Interactive GPU-based Visualization of Large Dynamic Particle Data

Synthesis Lectures on Visualization

Synthesis Lectures on Visualization publishes 50- to 100-page publications on topics pertaining to scientific visualization, information visualization, and visual analytics. Potential topics include, but are not limited to: scientific, information, and medical visualization; visual analytics, applications of visualization and analysis; mathematical foundations of visualization and analytics; interaction, cognition, and perception related to visualization and analytics; data integration, analysis, and visualization; new applications of visualization and analysis; knowledge discovery management and representation; systems, and evaluation; distributed and collaborative visualization and analysis.

A Guide to Visual Multi-Level Interface Design From Synthesis of Empirical Study Evidence

Heidi Lam and Tamara Munzner

2010

Interactive GPU-based Visualization of Large Dynamic Particle Data

Martin Falk, Sebastian Grottel, Michael Krone, and Guido Reina

www.morganclaypool.com

ISBN: 9781627052856 paperback
ISBN: 9781627054874 ebook

DOI 10.2200/S00731ED1V01Y201608VIS008

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON VISUALIZATION

Lecture #8
Series Editors: Niklas Elmqvist, *Yahoo! Labs*
David S. Ebert, *Purdue University*
Series ISSN
Print 2159-516X Electronic 2159-5178

Interactive GPU-based Visualization of Large Dynamic Particle Data

Martin Falk
Linköping University, Sweden

Sebastian Grottel
Technische Universität Dresden, Germany

Michael Krone
University of Stuttgart, Germany

Guido Reina
University of Stuttgart, Germany

SYNTHESIS LECTURES ON VISUALIZATION #8

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Prevalent types of data in scientific visualization are volumetric data, vector field data, and particle-based data. Particle data typically originates from measurements and simulations in various fields, such as life sciences or physics. The particles are often visualized directly, that is, by simple representants like spheres. Interactive rendering facilitates the exploration and visual analysis of the data. With increasing data set sizes in terms of particle numbers, interactive high-quality visualization is a challenging task. This is especially true for dynamic data or abstract representations that are based on the raw particle data.

This book covers direct particle visualization using simple glyphs as well as abstractions that are application-driven such as clustering and aggregation. It targets visualization researchers and developers who are interested in visualization techniques for large, dynamic particle-based data. Its explanations focus on GPU-accelerated algorithms for high-performance rendering and data processing that run in real-time on modern desktop hardware. Consequently, the implementation of said algorithms and the required data structures to make use of the capabilities of modern graphics APIs are discussed in detail. Furthermore, it covers GPU-accelerated methods for the generation of application-dependent abstract representations. This includes various representations commonly used in application areas such as structural biology, systems biology, thermodynamics, and astrophysics.

KEYWORDS

particles, visualization, GPU, molecules, rendering, visual analysis, object/image space methods, glyph rendering, atomistic visualization

Contents

Acknowledgments

This book is the result of more than a decade of active research. We have been supported by many people over the years, but we want to especially thank our common Ph.D. adviser, Professor Dr. Thomas Ertl, for support. We also want to thank our colleagues for their support and fruitful discussions as well as the undergraduate students supporting our work with theirs. Finally, our particular appreciation goes to the funding agencies that made this possible. The respective projects and agencies are, in no particular order:

- Landesstiftung Baden-Württemberg Project 688, "Massiv parallele molekulare Simulation und Visualisierung der Keimbildung in Mischungen für skalenübergreifende Modelle" (2004–2005)
- German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) collaborative research center SFB 716 as subprojects D.3 and D.4 (2007–2018)
- German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) Cluster of Excellence in Simulation Technology (EXC 310/1) (2008–2013)
- Centre Systems Biology in Stuttgart, Germany (2007–2010)
- Excellence Center at Linköping and Lund in Information Technology (ELLIIT) (2013–2015)
- Swedish e-Science Research Centre (SeRC) (2013–present)
- Federal Ministry of Education and Research (BMBF) Project No. 01IS14014, Scalable Data Services And Solutions (ScaDS, 2014–present)
- European Social Fund (ESF) Project No. 100098171, Visual and Interactive Cyber-physical Systems Control and Integration (VICCI, 2012–2014)

Martin Falk, Sebastian Grottel, Michael Krone, and Guido Reina
September 2016

Figure Credits

Figure 1.1 From: S. Grottel, P. Beck, C. Müller, G. Reina, J. Roth, H.-R. Trebin, and T. Ertl. Visualization of Electrostatic Dipoles in Molecular Dynamics of Metal Oxides. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2061–2068, 2012a. Copyright © 2012 IEEE. Used with permission.

Figure 1.2 Courtesy of Mathieu Le Muzic.

Figure 2.3 From: A. Knoll, Y. Hijazi, A. Kensler, M. Schott, C. Hansen, and H. Hagen. Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. *Computer Graphics Forum*, 28(1):26–40, 2009. Copyright © 2009 John Wiley & Sons, Inc. Used with permission.

Figure 3.3 From: S. Grottel, G. Reina, and T. Ertl. Optimized Data Transfer for Time-dependent, GPU-based Glyphs. In *IEEE Pacific Visualization Symposium (PacificVis 2009)*, pages 65–72, 2009a. Copyright © 2009 IEEE. Used with permission.

Figure 5.7 From: M. Le Muzic, J. Parulek, A. Stavrum, and I. Viola. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum*, 33(3):141–150, 2014. ISSN 1467-8659. Copyright © 2014 John Wiley & Sons, Inc. Used with permission.

Figure 5.8 Courtesy of Mathieu Le Muzic.

Figure 5.9 Courtesy of Mathieu Le Muzic.

Figure 5.10 Based on: M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *IEEE Visualization 2003*, 2003.

Figure 5.11 From: M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *IEEE Visualization 2003*, 2003. Copyright © 2003 IEEE. Used with permission.

Figure 5.12 From: M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *IEEE Visualization 2003*, 2003. Copyright © 2003 IEEE. Used with permission.

Figure 7.4 From: S. Grottel, G. Reina, C. Dachsbacher, and T. Ertl. Coherent Culling and Shading for Large Molecular Dynamics Visualization. *Computer Graphics Forum,* 29(3):953–962, 2010a. Copyright © 2010 John Wiley & Sons, Inc. Used with permission.

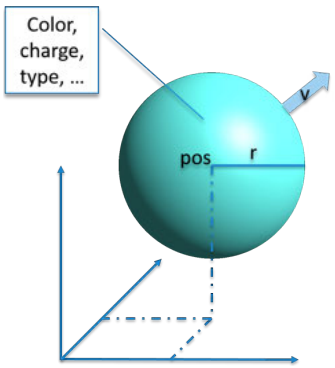
Figure 8.6 From: K. Bidmon, S. Grottel, F. Bös, J. Pleiss, and T. Ertl. Visual Abstractions of Solvent Pathlines near Protein Cavities. *Computer Graphics Forum*, 27(3):935–942, 2008. Copyright © 2008 John Wiley & Sons, Inc. Used with permission.

Figure 8.7 From: S. Grottel, G. Reina, J. Vrabec, and T. Ertl. Visual Verification and Analysis of Cluster Detection for Molecular Dynamics. *IEEE Transactions on Visualization and Computer Graphics,* 13(6):1624–1631, 2007. ISSN 1077-2626. Copyright © 2007 IEEE. Used with permission.

Figure 8.8 From: T. Ertl, M. Krone, S. Kesselheim, K. Scharnowski, G. Reina, and C. Holm. Visual Analysis for Space-Time Aggregation of Biomolecular Simulations. *Faraday Discussions*, 169:167–178, 2014. ISSN 1364-5498. Copyright © 2014 Royal Society of Chemistry. Used with permission.

CHAPTER 1

Introduction



Particle-based data has been an important subject of research for many years now. In its simplest form, such data consist of just a spatial coordinate. Each of those infinitesimal points can however be extended via additional attributes, hinting at some fitting representing geometry, color, or physical attributes like radius, velocity, mass, charge, or the like. A fundamental property of such data is the lack of any topological information. Each particle is an object of its own and does not require additional data to be represented meaningfully and individually. Relevant sources for this kind of data are mainly computational in nature, like molecular dynamics simulations and granular media simulations. Depending on the application area, the available attributes, and the research questions asked, a multitude of metaphors has been explored to depict such data.

1.1 SCOPE OF THIS LECTURE

This lecture will cover the interactive visualization of large, dynamic particle data consisting of discrete entities with their individual semantics. Given the computational power available today, the resulting data sets can grow quite large, currently in the order of 10^4 to 10^{12} [Eckhardt et al., 2013]. Simulating 10^{12} particles requires entire HPC systems (thousands of nodes) to be effective, so data of this magnitude is still uncommon at the time of writing. To further delineate the kind of data this lecture will focus on, it needs to be distinguished from what is nowadays called "big data." The most important additional property of such data over what we call "large" is that it cannot be effectively post-processed or, more specifically, visualized using a single workstation. This definition intentionally scales with the evolution of hardware and thus needs to be considered in context, but from a practical point of view, "large" is constant, describing feasibility for a current researchers' desktop workstation, so to say. Conversely, the approaches presented here enable visualization on a single machine and fulfill the additional requirements that a) only negligible preprocessing time is required, if any, thus enabling a domain scientist to directly inspect the latest simulation results; b) static data or data from at least a single time step must fit in host memory, so visualization is in-core with respect to the static case; c) the data must be streamed to the GPU since it is usually dynamic and it will not necessarily fit into GPU memory entirely; and d) consistent interactivity of the approach so the user can explore data and visualization parameters freely.

Since more elaborate visualization algorithms require more compute time, a tradeoff between data set size vs. computational complexity can be necessary to ensure interactive visualization. Thus, it is important to keep in mind that "large" data is relative to the visualization algorithm, not only to the hardware capabilities. Especially the interactivity requirement suggests the use of modern, programmable GPUs, where moderate implementation effort already results in highly interactive visualizations. Solutions that solely make use of the CPU exist, but here interactivity requires a much more involved implementation even for the baseline visualization [Wald et al., 2015]. In this case, large data sets are visualized using machines that have a considerably higher cost than the systems targeted by our approaches, a four-socket XEON system with 3TB RAM vs. what is basically a commercial off-the-shelf (COTS) gaming PC (consider that the list price of just a single XEON 8890v3 CPU as in the paper at the time of writing costs more than 7,000 USD, which exceeds the cost of the whole gaming PC by a factor of more than two). There are also approaches that solve the GPU memory problem at the cost of preprocessing and by reducing data accuracy somewhat, like the approach presented by Reichl et al. [2014], but the general idea will always be more severely memory limited or accuracy limited than when relying on the more ample host memory.

Application areas that make day-to-day use of simulations with the aforementioned properties include:

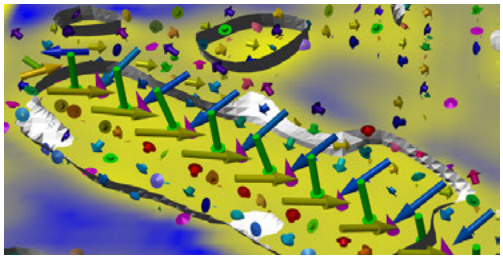


Figure 1.1: Direct and aggregated visualization of dipole behavior in crack propagation. From [Grottel et al., 2012a].

Physical sciences Many subfields in physics exploit particle simulations to study the behavior of systems under certain conditions. Thermodynamics and process engineering investigate phase interfaces and transitions, specifically evaporation and nucleation, as well as wetting or coating of surfaces, mixing of substances, and so on. Computer physics and material science investigate the inner structure of solids, i.e., crystal lattices and granular media, and how they behave under external influences like ablation, stress, rupture, and crack propagation.

Life sciences Biochemistry investigates the components of living organisms ranging from simplified scenarios like the interaction of proteins and solvent or DNA behavior over viruses to the processes in whole cells, like signal transduction. The combination of simulation and visualization has also been dubbed a *computational microscope* [Dror et al., 2012, Lee et al., 2009], since molecular simulation has nowadays advanced to a point where they can partially substitute wet lab experiments. With such a tool, the observation of small-scale systems is less costly than resorting to real-world experiments and expensive magnification equipment. Another significant advantage is that hazardous substances or environments are cheaper to handle and cannot pose any physical threat to the scientists involved.

In general, visualization can also be employed to check simulations for plausibility via arbitrary navigation in time and space. As a practical example, a domain scientist used to visually inspecting simulations is able to tell whether a simulation has been performed at some temperature that is completely wrong just from the movement and interaction of particles over time. Visualization can also serve for dissemination of results or education of students by explicitly showing the effect of underlying laws and principles.

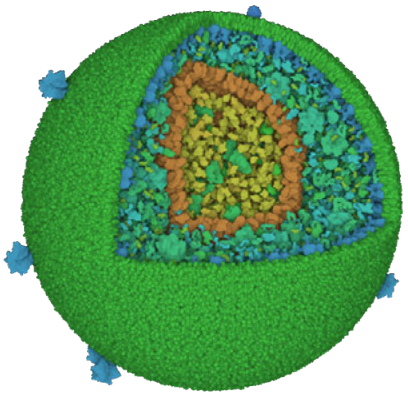


Figure 1.2: Level-of-detail rendering of a HI virus as generated by cellVIEW. Courtesy of Mathieu Le Muzic.

The data set sizes in practical use vary significantly depending on application area and use case. The number of attributes per particle is usually low enough to guarantee that a data set stays roughly in the same order of magnitude, however data can become arbitrarily large if long-term effects or rare events need to be observed. Thousands of time steps are commonplace despite the simulations already emitting only every $Nth (N > 1)$ time step, so data scales linearly with the number of time steps (e.g., by at least three orders of magnitude for a thousand time steps). To give an idea about the common size of a single step, one can start with simple proteins with thousands of atoms and thus only megabytes in size. Data is considerably larger in the physical sciences, where millions of particles are simulated and a time step can already require gigabyte(s) of memory. The upper end is represented by life sciences, where a HI virus including blood plasma was represented using 15 billion atoms [Le Muzic et al., 2015] which would require in the order of 200 GB just for storing an uncompressed time step, and cosmological simulations, like the DarkSky data set, where a single time step consists of one trillion particles and requires more than 30 TB [Skillman et al., 2014].

The described scope can be employed to derive a few simple requirements for particle-based visualizations. These lead to the concrete approaches that will be presented in the following chapters. First off, the resulting visualization must be intelligible, at best without any prior training. This means the metaphors used should be closely related to known approaches from textbooks or some other status quo the user can relate to. With few to no additions to the textbook prototypes, the result can be directly used for a broader public, meaning students of the respective topic or even for general dissemination. Interactivity is also key, since arbitrary exploration is oftentimes a central feature that distinguishes state-of-the-art visualization from the status quo found in commonly used visualization tools. Last, scalability to complex systems should be given. This can be achieved by technical measures plus abstractions that allow for better performance and easier, possibly hierarchical, exploration of extremely large data.

1.1.1 BASE-LINE RENDERING

Independent from the actual research question, the availability of a direct visualization is important throughout the analysis process and the evolution of problem-specific visualization approaches. This base-line rendering is essential for the user to trust in more complex visualizations, since it can always serve for comparison purposes and verification of more abstract visualization results. Exploratory analysis fosters unbiased, novel findings, as every abstraction cements assumptions by way of the transformed visualization. Its fidelity is also required for the scientist to check simulation results for plausibility even at a very small scale (visual debugging): a common result of a researcher being confronted with a visualization of his/her simulations for the very first time is the realization that something is unexpected or wrong. The first case is a good starting point for a scientific publication, while the second one at least helps improving the simulation model or implementation (in turn potentially leading back to case one).

The base-line rendering should be a neutral starting point when communicating with any application scientist. It should correspond to the unabridged raw data set, usually presented via a common-ground metaphor. For molecular dynamics the calotte (also called space-filling or CPK) model is used, resulting in hard spheres for every available atom. Here, the radius of the sphere equals the atomic radius of the corresponding chemical element. Granular media can have different particle representations, for example as instances of a pre-defined set of available geometries of varying complexity [Grottel et al., 2010b]. In this context the lecture will cover different visual metaphors and the according GPU-based rendering methods to achieve interactivity even for large numbers of particles (see Chapter 3). Additionally, concise data representations will be discussed in Chapter 5, easing storage but more importantly improving performance when transferring data to the GPU.

The direct representation of particles presents some issues as well. For example, this strategy cannot scale indefinitely: if we consider an arbitrary output device with a physical resolution of $w \times h$ pixels, it stands to reason that no more than $w * h$ particles in the projected image plane can be discerned. The resulting image will of course mostly present noise since all particles are just single colored pixels in this limit case, so the real limit is considerably lower. The actual data set will conversely be bigger than the product $w * h$ if the domain is $\Re^3$. Part of the complexity in depth is lost due to projection and occlusion. The analogy is still fitting as resolution is an effective cap for data set size if a faithful overview is to be given via base-line visualization. Interactive camera adjustment can of course partially remedy this issue, since it allows users to zoom into regions of interest and get a detailed view of interesting regions in the data set. The problem to identify these interesting regions in the data set, however, is still present since the base-line rendering does not always easily show structures formed by the particles in very large data, especially if they are on different scales.

1.1.2 HIGHER-LEVEL STRUCTURE

Practical experience shows us that we can render about 10^8 particles interactively without taking special measures, but even with high-resolution output the result is difficult to understand in its higher-level structure as this is hidden by the high-frequency details when just using local lighting and a simple shading model. One solution to this problem is the introduction of additional visual cues via global lighting effects or transparency, as detailed in Chapter 7. On the other hand, even though the data does not contain a topology, there is often a higher-level functional structure available. The solution is usually application-specific so the structures have hugely differing sizes and complexity, starting from droplets in thermodynamics, over amino acids and proteins in biochemistry to whole cells in systems biology. These structures can, for example, be summarized by their interface to the environment. Thus, metaphors result in application-specific visualizations like ellipsoids for droplets or molecular surfaces in different configurations and their respective semantics (see Chapter 8).

1.2 RELATED TOPICS BEYOND THE SCOPE

This lecture explicitly does not go into detail about two variants of particle-related visualizations. First, particle systems as they are employed for special effects, for example representing sparks, fire, or smoke. The visual metaphor here abstracts from the single particle to generate larger-scale visual effects geared toward a believable visual imitation of reality that does not necessarily require physico-chemical correctness, but to conserve the artistic appearance [Feldman et al., 2003, Reeves, 1983]. Second, particles representing a sampling of continuous functions. This category can itself be divided in two specific areas: point set surfaces and smoothed particle hydrodynamics (SPH).

The first subset, point set surfaces, describe surfaces which require explicit or implicit reconstruction from a limited number of discrete samples of an object, e.g., obtained by laser scanning. This has its own caveats as any holes can either be an artifact stemming from limited resolution or be intentional. The surfaces of such point sets are rendered with point-based methods like splatting or surface elements (surfels) [Pfister et al., 2000]. The GPU lends itself to efficiently render such surfaces in high quality [Botsch and Kobbelt, 2003]. The most recent point cloud rendering approaches benefit from the latest GPU features and utilize vertex buffer objects and bindless textures for improved performance [Goswami et al., 2013]. For more details, the reader is referred to the comprehensive overview over the whole process from acquisition over data handling to rendering as edited by Gross and Pfister [2007]. The idea of point-based surfaces has also been applied to the field of visualization, e.g., the rendering of stream surfaces in 3D flows [Schafhitzel et al., 2007].

Smoothed particle hydrodynamics (SPH) on the other hand allow the reconstruction of a density-based data set from discrete samples by way of a so-called *smoothing kernel*. This method is often employed for fluid simulations, and respective surface rendering approaches have been

presented. Examples include the work by Zhang et al. [2008] and Reichl et al. [2014], utilizing the GPU for both simulation and rendering.

Another prominent application for SPH is cosmology, employing this method for the simulation of dark matter distribution following the big bang. Several large-scale visualization techniques have been presented in the last decade, for example by Hopf and Ertl [2003] or Fraedrich et al. [2009].

Despite the differences, this whole body of work still represents a good source of information, especially regarding some of the technical aspects. Data handling and quantization are examined in detail and will still help today if the data can be condensed sufficiently to make a whole data set fit on the GPU, taking data transfer problems out of the equation entirely. Rendering optimizations can also be relevant, however, since much of the prominent work is about a decade old, recent additions to GPU features and API capabilities are not covered, and much has changed especially regarding memory management (bindless access, persistent mapping, raw shader storage, etc.). For more details on optimizations and GPU features, we refer the interested reader to the Radeon Rays SDK using OpenCL[1] and the OptiX Ray Tracing Engine relying on CUDA.[2]

[1]Radeon Rays SDK | AMD, Inc.: http://gpuopen.com/gaming-product/radeon-rays/ (last accessed 08/19/2016).
[2]OptiX™| NVIDIA Corp.: https://developer.nvidia.com/optix (last accessed 08/19/2016).

CHAPTER 2

History

The scenarios and requirements sketched out in the introduction have been a widespread application case in visualization for at least 30 years now [Max, 1983]. As such, the widely accepted solutions presented in this lecture shall be briefly put into historical context. Very well-known and early examples for particle visualization were all applied to molecular dynamics, for example VMD [Humphrey et al., 1996], PyMOL [Schrödinger, LLC, 2015], Chimera [Pettersen et al., 2004], and AtomEye [Li, 2003]. These tools offer comprehensive functionality that goes far beyond visualization and is specifically geared toward the application. However, they also are "legacy" in the sense of having been designed before the widespread availability of programmable GPU. Having grown over time, they support hardware of diverse capabilities, but are not organized from the ground up for optimal performance when run on a capable GPU. This concerns memory management as well as the traditional polygon-based rendering. VMD, for example, also supports some of the techniques that will be described further on, but as the focus is on generality, this is just an option and not the base strategy. Looking at polygon-based rendering especially in the context of molecular dynamics brings us to the observation that the basic element, a sphere, is rather costly in tessellated form. Memory is not a concern if instancing is employed. However, a spherical silhouette is only obtained using a considerable number of triangles—tens to hundreds, depending on the screen-space footprint (see Figure 2.1). The resulting load on the geometry stages of modern GPUs is thus quite high. This can be improved upon using multiple LoDs or dynamic tessellation from some basic representation like a tetrahedron, but the effort is better spent otherwise, as we will show in Chapter 3. Shading does not present a problem, as it can be performed per fragment and, thus, has optimal quality.

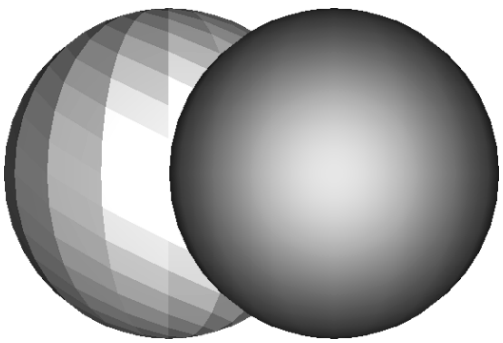


Figure 2.1: Tessellated sphere geometry with per-triangle shading and per-fragment shading.

Subsequently, a better-scaling approach for large numbers of atoms was presented by Bajaj et al. [2004]. TexMol employs an imposter in form of a depth and normal texture so each atom needs only a bounding geometry—a quad—that is correctly scaled and placed as a billboard. Using the depth replace feature in the fragment shader, the initially flat billboard is modified to even allow for depth-correct intersections of spheres.

Local shading can be performed based on the normal texture. The result is much faster than the geometry-based approach and offers nearly comparable quality: intersections are more precise than with discretized spheres, however shading presents slight banding artifacts from the limited precision of the employed normal texture. The billboard-based approach also does not allow for perspective-correct rendering and as such reduces the available depth cues.

This kind of strategy was an excellent fit at the time, but this is not the case anymore, as it causes high memory pressure due to the large number of texel fetches and has negligible arithmetic intensity. Modern GPUs, however, are designed to actually require lots of computations to hide the relatively high latency of their memory interface. Considering a Geforce FX 5900 Ultra (from 2004) and a Geforce 980 Ti (from 2015), the respective memory bandwidth has increased from about 27 GB/s to about 337 GB/s (12.4-fold). The respective compute power, however, has increased from about 15 GFlops/s to more than 6,000 GFlops/s (400-fold). Consequently, modern rendering techniques should expend many arithmetic operations if a texel fetch can thus be avoided.

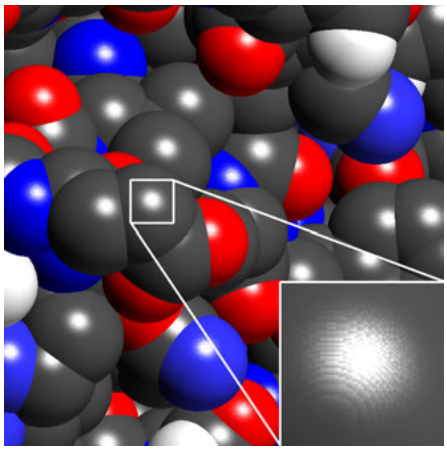


Figure 2.2: A protein data set rendered using TexMol [Bajaj et al., 2004]. The inset shows banding artifacts due to the limited precision of the normal texture.

These changes in GPU technology led to the current state-of-the-art. A desirable technique would compute a correct surface together with a normal and the resulting shading from as terse a representation as possible for each visible fragment. One classical approach is ray casting, i.e., ray tracing using only primary rays. Implementing a purely image-order ray caster requires a spatial data structure to reduce the number of hit tests against the whole scene. The implementation is a bit more intricate, but performs very well for very large data (see Chapter 5). A hybrid image/object-order approach can take advantage of the graphics pipeline to render billboards which tightly fit the desired geometry, similar to the approach in TexMol. However, the geometry itself is only represented implicitly, so very few parameters suffice to describe basic solids (position, orientation, dimensions). These parameters can be directly attached to the proxy geometry and can thus be stream-processed together in the graphics pipeline. As a result, the geometry load for the GPU is low, same as the memory pressure. In return, high arithmetic intensity is generated by the ray-object intersection tests, depending on the complexity of the represented geometry. Implicit functions up the order of four (i.e., quartics) can easily be solved explicitly, while higher-order surfaces require iterative approaches. Performance behaves also very beneficially: the more particles there are, the smaller their screen-space footprint becomes. The resulting geometry load is not increased significantly because of the simplicity of the billboards, while the fragment shader load is balanced by the rel-

ative size. The result fits modern GPU architecture quite well, so several approaches based on this concept have been presented over the years. A few examples will be shown before proceeding to the detailed explanation of the fundamental hybrid approach in the next chapter.

The starting point was marked by Gumhold [2003]. He presented depth-correct implicit ellipsoids which are ray cast in the fragment shader. The concept was then refined by Klein and Ertl [2004]. Here, the ray is first transformed into the respective parameter space of each glyph, thereby simplifying the problem to intersecting a unit sphere. In the same year, Toledo and Lévy [2004] presented general quadrics to be rendered on the GPU. Then, application-specific glyphs were developed, for example a dipole representation Reina and Ertl [2005] as well as very costly iterative hyperstreamtubes for DTI rendering Reina et al. [2006]. Sigg et al. [2006b] wanted to reduce computational cost and thus improved the bounding geometry to fit implicit quadrics more tightly. More generalized solutions were also devised: Loop and Blinn [2006] showed an approach to generate algebraic surfaces by way of Bézier tetrahedra, while Knoll et al. [2009] even allowed arbitrary implicits by employing interval arithmetic (Figure 2.3).

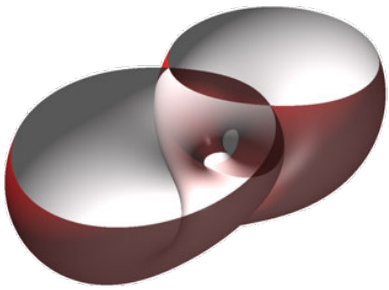


Figure 2.3: Arbitrary implicit shape using interval arithmetic. From [Knoll et al., 2009].