Glyphmaker: Creating Customized Visualizations of Complex Data

William Ribarsky, Eric Ayers, John Eble, and Sougata Mukherjea
Georgia Institute of Technology

Glyphmaker's general approach to data visualization and analysis lets users build customized representations of multivariate data and provides interactive tools for exploring patterns and relationships.

Current visualization/analysis tools offer a limited selection of visual representations, and they're optimized for only a few well-known applications, such as computational fluid dynamics (CFD) or finite-element analysis. The available representations — for example, surface or volume rendering of scalar or vector fields — are less useful for spatially complex multivariate data with a high correlation among variables. Such data is common in physics and materials science, but it can also appear in CFD and finite-element results.[1] In these cases, when users attempt to replace the usual visual presentation with one that better characterizes their data, they lose the system's built-in, programmerless functionality. Thus, with extensible systems, such as dataflow systems, they must suddenly know quite a bit about graphics programming — or employ someone who does. With nonextensible systems, they are just out of luck.

Our system, called Glyphmaker,[2] allows nonexpert users to customize their own graphical representations using a simple glyph editor and a point-and-click binding mechanism. In particular, users can create and then alter bindings to visual representations, bring in new data or glyphs with associated bindings, change ranges for bound data, and do these operations interactively. They can also focus on data down to any level of detail, including individual elements, and then isolate or highlight the focused region. These features empower users, letting them employ their specialized domain knowledge to create customized visual representations for further exploration and analysis.

For ease of design and use, we built Glyphmaker on top of Iris Explorer, the Silicon Graphics Inc. (SGI) dataflow visualization system. The current version of Glyphmaker has been successfully tested on a materials system simulation. We are planning a series of tests and evaluations by scientists and engineers using real data and welcome contacts from users with spatially complex multivariate data.

Glyphs

Glyphs are graphical objects whose attributes — position, size, shape, color, orientation, etc. — are bound to data. These objects can be effective in depicting discrete

0018-9162/94/$4.00© 1994 IEEE

data such as macromolecular structure and interactions, but they have also proved their usefulness in representing variables such as wind speed and direction in atmospheric dynamics simulations and observations.[3] Glyphs owe their effectiveness to the human eye-brain system's ability to discern finely resolved spatial relationships and differences in shape. They allow the user to display and correlate several variables at once. A researcher can, for example, distinguish two variables by binding them to the visually orthogonal representations of shape distortion and pseudocoloring.

We have shown[4] that complex, three-dimensional structures from molecular dynamics simulations and all components of the stress tensor acting on individual atoms can be distinguished and correlated by using shape distortion and spot coloring of spherical glyphs bound to the atomic positions. It is important to reveal this interdependence because the atomic stresses force the plastic deformations that result in significant structural changes.

In a different context, Ellson and Cox[1] have shown the usefulness of glyphs in depicting the flow patterns that result from finite-element simulations of hot plastic injected into a mold. They placed 100 to 200 glyphs according to a finite-element grid. Each glyph, a 3D object, represented velocity as a shape distortion of appropriate magnitude and direction. Temperature and pressure were represented by different color scales on the glyph and its base, respectively.

Other work uses color or gray-scale icons that merge color and texture perception to represent multiple parameters.[5,6] Here the icon is a generalization of the single pixel to objects having perceivable features and attributes. Typical icons are stick figures or simple collections of lines where orientation and length represent different parameters. Color coding has also been used to bind additional parameters to the icons for satellite images or MRI (magnetic resonance imaging) data. In another field, Haber[7] has used tensor glyphs in visualizing data from engineering mechanics.

These and many other examples show the range and power of glyph representations. Going beyond these cases, our thesis is that glyphs are generally useful for representing all types of multivariate data, especially those with variables that have significant dependence on two or three spatial dimensions.

Glyphmaker: Approach and objectives

Our goal is to create a general approach for building customized representations of multivariate data and provide interactive tools for exploring patterns and relationships within these data.

Approach and background. The historical motivation for our approach comes from the work of Foley and others. In a system developed for dynamic process visualization, Foley and McMath[8] showed how nonprogrammers could construct and customize dynamically updated scenes displaying process information. This information came from real-world environments such as factories, power plants, and refineries. It was displayed (and the display modified) to provide precise monitoring of these complex processes. The idea was to make available a library of icons (2D glyphs) or allow the user to create them; the user then employed binding tables to connect variables to icon elements. Thus, the user could create a customized, graphical representation describing a particular set of processes and do so without intensive programming. Later, Foley[9] proposed extending this idea to more general visualization schemes, such as those for scientific data. Here the data might be two- or three-dimensional, and the user might need to analyze several variables at once.

Several systems built around the same basic dataflow concept allow you to build your own applications.

The recent development of dataflow approaches provides a general, compatible framework for Glyphmaker. Although dataflow and similar schemes based on a visual programming style had been investigated for years, the first widely employed dataflow approach was the Application Vizualization System[10] by Advanced Visual Systems. Now there are several additional systems built around the same basic dataflow concept; these include SGI's Iris Explorer, IBM's Explorer, and the University of New Mexico's Khoros. They all allow you to build your own applications by constructing graphical networks of modules for data input, filtering, transformation to geometry, pseudocoloring, and rendering. These approaches provide high interactivity, since each module has its own control panel. They're also distributable in a heterogeneous environment and extensible, since users can program and insert their own modules.

The research described in the last section explored a wide range of glyph representations, starting with a few hundred highly detailed glyphs[1] and extending to several thousand glyphs where gestalt or collective effects were more important than individual glyph characteristics.[5,6] Our previous work explored the middle ground and showed that one could effectively perceive, use, and analyze individual glyph elements even when there were thousands of glyphs.[4] With our general approach to Glyphmaker, we hope to make it possible for users to effectively represent, visualize, and analyze data in all these categories.

Objectives. Glyphmaker is an exploratory tool. It's meant for users who do not fully understand their data and, in particular, do not know which visualization will promote understanding. Thus, it allows users to explore and then analyze their data. The current version of Glyphmaker is especially useful for the exploratory phase. In the future, we will develop approaches for more detailed and quantitative analysis.

Our main design objectives are

- *Detailed control of data visualization.* Glyphmaker offers a unique capability: It allows binding a distinguishable glyph to each element of a data set, thus providing user control of the visualization at the finest level of detail. Using conditionals, isolation, and highlighting, the user can bring out any detail with color or other glyph elements.
- *Highly responsive controls.* We're seeking a particular type of interactivity — interactivity in the controls. Adjusting a control should produce a quick response in the visualization. For some tasks, responses of a frac-

tion of a second to a few seconds may be adequate. However, our notion of interactivity calculates response time as the sum of the time to search for the control and set it, plus the update time once the control is set. Thus, we are prioritizing controls and placing the most important ones close at hand.

- *Interactivity in visualization.* Waiting many seconds to see the result of a control change — no matter how well-placed and direct the control — causes a significant loss in interactivity. To meet interactivity needs, we used the SGI graphics library (GL) to program our Editor module; however, the Explorer's Render module is adequate for final visualizations. Although use of this module ensures optimized rendering now and in the future, users will still face situations where response is slow — especially when visualizing thousand of objects. Glyphmaker's ability to replace sets of objects with points, or remove them altogether, can be powerful in this case. Removal is especially useful because we have incorporated the power to turn selected regions on or off at the touch of a dial.
- *Focus and conditions.* Statistical graphics research has yielded a useful set of principles for investigating multivariate data. Much of this work has concentrated on focusing and linking.[11] Focusing methods may involve selecting a subset of the data, reducing dimensions, or using perceptual techniques to focus user attention on a certain area. Traditional scientific visualization techniques, such as zooming, panning, and slicing, are examples of subset selection. However, complex data often require additional focusing techniques, for example, to bring out data in the same spatial area as other data that is not of interest or to view data that is surrounded and hidden by other data. Our Conditional Box meets this requirement.
- *Correlative linking.* This is a process from statistical graphics that links multiple views, each containing partial information about a complex data set, so that users can integrate them into a coherent picture of data relations.[11] Views can be linked by smooth animations in time sequences or by feature emphasis in simultaneous displays. Effective methods for linking multidimensional point data[11] — and undoubtedly other data as well — involve bringing up simultaneous multiple views (or quickly alternating views) and emphasizing corresponding parts with a distinctive color, brightness, or symbol. Methods such as highlighting to link simultaneous views displaying different properties are straightforward with Glyphmaker. Users can choose a certain part of the data or use the Conditional Box to select a range in a property or a spatial region. Then, they can bind the selected part to a special color, transparency, or glyph shape.
- *Efficient accommodation for data filtering and transformation.* Data from different sources is apt to be scaled differently and may require more complex transformations for comparative viewing. Transformation problems are becoming more common with the explosion of experimental, observational, and computational data that must be compared in detail. Thus, several investigators are studying the issue of precise filtering and transformation of large amounts of data. One outcome of these studies is an appreciation for the effectiveness of self-describing data structures. Glyphmaker uses a simple self-describing data structure to set up an automatic reading procedure that converts data in different formats to the dataflow system's common internal format.

Glyphmaker users can bind the selected part to a special color, transparency, or glyph shape.

To look at raw results from simulations or experiments in more detail, we must also deal with increasing noise and fluctuation levels. This requires straightforward, statistically balanced data filtering and smoothing tools — an area we are just beginning to attack for Glyphmaker.

Architecture and interfaces

Although we built Glyphmaker on top of the Iris Explorer, we could have built it on top of AVS or, possibly, Khoros, and we see no major obstacle in porting to one of these environments.

We wanted to build on, rather than replace, the functionality already in the dataflow system, so wherever possible we used existing Explorer modules such as the Generate Colormap and Renderer modules. In particular, using the appropriate Explorer modules, we can convert data into surface or volume geometry and pipe it to the Renderer for simultaneous display with glyph representations, thus achieving mixed representations. This approach allows us to concentrate on the novel aspects of our work and to easily incorporate our additions.

The modules we built are written in C; we used Motif for the user interface and GL for the graphics. The GL graphics appear only in the Editor and are thus quite modular.

The Glyphmaker system has four main interactive parts.

- The Read module converts input data from a user-specified format to the Explorer lattice format so that it can be read in by any existing Explorer module.
- The Glyph Editor creates glyphs and arranges them, if desired, into compound glyphs.
- The Glyph Binder allows highly interactive bindings or binding changes between data variables and glyph elements. It also allows users to quickly change the range of either data variables or glyph elements and to save new bindings or reload previous ones.
- The Conditional Box applies the focusing condition.

Figure 1 shows how these parts fit into the Explorer dataflow environment.

Read module. A major problem for data visualization systems is retrieving data from a variety of formats and placing them in a common structure for comparative viewing or analysis with a common set of tools. This problem has become especially acute because modern correlative analysis demands combining data from diverse simulations, observa-

tions, and experiments into joint displays or simultaneous manipulations to produce detailed, integrated understanding. The problem is exacerbated because the data may be organized into large, complex data sets that require specially optimized procedures to search, accumulate, and prepare the data for display. We have not attacked this general problem in our work, but our self-describing data structure is consistent with other work on the problem.[3]

Each self-describing file has a header, which describes the data and its format, followed by the data itself. Each variable in the header is labeled with a name, primitive type, number of instances, minimum value, and maximum value. The important label and minimum/maximum values are passed to the Glyph Binder. The number of records with identical format is indicated at the top of the header; this allows concatenation of an animation sequence or different views of the data. Each header can also contain comments about file contents. The data follows in the order given in the header. Headers and data can be placed anywhere in the file. Headers can also be placed together at the beginning, with all the data following the last header; this allows merging files while keeping header information accessible.

We chose this simple format for understandability and ease of use and because it fulfilled Glyphmaker's basic requirements. The Read module will accept the data file in straight ASCII form (for ease of editing and use), as a combination of ASCII header plus binary data, or as straight binary (for compactness and input speed).

Glyph Editor. The Glyph Editor is a simple 3D editor used to draw the glyphs and, if desired, to arrange and orient them in compound glyphs. There are four selection menus, as seen in the upper righthand corner of Figure 2: Glyphs, Views, Mode, and Action. The Glyphs menu items are points, lines, spheres, cuboids, cylinders, cones, and arrows. The Views menu, which helps the user manipulate the glyphs in three-dimensional space, has option buttons for selecting a perspective, top, side, or front view, or all four views simultaneously. The Action menu has bind, render, ungroup, and clear options for bringing up the Binder, rendering a set of glyphs (with full lighting and shading — useful for seeing how they will look in the finished visualization), ungrouping grouped glyphs, and clearing all glyphs from the Editor. We will soon include options to save and load useful glyph constructions, which frequently take some care and testing to develop.

Each glyph has several elements that might be bound to data. For example, a user might bind the radius, position (x, y, z), color, overall size, and transparency

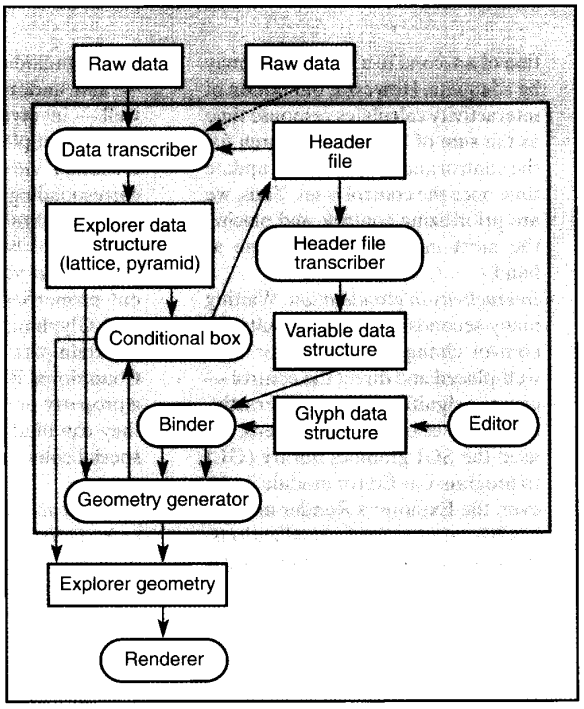


Figure 1. Flow of data in the Glyphmaker visualization environment. Raw data flows into the data transcriber (in the Read module), is converted to Explorer data structures, and is then used to generate geometry after binding. All elements inside the box were created for Glyphmaker, which is built on top of the Iris Explorer dataflow environment.

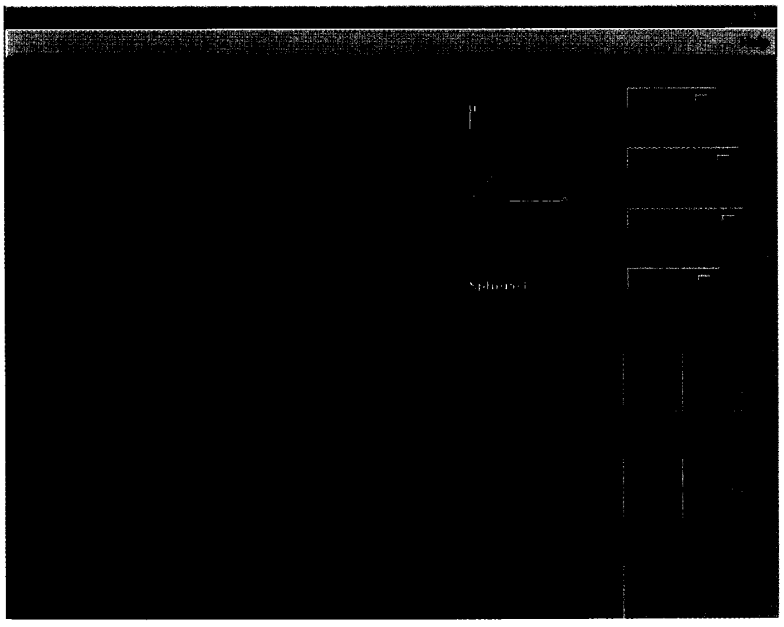


Figure 2. 3D Glyph Editor interface. Selection menus — Glyphs, Views, Mode, and Action — are in the upper right corner; a compound glyph composed of a sphere and a vector glyph is displayed.

of a sphere to individual data variables. Shape, such as elongation or shortening along each axis, might be bound to additional variables. We have, in fact, used this last element quite successfully to depict components of the atomic stress tensor in molecular dynamics visualizations.[4] Similar sets of elements are available for all glyphs in the Glyphmaker library. Users choose which elements to activate for the next phase, and only these elements appear in the Glyph Binder menu. But they can also bypass the active element selection entirely; in that case, a smaller default set of elements appears in the Binder. We took these steps to increase overall speed in setting up visual representations and interactivity in the Binder. Since each glyph has several elements, a complete display could make menus quite long and increase the time to find and manipulate a given element.

Figure 2 shows a grouped glyph. The cylinder and small sphere are grouped to make a vector and are embedded in the large sphere, which signifies position (color and other elements could also be bound to additional variables). This compound glyph can be bound to vector or tensor components. To facilitate construction of compound glyphs, each glyph has a body axis and an orientation axis. The body axis stays fixed with the glyph, while the orientation axis may be translated and rotated. The orientation axes allow users to position and orient glyphs with respect to one another; they can also be used to create new binding elements in compound glyphs. (For example, the "hinge angle" between two cuboid glyphs joined along a common edge could be bound to a data variable.)

Glyph Binder. For binding, the user chooses the bind option from the Editor's Action menu. A pop-up window shows all active elements and variables as toggle buttons. (See Figure 3.) To bind a variable to an element, the user simply selects an active element followed by a variable or a variable followed by an active element. To see which variable/element an element/variable is bound to, the user clicks an already-lit toggle button corresponding to the element or variable, and a pop-up window shows the desired information, as shown in Figure 3. Bindings can also be deleted and reset with the mouse. Text widgets attached to each variable and active element specify the minimum and maximum values for that variable or active site. Default values appear initially in these widgets. The default for the variable is taken from the self-describing file header. Using defaults permits establishing bindings in a few clicks and immediate display of results. Of course, the user can change any of these values, and the result will be transmitted to the renderer to update the display.

We tried to create the Binder with the interactivity and ease of control necessary for effective data exploration. The variables and elements are in separate scrollable lists (Figure 3), so users can quickly find variables and elements of interest, even if the lists are longer than the window size. The Binder window can be resized and moved using the mouse, so users can easily rearrange the display real estate. This is important, since the Binder is often used concurrently with the Renderer for interactive refinement of data visualizations. The Render button (below the window) allows users to pipe Binder changes to the Renderer for display. The OK button (bottom left) dismisses the Binder, with data/glyph bindings remaining in place for the interactive visualization. File (top left) has a pull-down menu with the usual save/reload options. Saving and loading bindings is important, since it may take some time to find an ideal set of bindings and associated minimum/maximum values for a particular set of data, and the user may want to recall that configuration at a later time.

To see how the interactive features of the Binder work in practice, suppose a researcher uses Glyphmaker on a complex finite-element simulation where she has only a general idea of the results. To begin, she selects a simple glyph and places it in the Binder. She then binds the glyph position to the coordinates of each finite-element patch and selects color and glyph height to display two properties. Since she doesn't know the range of properties in this simulation, she guesses some likely ranges and hits the Render button to bring up a visual representation. After a few time steps through the data, she discovers the variable ranges truncate the data, so she adjusts the ranges, hitting the Render button each time to display the result. After a few iterations, she finds the ideal range to bring out the behavior she wants to study. Then she notices that the glyph height range obscures some data. It should be reduced somewhat for the variable ranges selected. After some more adjustments and display, she arrives at a better choice. She may then try out the same set of bindings on different glyph shapes or different properties (for example, expansion of

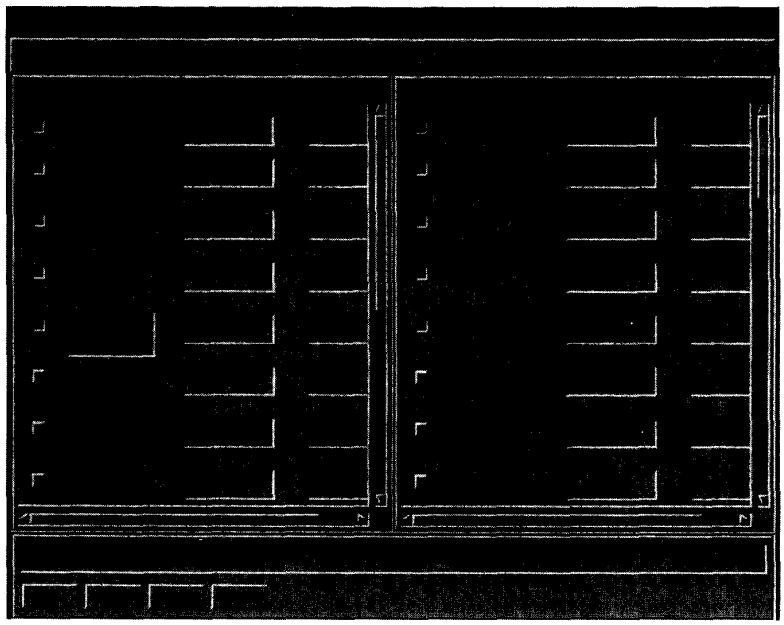


Figure 3. The Glyph Binder interface. On the left are the glyph active elements, and on the right are the variable names. This example includes position coordinates and properties for each atom type.

Figure 4. The full molecular-dynamics system with the semi-transparent Conditional Box surrounding the NaCl cluster and Ne in the impact region.

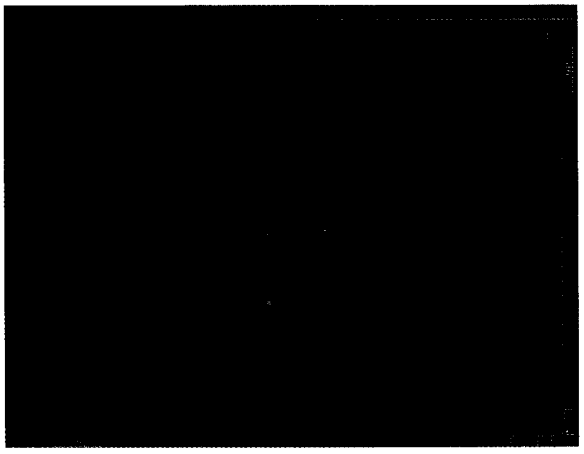


Figure 5. The full molecular-dynamics system after the cluster has embedded itself in the Ne; here the color glyph elements are bound to local temperature.

glyph size rather than height) to refine her view of the data. She may also bind an additional set of properties to the same glyph structure and display the results of the two bindings side by side for correlative analysis. Likewise, she could add active elements to her glyph structure (for example, another color scale or changes in width along with height) to display original and additional properties in the same image. This is the process of exploration and iterative refinement for which we designed Glyphmaker.

Conditional Box. The focusing and conditioning design objective could be expressed by applying conditions of the general form

$$A_i \leq X_i \leq B_i$$

where X_i is a data variable and A_i, B_i are bounds. Thus, we might limit our view to, say, those elements from a simulation that have an associated energy above (or below) a certain value. To apply these conditions to spatial variables, we implemented the Conditional Box. This three-dimensional box can be moved and sized to enclose a data subregion after it has been selected for display and bound to particular glyphs. The box allows us to distinguish what is inside from what is outside — in effect, to reclassify the data according to this condition. Thus, the data, which may have been originally classified according to, say, atom type, is now additionally classified according to this spatial condition. We can then choose new glyph bindings — for example, new colors or glyph shapes for what is inside the box — to focus on or even isolate an interesting spatial region. Even individual data elements can be selected with this method, giving the detailed control and immediate feedback mentioned in our design objectives.

In our current implementation (see system schematic in Figure 1), the box module receives information from the Explorer data structure and from the Geometry Generator to generate a box geometry with the appropriate scale and position with respect to the data. Then both bound data and box geometry are piped to the Rrenderer for simultaneous display. The Conditional Box controls can be used to update the box geometry and position it with respect to the data. This procedure is necessary because the Explorer Renderer allows only certain preconfigured interactions, and there is no way to add controls directly to the renderer for moving the box around.

The 3D Conditional Box can be moved and sized to enclose a data subregion.

Glyphmaker example

For an example, we selected a materials system simulation of a type that is becoming common as available computer power increases. In molecular dynamics (MD) simulations, one studies dynamic behavior of atomistic materials as they move and interact with one another. External and internal forces may cause the buildup of large, localized, and transient stresses and heating, resulting in such dynamic events as plastic flow, localized melting and resolidification, and even rupture or splattering of material.

Understanding these processes is important in developing microscopic theories of friction, lubrication, and wear; in explaining the operation of instruments like the atomic force microscope and scanning tunneling microscope; in developing micromachining methods; and in investigating the stability of surfaces and interfaces for microscopic applications. We thus have a problem where the complex, 3D behavior of several variables must be analyzed and understood. The simulation presented here was run on a Cray Y-MP at the US Department of Energy National Energy Research Center by Cheng and Landman of Georgia Tech's School of Physics.[12]

The simulation depicts a nanocluster of 64 sodium chloride (NaCl) atoms hitting and becoming embedded in a neon (Ne) surface. Initially, the system is cold

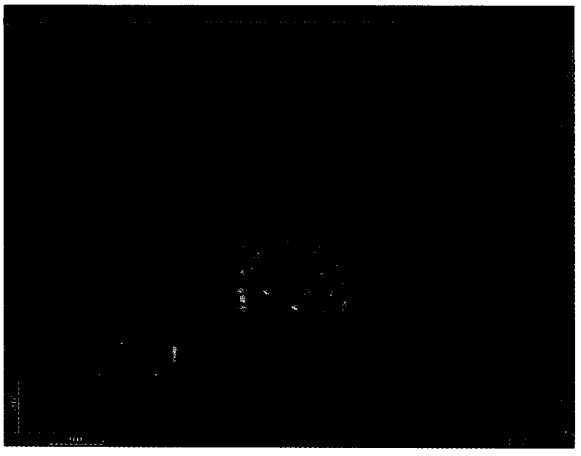


Figure 6. The part within the Conditional Box (same time step as Figure 5) with the local temperature binding turned on. The color bar for the local temperature binding is in the lower left corner; the 0-to-100 scale maps linearly onto 0 to 300° K.

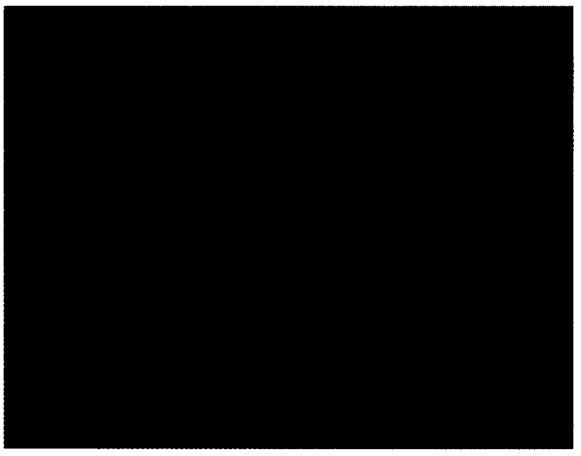


Figure 7. The crystalline substrate with Ne and NaCl bindings turned off (same time step as Figure 5) and with visualization of the on-axis (stretching/compressive) components of the atomic stress tensor added.

(50° K) and the impact velocity moderate (3 km/sec), but upon impact, the cluster and surface vicinity heat with much disorganization and eventually some vaporization. We chose simple sphere glyphs for binding to the separate components (Na and Cl separately in the cluster, dynamic Ne atoms in several layers stacked on several layers of crystalline substrate). Properties of this system include localized atomic kinetic energy and temperature and also the atomic stress tensor. In the visualizations presented here, we bound the temperature to the glyph color for each atomic type. Figure 4 is a visualization of the system just before the cluster penetrates the Ne layers; the crystalline substrate has not been bound to a glyph and thus does not appear.

In Figure 5, from a later time step, the cluster has fully penetrated the neon. Here the binding of color to local atomic temperature is turned on; it can quickly be turned on or off in the Binder. Figure 5 shows a common problem with analyzing simulations as they grow in size; often, important features are embedded in the system and hidden from view. To overcome this problem, we apply a Conditional Box. Using the Conditional Box controls (lower left, Figure 4), we adjust the semitransparent box to enclose the cluster and the surface region it's about to strike. Now we can choose to view only this selected region, the excluded region, or both (perhaps using the transparency control to make the excluded region semitransparent). We can also retain the Conditional Box classification for later time steps.

Figure 6 shows the selected region after the cluster embeds itself (with local temperature binding turned on). In this case, we can see (by comparing with earlier time steps) that the Ne atoms have diffused from the center of impact, but the edges of the selected region haven't moved much. It is also easy to distinguish either temperature variations or the structure of Ne versus NaCl by turning the property binding off or on. The color bar in Figure 6 shows the temperature binding used throughout. A temperature range of 0 to 300° K is mapped onto the absolute scale of 0 to 100 on the color bar.

With a click, we can remove the binding and explore shape and development.

We can thus see that even after total embedding of the cluster, the outer edges of the surrounding Ne remain quite cool, although the impact region heats to near 300° K. With a click, we can remove the cluster binding and explore the hole shape and development, thus discovering that the greatest heating of the Ne surface remains in a concentrated region below the cluster impact.

Figure 7 uses a vector glyph to show the effect of localized, on-diagonal atomic stress tensor components. It uses the same time step as Figure 5, but we've removed the Ne-layer binding and enabled the binding to the crystalline substrate atoms to see how they behave. (If bound, the NaCl cluster would be over the center of the substrate in this view.) The on-diagonal tensor elements represent stretching or compression along the component axes, depending on whether the sign of the tensor element is positive or negative, respectively. Thus, vector direction and magnitude indicate the direction and amount of on-diagonal force. Figure 7 shows that columns of atoms (perpendicular to the image) in the corners have larger components along the crystalline surface plane than columns running vertically or horizontally from the center of the image. This crystal-structure-dependent distribution of stresses, evidently due to the presence of the cluster over the center of the substrate, is brought out by the Glyphmaker bindings.

We can represent the off-diagonal stresses by adding another vector, or represent the stress tensor in a variety of other ways. These possibilities and the application of Glyphmaker to a CFD system are explored in another work.[2]

We are working to improve and extend several features of Glyphmaker. The Condi-

tional Box, for example, needs means for selecting other than spatial conditions (for example, local energy or temperature conditions), and we plan further work with linking mechanisms to highlight related variables in different views. We also plan a series of tests and evaluations by scientists and engineers using real data to refine Glyphmaker's interface and functionality.

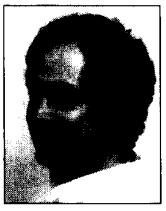
Next-generation visualization tools will need powerful and general data models.[13] Thus, we're building a Glyphmaker model that will provide two-way links, from original data to graphical object and vice versa. The latter links, especially, will permit imposing general conditions and bindings based on interaction with the visualization and selecting original data at any time for quantitative analysis. The data model, and the modules we build to use it, will also greatly increase speed in the dataflow environment. We are also developing a general approach for data visualization based on binding mechanisms — not only for the point glyphs presented here, but also for higher dimensional objects such as streamlines, surfaces, and volumes. Finally, since Glyphmaker presents discrete 3D objects that would fit naturally into immersive environments, we've extended Glyphmaker design procedures and visualizations to virtual environments. ■

Acknowledgments

This work is supported in part by grant numbers CDA-9200635 and NCR-9000460 from the National Science Foundation.

References

1. R. Ellson and D. Cox, "Visualization of Injection Molding," *Simulation*, Vol. 51, No. 5, 1988, pp. 184-188.
2. W. Ribarsky, S. Mukherjea, and E. Ayers, "Glyphmaker: An Interactive, Programmerless Approach for Customizing Visual Data Representations," Report GIT-GVU-93-26, Georgia Inst. of Tech, 1993.
3. L. Treinish, "Inside Multidimensional Data," *Byte*, Vol. 18, No. 4, Apr. 1993, pp. 132-133.
4. M.W. Ribarsky et al., "Visual Representations of Complex, Discrete Multivariate Data," *Visual Computer*, accepted for publication.
5. H. Lefkowitz, "Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Perception," *Proc. Visualization 91*, IEEE CS Press, Los Alamitos, Calif., Order No. 2245, 1991, pp. 164-170.
6. R.M. Pickett and G.G. Grinstein, "Iconographic Displays for Visualizing Multidimensional Data," *Proc. 1988 IEEE Conf. Systems, Man, and Cybernetics*, IEEE, Piscataway, N.J., 1988, pp. 164-170.
7. R. Haber, "Visualization Techniques for Engineering Mechanics," *Computing Systems in Eng.*, Vol. 1, No. 1, 1990, pp. 37-45.
8. J. Foley and C. McMath, "Dynamic Process Visualization," *IEEE Computer Graphics & Applications*, Vol. 6, No. 3, 1986, pp. 16-25.
9. J.D. Foley, "Scientific Data Visualization Software: Trends and Directions," *Int'l J. Sup. Appl.*, Vol. 4, 1990, pp. 154-157.
10. C. Upson et al., "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics & Applications*, Vol. 9, No. 4, 1989, pp. 30-42.
11. A. Buja et al., "Interactive Data Visualization Using Focusing and Linking," *Proc. Visualization 91*, IEEE CS Press, Los Alamitos, Calif., Order No. 2245, 1991, pp. 156-163.
12. H.-P. Cheng and U. Landman, "Controlled Deposition, Soft Landing, and Glass Formation in Nanocluster-Surface Collisions," *Science*, May 1993, pp. 1,304-1,307.
13. W. Ribarsky and J. Foley, "Next-Generation Data Visualization Tools," *Frontiers in Visualization*, Springer-Verlag, 1994.



William Ribarsky is manager of the Office of Information Technology Scientific Visualization Lab and associate director for service of the Graphics, Visualization, and Usability Center at the Georgia Institute of Technology. His research interests include programmerless methods for constructing visual representations of multivariate data, semiotics of visual representations, virtual environments applied to scientific visualization, statistical graphics methods for analysis and visualization of scientific data, and visualization and analysis of network data for management and planning of networking systems. Previously, he did research in computational physics, including molecular dynamics studies of surfaces and liquid-solid interfaces. He is a member of the IEEE Computer Society and the American Physical Society and has a PhD in physics.

Eric Ayers is a masters student in the College of Computing at Georgia Tech. Currently, he works in the College of Architecture as a member of the Interactive Media Architecture Group in Education. His research interests include interface design, multimedia delivery systems, and hypertext systems. Ayers received his BS in the architecture program at Georgia Tech.

John Eble is a PhD student in Georgia Tech's Department of Electrical and Computer Engineering. His primary research interests include computer architecture; Pica, a fine-grain, message passing architecture for high-throughput applications; and gigascale integration. Eble received the BCmpE degree in computer engineering from Georgia Tech. He is a student member of the IEEE and the IEEE Computer Society.

Sougata Mukherjea is a PhD student in the College of Computing at Georgia Tech. His primary research interest is in scientific, software, and information visualization. His other interests include user interfaces, graphics, and databases. Mukherjea obtained his bachelor's degree in computer science and engineering from Jadavpur University, India, and his MS in computer science from Northeastern University, Boston.

Readers may contact William Ribarsky at the SciVis Lab, OIT/Client Services, Rm. 229 Hinman, Georgia Institute of Technology, Atlanta, GA 30332-0710. E-mail contacts can be addressed to Ribarsky, bill.ribarsky@oit.gatech.edu; Ayers, zundel@cc.gatech.edu; Eble, gt0153c@hydra.gatech.edu; or Mukherjea, sougata@cc.gatech.edu.