The visualization management system approach to visualization in scientific computing

D. M. Butler
Limit Point Systems, 39120 Argonaut Way, Suite 397, Fremont, California 94538
M. H. Pendley
Sandia National Laboratories, Livermore, California 94551-0969

(Received 3 October 1988; accepted 29 December 1988)

This paper introduces the visualization management system (ViMS), a new approach to the development of software for visualization in scientific computing (ViSC). The conceptual foundation for a ViMS is an abstract visualization model that specifies a class of geometric objects, the graphic representations of the objects, and the operations on both. A ViMS implements a visualization model and provides four levels of mechanisms for developing applications. The ViMS requirements and a model-independent ViMS architecture are described. The ViMS approach presents several important opportunities for visualization standards.

INTRODUCTION

In this paper, we introduce the visualization management system (ViMS), a new approach to software for visualization in scientific computing (ViSC). As an emerging field of research, visualization in scientific computing has been the focus of much attention recently. In particular, the Panel Report on Visualization in Scientific Computing[1] described the dilemma of current, nonvisual computing practice: the need to deal with too much data, the need to communicate visual information, and the need to interact with calculations in progress. The Panel Report emphasized the development of greatly enhanced visualization facilities as a solution to these problems and proposed the development of advanced "visualization environments."

A visualization management system is such an environment. It is an abstraction of data structures and functions common to many ViSC applications. It is a generalized, application-independent facility for the definition, analysis, and presentation of visual representations of scientific data. In this paper, we introduce and develop the visualization management system approach. The paper is organized as follows. In Sec. I, we introduce and motivate the general notion of a visualization management system and its conceptual basis, the visualization model. In Sec. II, we discuss the requirements a ViMS must satisfy and in Sec. III, we use them to develop a ViMS architecture. In Sec. IV, we summarize the requirements for the visualization model. In Sec. V, we conclude by discussing the benefits of the ViMS approach and its relationship to visualization standards.

I. ViMS APPROACH

The design of visualization environments is an open research question that is best approached in the larger context of *systems for processing scientific data*. A systems approach emphasizes the overall productivity of the entire scientific computing environment. It ensures that the visualization facilities are smoothly integrated with other computing facilities and that the productivity of both users and developers is optimized. While the systems approach has not been emphasized in scientific data processing, it has dominated business data processing, where the productivity of the computing environment receives a great deal of attention. The problems faced in business data processing, although different in detail from those of scientific data processing, are the same in general: controlling and interpreting large amounts of data, while optimizing both programmer and user productivity. Hence it is useful to examine business data processing systems as a guide to developing systems for processing scientific data.

An outstanding feature of business systems is the central role played by database management systems (DBMS). Most business systems are designed and implemented around commercially available, application-independent database management systems. The database management system can be considered an abstraction of those data structures and functions common to many business applications. These generalized, application-independent data structures and functions facilitate the definition, storage, retrieval, and analysis of (business) data.

Two features common to most modern DBMS's are worth emphasizing.

First, the conceptual foundation of a DBMS is a data model. A data model is an abstract specification of a collection of data types, a collection of operations on the types, and a collection of constraints specifying the valid states of instances of the types. The best known example is the relational data model; other well-known models include the hierarchical model and the network model. A DBMS is a constellation of facilities that implements the data model.

Second, a DBMS usually provides four levels at which it can be specialized to a particular application. First, there are low-level mechanisms for optimizing, modifying, or extending internals of the system. Second, there are middle-level components, subroutine libraries, that can be embed-

ded in application software. Third, there are high-level tools, such as interactive report generators and query languages, that can be configured to an application's needs. Fourth, some or all of these facilities are integrated with the facilities of a general-purpose programming language to produce a "fourth generation" language, a very high-level language specialized for solving problems stated in terms of the data model. With these features, the modern DBMS is a general, flexible, and productive environment for solving problems in business data processing.

Similarly, a visualization management system is an abstraction of data structures and functions common to many ViSC applications. These abstract, application-independent data structures and functions facilitate the definition, analysis, and presentation of visual representations of scientific data. The conceptual foundation of a ViMS is a visualization model that specifies a class of geometric objects, their graphic representations, and their operations. As with database management systems, a visualization management system is an implementation of the model and the implementation provides four levels at which the model can be specialized to the application: extendable internals, embeddable components, configurable tools, and a very high-level language.

II. ViMS REQUIREMENTS

The analogy with DBMS has helped us define the general conceptual structure and implementation of a visualization management system, but the unique requirements of scientific visualization determine its architecture and features. In this section, we develop specific ViMS requirements: application independence, integrated visualization and computation, flexible geometric representation, flexible graphic representation, data representation independence, flexible distribution, and host independence. Wherever possible, we connect these requirements to the perceived limitations of existing systems. Our discussion is necessarily brief; many of these requirements are described in more detail in the Panel Report. We discuss the requirements starting with the more general and abstract and proceeding to the more concrete and implementation oriented.

A. Application independence

To be generally useful and justify the effort required to develop it, a ViMS must be application independent. Application independence encourages using existing software for new applications. Since it is the nature of scientific computing that every application is new, software development overhead is the major barrier to the routine use of visualization techniques. Application independence removes this barrier.

In contrast, existing systems have typically been developed for specific applications, and, as a result, are suitable only for their original application. Many scientists use visualization techniques rarely or not at all simply because the development effort required is unacceptable.

B. Integrated visualization and computation

The function of visualization is to help the scientist explore relationships in scientific data. Effective exploration requires rapid cycling between visualization and additional computation, or even visual "steering" of the computation. Integrated visualization and computation facilities encourage exploration. In addition, they reduce the effort required to develop new computational processes, since these processes can be embedded in the existing visualization facility.

Existing data processing systems are typically structured with computational processes separate from interpretive "postprocessors," greatly inhibiting the exploration process. Typically, developing a new computational process requires developing a new postprocessor.

C. Flexible geometric representation

An essential feature of scientific visualization problems, not usually present in other visualization fields such as computer-aided design, is that the data typically have no unique geometric interpretation. This is due in large part to the abstract nature of typical scientific models and the use of sophisticated mathematical formalisms. Data structures with large numbers of dimensions and complicated topologies occur frequently, making the data not directly visualizable. These properties require the visualization management system to have a flexible, interactive mechanism for associating the data with abstract, multidimensional geometric representations. The geometric representations must be distinct from the two- or three-dimensional graphic representations. Flexible geometric representation encourages exploring different geometric interpretations of the data, an important aid to understanding the data.

Existing systems typically interpret the data as some particular combined geometric/graphic structure of dimensionality three or less. Changing the representation, for instance by slicing an object into planes or combining several curves into a surface, requires reprogramming. Inflexible geometric representation severely limits the effectiveness of visualization as a technique for understanding scientific data.

D. Flexible graphic representation

As we just described, the graphic representation and the geometric representation must be distinct. A given geometric object may have several graphic representations. For instance, a real function of two real variables may be represented as a surface in three dimensions, a contour plot, or a pseudocolor image. Again, exploring different graphic representations of the data is an important aid to understanding the data, and the visualization system must provide a flexible mechanism for selecting a particular graphic representation.

Current systems typically present a given geometric structure in only one graphic style. Changing the presentation often requires reprogramming.

E. Data representation independence

To achieve maximum generality and to minimize maintenance effort, the ViMS software must be designed to avoid data representation dependence between the ViMS and the application or between the various subsystems of the ViMS itself. In particular, data representation dependence be-

tween the geometric representation and the graphic representation must be avoided.

Existing systems are typically highly data representation dependent. A particularly common and frustrating example of data representation dependence is file dependence. Current systems typically incorporate file access directly, utilizing FORTRAN READ's or WRITE's and their associated FORMAT statements throughout the code as needed. This lack of modularization makes the system dependent on the file format of some application. Thus data representation dependence can generate application dependence as well.

F. Flexible distribution

Data, visualization, and computation resources are increasingly distributed across computer networks. To maximize access to these resources, the various facilities of a ViMS must be flexibly distributable. Users with entry level facilities should be able to dispatch computation or graphic tasks to remote resources; users at large installations should share sophisticated, high-performance resources.

Existing systems are typically not distributable. Data transfer and access to resources are major limitations to the routine use of visualization techniques.

G. Host independence

The advantages of host independence, standards, and the "open systems" approach are recognized throughout the computing industry. Applied to visualization management systems, host independence has two particular advantages. First, host independence minimizes maintenance overhead associated with hardware or software upgrades. Second, host independence maximizes the availability of visualization functionality. If the functionality is available on a wide range of hosts, entry barriers based on cost are minimized and connectivity problems associated with getting the data to a particular host are avoided.

Current visualization software tends to rely on specific hardware and software environments; as a result, maintenance and access problems dominate its use. For instance, the cost of porting visualization software dependent on the features of specific display systems can effectively lock the visualization functionality onto obsolete display technology. As another example, visualization functionality dependent on specific, high-performance hardware is inaccessible to most users.

III. ViMS ARCHITECTURE

As described above, a visualization management system is an abstraction of data structures and functions common to many ViSC applications. It is a generalized, application-independent facility for the definition, analysis, and presentation of visual representations of scientific data. A ViMS is an implementation of an abstract visualization model. The model specifies a class of geometric objects, the graphic representations of the objects, and the operations on both. The ViMS requirements identified in Sec. II can be satisfied in part by choosing an appropriate architecture for the implementation; the remaining requirements must be satisfied by choosing an appropriate model. In this section we develop a model-independent ViMS architecture, identify the requirements it satisfies, and identify the remaining requirements that the visualization model must satisfy.

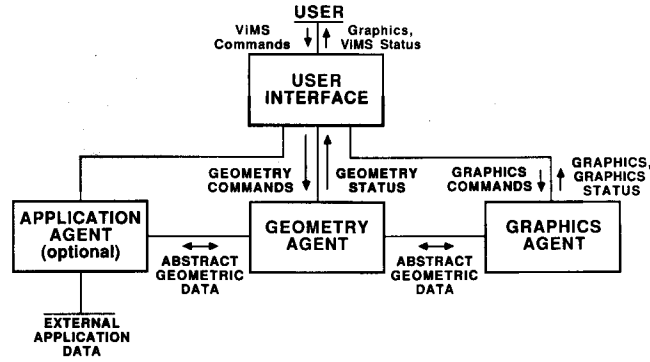

FIG. 1. A model-independent ViMS architecture.

To implement the visualization model, the ViMS architecture must provide an agent to implement the geometric objects, an agent to implement the graphic representations, and an agent to invoke and control the operations. This architecture is depicted in Fig. 1. Also shown is an optional entity for interfacing to external applications, the application agent. We describe the features of this design by reviewing it with respect to each of the requirements discussed above.

A. Application independence

The application agent provides an interface to external applications, abstracting geometric data from the application format and allowing the rest of the system to be application independent. The application agent itself is application dependent.

Except for the architectural support supplied by the application agent, the application independence of the system is determined by the visualization model; if we choose a model with a very limited class of geometric objects, oriented to a specific application, we have an application-dependent ViMS. Thus the visualization model is required to provide a class of geometric objects suitable for a wide range of applications.

B. Integrated visualization and computation

The architecture provides integrated visualization and computation within the context of the visualization model. Operations on the geometric objects are distinct from operations on the graphic representations, but the two types can be freely and interactively mixed. Any computational processes that can be defined in terms of the geometric objects can be embedded in the system and take full advantage of its facilities. The geometric objects of the visualization model must be chosen to provide useful and widely applicable computational operations.

C. Flexible geometric representation

The geometry agent implements the class of geometric objects defined by the visualization model. The major function of the geometry agent is to construct instances of this class using data acquired from the application, following

commands from the user. The term "agent" emphasizes the high degree of control the architecture provides the user. The user interface subsystem and four-level implementation can provide the user with interactive or programmed geometric object construction. The architecture thus provides flexible geometric representation within the class of geometric objects defined by the visualization model. To match the properties of scientific data, the model must support multiple dimensions and complex topologies.

D. Flexible graphic representation

The geometry agent and the graphics agent are separate subsystems, hence the graphic representation can be constructed and maintained independently of the geometric object. For each geometric object type, the graphics agent implements the set of graphic representations defined by the visualization model. The model must provide a flexible and complete set of graphic representations.

E. Data representation independence

Modern software engineering emphasizes the use of abstract data types to avoid representation dependence. An abstract data type provides a data structure and a collection of operations which encapsulate the data structure, protecting it from clients of the data type. The operations allow the data to be manipulated only in conformance with the specification of the data type. Systems developed using abstract data types tend to consist of collections of data types organized into layers, interacting at precisely defined interfaces.

We assume that the geometry agent, the graphics agent, the user interface, and the various subcomponents of all three are implemented as abstract data types, minimizing data representation dependence between them. Again, the success of this approach depends on the visualization model. The interfaces between the geometry agent and the application and graphics agents are defined in terms of the geometry. Flexible, efficient interaction between these agents requires the geometric objects defined by the visualization model to provide general, multilevel access to the geometric information.

F. Flexible distribution

The components of the system are loosely coupled and communicate with each other through a message passing mechanism. This allows the possibility of distributing the system in a variety of ways. Some of the more important distributions are the following.

(1) The application agent is remote while the remaining components are local. This is advantageous when the application file is very large, but only a small portion of it is visualized at one time. Only the objects viewed need be shipped across the network.

(2) The application agent and the geometry agent (or some portion of it) are remote and the remaining components are local. This distribution is useful for operations that are compute intensive or require large input data sets but produce relatively smaller output data sets. Only the smaller output data sets need be shipped across the network.

(3) Subcomponents of the graphics agent, such as compute-intensive rendering algorithms, are remote while the rest of the system is local. This distribution allows sharing of special-purpose rendering hardware.

(4) The user interface is local and the rest of the system is remote. This distribution provides full functionality to users with entry level facilities.

G. Host independence

Host independence is largely an implementation issue, rather than an architectural issue. Recently developed and evolving formal and *de facto* standards for operating systems, programming languages, windowing systems, and graphics can be used to provide host independence.

IV. VISUALIZATION MODEL REQUIREMENTS

In Sec. III, we reviewed each ViMS requirement, identifying those satisfied by the model-independent architecture and those remaining to be satisfied by the visualization model. For clarity and future reference, we recapitulate the visualization model requirements here.

(1) Application independence: the visualization model must provide a class of geometric objects suitable for a wide range of applications.

(2) Integrated visualization and computation: the geometric objects of the visualization model must provide useful and widely applicable computational operations.

(3) Flexible geometric representation: the geometric objects must support multiple dimensions and complex topologies.

(4) Flexible graphic representation: the model must provide a flexible and complete set of graphic representations.

(5) Data representation independence: the geometric objects defined by the visualization model must provide general, multilevel access to the geometric information.

The overall efficacy of a ViMS is largely determined by the generality, flexibility, and expressive power of the visualization model. Indeed, the feasibility of the entire approach relies on finding a visualization model which meets the rather stringent requirements we have listed here. In a companion article,[2] we describe a specific visualization model, based on the mathematics of fiber bundles, that satisfies these requirements.

V. POTENTIAL BENEFITS OF THE ViMS APPROACH

To conclude, we discuss the potential of the ViMS approach. Central to the ViMS approach is the emphasis on the formal visualization model, which in effect provides a reference model for scientific visualization. The visualization model and the interfaces in the ViMS architecture provide several opportunities for standardization.

At the interface between the application agent and the geometry agent there exists the potential for a standard application interface. The principle strategy of this standard would be to abstract from the application data the information required by the geometry. This approach is similar in spirit to the use of device drivers in operating systems or the Computer Graphics Virtual Device Inter-

face standard.[3] In practice, any application could gain access to the full ViMS functionality by writing a "file driver" meeting the requirements of the standard. This type of interface has already been used successfully in at least one organization.[4]

The interface between the geometry agent and the graphics agent provides another important opportunity for standardization. The existence of a standard, data-representation-independent interface between the geometry and graphics agents would greatly facilitate the development and dissemination of new functions in both.

A third, and extremely important, standardization opportunity generated by the abstract visualization model is at the user interface. The emphasis on a formal, abstract model frees the user from the details of the data representation and presents a uniform paradigm across a wide spectrum of tools and applications. Returning to the analogy with database management systems, one of the original motivations for the relational data model, and a major contributor to its wide acceptance, is that it presents the user with an abstract interface free from concerns with implementation details and suitable for standardization. Indeed, the relational data model has led to a standard database query language, the recently established ANSI SQL standard.[5]

A final, and probably most fundamental, standardization opportunity arises because a formal visualization model generates a taxonomy of visualization. It defines and classifies the geometric objects and their visual representations. A formal taxonomy introduces the potential for the standardization, in the sense of calibration, of visualization itself. If computer visualization is to realize its full potential as a research tool, the techniques must be precisely understood and communicated. Visualization cannot just produce compelling, aesthetically pleasing pictures; it must produce *well-defined* pictures. This need has been recognized in medical imaging, where visualizations such as CAT scans are defined by published protocols for acquiring, processing, and displaying the data and calibrated by standard intensity scales. In contrast, a precise, quantitative understanding of a scientific visualization can currently be obtained only by detailed inspection of the code producing it, a painful and unreliable process at best. A formal taxonomy provides the basis for establishing and communicating the definition of a visualization.

We close by returning to the analogy with database management systems. In business data processing, the database management system has generated a wealth of host-independent, increasingly standardized and productive software environments supported by an extensive theoretical foundation. We hope the visualization management system approach will have the same success in scientific data processing.

ACKNOWLEDGMENTS

We would like to thank Joe Harris and Stewart Keeton for helpful comments on a first draft of this article.

This work was supported by the United States Department of Energy under Contract No. DE-AC04-76DP00789.

REFERENCES

1. B. H. McCormick, T. A. Defanti, and M. D. Brown, Comput. Graphics **21**, 1 (1987).
2. D. M. Butler and M. H. Pendley, Comput. Phys. **3**(5), 45 (1989).
3. T. Powers, A. Frankel, and D. Arnold, IEEE Comput. Graphics Appl. **6**, 33 (1986).
4. L. A. Treinish and M. L. Gough, *A Software Package for the Data-Independent Management of Multidimensional Data* (National Space Science Data Center, NASA Goddard Space Flight Center, Greenbelt, MD, 1987).
5. C. J. Date, *A Guide to the SQL Standard* (Addison–Wesley, Reading, MA, 1987).