Visualizing Particle-Based Simulation Data on the Desktop

Christiaan P. Gribble
School of Computing
University of Utah

James E. Guilkey
Department of Mechanical Engineering
University of Utah

Abraham J. Stephens
School of Computing
University of Utah

Steven G. Parker
School of Computing
University of Utah

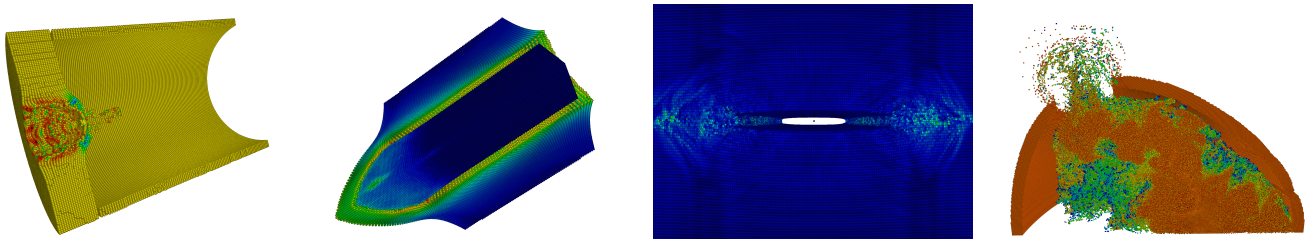


Figure 1: *Particle-based simulation.* Using particle methods, scientists can model complex physical phenomena and solve time-dependent problems of various scales. Illustrated here are the results of several such simulations, including (from left to right): an aluminum sphere impacting the end of an aluminum cylinder, a steel projectile penetrating a iron block, crack propagation in a solid block, and a container of high-energy explosives rupturing. Typical simulations contain millions of particles across many time steps, and our system visualizes large, time-varying particle datasets like these at interactive rates on desktop systems equipped with commodity graphics hardware.

ABSTRACT

Particle-based simulation methods are used to model a wide range of complex phenomena and to solve time-dependent problems of various scales. Effective visualization of particle-based simulation data requires communicating subtle changes in the three-dimensional structure, spatial organization, and qualitative trends within a simulation as it evolves, as well as allowing easier navigation and exploration of the data through interactivity. We describe an approach to rendering large, time-varying particle datasets using commodity graphics hardware on desktop systems. We advance the current state-of-the-art by bringing visualization of particle-based simulation data to the desktop. Our approach performs competitively with current interactive ray tracing systems and runs on hardware that is a fraction of the cost, making particle visualization and data exploration more accessible.

CR Categories: I.3.4 [Computer Graphics]: Graphics Utilities—Application packages I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: particle-based simulation, particle visualization, interactive rendering, occlusion algorithms

1 INTRODUCTION

Particle methods are commonly used to simulate complex phenomena in many scientific domains, including astronomy, biology, chemistry, and physics. Using techniques such as particle-in-cell [21], molecular dynamics [2, 31], smoothed particle hydrodynamics [14, 29, 30], element-free Galerkin [6], and discrete element methods [13, 35], computational scientists model such phenomena as a system of discrete particles that obey certain laws and possess certain properties. These methods are particularly attractive because they can be used to solve time-dependent problems on scales from the atomic to the cosmological. Figure 1 illustrates the results of several particle-based simulations that model a variety of physical systems.

The material point method (MPM) [32, 33] is a particular particle-in-cell simulation method. MPM is well-suited to problems with high deformations and complex geometries, and offers advantages over grid based methods for a certain class of problems [4]. Like other quasi-meshless techniques, MPM represents solid objects using collections of Lagrangian particles, or material points, each of which represents a part of the domain. Each particle has associated with it the properties of the material it represents, as well as its physical state, including position, mass, velocity, temperature, stress, and any other quantities of interest. During the simulation, the properties of each particle are projected to a regular structured grid, where the equations of motion are solved using either explicit [32] or implicit [19] time integration. The results from each time step are interpolated back to the particles, updating their physical state. Figure 2 illustrates the basic elements of a simple MPM simulation.

Investigators use particle visualization to assist efforts in data analysis and feature detection, as well as in debugging ill-behaved solutions. One approach to visualization of data from particle-based simulations interpolates particle values to a three-dimensional grid, possibly the same grid used during the computation, as in the case of MPM. The transformed data is then visualized using standard methods such as isosurface rendering [28] and direct volume rendering [25].

Grid-based representations are suitable for some, but not all, visualization tasks. The limited resolution of the grid itself can be problematic: fine structural details within the data may be lost. To alleviate this issue, the grid can be refined, either uniformly or adaptively. Scientists are often interested in simultaneously examining both the large- and small-scale structure within these datasets, however, so grid-based techniques may not be appropriate. Additionally, interpolation may hide features or problems present in the original particle data. For example, small particles that have extremely high temperatures or velocities may be invalid, and the

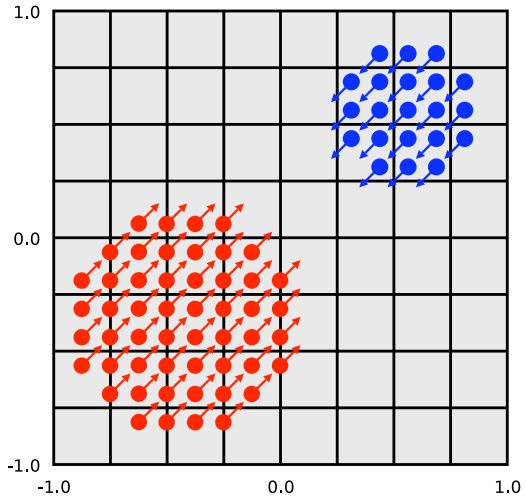


Figure 2: *The material point method.* Collections of Lagrangian particles, or material points, are used to represent solid objects. Particle state is projected to a grid, where the computations are performed. The results of each calculation are then interpolated back to the particles, updating their physical state.

influence of such particles can be masked by interpolation to the grid. Moreover, interpolation and isosurface extraction can be a time-consuming task, particularly for large datasets.

Particles 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. Glyph-based visualizations are able to preserve the fine details within the simulation data while maintaining the large-scale three-dimensional structure of the object the particles represent. We have found that glyph-based representations are particularly useful for the data analysis and feature detection tasks scientists often perform.

Frequently, millions of particles are required to capture the behavior of a system accurately. Such massive simulations lead to very large, very complex datasets, making interactive visualization a difficult task. Moreover, the need to simultaneously visualize both the large- and small-scale features of the data further exacerbate these issues. An effective particle visualization method will communicate subtle changes in the three-dimensional structure, spatial organization, and qualitative trends within a simulation as it evolves, as well as enable easier navigation and exploration of the data through interactivity.

Recently, Bigler et al. [7] described a system that represents the current state-of-the-art in particle-based data visualization. Their system leverages an interactive ray tracer running on large multiprocessor shared-memory platforms to visualize tens to hundreds of time steps, each with millions of particles, at interactive rates. While this approach satisfies the requirements of particle visualization, the hardware costs imposed by such a system are often prohibitive and impede accessibility. We address this issue and present an approach to rendering large, time-varying particle datasets using commodity graphics hardware on desktop systems. Our system satisfies all of the requirements of effective particle-based data visualization and is more accessible than previous systems.

2 Related Work

Our approach to particle visualization leverages results from the computer graphics, scientific visualization, and perception literature in order to achieve a system featuring data exploration capabilities and performance competitive with systems based on interactive ray tracing.

2.1 Sphere Rendering

In early work by Krogh et al. [24], massively parallel processors were used to render large particle datasets in parallel. This work recognized the need to simultaneously examine both macro- and microscopic features within the data, and the system gave investigators the ability to do so with large datasets. More recently, Liang et al. [27] also take a parallel approach to particle visualization. In particular, a parallel rendering cluster is used to visualize datasets containing as many as 256^3 particles at interactive frame rates. Using only seven rendering nodes, this system achieves 9 frames per second when visualizing 256^3 particles. The system also handles larger datasets (512^3 particles), but cannot do so interactively, achieving only about 1 frame per second.

Zemcik et al. [38] describe a hardware design tailored for rendering single spheres at very high speeds. They estimate that their design is capable of rendering up to 100 million spheres per second, which would enable large particle datasets to be rendered in real-time. Gumhold [20] also tackles the problem of rendering spheres and ellipsoids in hardware. This work presents a simple solution to the problem of ray-ellipsoid intersection using the vertex and fragment processing capabilities of commodity graphics hardware. Additionally, this method can be combined with software-based acceleration techniques because the fragment processing steps occur toward the end of the rendering pipeline. We harness this functionality to visualize large, time-varying particle datasets at interactive rates. Other visualization systems, for example, TexMol [3], also employ programmable graphics hardware to render a variety of geometric glyphs common to scientific visualization tasks.

2.2 Visibility Culling

Visibility culling algorithms attempt to reduce the number of primitives sent to the graphics hardware, saving the cost associated with rendering objects that do not contribute to the final image. View-frustum culling (VFC) in one such technique that quickly culls objects outside the view volume. Unfortunately, VFC does not eliminate objects that are occluded by other objects and so does not prevent overdraw. This problem, in which an area in image-space is covered than once, wastes computational resources in both the vertex and fragment processing stages found in current graphics hardware.

Occlusion culling algorithms attempt to address the overdraw problem and can be classified either as *from-region* (offline) or *from-point* (online) algorithms. From-region occlusion culling, which include the algorithms proposed by Airey et al. [1], Teller and Sequin [34], and Leyvand et al. [26], compute a potentially visible set (PVS) of objects in a preprocessing phase. Typically, a fixed subdivision of the scene is used, and the PVS for a given viewpoint can be determined quickly during runtime. In contrast, online occlusion culling techniques, which include those described by Greene et al. [15], Zhang et al. [39], Bittner et al.[9], and Klosowski and Silva [23], apply a visibility computation for each viewpoint encountered during rendering.

Many from-point algorithms employ the capabilities of graphics hardware during visibility computation. For example, Zhang et al. [39] propose hierarchical occlusion maps that make use of hardware texturing, while Wonka et al. employ occluder shadows for from-point visibility [36] and online from-region visibility [37]. Other algorithms exploit hardware occlusion queries [5]. For example, Hillesland et al. [22] employ hardware occlusion queries and a uniform, possibly nested, grid to determine visible geometry.

This method exploits neither spatial nor temporal coherence during rendering, and is restricted to regular spatial subdivision data structures. Coherent hierarchical culling (CHC) [10] overcomes both of these problems and provides an adaptation of earlier improvements [8] that is tailored for hardware occlusion queries. We explore the performance characteristics of several occlusion culling techniques, including CHC, in the context of particle visualization.

2.3 Enhancing Particle Visualization

A recent psychophysical study by Gribble and Parker [16] demonstrates that illumination effects from diffuse interreflection positively and significantly impact a viewer's ability to correctly evaluate the three-dimensional structure and spatial organization within particle datasets. These results motivate the use of advanced shading models in current systems based on interactive ray tracing [7, 16]. In these systems, a preprocessing phase samples the illumination across each particle using a Monte Carlo path tracer and stores the results in per-particle illumination maps, called precomputed luminance textures (PLTs), which are then mapped to the particles during interactive visualization. The memory requirements for storing PLTs are often prohibitive and limit the ability of investigators to explore large, time-varying datasets [7]. Compressing the PLTs using vector quantization or principal components analysis address this issue [16]. This visualization process enables interactive rendering of large particle datasets with effects from advanced shading models. In addition, Bigler et al. [7] use image-based silhouette edges to emphasize a set of view-dependent boundaries, further highlighting the three-dimensional structure of the particles. We implement both of these advanced particle visualization methods in graphics hardware via multi-pass fragment processing.

3 System Implementation

Our system satisfies the requirements of effective particle visualization and offers advanced data exploration capabilities and interactive performance on desktop systems. We now discuss the critical components of our approach, which include:

- rendering high-quality particle glyphs,
- achieving interactive performance with commodity graphics hardware,
- handling time-varying data while maintaining interactivity, and
- supporting data exploration and enhanced visualization techniques.

These components, taken in combination, form the basis of our interactive particle visualization system.

3.1 Rendering High-Quality Particle Glyphs

The need to simultaneously visualize both large- and small-scale structures within the data requires that each particle be rendered using a high-quality representation. One approach would simply render a highly tessellated, view-aligned hemisphere for each particle. However, because each time step contains millions of particles, the required geometry would quickly overwhelm the system and result in poor performance.

To alleviate this issue while maintaining particle quality, we employ view-aligned billboards as the base primitive. Each billboard is rendered efficiently using the point sprite rendering capabilities of modern graphics hardware. Individual point sprites are specified by a vertex corresponding to the particle's position. Per-vertex attributes control other aspects of the particle's representation, including its radius and the scalar value used for color mapping. Vertex and fragment programs manipulate this data to render a high-quality representation of each particle in an efficient manner.

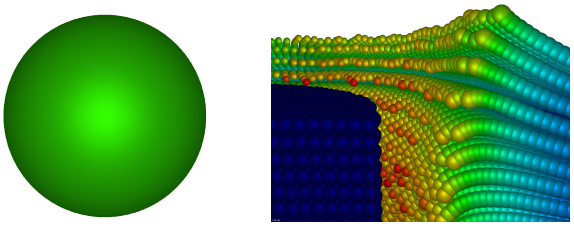


Figure 3: *High-quality particle rendering with point sprites.* View-aligned billboards, or point sprites, serve as the base primitive. An individual particle is specified by a vertex position and other per-vertex attributes that control the particle's representation (left). Rendering high-quality particle glyphs in an efficient manner enables simultaneous visualization of both large- and small-scale features within the data (right).

The vertex program performs three simple operations on each particle. First, it passes the particle's radius and associated scalar value to the fragment program without modification. (These values are used for the per-fragment lighting and depth correction operations in the fragment program, described below.) Second, the vertex program applies the current modelview and projection matrices to the vertex to determine the particle's final position in the viewing coordinate space. Third, it computes a point on the particle's surface by offsetting the vertex by the particle's radius and applying the projection matrix to the resulting point. The perspective-correct point size is simply the difference between the surface point and the vertex position in the viewing coordinate space.

The fragment shader is more complex due to the per-fragment lighting and depth correction operations necessary for rendering high-quality spheres. First, the fragment's normal is determined by indexing a two-dimensional quantized normal map. The alpha channel of this map indicates whether the texel is within the boundary of the sphere; if it is not, the current fragment is culled and processing terminates. Otherwise, the value is dequantized to produce the fragment's surface normal. Second, per-fragment lighting is computed by summing the diffuse and specular components of a Phong-style shading model. The surface color, which is determined by indexing a one-dimensional texture using the scalar value associated with the particle, is multiplied by the incident light at the point to produce the final fragment color. Third, the fragment's depth is corrected by applying the projection matrix to the z-component of the particle position in modelview coordinates, offset by the radius.

The results of this process are illustrated in Figure 3. As can be seen, point sprites provide a way to render high-quality particle glyphs in an efficient manner.

3.2 Achieving Interactive Performance

The per-fragment lighting and depth correction steps occur toward the end of the rendering pipeline, so the particle rendering process described above can be combined with software-based acceleration techniques to reduce the rendering workload in each frame. We employ three visibility culling algorithms to accelerate particle rendering.

A first approximation to the set of potentially visible particles can be obtained by simple view-frustum culling (VFC). With this technique, particles outside the view volume are quickly culled, saving the cost associated with rendering particles that are not vis-

ible. While each of the particles can be tested against the view volume individually, it is more efficient to cull large groups of particles by testing a bounding primitive encapsulating each group. In our implementation, each bounding primitive corresponds to a node in a spatial subdivision structure such as an octree or bounding volume hierarchy.

VFC can be combined with other, more sophisticated culling techniques that utilize hardware-based occlusion queries. Hierarchical stop-and-wait occlusion culling (HSW) is one such algorithm. In this method, the spatial subdivision structure is traversed, depth-first, and an occlusion query is issued for nodes whose bounding boxes either intersect or are contained within the view frustum. The CPU waits for the result of the query before proceeding. When the result is available, the number of pixels covered by the proxy geometry is used to determine the node's visibility. If the node is not visible, its subtree is culled (interior node) or the associated particles are not rendered (leaf node) and traversal continues. Otherwise, the node is visible, and the algorithm continues either by traversing the node's children in a front-to-back order (interior node) or by rendering the associated particles (leaf node).

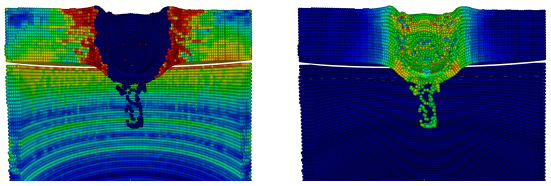
Hardware occlusion queries typically have a high latency, which leads to a load balancing problem between the CPU and the GPU. Coherent hierarchical culling (CHC) [10] is designed to address this issue. CHC exploits temporal coherence in a node's visibility classification when rendering arbitrary occluders by reusing the results of occlusion queries issued in the previous frame. Visibility information is maintained with each node in a spatial subdivision structure. This simple solution also reduces query overhead: previously visible interior nodes are processed without issuing an occlusion query because the visibility status of such nodes can easily be determined from their children. Queries are issued only for previously visible leaf nodes and for the largest possible occluded nodes in the hierarchy.

In addition to these occlusion culling algorithms, we exploit a particular use case of particle visualization to further accelerate rendering performance under certain conditions. Investigators are often interested in isolating particular subsets of the particles that possess physical properties falling within some range of interest. We exploit this constraint during traversal of the spatial subdivision structure. If the range of values contained within a node does not overlap the currently valid range, the node is skipped and its subtree is culled.

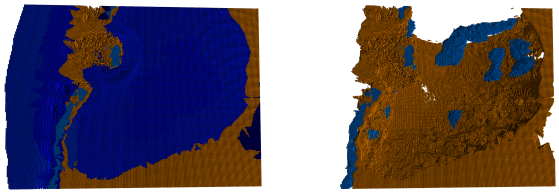
3.3 Handling Time-varying Data

We render time-varying data by treating each time step individually: A separate spatial subdivision structure is constructed for each time step as the corresponding data is loaded. The size of the structure is typically very small compared to size of the data itself, so the overhead of maintaining a separate structure for each time step is quite low. The data is animated temporally by simply cycling through each time step in successive frames.

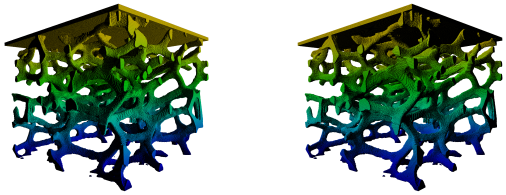
The time-varying nature of particle-based simulation data presents some interesting challenges to maintaining interactive performance. Each time step typically contains millions of particles, and there are often tens or hundreds of time steps, so the sheer number of particles can be problematic. **The naive use of the acceleration techniques described above often mitigates this issue quite effectively. However, using a naive approach to CHC in which separate visibility information is maintained for each node and each time step results in a loss of frame-to-frame coherence because many frames may have been rendered before a given time step is visible again during periods of temporal animation. To overcome this issue, we introduce a simple modification to the basic CHC algorithm that assumes a one-to-one correspondence between nodes of the spatial subdivision structures in each time step and stores only one set of visibility information across all time steps.** The visibility information for a given node is always used, even if that information was determined using the result of an occlusion query issued for the corresponding node in the previous time step. This approach (which we call CHC-TV, for time-varying data) often results in better coherence when the data is animated because changes in the actual particle positions tend to occur slowly between time steps.



(a) Color mapping



(b) Particle isolation



(c) Dynamic lighting

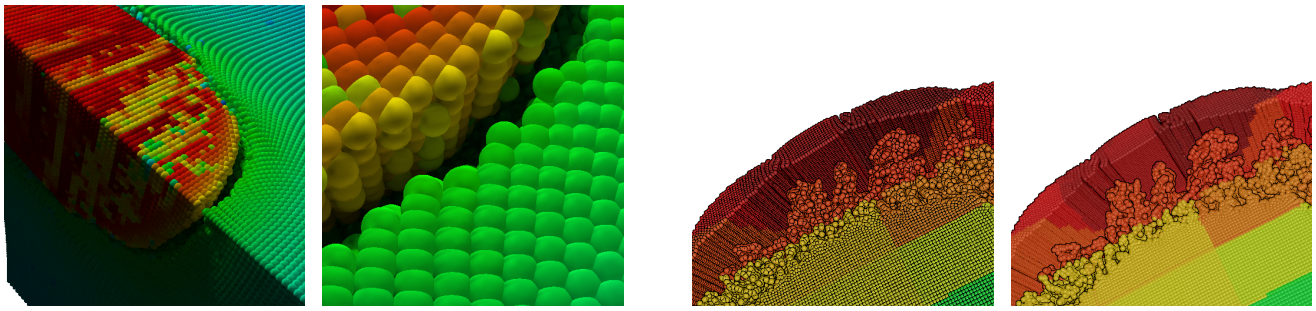
Figure 4: *Interactive data exploration and analysis.* These capabilities allow investigators to interrogate the data based on the physical state of each particle and contribute to an investigator's understanding of the structure, organization, and qualitative trends within the data.

3.4 Supporting Data Exploration and Enhanced Visualization

As the preceding discussion indicates, visualizing large, time-varying particle datasets presents many challenges to achieving interactive performance, but the complexity inherent in the data presents many data analysis and exploration challenges.

Basic data exploration and analysis tools such as color mapping and particle isolation, discussed above, allow investigators to interrogate the data based on the physical state associated with each particle. Figure 4 illustrates some of these features, including color mapping, particle isolation, and dynamic lighting. Each of these features can be controlled interactively at run time and contribute to the understanding of the structure, organization, and qualitative trends within the data.

Research has shown that particle visualization can be also enhanced by advanced shading models such as ambient occlusion and physically based diffuse interreflection [16], as well as by image-based silhouette edges [7]. Our system supports each of these visualization modes via multi-pass fragment processing. In particular, precomputed luminance textures that have been compressed using



(a) Compressed precomputed luminance textures

(b) Image-based silhouettes

Figure 5: *Enhancing particle visualization.* Advanced shading models such as ambient occlusion and physically based diffuse interreflection lead to enhanced perception of particle datasets (a). Additionally, silhouette edges can be used to accentuate a set of view-dependent boundaries. A user-defined threshold controls the edges that are displayed, enabling investigators to explore different particle groupings (b).

Dataset	Time steps	Total particles [millions]	Total data size [MB]
Bullet	3	8.4	143.5
Crack	1	34.9	666.9
Cylinder	2	0.8	26.1
Fireball	21	18.9	304.9
Foam	1	7.2	136.5
Thunder	55	154.0	4.7 (GB)
Torso	10	178.0	5.3 (GB)

Table 1: *MPM datasets.* The results of particle simulations from a range of physical problems are featured throughout this work. Using our approach, investigators can explore large, time-varying particle datasets like these on their desktop systems.

principal component analysis (PCA) can be reconstructed by the graphics hardware and mapped to the particles during interactive rendering. Similarly, image-based silhouettes can be computed by the graphics hardware and applied to the visualizations at interactive rates. In a two-pass algorithm, the graphics hardware first renders the geometry to a depth buffer, and then determines the edges by applying a Laplacian kernel to the results. These advanced visualization features are illustrated in Figure 5.

4 Results

We explore the utility and performance of our system using a variety of MPM simulation datasets. We emphasize, however, that our approach is applicable to the results of other particle-based methods such as particle-in-cell, molecular dynamics, smoothed particle hydrodynamics, element-free Galerkin, and discrete element methods. Table 1 lists the properties of the example datasets, and Figures 4 and 5 illustrate the data exploration and advanced visualization features of our system using these datasets.

4.1 Promoting Insight into Particle-Based Simulations

Visualization of simulation data by an investigator typically serves one of three purposes, data analysis, code development and generation of presentation or publication quality images. The ability to interact with high quality renderings of large scale particle datasets serves each of these purposes. For example, simulations using the foam geometries (as shown in Figure 4c) involve compressing the samples with the rigid plate at the top of the model. Compression is carried out to rouhgly the level of "full densification", or the point at which the compressed material occupies a volume equal to the initial volume of the constituent material. Throughout the simulation, one can gather as output not only the various states of deformation of the foam, but also the reaction force at the surfaces bounding the domain. By using interactive visualization tools, it is possible to correlate specific events in the simulation with the reaction force at the boundaries. An example of such an event is the collision of one foam strut either with another strut or with the domain boundary. By determining visually exactly when these events occur, it is possible to recognize what relationship they have with the features on the force-displacement curve.

While code development ideally refers to adding new capabilities, debugging ill-behaved solutions is another obvious, but important, consequence of highly accessible particle visualization tools. Another important application of the MPM code involves studying the effect of mechanical loading on angiogenesis, the growth of micro blood vessels in a collagen matrix. There, initial geometry is constructed by transforming three-dimensional images of vessels grown in-vitro into a particle representation. Images of samples that have been selectively stained with a fluorescent tag are collected using scanning-confocal microscopy. These images are then thresholded according to intensity, and for each bright voxel, a particle with properties of vessel is created, and for voxels below the threshold value, a particle with properties of collagen is created. Mechanical loading is simulated by prescribing the displacment of a rigid plate at the top of the sample. Ultimately, the goal is to correlate sprouting locations in a specific sample with the stress patterns predicted by simulation. Early in this investigation, it was recognized that the computed reaction force holding the sample in place during loading was behaving in an unusual and unpredictable manner. Only upon viewing the particle data directly (13.6 million particles in this case) was it evident that as particles moved through the computational grid, some were subject to very different levels of strain when they should have been nearly uniform. Recognition of this behavior led to modification of the algorithm, as described by Guilkey et al. [18]. When this same data was interpolated to the computational grid and volume rendered, these issues were not evident because interpolation had smoothed the non-uniformities.

Our approach to visualization of data from particle-based simulations brings powerful data analysis and debugging tools to the investigator's desktop, enabling interaction with millions of particles across the entire simulation. Moreover, because scientists can interact with the whole dataset, a clear understanding of the particles' physical state, as well as their spatial relationship to the full computational domain, can be achieved. These characteristics also make generation of high quality movies and still images, for presentation and publication, fast and straightforward. In particular,

the speed allows a user to quickly find the optimal view for which all frames of a movie will convey the most information.

4.2 Characterizing Interactive Performance

Interactivity is a key component of the particle visualization process. As discussed in Section 3, the vertex and fragment processing capabilities of commodity graphics hardware, combined with software-based acceleration techniques, enable interactive visualization of large, time-varying datasets.

We examine the performance of our system under each of the three occlusion culling algorithms described in Section 3. We also compare performance against to additional algorithms: (1) an "exact" algorithm that renders the exact visible set in each frame, as determined by a preprocessing phase; and (2) an "ideal" algorithm that renders only the particles within visible nodes, as determined by HSW occlusion culling, without performing any visibility tests. The the exact algorithm provides an upper-limit for a given interactive session on a given platform (no rendering algorithm running on the same hardware can produce the correct results any faster), which ideal algorithm provides an upper-limit on performance with respect to the given spatial subdivision structure (no occlusion culling algorithm using the same structure can be faster).

For the tests, we utilize a dual Opteron system with 8 GB of physical memory and an Nvidia 7800 GT graphics card. Using prerecorded interactive sessions, images were rendered at 1024x768 pixels using an octree (maximum depth restricted to 4). Table 2 shows the results for two datasets, Fireball and Thunder, and each algorithm, averaged over the session. Figure 6 details the behavior of the Thunder dataset graphically.

CHC/CHC-TV typically achieve the best performance, despite rendering more particles per frame than HSW. This result is a direct consequence of the algorithm's ability to hide the high latency of hardware-based occlusion queries by rendering previously visible geometry while waiting on outstanding query results. Moreover, CHC-TV maintains coherence during periods of temporal animation, and as as result, provides better average performance over the naive implementation of CHC for time-varying data.

5 Conclusions and Future Work

Effective visualization of particle-based simulation data presents many interesting and difficult challenges, both from an applications science and scientific visualization point of view. We have described an approach to rendering large, time-varying particle datasets using commodity graphics hardware on desktop systems at interactive rates. We have demonstrated the effectiveness of our approach using the results of several MPM simulations, but our approach is applicable to the results of other particle-based simulation methods as well.

In addition, we have characterized the performance of three occlusion culling algorithms in the context of particle visualization. This exploration led to a simple modification of the coherent hierarchical culling algorithm that often provides better performance when rendering time-varying data.

Our approach performs competitively with current interactive ray tracing systems while providing the same capabilities for the data analysis, feature detection, and code development tasks that investigators perform. Moreover, our system runs on hardware that is a fraction of the cost of large multi-processor shared-memory platforms, making particle visualization and data exploration more accessible. Our system advances the current state-of-the-art by bringing visualization of large, time-varying particle datasets to the desktop.

There remain several open areas of interest. For example, the results of rendering the exact visible set indicate that improving performance may be possible by obtaining a better approximation to the visible set in each frame than current techniques provide. One possibility is to explore more aggressive occlusion culling techniques that exploit assumptions about the structure of the underlying data. In addition, performance is influenced by the characteristics of the spatial subdivision structure used during traversal, and some structures perform better than others under certain conditions. Adaptively choosing the structure used in a given frame using performance-based heuristics may be one way to tailor interactive performance to the demands of an interactive session at runtime. Exploring the trade-offs among additional spatial subdivision structures may reveal more insights useful to such an approach. 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 be useful on desktop platforms. Commodity graphics hardware has long been used for volume rendering, so combining this visualization modality with the particle visualization methods we have described would be valuable.

Acknowledgments

We thank Biswajit Banerjee of the University of Utah for providing the Bullet and Cylinder datasets (Figures 1, 3, 4, and 5). We are indebted to Jerry Seidler and Erin Miller, both of the University of Washington, and to Scott Bardenhagen, of Los Alamos National Laboratory, for the Foam dataset (Figure 4). Finally, we thank the Scientific Computing and Imaging Institute's Render Group for the useful feedback on early drafts of this paper. This work was funded by the DOE ASC program.

References

[1] J. M. Airey, J. H. Rohlf, and Jr. F. P. Brooks. Towards image realism with interactive update rates in complex virtual building environments. In *Symposium on Interactive 3D Graphics*, pages 41–50, 1990.

[2] B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. i. genearl method. *The Journal of Chemical Physics*, 31(2):459–466, August 1959.

[3] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. In *Proceedings of IEEE Visualization 2004*, pages 243–250, 2004.

[4] S. G. Bardenhagen, J. U. Brackbill, and D. Sulsky. The material-point mehod for granular mechanics. *Comput. Methods Appl. Mech. Engrg.*, 187:529–541, 2000.

[5] D. Bartz, M. Meissner, and T. Huttner. Extending graphics hardware for occlusion queries in opengl. In *Proceedings fo the 1998 Workshop on Graphics Hardware*, pages 97–104, 1998.

[6] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

[7] 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*, To appear, May 2006.

[8] J. Bittner and V. Havran. Exploiting coherence in hierarchical visibility algorithms. *Journal of Visualization and Computer Animation*, 12:277–286, 2001.

[9] J. Bittner, V. Havran, and P. Slavik. Hierarchical visibility culling with occlusion trees. In *Proceedings of Computer Graphics International '98*, pages 207–219, 1998.

[10] J. Bittner, M. Wimmer, H. Piringer, and W. Purgathofer. Coherent hierarchical culling: Hardware occlusion queries made useful. *Computer Graphics Forum (Proc. of Eurographics 2004)*, 23(3):615–624, September 2004.

[11] S. Coorg and S. Teller. Temporally coherent conservative visibility. In *Proceedings of the Twelfth Annual ACM Symposium on Computational Geometry*, pages 78–87, 1996.

[12] S. Coorg and S. Teller. Real-time occlusion culling for models with large occluders. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 231–238, 1997.

[13] P. A. Cundall. A computer model for simulating progressive large scale movements in block rock systems. In *Symp. Intl. Society of Rock Mechanics*, 1971.

[14] R. A. Gingold and J. J. Monaghan. Smoothed particle hydronamics—theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices*, 181:375–389, November 1977.

[15] N. Greene, M. Kass, and G. Miller. Hierarchical z-buffer visibility. In *Proceedings of Siggraph '93*, pages 231–238, 1993.

[16] C. Gribble and S. Parker. Enhancing interactive particle visualization with advanced shading models. In *Proceedings of the Third Symposium on Applied Perception in Graphics and Visualization*, Submitted for publication, 2006.

[17] J. Guilkey and J. Weiss. An implicit time integration strategy for use with the material point method. In *Proceedings from the First MIT Conference on Computational Fluid and Solid Mechanics*, June 2001.

[18] J. E. Guilkey, J. A. Hoying, and J. A. Weiss. Computational modeling of multicellular constructs with the material point method. *Journal of Biomechanics*, To appear, 2006.

[19] J. E. Guilkey and J. A. Weiss. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *Int. J. Num. Meth. Eng.*, 57:1323–1338, 2003.

[20] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modelling and Visualization 2003*, pages 245–252, November 2003.

[21] F. H. Harlow. The particle-in-cell method for fluid dynamics. *Methods for Computational Physics*, 3:319–343, 1964.

[22] K. Hillesland, B. Salomon, A. Lastra, and D. Manocha. Fast and simple occlusion culling using hardware-based depth queries, September 2002.

[23] J. T. Klosowski and C. T. Silva. The prioritized-layered projection algorithm for visible set estimation. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):108–123, 2000.

[24] M. Krogh, J. Painter, and C. Hansen. Parallel sphere rendering. *Parallel Computing*, 23(7):961–974, 1997.

[25] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[26] T. Leyvand, O. Sorkine, and D. Cohen-Or. Ray space factorization for from-region visibility. *ACM Transactions on Graphics*, 22(3):595–604, 2003.

[27] K. Liang, P. Monger, and H. Couchman. Interactive parallel visulization of large particle datasets. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 111–118, 2004.

[28] 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*, pages 163–169, 1987.

[29] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, December 1977.

[30] J. J. Monaghan. Smoothed particle hydronamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.

[31] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2004.

[32] D. Sulsky, S. Zhou, and H. L. Schreyer. A particle method for history dependent materials. *Computer Methods in Applied Mechanical Engineering*, 118:179–196, 1994.

[33] D. Sulsky, S. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87:236–252, 1995.

[34] S. J. Teller and C. H. Sequin. Visibility preprocessing for interactive walkthroughs. In *Proceedings of Siggraph '91*, pages 61–69, 1991.

[35] J. R. Williams, G. Hocking, and G. G. W. Mustoe. The theoretical basis of the discrete element method. In *Numerical Methods of Engineering, Theory and Applications*, January 1985.

[36] P. Wonka, M. Wimmer, and D. Schmalstieg. Visibility preprocessing with occluder fusion for urban walkthroughs. In *Proceedings of Eurographics Workshop on Rendering*, pages 71–82, 2000.

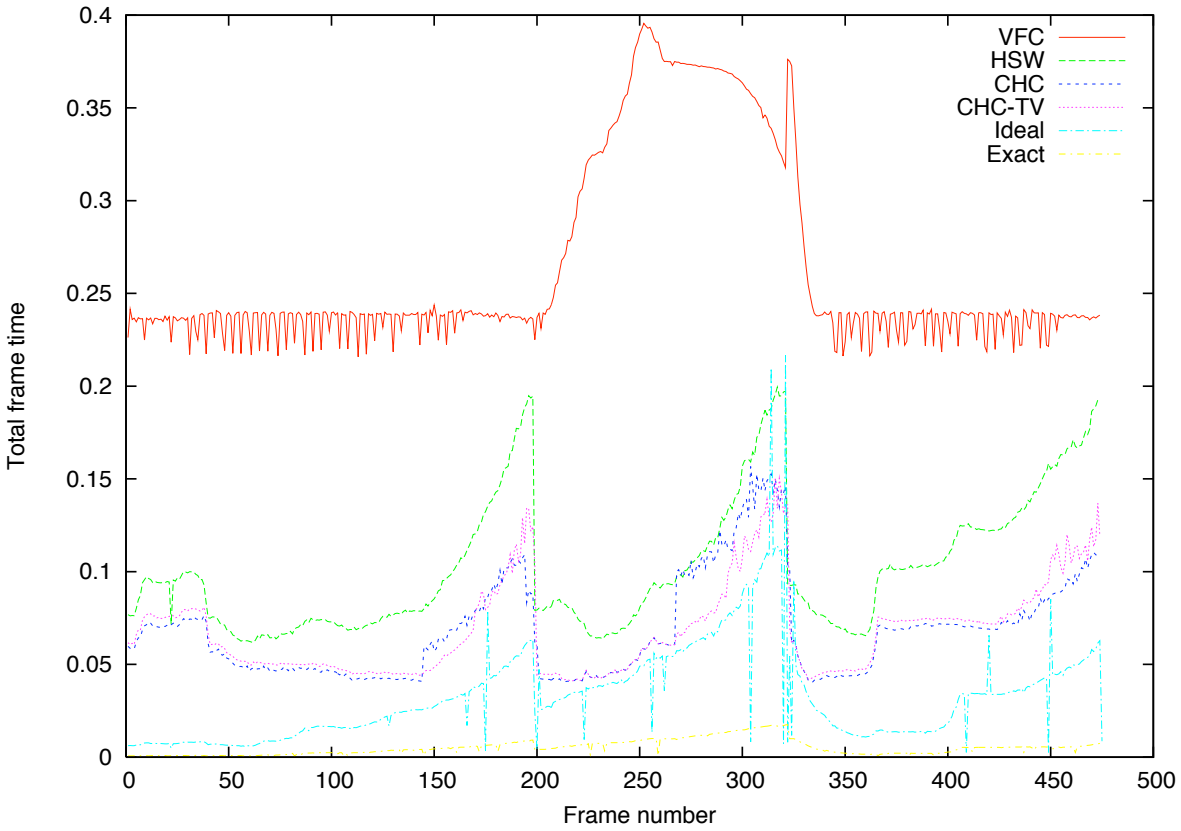
[37] P. Wonka, M. Wimmer, and F. X. Sillion. Instant visibility. In *Proceedings of Eurographics '01*, pages 411–421, 2001.

[38] P. Zemcik, P. Tisnovsky, and A. Herout. Particle rendering pipeline. In *Proceedings of the 19th Spring Conference on Computer Graphics*, pages 165–170, 2003.

[39] H. Zhang, D. Manocha, T. Hudson, and III K. E. Hoff. Visiblity culling using hierarchical occlusion maps. In *Proceedings of Siggraph '97*, pages 77–88, 1997.

Dataset	Method	Occlusion Queries	Wait time [ms]	Particles Rendered	Frame time [ms]	Speedup
Fireball	VFC	—	—	643296	73.76	1.00
	HSW	1541	46.07	218321	79.22	0.93
	CHC	1423	7.40	250246	52.82	1.40
	CHC-TV	1392	6.11	244200	53.89	1.36
	Ideal	—	—	210453	26.03	2.83
	Exact	—	—	27812	4.70	15.69
Thunder	VFC	—	—	2689380	264.48	1.00
	HSW	1471	52.08	357709	102.69	2.58
	CHC	1488	5.29	394507	68.99	3.83
	CHC-TV	1349	4.02	382918	69.06	3.83
	Ideal	—	—	358447	33.86	7.82
	Exact	—	—	42114	5.21	50.76

Table 2: *Comparing occlusion culling algorithms.* Several pertinent characteristics of each algorithm have been averaged over the given interactive session. In general, CHC/CHC-TV provide the best performance of the online occlusion culling algorithms, despite rendering more particles than HSW. Better performance results from the algorithm's ability to hide the high-latency of occlusion queries behind useful work.



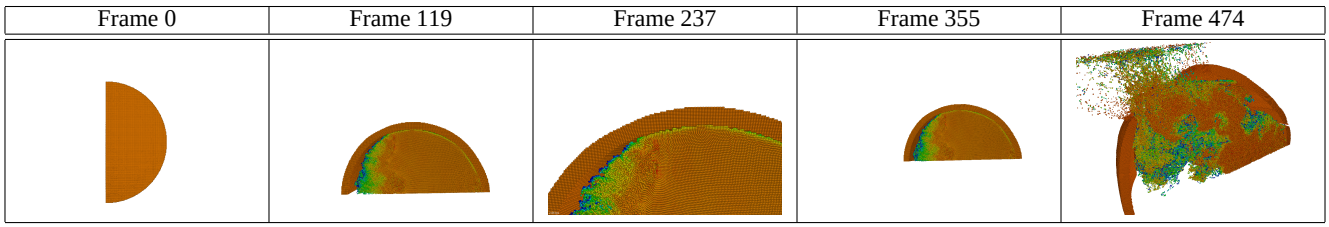


Figure 6: *Characterizing interactive performance.* This graph shows the results of rendering the Thunder dataset (55 time steps, 154 million particles) over a prerecorded interactive session. Our systems achieves roughly 10 frames per second on an Nvidia 7800 GT graphics card and offers a highly interactive environment for navigating and exploring large, time-varying particle-based simulation datasets.