on rendering, with interests in interactive ray tracing, global illumination algorithms, and photorealistic rendering.



Ingo Wald holds a PhD in engineering from Saarbrücken University. After his PhD, he was a post-doctoral research associate at the Max Planck Institute for Informatics in Saarbrücken, Germany, and is currently a Research Assistant Professor at the University of Utah. His work concentrates on all aspects of real time ray tracing, photorealistic rendering, scientific visualization, efficient parallel rendering, and massive model rendering. He has written numerous ray tracers, and has founded and led the OpenRT Realtime Ray Tracing Project.

Steven G. Parker is an Assistant Professor in the School of Computing and Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research focuses on problem solving environments, which tie together scientific computing, scientific visualization, and computer graphics. He is the principal architect of the SCIRun Software System, which formed the core of his PhD dissertation, and is currently the chief architect of Uintah, a software system designed to simulate accidental fires and explosions using thousands of processors. He was a recipient of the DoE Computational Science Graduate Fellowship. He received the BS degree in electrical engineering from the University of Oklahoma in 1992, and the PhD degree from the University Utah in 1999.