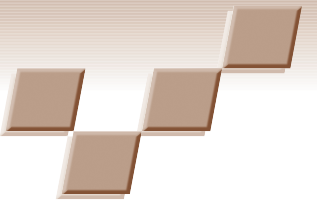


Real-Time Interaction with Complex Models

David J. Kasik
The Boeing Company

Dinesh Manocha
University of North Carolina at Chapel Hill

Philipp Slusallek
Saarland University, Germany

Over the past few decades, advances in acquisition, modeling, and simulation technologies have resulted in databases of 3D models that seem to be unlimited in size and complexity. Highly detailed models are increasingly produced in areas such as industrial CAD for airplanes, ships, production plants, and buildings; geographic information systems; oil and gas exploration; medical imaging; scanned 3D models; unorganized information spaces; and high-end scientific simulations. Such models might contain millions or even billions of 3D primitives, such as triangles, point sets, surfaces, and voxels. The models often have additional, higher-dimensional attributes associated with the primitives.

Challenges

The 3D data sets' enormous size poses several challenges for interactive applications such as walkthroughs, simulation-based design, and data visualization. These applications demand interaction and rendering rates of at least 10 Hz. The 3D data sets' size has been growing at a rate faster than Moore's law. At the same time, disk I/O and memory bandwidth has been growing at a rate slower than Moore's law. As a result, handling multi-gigabyte or terabyte data sets tends to overload the performance and memory capacity of current computer systems.

These trade-offs make real-time interaction with complex models a computing system problem. Solutions to the problem cannot be isolated to an application, a rendering approach, a modeling methodology, a network speed, or a delivery system design improvement.

Historical context

The problem of rendering complex models at interactive rates has been studied in computer graphics and related areas for more than three decades. There has been an across-the-board attack in terms of developing new representations and data structures, algorithms, and integrated systems. One of the first papers by Jim Clark in 1975 proposed the idea of hierarchical representations and use of multiple levels-of-detail of objects or scenes to accelerate the rendering. There were many significant research and development efforts in the '80s and '90s in terms of accelerating the rendering of large models. One of the first major efforts in this area was started by Fred Brooks in the mid-'80s at the University of North Carolina at Chapel Hill (UNC). The Walkthrough Group at UNC developed several pioneering concepts, such as visibility culling using cells and portals, progressive refinement for rendering acceleration, and use of a steerable treadmill, as a way of "experiencing" architectural buildings. Carlo Séquin started a major effort in 1990 at UC Berkeley to accelerate the rendering of large architectural models by developing faster visibility algorithms and memory management techniques. One of the first industrial systems to render large CAD models such as airplanes was the Boeing FlyThru system. Developed in the early '90s, FlyThru uses simple boxes to improve performance. While highly successful for the Boeing 777, FlyThru could only display as much data as could fit in a workstation's memory, about 1/50th of a complete Boeing 777. In the mid-'90s researchers at UNC developed a new set of visibility and simplification algorithms and systems to render very complex CAD models of submarines, power plants, and oil tankers at interactive rates.

Significant advances in 3D scanning in the '90s led to the availability of complex 3D models (such as the Stanford Bunny and digital archives of Michelangelo's famous sculptures) that contained tens of gigabytes of data. Dense point clouds were used to generate triangle mesh-based representation of these models. These models led to the development of polygon decimation and simplification algorithms along image-based sampled representations.

Advancements in high-performance and scientific computations generated terabyte data sets containing simulation data. These are high-dimensional data sets (that is, isosurfaces) with hundreds of millions or billions of triangles. One of the major benchmarks in the field is the 470 million triangle isosurface generated from a high-resolution 3D simulation of Richtmyer-Meshkov instability and turbulence mixing generated at Lawrence Livermore Labs. This led to many novel developments in the area of visualizing gigabyte and terabyte data sets.

Much of the early work in interactive rendering was focused on utilizing the rasterization capabilities of GPUs. Over the last seven to eight years, ray tracing has emerged as an alternate method. Though ray tracing has been extensively studied in computer graphics for more

 0272-1716/07/$25.00

than three decades, the exponential growth of processing power and the embarrassingly parallel nature of ray tracing algorithms have renewed interest in real-time rendering. The seminal work on ray coherence techniques at Saarland University helped develop almost real-time ray tracing systems for complex 3D models.

Current work

The field of real-time interaction with complex models is becoming increasingly active with novel and broad applications. Besides new developments in terms of rasterization and ray tracing, recent research efforts have addressed system integration and optimization techniques to extract higher performance from the underlying computer system. These include new software and hardware architectures, exploiting the computational capabilities of new multicore and many-core processors, and rendering these data sets on high-performance parallel processors for display on thin clients or high resolution displays.

We see increasing deployment of these algorithms and systems in many high-impact and high-volume commodity applications. These include computer games that use these algorithms and representations for generating very high frame rates (such as 60 Hz) on current PCs or game consoles with high-fidelity imagery. At the same time, client applications such as Google Earth and Microsoft Live Earth use these methods to interactively fetch 3D terrain primitives from a remote server with terabytes of data and render them on the local client. More than 200 million copies of the client real-time renderers for these applications have been downloaded, and usage in other domains is increasing.

The articles in this special issue

We've assembled an excellent set of tutorials and articles that introduce and address many of the system issues users and developers encounter when dealing with highly complex 3D models. Each article contains different system implementation approaches. Each addresses many, but not all, of the issues that allow real-time interaction with complex models. The application areas are similarly broad. This introductory article asks that you build a mental model of the sorts of issues that must be addressed and the types of approaches that people have developed. The choice of the "correct" approach depends greatly on the specific domain that you intend to address.

Tutorials

We start and end the issue with tutorials written from totally different points of view. In "Massive-Model Rendering Techniques," Dietrich, Gobbetti, and Yoon provide an overview of the state of the art in the field. Their intent is to give an overall context for real-time interaction with complex models. This tutorial gives you clear insight to the general issues that researchers and commercial vendors must consider to build a complete solution. After reading the tutorial, the reader should realize that developing a broad-ranging solution that covers several usage scenarios and several sources for 3D model data is a nontrivial task. Each individual component could by itself be the subject of a special issue. Because we address interactive applications with a wide variety of 3D models, understanding how to sustain real-time performance often outweighs all the other issues. Building a useful and usable interactive visualization system that can move into production takes a careful, balanced approach.

In "Visualization of Complex Automotive Data," Stevens examines the visualization challenge from a user perspective. He provides the context often missed once development starts in earnest. He also shows how General Motors addresses visualization for both its design and engineering analysis data. Pay careful attention to the breadth of system issues a user finds important in this context. The issues go beyond what the implementers have to consider in the Dietrich tutorial. Central to the issue of complex models is the notion that users want to look at everything at once. To do that, users must take responsibility for understanding where the visualization data is located and how its configuration is controlled. Eventually, all that data must be assembled and put in a location that users exercising the interactive visualization tool can access.

Technical articles

Reduced to its core, computer graphics renders 3D models to determine what color or intensity value should be assigned to each pixel on a user's screen. Adding "interactive" is what makes this issue so special. We included two articles that describe the two extant competing, yet ultimately complimentary, forms of rendering in common use today.

In "Exploring a Boeing 777: Ray Tracing Large-Scale CAD Data," Dietrich, Stephens, and Wald examine the system issues necessary to take traditional, CPU-intensive ray tracing into a real-time solution on a complex CAD data set. The authors describe the manner in which real-time ray tracing was molded into a system that could render the digital model of a Boeing 777 at 10 Hz and faster. The solutions from Saarland University (inTrace) and the University of Utah (Manta) differ in implementation details, but have a remarkable amount of similarity, though they were developed independently from each another. The authors address the system issues of dealing with multiprocessor- and multicore-computing architectures and how to deal with memory size and disk I/O issues well.

The contrasting rendering approach is rasterization (z-buffer), which is often implemented in GPUs. In "Interviews3D: A Platform for Interactive Handling of Massive Data Sets," Bruderlin describes a successful implementation of visibility-guided rendering that lets a GPU perform at interactive rates without extensive polygon decimation, level-of-detail precomputation, or other tricks. His approach is based on the observation that a user isn't actually viewing a billion polygons on a frame-by-frame basis. He describes techniques that determine which subset of the total 3D model is visible on a frame-by-frame basis and how to use the GPU, access disk, and CPU memory effectively enough to let the GPU take over the rendering task. Pay attention to the similarity of many of the system-level approaches to the manner in which Dietrich, Stephens, and Wald address the problem using ray tracing as the rendering approach.

The first two articles describe solutions that have been

primarily applied in the CAD environment. In CAD, complex models are defined geometrically and polygons are generated by tessellation routines to some level of granularity. Generally speaking, CAD models are used in physical products such as automobiles, airplanes, tankers, and power plants. Suppose the data comes from the opposite direction and is the result of scanning a physical model at a fine level of detail. Various forms of scanning technology are readily available and generate massive amounts of data. In "Visualizing and Analyzing the *Mona Lisa*," Borgeat and his colleagues describe an elegant approach to interactively viewing the large amounts of scanned data resulting from a finely detailed scan of da Vinci's *Mona Lisa*. A supplemental video gives a visual testament of the success of their approach.

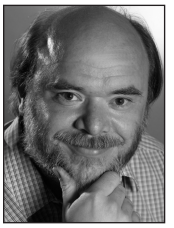
The final article deals with the system considerations of another form of a complex model: terrain with 3D buildings. Typical examples of this type of model are found in Google Earth and Microsoft Live Earth. In "Network-Based Visualization of 3D Landscapes and City Models," Royan and his colleagues offer an approach they developed in France. Having to deal with more 3D data superimposed on their terrain than Google Earth led the authors to develop new approaches. The article introduces a method for dynamic network delivery of the complex models to user workstations during real-time sessions. The other three articles assume that data is stored on a local disk, from which data is transferred much faster than data from servers accessed over conventional networks.

What does this all mean? The Applications article in this issue is a practical example from CSIRO (Australia) that lets users compares two different auto crash scenarios. Looking at lots of data in real time is a huge assist to the engineering staff doing the analysis.

This special issue on real-time interaction with complex models should give you an excellent introduction to the system issues and approaches. The issue is truly an international one, as it includes authors from the US, Europe, and Australia. Other research groups are active in Asia. It's been a pleasure for us to work with each other, the reviewers, and the authors. Many thanks to the excellent IEEE Computer Society staff who made our concept into readable reality. ■

David J. Kasik *is an information architect for visualization and user-interface technology at The Boeing Company. His research interests include combining basic research with practical implementation and increasing awareness of the impact of computing and technology innovation outside the computing community, especially for K-12 students. Kasik has an MS in computer science from the University of Colorado and a BA in quantitative studies from the Johns Hopkins University. He is a member of the* IEEE Computer Graphics and Applications *editorial board. He is a member of the IEEE, ACM, ACM Siggraph, and ACM Sigchi. Contact him at david.j.kasik@boeing.com.*

Dinesh Manocha *is the Mason Distinguished Professor of Computer Science at the University of North Carolina at Chapel Hill. His research interests include computer graphics, geometric and solid modeling, robotics, symbolic and numeric computation, virtual environments, and computational geometry. Manocha has a PhD in computer science from the University of California, Berkeley. He has received the Junior Faculty Award, Alfred P. Sloan Fellowship and NSF Career Award, Office of Naval Research Young Investigator Award, Honda Research Initiation Award, and Hettleman Prize for Scholarly Achievements. Contact him at dm@cs.unc.edu.*



Philipp Slusallek *is a professor in the computer graphics department at Saarland University and founding speaker of the Competence Center for Computer Science at the university. His research interests include real-time ray tracing, high-performance graphics hardware for ray tracing, real-time lighting simulation, massive model visualization, and network-integrated multimedia applications. Slusallek cofounded the spin-off companies inTrace and Motama and serves as scientific adviser to them. Slusallek has a PhD in computer science from Erlangen University, Germany. He is a member of the IEEE CS, ACM Siggraph, and Eurographics. Contact him at slusallek@cs.uni-sb.de.*