Science at the Speed of Thought

J. E. Devaney[1], S. G. Satterfield[1], J. G. Hagedorn[1], J. T. Kelso[1], A. P. Peskin[1], W. L. George[1], T. J. Griffin[1], H. K. Hung[1], and R. D. Kriz[2]

[1] National Institute of Standards and Technology, Gaithersburg, Maryland USA
[2] Virginia Tech, Blacksburg, Virginia USA

{judy.devaney, steve.satterfield, john.hagedorn, john.kelso, william.george, terence.griffin, howard.hung}@nist.gov, peskin@boulder.nist.gov, rkriz@vt.edu

http://math.nist.gov/mcsd/savg

Abstract. In this paper we describe a virtual laboratory that is designed to accelerate scientific exploration and discovery by minimizing the time between the generation of a scientific hypothesis and the test of that idea, enabling science at the speed of thought. This laboratory ties together computational experiments, laboratory experiments, and analysis tools in an open source immersive visualization environment by means of a loosely coupled distributed computing environment. We use this framework to provide scientists access to new representations of and interactions with their data through our image analysis, visualization, machine learning, and data mining tools as well as access to their traditional analysis tools. The design for our collaboration mechanism enables multiple people from geographically distributed locations to join and leave the environment at will, making distance irrelevant. We detail these components and our tools and present some real world examples drawn from a variety of scientific applications.

1 Introduction

Scientific discoveries occur with iterations of theory, experiment, and analysis. But the methods that scientists use to go about their work are changing [1].

Experiment types are changing. Increasingly, experiment means computational experiment [2], as computers increase in speed, memory, and parallel processing capability. Laboratory experiments are becoming parallel as combinatorial experiments become more common.

Acquired datasets are changing. Both computer and laboratory experiments can produce large quantities of data where the time to analyze data can exceed the time to generate it. Data from experiments can come in surges where the analysis of each set determines the direction of the next experiments. The data generated by experiments may also be non-intuitive. For example, nanoscience is the study of materials whose properties may change greatly as their size is reduced [3]. Thus analyses may benefit from new ways to examine and interact with data.

Two factors will accelerate these trends and result in increasing volumes of data:

– CPU speedup: as companies strive to keep Moore's law [4, 5] in effect

– Computer architecture speedup: as all computers benefit from architecture advances in high end computers.

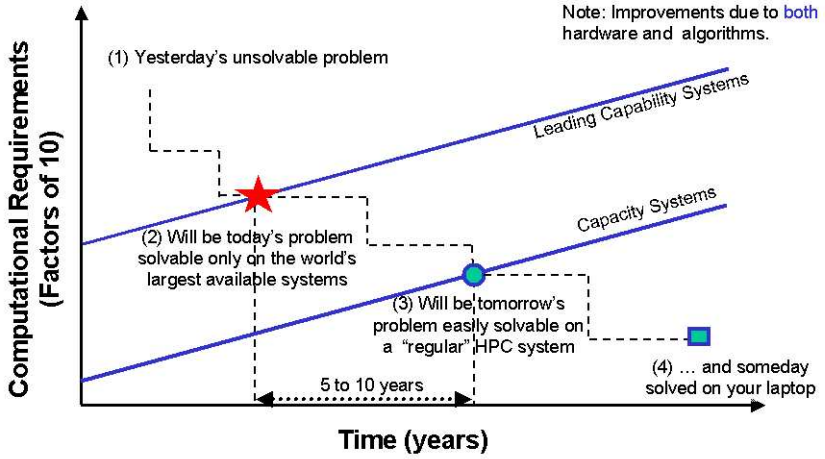


Fig. 1. Transition of solving important problems from unsolvable to solvable on your laptop

computers ever more capable and increase the move to computational experiments and automation.

But a third factor offers a partial solution: graphics speedup. Computer game enthusiasts are funding a fast pace of development of graphics processing units (GPUs) [7, 8]. The use of these GPUs in the support of science makes the future world increasingly computational, visual, and interactive.

We believe that representation and interaction drive discovery, and that bringing the experiments (computer and laboratory) of science into an interactive, immersive, and collaborative environment provides opportunities for speed and synergy. Adding traditional data analysis, machine learning, and data mining tools, with multiple representations and interactions can speed up the rate of exploration and lead to new insights and discovery. Creating an environment that is efficient, general and flexible enough to work well across a wide variety of scientific applications is at the heart of our Virtual Laboratory (VL) design.

In section 2, we describe the VL we are building at NIST to address these issues. In section 3, we describe some applications. We present conclusions and future work in section 4.

2 The Virtual Laboratory

The VL needs to be efficient, general, and flexible, but it also needs to be able to get applications into it quickly in order to speed up the process of science and not burden it. Representations and interactions of many types need to be available and easily accessed. To accomplish this, our design consists of the following components:

- A distributed computing environment that provides the communication fabric of the VL,
- An immersive visualization environment that provides representation, interaction, and collaboration capability in the VL,
- A suite of tools for machine learning and analysis.

We will discuss each of these in turn.

2.1 Distributed Computing Environment

An important part of the VL is the capability of users to interact with their data sources, analysis programs, and their experiments, either computer experiments or laboratory experiments, from any of the supported VL access points (see Figure 2 for a schematic). Examples of access points are the immersive visualization system or a remotely connected workstation. For the purposes of this discussion, data sources and analysis programs will be subsumed under computer experiment. The interactions range from viewing the status of their currently running experiment, or the results of a prior experiment, to providing feedback to the experiment in order to alter or restart the experiment. To provide this range of access requires a framework that enables communication between the user, the experiment, the visualization, and possibly with other collaborators actively interacting with the experiment.

The VL uses what is generically referred to as a coordination framework to provide a loose coupling between all participating entities of an experiment. The coupling is loose in that participants, that is, laboratory experiment equipment, computer simulations, visualization systems, user interfaces, and any other connection to the VL, can choose whether or not to utilize this coupling and can connect and disconnect from it at any time without disrupting the system. The VL distributed computing framework is implemented using the Java technology of Jini and JavaSpaces [9] [10]. Both of these packages, Jini and JavaSpaces, are available as pure Java implementations and so are portable to any system that supports a Java virtual machine, which includes systems running Linux, Microsoft Windows, and most Unix based operating systems.

Jini provides support for a form of distributed computing that explicitly handles common issues that arise in a distributed (networked) system, such as intermittent network outages and server crashes. Support includes automatic discovery of services available on the network, active leasing of services to help maintain current service information and to purge services that no longer exist, and distribution of events to remote applications to allow applications to communicate asynchronously and to react to expected or unexpected developments.

JavaSpaces is a specification for a Jini service that provides a coordination framework in the form of a tuple space. The concept of a tuple space was first introduced in

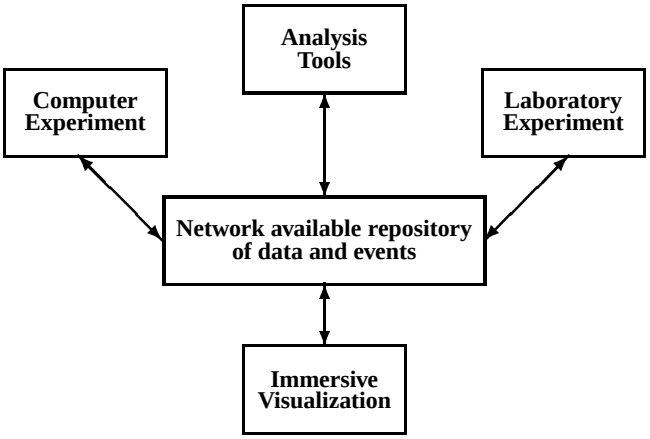


Fig. 2. Schematic of Distributed Computing Environment. Each component may also access other data sources

the early 1980s by computer scientists David Gelernter and Nick Carriero, within the context of the programming language Linda [11] [12]. This is referred to as a coordination framework since it allows a loose collection of applications, linked over a common network, to communicate asynchronously. This communication is so loosely coupled that the applications do not need to be running, or even exist, at the same time. To take the possibilities to an extreme, application A can send a message that is ultimately to be read by application B (and possibly others) which has not yet been written and will run on a machine that has not yet been built. When application B receives the message, application A, and the machine it ran on, may no longer exist.

There are many ways to describe the purpose and use of a JavaSpace. The concept of a coordination framework is a good high-level description. At a lower level, a JavaSpace can be thought of simply as a shared memory space, accessible from any machine on the network and addressed using an associative lookup rather than by memory address. Objects stored in a JavaSpace are instances of classes in Java that have a few special characteristics needed to support storage and retrieval from the JavaSpace. So the associative lookup uses the Java class type system to identify objects to be written or read from the JavaSpace. This is a very robust addressing scheme, compared to using raw memory addressing or simple string matching, since it ensures that you receive an object of the correct type when you read from the space. These objects can store any type of information that might be needed by the applications. It is also possible to maintain structures of objects in a JavaSpace, such as a linked list of objects, a tree of objects, or an array of objects. Objects can also simply be markers, holding absolutely no data, but giving information simply by their existence or non-existence in the JavaSpace.

In the VL, a JavaSpace is maintained to allow for the coordination of experiments, visualizations, and interacting collaborators accessing the VL through their workstation or other supported interface device. Typical information stored in the VL JavaSpace

includes experiment parameters, current status of an experiment such at the latest time-step of a computer simulation, or latest result from a laboratory experiment.

Of course, applications need not be written in Java to participate in the VL. In fact, it is expected that most computer simulations will likely be written in Fortran or C/C++, and most participating laboratory devices will likely not have direct network access. In Jini, applications and devices that are not capable of participating directly, either because of insufficient resources to run a Java virtual machine, or because they are closed systems, can still participate through the use of a surrogate [13]. These surrogates allow the application or device to participate by performing as a communications gateway to the Jini/JavaSpace network and also performing any computation needed in the process. The communication between the surrogate and the Jini/JavaSpace network uses the standard Jini/JavaSpaces protocols. The communication between the surrogate and the application or device uses an appropriate private protocol, unique to each application or device.

Although a JavaSpace is not expected to store objects that include large volumes of data, such as Gigabyte output and input files typically used in large experiments, it can be used to coordinate access to such files. An object in the space can represent such a file, giving status information about the file such as size and time written and allowing for the retrieval of the file contents over the network. A Remote File service (RFS) has been developed to handle this need in the VL. Applications can read and write large files within the VL if they need them to be available, normally for visualization and for feedback in the case of experiment input files.

The looseness of the coordination framework, together with the use of JavaSpaces, Java and Jini provides the communication fabric of the VL robustness and generality. Implementing this way means it is easy to include things. Thus it also provides speed of implementation.

This Jini/JavaSpace based remote file service was developed as part of a related project, Screen Saver Science (SSS) [14], which provides a distributed computing environment for general scientific computing. In SSS, applications utilize otherwise idle processors to complete compute-intensive tasks within a distributed algorithm and use RFS for large remote file input and output. As with the participants in the VL, machines participating in SSS can join in the SSS system to compute tasks at any time and can leave at any time without disrupting the overall operation of SSS. Although this is called Screen Saver Science, there is actually no screen saver application within SSS, but screen savers are sometimes used to help determine when to join SSS and when to leave (other techniques are also used).

2.2 Immersive Visualization Environment

The second component of our VL is an immersive visualization (IV) environment (IVE). IVE is widely used across government, industry, and academia [15]. It is also increasingly used to great advantage in scientific visualization [16, 17, 1]

"By immersive virtual reality we mean technology that gives the user the psychophysical experience of being surrounded by a virtual, that is, computer-generated, environment. This experience is elicited with a combination of hardware, software, and interaction devices. Immersion is typically produced by a stereo 3D visual display,

which uses head tracking to create a human-centric rather than a computer-determined point of view [16]." Thus, users interact with "things rather than pictures of things [18]" because the three dimensional data models reside in the same physical space that the user occupies. This enables the user to move around and to view the objects from different angles, to understand physical relationships in an natural way, and to interact directly with the objects in the environment in much the same way that he or she would inspect an apparatus or a sample in the laboratory. Of course, in a virtual environment, the user is not limited to normal physical interactions. Objects in the immersive environment are completely mutable; the user can make objects transparent or invisible, change the scale of objects, move through objects at will, and so on.

IVE provides a true three-dimensional (3D) view of data. But IV provides more than another dimension. Because our natural habitat is 3D, we bring with us into the IV environment all our understanding of relationships in a 3D world. So we do not have to try to understand the relationships we see. We comprehend them naturally.

The IVE is also a scientific instrument like any other. It requires multiple types of calibration and specialized software, for example. But it is also a scientific instrument unlike any other. While it is a finite physical space defined by the physical setup, it is also an infinite virtual space where we can wander indefinitely. It is up to the visualization scientists to decide what to put in the space and how to arrange it. Things at very different scales or types can be placed side by side. For example, a nanostructure can be placed next to a plot of one of its properties. In fact, immersive visualization is where scientific visualization and information visualization [19] can be joined.

Our philosophy for the IVE is similar to the philosophy for the distributed computing environment. It is loosely coupled. In this context, we mean we do not have a single monolithic program but rather a collection of small programs that can be snapped together to create new programs. Each small program is good at doing one thing well. All are designed to work together.

There are three ways we join programs together:

- UNIX [20] pipes and filters for creation and transformation
- Dynamically Shared Objects (DSO's) for functionality
- scenegraph objects for simple placement.

This gives us the generality, flexibility, and speed that we need for the VL.

We have four main categories of tools:

1. IV Infrastructure software
2. IV Representation Software
3. IV Scene Interaction Software
4. IV Collaboration Software

We will discuss each of these in turn.

IV Infrastructure Software. This is software to get visualizations into the IVE and interact with the IVE hardware. We use as our base software the open source software DIVERSE [21] in our IV environment. It uses OpenGL Performer [22] to create a

scenegraph that places the objects in the immersive environment. That means that anything that can be loaded into Performer can be immediately loaded into the IVE. It also means that getting a data set into the IVE may require writing a simple filter to convert one file format to another. New file loaders can be written and we have created some for special needs.

IV Representation Software. In the context of an IVE, representation refers to the process of transforming raw data into a visual geometric format that can be viewed and manipulated in the IVE. Representation is an important component for learning. The key to enabling scientists to quickly and easily visualize their scientific data in an immersive environment is to develop useful tools to help convert raw data to immersive data. We have designed our system to use our own representations but also to take advantage of representations computed by other packages. For these tasks, we have created a Glyph Toolbox to easily create our own geometry, and we have built software to convert the output of other packages to a form that can be displayed in the IVE.

Glyph Toolbox: The Glyph Toolbox is a set of programs to create and manipulate three-dimensional objects in a format readable by a variety of visualization programs. Glyphs are visual symbols used to convey information based on appearance or position. Simple glyphs include bars on a bar chart and dots on a statistical plot. The Glyph Toolbox is a set of three-dimensional glyphs both complex and intuitive enough to convey information to a wide audience, while simple enough to generate geometry from scientific data.

The purpose of the Glyph Toolbox Project is to build a collection of individual Unix style command line programs. Each program accomplishes a single task to create or manipulate geometry that can be used to build a polygon based virtual environment. A series of individual UNIX style simple commands can be combined to create objects, and then scale, color, rotate, and/or translate them to a particular specification. The tools (command and filters) output an ASCII based file that is machine and rendering independent. The actual display of the ASCII files is handled by converting the output polygon file into a format suitable for display by a viewing program, such as DIVERSE/diversifly, VRML, Open Inventor [23], etc.

The Glyph Toolbox has a wide range of applications. It has been used to create traditional glyphs for displaying data, such as error bars, menu items, or logos. More complex examples include transforming molecular data, i.e. positions of atoms in a grid, into a three-dimensional display of a crystalline structure, as in Figure 3. Each atom is represented by a Glyph Toolbox sphere, scaled, colored, and put in its proper place in the crystal.

SAVG Format and Converting Other Package Output: Currently under development is another type of Performer-based file format, called the SAVG format (named after our research group, the Scientific Applications and Visualization Group). Unlike the Glyph Toolbox file format, which was developed to help create original objects for visualization, the SAVG format was initially developed as a conversion format to allow data from a variety of scientific modeling packages to be used in our immersive visualization environment. The SAVG format has been enhanced since its origin with features

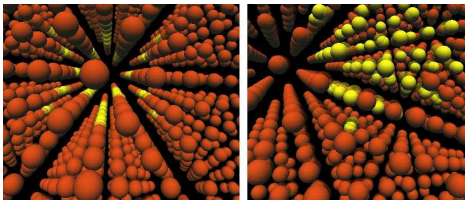


Fig. 3. A nano-structure of gold and copper atoms. Visualization is created with the Glyph Toolbox

to solve issues with transparency, lighting, and efficiency, and has grown into a very robust format for our virtual laboratory. Geometry can be defined using polygons, lines, points, and triangle strips. Polygons can be converted into their corresponding wire frame objects, or individually shrunk for better viewing. Examples of visualizations using the SAVG format are shown below in Figures 4 and 5, which demonstrate SAVG capabilities in transparency, as well as the shrinkage effect.

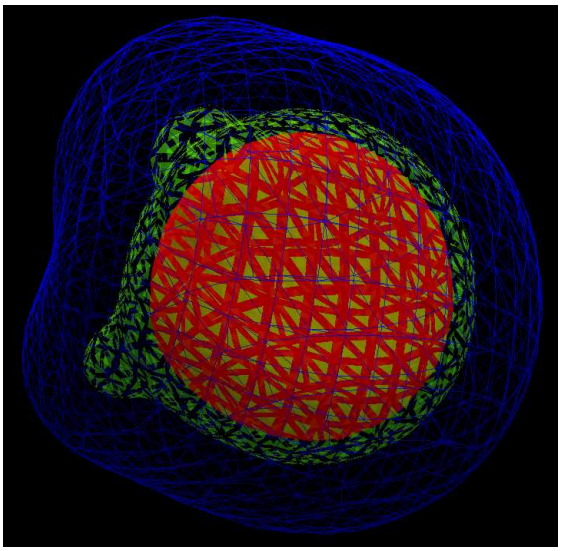


Fig. 4. Use of transparency and shrinkage in the SAVG file format to display isosurfaces of electron density of a water molecule

In order to visualize data as quickly as possible, it is important to use what has already been done. To this end we use other software to create a variety of representations. For example visualization packages such as openDX [24] and VTK [25] easily generate representations such as isosurfaces. Output from these packages is converted

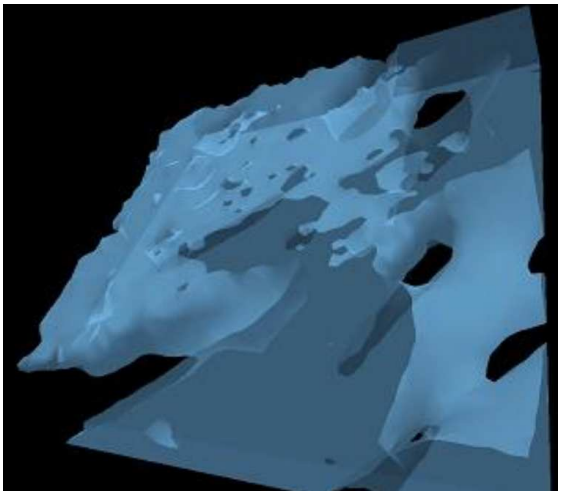


Fig. 5. Use of transparency to display polymer scaffolding

to our own format and loaded into the IVE via DIVERSE. See Figure 6 for an example of a set of isosurfaces visualized this way.

We can get data into the IVE quickly, once we have created converters from the format of outside software packages to a format that our environment recognizes. But the software packages providing the data need not be visualization packages. Representation is key to learning and any way to convert a data set to another representation may increase our insight. See section 2.3 on Machine Learning.

IV Scene Interaction Software. For efficient scientific exploration, it is important to have user interaction that is both easy to set up and easy to modify. It is also important that the software work on a wide variety of platforms.

A variety of different graphic displays and input devices can be used with the DIVERSE graphics interface by the use of separately compiled and loaded Dynamic Software Objects (DSOs). By plugging in one of many DSOs for graphics display, for example, the same visualization programs can be used on desktop, laptop, or full three-dimensional immersive systems.

Individual DSOs are also used to add new functionality to our visualization software system. Our group has developed a wide range of DSOs that allow a user to interact with the objects he or she is viewing. This includes functionality to move objects around, select individual objects or sets of objects and assign functionality to the selected objects, interact with outside software programs, bringing data into the system and sending data out of the system, and loading or unloading objects during a visualization.

Individual DSOs can be loaded to add simple functionality to a scenegraph, such as adding a particular light source or an object to represent a pointer for the user to select objects. Our **wandPointer** DSO for example, is loaded with the **objectMover**

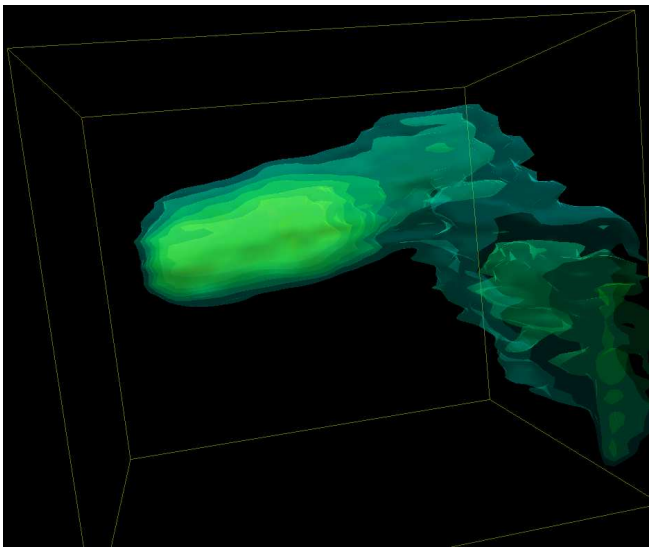


Fig. 6. Bayesian reconstruction of a circuit with an electromigration void. Visualization is via nested transparent isosurfaces

DSO, as shown in Figure 7, to allow a user to select an object and then move it with the movement of the wand.

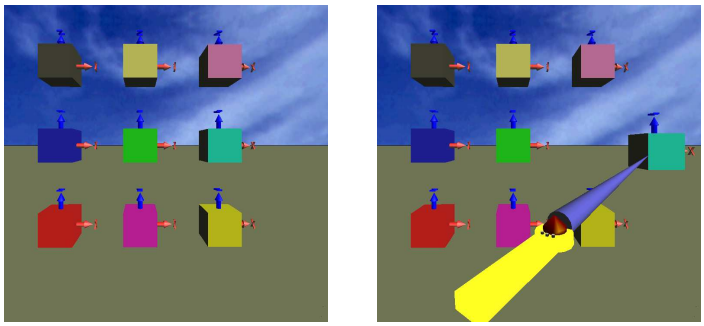


Fig. 7. Demonstration of the **objectMover** and **wandPointer** DSOs

Another set of DSOs that allow a user to take a look inside an object is particularly useful for viewing large sets of volume data. Figure 8 below demonstrates the **clipWand** DSO. Through the use of shared memory, a user can enter shell script commands to an accompanying program, `hev-clipwand`, to specify up to 6 different clipping planes that are defined with respect to the direction and position of the wand. The wand is a

hand-held device and its position is continually tracked by this system. This enables the user to interactively position the clipping planes in real time in order to reveal the interior detail of three-dimensional data sets. These clipping planes can be turned on and off interactively during a visualization. Volume data can also be manipulated with DSOs that edit lookup tables, making only objects within certain color ranges visible, for example.

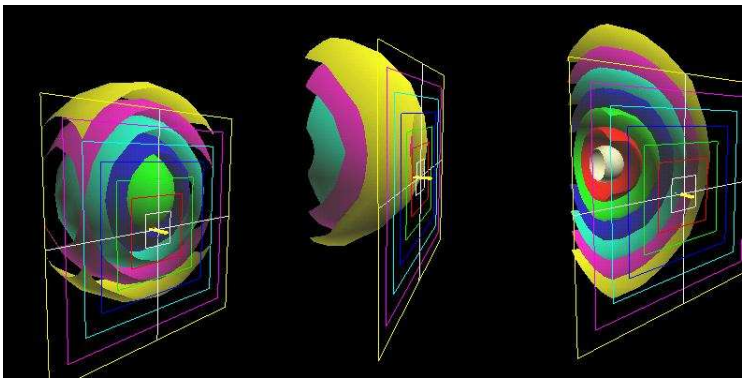


Fig. 8. Layered spheres demonstrate the capabilities of the **clipWand** DSO

A DSO known as **objSwitchExec** has evolved to give functionality to the objects of a scenegraph. When selected, an object can be assigned the function of turning on or off another object or itself, or executing a shell command to interact with another DSO or an outside program. In this way, an internal menu structure can be assembled, as in Figure 9. Menu items can be selected which make objects visible, initiate time simulations, define clipping planes, etc. Different DSOs can be loaded to allow a user to select objects differently depending on whether they are using desktop mode on are in a three-dimensional immersive environment. For example, the **desktopWandHPR** DSO can be loaded to use simple keyboard controls for object selection at the desktop.

The objects themselves can be loaded or unloaded from a scenegraph during a visualization with the use of the DSO, **blink**. Using an accompanying program, a user can issue shell commands to load or unload objects, or make them visible or not visible in the scene. By individually loading pre-compiled dynamic objects, a user has access to a wide variety of combinations of functionality useful for desktop, laptop, and full immersive systems.

IV Collaboration Software. Scientists rarely work alone today. A glance at any current scientific journal shows the extent of collaboration as well as how geographically distributed scientific collaborators are. Scientists need to be able to enter and leave the VL at will and discuss results in the VL regardless of distance. Collaboration in virtual

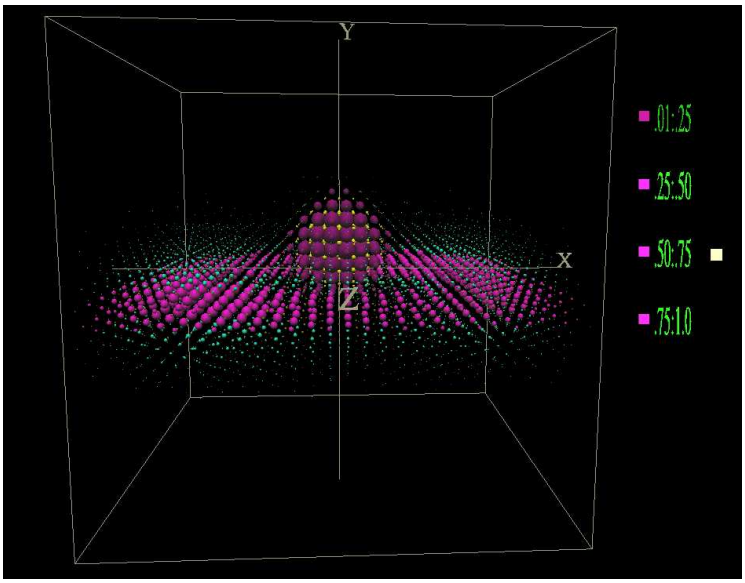


Fig. 9. Use of menuing objects to turn on and off visibility of data, using the **objSwitchExec** and **desktopWandHPR** DSOs

environments (VEs) occurs when spatially separated participants interact in the same VE.

A variety of collaborative VE's have been built both for specific applications and for more generic purposes. These systems include Crumbs [26–28], NICE [29, 30], LIMBO [31], Collaborative CAVE Console (CCC) [32, 33], and D Atomview [34]. In the future, it is hoped that participation in device-independent collaborative VEs will become as simple and commonplace as today's web browsers and networked multi-user games. Challenges remain in the areas of standardization, especially in the area of interface design, to minimize the learning curve for using different VEs, and provide support for a plethora of hardware configurations.

At NIST we are building a framework for collaborative VE applications which we hope will be a step towards this goal. Our work uses the DIVERSE VE library and a modified and extended version of the collaborative tools currently available with DIVERSE. DIVERSE was chosen because our group already has extensive experience with this package, and DIVERSE's open source license allows us to build on prior work on collaborative VEs. Our system will also be open-source, available for free download from our web site.

Our collaborative VE system will support both synchronous and asynchronous applications, using a central server to provide persistence. The server will keep track of the state of the VE, but not the application itself. Network communications will be based on the Jini/JavaSpace distributed computing infrastructure described earlier. Each participant will load a collaborative dynamically shared object (DSO) to communicate with

the server, and will load different sets of DSOs for the application based upon whether they are using a stereo head-tracked immersive CAVE, a workstation, or a standard desktop Linux system.

Our system will use SGI's OpenGL Performer as its graphics engine. Performer is designed to provide optimal graphics performance in a platform independent manner. It uses a scenegraph data structure to represent its graphical elements, easily allowing modeling transformations which affect an entire subtree. Since Performer and DIVERSE are written in C++, new classes have been built to provide a data-driven graphical capability. For example, a node in a scenegraph might contain a transformation matrix. A DIVERSE class can read shared memory every graphics frame and update the matrix in the node based on changes to the shared memory. In this way a simulation, or the collaborative server, can directly modify the VE by merely updating data in memory. In fact, the simulation needn't be written as a graphical collaboration. All it needs to do is update shared memory.

Our system is designed to use as little network bandwidth as possible. For example, the position of an object is given by six floating point numbers representing its location (X,Y,Z) and orientation (H, P, R Euler angles), and these six numbers are updated only if the object actually moves. Six numbers can also be used to specify the orientation of a node in a scenegraph (seven if a uniform scale factor is needed), so complex animations can be achieved with minimal data transfer.

Simulations also use shared memory as a data-passing mechanism, and multiple simulations can read and write between themselves and the VE. Participants in the VE will communicate with the simulations using shared memory too, so the difference between modifications from a participant and from a simulation is completely transparent.

Permissions and file loading/unloading will be handled using the Jini/JavaSpace distributed computing environment. This system will keep track of who is allowed to do what, transfer and update data as needed, and handle problems of resource allocations and conflicts. This functionality will be incorporated into a DSO that will work in conjunction with the central server.

Additional DSOs will implement awareness tools that reflect the status of the collaborative VE. For example, one DSO can display the relative position of all participants in the VE on a two-dimensional radar display. DSOs can be written to work best in an immersive environment or on a desktop, and users can choose the DSOs that best suit their needs.

2.3 Machine Learning Tools

Representation is critical to understanding and learning. This is true for immersive visualization, where productivity is dependent upon how well the data is represented in the 3D world. Representation is also important for machine learning. In that context, representation means the feature space of the data, and the formalism for describing the data [35].

Finding the appropriate representation is important, but more than one representation may be beneficial. A scientific paper uses multiple representations: equations, plots, images, tables, diagrams, charts, drawings, and text. A scientific presentation may also

contain movies or animations of various types. All of these are in support of communication of ideas. But these are also used by scientists as they work towards their own understanding.

With the VL, different representations of data can exist side by side. They can also be created interactively. Whatever aids understanding can be placed in the virtual space.

Very large data sets present a challenge, even for visualization. Data sets can be large in terms of the number of dimensions or the size of the dataset, or both. While there exist techniques for viewing high dimensional data sets, such as parallel coordinate plots [36], this may not be the best way to acquire knowledge from the data. A different approach is to look for those dimensions that are most relevant and discard the rest. One recent winner of the Knowledge Discovery in Databases competition reduced the number of features from the original 139,351 down to 4 features in his final model [37]. An alternate approach is summarizing. For example, data can be clustered and the individual components can be studied in various ways.

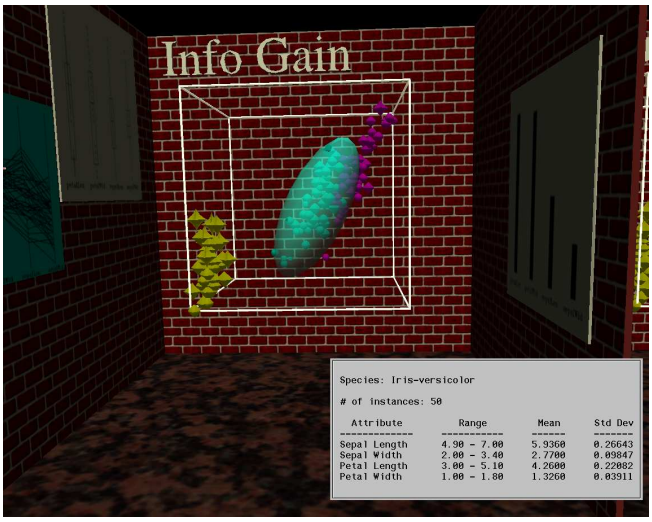
Machine learning tools can help with both dimension reduction and summarizing. We use the Weka [38] implementations of many machine learning algorithms. We also use the autoclass [39] clustering software. In addition we have developed our own genetic programming software package, GPP [40]. We also have our own equation discovery software [41]. This provides us with many avenues for displaying, interacting, and gaining insight into our results.

Our visualization of the Iris data set [42] contains multiple representations. Figure 10-a shows part of our visualization. On the near side of the left wall is a parallel coordinate plot [36] of the cluster identified with the transparent envelope. On the far side of the left wall is a plot of the probability density distribution of each of the attributes in the data set. The right wall shows how the attributes rank with Information Gain [38]. In the foreground is a set of statistics that have been computed on the fly in response to a user command. The points of the data set are represented as glyphs where the attributes have been mapped to glyph attributes using our glyph toolbox. The points are plotted in the central cube. A user can also interact with this visualization by turning the transparent envelops of the clusters on and off individually, and the parallel coordinate plots with them. Figure 10-b shows the same dataset visualized in three different ways, shown in three separate rooms.

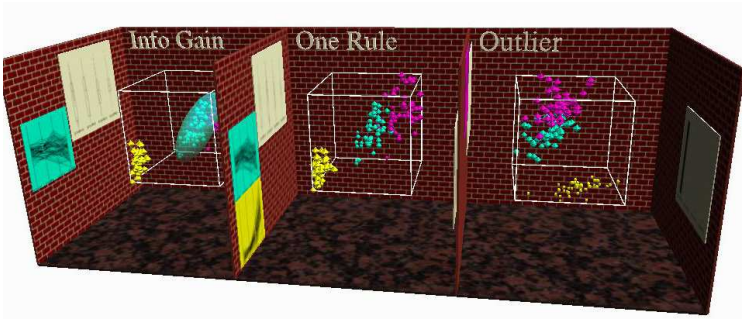
Figure 10 helps to bring together all of the main components of our VL. The visualization is run through a distributed computing environment, in which multiple users can interact with the data. The figure demonstrates the interactive IVE, in which users can move, hide, and select objects in the system to control the display the data and the movement of the data into and out of the visualization. Figure 10 also displays results of our machine learning tools, used to analyze the data and select which components to study. With all three of these components, we can speed up concept development.

3 Applications

We speed up insight into our data through representation of the data in the IVE and through interactions with the data. One representation may not be sufficient, so the



a.



b.

Fig. 10. An immersive visualization of the Iris data set. Image **a** shows one of the rooms of the multiple-view visualization shown in image **b**.

ability to switch between and interact with representations is important. We describe a set of applications that highlight our approach.

3.1 Multi-Modal Imaging and Visualization

In this project we are developing methods for combining related three-dimensional data sets from a variety of sources into visualizations that enable exploration and understanding of the data at a variety of scales and with a variety of techniques. The data is produced by OCM (Optical Coherence Microscopy) and CFM (Confocal Fluorescence Microscopy) techniques being developed at NIST.

These data are being used to investigate structural and functional properties of tissue engineering scaffolds. The scaffolds are porous polymers that act as support structures for the growth of cells. Our visualization techniques are being used in an effort to characterize properties such as mechanical performance and induction of tissue development.

The data are produced at differing scales and data from different instruments show different structures. Our task is to combine these data sets into coherent immersive visualizations. Challenges include registration, segmentation, and exposure of volume structures. We use several important visual and geometric representations of the data, including derived surfaces, two-dimensional slices, and three-dimensional volume renderings. In all cases, we visualize the data in a three-dimensional space that is analogous to the physical dimensions of the sample. We combine these representations with various rendering techniques and with user interactions to enable exploration of the data.

The most useful information is generated by deriving surfaces from the data. Figure 11-a shows a polygonal surface representing the polymer scaffold. This data was produced by NIST's OCM instrument. The surfaces are derived using simple isosurface algorithms as well as more elaborate level-set approaches.

We enhance the depiction of these surfaces with lighting effects and the use of transparency. Simulated lighting effects help to expose the shape of three dimensional surfaces. Transparency enables the user to see the internal structure of three dimensional data sets. For example, Figure 11-b shows the same scaffold surface as Figure 11-a, but the surface is transparent and it is combined with surfaces that represent cells growing on the scaffold. The cell data was produced by CFM techniques and is combined with the OCM data in this scene.

We also make use of two-dimensional representations, but we embed them as appropriate into the three-dimensional scene. For example, in Figure 11-c, we show a two-dimensional cross section through the three dimensional volume of OCM/CFM data. The scaffold surface has been removed to enable the user to see the cross section.

Finally, we use three-dimensional volume renderings to produce views of the data that show more complete structural information throughout the volume. We find that volume rendering techniques must be used judiciously and are often most useful when combined with other techniques. In Figure 11-d, we combine a volume rendering of OCM scaffold data with a surface rendering of the same data. This visualization provides useful verification of the surface derivation technique.

The user interaction techniques are an extremely important aspect of enabling exploration of the data sets. Our approach to user interactions is based on direct manipula-

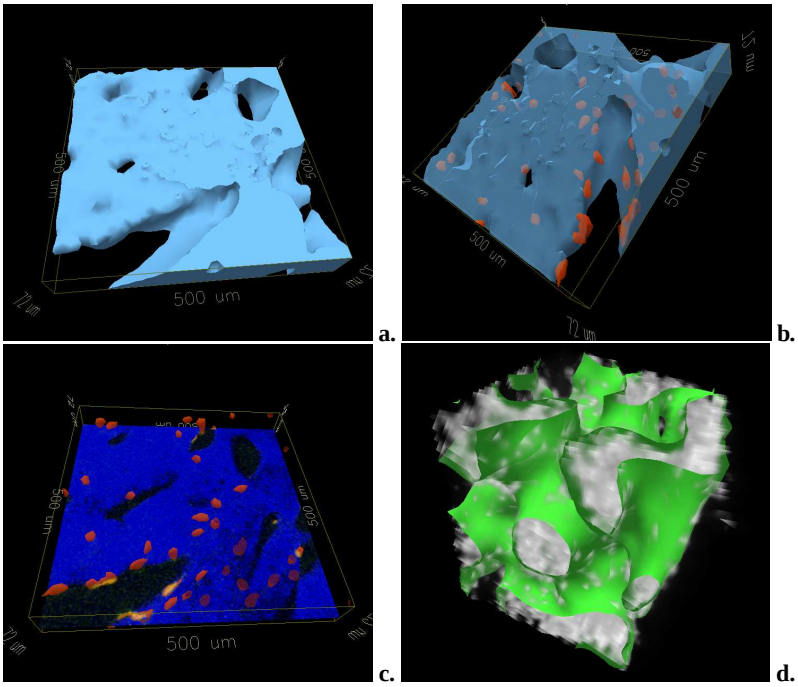


Fig. 11. a. Polymer scaffold material represented as a polygon surface. **b.** Polymer scaffold material with cells. Scaffold surface is represented as a transparent surface. **c.** A two-dimensional cross section of scaffold and cell data embedded within a three-dimensional scene. **d.** Combined volume and polygon representation of OCM scaffold data

tion and navigation of the three-dimensional scene, and on menus that are presented visually within the immersive environment. Navigation through the scene is accomplished through user movement within the immersive environment as well as translations and rotations of a track ball that is part of a hand-held device (the *wand*), which is position-tracked within the immersive environment. The on-screen menus enable the user to turn on and off various components of the scene and change transparency/opacity, and to use our interactive DSOs, such as the **clipWand** DSO described above. Figure 12 shows an example of clipping the data used in previous examples.

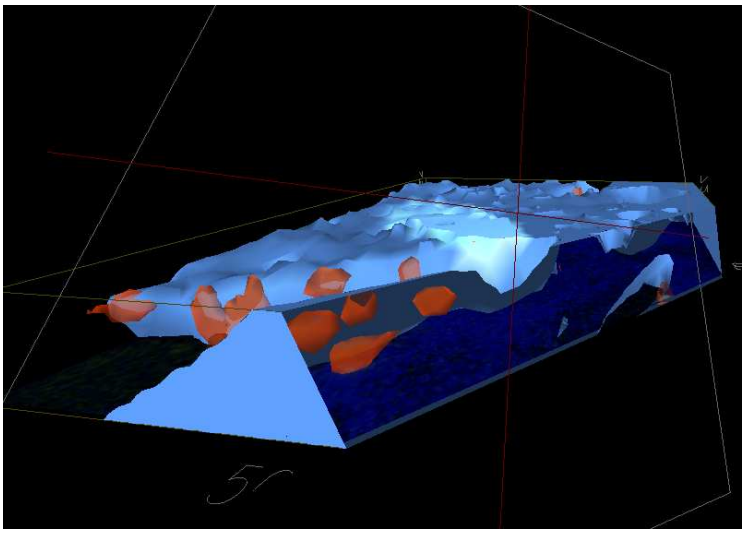


Fig. 12. Interactive clipping of scaffold and cell data visualization. A three-dimensional plane under user control cut data away to expose interior features

Another interaction technique that is useful during collaborative sessions is the use of a three-dimensional hand-tracked pointer. This pointer is depicted as a three-dimensional geometric object in the virtual scene. It is attached to the position-tracked wand so that it moves with the user's hand. This type of pointer has proved to be very useful because in the immersive environment, non-tracked users have slightly different views, so it is difficult to point accurately at an object with one's hand. But with a geometric pointer embedded in the three-dimensional scene, there is not such ambiguity. This pointer technique will also be very useful when collaborative sessions are being run at multiple sites. It will enable researchers to indicate points of interest to collaborators who are at different sites.

3.2 3D Chemical Imaging at the Nanoscale

The goal of this project is to develop methods for measuring the distribution of chemical species in three-dimensional samples at nanoscale resolution. This work is important for a variety of industrial applications, such as semiconductors, optoelectronics, and biotechnology. Our role in this project is to develop methods for visualizing and interacting with datasets of nanoscale phenomena that enable researchers to investigate three-dimensional structure and interfaces. The data sets will be produced by different instruments at a variety of scales. In some respects these data present challenges similar to those in the MMIV project; we address similar problems of registration and data fusion.

Derived surfaces are a primary means of representing these data sets. For example, Figure 13 shows a surface representing a three-dimensional scan of photonic band gap material. As in the MMIV project, these surfaces are generated using techniques such as isosurface and level set segmentation methods. As we get data from more and diverse instruments, we expect that the segmentation of the data into different chemical species will prove to be a particularly challenging aspect of the project.

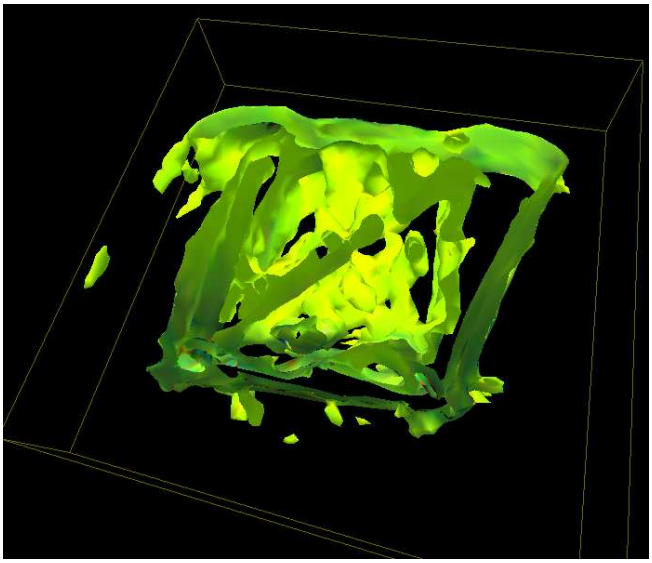


Fig. 13. A surface representing a three-dimensional scan of photonic band gap material

User interaction techniques are a critical component in the exploration of these data sets. As in the MMIV project, we provide to the user a variety of navigation and menu-based interaction techniques. Particularly useful is the interactive clipping described above. Transparency and lighting remain very important elements of visualization methods used in this project.

3.3 Smart Gel

NIST scientists and collaborators are using the Virtual Laboratory to study *smart gels*, which might someday be used to make exotic foods, cosmetics, medicines, sensors, and other technological devices. Smart gels are inexpensive materials that expand or contract in response to external stimuli. This property could be useful in applications such as an artificial pancreas that releases insulin inside the body in response to high sugar levels. Scientists need to understand how the molecules in these materials behave in order to utilize them in new products.

For this project NIST scientists are studying a subclass of these materials called *shake gels*. Through some complex and as yet unknown process, these watery mixtures of clays and polymers firm up into gels when shaken, and then relax again to the liquid phase after some time has passed. A shake gel might be used, for example, in shock absorbers for cars. The material would generally be a liquid but would form a gel when the car drove over a pothole; the gel thickness would adjust automatically to the weight of the car and the size of the pothole.

The VL helped the scientists see that it is the polymer's oxygen atoms, instead of the hydrogen atoms as previously thought, that attach to the clay. The team has also made theoretical calculations that may help to explain why and how the components of the liquid mixture bind together into a semisolid form. Electrical charges affect the binding process, resulting in water binding to clay surfaces in a perpendicular arrangement, which is believed to help create the firmness of the gel. Theoretical aspects of the smart gels research are discussed in [43]. The work is sponsored by Kraft Foods and involves scientists from NIST, Los Alamos National Laboratory, and Harvard University.

The data provided by the researcher consisted of position data for a collection of atoms over a series of time steps. We created two visual representations from these data. The first depicts stick figure representations of the molecules. Bonds between atoms are derived from proximity of the atoms and the atoms are colored by element. The time series is shown by animating the motion of the atoms and bonds. Figure 14 shows a still frame from this representation.

The second representation draws paths for the atoms as they move over the course of the time series. Each path is shown as connected line segments. The color of the each path indicates the element. In this representation, the movement of atoms over time is represented within a static scene rather than by an animation. Figure 15 shows the path representation of the data.

Note that the two different representations can be used together. When both representations are turned on, the immersive display provides both an atom's motion and its history.

User interactions consist largely of menu interactions; these were implemented within our existing general purpose immersive display utility with menu enhancements. The added menus were easily implemented using the techniques previously described. Menu items allow the different visual representations of the data to be turned on and off, the animation to be stopped and started, and paths to be made visible based on identity.

The implementation of the immersive visualization for this project was well served by our philosophy used in the IVE (loosely coupled programs that work well together). A new data filter was created using a scripting language to convert the time series data

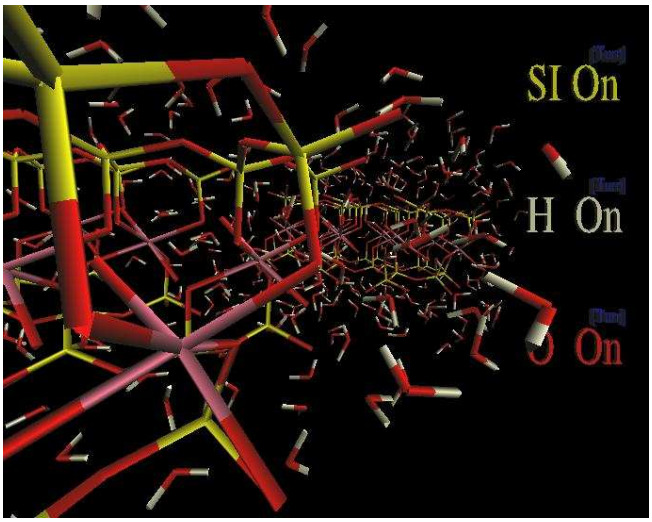


Fig. 14. Animated stick-figure representation of the *smart gel* simulation

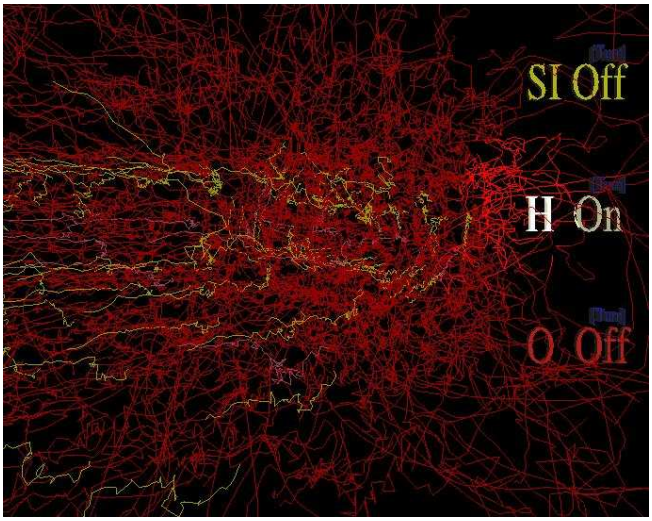


Fig. 15. Atom path representation of the *smart gel* simulation

produced by the numerical simulation program into a form suitable for loading into the immersive environment. In this case, an existing file loader implements a *flip book* style animation. The input pipe line thus quickly converts the data of atomic positions at each time step into an visual stick figure representation of the each atom, which animates for the duration of the simulation time frame. Since only a small part of the input pipeline needed new software, the project implementation time was short. The newly created software filter focuses on producing the stick figure representations for the atoms, while the existing file loader handles the task of animating the atoms.

As mentioned above, within minutes of being immersed in the animation, the primary scientists involved realized their previously conceived ideas about which molecules attached to the clay were reversed.

The wide range of our current scientific collaborations has led to an enhanced set of VL tools in the areas of surface and volume rendering and display, creative data representations, animation, and time series data analysis.

3.4 Nanostructures

In the nano-optics project we are working with NIST scientists to understand atomic scale variation in the calculated properties of nano-scale structures.

The data that we receive consists of atom positions and values for s and p orbitals. We represent each atom with a sphere that corresponds to the s orbital and appropriate shapes for the x, y, and z p orbitals. The size of these geometric forms correspond to the calculated values. The colors of the atoms correspond to the element.

As in the other applications, we use our approach of loosely coupled tools to convert the incoming data to the desired geometric representation. In this case, our primary set of tools is the Glyph ToolBox (GTB). These tools facilitate the creation, sizing, and coloration of the orbital shapes.

As part of the creation of these geometric representations, we segment the data into subsets of atoms based on orbital size and symmetry (s or p) and on the spatial location of the atoms. User interactions were implemented using our standard menu tools within the immersive environment. These interactions enable the user to turn on and off the different subsets of atoms. This lets the user explore various features of the nano-scale structures. A sample image from this project is shown in Figure 9.

4 Conclusions and Future Work

According to Louis Pasteur “In the field of observation, chance favors only the prepared mind” [44]. Scientists are prepared, but can we increase chance? We believe ***representation and interaction drive discovery***, and when applied in the most general sense with the tools of immersive visualization, data mining and other analysis, it is possible to accelerate concept development.

In our current scientific world of increasing computing power, parallel laboratory experiments and enormous data sets to explore, a Virtual Laboratory becomes a powerful tool to ferret out important scientific results from otherwise incomprehensible volumes of data. Each component is essential to create a productive environment. Software

and hardware capabilities must handle issues of networking and communicating over distances and times. They must take into account the varying resources of the scientists using them, and be capable of growing with new technology as it emerges. The software must be easily available on user-accessible systems. Software tools must handle all types of needs in terms of making scientific data available for viewing in a visualization system, and then making it accessible for user interaction. Limits of scientific resolution lie in how well data can be represented in a visualization system. Tools must be available for studying a wide range of different types of data, on all different size scales, and in a wide range of incoming formats.

The challenges lie in creating a foundation for the Virtual Laboratory that can be built upon as technology changes, and as computational tools improve. Assembling packages of individual tools that can be put together in a variety of useful ways, our current Virtual Laboratory is the beginning of a robust, encompassing scientific laboratory of the future.

5 Disclaimer

Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

References

1. Sims, J.S., George, W.L., Satterfield, S.G., Hung, H.K., Hagedorn, J.G., Ketcham, P.M., Griffin, T.J., Hagstrom, S.A., Franiatte, J.C., Bryant, G.W., Jaskolski, W., Martys, N.S., Bouldin, C.E., Simmons, V., Nicolas, O.P., Warren, J.A., am Ende, B.A., Koontz, J.E., Filla, B.J., Pourprix, V.G., Copley, S.R., Bohn, R.B., Peskin, A.P., Parker, Y.M., Devaney, J.E.: Accelerating scientific discovery through computation and visualization II. Journal of Research of the National Institute of Standards and Technology **107** (2002) 223–245 May-June issue. `http://nvl.nist.gov/pub/nistpubs/jres/107/3/cnt107-3.htm`.
2. Westmoreland, P.R., Kollman, P.A., Chaka, A.M., Cummings, P.T., Morokuma, K., Neurock, M., Stechel, E.B., Vashishta, P.: Application of Molecular and Materials Modeling. Technology Research Institute, World Technology Division, Kluwer Academic Publishers (2002)
3. Alivisatos, A.P.: Semiconductor clusters, nanocrystals, and quantum dots. Science **271** (1996) 933–937
4. Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38** (1965) `http://www.intel.com/research/silicon/mooreslaw.htm`.
5. Moore, G.E.: No exponential is forever ... but we can delay 'forever'. presented at: Int'l Solid State Circuits Conf. (ISSCC) (2003) `http://www.intel.com/research/silicon/mooreslaw.htm`.
6. Grosh, J., Kaluzniacki, R., Dongarra, J.: Transition of solving important problems. unpublished (2004)
7. Salvator, D.: GPU wars heat up again. PC Magazine (2004) 34–35 `http://www.extremetech.com/6800`.

8. Salvator, D.: Nvidia readies geforce 6800. Ziff-Davis online magazine: ExtremeTech.com (2004) http://www.extremetech.com/article2/0,1558,1624009,00.asp.
9. Edwards, W.K.: Core Jini. 2nd edn. Prentice Hall PTR (2000)
10. Freeman, E., Hupfer, S., Arnold, K.: JavaSpaces Principles, Patterns, and Practice. Addison-Wesley (1999)
11. Gelernter, D.: Linda in context. Comm. of ACM **32** (1984) 444–458
12. Gelernter, D.: Generative communication in Linda. ACM Trans. Prog. Lang. and Sys. **7** (1985) 80–112
13. Sun Microsystems, Jini Community: Surrogate project. http://surrogate.jini.org (2004)
14. George, W.L., Scott, J.: Screen saver science: Realizing distributed parallel computing with Jini and Javaspaces. In: Conf. on Parallel Architectures and Compilation Techniques (PACT2002). (2002) http://math.nist.gov/mcsd/savg/papers/SSS_PACT2002.ps.gz.
15. Sherman, W.R., Craig, A.B.: Understanding Virtual Reality. Morgan Kaufmann (2003)
16. van Dam, A., Forsberg, A.S., Laidlaw, D.H., LaViola, Jr., J.J., Simpson, R.M.: Immersive VR for scientific visualization: A progress report. IEEE Computer Graphics and Applications **20** (2000) 26–52
17. van Dam, A., Laidlaw, D.H., Simpson, R.M.: Experiments in immersive virtual reality for scientific visualization. Computers and Graphics **26** (2002) 535–555
18. Bryson, S.: Virtual reality in scientifi c visualization. Comm. ACM **39** (1996) 62–71
19. Card, S.K., Mackinley, J.D., Shneiderman, B.: Readings in Information Visualization. Morgan Kaufmann (1999)
20. Kerighan, B.W., Pike, R.: The UNIX Programming Environment. Prentice Hall, Inc. (1984)
21. Kelso, J., Arsenault, L., Satterfield, S., Kriz, R.: DIVERSE: A framework for building extensible and reconfigurable device independent virtual environments. In: Proc. IEEE Virtual Reality 2002 Conf. (2002) DIVERSE source code available at http://diverse.sourceforge.net.
22. SGI: OpenGL Performer™. http://www.sgi.com/software/performer (2004)
23. SGI: Open Inventor™. http://oss.sgi.com/projects/inventor (2003)
24. IBM: OpenDX. http://www.opendx.org (2004)
25. Kitware Inc.: VTK: The Visualization Toolkit. http://www.vtk.org (2004)
26. Brady, R., Pixton, J., Baxter, G., Moran, P., Potter, C.S., Carragher, B., Belmont, A.: Crumbs: a virtual environment tracking tool for biological imaging. In: Proc. 1995 Biomedical Visualization (BioMedVis '95), IEEE Computer Society (1995) 18
27. VRCO: CAVElib user manual. http://www.vrco.com/CAVE_USER/index.html (2003)
28. CAVEav: CAVE audio & video library. http://www-fp.mcs.anl.gov/~judson/CAVEav/CAVEav.html (2004)
29. Roussos, M., Johnson, A., Moher, T., Leigh, J., Vasilakis, C., Barnes, C.: Learning and building together in an immersive virtual world. Presence **8** (1999) 247–263
30. Leigh, J., Johnson, A.E., DeFanti, T.A., Brown, M.D.: A review of tele-immersive applications in the CAVE research network. In: VR. (1999) 180
31. Leigh, J., Rajlich, P., Stein, R., Johnson, A.E., DeFanti, T.A.: LIMBO/VTK: A tool for rapid tele-immersive visualization. In: Proc. IEEE Visualizaton '98. (1998) http://www.evl.uic.edu/cavern/cavernpapers/viz98/leigh_j.pdf.
32. Morgan, T., Kriz, R.D., Howard, S., Das-Neves, F., Kelso, J.: Extending the use of collaborative virtual environments for instruction to K-12 schools. In>>sight **1** (2002) 67–82 http://www.sv.vt.edu/future/cave/pub/kriz_ael/insight.pdf.

33. Kriz, R.D., Farkas, D., Ray, A.A., Kelso, J., Jr., R.E.F.: Visual interpretation and analysis of HPC nanostructure models using shared virtual environments. In: Conf. Proc. High Performance Computing: Grand Challenges in Computer Simulations, The Society for Modeling and Simulation International (SCS), San Diego, California (2003) 127–135 http://www.jwave.vt.edu/~rkriz/Pubs/HPC_2003/hpc2003distribute.pdf.
34. Ray, A.A.: The collaborative toolkit for diverse. http://www.sv.vt.edu/future/cave/software/D_collabtools/D_collabtools.h%tml (2003)
35. Rich, E., Knight, K.: Artificial Intelligence. McGraww-Hill (1991)
36. Inselberg, A.: The plane with parallel coordinates. The Visual Computer **1** (1985)
37. Cheng, J., Hatzis, C., Hayashi, H., Krogel, M.A., Morishita, S., Page, D., Sese, J.: KDD Cup 2001 report. SIGKDD Explorations **3** (2002)
38. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kauffann (1999)
39. Cheeseman, P., Kelley, J., Self, M., Taylor, W., Freeman, D.: Autoclass: A Bayesian classification system. In: Proc. 5th Int'l Conf. on Machine Learning. (1988)
40. Hagedorn, J.G., Devaney, J.E.: A genetic programming system with a procedural program representation. In: 2001 Genetic and Evolutionary Computation Conf. Late Breaking Papers. (2001) http://math.nist.gov/mcsd/savg/papers/g2001.ps.gz.
41. Devaney, J.: The role of choice in discovery. Lecture Notes in Computer Science **167** (2000) eds. S. Arikawa and S. Morishita, Springer.
42. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) http://www.ics.uci.edu/~mlearn/MLRepository.html.
43. Aray, Y., Marquez, M., Rodríguez, J., Coll, S., Simón-Manso, Y., Gonzalez, C., Weitz, D.A.: Electrostatics for exploring the nature of water adsorption on the laponite sheets' surface. J. Phys. Chem. B **107** (2003) 8946–8952
44. Louis Pasteur: Lecture. http://www.quotationspage.com/quotes/Louis_Pasteur/ (1854)