Information Visualization in Virtual Environments

Steve Bryson, NASA Ames Research Center

1. Introduction

Virtual Environments provide a natural setting for a wide range of information visualization applications, particularly when the information to be visualized is defined on a three-dimensional domain (Bryson, 1996). This chapter provides an overview of the issues that arise when designing and implementing an information visualization application in a virtual environment. Many design issues that arise, such as, e.g., issues of display, user tracking are common to any application of virtual environments. In this chapter we focus on those issues that are special to information visualization applications, as issues of wider concern are addressed elsewhere in this book.

1.1. What is the problem?

In this chapter, we take information visualization to be the visual investigation of numeric data. This data may represent real-world observations or the result of simulation. The data may express tangible, real-world quantities such as water temperature, or may represent abstract quantities such as stock market prices. We restrict the data to "information" rather than "objects", in order to avoid having to discuss representation of conventional real-world objects. Using this restriction we can concentrate on those aspects of information visualization that are unique to non-realistic, abstract representations of information. Of course there is considerable overlap between "information" and "object", e.g. we may want to represent topographic data. If that representation is abstract in order to bring out some feature of the topography, the discussion in this chapter is relevant. If the topography is to be represented realistically, then the discussion here may not apply.

Virtual Environments (VEs) provide unique opportunities for information visualization (Bryson, 1996, Cruz-Neira et. al., 1993, Lin et. al., 2000). The inherent three-dimensional nature of virtual environments

makes VEs natural for the investigation of three (or higher)-dimensional data on three-dimensional domains. The natural, anthropomorphic interface suggested in virtual environments inspires natural, intuitive interfaces for the investigation of data. This intuitive interface may be particularly suitable for visualizing highly complex data sets, where phenomena in the data cannot be effectively displayed all at once. In such situation an intuitive three-dimensional interface may allow for the rapid investigation of data employing a "what's happening here?" paradigm.

In this chapter we will explore how virtual environments can be effectively used for information visualization. After characterizing the information visualization process, we will examine how information visualization environments differ from other, more "real-world"-oriented applications. Using these observations we will develop implementation strategies tailored for information visualization in virtual environments, including issues of run-time software architectures, distribution, and control of time flow. These implementation strategies will be driven by consideration of the human factors of interaction. We will close with a discussion of the challenging aspects of information visualization in virtual environments.

1.2. The Data Analysis Pipeline

In order to describe the special issues that arise in the implementation of an information visualization system in virtual reality, we require a conceptual model of an information visualization system. There are many ways to conceptualize information visualization, and we make no claim to the most complete or optimal conceptualization. We have, however, found the following very informative when considering implementation issues.

We consider the information visualization process as a pipeline, which in its most generic form starts with the data to be visualized. From this data visualization primitives are extracted. These primitives may consist of vertices in a polygonal representation, text for a numerical display, or a bitmap resulting from, for example, a direct volume representation. Primitive extraction typically involves many queries for data

values. The extracted primitives are then rendered to a display. This pipeline allows user control of all functions, from data selection through primitive extraction to rendering. We show this pipeline in figure 1.

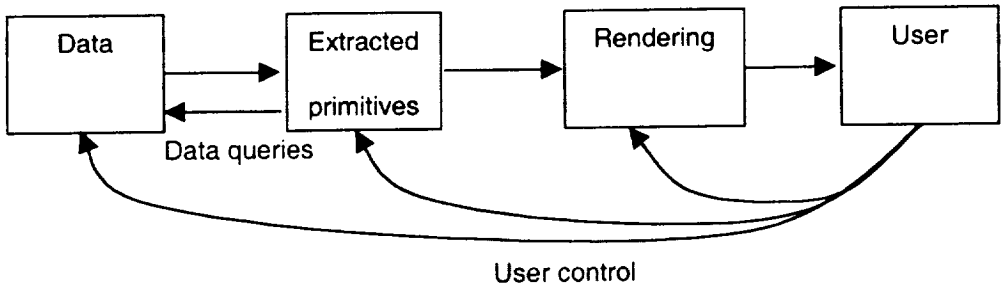

Figure 1: The Data Analysis Pipeline

As an example of this pipeline in operation, consider streamlines of a vector field (see, e.g., Bryson, 1998) . Given a starting point of a streamline, data (the vectors at that point) are accessed by the streamline algorithm. The vector value is then added (sometimes with a complex high-accuracy algorithm) to the starting point, creating a line primitive. This process is iterated to build up a (typically curved) line with many vertices. These vertices are the streamline's extracted representation. These vertices are then rendered in the visualization scene. The extraction of these primitives may involve significant computation even though the data may exist as a pre-computed file. Computations like those in this example will turn out to be a significant issue in the implementation of information visualization in virtual environments.

1.3. Why Perform Information Visualization in a Virtual Environment?

Virtual environments offer several advantages for many classes of information visualization (Bryson, 1996, Cruz-Neira et. al., 1993, Lin et. al., 2000, Song and Norman, 1993). The inherent three-dimensional interface makes virtual reality a natural setting for information which exists on a three-dimensional domain. Examples of such data include three-dimensional scientific simulations such as fluid flow, electromagnetic fields or statistical data indexed by location in three-dimensional space. Three-dimensional interaction techniques common in virtual environments provide natural ways to control visualization selection and control in 3D. In addition experience has shown that one of the greatest advantages of information visualization in virtual environments is the inherent "near-real-time" responsiveness required by a head-

tracked direct-manipulation virtual environment. This responsiveness allows rapid queries of data in regions of interest. Maintaining this responsiveness in an information visualization environment is the most challenging aspect of such an application and will be one of the primary foci of this chapter. When designed well, the combination of three-dimensional display, three-dimensional interaction and rapid response creates an intuitive environment for exploration and demonstration.

Figures 2 and 3 show examples of virtual environments for the visualization of information arising from scientific computation. Figure 2 shows a system developed at the University of Houston for the exploration of geophysical flows (Lin et. al., 2000). Figure 3 shows the Virtual Windtunnel, developed at NASA Ames Research Center, which is used to investigate the results of simulations in Computational Fluid Dynamics (Bryson and Levit, 1991, Bryson et. al. 1997). Both examples exhibit the use of multiple visualization extracts in a single environment. All of these extracts are interactive and can be changed under user control.

Figure 2: an information visualization virtual environment for the exploration of geological data sets.

Courtesy of the University of Houston/Bowen Loftin

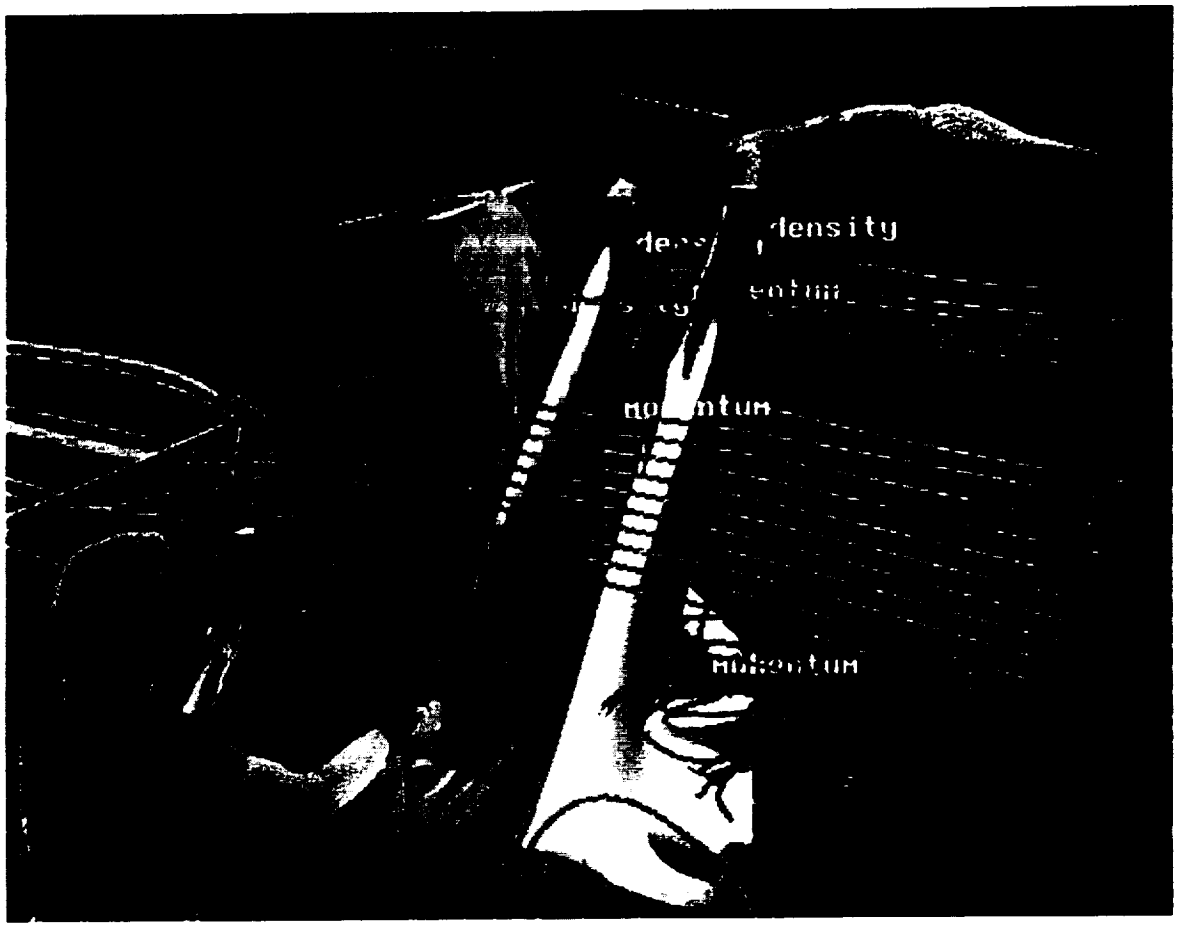

Figure 3: The Virtual Windtunnel, a virtual environment for the visualization of results of simulations arising in Computational Fluid Dynamics

Multimodal interfaces, using sound, touch and force displays for virtual environments have been explored for information visualization in virtual environments (e.g. Taylor et. al., 1993). With the possible exception of force displays, however, the efficacy of such interfaces has not been demonstrated.

1.4. How Information Visualization Differs from Other VE Applications

The design and development of an information visualization application within a VE is different from most virtual reality application development. Information visualization environments are often abstract and involve large amounts of data access and computation in response to a query. For time-varying data, different senses of time arise, in which the data may evolve more slowly or even backwards relative to user

time. Such differences between information visualization environments and more conventional VR applications can be generalized into the following areas:

- **Greater flexibility in graphical representation:** The inherently abstract nature of information implies opportunities for simpler, faster graphics, such as representing a streamline as a simple polyline. Conventional applications, such as training or entertainment, are typically more concerned with realistic graphical environments, and so may have less flexibility in the selection of graphical representation. Of course this is not a universally applicable rule, as some representations of information such as direct volume rendering can be very graphically intensive.
- **A large amount of numerical computation may be required:** While information visualization typically addresses pre-existing data, visualization extracts may themselves require considerable computation. Streamlines or isosurfaces in the visualization of continuous vector and scalar fields are well-known examples that require large amounts of computation. Statistical or data mining techniques for abstract data may also require computation. As more sophisticated data analysis techniques are used in virtual environments more computational demands can be expected.
- **A large amount of data access may be required:** Visualization extracts require access to data, and some extracts require more data access than others. Complete isosurfaces, for example, require traversing an entire data set (at a particular timestep for time-varying data) for the computation of the data set. Time-varying data sets can be extremely large, requiring access to hundreds of gigabytes of data in a single analysis session, albeit a single timestep at a time.
- **There may be various senses of time:** As discussed below in section 4, several senses of time can arise in an information visualization system, particularly when addressing time-varying data sets. While some of these senses of time correspond to conventional time in other VE applications, completely new ways of thinking of time arise from the fact that a user may wish to manipulate time flow in a time varying environment. This issue and its impact are discussed in much more detail in section 4.

These differences between an information visualization virtual environment and other applications have major impacts on the design of the virtual environment system. These impacts are the focus of this chapter.

2. System Architecture Issues Specific to Information Visualization in VEs

There are several issues that arise in the design and implementation of virtual environments for information visualization. In this section we examine some of these issues in detail. First, however, we must classify the types of interaction that may occur, which in turn depends on the time flow of the data.

2.1. Classification of interaction and time flow

There are two design questions that must be answered before considering an appropriate implementation of an information visualization application in virtual reality:

- **Is the user allowed to interactively query the data at run time, generating new visualization extracts?** If so, the system will likely have user-driven data accesses and computation to support extraction of new visualization geometry.
- **Is the data time-varying?** If so, there will be at least two senses of time in the virtual environment: user time and data time. The flow of data time may be put under user control so that it may be slowed, stopped, reversed, or times randomly accessed.

These questions are independent, and both must be answered in order to determine which kind of implementation strategy will be appropriate. There are four combinations that arise from the answers to these two questions:

Non-interactive, non-time-varying data: This is the simplest information visualization environment, where the visualization geometry can be extracted ahead of time and displayed as static geometry in a head-tracked virtual environment. No data access or computation issues occur in this case. The user may be able to rotate or move the static geometry. The design issues that arise in this case are common to all virtual reality applications and so will not be considered further in this chapter.

Non-interactive, time-varying data: In this case the visualization extract geometry can be pre-computed as a time series which may be displayed as a three-dimensional animation in the virtual environment. The

user may be given control over the flow of the animation, to slow it down, stop it or reverse direction. Such user-controlled time flow implies a special interface which may include rate controls to determine the speed and direction of the data time or a timestep control for the random access of the extracted geometry at a particular data time. When the extracts are defined for discreet timesteps, interpolation in time may be desirable to allow for a smooth flow between timesteps. What kind of interpolation is appropriate will be highly domain and application dependent. Except for these considerations, however, the issues that arise in this case are common to most virtual reality applications and so will not be considered further in this chapter.

Interactive, non-time-varying data: In this case the data does not change in time, but the user specifies the visualization extracts at runtime. In a virtual environment such extracts may be specified via a direct-manipulation interface where the user either specifies a point or manipulates an object in three-dimensional space. The visualization extract may require significant computation, which will have an impact on the responsiveness of the system. This impact is discussed at length in section 2.2. When visualization extracts are do not change they are typically not recomputed.

Interactive, time-varying data: For this type of environment, the data itself is changing with time so any existing visualization extracts must be recomputed whenever the data time changes. This can result in a large amount of computation for each data timestep, the implications of which are discussed below in section 2.3. In addition, the user may be given control over the flow of data time, allowing data time to run more quickly or slowly, or in reverse.. The user may wish to stop time and explore a particular timestep. When time is stopped the system should act like an interactive, non-time-varying data environment, allowing visualization extracts to be computed in response to user commands.

2.2. Interaction vs. Data Time Scales

Responsive interaction plays two important roles in virtual reality:

- **Fast graphics** is required to permit the head-tracked display required for a strong sense of object presence and immersion.

- **Fast response** to user commands are required for a direct manipulation interface paradigm, where the user is allowed to directly "pick up and move" objects in the virtual environment.

In information visualization environments we have the added requirement that the user be able to move a data probe displaying a visualization extract, and see that extract update in near-real-time as appropriate to its new position. If the data probe's extract updates sufficiently quickly, the user will have a sense of exploring the data in real time.

How fast the graphics and interaction response must be turns out to be both application and domain dependent. A virtual environment for training manual tasks such as a flight simulator requires response times of less than 1/30 of a second. Longer response times have been observed to train the subject to expect delays that may not exist in the real world and lead to incorrect operation (Sheridan and Ferrill, 1974). Information visualization, however, does not typically require fidelity to any real-world time scales. In fact it is often desirable that the information being visualized be presented with very different time scales from the real world, with data time being either slowed down or sped up depending on the user's needs.

While fidelity to real-world time scales is not a guide to the required responsiveness of an information visualization environment, human factors considerations will set performance requirements. Three human factors issues turn out to be important for information visualization in virtual environments:

- **Graphics Update Rate:** How fast must the graphical display update rate (graphical animation rate) be to preserve a sense of object presence and/or immersion? By graphical update rate we mean the rate at which frames are drawn to the frame buffer(s), not the display device's refresh rate.
- **Interaction Responsiveness:** How quickly must objects in the environment respond to user actions in order to maintain a sense of presence and direct manipulation? This requirement applies to the overall head-tracked display as well as ability to interact with objects.
- **Data Display Responsiveness:** How fast must interactive data display devices, such as a data probe, update to give the user a sense of exploring the data?

The relationships and differences between these time scales are subtle. Graphics update rate will limit the interaction and data display responsiveness because the interactive displays cannot be presented faster than

the graphics update rate. Update rate and responsiveness, however, are very different kinds of measures: update rate is measured in frames/second, while responsiveness is the **latency**, measured in seconds: the time interval between a user action and when the system's response is displayed. This latency is determined by all processes triggered by the user's action, from reading the user tracking devices through processing the user's commands through possible subsequent computation to the display of the result.

Experience has shown that the limits on these time scales are (see Bryson, 1996 and Sheridan and Ferrill, 1974):

- **The graphics update rate must be greater than 10 frames/second.** While faster update rates are desirable, 10 frames/second is sufficient to maintain a sense of object presence even though the discrete frames of the display are easily perceptible. Slower update rates result in a failure of the sense of object presence, eliminating the enhanced three-dimensional perception advantages of a virtual environment.
- **Interaction responsiveness must be less than 0.1 seconds.** While lower latencies and faster responsiveness is desirable, a latency of 0.1 seconds is fast enough to give the user a good sense of control of objects in the virtual environment. Longer latencies typically cause the user to experience unacceptable difficulty in selecting and manipulating objects in three-dimensional space.
- **Data display responsiveness must be less than about 1/3 of a second.** While faster responsiveness is desirable, a data display latency of 1/3 of a second maintains a sense of "exploring" the environment, though the user may use slow movements to adapt to this relatively long latency. Longer latencies in data display require such slow movements on the part of the user as to lose usability.

The graphics update rate and the interaction responsiveness requirements are similar: one graphics frame of latency is allowed for to maintain good responsiveness for user interaction. The data display responsiveness requirement, however, is less restrictive. The difference in latency requirement between interactivity and data displays is due to the fact that user interaction (selecting, acquiring and moving objects) is a manual task driven by the human factors of manual control, while observing data display

during movement is an intellectual task, in which the user observes what happens as a data probe is moved through space.

Interactive time-varying environments potentially present a difficult challenge in meeting the above requirements: all non-precomputed visualization extracts must be computed whenever the timestep changes. Furthermore, when the timestep changes, all computations must take place before any of the extracts can be displayed so that extracts from different data timesteps are not displayed at the same time. Experience has shown that it is acceptable for the data timestep to change quite slowly, so long as the 10 frames/second graphical update rate and the 1/3 second data responsiveness requirements are met.

These observations suggest that graphical display and user interaction be treated independently of the visualization. For example, interaction and visualization display may be treated independenttly by having simple graphical tools in the virtual environment which respond to the user without waiting for data to be displayed (Herndon and Meyer, 1994 and Bryson et. al., 1997). These tools may trigger associated data displays, but these data displays should occur asynchronously from the display of the tool, allowing the tool to respond to the user's input as quickly as possible.

A simple crosshair in three-dimensional space, with an associated streamline of a vector field, is an example of such a tool. In this example, the user can "pick up and move" the crosshair, which will be very responsive (within the limits of the graphical update rate) due to its simple graphical nature. When this crosshair is moved it will trigger the computation of a streamline at its current location. This streamline may take much longer to compute and display, after which time the tool will trigger a new streamline from its new location. If the process computing the streamline runs asynchronously from the process handling user interaction and graphical display, interaction responsiveness will be only slightly impacted by the computation of the streamline (assuming a pre-emptive multi-tasking operating system). This example shows that the difference in time scales between the interaction responsiveness and data display responsiveness has strong implications for the run-time architecture of the information visualization system. These implications for the overall system architecture are discussed in section 2.3.

The method of triggering computations outlined in this simplified scenario will be discussed in more depth in section 4 below.

2.3. System Architecture

The observations in the previous section imply that any implementation of an interactive information visualization virtual environment in which computation of visualization extracts takes place should contain at least two asynchronous processes: a graphics and interaction process and a visualization extract computation process. More generally, the graphics and interaction task may be performed by a group of processes we shall call the **interaction (process) group**, one or more processes for graphics display and possibly separate processes for reading and processing user tracking data. Similarly the visualization extraction task may be performed by several processes, called the **computation (process) group,** possibly operating on multiple processors in parallel. We choose these groupings because processes in the interaction group all have the same 10 frames/second/0.1 second latency requirements, while the computation group has the 1/3rd of a second latency requirement. This process structure decouples display and computation, so that a slow computation does not slow down the display process, and the speed requirements of the display process do not limit the computation.

The interaction process group passes user commands to the computation process group, which triggers the computation of visualization extracts. These resulting extracts are passed back to the interaction process group for display. This basic architecture is outlined in figure 4

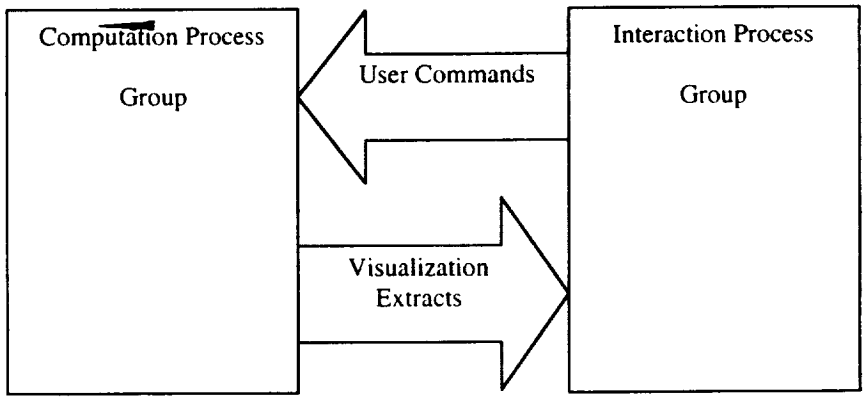

Figure 4: Runtime process architecture of an information visualization system for interactive and/or time-varying data

3. Distributed Implementation

Distributed data analysis can be highly desirable, particularly in one or more of the following circumstances:

- The data exists on a remote system and it is inconvenient to move the data to the local visualization system
- The desired visualization extracts require more computation than the local visualization system can easily deliver
- The data is the result of a concurrently running computation on a separate high-performance computational system
- Remote collaboration is desired, where several clients have (effective) access to the same data.

The use of separate, asynchronous computation and interaction process groups communicating via buffers as described in section 2.3 facilitates a distributed implementation, where the process groups exist on separate, possibly remote machines communicating over a network.

3.1. Distribution Strategies

There are several strategies for the distribution of data analysis. These strategies are determined by selection of where to place which operations in the data analysis pipeline of figure 1. These strategies and their advantages and disadvantages:

- **Remote Data, Local Extraction and Rendering:** In this option the data exists on a remote system, and individual data values are obtained over the network as required by the local visualization extraction algorithm. This strategy has the advantage of architectural simplicity, with all visualization activities taking place on the local system as if the data were local. This strategy has the disadvantage that it requires a network access each time data is required, which can be time consuming. There are

techniques to overcome this disadvantage, such as clever pre-fetching, where data is delivered in groupings which anticipate expected future queries. For some information types the remote data distribution strategy may be too slow for a VR interface, for example when analyzing continuous fields that arise in computational fluid dynamics, where large visualization extracts may require many data accesses for their computation. For other applications, however, where data accesses are relatively few, remote data distribution may be a viable strategy.

- **Remote Data and Extraction, Local Rendering:** With this strategy visualization extraction occurs on a remote system, typically the same system that contains the data. In an interactive system the extraction computations are in response to user commands passed from the user's local system. The results of the extraction, typically geometrical descriptions of three-dimensional visualization objects, are transmitted to the user's local system for rendering. Architecturally, this strategy maps closely to the runtime architecture illustrated in Figure 4, with the computation process group on (one or more) remote machines and the display and interaction process on the local system. This strategy has the advantage that the extractions algorithms are "close to the data", so data access is not a bottleneck. It also has the advantage of local rendering, which allows head-tracking display for each participant as required in virtual environments. The disadvantages of this strategy include the fact that response to user commands requires a round trip over the network, and the requirement that the user's local system be capable of rendering the extract's geometry.
- **Remote Data and Extraction, Distributed Rendering:** This strategy is a variation of the above Remote Data and Extraction, Local Rendering strategy. In this case the rendering commands occur on the remote system and are passed as distributed rendering calls to the user's local system for actual rendering. A local client program is still required to read the user's trackers and process and send user commands. This strategy has advantages and disadvantages similar to the Remote Data and Extraction, Local Rendering strategy, except that the network round-trip time is now part of the graphics display loop. This may introduce unacceptable delays into head-tracking responsiveness.
- **Remote Data, Extraction and Rendering:** This strategy places all the operations of the data analysis pipeline on the remote system(s), with the final rendered frames returned to the user's local system over the network. This strategy has the advantage that very powerful remote systems can be used

when the user has a very low-power local system. The disadvantage is that the rendered frames can be large, for example for a 1024x1024 24-bit RGB-alpha display requires a 4 megabytes frame buffer, and two such frame bufferss are required for stereo display. This implies an 80 megabyte transfer every second in our example to maintain the frame rate of 10 frames per second. This bandwidth requirement is beyond most available large-area networks. There are also serious issues of latency control in this strategy because the network time is part of the display responsiveness loop. There are, however, situations where the local system is incapable of the kinds of rendering desired and this strategy may be the only viable option. Direct volume rendering is an example when this strategy may provide the optimal choice.

The above strategies are not exclusive: one may have remote visualization extraction taking place on one remote system while the data resides on a third system.

3.2. Remote Collaboration Strategies

Once a system is distributed, the opportunity arises for remote collaboration, where two or more non-co-located users examine the same data together. Strategies for remote collaboration are related to, but different from distribution strategies. We briefly summarize the common remote collaboration strategies:

- **Distributed data:** This collaboration strategy places copies of the data to be examined on all participants' client systems. Collaboration is implemented by passing either user commands or computed visualization extract results among each of the participants' systems. The primary advantage of this strategy is that the software used is similar to stand-alone versions of the same systems. The main disadvantages include the difficulty of ensuring synchronization among the participants, and the requirement that each participant's system be capable of storing the data and computing the (at least locally generated) visualization extracts. Many distributed collaborative VE systems, such as military training systems, utilize the distributed data collaboration strategy.
- **Data Server:** This collaboration strategy builds upon the remote data distribution strategy. The data typically resides on a single remote system and is accessed by the participant's local system as needed. The visualization extracts are computed locally, and are communicated in the same manner as in the distributed data collaboration strategy above.

- **Visualization Extract Server:** This collaboration strategy builds upon the remote extraction distribution strategy, in which the visualization extracts are computed on a remote system, which is typically where the data being examined is stored. The extracts are sent to each participant's system for local rendering. The advantages of this strategy include:
  - As there is only one set of extracts associated with each set of data, synchronization is greatly simplified
  - Local rendering allows each participant to render the visualization extracts from a local point of view, as required for head-tracked displays
  - The extract server system can arbitrate conflicting user commands

  The visualization extract server collaboration strategy has the disadvantage of poor scaling to large numbers of users, though this problem will be alleviated when reliable multicast technologies become available. The other disadvantages of this strategy are the same as those for the remote extraction distribution strategy.
- **Scene Replication:** This collaboration strategy has a privileged user who's view is presented to the other participants. This collaboration strategy is similar to the remote rendering distribution strategy. This strategy has the same advantages and disadvantages as the remote data, extraction and rendering distribution strategy, with the added disadvantage that all participants will see the same view thereby precluding the use of head tracking for all participants.

4. Time Flow and Control

When visualizing time-varying data, a user may desire to control the flow of time, e.g. make time speed up, slow down or reverse. Furthermore, to view a particular phenomenon in more detail at a specific point in time, the user may stop the time flow. These non-real-world manipulations of the flow of time require several new ways of describing time in a virtual environment. The discussion in this section closely follows Bryson and Johan, 1996, where more details can be found.

4.1. Requirements of time-varying interactive visualization

There are several design principles that must be considered before developing time-varying interactive visualization applications:

- **Correct Simultaneity:** All visualization extracts displayed at the same time should have been computed for the same data time, unless explicitly directed otherwise by the user
- **Completeness:** All visualization extracts requested for a particular data time should be displayed unless the user directs otherwise.
- **Time Flow Control:** The user should be able to change the rate and direction of the flow of the data timesteps
- **Interactive exploration should be allowed even when the data timestep does not change:** If the user slows or stops the flow of time, interactive exploration should still be allowed. This implies that new visualization extracts may be computed even if the data timestep does not change.
- **Minimize delay:** The system should display the most recently computed visualization extracts, unless otherwise explicitly directed by the user.

4.2. Types of Time

In time-varying information visualization environments time assumes an unusual and subtle meaning. There are several senses of time that arise in such environments:

- **User time:** The time that the user experiences, the "wall-clock" time.
- **Data display time:** The displayed timestep, defined as the timestep of the data from which the currently visible visualization objects were extracted.
- **Data time:** The timestep from which visualization extracts have been computed. Data time may or may not be equal to data display time.
- **Frame time:** The number of times a set of visualization extracts have been computed for a given timestep. Frame time is measured by counting the number of times the computation process has been called for a given data time.

We distinguish between data display time and data time so that when data time is moving forward visualization extracts can be computed from the next data time while the current data display time is

presented to the user. The next data time may not be related to the current data display time: for example the user may choose a particular data time for the next display frame unrelated to the current data display time.

The purpose of the frame time counter is to support the scenario in which the advance of data time is stopped and the user wishes to interactively explore the current timestep's data. By stamping the visualization extracts with their frame time in addition to their data time the most recently computed extract may be displayed, maximizing interaction responsiveness. The use of frame time is discussed in the next section on buffering strategies.

4.3. Buffering strategies

In the runtime architecture described in section 2.3 the computation process group produces visualization extracts which are displayed to the user by the interaction process group. The method of communication between these groups is driven by the principles outlined in section 4.1 and is implemented using the time parameters defined in section 4.2.

The most difficult case is when the information visualization system supports several independent visualization objects, such as streamlines and isosurfaces, which can be independently specified and displayed. We shall therefore assume this is the case. We further assume that the graphics process keep a list of visualization extracts to be displayed. It is desirable that visualization extracts that do not change be displayed without recomputation. The buffers that pass the visualization extracts from the computation process to the graphics process should be capable of passing only those extracts that are newly computed, without destroying those older extracts that should continue to be displayed. This requirement is most easily met if the buffering takes place at the visualization object level, where each visualization object manages its own buffer.

There are two approaches to buffering, **local time** and **time-cached** buffering. Local-time buffering does not retain visualization extracts over time, while time-cached buffering saves visualization extracts for possible reuse when appropriate.

4.3.1. Local Time Buffering

In local time buffering, only the most recently computed visualization extracts are displayed and there is no attempt to cache visualization objects from previous timesteps. This approach to buffering is appropriate to non-time-varying data, because new visualization techniques are generated only in response to user commands.

Local-time buffering should be implemented so that

- The most recently computed extracts are available
- The computation process does not have to wait while a visualization extract is displayed.

Conventional double-buffering techniques, where one has a *write buffer* into which computed visualization extracts are stored and a *read buffer* out of which previously computed extracts are displayed only partially meet these requirements. Consider the case where the computation process is faster than the graphics process. This will often happen in non-time-varying environments where perhaps one visualization extract is being computed in response to a user command, while all current visualization extracts need to be displayed. When using a double-buffer approach and the computation process is faster than the graphics process, the write buffer will be filled before the graphics process' read buffer has been used and released for writing. This will cause the computation process to wait until the read buffer has been released. Such waiting will result in a delay if the user is "sweeping" a visualization object in space, producing a continuous stream of visualization extracts to be computed.

A better approach is to use a triple-buffer strategy, where three buffers are available. During typical operation, the computation process will fill one or more of these buffers, so (after the buffers have been filled) there may be two or more buffers marked "read". Each of these buffers will be marked with the time frame number defined in section 4.2, as well as the data time number if the data is time varying. In order to minimize delay the graphics process will choose the buffer with the highest time frame number, which will be the visualization extract most recently computed. In the meantime, if the computation process wants to proceed it can compute the response to the most recent user commands, overwriting the older of the two

available read buffers thus minimizing latency. In this way a triple-buffer strategy allows local-time buffering to meet the requirement to minimize delay.

In a time-varying data environment, the buffers will also be stamped with the timestep in which the extract was computed. The graphics process then selects buffers from the timestep being currently displayed in addition to selection of the highest data frame time. This assures that the requirement of correct simultaneity of section 4.1 is met.

4.3.2. Time-Cached Buffering

In a time-varying data environment one may wish to cache previously computed visualization extracts. When this is done, if the timestep from which those visualization extracts were computed is encountered again those extracts can be displayed without recomputation.

Time-cached buffering requires local-time buffering within each time step. When the timestep advances the most recent visualization extracts for that timestep can be stored. Then, when that timestep is reselected, either by user command or because the time flow is periodic, those extracts can be displayed without recomputation if their specifying data (e.g. seed positions for streamlines or values for isosurfaces) is still valid.

5. Time-Critical Techniques

One of the prime requirements of virtual environments is responsiveness. In section 2.2 we discussed the performance requirements for various aspects of an information visualization application within a virtual environment. These requirements can be difficult to meet in light of the possibly complex graphics and extensive computation required for the computation of visualization extractions. One is often faced with a conflict between the requirements of a complete or accurate visualization and the requirements for responsiveness and fast graphical display rates. While accuracy is often critical in an information visualization environment, users often prefer fast response with a known degradation in accuracy for purposes of exploration. When a phenomenon of interest is found in the more responsive but less accurate

mode, the user can request that this phenomenon be recomputed and displayed more slowly with higher accuracy. The automatic resolution of the conflict between accuracy and responsiveness, finding the appropriate balance, is known as "time-critical design", the topic of this section.

5.1. The Time-Critical Philosophy

Time-critical design attempts to automate the process of finding a balance between required accuracy and responsiveness. This approach is very different from real-time programming, which guarantees a particular result in a specified time. Real-time programming typically operates in a fixed, highly constrained environment while time-critical programs are typically highly variable. This variability is particularly evident in an information visualization environment, where the data and extracts computed and displayed may vary widely within a single user session. Time-critical design does not guarantee a particular result, instead delivering the "best" result possible within a given time constraint. A successfully designed time-critical system will provide a graceful degradation of quality or accuracy as the time constraint becomes more difficult to meet.

Time critical design for a particular aspect of a program begins with defining a cost and benefit metric for the task to be completed. The task is then parameterized in a way that controls both costs and benefits. When the cost and benefit of a task is known as a function of the task's parameters before that task is performed, the appropriate choice of parameters is selected to maximize the benefit/cost ratio. There are often many tasks to be performed in an environment, and the benefit of a particular task can be a function of the state of that environment. The solution of this problem is often approached as a high-dimensional constrained optimization problem, maximizing the total benefit/cost ratio for the sum of the tasks to be performed given the constraint of the total time allowed for all tasks.

As we shall see below, however, it is often very difficult to know the benefit and cost of a task before that task is performed. In such situations, hints provided by the user or simple principles such as assuming equal benefit within a set of tasks are often used.

5.2. Time-Critical Graphics

Time-critical techniques were pioneered in computer graphics (Funkhouser and Sequin, 1993), where objects were drawn with higher or lower quality depending on such benefit metrics as position in the field of view and distance from the user. Such implementations often used multiple representations of an object at varying levels of detail. In information visualization, however, many visualization extracts are already in a minimal form such as streamlines defined as a set of points. There are opportunities for graphical simplification in information visualization, however. For example, It may be the case that many more primitive elements are used to define a surface than is necessary for its display. An isosurface may have regions that are close to flat but the algorithm used to derive the isosurface may create many surface elements in that flat region. Display of that surface would be faster if that flat region were represented by fewer surface elements (see, e.g., Chen et. al., 1999 and subsequent papers in that volume). A surface may also be represented in simplified form until it became the focus of attention so that small variations from flatness may be important. Unfortunately, algorithms that identify such opportunities for surface simplification are computationally intensive and may therefore be unsuited to re-computation in every frame.

From similar considerations we conclude that unlike general computer graphics based on pre-computed polygonal models, the use of time-critical graphics in information visualization will be highly dependent on the domain-dependent specifics of the visualization extracts. It may be very difficult, for example, to assign a benefit to a particular extract, especially when that extract may extend to many regions of the user's view. While simple benefit metrics such as the location of the extract on the screen may be helpful, one should keep in mind that the user's head may be pointed in one direction while the user's eyes may be scanning the entire view. Such scanning is to be expected in an information visualization environment where the scene may contain many related, extended objects.

From these considerations there are few generalizations that can be drawn about the use of time-critical graphics techniques in information visualization. Opportunities do arise, however. Simple examples of time-critical graphics techniques that may be useful in information visualization include:

- Simplified representations, such as wireframe rather than polygonal rendering
- Surfaces represented as two-dimensional arrays, where simplified versions of the surface may be obtained by rendering every n points in the array
- Techniques that have been developed for time-critical direct volume rendering (e.g. Wan et. al. 1999)

A more general approach to time-critical graphics is to use multi-resolution representations of the data or the resulting visualization extracts. This is a new area of research, however, with relatively few results at the time of this writing.

5.3. Time-Critical Computation

Computation of visualization extracts can provide several opportunities for time-critical design (Bryson and Johan, 1996) because such computation is often the most time-consuming aspect of a visualization system. As in the case of time-critical graphics, the specifics of time-critical computational design will be highly dependent on the nature of the extract computed. We can, however, make several general observations:

- Both the cost and benefit of a visualization extract can be very difficult to estimate *a priori* based on its specification and control parameters, especially since the extent of an extract is difficult to predict based on its specification alone
- The cost of an extract can be roughly defined as the time required for its computation. Experience has shown that this cost does not vary widely between successive computations
- The cost of a visualization extract may be most easily controlled by parameterizing the extent or resolution of the computation. Techniques that begin their computation at a particular location in space, such as a streamline emanating from a point in space, lend themselves well to controlling their extent in space, which controls the time required by their computation. The cose of visualization techniques which rely on abstract or indirect specification is more effectively controlled by varying resolution.
- Other ways to control the cost of the visualization extract include choice of computational algorithm and error metrics for adaptive algorithms. These control parameters have a more discrete nature and may be set by the user or set automatically via specific trigger criteria.

Given that the benefit of a visualization extract is hard to predict, one may treat all extracts as having equal benefit unless specified by the user. In combination with the observation that costs do not change dramatically in successive computations, this allows the following simple solution to the problem of choosing the control parameters. For simplicity we consider the situation in which all of the visualization extract's costs are controlled by limiting their extent. Here, each extract computation is assigned a time budget, and each extract's computation proceeds until its time budget is used up. Then the time taken to compute all extracts is compared to the overall time constraint. Each extract's time budget is divided by a scale factor determined by the total actual time divided by the total time constraint. This scale factor may have to take into account any parallel execution of the extract computations. . If the time required to compute all the extracts is greater than the time constraint, this will result in smaller visualization extracts which will take less time to compute. If the extracts become too small, a faster but less accurate computational algorithm may be chosen. If the time required to compute all extracts is smaller than the time constraint, the time budget of each extract is increased, resulting in larger visualization extracts. A similar approach may be used to choose the resolution with which visualization extracts may be computed.

It may be evident from the this discussion that the types of control parameters one may use in time critical design will be highly dependent on the nature of the extract and how it is computed. Clever approaches can significantly enhance the time-critical aspects of a visualization technique. As an example, consider isosurfaces of a three-dimensional scalar field: these isosurfaces are traditionally specified by selecting a value, resulting in a surface showing where that value is attained in the scalar field. Controlling the cost of a traditional isosurface by limiting the time of the computation will have unpredictable results: in the conventional marching cubes algorithm for computing isosurfaces the entire dataset is traversed. If the marching cubes algorithm is stopped before completion and before regions of the field where the isovalue is attained are traversed, no isosurface will appear at all. Controlling the cost of the marching cubes algorithm by controlling the resolution with which the dataset is traversed is possible, but this strategy does not provide fine control and may result in a significant degradation in quality. A different approach to isosurfaces, *local isosurfaces* (Meyer and Globus, 1993), directly addresses this problem. Rather than

choosing an isovalue. for a local isosurface the user chooses a point in the dataset, from which the isovalue is determined as the value of the scalar field at that point. The isosurface is then computed (via a variation on the marching cubes algorithm) so that it eminates from that point in space, and is spatially local to the user's selection. The cost of a local isosurface in controlled by computing the isosurface until that isosurface's time budget has been used up. Two examples of local isosurfaes can be seen in figure 3.

6. Case Study: The Virtual Windtunnel

The Virtual Windtunnel (VWT) (Bryson and Levit 1991, Bryson et. al. 1997) is a virtual environment for the analysis of time-varying fluid flow data sets that are the results of computational fluid dynamics computations. VWT supports distributed collaborative operation, as well as several types of data analysis tools (see figure 3). In this section we shall outline the implementation strategy of VWT.

VWT is designed to address the case of interactive, time-varying data from section 2.1, and therefore uses the system architecture described in section 2.3. There are several processes in the display and interaction process group, including a graphics process, and multiple processes to read user tracking. The computation process group uses multiple processes to perform computations in parallel when multiple processors are available. VWT represents the visualizations as object classes, and maintains lists of object instantiations for both computation and display. For distributed operation the remote data and extraction, local rendering strategy of section 3.1 is used, in combination with the visualization extract server collaboration strategy of section 3.2.

Time flow control fully implements the discussion in section 4 using time-local buffering with the triple-buffering technique (Bryson and Johan, 1996). Each visualization object contains the triple buffer structure in the object definition, and each object manages its own buffer. Time critical computation is implemented for several of the visualization techniques using the simple scaling strategy described in section 5.3. When the data is not time varying, the computation of streamlines and local isosurface extracts is reentrant, so in subsequent time frames these extracts may be enlarged if their defining data has not changed. Time-critical computation of the streamline extracts implements a choice of several computational algorithms,

listed here in order of increasing accuracy and decreasing speed: one-step euler integration, 2nd -order Runge Kutta, and adaptive 4th-order Runge-Kutta. Which algorithm is used is automatically chosen based on the size of the streamline relative to a user-controlled *preferred size*. When the streamline falls significantly below the preferred size a faster but less accurate streamline computation algorithm is chosen. Details can be found in Bryson, 1998.

VWT has been tested in distributed collaborative mode with as many as four participants, two in Washington, DC and two in California (a distance of 3900 km), with the data and visualization extract server in California. Communications in this test were over a conventional network. When examining a non-time-varying data set the performance requirements of 0.1 second latency were maintained. Time-varying data sets also maintained the latency performance requirement for a small number of visualization extracts, however as more and larger extracts were used (e.g. large isosurfaces) network bandwidth constraints became significant.

7. Challenges

While several information visualization systems have been implemented in a virtual environment setting, there are several challenges that have been encountered which at the time of this writing (2000) do not have satisfactory solutions:

- **Large data sets:** Many information visualization applications involve very large data sets. Examples from computational fluid dynamics include time-varying datasets hundreds of gigabytes in size. Examples from the NASA Earth Observing System are expected to generate a terabyte of data a day. Accessing this data within the performance constraints of virtual reality is beyond the reach of current technology as of 2000. Higher bandwidth and lower latency access to mass storage devices will help, but the tendency in the data-generating community is to use higher computational power to generate higher resolution and therefore lager data sets. Techniques to pre-extract interesting features from such data sets hold the greatest promise as a solution to this problem.
- **Representation of abstract data:** Most information visualization systems that have been successfully implemented in virtual environments are in domains which map closely to the real world, such as fluid

flows or geophysics. Other domains involve the use of more abstract data. Appropriate mappings of abstract data into a three-dimensional virtual environment is an area of ongoing research.

- **High-dimensional data:** Most information visualization systems that have been successfully implemented in virtual environments have data that is defined in three-dimensional space. Many information domains have data defined on a higher dimensional domain. Visualizing this data in two dimensions has been an ongoing area of research. We expect that visualizing this data in the inherently three-dimensional environment of virtual reality would be of benefit. Mapping these higher dimensions into the three dimensions of virtual reality is a problem that has yet to be thoroughly explored.
- **Time-critical distributed design for distributed environments:** Time-critical design for distributed virtual environments, where the network traffic and latency times must be considered, is an unexplored area. This challenge becomes more complex in the context of distributed computation, where different extracts are computed on separate systems.

8. Conclusions

Information visualization is an active and fruitful area of virtual environment research and development. The advantages of information visualization in virtual environments include:

- Three-dimensional display
- Three-dimensional user interaction
- An intuitive interface that facilitates the exploration of complex data

In order to implement an effective information visualization application in a virtual environment, however, issues of responsiveness and fast updates must be addressed. These issues may be resolved via use of appropriate system architectures, design based on human factors issues, appropriate time control for time-varying data, implementation of time-critical techniques whenever possible, and appropriate choices for distributed implementations. The details of how these solutions are implemented will be highly dependent on the target domain and the specifics of the visualization techniques used.

9. References

Bryson, S. (1996). Virtual Environments in Scientific Visualization Communications of the

ACM, 39(5) 62-71

Bryson, S. (1998). Time-Critical Computational Algorithms for Particle Advection in Flow Visualization in Late Breaking Hot Topics Proceedings, IEEE Visualization '98 (pp. 21-24) Research Triangle Park, NC

Bryson, S. and Levit, C. (1991). The Virtual Wind Tunnel: An Environment for the Exploration of Three Dimensional Unsteady Flows in Proceedings of IEEE Visualization '91 (pp 17–24). San Diego, CA: IEEE Press.

Bryson, S. and Johan, S. (1996). Time Management, Simultaneity and Time-Critical Computation in Interactive Unsteady Visualization Environments in Proceedings of IEEE Visualization '96 (pp. 255-261). San Francisco, CA: IEEE Press.

Bryson, S., Johan, S. and Schlecht, L. (1997). An Extensible Interactive Framework for the Virtual Windtunnel in Proceedings of the Virtual Reality Annual International Symposium '97 (pp 106–113). Albuquerque, NM: IEEE Press

Chen, B., Swan, J. E. III, Kuo, E., and Kaufman, A. (1999) LOD-Sprite Techniques for Accelerated Terrain Rendering in Proceedings of IEEE Visualization '99 (pp. 291-298) San Francisco, CA: ACM Press.

Cruz-Neira, C., Leigh, J., Barnes, C., Cohen, S., Das, S., Englemann, R., Hudson, R., Papka, M., Siegel, L., Vasilakis, C., Sandin, D. J., and DeFanti T. A. (1993). 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 (pp 59-66). San Jose, CA: IEEE Press

Funkhouser, T. A. and Sequin, C. H. (1993). Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments in Computer Graphics: Proceedings of SIGGRAPH 93 (pp. 247-254), Anaheim, CA: ACM Press

Herndon, K.P. and Meyer, T. (1994). 3D Widgets for Exploratory Scientific Visualization in Proceedings of User Interface Software Technology '94 (pp 69-70), Marina del Rey, CA: ACM press

Lin, C-R., Loftin, R. B., and Nelson, H. R. Jr. (2000) Interaction with Geoscience Data in an Immersive Environment in Proceedings of IEEE Virtual Reality 2000, New Brunswick, NJ: IEEE Press

Meyer, T. and Globus, A. (1993). Direct Manipulation of Isosurfaces and Cutting Planes in Virtual Environments, (RNR Technical Report RNR-93-019). Moffett Field, CA: NASA Ames Research Center

Sheridan, T. B. and Ferrill, W. R. (1974). Man Machine Systems Cambridge, MA: MIT Press

Song, D. and Norman, M. L. (1993) Cosmic Explorer: A Virtual Reality Environment for Exploring Cosmic Data in Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality, San Jose; CA: IEEE Press

Taylor, R. M., Robinett, W., Chi, V. L., Brooks, F. P. Jr., and Wright, W. (1993), The Nanomanipulator: A Virtual Reality Interface for a Scanning Tunnelling Microscope, in Computer Graphics: Proceedings of SIGGRAPH '93 (pp 127-134), Anaheim, CA: ACM Press

Wan, M., Kaufman, A., and Bryson, S. (1999). High-Performance Presence-Accelerated Ray-Casting in Proceedings of IEEE Visualization '99 (pp. 379-386) San Francisco, CA: ACM Press.

Keywords: Virtual Reality, Virtual Environments, Information Visualization, Scientific Visualization, Visual Data Analysis, Interactive Data Analysis, Time-Critical techniques, Time-Critical Computation, Distributed Virtual Environments, Collaborative Virtual Environments, Time Control, Interactive Analysis of Time-Varying Data Sets