British HCI 2006 Workshop on Combining Visualization and Interaction to Facilitate Scientific Exploration and Discovery (September 2006) (to appear)

Visualizing Particle-Based Simulation Datasets on the Desktop

Christiaan P. Gribble, Abraham J. Stephens, James E. Guilkey, and Steven G. Parker
Scientific Computing and Imaging Institute, University of Utah
50 S Central Campus Drive, MEB 3490
Salt Lake City, Utah 84105
United States of America
cgribble@sci.utah.edu, abe@sci.utah.edu, james.guilkey@utah.edu, sparker@sci.utah.edu

We present an approach to rendering large, time-varying particle-based simulation datasets using programmable graphics hardware on desktop computer systems. Particle methods are used to model a wide range of complex phenomena, and effective visualization of the resulting 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 highlight the critical components of our approach, and introduce an extension to the coherent hierarchical culling algorithm that often improves temporal coherence and leads to better average performance for time-varying datasets. Our approach performs competitively with current particle visualization systems based on interactive ray tracing that require tightly coupled supercomputers. Moreover, our system runs on hardware that is a fraction of the cost of these systems, making particle visualization and data exploration more accessible. We thus advance the current state-of-the-art by bringing visualization of particle-based simulation datasets to the desktop.

Key words: particle-based simulation, particle visualization, interactive rendering, occlusion algorithms

1. INTRODUCTION

Over the past several years, commodity graphics processing units (GPUs) have become an attractive option for visualizing a wide variety of scientific datasets. This hardware has evolved quite rapidly from a fixed-functionality pipeline into a programmable machine with powerful vertex and fragment processors. Moreover, extreme parallelism and the advent of high-level programming languages for this hardware have led to flexible processors providing compute power in the multi-GFLOPS to TFLOPS range [1].

We investigate the use of programmable GPUs to interactively visualize the results of particle-based simulations on desktop computer systems. Particle methods are commonly used to simulate complex phenomena in many scientific domains, including astronomy, biology, chemistry, physics, and others. Using particle-based simulation techniques, 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. Frequently, millions of particles are required to capture the behavior of a system accurately, leading to very large, very complex datasets.

Investigators use particle visualization to assist efforts in data analysis and feature detection, as well as in debugging ill-behaved solutions. As a result, effective visualization of particle-based simulation data requires communication of subtle changes in the three-dimensional structure, spatial organization, and qualitative trends within a simulation as it evolves. Most importantly, an effective visualization method will enable investigators to interrogate their data interactively.

Unfortunately, the size and complexity of typical particle datasets make interactive visualization a difficult task. Recently, Bigler et al. [2] have described an approach that is based on interactive ray tracing and requires tightly coupled supercomputing platforms. This system represents the current state-of-the-art in particle-based simulation data visualization, and while the approach satisfies the requirements of effective particle visualization, the hardware costs are prohibitive and thus impede accessibility.

We address this issue and present an approach to rendering large, time-varying particle datasets using programmable graphics hardware on desktop systems. In particular, we combine point sprite rendering and software-based acceleration techniques to achieve interactive frame rates, and introduce an extension to the coherent hierarchical culling [3] algorithm that provides improved temporal coherence and better average performance for time-varying datasets. Our system also supports enhanced data exploration and visualization techniques and offers greater accessibility than previous systems.

2. BACKGROUND AND RELATED WORK

Our approach to particle visualization leverages and extends results from the computer graphics and scientific visualization literature in order to achieve a system featuring data exploration capabilities and performance competitive with systems based on interactive ray tracing. After providing a short overview of particle visualization methodology, we briefly review the relevant research related to our approach.

2.1 Particle Visualization

As noted, particle visualization is typically used for data analysis and code development tasks. One approach to particle visualization projects the particle values to a three-dimensional grid, and the transformed data is then visualized using standard techniques such as isosurface rendering [4] and direct volume rendering [5]. Grid-based representations are suitable for some, but not all, particle 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. Investigators 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 velocities may be invalid, but the influence of such particles can be masked by interpolation. This possibility may confound efforts to validate simulation codes. Finally, interpolation and isosurface extraction can be very time-consuming, particularly for large datasets with many time steps.

Particles can also be represented directly by simple, iconic shapes called glyphs. Glyph-based visualizations are able to convey both the fine details within the simulation data as well as the large-scale three-dimensional structure of the entire domain. In many scientific visualization applications, directed arrows or other glyphs are used to convey important information from the simulation. For particle-based simulation data, however, a simple sphere or ellipsoid is often the most natural representation of an individual particle. As a result, our approach to particle visualization leverages the point sprite rendering capabilities of modern GPUs to render large numbers of spherical glyphs in an efficient manner.

2.2 Efficient Sphere Rendering

Several efforts have investigated techniques to render large numbers of spheres efficiently, from using massively parallel processors [6] and visualization clusters [7] to designing custom hardware [8]. We leverage the simple solution to ray/ellipsoid intersection using the vertex and fragment processors of programmable graphics hardware described by Gumhold [9]. This approach can be combined with software-based acceleration methods to reduce the rendering workload in each frame. For example, view-frustum culling (VFC) is a simple technique that quickly culls objects that lie outside of the view volume. However, VFC does not prevent occluded objects from being sent to the graphics hardware, so an area in image-space may be covered more than once. This problem, which is called overdraw, wastes computational resources in both the vertex and fragment processing stages of the rendering pipeline.

Occlusion culling algorithms attempt to address the overdraw problem. In particular, *from-point* (online) algorithms apply a visibility computation for each viewpoint encountered during rendering. Many such algorithms employ hardware occlusion queries [10] for these visibility tests. For example, Hillesland et al. [11] use these queries and a uniform, possibly nested, grid to determine visible geometry. This method exploits neither spatial nor temporal coherence during rendering, however, and is restricted to regular spatial subdivision data structures. Coherent hierarchical culling (CHC) [3] overcomes both of these problems and provides an adaptation of earlier improvements [12] that is tailored for hardware occlusion queries. We explore the performance characteristics of several software-based acceleration techniques, including CHC, in the context of particle visualization. We also introduce a simple extension to the basic CHC algorithm that often improves performance with time-varying geometry.

3. SYSTEM IMPLEMENTATION

Our approach to particle visualization offers advanced data exploration capabilities and interactive performance on desktop systems that are consistent with those provided by interactive ray tracing systems. The critical components of our approach include: (1) rendering high quality particle glyphs efficiently, (2) achieving interactive performance with programmable graphics hardware, (3) handling time-varying data while maintaining interactivity, and (4) supporting enhanced data exploration and visualization techniques. After highlighting some important details of these components, we characterize the interactive performance of our implementation by presenting results from two datasets that stress the capabilities of our system.

3.1 Rendering High Quality Particle Glyphs

The need to simultaneously visualize both large- and small-scale structures within a dataset requires that each particle be rendered using a high quality representation. Such a representation permits useful interrogation of individual particles, as well as application of advanced visualization techniques to enhance the perception of the three-dimensional structure and spatial relationships across the entire computational domain. One approach would simply render a highly tessellated, view-aligned hemisphere for each particle. However, typical datasets contain millions of particles, so the required geometry would quickly overwhelm the GPU and result in poor performance.

To alleviate this issue while maintaining visual quality, we employ view-aligned billboards as the base primitive. Each billboard is rendered efficiently using the point sprite rendering capabilities of modern GPUs. A vertex corresponding to the position of each particle specifies an individual point sprite, and per-vertex attributes control other aspects of its representation, including the radius and color. Vertex and fragment programs manipulate this data to render a high quality representation of each particle in an efficient manner, as illustrated in Figure 1.

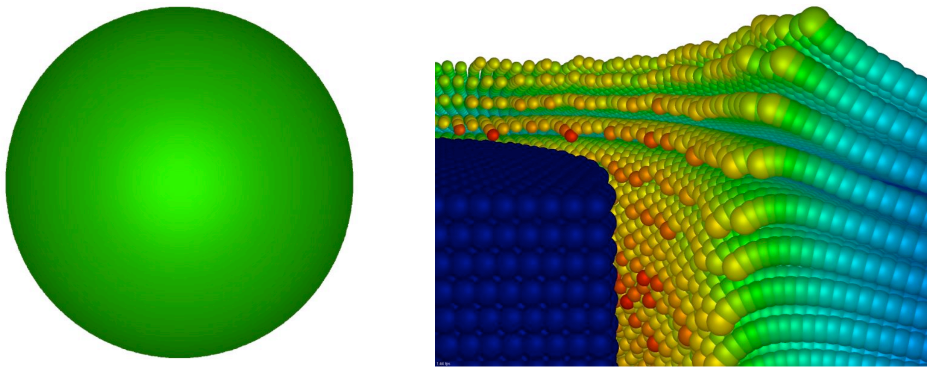


FIGURE 1: *High quality particle rendering with point sprites*. View-aligned billboards serve as the base primitive. An individual particle is specified by a vertex position and other per-vertex attributes that control its representation (left). Efficiently rendering high quality particle glyphs enables simultaneous visualization of both large- and small-scale features within the data (right).

3.2 Achieving Interactive Performance

The particle rendering process described above can be combined with software-based acceleration techniques to reduce the workload in each frame. For example, we use view-frustum culling to provide a first approximation to the set of potentially visible particles. Particles that lie outside the view volume are quickly culled, saving the cost associated with rendering particles that will not contribute to the final visualization. VFC can be combined with other approaches that employ hardware-based occlusion queries, for example, hierarchical stop-and-wait occlusion culling (HSW). In this algorithm, a spatial subdivision structure is traversed, depth first, and occlusion queries are issued for nodes that overlap or are contained within the current view frustum. At each potentially visible node, a query is issued and the CPU waits for the results. When the results are available, the algorithm proceeds by resuming traversal (interior node) or by rendering the associated particles (leaf node) if the node is visible; however, if the node is occluded, the subtree (interior node) or particles (leaf node) are culled.

Hardware occlusion queries typically have a high latency and can lead to load balancing problems between the CPU and the GPU. Coherent hierarchical culling [3] is designed to address this issue. CHC exploits temporal coherence in the visibility classification of scene geometry by reusing the results of occlusion queries issued in the previous frame. CHC 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 structure.

3.3 Handling Time-Varying Data

When rendering time-varying data, each time step is handled individually: the data structures required to render large numbers of particles efficiently are constructed and maintained for each time step. The size of these structures is typically very small compared to the size of the data itself, so the overhead of maintaining separate structures for each time step is quite low. The data is animated temporally by simply rendering all of the times steps across successive frames.

The time-varying nature of particle datasets 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. Unfortunately, the naïve use of the CHC algorithm, in which separate visibility information is maintained for each time step, results in a loss of frame-to-frame coherence. In particular, the visibility information for a particular time step quickly becomes out-of-date during periods of temporal animation because many frames may have been rendered before that time step is visible again. To overcome this issue, we introduce an extension to the basic CHC algorithm that stores only one set of visibility information across all time steps. This information is always used to estimate the currently visible geometry, even if the estimate is based on the results of a 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 the visibility classification of the geometry tends to change slowly between time steps.

3.4 Supporting Enhanced Data Exploration and Visualization

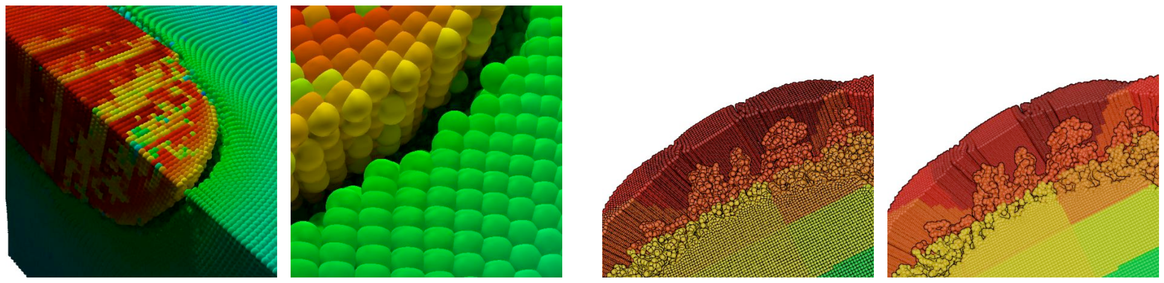
As the preceding discussion indicates, visualizing large, time-varying particle datasets presents many challenges to achieving interactive performance. Most importantly, however, the complexity inherent in the data itself may confound attempts to correctly interpret the simulation results.

Basic data exploration capabilities such as interactive viewing and lighting enable investigators to identify and interrogate specific features within the data more easily. In addition, more advanced data analysis tools like color mapping and parameter range culling allow investigators to interrogate the data based on the state parameters from the simulation that are associated with each particle. Our system supports each of these features, all of which can be controlled interactively at run time.

Advanced particle visualization techniques are also supported. For example, recent research has shown that advanced shading models such as ambient occlusion and physically based diffuse interreflection can enhance the perception of three-dimensional structure and spatial relationships within particle datasets [13]. Our system supports these shading models via multi-pass fragment processing. In particular, compressed precomputed luminance textures [13] can be reconstructed on the GPU and mapped to the particles during interactive rendering. Similarly, the GPU can compute image-based silhouettes [2] and apply the results to the visualizations at interactive rates. Figure 2 illustrates the results of these techniques for typical particle datasets.

3.5 Characterizing Interactive Performance

Interactivity is a key component of the data analysis process. We compare the performance of the CHC-TV algorithm to several other software-based acceleration techniques, including view-frustum culling, hierarchical stop-and-wait occlusion culling, and the original CHC algorithm. To provide a bound on


(a) Compressed precomputed luminance textures

(b) Image-based silhouettes

FIGURE 2: *Enhancing particle visualization*. Advanced shading models such as 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 (b). These advanced visualization modes are supported by our approach using multi-pass fragment processing.

achievable performance, we also investigate two additional algorithms: (1) an “exact” algorithm that renders the exact visible set in each frame, as determined by a lengthy preprocessing phase; and (2) and “ideal” algorithm that renders only the particles contained within the visible nodes of a spatial subdivision structure as determined by HSW occlusion culling, also during a preprocessing phase. The exact algorithm provides a lower bound on frame time for a given interactive session (no rendering algorithm running on the same hardware can produce the correct results any faster), while the ideal algorithm provides a lower bound 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 processor system with 8 GB of physical memory and an NVIDIA 7800 GT graphics card. Measurements were gathered for images rendered at 1024x768 pixels using pre-recorded interactive sessions with the two datasets, Container and Fireball, that are illustrated in Figure 3. Table 1 shows the results for each dataset and each rendering algorithm averaged over an interactive session, and Figure 4 details the behavior of the Container dataset graphically.

When a dataset exhibits a high degree of self-occlusion, as does the Container dataset, CHC-TV typically provides the best performance despite rendering more particles per frame than HSW. This result is a direct consequence of its ability to hide high latency hardware-based occlusion queries behind useful work. Moreover, CHC-TV maintains coherence during periods of temporal animation and thus provides better average performance over the naïve implementation of CHC for time-varying data. However, if a dataset does not exhibit a high degree of self-occlusion (as in the case of the Fireball dataset, for example), the visibility tests required by the occlusion culling algorithms add overhead, and simple view frustum culling may provide better performance.

4. DISCUSSION

Visualization of simulation data typically serves one of three purposes: data analysis, code development, or generation of presentation and publication quality images. An interactive visualization process enhances these tasks by enabling users to identify and explore the salient feature of their data more effectively.

One example of particle visualization in data analysis involves simulations of foam compression. In these simulations, a rigid plate compresses a foam material sample to roughly the level of “full densification,” or the point at which the total compressed volume is equal to the initial volume of the uncompressed sample. 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. Using interactive visualization, it is possible to correlate specific events in the simulation with the reaction force at the boundaries, for example, 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, leading to specific insights such as those described by Bardenhagen et al. [14].

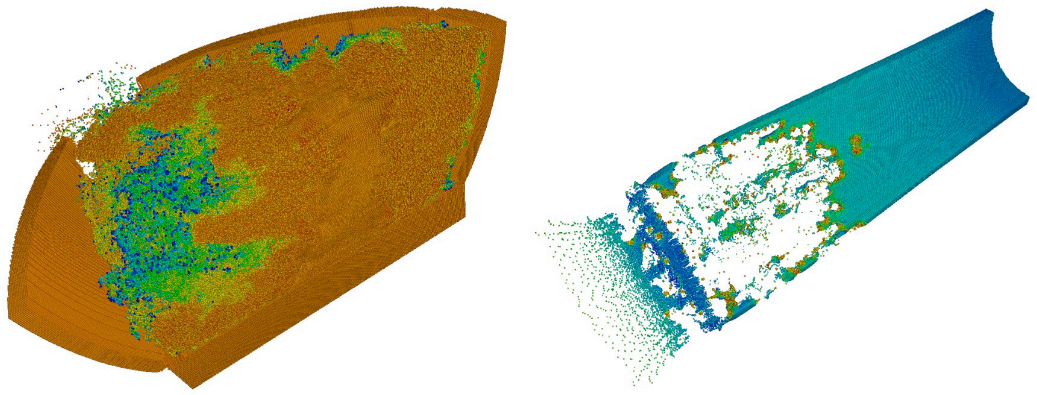


FIGURE 3: *Particle datasets used in performance testing*. The Container dataset (left) features densely packed particles and exhibits a high degree of self-occlusion. This dataset is composed of 50 time steps, each with roughly 2.8 million particles. Occlusion culling algorithms typically provide the best performance for datasets with similar properties. In contrast, these algorithms simply add overhead to the rendering process and lead to lower performance for datasets that do not exhibit a high degree of self-occlusion, as in the case of the Fireball dataset (right), which is composed of 20 time steps, each with approximately 1 million particles.

Another important application of particle visualization involves studying the effects of mechanical loading on angiogenesis, or the growth of micro blood vessels in a collagen matrix. Here, 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. Using these images, particles with the material properties of vessel are created for voxels with intensities above a given threshold, while particles with properties of collagen are created for the remaining voxels. Mechanical loading is simulated by prescribing the displacement of a rigid plate at the top of the sample. The goal of this study is to correlate sprouting locations in a specific sample with the stress patterns predicted by simulation.

While code development ideally refers to adding new functionality, the ability to debug ill-behaved solutions is another obvious, but important, consequence of highly accessible interactive particle visualization. Early in the angiogenesis investigation, 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 (13.2 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 these levels should have been nearly uniform. Recognition of this behavior led to an important modification of the simulation algorithm, as described by Guilkey et al. [15]. When this same data was interpolated to the computational grid and visualized by volume rendering the transformed data, these issues were not evident because interpolation had smoothed the non-uniformities.

Finally, an interactive environment also makes generation of high quality animations and still images for presentation or publication fast and straightforward. In particular, interactivity allows users to quickly identify optimal views in which each image will convey the most pertinent information.

Our approach to visualization of particle datasets brings these powerful data analysis and code development tools to the investigator's desktop, enabling interaction with millions of particles across the entire simulation. Moreover, because an investigator can interact with the whole dataset, a clear understanding of the state of each particle, as well as its relationship to the full computational domain, can be achieved. As particle-based simulation techniques continue to advance, the combination of interactive visualization and so-called "human-in-the-loop" problem solving environments may provide opportunities for additional insights through computational steering.

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 perspective. We have described an approach to rendering large, time-varying particle datasets at interactive rates using programmable graphics hardware on desktop systems. Our approach performs competitively with current interactive ray tracing systems and offers many of the same capabilities that aid the data analysis, feature detection, and code development tasks investigators perform. Moreover, our system runs on hardware that is a fraction of the cost of previous systems, which makes 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.

Several areas of interest remain open. For example, 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. Programmable graphics hardware has long been used for volume rendering, so combining this visualization modality with the particle visualization method we have described would be valuable. Improved performance is also of interest. Exploring more aggressive occlusion culling algorithms that exploit assumptions about the structure of the underlying data is one of many possible research directions that may improve interactive performance.

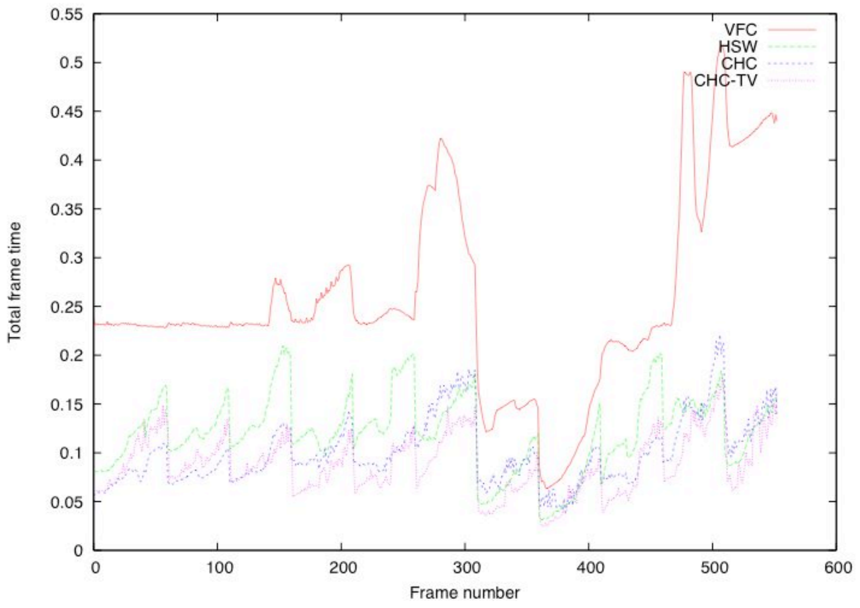
This work was supported by the U.S. Department of Energy through the Center for the Simulation of Accidential Fires and Explosions, under grant W-7405-ENG-48.

REFERENCES

[1] Polkowski, D. E. (2006) ATI's Radeon X1900 Heats Up with 48 Shader Units. http://www.tomshardware.com/2006/01/24.
[2] Bigler, J., Guilkey, J., Gribble, C., Parker, S., and Hansen, C. (2006) A Case Study: Visualizing Material Point Method Data. Proceedings of the Eurographics/IEEE Symposium on Visualization, May, pp. 299—305.
[3] Bittner, J., Wimmer, M., Piringer, H., and Purgathofer, W. (1994) Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful. Computer Graphics Forum (Proceedings of Eurographics 1994), 23, 615—624.
[4] Lorensen, W. E., and Cline, H. E. (1987) Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proceedings of the International Conference on Computer Graphics and Interactive Techniques, pp. 163—169.
[5] Levoy, M. (1988) Display of Surfaces from Volume Data. IEEE Computer Graphics and Applications, 8, 29—37.
[6] Krogh, M., Painter, J., and Hansen, C. (1997) Parallel sphere rendering. Parallel Computing, 23, 961—974.
[7] Liang, K., Monger, P., and Couchman, H. (2004) Interactive Parallel Visualization of Large Particle Datasets. Eurographics Symposium on Parallel Graphics and Visualization, pp. 111—118.
[8] Zemcik, P., Tisnovsky, P., and Herout, A. (2003) Particle Rendering Pipeline. Proceedings of the 19th Spring Conference on Computer Graphics, pp. 165—170.
[9] Gumhold, S. (2003) Splatting Illuminated Ellipsoids with Depth Correction. Proceedings of 8th International Fall Workshop on Vision, Modelling, and Visualization, November, pp. 245—252.
[10] Bartz, D., Meissner, M., and Huttner, T. (1998) Extending Graphics Hardware for Occlusion Queries in OpenGL. Proceedings of the 1998 Workshop on Graphics Hardware, pp. 97—104.
[11] TR02-039. (2002) Fast and Simple Occlusion Culling Using Hardware-Based Depth Queries. Department of Computer Science, University of North Carolina—Chapel Hill.
[12] Bittner, J. and Havran, V. (2001) Exploiting coherence in hierarchical visibility algorithms. Journal of Visualization and Computer Animation, 12, 277—286.
[13] Gribble, C., and Parker, S. (2006) Enhancing Interactive Particle Visualization with Advanced Shading Models. Proceedings of the Third Symposium on Applied Perception in Graphics and Visualization, to appear.
[14] Bardenhagen, S. G., Brackbill, J. U., and Sulsky, D. (2000) The material point method for granular mechanics. Comput. Methods Appl. Mech. Engrg., 187, 529—541.
[15] Guilkey, J. E., Hoying, J. A., and Weiss, J. A. (2006) Computational modelling of multicellular constructs with the material point method. Journal of Biomechanics, to appear.

Dataset	Method	# occlusion queries	Wait time [ms]	# particles rendered	Frame time [ms]	Speedup
Container	VFC	—	—	2108550	254.93	—
	HSW	1448	71.64	351060	120.83	2.11
	CHC	1939	10.69	469418	100.52	2.54
	CHC-TV	**1407**	**5.81**	**414098**	**88.37**	**2.88**
	Ideal	—	—	351060	47.11	5.41
	Exact	—	—	35425	7.45	34.22
Fireball	**VFC**	**—**	**—**	**711964**	**70.90**	**—**
	HSW	1623	49.56	363731	93.52	0.76
	CHC	1732	14.05	459410	77.03	0.92
	CHC-TV	1576	8.87	482659	81.82	0.87
	Ideal	—	—	363731	30.34	2.34
	Exact	—	—	25795	3.39	20.91

TABLE 1: *Comparing occlusion culling algorithms*. Statistics about the performance of each algorithm have been averaged over the given interactive session. When a dataset exhibits a high degree of self-occlusion (Container), CHC-TV provides the best performance. However, if a dataset does not exhibit this property (Fireball), occlusion culling algorithms add unnecessary overhead that reduces interactive performance.



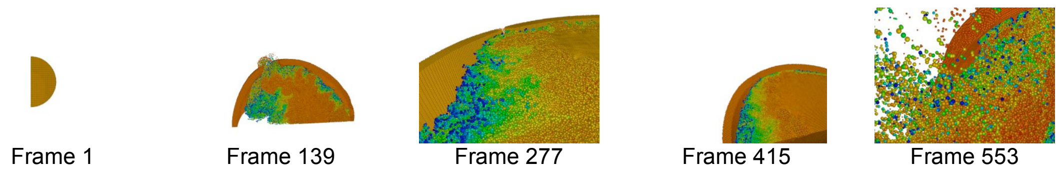


FIGURE 4: *Characterizing interactive performance*. This graph shows the results of rendering the Container dataset (50 time steps, 140 million particles) over a pre-recorded interactive session using four occlusion culling algorithms: view-frustum culling (VFC), hierarchical stop-and-wait occlusion culling (HSW), coherent hierarchical culling (CHC), and our new algorithm, coherent hierarchical culling for time-varying data (CHC-TV). Our approach maintains about 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 datasets.