The DirectModel Toolkit: Meeting the 3D Graphics Needs of Technical Applications

Brian E. Cripe

Thomas A. Gaskins

The increasing use of 3D modeling for highly complex mechanical designs has led to a demand for systems that can provide smooth interactivity with 3D models containing millions or even billions of polygons.

DirectModel* is a toolkit for creating technical 3D graphics applications. Its primary objective is to provide the performance necessary for interactive rendering of large 3D geometry models containing millions of polygons. DirectModel is implemented on top of traditional 3D graphics applications programming interfaces (APIs), such as Starbase or OpenGL®. It provides the application developer with high-level 3D model management and advanced geometry culling and simplification techniques. **Figure 1** shows DirectModel's position within the architecture of a 3D graphics application.

This article discusses the role of 3D modeling in design engineering today, the challenges of implementing 3D modeling in mechanical design automation (MDA) systems, and the 3D modeling capabilities of the DirectModel toolkit.

Visualization in Technical Applications

The Role of 3D Data

3D graphics is a diverse field that is enjoying rapid progress on many fronts. Significant advances have been made recently in photorealistic rendering, animation quality, low-cost game platforms, and state-of-the-art immersive

* DirectModel was jointly developed by Hewlett-Packard and Engineering Animation Incorporated of Ames, Iowa.

Brian E. Cripe

With HP since 1982, Brian Cripe is a project manager at the HP Corvallis Imaging Operation. He is responsible for DirectModel relationships with developers, Microsoft®, and Silicon Graphics®. He has worked on the HP ThinkJet and DeskJet printers and the Common Desktop Environment. He received a BSEE in 1982 from Rice University. Brian was born in Anapolis, Brazil, is married and has two daughters.

Thomas A. Gaskins

Thomas Gaskins was the project leader for the DirectModel project at the HP Corvallis Imaging Operation. With HP since 1995, he received a BS degree in mechanical engineering (1993) from the University of California at Santa Barbara. He specialized in numerical methods. His professional interests include 3D graphics and software architecture.

Figure 1

Application architecture.

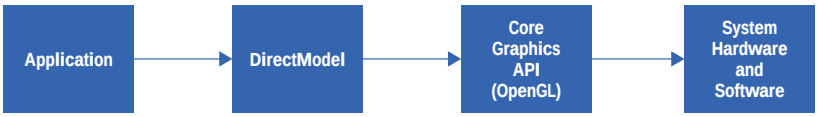
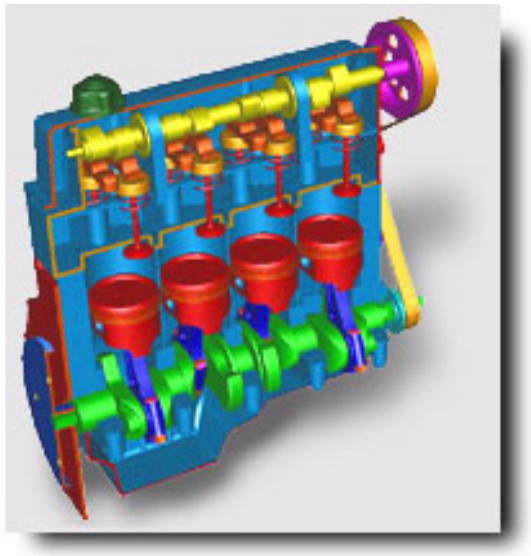


Figure 2

A low-resolution image of a 3D model of an engine consisting of 150,000 polygons.



virtual reality* applications. The Internet is populated with 3D virtual worlds and software catalogs are full of applications for creating them. An example of a 3D model is shown in **Figure 2**.

What do these developments mean for the users of technical applications (the scientists and engineers who pioneered the use of 3D graphics as a tool for solving complex problems)? In many ways this technical community is following the same trends as the developers and users of nontechnical applications such as 3D games and interactive virtual worlds. They are interested in finding less expensive systems for doing their work, their image quality standards are rising, and their patience with poor interactive performance is wearing thin.

However, there are other areas where the unique aspects of 3D data for technical applications create special requirements. In many applications the images created from the 3D data that are displayed to the user are the goal. For example, the player of a game or the pilot in a flight simulator cares a lot about the quality and interactivity of the images, but cares very little about the data used by the system to create those images. In contrast, many technical users of 3D graphics consider their data to be the most important component. The goal is to create, analyze, or improve the data, and 3D rendering is a useful means to that end.

This key distinction between data that is the goal itself and data that is a means to an end leads to major differences in the architectures and techniques for working with those data sets.

3D Model Complexity

Understanding the very central role that data holds for the technical 3D graphics user immediately leads to the questions of what is that data and what are the significant trends over time? The short answer is that the size of the data is big and the amount and complexity of that data is increasing rapidly. For example, a mechanical engineer doing stress analysis may now be tackling problems modeled with millions of polygons instead of the thousands that sufficed a few years ago.

The trends in the mechanical design automation (MDA) industry are good examples of the factors causing this growth. In the not-too-distant past mechanical design was accomplished using paper and pencil to create part drawings, which were passed on to the model shop to create prototype parts, and then they were assembled into prototype products for testing. The first step in computerizing this process was the advent of 2D mechanical drafting applications that allowed the mechanical engineers to replace their drafting boards with computers. However, the task was still to produce a paper drawing to send to the model shop. The next step was to replace these 2D drafting applications with 3D solid modelers that could model the complete 3D geometry of a part and support tasks such as static and dynamic design analysis to find such things as the stress points when the parts move. This move to 3D solid modeling has had a big impact at many companies as a new technique for designing parts. However, in many cases it has not resulted in a fundamental change to the process for designing and manufacturing whole products.

Advances. In the last few years advances in the mechanical design automation industry have increasingly addressed virtual prototyping and other whole-product

* Immersive virtual reality is a technology that "immerses" the viewer into a virtual reality scene with head-mounted displays that change what is viewed as the user's head rotates and with gloves that sense where the user's hand is positioned and apply force feedback.

Fahrenheit

Hewlett-Packard, Microsoft, and Silicon Graphics are collaborating on a project, code-named "Fahrenheit," that will define the future of graphics technologies. Based on the creation of a suite of APIs for DirectX on the Windows® and UNIX® operating systems, the Fahrenheit project will lead to a common, extensible architecture for capitalizing on the rapidly expanding marketplace for graphics.

Fahrenheit will incorporate the Microsoft Direct3D and DirectDraw APIs with complementary technologies from HP and Silicon Graphics. HP is contributing DirectModel to this effort and is working with Microsoft and Silicon Graphics to define the best integration of the individual technologies.

design issues. This desire to create new tools and processes that allow a design team to design, assemble, operate, and analyze an entire product in the computer is particularly strong at companies that manufacture large and complex products such as airplanes, automobiles, and large industrial plants. The leading-edge companies pioneering these changes are finding that computer-based virtual prototypes are much cheaper to create and easier to modify than traditional physical prototypes. In addition they support an unprecedented level of interaction among multiple design teams, component suppliers, and end users that are located at widely dispersed sites.

This move to computerized whole-product design is in turn leading to many new uses of the data. If the design engineers can interact online with their entire product, then each department involved in product development will want to be involved. For example, the marketing department wants to look at the evolving design while planning their marketing campaign, the manufacturing department wants to use the data to ensure the product's manufacturability, and the sales force wants to start showing it to customers to get their feedback.

These tasks all drive an increased demand for realistic models that are complete, detailed, and accurate. For example, mechanical engineers are demanding new levels of realism and interactivity to support tasks such as positioning the fasteners that hold piping and detecting interferences created when a redesigned part bumps into one of the fasteners. This is a standard of realism that is very different from the photorealistic rendering requirements of other applications and to the technical user, a higher priority.

Larger Models These trends of more people using better tools to create more complete and complex data sets combine to produce very large 3D models. To understand this complexity, imagine a complete 3D model of everything you see under the hood of your car. A single part could require at least a thousand polygons for a detailed representation, and a product such as an automobile is assembled from thousands of parts. Even a small product such as an HP DeskJet printer that sits on the corner of a desk requires in excess of 300,000 triangles[1] for a detailed model. A car door with its smooth curves, collection of controls, electric motors, and wiring harness can require one million polygons for a detailed model—the car's power train can consist of 30 million polygons.[2]

These numbers are large, but they pale in comparison to the size of nonconsumer items. A Boeing 777 airplane contains approximately 132,500 unique parts and over 3,000,000 fasteners,[3] yielding a 3D model containing more than 500,000,000 polygons.[4] A study that examined the complexity of naval platforms determined that a submarine is approximately ten times more complex than an airplane, and an aircraft carrier is approximately ten times more complex than a submarine.[5] 3D models containing hundreds of millions or billions of polygons are real today.

As big as these numbers are, the problem does not stop there. Designers, manufacturers, and users of these complex products not only want to model and visualize the entire product, but they also want to do it in the context of the manufacturing process and in the context in which it is used. If the ship and the dry dock can be realistically

modeled and combined, it will be far less expensive to find and correct problems before they are built.

Current System Limitations

If the task faced by technical users is to interact with very large 3D models, how are the currently available systems doing? In a word, badly. Clearly the graphics pipeline alone is not going to solve the problem even with hardware acceleration. Assuming that rendering performance for reasonable interactivity must be at least 10 frames per second, a pipeline capable of rendering 1,000,000 polygons per second has no hope of interactively rendering any model larger than 100,000 polygons per frame. Even the HP VISUALIZE fx^6, the world's fastest desktop graphics system, which is capable of rendering 4.6 million triangles per second, can barely provide 10 frames per second interactivity for a complete HP DeskJet printer model.

This is a sobering reality faced by many mechanical designers and other technical users today. Their systems work well for dealing with individual components but come up short when facing the complete problem.

Approaches to Solving the Problem

There are several approaches to solve the problem of rendering very complex 3D models with interactive performance. One approach is to increase the performance of the graphics hardware. Hewlett-Packard and other graphics hardware vendors are investing a lot of effort in this approach. However, increasing hardware performance alone is not sufficient because the complexity of many customers' problems is increasing faster than gains in hardware performance. A second approach that must also be explored involves using software algorithms to reduce the complexity of the 3D models that are rendered.

Complex Data Sets

To understand the general data complexity problem, we must examine it from the perspective of the application developer. If a developer is creating a game, then it is perfectly valid to search for ways to create the imagery while minimizing the amount of data behind it. This approach is served well by techniques such as extensive use of texture maps on a relatively small amount of geometry. However, for an application responsible for producing or analyzing technical data, it is rarely effective to improve the rendering performance by manually altering and reducing the data set. If the data set is huge, the application must be able to make the best of it during 3D rendering. Unfortunately, the problem of exponential growth in data complexity cannot be solved through incremental improvements to the performance of the current 3D graphics architectures—new approaches are required.

Pixels per Polygon

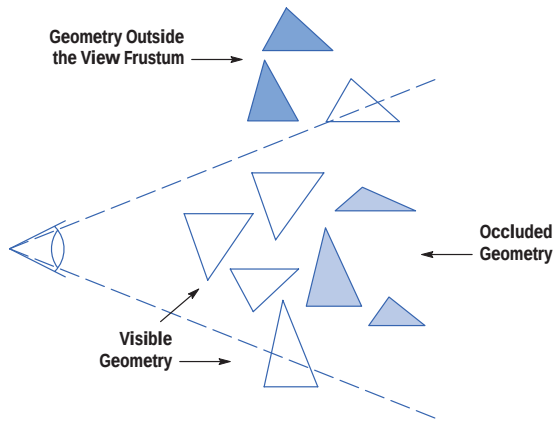
Although the problem of interactively rendering large 3D models on a typical engineering workstation is challenging, it is not intractable. If the workstation's graphics pipeline is capable of rendering a sustained 200,000 polygons per second (a conservative estimate), then each frame must be limited to 20,000 polygons to maintain 10 frames per second. A typical workstation with a 1280 by 1024 monitor provides 1,310,720 pixels. To cover this screen completely with 20,000 polygons, each polygon must have an average area of 66 pixels. A more realistic estimate is that the rendered image covers some subset of the screen, say 75 percent, and that several polygons, for example four, overlap on each pixel, which implies each polygon must cover an area of approximately 200 pixels.

On a typical workstation monitor with a screen resolution of approximately 100 pixels per inch, these polygons are a bit more than 0.1-inch on a side. Polygons of this size will create a high enough quality image for most engineering tasks. This image quality is even more compelling when you consider that it is the resolution produced during interactive navigation. A much higher-quality image can be rendered within a few seconds when the user stops interacting with the model. Thus, today's 3D graphics workstations have enough rendering power to produce the fast, high-quality images required by the technical user.

Software Algorithms

The challenge of interactive large model rendering is sorting through the millions of polygons in the model and choosing (or creating) the best subset of those polygons

Figure 3
Geometry culling.



that can be rendered in the time allowed for the frame. Algorithms that perform this geometry reduction fall into two broad categories: *culling*, which eliminates unnecessary geometry, and *simplification*, which replaces some set of geometry with a simpler version.

Figure 3 illustrates two types of culling: *view frustum* culling (eliminating geometry that is outside of the user's field of view) and *occlusion* culling (eliminating geometry that is hidden behind some other geometry). The article on page 9 describes the implementation of occlusion culling in the VISUALIZE fx graphics accelerator.

Figures 4 and **5** show two types of simplification. **Figure 4** shows a form of geometry simplification called *tessellation*, which takes a mathematical specification of a smooth surface and creates a polygonal representation at the specified level of resolution.

The *decimation* simplification technique is shown in **Figure 5**. This technique reduces the number of polygons in a model by combining adjacent faces and edges.

The simplified geometry created by these algorithms is used by the level of detail selection algorithms, which choose the appropriate representation to render for each frame based on criteria such as the distance to the object.

Most 3D graphics pipelines render a model by rendering each primitive such as a polygon, line, or point individually. If the model contains a million polygons, then the polygon-rendering algorithm is executed a million times. In contrast, these geometry reduction algorithms must operate on the entire 3D model at once, or a significant portion of it, to achieve adequate gains. View frustum culling is a good example—the conventional 3D graphics pipeline will perform this operation on each individual polygon as it is rendered. However, to provide any significant benefit to the large model rendering problem, the culling algorithm must be applied globally to a large chunk of the model so that a significant amount of geometry can be eliminated with a single operation. Similarly, the geometry simplification algorithms can provide greatest gains when they are applied to a large portion of the model.

Desired Solution

The performance gap (often several orders of magnitude) between the needs of the technical user and the capabilities of a typical system puts developers of technical applications into an unfortunate bind. Developers are often experts in some technical domain that is the focus of their applications, perhaps stress analysis or piping layout. However, the 3D data sets that the applications manage are exceeding the graphics performance of the systems

Figure 4
Geometry tessellation.

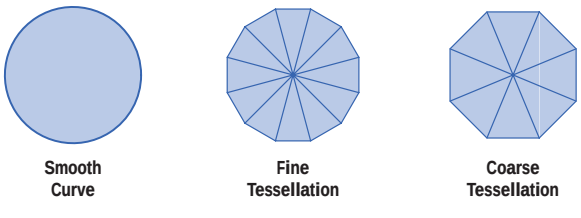


Figure 5
Geometry decimation.

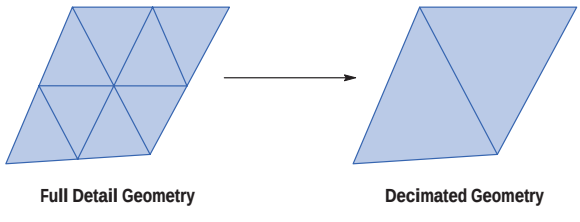
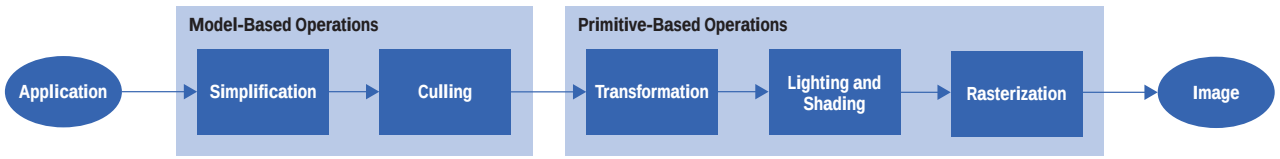


Figure 6

Extended graphics pipeline.



they run on. Developers are faced with the choice of obtaining the 3D graphics expertise to create a sophisticated rendering architecture for their applications, or seeing their applications lag far behind their customers' needs for large 3D modeling capacity and interactivity.

To develop applications with the performance demanded by their customers, developers need access to graphics systems that provide dramatic performance gains for their tasks and data. As shown in **Figure 6**, the graphics pipeline available to the applications must be extended to include model-based optimizations, such as culling and simplification, so that it can support interactive rendering of very large 3D models. When the graphics system provides this level of performance, application developers are free to focus on improving the functionality of their applications without concern about graphics performance. The article on page 9 describes the primitive-based operations of the pipeline shown in **Figure 6**.

DirectModel Capabilities

DirectModel is a toolkit for creating technical 3D graphics applications. The engineer or scientist who must create, visualize, and analyze massive amounts of 3D data does not interact directly with DirectModel. DirectModel provides high-level 3D model management of large 3D geometry models containing millions of polygons. It uses advanced geometry simplification and culling algorithms to support interactive rendering. **Figure 1** shows that DirectModel is implemented on top of traditional 3D graphics APIs such as Starbase or OpenGL. It extends, but does not replace, the current software and hardware 3D rendering pipeline.

Key aspects of the DirectModel toolkit include:

- A Focus on the needs of technical applications that deal with large volumes of 3D geometry data
- Capability for cross-platform support of a wide variety of technical systems
- Extensive support of MDA applications (for example, translators for common MDA data types).

Technical Data

As discussed above, the underlying data is often the most important item to the user of a technical application. For example, when designers select parts on the screen and ask for dimensions, they want to know the precise engineering dimension, not some inexact dimension that results when the data is passed through the graphics system for rendering. DirectModel provides the interfaces that allow the application to specify and query data with this level of technical precision.

Technical data often contains far more than graphical information. In fact, the metadata such as who created the model, what it is related to, and the results of analyzing it is often much larger than the graphical data that is rendered. Consequently DirectModel provides the interfaces that allow an application to create the links between the graphical data and the vast amount of related metadata.

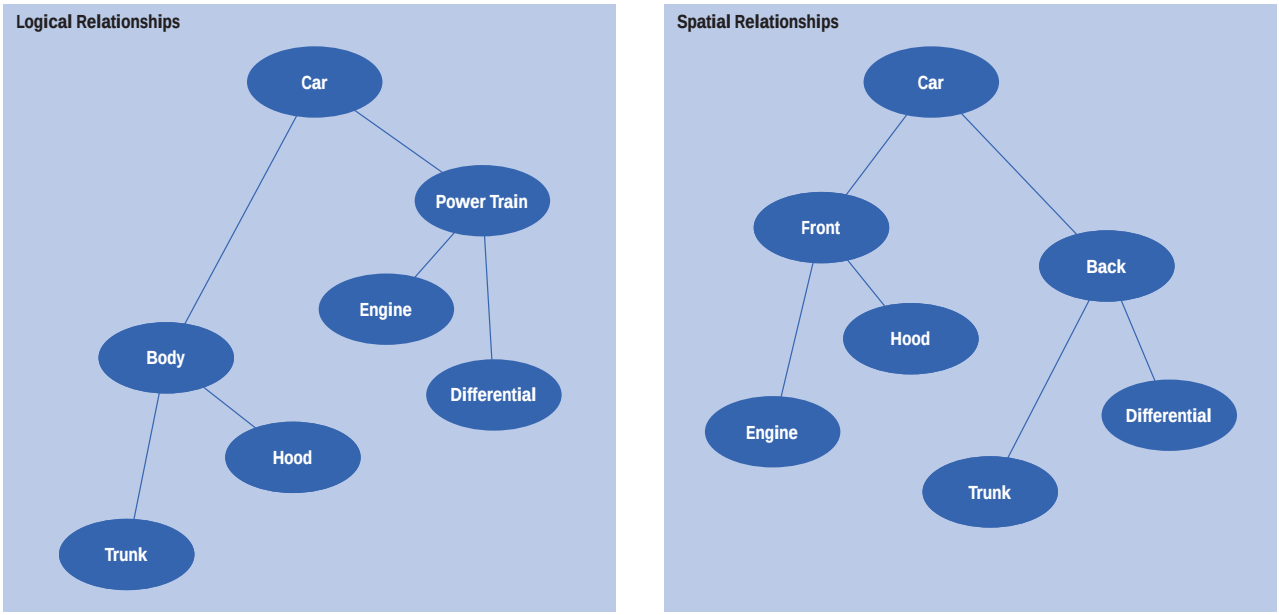
Components of large models are often created, owned, and managed by people or organizations that are loosely connected. For example, one design group might be responsible for the fuselage of an airplane while a separate group is responsible for the design of the engines. DirectModel supports this multiteam collaboration by allowing a 3D model to be assembled from several smaller 3D models that have been independently defined and optimized.

Multiple Representations of the Model

The 3D model is the central concept of DirectModel—the application defines the model and DirectModel is responsible for high-performance optimization and rendering of

Figure 7

Logical and spatial organization.



it. The 3D model is defined hierarchically by the model graph, which consists of a set of nodes linked together into a directed, acyclic graph. However, a common problem that occurs when creating a model graph is the conflict between the needs of the application needs and the graphics system. The application typically needs to organize the model based on the logical relationships between the components, whereas the graphics system needs to organize the model based on the spatial relationships so that it can be efficiently simplified, culled, and rendered. **Figure 7** shows two model graphs for a car, one organized logically and one spatially.

Graphics toolkits that use a single model graph for both the application's interaction with the model and for rendering the model force the application developer to optimize for one use while making the other use difficult. In contrast, DirectModel maintains multiple organizations of the model so that it can simultaneously be optimized for several different uses. The application is free to organize its model graph based on its functional requirements without consideration of DirectModel's rendering needs. DirectModel will create and maintain an additional spatial organization that is optimized for rendering. These multiple organizations do not significantly increase the memory or disk usage of DirectModel because the actual geometry, by far the largest component, is multiply referenced, not duplicated.

The Problem of Motion

Object motion, both predefined and interactive, is critical to many technical applications. In mechanical design, for example, users want to see suspension systems moving, engines rocking, and pistons and valves in motion. To use a virtual prototype for manufacturing planning, motion is mandatory. Assembly sequences can be verified only by observing the motion of each component as it moves into place along its prescribed path. Users also want to grab an object or subassembly and move it through space, while bumping and jostling the object as it interferes with other objects in its path. In short, motion is an essential component for creating the level of realism necessary for full use of digital prototypes.

DirectModel supports this demand for adding motion to 3D models in several ways. Because DirectModel does not force an application to create a model graph that is optimized for fast rendering, it can instead create one that is optimized for managing motion. Parts that are physically connected in real life can be connected in the model graph,

allowing movement to cascade easily through all of the affected parts. In addition, the data structures and algorithms used by DirectModel to optimize the model graph for rendering are designed for easy incremental update when some portion of the application's model graph changes.

Models as Databases

3D models containing millions of polygons with a rich set of rendering attributes and metadata can easily require several gigabytes of data. Models of this size are frequently too big to be completely held in main memory, which makes it particularly challenging to support smooth interactivity.

DirectModel solves this problem by treating the model as a database that is held on disk and incrementally brought in and out of main memory as necessary. Elements of the model, including individual level-of-detail representations, must come from disk as they are needed and removed from main memory when they are not needed. In this way memory can be reserved for the geometric representations currently of interest. DirectModel's large model capability has as much to do with rapid and intelligent database interaction as with rendering optimization.

Interactive versus Batch-Mode Data Preparation

Applications that deal with large 3D models have a wide range of capabilities. One application may be simply an interactive viewer of large models that are assembled from existing data. Another application may be a 3D editor (for example, a solid modeler) that supports designing mechanical parts within the context of their full assembly. Consequently, an application may acquire and optimize a large amount of 3D geometry all at once, or the parts of the model may be created little by little.

DirectModel supports both of these scenarios by allowing model creation and optimization to occur either interactively or in batch mode. If an application has a great deal of raw geometry that must be rendered, it will typically choose to provide a batch-mode preprocessor that builds the model graph, invokes the sorting and simplification algorithms, and then saves the results. An interactive application can then load the optimized data and immediately allow the user to navigate through the data. However, if the application is creating or modifying the elements of the model at a slow rate, then it is reasonable to sort and optimize the data in real time. Hybrid scenarios are also possible where an interactive application performs incremental optimization of the model with any spare CPU cycles that are available.

The important thing to note in these scenarios is that DirectModel does not make a strong distinction between batch and interactive operations. All operations can be considered interactive and the application developer is free to employ them in a batch manner when appropriate.

Extensibility

Large 3D models used by technical applications have different characteristics. Some models are highly regular with geometry laid out on a fixed grid (for example, rectangular buildings with rectangular rooms) whereas others are highly irregular (for example, an automobile engine with curved parts located at many different orientations). Some models have a high degree of occlusion where entire parts or assemblies are hidden from many viewing perspectives. Other models have more holes through them allowing glimpses of otherwise hidden parts. Some models are spatially dense with many components packed into a tight space, whereas others are sparse with sizable gaps between the parts.

These vast differences impact the choice of effective optimization and rendering algorithms. For example, highly regular models such as buildings are amenable to preprocessing to determine regions of visibility (for example, rooms A through E are not visible from any point in room Z). However, this type of preprocessing is not very effective when applied to irregular models such as an engine. In addition, large model visualization is a vibrant field of research with innovative new algorithms appearing regularly. The algorithms that seem optimal today may appear very limiting tomorrow.

DirectModel's flexible architecture allows application developers to choose the right combination of techniques, including creating new algorithms to extend the system's capabilities. All of the DirectModel functions, such as its culling algorithms, representation generators, tessellators, and picking operators, are extensible in this way. Extensions fit seamlessly into the algorithms they extend, indistinguishable from the default capabilities inherent to the toolkit.

In addition, DirectModel supports mixed-mode rendering in which an application uses DirectModel for some of its rendering needs and calls the underlying core graphics

API directly for other rendering operations. Although DirectModel can fulfill the complete graphics needs of many applications, it does not require that it be used exclusively.

Multiplatform Support

A variety of systems are commonly used for today's technical 3D graphics applications, ranging from high-end personal computers through various UNIX-based workstations and supercomputers. In addition, several 3D graphics APIs and architectures are either established or emerging as appropriate foundations for technical applications. Most developers of technical applications support a variety of existing systems and must be able to migrate their applications onto new hardware architectures as the market evolves.

DirectModel has been carefully designed and implemented for optimum rendering performance on multiple platforms and operating systems. It presumes no particular graphics API and is designed to select at run time the graphics API best suited to the platform or specified by the application. In addition, its core rendering algorithms dynamically adapt themselves to the performance requirements of the underlying graphics pipeline.

Conclusion

The increasing use of 3D graphics as a powerful tool for solving technical problems has led to an explosion in the complexity of problems being addressed, resulting in 3D models containing millions or even billions of polygons. Unfortunately, many of the applications and 3D graphics systems in use today are built on architectures designed to handle only a few thousands polygons efficiently. These architectures are incapable of providing interactivity with today's large technical data sets.

This problem has created a strong demand for new graphics architectures and products that are designed for interactive rendering of large models on affordable systems. Hewlett-Packard is meeting this demand with DirectModel, a cross-platform toolkit that enables interaction with large, complex, 3D models.

References

1. Data obtained from design engineers at the Hewlett-Packard Vancouver Division.

2. Estimates provided by automotive design engineers.

3. S.H. Shokralla, "The 21st Century Jet: The Boeing 777 Multimedia Case Study," http://pawn.berkely.edu/~shad/b777/main.html

4. E. Brechner, "Interactive Walkthrough of Large Geometric Databases," *SIGGRAPH tutorial*, 1995.

5. J.S. Lombardo, E. Mihalak, and S.R. Osborne, "Collaborative Virtual Prototype, *John Hopkins APL Technical Digest*, Vol. 17, no. 3, 1996.

▶ Go to Next Article

▶ Go to Journal Home Page