Proposal for an EG2006 Tutorial (full day)

Real-time, Interactive Massive Model Visualization

David Kasik, Boeing
Philipp Slusallek, Saarland University

Summary

Real-time interaction with complex models has always challenged interactive computer graphics. Such models can easily contain gigabytes of data. This tutorial covers state-of-the-art techniques that remove current memory and performance constraints. This allows a fundamental change in visualization systems: users can interact with huge models in real time.

Keywords

Massive Model Visualization, Realtime rendering, Ray-Tracing, GPU-Based Rendering

Abstract

The amount of data produced by today's engineering design and scientific analysis applications often exceeds the capability of conventional interactive computer graphics. Users produce tens of gigabytes of data while designing a product or analyzing results. Techniques for examining all this data simultaneously and interactively are not readily available in today's visualization or CAD tools.

Combining specific algorithms, specialized data structures, and high performance hardware has enabled real-time visualization and is a significant research area. As a result, users can see an entire airplane instead of a subsection or full level-of-detail of a building instead of a simplified form.

This tutorial presents seven different solutions to the problem. Each instructor will focus on the practical aspects of their implementation and provide examples, either as movies or live demos. The tutorial will provide participants with the knowledge to identify trade-offs and weigh benefits. In addition, we discuss system implementation issues, the conceptual basis for the work, the impact on the user community, how to accelerate user acceptance of the technology, and methods to increase the amount of test data for the research community.

Key technical topics include: software techniques to overcome performance and memory size limitations (e.g., kd-trees, occlusion culling, multi-threaded programming, parallel processor transaction management, memory-mapped files, display lists, cache coherent layouts); computing architecture (e.g., parallel processor architectures, single and multi-GPU hardware, thin client access to rendering services, hardware occlusion culling, cell computers, multi-core CPUs); and overall system architecture (e.g., preprocessing, large

user communities, model configuration management, network transfer of basic geometry).

The instructors come from academia, start-up companies, and industry. Each has built an approach that combines one or more of the above technologies. The tutorial will be organized around the instructor's technical approach, what parts have worked, and lessons learned when applying these technologies to real-world problems.

Instructors

Dave Kasik

Technical Fellow
The Boeing Company
david.j.kasik@boeing.com

Dinesh Manocha

Professor
University of North Carolina
dm@cs.unc.edu

Abe Stephens

PhD Student
Scientific Computing and Imaging Institute
University of Utah
abe@sci.utah.edu

Beat Bruderlin

Professor of Computer Science
Technical University of Ilmenau (Germany)
bdb@imp.tu-ilmenau.de

Philipp Slusallek

Professor
Saarland University (Germany)
slusallek@cs.uni-sb.de

Enrico Gobbett

Director of Visual Computing
CRS4 (Italy)
gobbetti@crs4.it

Wagner Correa

Research Scientist
IBM
wtcorrea@us.ibm.com

Inigo Quilez

Visual Technologist
VRContext (Belgium)
i.quilez@vrcontext.com

Prerequisites

- Familiar with fundamental computer graphics concepts and terminology (rasterization, ray tracing, visibility culling, etc.).

- Basic knowledge of computer programming (graphics APIs, threads, etc.).

Audience

This tutorial is intended for producers of complex models, visualization users in industry and research, and practitioners who design interactive, real-time rendering software. Each attendee will be able to contrast the different strategies needed to provide real-time interaction in a resource-limited computing environment.

Tutorial History

The tutorial (in this general form) has not been taught at SIGGRAPH. There have been affine tutorials dedicated to specific aspects, including:

- SIGGRAPH2005: Philipp Slusallek et al. taucht a course on realtime ray tracing.

- Dinesh Manocha was involved in courses related to Interactive Walkthroughs of large datasets in the 90s and organized two such courses in 1999 and 2000. Some aspects of handling large datasets were also covered in SIGGRAPH courses on GPU-based computations in 2002 and 2003. All these courses were well attended.

This tutorial is the first to teach the details of both ray tracing and GPU-based approaches to handle massive models. Importantly, we introduce many of the practical aspects of moving research to full production implementation, a difficult problem when dealing with huge 3D datasets and a large number of users.

Tutorial Notes

Attached to this proposal is a set of commented tutorial slides. We will provide those in electronic form to the attendees.

Syllabus

Tutorial goal: Each attendee should understand the key system software strategies and computing system components needed to analyze and evaluate different solutions, implement a solution, and understand where the next set of research challenges lie. In addition, the attendee will gain an appreciation for some of the business aspects of implementing massive model applications in a production environment.

The tutorial starts with an introduction to the motivation and challenges for massive model visualization. We then interleave Z-buffer and real-time ray tracing approaches and cover production implementation issues.

Here is a brief tutorial overview before we discuss the individual sections in detail.

- 8:30 – 9:15 Motivation and Challenges, ***Dave Kasik**, Boeing*
- 9:15 – 10:05 Interactive View-Dependent Rendering and Shadow Generation in Complex Datasets, ***Dinesh Manocha**, UNC*

- 10:20 – 11:10 Ray Tracing with Multi-Core/Shared Memory Systems
  ***Abe Stephens**, University of Utah*
- 11:10 – 12:00 Visibility-guided Rendering to Accelerate 3D Graphics Hardware Performance
  Beat Bruderlin, TU Ilmenau and 3DInteractive GmbH

- 1:00 – 1:50 Massive Model Visualization using Realtime Ray Tracing
  Philipp Slusallek, Saarland University and inTrace GmbH
- 1:50 – 2:40 GPU-friendly accelerated mesh-based and mesh-less techniques for output-sensitive rendering of huge complex 3D models
  ***Enrico Gobbetti**, Director of Visual Computing, CRS4*

- 2:55 – 3:45 Interactive Out-Of-Core Visualization of Large Datasets on Commodity PCs
  ***Wagner Correa**, IBM*
- 3:45 – 4:35 Putting Theory into Practice
  ***Inigo Quilez**, VRContext*
- 4:35 – 5:00 Panel discussion
  All, Moderator: David Kasik

8:30 – 9:15 Motivation and Challenges (Dave Kasik, Boeing)

Goal: Provide clear understanding of the implications of complex, massive model visualization and how the user community can effectively assist the research community.

The human visual system provides an extremely efficient way to communicate both context and detail. The amount of data that's being generated is actually exceeding the rate of change of Moore's law.

The domains in which data is expanding range from computer-aided design and manufacturing to arts and entertainment to intelligence analysis. The most effective way people have to comprehend and communicate the overall implications relies on computer graphics. Clear examples from specific domains show the effectiveness of interactive, real-time 3D graphics.

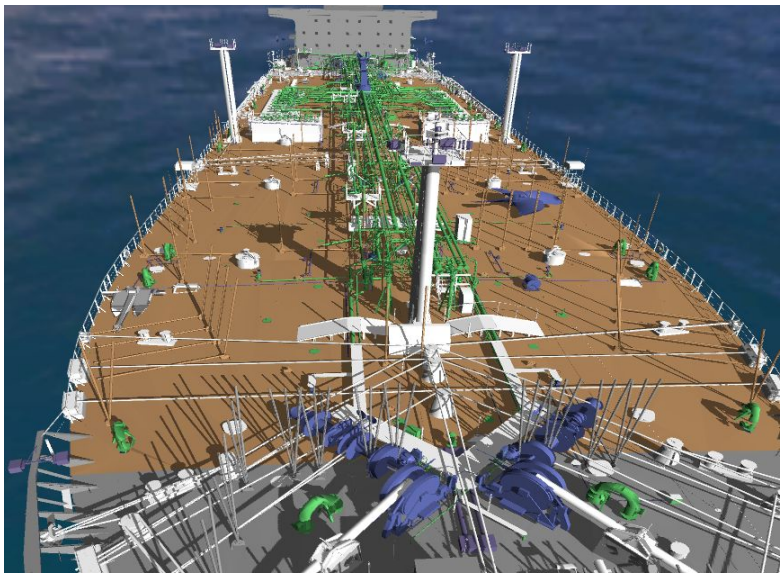
The overall system implications of real-time interaction are essential to ultimately make the technology implementable. The attendees will get a clear understanding of all aspects that will make or break the widespread adoption of the techniques described in the other sections of the tutorial.

Section outline:

1. Motivation for effort from a user's perspective, including sample use cases
2. Characterization of user tasks that can be addressed by visual analysis
3. General architecture to integrate with other visualization components
4. Contrast of issues between GPU and CPU-based approaches
5. Pragmatics of getting data released to the research community
6. Additional challenges:
    a. Network impact.
    b. Collision detection.
    c. Pre-processing burden.
    d. Visual model update.

9:15 – 10:05 Interactive View-Dependent Rendering and Shadow Generation in Complex Datasets (Dinesh Manocha, University of North Carolina),

Goal: This section will expose recent methods for interactive visualization of complex datasets based on view-dependent rendering and shadow generation. Attendees will also learn some issues in getting high performance throughput from current GPUs for such complex datasets.



Data courtesy Newport News Shipbuilding

Current GPUs are progressing at a rate faster than Moore's Law. In theory, they are capable of achieving peak throughput of tens of millions of triangles per second. However, they are optimized for game-like environments and it is a major challenge to render complex models composed of tens or hundreds of millions of triangles at interactive rates. We outline a number of algorithms to overcome these problems.

We outline techniques to build good scene graph representations of complex datasets using partitioning and clustering algorithms. Furthermore, we present an optimization-based algorithm to compute cache coherent layouts for improved CPU and GPU throughputs. In order to achieve high frame rates, we only render triangles that the user can "ultimately see". We present efficient and practical techniques for view-dependent simplification and occlusion culling on large models. Furthermore, we describe novel hierarchical data structures to integrate these algorithms. Finally, we present novel algorithms for shadow generation. Specifically, we present subdivided shadow maps, which can overcome perspective aliasing problems and work well on current GPUs.

Finally, we demonstrate the application of our algorithms to different types of complex models on a commodity desktop or laptop. The set of models include large scanned datasets, isosurfaces extracted from simulation data, CAD environments of powerplants, airplanes and tankers, and terrain datasets. We also outline many open problems in this area.

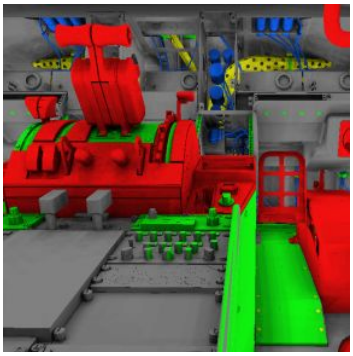
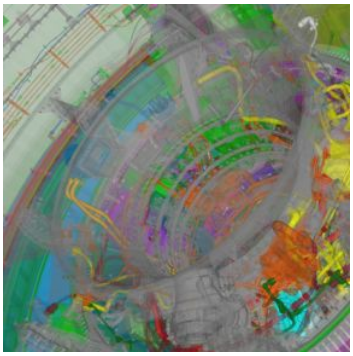
Section outline:

1. Real-Time Occlusion Culling
2. View-Dependent Rendering
3. Out-of-Core Computations
4. Cache-Coherent layouts, in order to get good runtime throughput on massive models

5. Interactive shadow generation in complex models
6. Integrating different rendering acceleration techniques to handle different kind of models: scanned datasets, textured datasets, CAD datasets, and scientific simulation.
7. Real-time demo

10:20 – 11:10 Ray Tracing with Multi-core/Shared Memory Systems (Abe Stephens, University of Utah)

Goal: Understand the implications of shared memory systems.

Data courtesy The Boeing Company

Shared memory computer systems provide a strong platform for interactive software rendering. They combine a large number of processors, enormous amounts of memory and many shared devices across a single system.

Special hardware in these systems provides many facilities that must be implemented by hand on a cluster. Still multi-processor servers, or even multi-core workstations, behave differently from standard single processor desktops and the graphics programmer must pay attention to the system architecture in order to obtain optimal performance. The challenges encountered implementing an interactive renderer on such a system are not well addressed by existing HPC tools and programming techniques and require the programmer to interact with the system at a much lower level.

Moving beyond the rendering algorithm content discussed in the SIGGRAPH 2005 ray tracing course we will provide detailed examples of how parallel system architecture effects renderer implementation. We will examine both dual core AMD Opteron systems and a Itanium2 processor SGI systems. Both types of systems are sold as an interactive ray tracing platforms for engineering design situations.

Our presentation and tutorial notes will attempt to address three basic questions: How are the user's expectations different when rendering on a moderately sized parallel system? (collaborative, remote visualization).

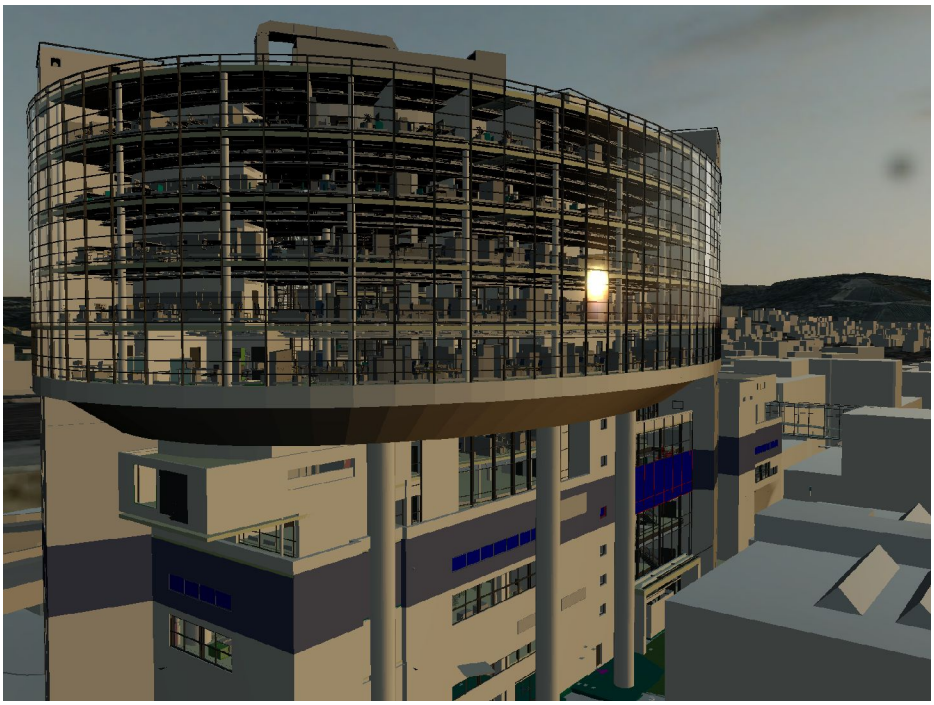
How are parallel systems constructed and how well suited are their memory systems for interactive rendering? (parallel system behavior) How does the software design effect the renderer's ability to scale on larger systems or larger problems? (Parallel software architecture)

Section outline:

1. Strengths of large multi-processor systems.
    a. Collaborative visualization.
    b. Need for remote rendering.
    c. Solutions including: X, VNC, Vizserver.
2. Parallel system behavior
    a. Summary of Altix and Opteron memory systems
    b. Remote and local memory access patterns
    c. Memory latency and its effect on performance
3. Parallel software architecture
    a. Implementation ease on shared memory systems.
    b. Transactions with Barriers (Manta)
    c. Multi-buffered state without Barriers
4. Architecture specific optimizations & Parallel ray tracing examples

11:10 – 12:00 Visibility-guided Rendering to Accelerate 3D Graphics Hardware Performance (Beat Bruderlin, Technical University of Ilmenau and 3DInteractive GmbH)

Goal: Understanding the foundations of Visibility-Guided Rendering (VGR) of complex scenes, using hardware-accelerated OpenGL. The key issues covered in this lecture are hardware-based occlusion culling, memory management and out-of-core visibility determination, as well as a host of low-level operating system issues.



Hardware accelerators for 3D graphics (GPUs) have become ubiquitous in PCs and laptops. They are very powerful for real-time visualization of scenes up to a few million polygons at very high (almost photo-realistic) quality. GPUs have been successfully applied in computer games and engineering visualization. However, a straightforward use of GPUs, as is the current practice, can no longer deal with the ever larger datasets of fully detailed engineering models such as airplanes, industrial plants, cars, etc. This puts a severe limitation on the data explosion we currently encounter in industry. Already some large models are 100 to 1000 times larger than what can be handled by GPUs in real time.

Visibility-guided Rendering is a novel approach for real-time rendering of very large 3d datasets.

We start by identifying the main differences between sampling-based rendering (e.g. ray tracing) and rasterization-based rendering (e.g. hardware-accelerated OpenGL). By comparing the pros and cons of the two opposite approaches at a high abstraction level, we can explain some of the current limitations of OpenGL and develop ideas for overcoming these limitations.

The key advantage of VGR is to efficiently determine visibility of the vast majority of polygons before they are sent to the graphics hardware. Different culling techniques are presented which can be used in combination. Spatial data structures, as well as hardware features of the GPU can be exploited to implement the culling techniques optimally. The visibility-guided rendering approach relies heavily on efficient memory management, concurrency between CPU and GPU, as well as optimal use of low-level OS functionality to handle very large models.

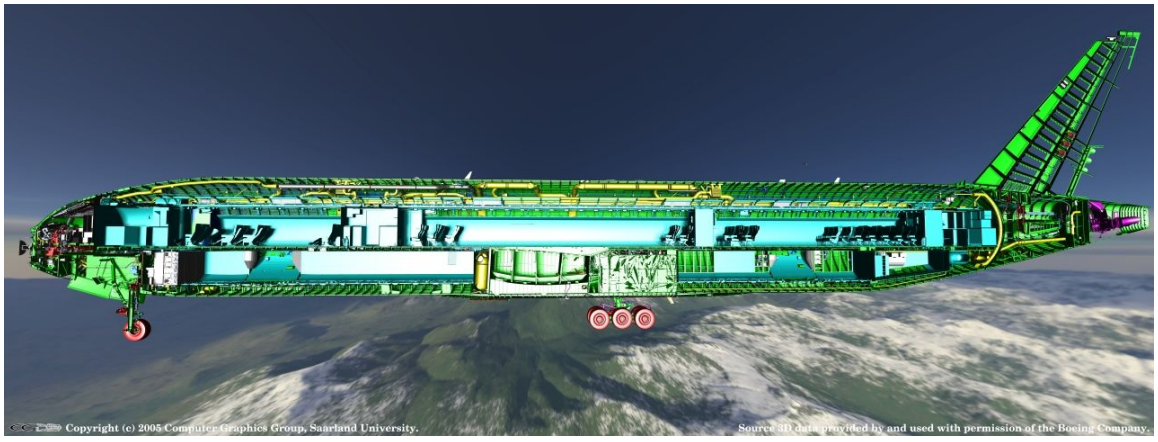
In the outlook we touch on modern multi GPU hardware architecture, the use of programmable shaders in the context of VGR, as well interactive object manipulation functionality. The presentation concludes with a life demo of our software.

Section outline:

1. Basics of Visibility-guided Rendering - VGR (15 minutes)
    a. Strengths and limitations of sampling and hardware rasterizing / z-buffer approaches
    b. Output-sensitive approach to rendering massive scenes
    c. Culling techniques
    d. Hardware visibility tests
    e. Spatial data structures and algorithms
2. System and Performance Issues for Real Time VGR (25 minutes)
    a. Spatial queries, pre-fetching, temporal and spatial coherence
    b. Preprocessing vs. real time.
    c. Rendering efficiency: Overdraw vs. nr. of queries
    d. Parallelism between GPU and CPU, avoiding pipeline stalls
    e. 2-level caching between hard disk, main memory and VRAM
    f. Multi-threading, memory management
3. Additional Features (5 minutes)
    a. Scene graph / object representation / interaction / dynamic data
    b. The use of pixel and vertex shaders
4. Live Demo (5 minutes)

1:00 – 1:50 Massive Model Visualization using Realtime Ray Tracing (Philipp Slusallek, Saarland University and inTrace)

Goal: Demonstrate how Real-Time Ray Tracing can easily deal with many of the challenges of visualizing massive models of surfaces and volumes while enabling advanced visual effects like shadows and global illumination. Both HW and SW approaches will be discussed and demonstrated.



Data courtesy The Boeing Company

Interactive placement of a cut-plane through the full model of a Boeing 777 (350 million triangles). The full detail of the model is rendered and smooth shadows are added for increased realism.

Interactively ray traced image of a complex outdoor environment consisting of roughly 1.5 billion polygons. The rendering includes shadows, smooth environment lighting, complex reflection properties, and more.

Real-time ray tracing has become an attractive alternative to rasterization based rendering, particularly for highly complex data sets including both surface and volume data. Ray tracing handles massive datasets well because of output sensitivity and the logarithmic complexity of the ray tracing computations with respect to the scene size. Ray tracing easily handles huge scenes as long as they fit into main memory. For even larger data sets, active memory management is necessary to always keep the working set in main memory and on-demand swap data in and out depending on its visibility.

All these operations require spatial index structures (e.g. kd-trees) that need to be built out-of-core and often offline. We will discuss efficient techniques for this task, including approaches that allow efficient memory management at runtime. In addition, we will discuss several extensions that are necessary for efficiently ray tracing large models.

An intrinsic property of ray tracing is that once a scene can be ray traced, adding advanced optical effects or lighting simulation is fairly straightforward. We will discuss how such advanced effects can be used for achieving photorealistic visualization even of highly complex models such as natural environments with environment lighting, complex shading, and efficient anti-aliasing.

Handling dynamic scenes has been a major issue with ray tracing. We outline two areas where large progress has recently been made: Designing scenes graphs for efficiently

handling changes in very large models and novel index structures that allow fast updates after changes to the geometry.

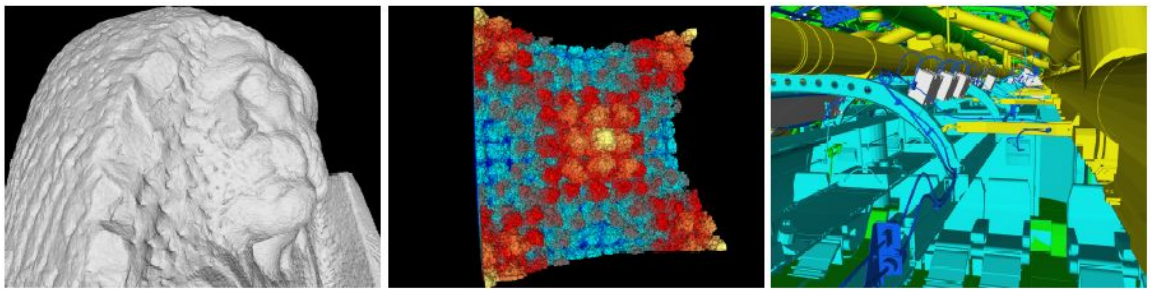
Finally, we will discuss trends in hardware and how they will help in making ray tracing available to a larger and larger set of applications. In particular we will briefly compare the capabilities of multi-core CPUs (including the Cell processor), GPUs, and custom hardware with respect to ray tracing of complex models.

Section outline:

1. Simplicity of data preparation for ray tracing complex scenes
    a. Efficient spatial index structures for ray tracing
    b. Off-line Construction of Index Structures
2. Adapting Ray-Tracing to Complex Models
    a. Active memory management
    b. Asynchronous data loading
    c. Level-of-detail management
3. Interactive view manipulation
4. Photorealistic Rendering and Lighting Simulation in Highly Complex Models
    a. Efficient integration of environment Lighting
    b. Simulation of complex optical properties
    c. Handling aliasing for complex geometries
5. Example: outdoor environments
6. Brief Review of Hardware-Trends for Realtime Ray Tracing
    a. Comparing Multi-core CPUs, GPUs, Cell-processor, and custom hardware

1:50 – 2:40 GPU-friendly accelerated mesh-based and mesh-less techniques for the output-sensitive rendering of huge complex 3D models (Enrico Gobbetti, Director of Visual Computing, CRS4)

Goal: Provide clear understanding of output sensitive techniques for models that exhibit complicated geometry and topology, heterogeneous material attributes, as well as large variations in depth complexity. Describe how these techniques can be efficiently implemented on what are now commodity graphics platforms.



Data courtesy The Boeing Company

In recent years, the large entertainment and gaming market has resulted in major investments in commodity graphics chip technology, leading to state-of-the-art programmable graphics units (GPUs) with greater complexity and computational density than current CPUs. GPUs are not only powerful, ubiquitous, and cheap, but their programmability is leading to new ways to tackle the large scale data visualization problems.

This section of the tutorial will discuss GPU friendly output sensitive techniques for harnessing the raw power and programmability features of these chips to interactively render very large complex 3D models. In this context, we will discuss and compare two different approaches: a mesh-based framework based on multi-scale geometric models (Batched Multi-Triangulation, IEEE Viz 2005), that is well suited to models with dense geometric details, and a mesh-less framework (Far Voxels, SIGGRAPH 2005), that handles datasets that combine complicated geometry and appearance with a large depth complexity by modeling model appearance rather geometry.

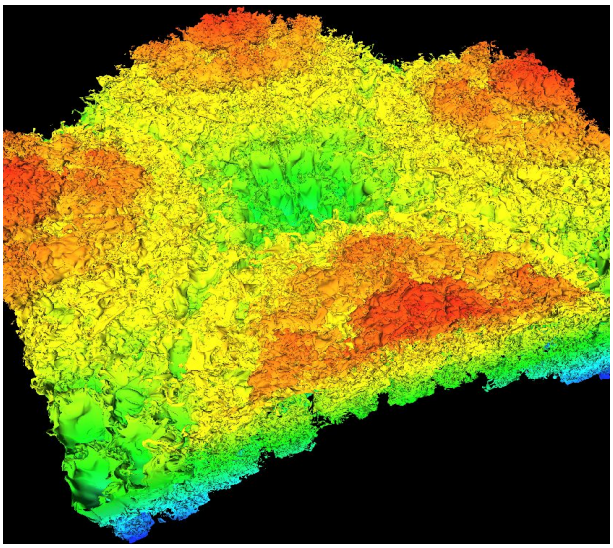
The tutorial section will be illustrated with practical examples of the visual inspection of very different kinds of models, including very large CAD assemblies, terrains, isosurfaces, and laser scans visualized on a laptop.

Section outline:

1. The need for multi-resolution structures
2. Handling different kinds of models: size and visual complexity issues
3. Multi-scale geometry vs. Multi-scale appearance
4. A mesh based framework based on multi-scale geometric modeling
    a. Batched Multi-Triangulations (BMT): concept and implementations
    b. BMT specializations: BDAM, Tetrapuzzles, V-Partitions
    c. Harnessing GPU programmability: P-BDAM
5. A mesh-less framework based on multi-scale appearance modeling
    a. Far Voxels: concept and implementation
    b. Practical issues and test cases
6. Real-time demos

2:55 – 3:45 Interactive Out-Of-Core Visualization of Large Datasets on Commodity PCs (Wagner Correa, IBM)

Goal: Provide understanding of octrees, precomputed coefficients, levels of detail, and multi-threaded programming.



This section of the tutorial will focus on interactive visualization of large datasets on commodity PCs. Interactive visualization has applications in many areas, including computer-aided design, engineering, entertainment, and training. Traditionally, visualization of large datasets has required expensive high-end graphics workstations. Recently, with the exponential trend of higher performance and lower cost of PC graphics cards, inexpensive PCs are becoming an attractive alternative to high-end machines. But a barrier in exploiting this potential is the small memory size of commodity PCs. To address this problem, we will present out-of-core techniques for visualizing datasets much larger than main memory.

We will start by presenting out-of-core preprocessing techniques. We will show how to build a hierarchical decomposition of the dataset using an octree, precompute coefficients used for visibility determination, and create levels of detail.

We will then present out-of-core techniques used at runtime. We will describe how to find the visible set using a fast approximate algorithm followed by a hardware-assisted conservative algorithm. We will also show how to use multiple threads to overlap visibility computation, cache management, prefetching, and rasterization.

We will finish by describing a parallel extension of the system that uses a cluster of PCs to drive a high-resolution, multi-tile screen. A thin client process manages interaction with the user, and a set of server processes render the multiple screen tiles. Large shared file systems (network or server-attached) provide storage for the complex dataset.

A system based on these techniques is a cost-effective alternative to high-end machines, and can help bring visualization of large datasets to a broader audience.

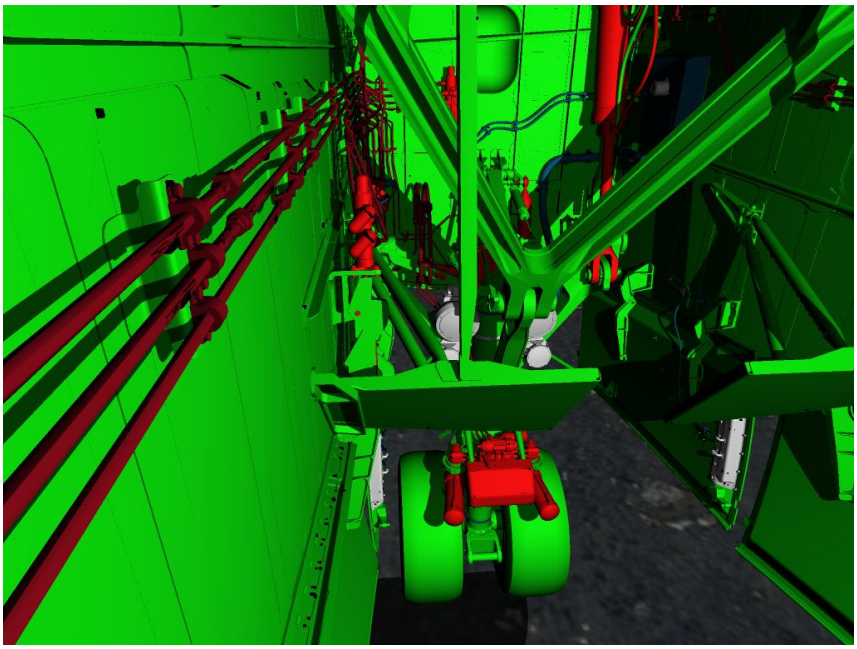
Section outline:

1. out-of-core preprocessing
    a. spatialization using an octree

    b. precomputation of static levels of detail
    c. precomputation of visibility coefficients
2. out-of-core rendering
    a. approximate visibility computation
    b. conservative visibility computation
    c. multi-threaded rendering, caching, and prefetching
3. out-of-core distributed parallel rendering
4. (Potential) cell processing technology and remote visualization

3:45 – 4:35 Putting Theory into Practice (Inigo Quilez, VRContext)

Goal: Review the most common problems, limitations, and considerations that arise from using state-of-the-art techniques coming from the research community into a real product.



Data courtesy The Boeing Company

Moving academic results to a real product is not always easy. Not only is a big part of the papers normal focus on very specific cases but research experiments are also often made with different constraints than those found in production. Even when applicable to a product, many techniques have been only tested in a few well known models used in CG literature, while customers demand algorithms to work in all kind of data, from best case models to ill-formed geometry, that must not be removed from the dataset and that can degrade significantly an algorithm's performance (as accelerating data structures).

On the other hand, the kind of data used in most 3D massive datasets visualization or collision detection research papers handle high density (e.g., the happy Buddha) or medium density models (the PowerPlant or the Boeing 777). However, a product has to

also deal with low density massive models. Low density models make some techniques not applicable or less efficient and add a new set of problems.

Another common source of problems is that many techniques assume that pre-computation time and effort is not important for the user. In practice, both the pre-calculation time and the complexity of the pre-compute process is of significant concern to production users. From the marketing point of view, there is also a resistance present in users to change to new technologies (even if a lot better than old ones); and lot of work must be done to give the application the look and feeling of the tools users are already used to.

Finally, we will demonstrate how a visualization and collision detection on massive model looks like in a real time application, both using OpenGL and software ray-tracing.

Section outline:

1. Problems
    a. Scalability of research methods
        i. advanced shading
        ii. shadows
    b. Ill-formed data
        i. t junctions
        ii. non-orientable geometry
        iii. clashes
        iv. noise.
    c. Low density massive models
    d. Precision problems
        i. in rendering
        ii. in collision detection
2. Limitations
    a. Solution cost
    b. Preprocessing time for HQ acceleration structures
    c. Marketing
3. Hidden but mandatory work
    a. Exporters
    b. SDKs
        i. how to hide implementation, but still keep efficiency
4. Realtime Demos
    a. OpenGL in massive models
    b. Ray tracing in massive models

4:35 – 17:00 Panel discussion (All, Moderator: David Kasik)

Presenter Information

Dave Kasik

Technical Fellow
The Boeing Company
david.j.kasik@boeing.com

Dave Kasik is the Boeing Enterprise Visualization Architect. His research interests include innovative combinations of basic 3D graphics and user interface technologies and increasing awareness of the impact of visualization technology inside and outside Boeing. Dave has a BA in Quantitative Studies from the Johns Hopkins University and an MS in Computer Science from the University of Colorado. He is a member of IEEE, ACM, ACM SIGGRAPH (he has attended all SIGGRAPH conferences), and ACM SIGCHI. He is a member of the editorial board for IEEE Computer Graphics and Applications.

Dinesh Manocha

Professor
University of North Carolina
dm@cs.unc.edu
http://www.cs.unc.edu/~dm/

Dinesh Manocha is currently a professor of Computer Science at the University of North Carolina at Chapel Hill. He was selected as an Alfred P. Sloan Research Fellow. He received NSF Career Award in 1995, Office of Naval Research Young Investigator Award in 1996, Honda Research Initiation Award in 1997, and the Hettleman Prize for scholarly achievement at UNC Chapel Hill in 1998. He has also received best paper and panel awards at ACM SuperComputing, ACM Multimedia, ACM Solid Modeling, Pacific Graphics, IEEE VR, IEEE Visualization, and Eurographics. He has served on the program committees and editorial boards of leading conferences in computer graphics and geometric modeling.

Manocha has been working on large model visualization for more than 10 years. His research group at UNC Chapel Hill has published numerous papers on model simplification, visibility computations, large data management and integrating these techniques at ACM SIGGRAPH and other conferences. He has also organized SIGGRAPH courses on interactive walkthroughs, large model visualization, and GPGPU.

Abe Stephens

PhD Student
Scientific Computing and Imaging Institute

University of Utah
abe@sci.utah.edu
http://www.sci.utah.edu/~abe

Abe Stephens is a PhD student at the University of Utah working in the Scientific Computing and Imaging Institute under the direction of Dr. Steven Parker. His work focuses on interactive large data visualization using ray tracing. He has worked closely with Silicon Graphics to improve interactive ray tracing techniques on their platform. Abe received a BS in Computer Science from Rensselaer Polytechnic Institute in 2003.

Beat Bruderlin

Professor of Computer Science
Technical University of Ilmenau (Germany)
bdb@imp.tu-ilmenau.de
http://rabbit.prakinf.tu-ilmenau.de/bdb.html

Beat Bruderlin is professor of Computer Science at the Technical University of Ilmenau, Germany. His work focuses on computer geometry with applications to computer aided design and engineering visualization. Other interests include new interaction techniques for 3D design. Beat Bruderlin received his M.S. degree in Physics from the University of Basel and a PhD in Computer Science from the Swiss Federal Institute of Technology (ETH) – Zurich, Switzerland. He was a faculty member at the University of Utah, before joining TU Ilmenau. In 2004 he founded 3Dinteractive GmbH, a spin-off company developing interaction and rendering software.

Philipp Slusallek

Professor
Saarland University (Germany)
slusallek@cs.uni-sb.de
http://graphics.cs.uni-sb.de/~slusallek/

Philipp Slusallek is professor for computer graphics and digital media at Saarland University, Germany. Before joining Saarland University he was visiting assistant professor at Stanford University. He received a Diploma/MSc in physics from the University of Tübingen and a Doctor/PhD in computer science from the University of Erlangen. Philipp has published and taught extensively, including a SIGGRAPH05 course, about real-time ray tracing.

He is the principal investigator for the OpenRT project, which aims at establishing real-time ray-tracing as an alternative technology for interactive and photorealistic 3D graphics. This work includes the development of a highly optimized ray tracing software, custom hardware for ray tracing, approaches to

massive model visualization, and real-time lighting simulation algorithms. Recently he co-founded "inTrace", a spin-off company that commercializes real-time ray tracing technology.

Enrico Gobbetti

Director of Visual Computing
CRS4 (Italy)
gobbetti@crs4.it
http://www.crs4.it/vic/cgi-bin/people-page.cgi?name='enrico.gobbetti'

Enrico Gobbetti is the founder and director of the Visual Computing (ViC) group at the Center for Advanced Studies, Research, and Development in Sardinia (CRS4). At CRS4, Enrico developed and managed a graphics research program supported through industrial and government grants. His research interests span many areas of computer graphics. His most recent contributions include a new breed of coarse-grained adaptive multiresolution techniques for processing and rendering large scale geometric models. Enrico holds an Engineering degree (1989) and a Ph.D. degree (1993) in Computer Science from the Swiss Federal Institute of Technology in Lausanne (EPFL). For more information, see www.crs4.it/vic

Wagner Correa

Research Scientist
IBM
wtcorrea@us.ibm.com
http://domino.research.ibm.com/comm/research_people.nsf/pages/wtcorrea.index.html

Wagner Correa is a Research Staff Member with the IBM Watson Research Center. He is part of the Visualization Systems Group, which recently released the Deep Computing Visualization (DCV) suite of programs for immersive and remote visualization. Prior to joining IBM, Wagner was teaching Computer Graphics at the Federal University of Minas Gerais (UFMG) in Brazil. Wagner holds a B.S. degree (1994) and an M.S. degree (1996) in Computer Science from UFMG, and an M.A. degree (1998) and a Ph.D. degree (2004) in Computer Science from Princeton University.

Inigo Quilez

Visual Technologist
VRContext (Belgium)
i.quilez@vrcontext.com
http://www.vrcontext.com
http://rgba.scenesp.org/iq/

Iñigo Quilez received a degree as a Telecommunications Engineer from the University of Basque Country (Spain), with intensification in digital signal processing. He has extensively worked in real-time computer graphics, within the "demoscene" since 1998, especially in the subject of extreme procedural content creation and data compression. Well known in the fractals community, work is still needed to give aesthetics a more important role in the scientific work.

Since he joined VRcontext in 2003, his work focuses on research and development of photorealistic rendering and massive model visualization techniques among others, focusing in shared memory multi-cpu and multipipe systems (especially the Silicon Graphics architecture).