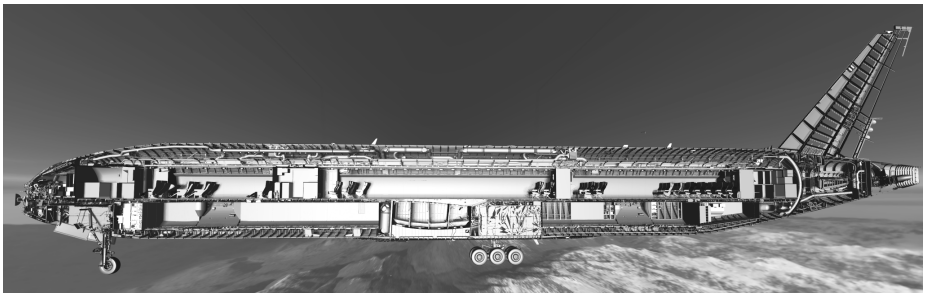
Large-Scale CAD Model Visualization on a Scalable Shared-Memory Architecture

Andreas Dietrich[†], Ingo Wald[‡], and Philipp Slusallek[†]

[†]Computer Graphics Group,
Saarland University, Saarbrücken
{dietrich,slusallek}@cs.uni-sb.de

[‡]Max-Planck-Institut Informatik,
Saarbrücken
wald@mpi-inf.mpg.de



Abstract

One of the most pervasive problems in large-scale engineering projects is the difficulty in properly fitting all individual parts together. The prohibitively high investment of using physical mockups has led to pre-assembly being performed almost entirely digital. Unfortunately, the vast complexity of full CAD datasets can not be handled by available high-end graphics hardware. In this article we present a ray tracing based software system running on a scalable shared-memory architecture, which allows for interactive high-quality visualization and evaluation of huge CAD models. Special features like cutting planes, model interrogation, sophisticated shading, and collaborative remote visualization are also supported. The capabilities of our framework will be demonstrated on a practical example, the collaborative design review of a complete Boeing 777 airliner.

1 Introduction

Computer Aided Design (CAD) has practically become ubiquitous in all of today's large-scale industrial engineering projects. This development has lead to the purely digital design of complete aircrafts, ships, cars, etc. In such a process a large number of concurrently working design teams are involved [1], resulting in the development of typically thousands of different parts, each modeled with the highest possible accuracy.

According to studies the most eminent problems in large-scale manufacturing are potential overlaps of assembly parts, as well as the difficulty in properly fitting all individual components together in final assembly. In order to avoid traditional physical mockups that make planning and construction extremely expensive, there naturally arises the need to perform pre-assembly on a completely digital basis. It would therefore be desirable to directly perform a 3D visualization of the full CAD dataset with all its detail, thereby providing a greater analysis context during design reviews.

Unfortunately, CAD models for large-scale engineering projects tend to become extremely large: Before all individual parts are eventually assembled, each individual component is usually modeled independently at full geometric accuracy affordable on the designer's workstation. With every individual component having full geometric detail, the complete database can contain up to billions of individual polygons, which cannot be efficiently handled by the sequential *triangle rasterization* approach implemented in todays graphics cards, even when using the most high-end graphics hardware.

[1] In the case of the Boeing 777 airplane program more than 230 geographically dispersed groups had to be coordinated.

As a result, usual virtual reality systems only display manually selected parts of the complete scene, or rely on geometric simplifications that often require manual tuning, and are prone to rendering artifacts.

1.1 Parallel Ray Tracing

For real-time display of highly complex models, *ray tracing* provides a better alternative. Ray tracing algorithms [6] closely model physical light transport by shooting rays into the virtual scene. By employing spatial index structures, ray-object intersections can be found efficiently, resulting in a logarithmic time complexity with respect to scene size. Additionally, because of the algorithm's *output sensitivity*, only data that is actually visible is eventually accessed.

Since the colors of different pixels can be calculated independently of each other, ray tracing offers an extremely high degree of parallelism. By assigning different pixels to different processing units, it is therefore possible to reach even real-time performance. This was first shown by Muuss [10] and Parker et al. [11], who demonstrated interactive ray tracing using massively parallel shared-memory supercomputers. More recently Wald et al. [15, 14] have shown that interactive frame rates can also be achieved on clusters of low-cost commodity PCs. Although the use of PC clusters enables linear scaling in performance, memory scalability still remains a problem. Because every cluster node might potentially need to access the complete model, the scene database has to be replicated on each PC. For complex industrial CAD models of dozens or hundreds of GBytes in size, this is not feasible. Special PC-based out-of-core variants for ray tracing massively complex models exist as well [16], but cannot yet deliver the performance and quality demanded by industrial application scenarios.

1.2 Contributions

In this paper we present a ray tracing based interactive visualization system, suited for display and design evaluation of extremely large CAD models *without* approximations, simplifications, or rendering artifacts. By efficiently combining a highly optimized ray tracing engine with a shared-memory multiprocessor architecture, it is possible to do real-time walkthroughs in large-scale highly detailed scenes, which is demonstrated at the example of a complete Boeing 777, consisting of more than 350 million individual polygons. Additionally, our system incorporates several features required for design review, such as distance measurement between arbitrary points, interactive identification and movement of individual model components, and sophisticated shading (including soft shadows and highlights).

With the help of the OpenGL Vizserver frame buffer streaming system, there is even the possibility to do the compute intensive image generation on a centralized visualization server, while the walkthrough can be controlled from remote lightweight clients, even over standard Internet wide-area connections.

1.3 Paper Overview

The remainder of the paper is structured as follows: Section 2 starts with a brief overview over some existing massive model walkthrough systems. Section 3 will then provide some insight into our ray tracing software, the underlying shared-memory multiprocessor architecture, and the remote visualization features of the system. We will demonstrate capabilities and features using the example of a complete Boeing 777 aircraft in Section 4. We conclude in Section 5, followed by some thoughts about future extensions in Section 6.

2 Related Work

Due to its practical relevance, the problem of visualizing massively complex models has already received a lot of attention, which we will briefly discuss.

2.1 Rasterization Based Systems

The UNC GigaWalk system [2] runs on an SGI Onyx workstation (300 MHz MIPS R12000 CPUs, 16 GByte RAM) with Infinite Reality graphics, and makes use of two rasterization pipes and three processes running in parallel on individual CPUs. The visible geometry of each frame is treated as potential occluders for successive frames. Using occlusion culling based on these occluders in combination with a Hierarchical Z-Buffer [7], the system is reported to be able to render scenes with up to 82 million triangles at 11-50 frames per second.

Another recently proposed framework is iWalk [5]. It can handle models consisting of up to 13 million triangles at 9 frames per second on a single commodity PC (2.8 GHz Intel Pentium 4 CPU, 512 MByte RAM) with an NVIDIA Quadro 980 XGL card. However, the system relies on approximated visibility, and uses an object-space algorithm [9] to estimate a potentially visible geometry set, which can result in visible polygons being omitted.

In contrast to the above mentioned applications that are primarily meant for visualization only, the Boeing FlyThru [1] system, a proprietary in-house application originally conceived for the 777 twin-engine airliner program (see Section 4), comprises a great number of features aiding collaborative CAD. Apart from displaying thousands of parts at one time, it facilitates detection of motion anomalies and interference between structures, interactive design reviews across a network, modeling, kinematics, and remote control by other applications. Unfortunately, no detailed information about its interactive rendering capabilities is available. It can, however, not display the full 777 dataset at real-time rates without geometric simplifications [8].

2.2 Ray Tracing Based Systems

As sketched in Section 1.1, ray tracing technology efficiently supports interactive visualization of large unsimplified datasets. The OpenRT real-time ray tracing engine [15, 14] has been shown to be capable of handling scenes with up to several million triangles in real-time. On a setup of 24 commodity dual-processor PCs (AMD AthlonMP 1800+ CPUs, 512 MByte RAM) this system has been reported to achieve up to 23 frames per second. Additionally, it incorporates physically correct and global lighting simulations [3], and features interactive placement of geometric parts. It relies, however, on the fact that each cluster node can keep the complete scene in main memory.

Wald et al. [17] have also presented an out-of-core rendering variant of the OpenRT system that combines explicit memory management, demand-loading of missing parts, and computation reordering. While this system has been shown to render scenes that are much larger than main memory, it can only handle scenes where only a small fraction of data has to be loaded between successive frames, and does not easily scale to scenes of a more realistic complexity.

In a more recent publication [16] it was demonstrated that even on a single desktop PC (dual 1.8 GHz AMD Opteron 246, 6 GByte RAM), out-of-core ray tracing can be used for interactively visualizing a complete Boeing 777 CAD dataset containing more than 350 million individual surface polygons. Even including the calculation of pixel-accurate shadows and highlights, the system reaches up to 5 frames per second. Due to the out-of-core nature of the approach, model parts are only loaded on demand, and – as not all missing data can be loaded within the same frame – an approximation scheme has to be employed to represent data not loaded yet. This frequently leads to rendering artifacts that are not tolerable for practical applications. Additionally, the framework does not easily parallelize due to the need to synchronize all memory operations on all client machines, and thus cannot deliver sufficient performance.[2]

3 Visualization System Outline

The presented rendering architecture basically builds on the system of Wald et al. [16] with OpenRT as ray tracing core. The rendering artifacts introduced through the out-of-core mechanism required on a PC platform made this system not applicable for practical applications. In contrast, the eventual end users of our visualization system explicitly demanded display of an entire complex dataset at *any time*, without *any* kind of approximations, demand-loading stalls, or rendering artifacts.

To meet these demands, it was decided to port the initial system to a scalable shared-memory multiprocessor architecture, and thereby couple the performance scalability of the OpenRT system with the memory scalability of this platform.

3.1 Hardware Architecture

All our experiments were conducted on an SGI Altix 350 mid-range server [12], composed of 8 dual-processor nodes. Each of the nodes is equipped with two Intel Itanium 2 CPUs clocked at 1.4 GHz, and contains 4 GByte local memory.

[2] Design reviews are usually considered to require 10-20 frames per second.

In this setup, the memory banks of the nodes form a system-wide 32 GByte large, shared-memory address space. This is made possible by the Altix NUMAflex architecture (see Figure 1) that provides a low-latency, high-bandwidth interconnect between the distinct nodes, gaining peak transfer rates of up to 6.4 GByte per second. As this works completely application transparent, each CPU can directly access every desired part of the model in the global memory space. The geometric database is actually distributed over the nodes' physical memory banks without any replication of data.

By just adding new nodes – and connecting them to the NUMAflex interconnect – the Altix can easily be scaled in both memory and number of CPUs. Therefore, the current framework can easily be scaled to almost arbitrary model sizes and performance requirements. Although all of the following results are reported for a 16-processor setup only, the same software system is currently also being evaluated on significantly larger installations.

3.2 OpenRT

The OpenRT real-time ray tracing core [14] serves as a high-performance rendering back-end for our 3D CAD browser application. It supports physically correct lighting simulation, plug-and-play shading by means of dynamically loaded shader libraries (i.e. custom programs that perform the actual light propagation calculations), and handling of dynamic and complex 3D environments.

3.2.1 Client-Server Rendering

Highly optimized code, and distributing computation among several parallelly working CPUs, allows the OpenRT engine to reach interactive and even real-time frame rates. In this client-server approach a single master process centrally manages a number of client processes: The image is decomposed into a number of disjunct regions that are asynchronously assigned as tasks to the clients *on demand*. After a client has finished computation of an assigned image-tile, it sends back the respective pixel color values to the master, which composes them into the resulting image. Although, the system has been specifically designed to run on a cluster of PCs, the setup on the Altix is practically the same. All client processes are started on the same machine as the master process, while the operating system takes care of distributing the processes among the available CPUs.

3.2.2 Memory Management

Memory management of large CAD databases can be done in a very straightforward manner. Since the Altix provides enough RAM to keep the full model in memory, the whole dataset (including all spatial index structures) it simply mapped from disk into the global address space, using Linux memory mapping facilities. This can be independently done by each client process because the operating system takes care that no part is paged into shared memory more than once. Although OpenRT incorporates a memory management subsystem that can deal with scenes larger than main memory in an out-of-core fashion (see [16] for details), this is not required here.

3.3 Remote Visualization

For the purpose of collaborative design reviews, the Altix can also act as a centralized visual server for multiple clients in geographically diverse locations. To this end the system make use of the OpenGL Vizserver [13] technology: A frame rendered on the visualization server is captured, and the compressed pixel data is sent to the clients over standard local as well as wide-area networks. Each client then uncompresses the pixel stream, displays the uncompressed image, and directs back all user interaction to the server. As only the final image is transmitted, the clients themselves do not need any high-performance graphics capabilities at all, and can thus be lightweight clients such as desktop PCs or laptops.

In order to be as transparent to the application as possible the Vizserver installs wrapper libraries that monitor all calls to the OpenGL or X11 window system libraries. A buffer swap in a window, for example, triggers a read back of the frame buffer on the server, which is then sent to the client. Only the region of interest used for rendering is transferred, including GUI widgets and OpenGL rendering area. In case of low-quality network connections, the server can drop and repeat frames, as well as use several kinds of (lossy and lossless) compression rates and mechanisms.

Although a simple video streaming approach could have been more efficient, especially since we

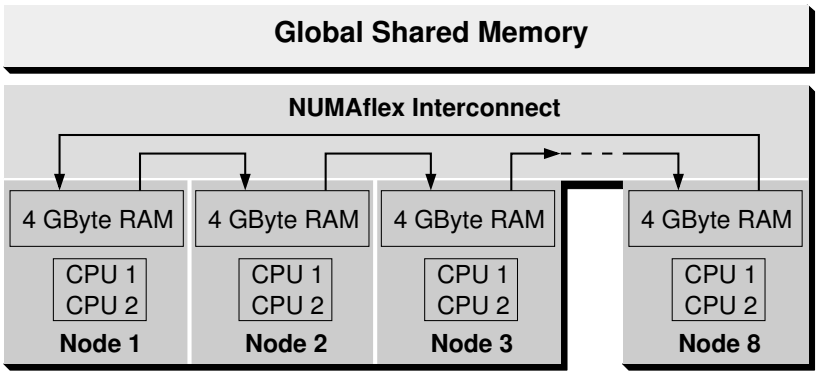


Figure 1: Altix global shared memory. Using an application transparent NUMAflex interconnect between individual nodes, a large global shared memory address space is formed, which can be directly accessed by each CPU. Though our example setup uses only 8 nodes, the architecture can also be scaled to much larger configurations.

do not require support for hardware accelerated rendering, we opted for the Vizerserver because it provides a stable, mature, and widely used industrial remote visualization framework.

4 Application: Design Review of a Complete Boeing 777

One of the main objectives we targeted was the interactive walkthrough of a fully-detailed 3D model of a Boeing 777 twin-engine airplane. There should not only be the possibility to directly render every single part of the original CAD data without any kind of geometric simplification, visual approximation, or artifacts. The system should also be suited for engineering design review sessions.

4.1 The Boeing 777 Model

The Boeing 777 model used in our experiments results from a direct export of the original construction CAD data out of the CATIA CAD/CAM system. Although some components are missing, the model already consists of more than 350 million individual surface triangles. Organized in over 13,000 compressed files, all components, including cables, screws, valves etc., have been modeled at extremely high accuracy. Without any additional spatial index structures, the raw model requires more than 12 GByte of hard disk space. Because the polygons were provided without any mesh connectivity information (i.e. coming as a "soup of triangles"), and with all vertices being randomly displaced to prevent data theft, the model is extremely difficult to handle for surface simplification algorithms found in most large model rendering systems.

4.2 Visualization Workflow

For the purpose of efficient model access during ray traversal calculations, spacial index structures are needed in addition to the geometric triangular surface information. In a first step, the original files are decompressed, parsed, and transformed into an unordered triangle stream. This stream gets then sorted into a k-d tree [4], which is stored in binary form. Like the ray tracing engine, the preprocessing tool chain can make use of multiple processors. Thus, it is able to preprocess the incoming data in a parallel manner during approximately 2 hours.

Including all additional index data, the resulting binary data files cover roughly 20 GByte of hard disk space. Since the files fit completely into main memory, they can then be copied into the Altix RAM disk, from where they are mapped into main memory. This enables the ray tracing engine to virtually start in an instant, and to provide the first images after less than 30 seconds.

Upon startup of the 3D browser application, all the binary files are mapped into the Altix global shared memory space, and are therefore immediately visible in the address space of each client process. A user can now freely browse the fully-detailed model, without having to wait for data being fetched from disk, and without encountering visual artifacts caused by not yet loaded data.

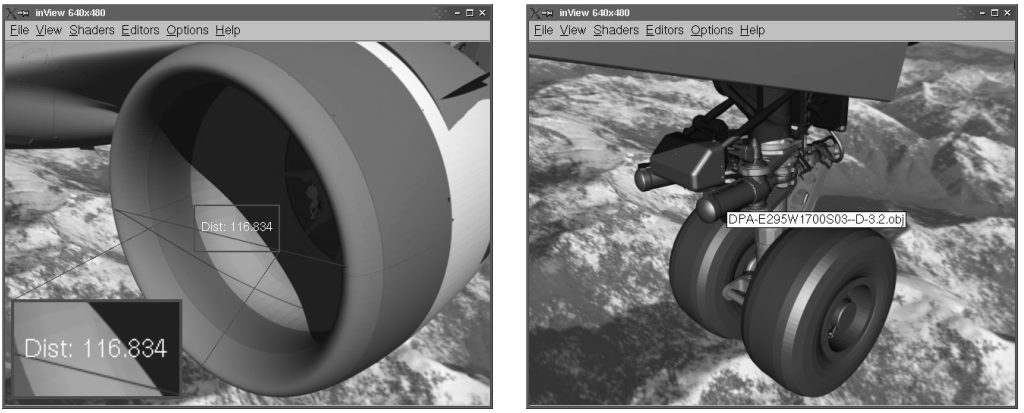


Figure 2: 3D CAD review features: (a) Measuring the diameter of a Boeing 777 engine. (b) All components can be pixel-accurately identified by simply moving the mouse pointer over them.

4.3 Design Review Functionality

The prime requisite for our system to be useful for design reviews is to deliver high-quality real-time rendering performance. In particular, these goals were specified by the users as: 1. The system should achieve at least 10 frames per second at a resolution of 640×480 pixels, even for complex views and during interaction. 2. It should not generate *any* visual artifacts at all during rendering, especially it should not generate any approximate views (as done in [16]). 3. It should have maximum startup times of only a few seconds. Using the afore-mentioned visualization system, where the distributed ray tracing engine delivers real-time performance, and the global shared memory of the Altix allows for keeping the entire model data in main memory, these demands can be fulfilled.

Apart from the capability of interactively displaying arbitrary parts of the Boeing 777 model, our visualization framework offers a number of additional functions required for collaborative CAD evaluation.

4.3.1 Distance Measurement

A very important feature that eases fitting together a model's components, is the ability to measure the exact distance between arbitrary three-dimensional points in the dataset. The user simply has to click at two different points in the browser window. By shooting rays from the projection center through the respective pixels into the scene, the ray tracer can easily find the distance between the visible surface points and the observer. The application can optionally insert a line object into the scene that helps visualizing the connection between the two points in question. The distance value is also shown besides the line (Figure 2a). Because this line object (actually a slim box) behaves like any other geometric object, it too can cast shadows that provide important visual cues on the exact location of that measuring line.

4.3.2 Object Identification

By applying the same technique, i.e. firing a ray through the pixel the mouse pointer is currently hovering above, not only the distance to a surface point can be determined. Because the ray tracing core can provide the front-end application with an identification of the object being hit by the ray, arbitrary information regarding this part can be looked up and displayed (Figure 2b). Few application-specific code (except displaying the identification windows) was required for that feature, as the usually complex picking-operation could easily be realized by using the ray tracer.

4.3.3 Cutting Planes

One of the advanced features of the ray tracing core is its ability to instantly cut away large parts of a model by specifying a number of freely orientable clipping planes. This works most efficiently for a ray tracer since it simply has to clip rays, whereas rasterization techniques need to clip all potentially visible polygons. Although inserting a cutting plane

Figure 3: Advanced rendering features: (a) An axis-aligned cutting plane slicing the airplane in half. The resulting cross-section view gives a much better insight into the model's overall structure. Multiple freely orientable cutting planes can be placed interactively. (b) Soft shadow effects in the cockpit providing a better impression of the relative placement of components.

can completely change the set of visible triangles, this is not a problem at all for our system: Since all scene data completely resides in memory, even such a drastic change of the visible set does not introduce any loading cost.

Cutting planes are particularly useful for structural analysis. For example, they easily allow for producing cross-sectional views (Figure 3a) that may, for example, serve as technical illustrations. Note that these cutting planes do not simply cut away the geometry, but can be configured to only affect viewing rays, therefore making it possible to look into the airplane without influencing e.g. the shadow computations inside.

4.3.4 Sophisticated Shading

Due to accurate simulation of physical light transport, sophisticated shading and lighting (e.g. pixel-accurate shadows, highlights, or reflections off of curved surfaces) can easily be incorporated in a plug-and-play fashion. In particular for complex geometry, the projection of 3D data onto a 2D display often incurs an undesired loss of depth impression. In that case, shadows often significantly help in the perception of the relative position of objects. Especially *soft* shadows help in judging the distance between shadow caster and receiver. Figure 3b shows the impact of rendering soft shadows that can significantly enhance the impression of shape and depth (see also Figure 2a).

5 Summary and Conclusions

In this paper we have shown that the combination of a shared-memory multiprocessor architecture and a high-performance ray tracing implementation can be efficiently used for real-time walkthroughs of highly detailed, large-scale industrial CAD databases. We have demonstrated that even a complete model of a Boeing 777 aircraft can be handled without approximation, simplification, or rendering artifacts. Our system supports several features important for virtual design review sessions, like distance measurement, identification, interactive placement of individual model components, and sophisticated shading.

The proposed system is currently being evaluated at Boeing, SGI, and Dassault on how it can best be integrated into the digital design workflow of large-scale engineering projects, even more complex than the 777 program.

6 Future Work

One possible direction of further research is to enhance the current setup into a "visualization service" similar to the grid computing philosophy. For example, instead of using one fixed visualization server, it would be possible to transparently provide a visualization service onto which the clients could connect without even knowing which machine they are communicating with.

Another field is the investigation of real-time lighting simulation algorithms that build on the current architecture. In particular the Altix's shared memory model and facile scaling in available compute performance provides a huge potential for interactive global illumination, even in extremely complex datasets.

Acknowledgments

This project was supported by a large number of people. In particular, we would like to thank Boeing for providing the 777 airliner dataset, for funding the project, and for supplying valuable information on practical needs of end users. The hardware for this project, the remote visualization setup, as well as help in numerous instances was provided by Silicon Graphics, Inc. We are also indebted to the CGUdS system administration group for setting up the hardware. Finally, we would like to thank Krzysztof Kobus, Kai Renner, Gerd Marmitt, and *in*Trace GmbH. Disclaimer: Source 3D data provided by and used with permission of the Boeing Company.

References

[1] Robert M. Arbabanel, Eric Brechner, and William McNeely. FlyThru the Boeing 777. In *ACM SIGGRAPH, Visual Proceedings*, 1996.

[2] William V. Baxter III, Avneesh Sud, Naga K Govindaraju, and Dinesh Manocha. GigaWalk: Interactive Walkthrough of Complex Environments. In *Rendering Techniques 2002*, pages 203–214, 2002. (Proceedings of the 13th Eurographics Workshop on Rendering).

[3] Carsten Benthin, Ingo Wald, and Philipp Slusallek. A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum*, 22(3):621–630, 2003. (Proceedings of Eurographics).

[4] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] Wagner T. Corrêa, James T. Klosowski, and Cláudio T. Silva. Visibility-Based Prefetching for Interactive Out-Of-Core Rendering. In *Proceedings of Parallel Graphics and Visualization (PGV)*, pages 1–8, 2003.

[6] Andrew Glassner. *An Introduction to Ray Tracing*. Morgan Kaufmann, 1989. ISBN 0-12286-160-4.

[7] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical Z-Buffer Visibility. In *Computer Graphics (Proceedings of ACM SIGGRAPH)*, pages 231–238, 1993.

[8] David J. Kasik. Boeing Company. Personal Communication, 2005.

[9] James T. Klosowski and Cláudio T. Silva. The Prioritized-Layered Projection Algorithm for Visible Set Estimation. In *IEEE Transaction on Visualization and Computer Graphics*, pages 108–123, 2000.

[10] Michael J. Muuss. Towards Real-Time Ray-Tracing of Combinatorial Solid Geometric Models. In *Proceedings of BRL-CAD Symposium '95*, 1995.

[11] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan. Interactive Ray Tracing. In *Proceedings of Interactive 3D Graphics*, pages 119–126, 1999.

[12] Silicon Graphics, Inc. SGI Altix 350 Server. http://www.sgi.com/products/servers/alitx/350, 2004.

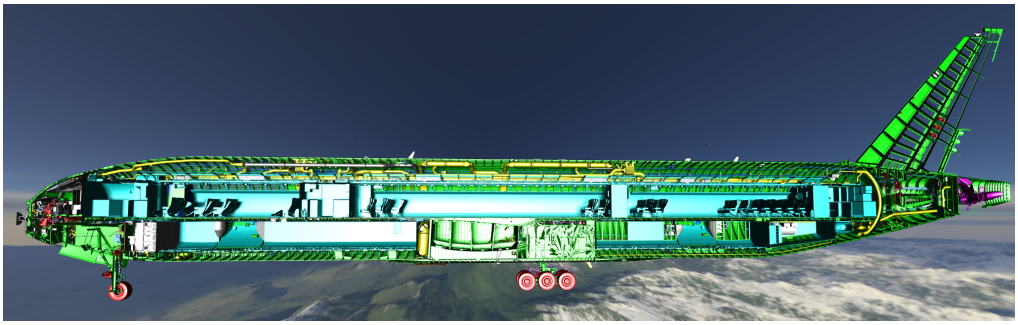
[13] Silicon Graphics, Inc. SGI OpenGL Vizserver. http://www.sgi.com/products/ software/vizserver, 2004.

[14] Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. Available at http://www.mpi-sb.mpg.de/~wald/PhD/.

[15] Ingo Wald, Carsten Benthin, Andreas Dietrich, and Philipp Slusallek. Interactive Ray Tracing on Commodity PC Clusters – State of the Art and Practical Applications. In *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference, 2003. Proceedings*, volume 2790 of *Lecture Notes in Computer Science*, pages 499–508. Springer, 2003.

[16] Ingo Wald, Andreas Dietrich, and Philipp Slusallek. An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models. In *Rendering Techniques 2004, Proceedings of the Eurographics Symposium on Rendering*, pages 81–92, 2004.

[17] Ingo Wald, Philipp Slusallek, and Carsten Benthin. Interactive Distributed Ray Tracing of Highly Complex Models. In *Rendering Techniques 2001*, pages 274–285, 2001. (Proceedings of the 12th Eurographics Workshop on Rendering).



Cross-section view of a Boeing 777 airplane. The model contains more than 350 million individual polygons, and has been directly exported from the original CAD database. Using fast ray tracing, every single part of the aircraft can be interactively inspected without any kind of simplification or approximation.

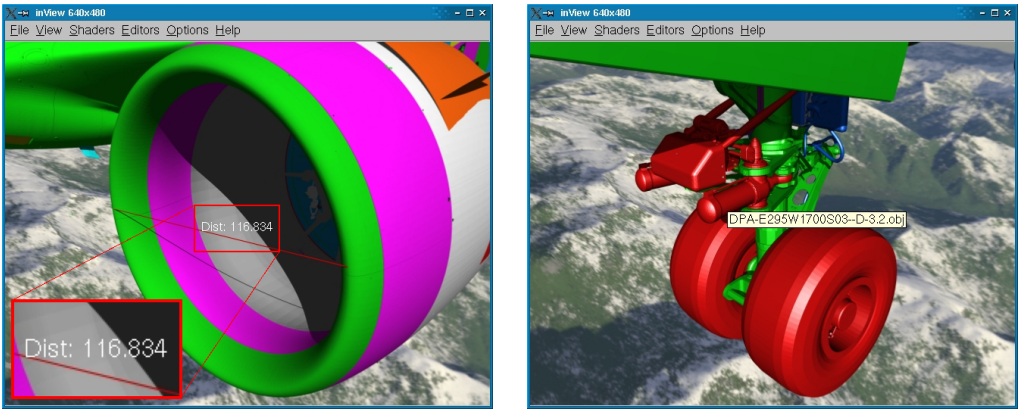


Figure 2: 3D CAD review features: (a) Measuring the diameter of a Boeing 777 engine. (b) All components can be pixel-accurately identified by simply moving the mouse pointer over them.

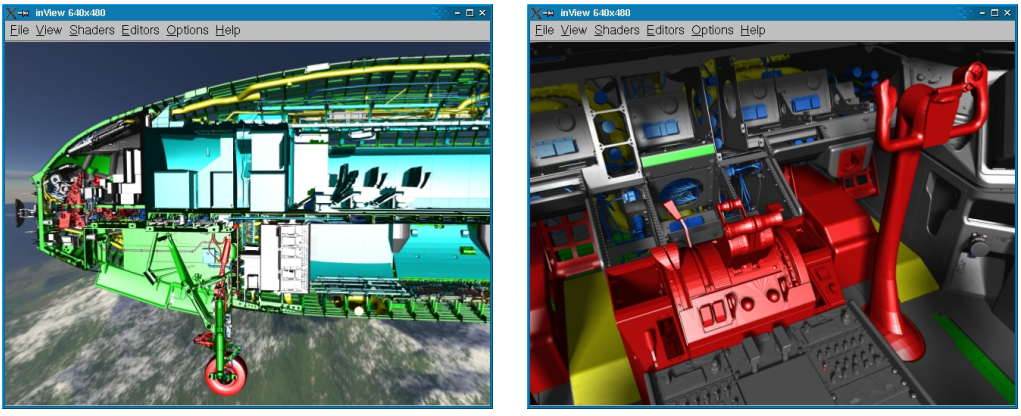


Figure 3: Advanced rendering features: (a) An axis-aligned cutting plane slicing the airplane in half. The resulting cross-section view gives a much better insight into the model's overall structure. Multiple freely orientable cutting planes can be placed interactively. (b) Soft shadow effects in the cockpit providing a better impression of the relative placement of components.