

Virtual Reality in Scientific Visualization

Steve Bryson

Immersing the user in the solution, virtual reality reveals the spatially complex structures in computational science in a way that makes them easy to understand and study. But beyond adding a 3D interface, virtual reality also means greater computational complexity.

VIRTUAL reality, also called virtual environments, is a new interface paradigm that uses computers and human-computer interfaces to create the effect of a three-dimensional world in which the user interacts directly with virtual objects. The term “virtual reality” has received quite a lot of attention in the last few years, so we feel it is important to be clear about its meaning. For

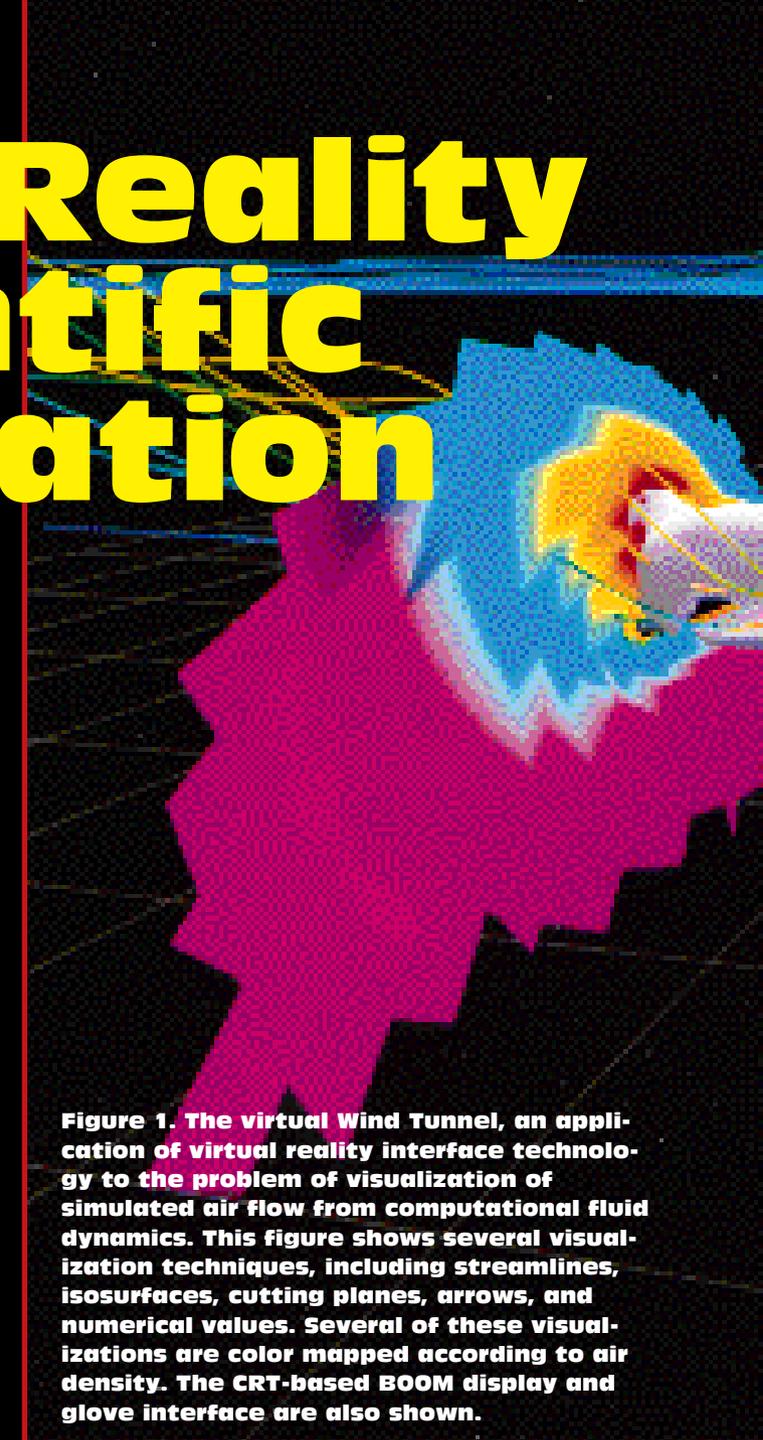
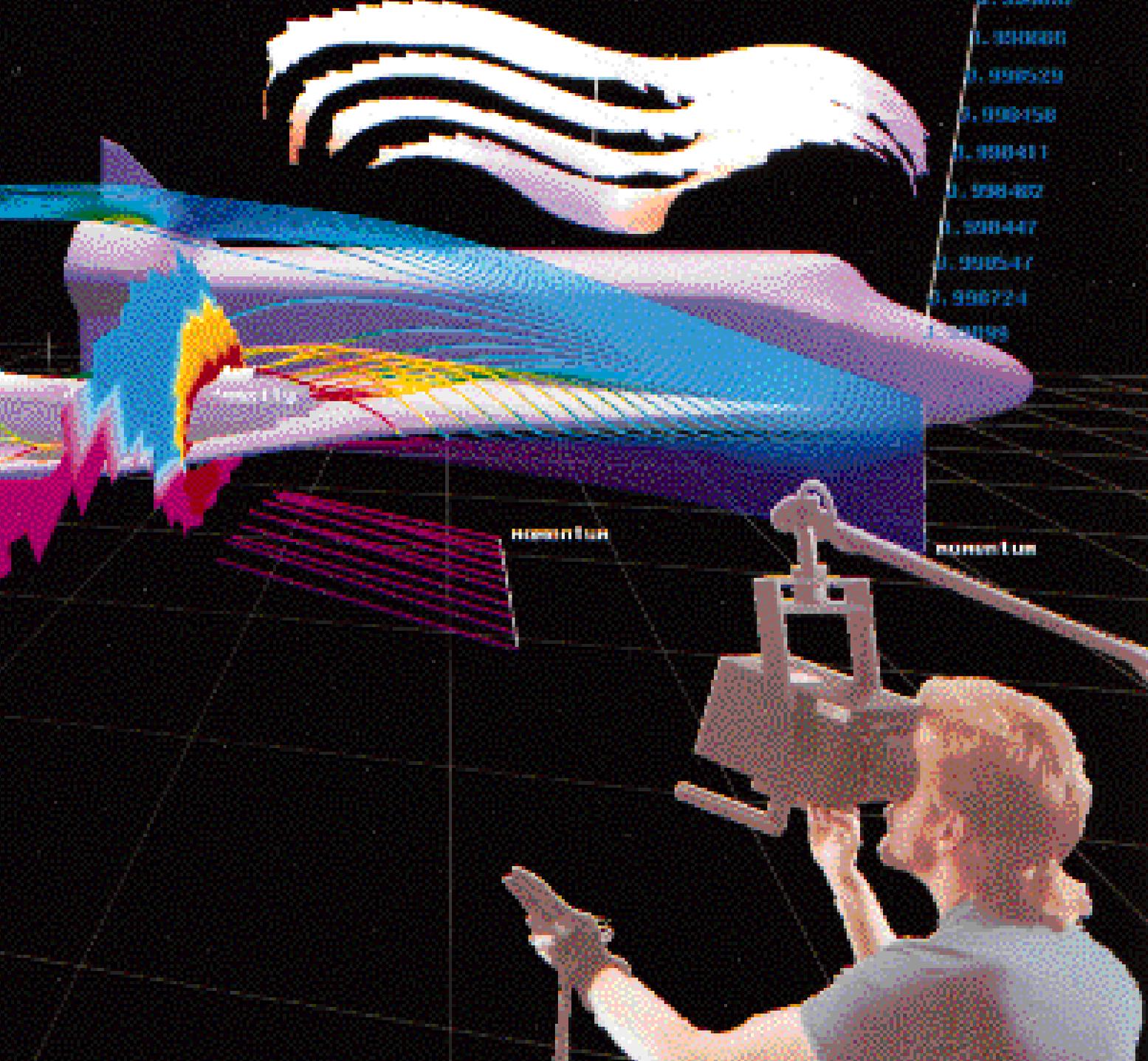


Figure 1. The virtual Wind Tunnel, an application of virtual reality interface technology to the problem of visualization of simulated air flow from computational fluid dynamics. This figure shows several visualization techniques, including streamlines, isosurfaces, cutting planes, arrows, and numerical values. Several of these visualizations are color mapped according to air density. The CRT-based BOOM display and glove interface are also shown.

the purposes of this article, we adopt the following definition:

Virtual reality is the use of computers and human-computer interfaces to create the effect of a three-dimensional world containing interactive objects with a strong sense of three-dimensional presence.



Important in this definition is that virtual reality is computer-generated, three-dimensional, and interactive. We want to create the effect of interacting with things, not with pictures of things. Note that we claim that virtual reality is an effect, not an illusion. In partic-

ular, virtual reality does not, a priori, attempt to create an illusion of the real world (although some applications do attempt to do so). Note also that it is the interface, not the content, that characterizes virtual reality [5].

The effect of virtual reality is

attained through several components:

- A head-tracked, usually stereoscopic, display that presents the virtual world from the user's current head position, including the visual cues required so the virtual scene is perceived as independent of the user, that is, has "object constancy," while the user moves about;
- A high-performance computer graphics system that computes and renders the virtual world; and
- Three-dimensional input devices that allow the user to provide input into the system directly in three dimensions.

Virtual reality offers the possibility of custom-designed, three-dimensional environments allowing intuitive "transparent" interfaces—in the sense that the computer interface is not visible to the user. Further, the three-dimensional display and interaction capabilities of virtual reality allow for significantly enhanced three-dimensional perception and interaction over conventional three-dimensional computer graphics. It is reasonable to expect that these enhancements would be useful for a variety of applications. But, somewhat surprisingly, virtual reality applications have proven difficult to develop. We feel there are two primary difficulties impeding development of virtual reality applications.

The first is that the virtual reality interface is a completely new paradigm to which two-dimensional interface paradigms do not easily apply. This difficulty has forced virtual reality application developers to reinvent the human-computer interface all over again—a difficult task. Although this research area is promising, we do not discuss it further here.

The second, which is the focus of this article, is that to attain the virtual reality effect, the system must deliver very high performance. For example, the graphical images must be updated quickly enough so object and self motion are perceived as motion while the results of user actions have low latency. Specifically, we have what we call virtual reality, or real-time performance, requirements:

- The system must provide feedback to the user regarding the effect of continuous user input in less than 0.1 second, allowing fast, accurate manipulation of the environment [15].
- The animation must have a frame rate—particularly the graphics animation rate—of at least 10 frames per second. This rate is required for the head-tracked display to supply the required sense of three-dimensional presence. Note, too, that the requirement is not that the motion appear "smooth"; at 10 frames per second, the frame rate is clearly visible.
- The virtual environment must contain application objects of a sufficient level of fidelity to allow the performance of meaningful tasks.

These requirements often conflict.

Images for Science

SCIENTIFIC visualization [14] is the use of computer graphics to create visual images that aid in the understanding of complex (often massive) numerical representations of scientific concepts or results. Such numerical representations, or data sets, may be the output of numerical simulations, as in computational fluid dynamics (CFD) or molecular modeling; recorded data, as in geological or astronomical applications; or constructed shapes, as in visualization of topological arguments. These simulations often contain high-dimensional data in a three-dimensional volume. The display of phenomena associated with this data may involve complex three-dimensional structures. Non-immersive interactive visualization systems implemented for the conventional desktop and mouse [2, 12] are effective for moderately complex problems. Virtual reality displays aid in the unambiguous display of these structures by providing a rich set of spatial and depth cues. Virtual reality interface concepts allow the rapid and intuitive exploration of the volume containing the data, enabling the phenomena at various places in the volume to be explored, as well as provide simple control of the visualization environment through interfaces integrated into the environment.

Evidence for the naturalness of applying virtual reality to scientific visualization is the relatively rich set of applications developed in spite of the immature state of virtual reality technology. Such applications include the Virtual Wind Tunnel (see Figure 1) [7, 8] and Virtual Spacetime [6], molecular modeling [3], scanning tunnelling microscope display and control [18], and medical visualization systems [1] developed at the University of North Carolina at Chapel Hill. A system to investigate cosmic structure formation has been implemented at the National Center for Supercomputing Applications [17]. The Electronic Visualization Laboratory at the University of Illinois, Chicago, has implemented several scientific visualization applications [9, 10]. In addition, there was a large number of exhibits at the VROOM event, a showcase of virtual reality applications for scientific visualization that was part of SIGGRAPH '94.

Virtual reality and scientific visualization are well matched for several reasons, in addition to inherently three-dimensional display and control. Scientific visualization is oriented toward the informative display of abstract quantities and concepts, as opposed to an attempt to realistically represent objects in the real world. Thus, the graphics demands of scientific visualization can be oriented toward accurate, as opposed to realistic, representations. Some graphical representations are feasible with current technology. Further, as the phenomena being represented are abstract, a researcher can perform investigations in virtual reality impossible or meaningless in the real world. In this way, the best aspects of computer simulation and intuitive real-world interaction are combined. Finally, scientific visualization encompasses a wide set of problems, some feasible with current tech-

nology, some not feasible. Virtual reality application developers can identify scientific visualization problems that can immediately benefit from a virtual reality interface with the current state of the technology. Such applications demonstrate the effectiveness of virtual reality technology while providing valuable scientific visualization tools.

In many ways, the major impact of virtual reality technology on scientific visualization is in providing a “real-time” intuitive interface for exploring data while facilitating the use of scientific visualization in the research process. We offer two points: that “real-time” exploration is a desirable capability with special demands and requirements, and that virtual reality interfaces greatly facilitate exploration capabilities.

A system for supporting interactive exploration of complex data sets needs two capabilities:

- Real-time interaction
- A natural, “anthropomorphic” three-dimensional interface

that can occur in a three-dimensional volume. It is also in these latter cases, however, that real-time interactive visualization is most difficult—due to the size of the data sets and the complexity of the computations involved in the visualization process.

An instructive example is the visualization in CFD. Data sets resulting from CFD computations typically provide five parameters—the three components of velocity, as well as pressure and energy—at every point and typically are very large, with millions of points per timestep for hundreds or thousands of timesteps. These data sets are rapidly growing in size due to the increase in available computational power. The CFD visualization problem is further complicated by the fact the data is provided either on multiple overlapping stretched computational grids or on unstructured grids. In both cases, position data for each point must be provided explicitly, further increasing the data size, particularly for simulations in which the computational grid points move in time. For example, data sets exceeding 200 gigabytes are being generated at NASA’s Ames

We want to create the effect of interacting with things, not with pictures of things.

These capabilities provide three main advantages:

- The ability to quickly sample a data set’s volume without cluttering the visualization
- The ability to investigate regions not expected to be of interest without penalty
- The ability to see the relationship between data nearby in space or time without cluttering up the visualization

These advantages can be summed up as: Real-time interaction encourages exploration.

Demands of Real-Time Visualization

MANY visualization applications involve two-dimensional data sets containing only a single value at every point. For such problems, a single well-designed picture can provide a great deal of insight into that data. However, for many problems, particularly modern engineering simulations, the data sets consist of a time series of three-dimensional volumes with many parameters at each point. Further, users are often interested in behavior induced by that data, such as streamlines in a vector field, rather than in the data values themselves. It is in such cases we expect real-time interactive visualization to pay off—due to the complexity of the phenomena

Research Center, and it is anticipated that terabyte CFD data sets are not far in the future. The computations involved in CFD visualization are computationally intensive. Widely used visualization techniques include particle advection techniques, such as streamlines and particle paths, and scalar isosurface techniques that involve marching through an entire timestep of data to determine the location of the isosurface.

To focus the discussion, consider a modest modern data set—that of a harrier jump-jet aircraft hovering above the ground [16]. This data set contains in each timestep five floating-point numbers at 2.8 million points distributed among 18 overlapping computational grids. There are 106 timesteps. The total data set size is 5.6 gigabytes, which breaks down to 56 megabytes per timestep, not including the single 42-megabyte data set that provides the positions of the grid points and other necessary geometry data.

To illustrate the computational requirements of a visualization system, consider streamlines, a technique often involving several hundred integrations in the computation of a single streamline. Each integration should use a technique with a high order of accuracy, increasing the computational burden. Further, several hundred streamlines are often desirable. The result is that streamlines can be very computationally expensive. To estimate more precisely the computa-

tional demands of a streamline computation, consider the second-order Runge-Kutta integration technique. When highly optimized by the performance of calculations in computational grid coordinates so the position directly provides the grid coordinates (assuming the initial position is provided in computational grid coordinates), a single integration of a streamline requires about 200 floating-point operations and 24 accesses to the data set. A typical modern high-performance workstation has peak scalar performance ratings ranging from 10 megaflops to 100 megaflops, with perhaps 20 megaflops being typical. Such systems are capable of computing 2 million

one streamline. Thus, both bandwidth and seek time imply a severe limit on the ability to do real-time computations from disk. External memory, on so-called solid-state disk, has better bandwidth behavior but still suffers from unacceptably long seek times.

Precomputation or data-buffering strategies do not address these problems in an interactive real-time scientific visualization application, as there is no a priori information about what kind of visualization the researcher will be using and what data will be accessed.

The obvious solution is to place the data in physical memory. Systems that accommodate multi-giga-

Massively parallel architectures that use distributed memory add another impediment to performance, as the interprocessor communication required for each memory access can dramatically slow the system.

floating-point operations in 0.1 seconds, implying the ability to perform just under 10,000 integrations. If each streamline involves 200 integrations, the system can compute at most 50 streamlines in 0.1 seconds. In actual use, most systems perform at about half their rated performance, implying that one could expect to compute about 25 streamlines. Similar analyses can be performed for the other visualization techniques.

The situation with respect to data management is similarly demanding. While the amount of data required in the preceding example computation is modest—three trilinear interpolations each, requiring eight vectors with three components of four bytes each in IEEE 32-bit representation, or 288 bytes—users wish to perform many of these computations. Moreover, the 288 bytes are scattered across the data set in 24 separate places, assuming again that the components of each vector are stored contiguously.

Now consider data stored on disk. Assuming the data on the disk can be accessed at three megabytes/second, bandwidth considerations alone impose a limit of about 1,000 integrations in 0.1 seconds. Further, as the data is scattered across the disk, the disk must locate the data, a process called “seek time,” which requires more time. A typical workstation disk can perform about 23,000 seeks/second, implying that only about 95 integrations can be performed off-disk in 0.1 seconds based on seek time considerations alone—insufficient to compute even

byte data sets are rare, however, and tend to be multiple-user systems. Multi-gigabyte workstations are already available, and the size of data sets is expected to grow dramatically.

Finally, when the data set is rich and complex, it is often desirable to have many visualizations operating simultaneously. Thus, graphical performance becomes an issue, particularly when the visualizations include objects like isosurfaces containing tens or hundreds of thousands of polygons. Similar problems are encountered in the other visualization techniques.

How to Meet Real-Time Performance Demands

EVERYONE knows that faster visualization systems are desirable. Until recently, however, speed and performance were two of several desirable values—along with accuracy, ease of use, access to very large data sets, and elegance of program structure. The point here is to elevate real-time performance to the status of a requirement for interactive visualization systems.

To some extent, advances in technology will provide relief for the difficulties discussed earlier. Processors and disks will be faster, and we will see workstations with more and more memory. It is clear, however, that the response to higher performance will be to make greater demands on the system, as we wish the frame rate to be as high as possible and wish to have many visualization techniques in a single envi-

ronment. Disk access times have to improve by two orders of magnitude to open up the current computational bottleneck. By the time this occurs, the computational performance will have improved by as much as two more orders of magnitude with the availability of 300-megaflops processors. Available memory is also taking an order-of-magnitude leap, with 16-gigabyte systems expected soon.

However, the problems addressed in real-time scientific visualization will far outstrip the advances in technology. Increased computational capability will be used to perform simulations that produce far larger data sets and contain far more phenomena. This increase will occur in several ways. For example, spatial and temporal resolution will increase for enhanced accuracy, increasing the number of points and timesteps. And multidisciplinary simulations will be performed, including many phenomena in a single data set. An example currently under development is to include chemistry and airframe structure dynamics and data into CFD simulations. Thus, the size of data sets will increase dramatically over the next few years. The complexity of the simulations will place further demands on the visualization system; this implies greater computational demand. Yet it is exactly the additional complexity of these anticipated data sets that makes real-time interactive visualization capability so desirable.

There are no simple, general rules for meeting the performance requirements applicable across visualization applications. Sometimes precomputation of derived quantities is required for computational speed, and sometimes computation on the fly is forced by data management considerations. We consider several of the related issues in the following sections.

Computation

The computational demands of a real-time interactive visualization system are determined by the particular visualization techniques used in that system. A visualization based on simply color-mapping an existing geometry (e.g., a face of the computational grid) requires little computation. A technique based on particle integration involves the computational demands described in the section on real-time visualization. Isosurface computation involves computational demands that depend on the number of points in the data set and thus on the size of the data set. Even something as seemingly simple as a color-mapped cutting plane has some computational demand, particularly for stretched curvilinear grids, as the spacing of vertices of the cutting plane must match the nonlinear spacing of the data points, and the values of the data at those vertices must be obtained through trilinear interpolation.

In spite of the dependence of the computational demands on the specific visualization techniques implemented, some general observations can be made. The following paragraphs offer representative examples and are not intended to be exhaustive.

Meeting computational performance demands involves several considerations:

Computational architecture: scalar vs. vector vs. parallel. Scalar architectures are the most versatile, as they make few assumptions about the types of computations being performed, although they are also the slowest architectures. Vector architectures assume that many sequential steps are taken in each computation, all aspects of which can be loaded into a computational pipeline on the processor itself. While true in the abstract for such iterative computations as streamlines, the pipeline is broken when memory accesses and subroutine calls are required in the computation. These memory accesses and subroutine calls are typical in general visualization environments. Finally, massively parallel architectures rely on large numbers of relatively weak scalar processors and give very high performance when a large number of relatively small independent computations are required. Such parallelization is often not the case in visualization, as many visualization techniques require iterative computations. Massively parallel architectures that use distributed memory add another impediment to performance, as the inter-processor communication required for each memory access can dramatically slow the system.

Computational algorithm. There are typically many ways to perform a computation. Faster computational algorithms are generally less accurate, implying a tradeoff between accuracy and performance. In an exploratory environment, it is often desirable to have a fast, less-accurate result. Examples may include second-order rather than fourth-order approximations, subsampling of data, and so on. Of course, if the answer is too inaccurate, it is not useful, so great care must be taken when choosing this tradeoff. This choice is further complicated by the fact that the appropriate tradeoff is typically context-dependent, in that the parts of a visualization that need high accuracy depend on what the user is doing. A simple approach to this problem is to allow the user to directly determine the appropriate tradeoffs at runtime. A more attractive approach is to automate these choices, using time-critical computational algorithms that attempt to automatically adjust the tradeoffs in ways appropriate to the current computational situation. The design of time-critical computational algorithms appropriate in scientific visualization is very difficult and complex. Appropriate metrics for the importance of a visualization must be developed—an undertaking today in its infancy.

Choice of data representation to facilitate computation. Can interesting aspects of the data be precomputed? A CFD example is the visualization of vorticity given by the curl of the velocity vector field. The curl of a vector field is somewhat time-consuming to compute, so significant computational efficiency can be gained by computing the vorticity off-line and load-

ing the result into the visualization system. This approach may conflict with some of the data management issues described in the next section. Another example of a data representation that facilitates computation is the use of grid-coordinate-based representations of vector fields; the components of vector fields on highly stretched curvilinear grids may be given in terms of the grid components. In such representations, iterative computations, such as streamlines, are much faster, as they do not require time-consuming search routines to determine the grid coordinates of the next computation.

Careful code optimization. The current trend in workstation architectures toward several very fast scalar processors running in parallel and sharing the same physical memory space provides a very good computational architecture for high-performance visualization. Presumably the near future will see this architecture scaled up to massively parallel processor range (hundreds to thousands of processors). While it remains to be seen if performance actually scales in such a system, we have reason to be optimistic.

Data Management

As described in the section on real-time visualization, data management problems in modern scientific visualization can be severe. Given the current data-access overhead encountered in mass storage systems, the only viable data management strategy seems to be to have all data accessed in the course of the visualization session stored in physical memory. This approach implies that real-time interactive visualization systems rely on hardware platforms with large amounts of physical memory. Happily, recently introduced graphics workstations support as much as 16 gigabytes of physical memory. Such systems are quite expensive in the short term, however, and interesting data sets can be larger than this and are growing rapidly. Thus, large physical memory alone is not sufficient to solve all data management problems. Users can implement a number of strategies:

Load only a subvolume of the data. This strategy attempts to limit the visualization problem in volume, allowing more timesteps of a smaller volume to be loaded into physical memory. The volume can be specified by the user at startup time. A more ambitious scheme is to load volumes of data when required by a computation. However, the latter scheme encounters great difficulties when there are several different kinds of visualization present in the environment, all computing in parallel.

Subsample the data. The large data set can be subsampled in time as well as in space. This approach is extremely risky, however, as subsampling may not only mask phenomena but also introduce incorrect behavior in, for example, streamlines. Large numerical simulations are typically done on grids with the minimum required number of points, so

further decreasing the number of points is not recommended.

Compress the data. There are various schemes for compressing data. Unfortunately, most are “lossy,” in that data cannot be reconstructed with complete accuracy after compression. Multiple-scale representations, such as wavelets, are somewhat more promising for the compression of data, but good results have yet to be delivered. Finally, the decompression of the data places a burden on the computation, possibly conflicting with the system’s other computational requirements.

Optimize the organization of the data on disk. The data can be placed on disk so that everything required for a computation may be loaded at run time in a single load. For example, for an isosurface of a scalar field, the field values may be ordered on disk by the data field value so only the data with values near the isosurface field value are loaded. The difficulty with this method is that data storage is optimized for a particular visualization technique; for another technique, optimization would be different.

Use widely striped disk arrays. Systems with large numbers of small disks can in theory attain bandwidths up to 500 megabytes/second, allowing the possibility of loading individual timesteps from disk as long as the individual timestep is smaller than 50 megabytes. Handling such a set of disks is a significant computational burden, and it is not clear whether it is possible to maintain such high bandwidth while performing visualization tasks. We are now attempting to develop such a system for the Virtual Wind Tunnel at NASA’s Ames Research Center.

Another aspect of data management is the use of networks for shared and distributed environments. The issues encountered for networks are essentially the same as those encountered for disks—bandwidth and latency. There are also significant software issues in the design of networked environments, but they are beyond the scope of this article.

Graphics

Luckily for many scientific visualization applications, the graphical rendering is to some extent arbitrary and can be chosen specifically to satisfy the performance constraints. This situation contrasts with the high-fidelity visual simulation of the real world, which can require time-consuming photo-realistic rendering techniques. Consider again the example of streamlines in CFD. A streamline is simply an array of points in three dimensions. The array can be simply rendered as lines, allowing very fast rendering.

This example can be generalized into a principle for developing real-time interactive visualization systems: Keep the graphical rendering as simple as possible. Use points instead of spheres, use lines instead of tubes, and so on. While it is true that, for example, using lighted tubes rather than lines provides more

depth cues to show three-dimensional structure, the ability to provide real-time interaction can also provide strong depth cues, either through allowing interactive rotations or through the use of head-tracked rendering. Also remember that while a technique may not be time consuming when done once, a user may desire many instances of this technique in an environment.

There are, of course, visualization techniques that demand considerable graphics performance, such as isosurfaces and volumetric rendering. In a large data set, a typical isosurface may contain tens of thousands of (logically) disconnected polygons. These polygons are difficult to render within real-time performance constraints. Ways of meeting this problem for isosurfaces include subsampling the surface (potentially masking interesting features), rearranging the poly-

Virtual reality interfaces attempt to provide the most anthropomorphic interfaces possible. Virtual reality interfaces must include two components: display and user control. We address them separately, discussing the concepts involved as well as the difficulties encountered in their use.

3D Display

Scientific visualization makes particular demands on virtual reality displays. The phenomena to be displayed in a scientific visualization application often involve delicate and detailed structure, requiring high-quality, high-resolution full-color displays. Experience has shown that displays with 1,000-by-1,000-pixel resolution are sufficient for many applications. In addition, a wide field of view is often desirable, because it allows the researcher to view how detailed

A principle for developing real-time interactive visualization systems: Keep the graphical rendering as simple as possible.

gons so they are optimized for the graphics hardware, hashing the polygon list to replace coplanar neighboring polygons with a single polygon, and limiting the spatial extent of the isosurface. Many or all of these techniques may be necessary to achieve real-time performance demands, and even then a limited number of isosurfaces may be practical. Time-intensive methods, such as ray-traced or volumetric rendering, may be impractical in real-time applications.

Virtual Reality Interfaces

OUR discussion has, up to this point, primarily addressed the issue of real-time performance. However, in the section on scientific visualization, we suggested that real-time performance is only one of two requirements of a scientific exploration environment; the other requirement is a natural, inherently three-dimensional, human-conforming interface. By this we mean an interface that is used in as natural a way as possible, that provides as unambiguous a three-dimensional display as possible, and that requires as little as possible of the user's attention. This approach contrasts with the current interaction paradigm in scientific visualization based on text or two-dimensional input through graphical user interfaces and two-dimensional projections of three-dimensional scenes. Conventional interfaces make it difficult to specify positions in three dimensions and do not provide unambiguous display of three-dimensional structure.

structures are related to larger, more global phenomena.

The combined demands of wide field of view and high display quality and resolution are, as of this writing, extremely difficult to put into a head-mounted display. The alternatives of the CRT-based Binocular Omni-Oriented Monitor (BOOM) [5] displays and the projection-screen-based Cave Automatic Virtual Environment (CAVE)-type displays [10] are proving popular for many scientific visualization applications. Both types of displays are, however, high in cost and have several disadvantages. Head-tracked stereoscopic displays using conventional workstation monitors [11] are effective for visualizations that do not involve immersion.

Another factor to be considered in selecting virtual reality visual displays for scientific visualization is the issue of user acceptance. Many researchers have expressed distaste for donning helmets or strapping displays onto their heads. Both the BOOM-type and the CAVE-type displays avoid many of these problems.

Interaction

The main point of this article is that virtual environment interfaces can enhance the role of scientific visualization in the scientific discovery process. Such a system clearly requires interactive capabilities that allow intuitive control of the data visualization displays, avoiding as much as possible difficulties due to the arbitrariness of the interface. The most obvious

need is for the ability to rapidly and precisely select a location in three dimensions. There are three components to this ability:

- The ability to specify a location
- The ability to specify an action at that location
- The ability to provide feedback as to the location selected, typically by the three-dimensional display

Here we focus on the specification of the location and the action to occur at that location.

The most intuitive method of specifying location is by tracking the user's hand position in three dimensions through one of a variety of available tracking technologies [4]. However, all of these technologies suffer from various problems, including cost and lack of accuracy, although the advantages seem to significantly outweigh the disadvantages.

The specification of the action to occur at the

state of objects, lack an obvious intuitive interface. Methods of addressing these issues include the extension of such conventional graphical user interface techniques as menus and sliders to objects in the three-dimensional visualization environment. Much of the problem is a lack of understanding of the human factors involved in the design of these types of environments. This is clearly an area that needs more research and experimentation.

Sound

The use of three-dimensional sound in scientific visualization is relatively unexplored but holds promise. Aside from the conventional uses of sound to provide user feedback as to the state of the environment, sound can be used as an additional data display channel. Scalar quantities can be mapped to the frequency, timbre, or amplitude of a sound.

The problems with glove devices include inaccuracies in measurement and lack of a standard gestural vocabulary.

selected location can be accomplished with conventional hand-held button devices (or key strokes). Buttons have the advantage of being unambiguous and easy to learn. They have the disadvantage that they must be held in the hand and provide an arbitrary method of indicating commands. Particularly in immersive environments, where users cannot see their real hands, there may be fatigue problem with the long-term use of hand-held button devices. An alternative is to use a glove device, which tracks the angle of bend of the user's fingers. Using this device, intuitive gestures, such as "fist" and "point," can be used to provide intuitive control of the environment. Examples include using the "fist" gesture to grab and move objects and using the "point" gesture to indicate an object. The problems with glove devices include inaccuracies in measurement and the lack of a standard gestural vocabulary. The use of gloves must be implemented with great care to obtain good results, inaccuracies of measurement must be calibrated out, the glove should be used with low arm positions to avoid fatigue, and appropriate visual feedback for gesture recognition must be provided. It remains to be seen if users prefer gloves or button devices.

The use of natural, three-dimensional interaction raises the issue of how that interaction is used in the environment. Some tasks, such as grabbing and moving a visualization object, are obvious. Other tasks, however, such as selecting or changing

Haptics

Haptics, the senses of touch and force, have been applied to scientific visualization in virtual reality contexts in a few laboratory systems [3, 7, 13]. The effectiveness of haptics in these research projects indicates that haptics may prove to be a useful data display channel. The immaturity of haptic technology, however, makes it very difficult to implement a haptic display in a general, easily reproducible visualization system.

Future Directions

SCIENTIFIC visualization is potentially a very fruitful application area for virtual reality and should be pursued aggressively. Unlike the ease of some application areas, significant scientific-visualization applications can be developed with existing technology. Critical to this capability are high-performance three-dimensional graphics workstations with multiple scalar processors having total floating-point power of hundreds of megaflops; operating systems that support regularly scheduled preemptive lightweight threads, so that processes are reliably and regularly scheduled; and very large (more than 10 gigabytes) physical memory capability. The trend in workstation design toward faster graphics, higher computational power, operating systems supporting concurrent execution, and very large memory should be encouraged and accelerated.

While many significant applications of virtual reality to scientific visualization may be implemented with existing technology, the current state of the technology imposes significant limits. Eliminating these limits depends on work in the following areas:

Data management. High-bandwidth, low-latency mass storage systems are required to handle modern visualization problems containing from many gigabytes to terabytes of data in a virtual reality application.

Rapid prototyping. Systems should provide capabilities similar to the dataflow scientific visualization packages AVS and IRIS Explorer, but tailored toward the virtual reality interface. This capability would greatly facilitate application development.

Networks. High-bandwidth, low-latency networks already exist in the form of gigabit local-area networks. Such high performance needs to be implemented on long-haul networks to facilitate shared visualization by researchers in widely scattered locations.

Architectures. Software architectures need to support access to data and computations resident on both local and remote computers. This architecture should scale and make transparent the location of the data or computation, while satisfying real-time performance constraints. Also needed are software architectures that facilitate development of highly concurrent yet synchronized environments. Operating systems that provide multiprocessing capability only partly address this need.

Visual displays. Promised are higher resolution, wider field of view, lower levels of optical distortion, and improved form factors. It is possible that head-mounted displays will not be widely accepted by researchers until they are similar in form factor to sunglasses.

Input devices. Needed are higher-accuracy and longer-range trackers, as well as algorithms for inferring user intentions from the tracker data. Devices may include improved gloves, better button devices, speech recognition, and so on. Which device is appropriate for which context is yet to be determined.

In addition to the immature state of the technology, there is also an immaturity in our knowledge of how to build useful applications. Scientific visualization provides an opportunity to experiment with such applications in a setting that makes reasonable demands of the technology yet requires interesting and imaginative approaches to system design. For this reason, we feel that virtual reality and scientific visualization will provide a rich and fruitful interplay for years to come. □

References

1. Bajura, M., Fuchs, H., and Ohbuchi, R. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. In *Computer Graphics, the Proceedings of SIGGRAPH*

- '92 26, 2 (Chicago, July 1992).
2. Bancroft, G.V., Merritt, F.J., Plessel, T.C., Kelaita, P.G., McCabe, R.K., and Globus, A. FAST: A multi-processed environment for visualization of computational fluid dynamics. In *Proceedings of IEEE Visualization '90* (San Francisco, Oct. 1990).
 3. Brooks, F.P., Jr., Ouh-Young, M., Blatter, J.J., and Kilpatrick, P.J. Project GROPE—Haptic displays for scientific visualization. In *Computer Graphics, the Proceedings of SIGGRAPH '90* (Dallas, Tex., Aug. 1990).
 4. Bryson, S. Virtual environments for scientific visualization. Course notes for IEEE Visualization '92 (Boston, 1992); and The implementation of immersive virtual environments, course notes, SIGGRAPH '92 (Chicago, July 1992).
 5. Bryson, S. Virtual reality takes on real physics applications. *Comput. Phys.* 6, 4 (July/Aug. 1992).
 6. Bryson, S. Virtual Spacetime: An environment for the visualization of curved spacetimes via geodesic flows. In *Proceedings of IEEE Visualization '92* (Boston, 1992).
 7. Bryson, S., and Gerald-Yamasaki, M. The distributed Virtual Wind Tunnel. In *Proceedings of Supercomputing '92* (Minneapolis, Nov. 1992).
 8. Bryson, S., and Levit, C. The Virtual Wind Tunnel: An environment for the exploration of three dimensional unsteady flows. In *Proceedings of Visualization '91* (San Diego, Calif., Oct. 1991); also in *Computer Graphics and Applications* (July 1992).
 9. Cruz-Neira, C., Leigh, J., Barnes, C., Cohen, S., Das, S., Englemann, R., Hudson, R., Papka, M., Siegel, L., Vasilakis, C.D., Sandin, J., and DeFanti, T.A. Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality* (San Jose, Calif., Oct. 1993).
 10. Cruz-Neira, C., Sandin, D.J., and DeFanti, T.A. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics, the Proceedings of SIGGRAPH '93* (Las Vegas, Nev., Aug. 1993).
 11. Deering, M. High-resolution virtual reality. In *Proceedings of SIGGRAPH '92* (Chicago, July 1992).
 12. Hultquist, J.P., and Raible, E.L. SuperGlue: A programming environment for scientific visualization. In *Proceedings of IEEE Visualization '92* (Boston, Oct. 1992).
 13. Iwata, H. Volume haptization. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality* (San Jose, Calif., Oct. 1993).
 14. McCormick, B., DeFanti, T.A., and Brown, M.D., Eds. Visualization in scientific computing. *Comput. Graphics* 21, 6 (1987).
 15. Sheridan, T.B., and Ferrill, W.R. *Man-Machine Systems*. MIT Press, Cambridge, Mass., 1974.
 16. Smith, M., Chawla, K., and Van Dalsem, W. Numerical simulation of a complete STOV aircraft in ground effect. Paper AIAA-91-3293. American Institute of Aeronautics 9th Aerodynamics Conference (Baltimore, Md., 1991).
 17. Song, D., and Norman, M.L. Cosmic explorer: A virtual reality environment for exploring cosmic data. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality* (San Jose, Calif., Oct. 1993).
 18. Taylor, R.M., Robinett, W., Chi, V.L., Brooks, F.P., Jr., and Wright, W. The Nanomanipulator: A virtual reality interface for a scanning tunnelling microscope. In *Proceedings of SIGGRAPH '93* (Las Vegas, Nev., Aug. 1993).

About the Author:

STEVE BRYSON is a research scientist with MRJ, Inc., under contract to NASA Ames Research Center. **Author's Present Address:** MS T27A-1, NASA Ames Research Center, Moffett Field, CA 94035; email: bryson@nas.nasa.gov

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© ACM 0002-0782/96/0500 \$3.50