GLYPHMAKER: An Interactive, Programmerless Approach for Customizing, Exploring, and Analyzing Visual Data Representatives

by

William Ribarsky, Jack Tumblin, Gregory Newton,
Robert Nowicki, Jeffrey Vetter

GIT-GVU-93-26
April 1994

Graphics, Visualization & Usability Center

Georgia Institute of Technology
Atlanta GA 30332-0280

Report GIT-GVU-93-26

Glyphmaker: An Interactive, Programmerless Approach for Customizing, Exploring, and Analyzing Visual Data Representations

William Ribarsky and Jack Tumblin
Georgia Institute of Technology
OIT/Client Services and GVU Center
Atlanta, GA 30332-0710
(404) 894-6148; {ccsupwr, ccsupjt}@prism.gatech.edu

Gregory Newton, Robert Nowicki, and Jeffrey Vetter
Georgia Institute of Technology
GVU Center
Atlanta, GA 30332-0280
(404) 894-6148; {gregn, robn, vetter}@cc.gatech.edu

Category: Systems **Format:** Long

Abstract

Glyphmaker is an interactive system for data visualization and analysis that is built in a dataflow environment. Glyphmaker provides a quite general platform for user-defined visualizations, one that can be extended in many ways and that, in particular, offers the first stage for an integrated visualization/analysis system. It is well-suited for the exploration and analysis of multivariate, highly correlated 3D data, which is increasingly the form of modern data. This is both because of the glyph-based visual representations and the interactivity that are built into the system. To summarize its capabilities, Glyphmaker allows the user to: (a) design glyphs by assembling them from different shapes; (b) choose binding sites on these shapes; (c) bind user data attributes to glyph elements; (d) conditionally isolate certain variable ranges; (e) interactively view, explore, remove, or rebind the resulting glyph sets as solid 3D objects. Each of these capabilities has a graphical user interface and is built into Glyphmaker in a modular form. The Glyphmaker approach of arranging data into data objects that are then bound to 3D graphical objects (glyphs) or manipulated by interactive tools is quite extensible in terms of visual representations and analysis methods.

Glyphmaker: An Interactive, Programmerless Approach for Customizing, Exploring, and Analyzing Visual Data Representations

Category: Systems **Format:** Long

Keywords: glyphs, scientific and engineering visualization, information visualization, interactive systems, multivariate analysis, graphical user interfaces, dynamic mapping of variables

Abstract

Glyphmaker is an interactive system for data visualization and analysis that is built in a dataflow environment. Glyphmaker provides a quite general platform for user-defined visualizations, one that can be extended in many ways and that, in particular, offers the first stage for an integrated visualization/analysis system. It is well-suited for the exploration and analysis of multivariate, highly correlated 3D data, which is increasingly the form of modern data. This is both because of the glyph-based visual representations and the interactivity that are built into the system. To summarize its capabilities, Glyphmaker allows the user to: (a) design glyphs by assembling them from different shapes; (b) choose binding sites on these shapes; (c) bind user data attributes to glyph elements; (d) conditionally isolate certain variable ranges; (e) interactively view, explore, remove, or rebind the resulting glyph sets as solid 3D objects. Each of these capabilities has a graphical user interface and is built into Glyphmaker in a modular form. The Glyphmaker approach of arranging data into data objects that are then bound to 3D graphical objects (glyphs) or manipulated by interactive tools is quite extensible in terms of visual representations and analysis methods.

1 Introduction

We created Glyphmaker as a prototype environment to attack the general problem of giving users the means to visualize, explore, and analyze data containing several (or more) variables. In particular this requires giving users tools to customize (a) how the data is represented visually (perhaps at the level of the individual datum) and (b) how much of the data is presented at one time or which variables are presented. For the purpose of successful exploration and analysis, it is quite important that the user be able to apply these customizations interactively and iteratively, based on current views of the data, and that, aided by these customizations, she be able to pick out patterns in and relations between the data. We will show in this paper how Glyphmaker is the first stage in a next generation of information visualization tools that will be necessary for the successful analysis and understanding of modern data.

Current visualization approaches have made major strides in providing high-functionality, user-oriented tools. Interactive editors, point-and-click visualization tools [1], and dataflow systems [2] have freed users from the burden of subroutine-based programming tasks requiring expert knowledge of 3D graphics techniques, interaction techniques, and visualization design. In fact, the dataflow systems are extendible enough to provide platforms for further improvements, as is the case with Glyphmaker. However, a major problem with current versions of all these systems is that they deal with fixed vocabularies of visualization techniques that the user can extend only by knowing interactive graphics programming. Thus for example, although the Wavefront Data Visualizer (a point-and-click system) [1] may have tools or the Iris Explorer (a dataflow system) [3] may have modules that will usefully visualize and analyze certain computational fluid dynamics (CFD) behavior by probing with streamlines, particle traces, or other tools, neither has the means to visualize several variables at once (e.g., momentum, pressure, temperature, internal energy, etc.) throughout the 3D space. One may be able to probe certain variables

by slicing the CFD space with selected 2D contours while at the same time displaying streamlines or particle traces, but the behavior of these variables in all 3 dimensions cannot be probed and compared at once nor can one use representations other than contours (and a few others). Further if one wanted to study, say, molecular dynamics (MD) simulations, one would first of all have a restricted set of visualization techniques since available visual representations, such as surface or volume rendering of scalar or vector fields, are limited in the amount of detail they can present and tend to be optimized for a few well-known applications, such as CFD or finite element analysis. It would also be harder to apply tensor analysis tools worked out for CFD [4] to MD or other data since these assume that the data is arranged in a certain structure common to CFD data. At the least the user would have to program a filter to arrange her data into a format appropriate for these tools.

We have created Glyphmaker to help solve these problems. First, Glyphmaker allows the user to develop visualizations from a set of visualization primitives, perhaps combining primitives to build visual representations of whatever variables she selects. Since she can do this with a series of point-and-click interfaces, she is relieved of any interactive graphics programming duties. This approach is highly flexible and extendible to any type of data brought into Glyphmaker; as the set of primitives is expanded so are the possibilities for visualization and analysis of all types of data. Thus tensor analysis techniques taken from CFD approaches, which we are now adapting for the Glyphmaker environment (see the last section of this paper), will immediately be available for any type of data brought into the environment. Second, Glyphmaker is well-suited for multivariate, highly correlated 3D data. This is both because of the glyph-based visual representations and the interactivity that are built into the system. The glyph-based representations, because one can use visually orthogonal attributes (e.g., shape, color, size, etc.) each of which could be bound to a different variable, are excellent for depicting several variables simultaneously and comparing them. The interactive binding coupled with related forms of interactivity will

significantly enhance one's ability to analyze complex data. We will expand on these last two points in detail in the sections below.

2 Attacking Modern Data

Data from modern observations, experiments, and computer simulations are growing in complexity and size. This trend will accelerate. Already MD simulations on the CM-5 model as many as 250 million interacting particles, each changing position over time [5]. On the horizon are simulations of a billion particles or more. (Just a few years ago a simulation with 10,000 particles was considered large.) The change in scale and complexity indicated in this one example is reflected in many other areas, especially Grand Challenge problems.

Modern data is complex in that it tends to have many variables that are closely interdependent, that vary in all 3 spatial dimensions, and that change markedly in time. Analyses of these data must have a significant visual component and must allow the comparison of variables as they develop in time. Furthermore, comparison of data from different sources in the same visualizations will become commonplace, as Treinish [6] and others have pointed out. These sources may be different computer simulations or, increasingly, simulation and observation. An example of the latter is comparison of satellite images of ozone hole development over the Arctic region with simulations depicting amounts of hydrocarbon build-up and ozone-depleting reactions. As data size grows it will eventually be fruitless to fully pre-analyze these data; that will occur on the fly and most likely as a result of what the user has seen in a previous visualization. At this point interactive visualization will be essential and a complete and integrated visualization/analysis approach will be needed.

3 Using Glyphs

Glyphs are graphical objects whose attributes (e.g., position, size, shape, color, orientation, transparency, etc.) are bound to data. Most current applications involve 0-D glyphs since the glyph/icons describe points and their associated properties. There is, however, no intrinsic reason for this restriction, and one could have 1-, 2-, or 3-D glyphs with appropriate bindings. (In the following sections "glyph" signifies the 0-D glyph; we will return to its higher dimensional counterpart in the last section.) Glyphs can clearly be effective in depicting discrete data and properties from, say, macromolecular dynamics simulations, but they have also proved useful in representing variables such as wind speed and direction in atmospheric dynamics simulations and observations [7]. Glyphs owe their effectiveness to the ability of the eye-brain system to discern finely resolved spatial relationships, patterns, and differences in shape. They thus allow the user to display and correlate several variables at once; a researcher can, for example, choose to distinguish two variables by binding them to the visually orthogonal representations of shape distortion and pseudocoloring. In a recent application to MD results, it has been shown [8] that the complex, 3-D structure of atoms and all components of the stress tensor acting on each atom 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 quite a different context, Ellson and Cox [9] have shown that glyphs are quite useful in depicting the flow patterns that result from finite element simulations of hot plastic injected into a mold. In this work 100-200 glyphs were placed according to a finite-element grid; each glyph was a 3D object with the velocity in a finite element represented as a shape distortion of appropriate magnitude and direction, and temperature and pressure represented

by different color scales on the glyph and its base, respectively. Other important work employing glyphs concerns the use of color or gray scale "icons" that merge color and texture perception to represent multiple parameters [10,11]. Here the icon is a generalization of the single pixel to objects having perceivable features and attributes. Typical icons are stick figures or other simple collections of lines with such characteristics as line orientation, length, etc. representing different parameters. Color coding is also used to bind additional parameters to the icons. The method has been used, for example, on satellite imaging or MRI data [12] where thousands of small icons are used to build up collective perceptions of texture and contour.

Some final examples in the current use of glyphs come from work by Haber [13] on the use of tensor glyphs in visualizing data from engineering mechanics. All these and many other examples show the range and power of glyph representations. They are often superior to traditional surface and volume techniques in, for example, displaying and elucidating 3D vector or tensor fields. It is apparent from this selection of applications, that glyphs can be effectively used to depict a variety of data, including continuum data, in addition to the obvious data (e.g., molecular modeling applications). Taken together, all this research explores a wide range of glyph representations starting from a few hundred highly detailed glyphs [9] and extending to several thousand glyphs where it was most important to perceive "gestalt" or collective effects rather than individual glyph characteristics [10]. With Glyphmaker we attempt to make it possible for the user to effectively represent, visualize, and analyze data in all these categories.

4 Glyphmaker Background

The historical motivation for our approach comes from work of Foley and others. In a system developed for dynamic process visualization, Foley and McMath [14] showed how

non-programmers could construct and customize dynamically updated scenes displaying process information. This information came from real world environments such as factories, power plants, and refineries and 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. The user was thus able to create a customized, graphical representation describing a set of time-varying processes variables and to do so without intensive programming. Later Foley [15] proposed extending this idea to more general visualization schemes, such as with scientific data. Here the data might be 2D or 3D, and the user might need to analyze several variables at once.

The recently developed dataflow approaches provide a general, compatible framework for Glyphmaker. Although dataflow and similar schemes based on a visual programming style had been investigated for a period of years, the first widely-used dataflow system was AVS [2]. Now there are several additional systems built around the same basic dataflow concept including the SGI Iris Explorer , IBM Explorer , Khoros, and apE . They all allow you to build your own applications by constructing graphical networks of modules--such as for data input, filtering, transformation to geometry, pseudocoloring, and rendering. We have chosen the SGI Iris Explorer as our specific developmental platform. However, since we used Motif to build the Glyphmaker interfaces, it would be straightforward to port to another dataflow environment; the main change necessary would be in revising the way data is transformed into the dataflow internal data structure in the Read module

5 The Glyphmaker Environment

Foley [16] has recently given a useful general definition of visualization: the binding (or mapping) of data to a representation that can be perceived. These bindings might be visual,

auditory, tactile, or any combination of these. The visual bindings could be further subdivided into sets of marks (e.g., points, lines, areas, volumes) that express position or pattern and retinal properties (e.g., color, shape, size, orientation, texture) that enhance the marks and may also carry additional information [17]. This is the approach we present here. The auditory bindings could be used for iconization or to map data onto elements such as pitch, spatial localization, etc. The tactile bindings could be, for example, through force feedback devices, controls with multiple degrees of freedom, and so on. In the last section of this paper we will discuss how Glyphmaker can be extended to fit this more general definition of visualization.

What we have described implies the visualization environment of Fig. 1. If we have a set of data objects and separately have a set of graphical objects, we can delay decisions about how they are interactively joined together (in the Binder) until those decisions need to be made—as the data is being viewed and explored. This is the most flexible way to create data visualizations. Furthermore, if we attach a 3D graphical editor to define and group graphical objects (such as glyphs) for subsequent binding, we can put all these modules in a feedback loop that processes through the user's brain to iteratively refine the visualization or to focus on certain aspects of the data. This is the structure of Glyphmaker.

In keeping with our desire to build upon, rather than replace, the functionality already in the dataflow system, we have used existing Explorer modules wherever possible. In particular, using the appropriate modules we can convert data into surface or volume geometry and then pipe it to the renderer for simultaneous display with glyph representations, thus achieving useful mixed representations of the data. Our approach allows us to concentrate on the novel aspects of our work and to incorporate our additions easily into the Explorer system. The modules that we built are written in C; we used Motif for the user-interface parts and GL for the graphics parts. The GL graphics appear in only

the Editor and are thus quite modular. We expect eventually to port the graphics to Open GL in order to enhance the portability of our codes.

The Glyphmaker system has 4 main interactive parts:

- The Read Module, which converts input data from a user-specified format to the Explorer internal format so that it can be read-in by any existing Explorer module
- The Glyph Editor, which creates glyphs and arranges them, if desired, into compound glyphs
- The Glyph Binder, which allows interactive bindings or binding changes between data variables and glyph elements. It also allows one to quickly alter the range of data variables or to alter the range of change in the glyph elements as the data varies from minimum to maximum. Finally, the Binder permits the saving of new bindings or reloading of previous ones.
- The Conditional Box, which allows the user to set conditions on the data, reclassify according to those conditions, and then employ the new set of bindable properties that result

We will describe these parts below. In addition, some Explorer modules are important enough that we integrate them into our system; two main modules here are the Generate Colormap module and the Renderer module. Fig. 2 shows how the parts of Glyphmaker fit together. Note that the structure in Fig. 2 fits into the general structure shown in Fig. 1. Data management is handled through the Data Transcriber and the Header/Label Transcriber, and the data model is organized around the Explorer data structures and the Header/Label file.

The Read Module

To deal with the need to retrieve ever-growing amounts of data from a variety of formats and place them in a common structure for easy comparative viewing or for analysis with a common set of tools, we have developed a simple "self-describing" data structure that is consistent with other work on this problem [6]. Each self-describing file has a header, which describes the data and its format, followed by the data itself. The data is assumed to be arranged in *data objects*, each object having a set of position coordinates, associated properties for that position, and classes. (See Fig. 3.) The classes might be chemical species if one were analyzing, say, organic molecule structures. As we will show below, the Conditional Box can also be used to make new classes—in which case the header information is updated accordingly. The data objects do not have to be combined together in one file; one could have separate position and properties files. In the header each variable or property is labeled with a name, primitive type, number of instances, minimum value, and maximum value. The label, minimum, and maximum values are important information that is passed along to the Glyph Binder (and can be changed there). The number of records following in the file 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 what the file contains. The data in the file then follows in the order given in the header. Headers (and the data following) can be placed anywhere in the file or in separate files. They can also be placed together at the top with all the data following the last header; this allows one to merge files and still keep header information accessible.

We chose this very simple format because it was easy to understand, easy to create, and it fulfilled the basic needs of the Glyphmaker. In the future we will modify the Read Module to handle more complicated data formats than described here, but we will try to do so while retaining the basic simplicity and intuitive feel of our present format.

The 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, seen in the upper right hand corner of Fig. 4: Glyphs, Views, Mode, and Action. The Glyph menu items are: lines, spheres, cuboids, cylinders, and arrows. The Views menu helps the user manipulate the glyphs in 3D space by providing menu items for selecting either perspective, top, side, or front views singly or all four views simultaneously. The Action menu has bind, render, ungroup, and clear items for (respectively) 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 glyphs, since creating good glyphs takes care and testing.

Each glyph has several elements that can be bound to data. For example, one might bind the radius, position (x,y,z), color, overall size, and transparency of a sphere to individual data variables. Shape, such as elongation or shortening along each axis, might be bound to additional variables. Similar sets of elements are available for all glyphs in the Glyphmaker library. One must choose which elements to make active for the next phase; only these elements appear in the Glyph Binder menu. To do this one clicks on a glyph with the right mouse button; a window pops up showing all elements for the glyph as toggle buttons. These buttons allow the user to set or unset the active elements easily. One can also bypass the active element selection entirely; in this case only a smaller default set of elements appears in the Binder. We take these steps to increase overall speed in setting up visual representations and interactivity in the Binder—each glyph has several elements so that if all were displayed the menus could be quite long, and the time needed to find and manipulate a given element would increase.

The Mode menu has two modes for active glyphs: body axis or orientation axis. The body axes stay fixed with the glyph while the orientation axes may be translated and rotated with respect to the glyph. The orientation axes allow one 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.) For precise translation and rotation, one can use the two sets of sliders shown in Fig. 4. A grouped glyph is shown in Fig.4—in this case the cylinder and small sphere are grouped to make a vector and 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.

The Glyph Binder

After the user chooses the bind option in the Editor, a Binder window pops up that shows all the active elements and the variables as toggle buttons. (See Fig. 5.) 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. As shown in Fig. 5, the user can also get a pop-up window with binding information by clicking an already-lit toggle button. Bindings can also be deleted and reset with the mouse. Attached to each variable and active element are text widgets specifying the minimum and the maximum value for that variable or active site. Default values, taken from the header file, appear initially in these widgets. The use of defaults means that bindings can be established in a few clicks, and the results displayed immediately. Of course, the user has the ability to change any of these values, and the result will be transmitted to the renderer for update of the display.

The Binder design emphasizes "highly responsive controls", i.e., the interactivity and ease of control necessary for effective data exploration. The variables and elements are in separate scrollable lists (Fig. 5) so that one can quickly find the variables/elements of

interest, even if the lists are longer than the window size. The Binder window is also resizable and movable using the mouse so that one can easily rearrange the display real estate—this is important since the Binder will often be used concurrently with the renderer for interactive refinement of data visualizations. The Render button at the bottom of the window pipes any changes made in the Binder to the renderer for display. The OK button dismisses the Binder, but with data/glyph bindings remaining in place for the interactive visualization. At the top left of the window, the File button has a pull-down menu with the usual save/reload options. Saving and loading of 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. In addition to the local presentation of binding information obtained by clicking on a particular toggle button, the user needs a global view—especially when several glyphs are involved. To handle the global view, we are near completion and will soon implement a binding visualization panel in the Binder window that will illustrate the overall binding structure. We are also investigating ways to display and highlight glyphs (and active elements) in the Editor window as they are being bound. This capability will be especially worthwhile for users who are bewildered by a plethora of glyphs and attributes or who have loaded a set of bindings that they have forgotten or that have been generated by someone else

Conditional Box

The Conditional Box is a specific and particularly useful instance of a concept from statistical graphics called *focusing* and *linking* [18]. In general it involves interactively classifying any data variables according to the condition

$$A_i \leq X_i \leq B_i$$

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. The condition generates a new hierarchical classification structure with additional

classes (The original classes are moved down one level, under the branches for inside and outside the box.) One can then bind according to these new classes. The data objects have certain global elements (e.g., position, overall size, color, transparency) that can be bound at once for all data objects in a conditional class. On the other hand, users can, if they wish, bind data variables to individual glyph elements.

Our first implementation of the Conditional Box is a 3D box that can be moved and sized to enclose a subregion of the data after it has been selected for display and bound to particular glyphs. The box allows one to distinguish what is inside from what is outside. One can then choose new glyph bindings--for example, new colors or glyph shapes for what is inside (or outside) the box. One can even remove bindings from (or make semi-transparent) data so classified to focus on or even isolate an interesting spatial region.

The Conditional Box module receives information from the Explorer data structure and from the Geometry Generator to generate a box with the appropriate scale and position with respect to the data. (See Fig. 6.) Then data and box geometry are piped to the renderer for simultaneous display. The interactive controls shown in Fig. 6 can update the box geometry and position it with respect to the data.

The Conditional Box provides a powerful new weapon in the visualization arsenal that fits naturally into the Glyphmaker approach. It goes beyond traditional scientific visualization techniques such as zooming, panning, or slicing, that are examples of subset selection and that are already part of standard visualization systems (such as dataflow or point-and-click). It provides additional focusing techniques in order, for example, to bring out parts of the data that may be in the same spatial area as other data that is not of interest or to bring into view data that would otherwise be surrounded and hidden by other data. Moreover, it can be extended to make possible additional linking and focusing capability.

Linking is a process from statistical graphics where multiple views, each containing partial information about a complex data set, are linked (e.g, perceptually) so that the user can integrate them into a coherent picture of the data relations [18]. Views can be linked in a time sequence or by using techniques to bring out features in simultaneous displays. Smooth animations provide linking in time-sequenced views. The interactive feedthrough of multiple records in dataflow systems and the current or imminent availability of record and playback features (for renderings that are too slow for real-time display) in many dataflow systems make smooth animations feasible. Beyond these techniques there are linking methods that have proved quite useful for multidimensional point data [18] and will undoubtedly be effective for other data as well. These methods involve bringing up simultaneous multiple views (or quickly alternating views) and emphasizing corresponding parts with a distinctive color, change in brightness, different symbol, etc.. For example, we might use highlighting of the same atom or collection of atoms to link simultaneous views, each displaying different atomic properties or a different spatial orientation of the system. This will be a straightforward thing to do with the Conditional Box in Glyphmaker. We are designing the capability to change a conditional region and have the changes flow through preset bindings to automatically update a display of the data. As we develop general Conditional Boxes that will handle any variable this will be a powerful capability. One can also plan probes of correlations involving sets of conditions or more complex conditions than presented here.

6 Glyphmaker Applications

We have chosen two applications to show the capability of Glyphmaker on disparate data. One application is for the visualization and analysis of an MD simulation of a material system. Here the data objects are individual atoms that move and change patterns over time

due to their interactions. The other application is from a CFD simulation where unsteady, 3D fluid flow must be studied in detail. In this case the data objects are grid elements that tile the 3D region of interest and whose properties make 3 dimensional patterns.

Visualizing Molecular Dynamics

In MD simulations of materials systems, 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. It is necessary to analyze the complex interrelation of stresses, heating, and dynamic behavior in order to understand the processes that occur in these simulations. The simulation presented here was run on a Cray Y-MP at the DOE National Energy Research Center by Dr. Hai Peng and Professor Uzi Landman of the School of Physics, Georgia Tech [19].

The simulation depicts a "nano-cluster" of 64 atoms of NaCl hitting a Ne surface and becoming embedded in the surface. Although the system was initially quite cold (50 ^{o}K) and the impact velocity moderate (3 Km/sec), upon impact the cluster and surface in its vicinity heats up greatly with much disorganization and eventually some vaporization. We have chosen simple sphere glyphs for binding to the separate components of the system (Na and Cl separately in the cluster, dynamic Ne atoms in several layers stacked on several layers of crystalline substrate), but we could have easily used, say, spheres and cuboids to distinguish between atom types. Properties of this system include localized atomic kinetic energy and temperature and also the atomic stress tensor--in the visualizations presented here we have bound the temperature to the color of the glyph for each atomic type. We have applied the Conditional Box to the system at a time step just before the cluster penetrates the surface (Fig. 6). This is to avoid the common problem with large scale simulations; often important features are embedded in the system and hidden from view, as

shown in Fig.7 where the cluster has fully penetrated the Ne (with binding of color to local temperature turned on). Using the interactive rotation and zooming capabilities in the renderer and the Conditional Box controls (shown at the lower left in Fig.6), we adjust the semi-transparent box so that it encloses the cluster and the region of the Ne surface that the cluster is 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 semi-transparent). We can also retain the Conditional Box classification for later time steps; Fig. 8 shows the selected region as the cluster embeds itself. Here we have bound local temperature to sphere color (except for the NaCl cluster since its high temperature would dominate the mapping), on-axis atomic stresses to vectors (see discussion below), and the magnitude of off-axis shear stresses to vector color. In this case, we can see (by comparing with previous time steps) that Ne atoms have diffused out from the center of the impact but the edges of the selected region have not yet moved out much. We can also see that the Ne atoms directly under the cluster gain high temperatures and that the vectors indicate large compressive forces on the NaCl cluster but smaller forces on the soft Ne substrate. Finally, we show in Fig. 9 the final step, after the cluster is embedded, in a time sequence of the full system with the cluster removed (achieved by just clicking off the NaCl bindings). By collecting several images in the time sequence, one can explore the shape and development of the hole. One can also use the interactive rotation and zoom controls in the renderer to study the 3D structure at any time step. Though it wasn't apparent when the cluster was visible, one can now see clearly that the greatest heating of the Ne surface remains in a concentrated region below the cluster impact. The color bar in Fig.9 shows the temperature binding used throughout. A temperature range of 0 to 300 ^{o}K is mapped onto the absolute scale of 0 to 100 on the color bar. We can thus see that even after total embedding of the cluster, the outer edges of the surrounding Ne remain quite cool. However, Ne atoms in the center are between 200 and 300 ^{o}K--much greater than the vaporization temperature of Ne.

In Fig. 10, we show how more properties can be built into the visualization by depicting the effect of localized, on-diagonal atomic stresses using a vector glyph. The same time step as in Fig. 9 is shown, but here we have removed the binding to the Ne layers and enabled the binding to the crystalline substrate atoms below so that we can see clearly how the latter behave. Since the stress tensor is symmetric, it has only 6 components that can be divided intuitively into on- and off-axis sets. The on-axis set represents stretching or compression along the component axes depending on whether the sign of the component is positive or negative, respectively; the off-axis set represents shear forces. We can depict the on-axis forces both with a grouped glyph consisting of a sphere and vector (not shown here, the spheres locate atoms precisely) and with vectors alone. One can switch between the two representations with a few clicks in the Binder. The direction and magnitude of the vector indicates the amount and direction of on-diagonal force. (The magnitude of the vector is also proportional to the pressure at that point). Let us label the horizontal and vertical directions in Fig. 10 as x and y with z perpendicular to the image; the NaCl cluster, if it were visible, would be directly over the center of the image. Fig. 10 shows that columns of atoms (in the z direction) directly below the cluster (and below rows along x and y that intersect these atoms) have vectors mostly pointing downward indicating compressive forces along z. However, it also shows that columns that are more along the diagonals from the center of the image have larger components along the crystalline surface plane. This crystal-structure dependent distribution of stresses is brought out by the Glyphmaker bindings and could not be noticed except in a 3D visualization. We could easily represent the off-diagonal stresses by adding another vector with a different shaped head, or the stress tensor could be represented in a variety of other ways.

We could have tried other binding combinations to study the relation between several variables at once. For example, we could easily represent the shear stresses by adding another vector with a different shaped head, or the stress tensor could be represented in a

variety of other ways. This is the advantage of Glyphmaker since it allows one to interactively choose, discard, or refine visualizations.

A CFD Application

We next apply Glyphmaker to CFD simulations generated by Marilyn Smith of the Georgia Tech Research Institute to investigate turbulence on large tilt-rotor aircraft. These are simulations of non-steady, 3D fluid flow with dynamic changes in fluid structures, such as vortices, that must be followed in order to determine the sources and effects of turbulence. Non-steady, 3D simulations are among the most important and most difficult to investigate [20]. They have complex spatial structures and patterns involving several correlated variables that should be well-suited for Glyphmaker.

The simulation presented here was carried out because investigators have found that hovering aircraft sometimes re-ingest their own downwash, forming a "fountain" of backwards-circulating air that increases noise, wastes engine power, and may affect stability and maneuverability. Smith's "fountain effect" simulation finds time-sequences of 3D vector and scalar fields that may hold small vortices obscured by large, steady-state flows. These effects have 3D structure and change in time. The simulation presents flow around and below a tilt-rotor with the flow space bounded on one side by the nacelle of the rotor and below by a reflecting surface representing the ground. For accurate results a finely spaced, two zone 3D grid must be used; the first zone is 100 x 42 x 34 and the second is 100 x 42 x 37 for a total of 300K grid elements. Since a complete simulation would require many time steps, this can be a problem of major size. The properties derived during the simulation include temperature, momentum vector, internal energy, density, vorticity, and pressure. Typically, one would use Plot3D [21] or a similar package to visualize and study the simulation. Although they provide several useful probes of fluid flow structure, major limitations of these packages include the difficulty of exploring

extended 3D structures or isolating 3D regions of interest (usually only cut planes are used) and the inability to display more than a few variables at once. Glyphmaker can overcome these limitations.

In Fig. 11 we display the two zone grid with cuboids representing zone 1 and spheres representing zone 2. We could easily remove one zone for clearer viewing of the other one. To handle large sets of data such as these, we have created a "Sparser" module in the Glyphmaker system that retains every n_uth, n_vth, n_wth grid index for the u, v, w directions, respectively. Thus if, as here, we retain every 4th element in each direction, the number of elements in each zone is reduced by a factor of 64. Since it is in the Glyphmaker system, the Sparser module can be used on the MD data or any other data that is read in. We plan to make the module interactive so that users can choose levels of data reduction based on what they see. The Sparser module is the first of a class of interactive data reduction schemes that we will investigate within the Glyphmaker structure.

The position of the tilt-rotor blade is indicated by the higher density concentration of grid elements extending horizontally across the center of Fig. 11. The tip of the blade is at the right end of the grid element concentration , and the nacelle housing (unseen) runs vertically along the left side of the grid. The red cursor arrow in Fig. 11 points at the tip region. In Fig. 12, we use spheres and vectors bound to grid element positions to represent several variables. For each element we have then bound momentum to the vectors, pressure to the vector color, and density to sphere color. We use the same color spectrum as in Fig. 10. The density in Fig. 12 is close to 1 (the scaled value of the stationary flow density in this simulation) and is thus one color except for a region at the tip where it is lower. This is a vortex region. The pressure is slightly lower than the steady flow value directly above the blade and slightly higher directly below. We see that the momentum vectors run parallel to the blade underside and then begin curving downward at

the tip and at the nacelle surface. This is an aspect of the fountain effect downwash mentioned above. It turns out that the tip region spawns vortices that significantly affect the structure of the backwash. If we bound vorticity magnitude to, say, grid element size, we could pick out these regions more clearly. These regions are not easy to see with tools like Plot3D since they are small and tend to be obscured by large steady state flows, but with tools like our Conditional Box, we could isolate or focus on them for closer inspection. We could, for example, rebind to show only the momentum patterns in the vortex regions thus showing the dynamic structure there.

We are now working on extensions to allow simultaneous binding and display of streamlines, particle traces, and other fluid flow representations along with the glyphs shown here. (See the last section.) These will greatly increase the power of Glyphmaker since they will be available not only for fluid flow data but for any data read into the system.

7 Conclusions and Experiences

Glyphmaker provides a quite general platform for user-defined visualizations, one that can be extended in many ways and that, in particular, offers the first stage for an integrated visualization/analysis system. We have shown that Glyphmaker is well-suited for multivariate, highly correlated 3D data, which is increasingly the form of modern data. This is both because of the glyph-based visual representations and the interactivity that are built into the system. To summarize its capabilities, Glyphmaker allows the user to: (a) design glyphs by assembling them from different shapes; (b) choose binding sites on these shapes; (c) bind user data attributes to glyph elements; (d) conditionally isolate certain variable ranges; (e) interactively view, explore, remove, or rebind the resulting glyph sets as solid 3D objects.

Our experiences with Glyphmaker have shown some areas that need improvement. Since it can generate thousands of 3D objects for complicated datasets, Glyphmaker often does not have the response to commands that we desire. We are revamping the internal data structure to make it use the Iris Explorer pyramid structure more efficiently, and we believe that this will improve interactivity. However, there is always likely to be a slowness of response in Glyphmaker for data past a certain size and complexity, although this boundary should move farther away as workstation performance improves. The user will be helped by the Glyphmaker attribute allowing her to choose subsets of data and thus keep response times reasonable. We will discuss in the next section some additional capabilities that we are developing.

The Glyphmaker user interface needs improvement, especially in the binding of glyphs. Users can easily generate a multitude of glyphs and bindings and get lost in the process. We are designing some graphical ways to show more clearly which glyph elements are bound as one operates the Binder and also to show the overall set of bindings. However, the multitude of binding combinations that can occur, especially as we enlarge our set of primitive glyphs, belie a deeper problem. To an extent this problem occurs for any highly flexible system of tools; having a large number of choices may confuse non-expert users. With Glyphmaker these users may only try a few familiar visual representations and leave large areas of the binding space unexplored. We discuss in the next section how this problem can be attacked by developing organizations of graphical objects according to their best uses and also by developing expert system approaches.

8 Future Directions

We are building, planning, or envisioning a wide set of next-generation tools based on Glyphmaker or its concepts. As we have stated above, we are extending the Conditional

Box concept to be used on any variable in a data object. In addition, there is no reason to restrict users to just one Conditional Box; they should be able to generate multiple boxes and choose several ranges, if need be. Finally, of course, one should be able to apply conditions more complicated than the selection of variable ranges. These might involve algebraic expressions for data variables or correlations between two or more Conditional Boxes. We are revamping our data object structure to make these developments easier and more efficient.

We are also extending our idea of glyph bindings to include probe glyphs placed by the user at or near data objects. As [22] and others have shown this allows the user to take a compound glyph describing a number of variables at once and place it at several key locations to track the development and structure of important processes. [22] have shown that this works quite well in following the tensor fields associated with fluid flow. We believe that probes like these will be generally useful for tensor fields of all types, especially when combined with our other glyph binding methods. It appears that a straightforward extension of Glyphmaker will allow probes like these.

We are also taking the logical step of making higher dimensional glyphs available within Glyphmaker. Although we and others have demonstrated the wide applicability (and sometimes superiority) of point glyphs, there are cases where other representations are more effective. For example, ribbons are best for showing twisting spatial structures or flowlines that vary in time. Surfaces or volumes may be best for bringing out the general features and patterns in certain continuum data, especially data that is fairly smooth. It would be advantageous to bring these representations into Glyphmaker and make use of the interactive binding and conditional focusing and linking that we value. Also we will be able to build mixed representations directly in Glyphmaker (e.g., surfaces arranged with point glyphs) that are customized for the data at hand.

To handle higher dimensional representations, we can take a cue from spreadsheets, which allow the interactive binding of data sets to lines, columns, pie charts, etc. The concept we have followed so far is "one-to-one" binding of data objects to point glyphs. (Actually we also allow "one-to-many" since the Glyphmaker binder allows any number of glyph elements to be bound to one variable.) We are extending this to include the concept of "many-to-one" (i.e., many data objects to one glyph). For example, the glyph might be a ribbon with associated elements of color, width, or a set of transverse spatial dimension that are orthogonal to the ribbon trajectory. The latter, along with color, have been used by [4] to depict continuous tensor fields. The appropriate data in this case would be a collection of data objects along a trajectory. Thus whether the glyph is a ribbon, surface, or volume it implies a particular data structure for the data objects and also a data model that allows the handling of the glyph in the same way in the editor and the binder no matter what its dimensionality. Beyond that a particular data preparation is also implied. It doesn't make sense to bind data to a ribbon if it isn't processed as a trajectory (e.g., computed as a streamline or hyperstreamline in the case of CFD). The best way to ensure that trajectories are available when needed is to provide the facility to compute them from more primitive data within the visual analysis system. This approach implies an analysis model that must be integrated with the visualization tools. Similar considerations hold for surface and volume data. In principle it appears that a data model and an appropriate set of higher dimensional glyphs with elements can be developed and inserted into Glyphmaker. This will provide the glyph-based approach with the widest possible set of visual representations for depicting all types of data.

A program has also been begun to extend the Glyphmaker concepts to virtual environments [23]. This is a natural extension because the 3D graphical objects can easily fit in and be manipulated from the virtual environment. The goal is to allow users to design their own virtual spaces including visual representations while inside the virtual environment. A

further goal is to allow them to customize and apply interactive tools within the environment. Interesting questions arise as to the complexity of the graphical interface that the user can effectively employ within the virtual environment and what are the best analogs to the menus and point-and-click methods that Glyphmaker employs. A direct manipulation interface for binding of data objects to graphical objects has already been designed within this environment [23] and its effectiveness is being tested. It is also natural to consider auditory modes here; we are looking at both voice commands and the binding of data to sound ranges. The latter especially could be brought into Glyphmaker and would serve Foley's broadened definition of visualization mentioned at the beginning of Sec. 5. Finally, the performance requirements for effective update and interaction within the virtual environment put a great strain on the capabilities of graphics workstations. The user needs tools to cull the number of polygons in a scene and to retain only those of interest. We are developing an extension of the Conditional Box idea to allow the replacement of chosen regions of the data with simple polygons whose elements could be bound to average properties of the data inside. With tools like these users could customize their own level of detail, retain interactivity, and still keep a feel for the overall structure of the data. We could also bring these extensions in the Glyphmaker environment.

Since it is organized around libraries of 3D primitives that are used to build visualizations of data, Glyphmaker is a good platform for studying the semiology of data visualizations. It might be argued, for instance, that a fairly limited set of simple glyphs is all that is needed for most visualizations. If this is so, it would be a great advantage in the visualization system design (especially in developing examples to guide users in designing their own visualizations) since the number of possible combinations for visualizations (though still very large) would be drastically cut down. We will investigate this premise by studying various types of data, previous representations of these data, and by gathering expert opinion from researchers who use these data. There has been, in fact, a good deal

of work already done on developing rules for visualization design [17] including some outstanding recent work that investigates glyph-based structures more closely [24]. Human visual perception issues play a significant role in these studies. We will base our work on this previous work, adapting it to our glyph-based environment. However, we will attempt to go beyond this set of rules by developing an analytic model to give some quantitative weight and ranking to perceptual attributes. This model will be of necessity crude to begin with, but it could provide us with a valuable measure to determine the relative importance of glyph structures to add to our visualization library, and it could be invaluable in developing the expert advisor we discuss next.

Even a limited set of simple glyphs, with all the grouping possibilities and all the possible combinations of bindable elements, provides an almost uncountably large set of visual representations. Our typical user may be overwhelmed with all the hard-won flexibility of our system and consigned to trying only a very limited number of glyph mappings that may not meet her needs. This possibility has been recognized, and there are now some mostly rule-based systems that use artificial intelligence techniques to automatically generate visualizations. Presently these systems tend to use "generate and test" approaches that build from a set of graphical primitives using rules for generating visualizations and then test each generated visualization based on a set of criteria [17, 25].

We envision that the glyph-based methods might be used with various AI techniques to build a useful expert advisor. Besides the above generate and test approach, it has recently been suggested that [16] a "case-based reasoning" approach [26] might be particularly effective. This approach has not been used for scientific visualization and analysis, but we think it holds promise. For one thing, the approach of scientists for representing their data tends to be case-based; they tend to work from previous examples, modifying them to their needs. There is also a large and growing list of scientific visualization examples available

from which to choose, and our Glyphmaker environment, as we propose to extend it, will allow us to produce many of these cases. Finally, whereas the success of the generate and test approach depends on validating a step-by-step procedure that can be quite involved for complicated visualizations, the case-based approach should require less involved modifications of existing examples.

References

1. Wavefront Data Visualizer User's Guide, Version 3.0. Wavefront Technologies, Inc., Santa Barbara, CA, 1993.
2. C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE CG&A 9* (4), 30-42, 1989.
3. SGI Iris Explorer User's Guide, Version 2.0. Silicon Graphics, Inc., Mountain View, CA, 1992.
4. Delmarcelle, T. and L. Hesselink. Visualizing Second-Order Tensor Fields with Hyperstreamlines. *IEEE Computer Graphics and Applications 13* (4), July 1993, pp. 25-33.
5. Allen Zeyher. Simulated Materials Reveal Microfractures. *Computers in Physics 7* (4), 382-383, 1993.
6. L. A. Treinish and C. Goettsche. Correlative Visualization Techniques for Multidimensional Data. *IBM J. Research & Dev. 35* (1/2), 184-204, 1991.
7. Lloyd Treinish. Inside Multidimensional Data. *Byte 18* (4), 132-133, (April 1993).
8. M. W. Ribarsky, L. F. Hodges, Robert Minsk, and Marcel Bruchez. .Visual Representations of Complex, Discrete Multivariate Data. *The Visual Computer*, accepted for publication.
9. Ellson, R., and D. Cox, Visualization of Injection Molding, *Simulation 51* (5), 184-188, 1988.

10. Haim Lefkowitz. Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Perception. *Proceedings, Visualization '91*, San Diego, 164, 1991.

11. R.M. Pickett and G.G. Grinstein. Iconographic Displays for Visualizing Multidimensional Data. *Proceedings of the 1988 IEEE Conference on Systems, Man, and Cybernetics*, Beijing and Shenyang 1988.

12. S. Smith, G. Grinstein, and R.D. Bergeron (1991). Interactive Data Exploration with a Supercomputer. *Proceedings Visualization '91*, San Diego, 248, 1991.

13. Haber, R. and D. A. McNabb. "Visualization Idioms: A Conceptual Model for Scientific Visualization Systems," in *Scientific Visualization*, Shriver and Nielson (eds.), (1990) IEEE Computer Society Press, Los Alamitos, CA.

14. Foley, J. and C. McMath, Dynamic Process Visualization, *IEEE Computer Graphics and Applications 6* (3), March 1986, pp. 16-25.

15. J. D. Foley. Scientific Data Visualization Software: Trends and Directions. *Int. J. of Sup. Appl. 4*, 154, 1990.

16. J. D. Foley and W. Ribarsky. Next-Generation Data Visualization Tools. Proceedings of the ONR Darmstadt Conference on Data Visualization. Springer-Verlag, Berlin, 1993.

17. Senay, H. and E. Ignatius, Rules and Principles of Scientific Data Visualization, Technical Report GWU-IIST-90-13, The George Washington University, Washington, DC, May 1990. Also in *State of the Art in Scientific Visualization*, SIGGRAPH '90 tutorial notes, August 1990.

18. A.Buja, J.A. McDonald, J. Michalak, and W. Stuetzle. Interactive Data Visualization Using Focusing and Linking. *Proceedings of Visualization '91*, San Diego, 156-163, 1991

19. Hai-Ping Cheng and Uzi Landman. Controlled Deposition, Soft Landing, and Glass Formation in Nanocluster-Surface Collisions. *Science 260* (May), 1304-1307, 1993.

20. Steve Bryson and Creon Levitt. The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows. *Proceedings of Visualization '91*, San Diego, 17-24, 1991.

21. Plot3D User's Manual, Version 3.5. NASA Fluid Dynamics Division, Ames Research Center, Moffett Field, CA, 1988.

22. W. C. Leeuw and J. J. van Wijk. A Probe for Local Flow Field Visualization. *Proceedings of Visualization '93*, San Jose, 39-45, 1993.

23. W. Ribarsky, J. Bolter, A. Op den Bosch, and R. van Teylingen. Information Visualization and Analysis using Virtual Reality. *IEEE Computer Graphics and Applications*, to be published, January, 1994.

24. L. Hesselink and T. Delmarcelle. Visualization of Vector and Tensor Datasets. Proceedings of the ONR Darmstadt Conference on Data Visualization. Springer-Verlag, Berlin, 1993.

25. Beshers, C., and S. Feiner. Automated Design of Virtual Worlds for Visualizing Multivariate Relations. *Proceedings Visualization '92*, Boston, pp. 283-290, 1992.

26. J. Kolodner. Case-Based Reasoning. (Ablex, 1993).

Figure Captions

Fig.1 A general visualization environment.

Fig. 2 Glyphmaker visualization environment. The figure shows the flow of data between the parts of Glyphmaker as described in the text.

Fig.3 Data objects and contents.

Fig. 4 3D Glyph Editor interface. Selection menus are in the upper right corner; a compound glyph composed of a sphere and a vector glyph is displayed.

Fig.5 The Glyph Binder Interface. On the left are the glyph active elements and on the right are the variable names (in this example including position coordinates and properties for each atom type).

Fig.6 The Conditional Box surrounding the NaCl cluster and a portion of the Ne substrate from a molecular dynamics simulation. The control panel is at the lower left.

Fig.7 The full MD system at a later time step after the cluster has embedded itself in the Ne; here the color glyph elements are bound to local temperature. (Color scale same as shown in Fig. 9.)

Fig.8 The part selected by the Conditional Box at a time step between Fig. 6 and Fig. 7. Here local temperature is bound to sphere color, shear stress magnitude is bound to vector color, and on-axis stress components are bound to vector magnitude and direction. (Color scale same as shown in Fig. 9.)

Fig.9 The MD system (same time step as Fig. 7), sliced in half using the Conditional Box to expose the impact region, with the NaCl cluster bindings turned off, and with color bound to local temperature. 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 ^{o}K.

Fig.10 The crystalline substrate with both NaCl and Ne bindings turned off (same time step as Fig. 7) with visualization of the on-axis (stretching/compressive) components of the atomic stress tensor added. The viewpoint is directly over the NaCl position and perpendicular to the crystalline substrate surface.

Fig.11 Two zone grid for a computational fluid dynamics simulation of a tilt-rotor blade in hover mode. The cuboids are bound to zone 1 elements and the spheres are bound to zone 2 elements.

Fig.12 A close-up of zone 1 from Fig.11. The red cursor points at the tip region of the tilt-rotor blade. For each element, momentum is bound to the vectors, pressure to

the vector color, and density to sphere color. The color spectrum used is the same as shown in Fig. 10 but scaled to the appropriate property values.

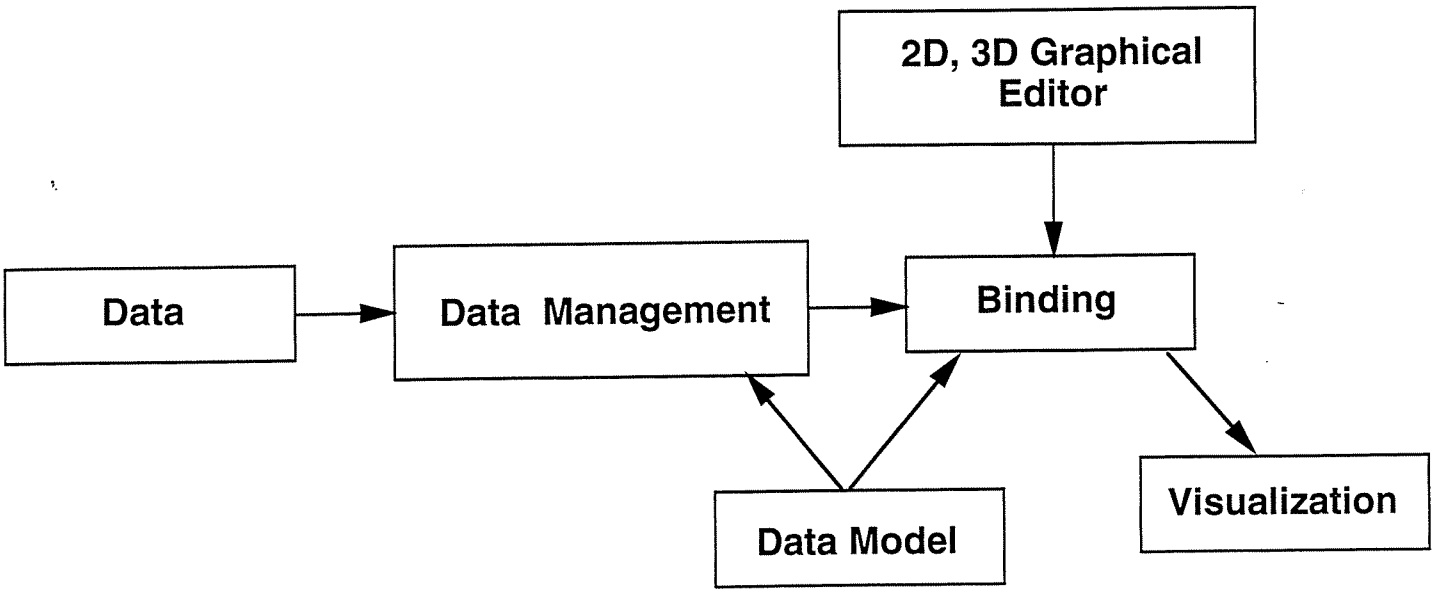

Fig. 1 A general data visualization environment

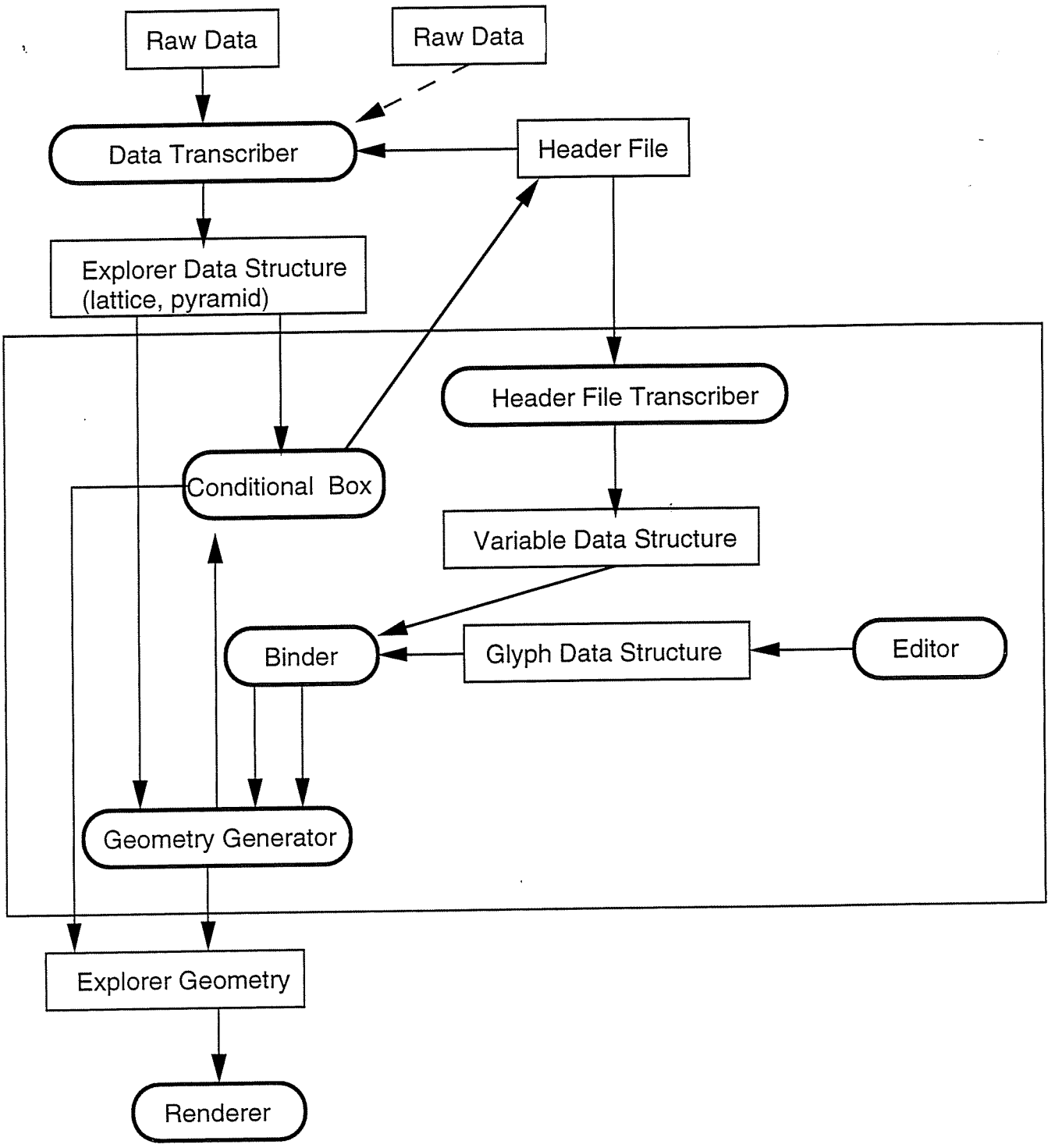

Fig. 2 Glyphmaker visualization environment. The figure shows the flow of data between the parts of Glyphmaker as described in the text.

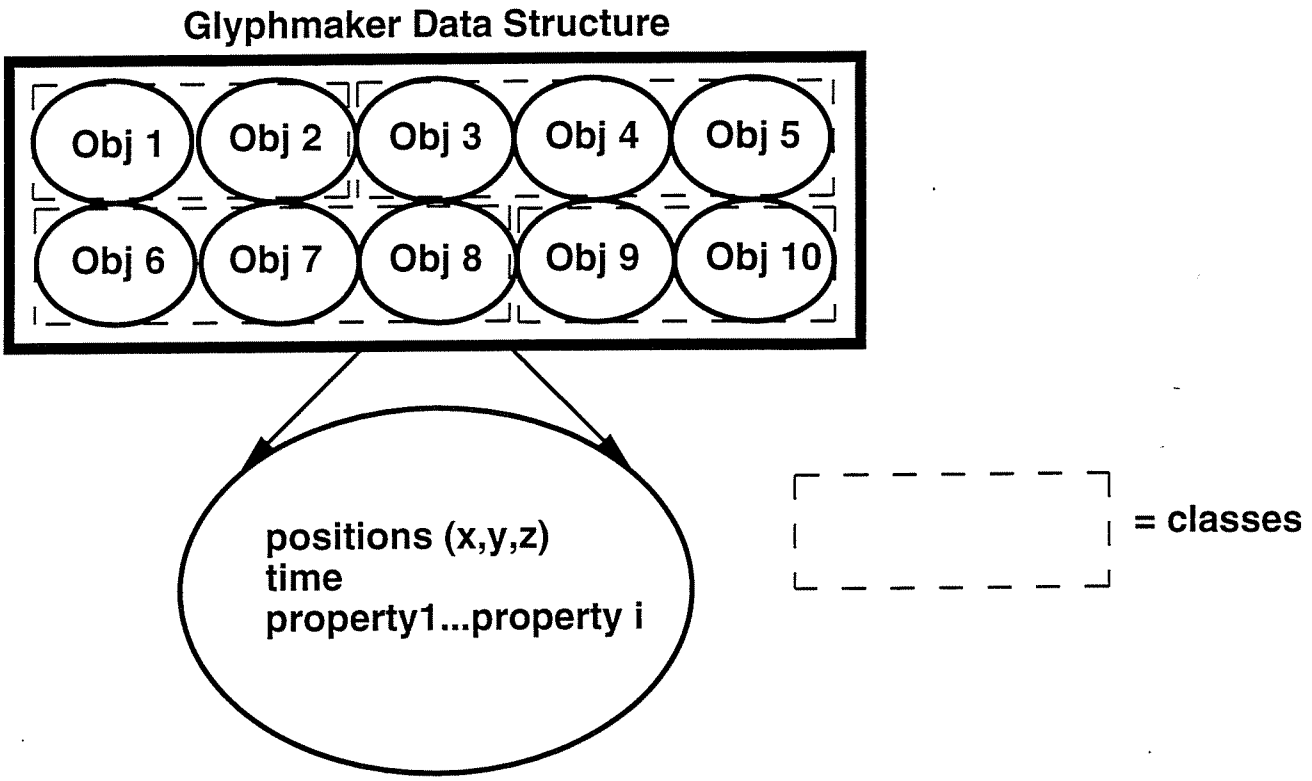

Figure. 3 Glyphmaker data objects and contents

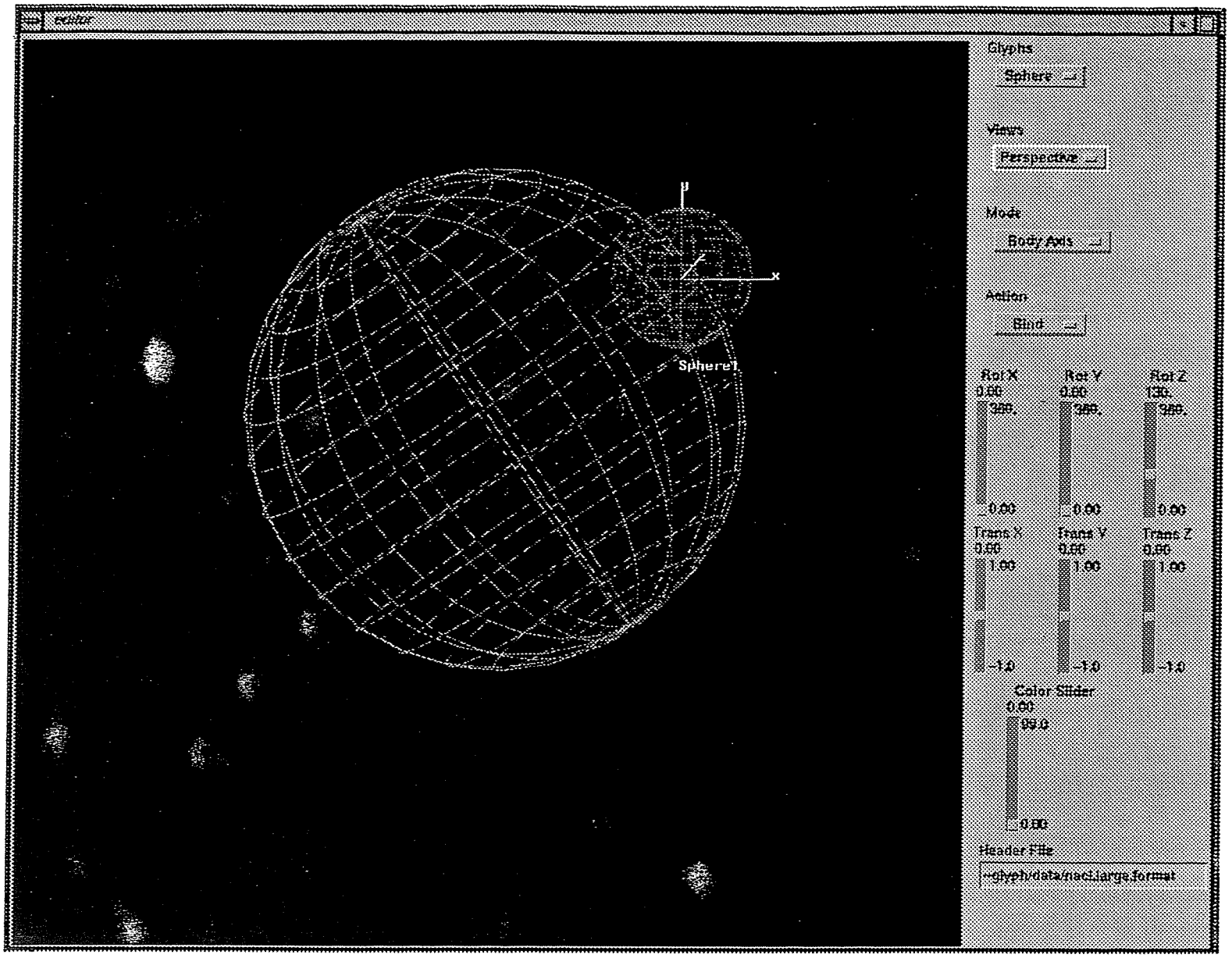
Glyphs
Sphere
Views
Perspective
Mode
Body Axis
Action
Bind
Rot X
0.00
360.
0.00
Rot Y
0.00
360.
0.00
Rot Z
130.
360.
0.00
Trans X
0.00
1.00
-1.0
Trans Y
0.00
1.00
-1.0
Trans Z
0.00
1.00
-1.0
Color Slider
0.00
0.00
Header File
~glyph/data/naci.large.format
y
x
Sphere1

Fig. 4

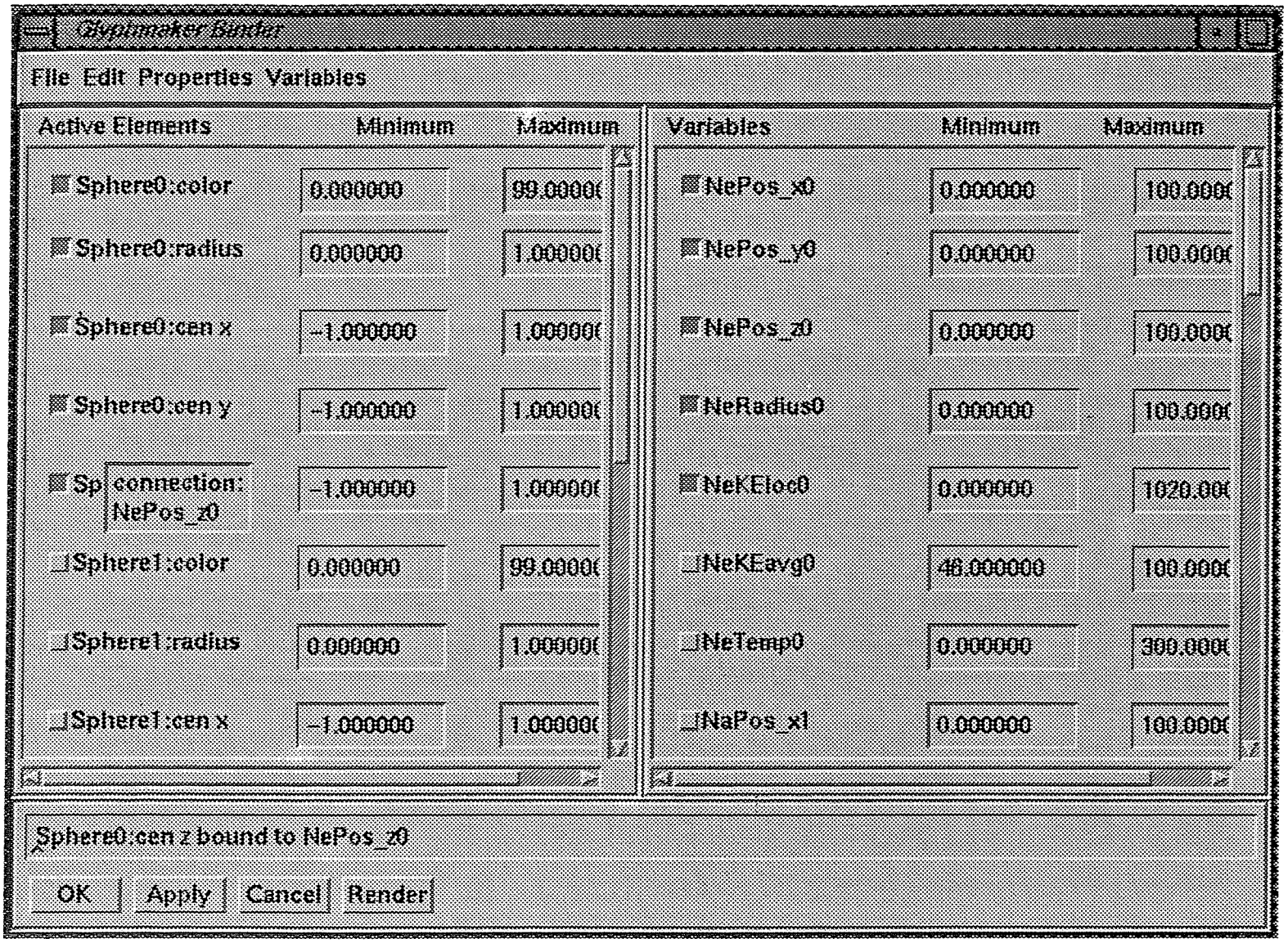
Toymaker Builder
File Edit Properties Variables
Active Elements
Minimum
Maximum
Sphere0:color
0.000000
99.0000(
Sphere0:radius
0.000000
1.00000(
Sphere0:cen x
-1.000000
1.00000(
Sphere0:cen y
-1.000000
1.00000(
Sp
connection: NePos_z0
-1.000000
1.00000(
Sphere1:color
0.000000
99.0000(
Sphere1:radius
0.000000
1.00000(
Sphere1:cen x
-1.000000
1.00000(
Variables
Minimum
Maximum
NePos_x0
0.000000
100.000(
NePos_y0
0.000000
100.000(
NePos_z0
0.000000
100.000(
NeRadius0
0.000000
100.000(
NeKEloc0
0.000000
1020.00(
NeKEavg0
46.000000
100.000(
NeTemp0
0.000000
300.000(
NaPos_x1
0.000000
100.000(
Sphere0:cen z bound to NePos_z0
OK
Apply
Cancel
Render

Fig. 5

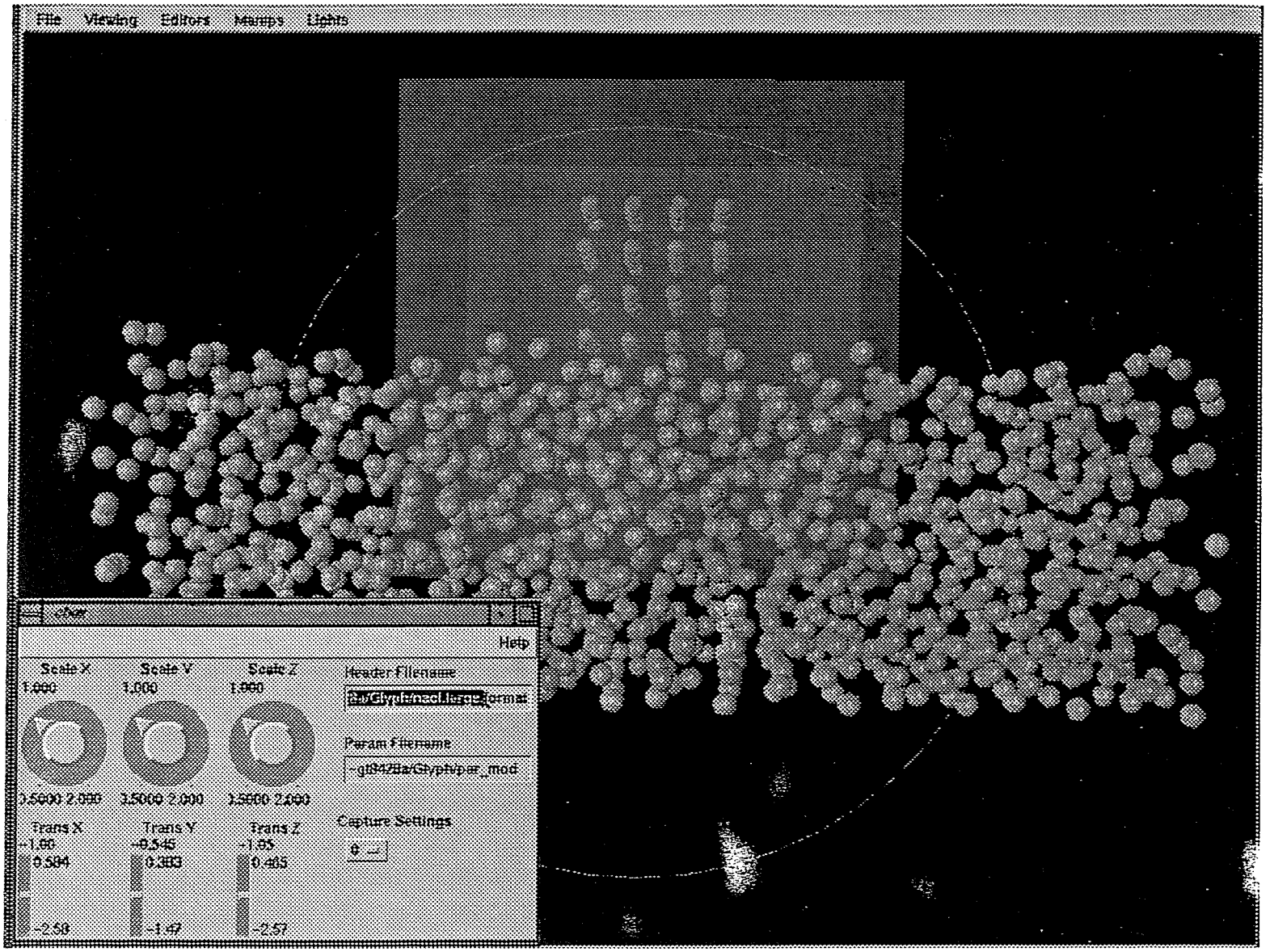
File
Viewing
Editors
Manips
Lights
Help
Scale X
1.000
Scale Y
1.000
Scale Z
1.000
Header Filename
Param Filename
Capture Settings
Trans X
Trans Y
Trans Z

Fig. 6

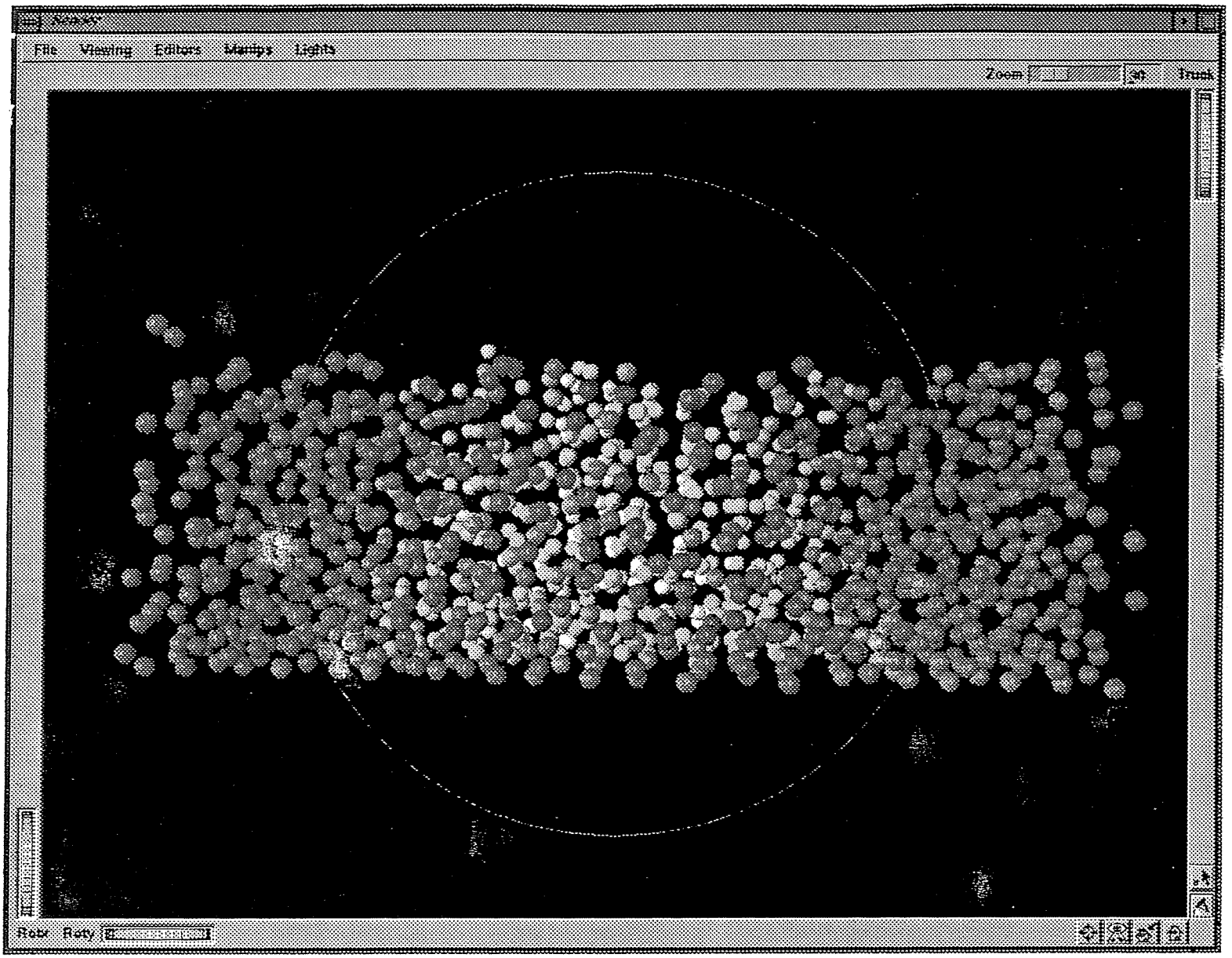

Fig. 7

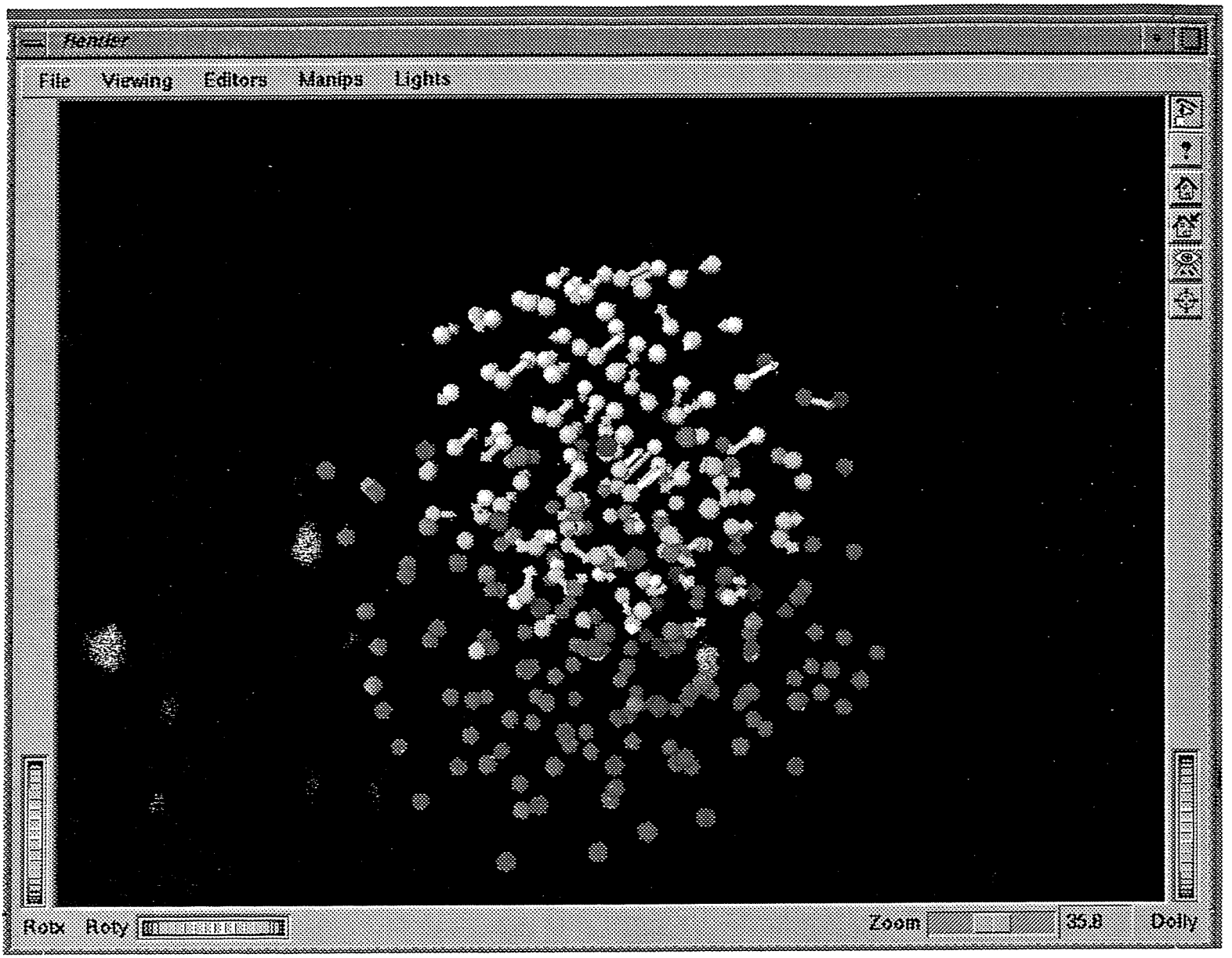
File
Viewing
Editors
Manips
Lights
Rotx
Roty
Zoom
35.8
Dolly

Fig. 8

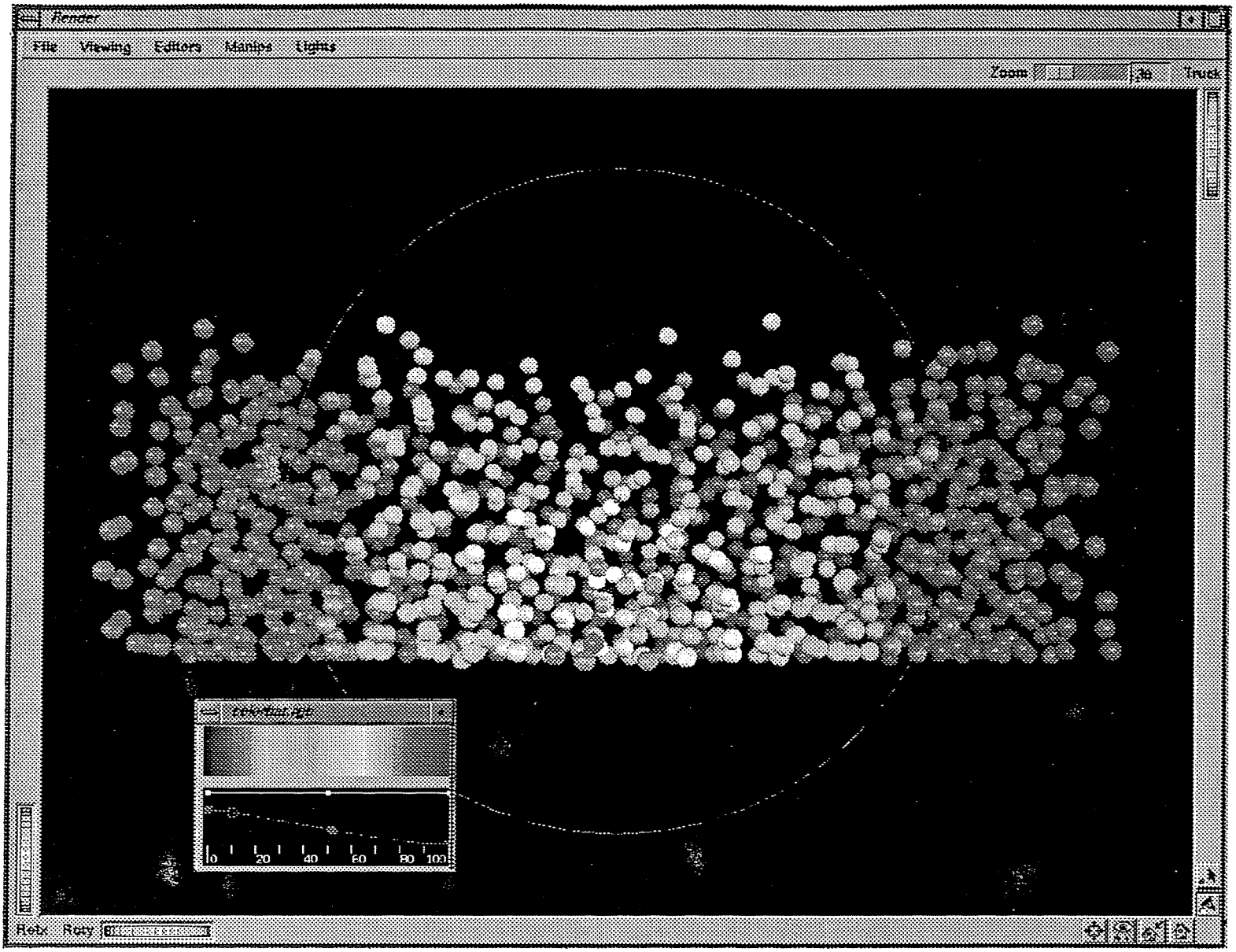
Render
File
Viewing
Editors
Manips
Lights
Zoom
Truck
0
20
40
60
80
100
Rotx
Roty

Fig. 9

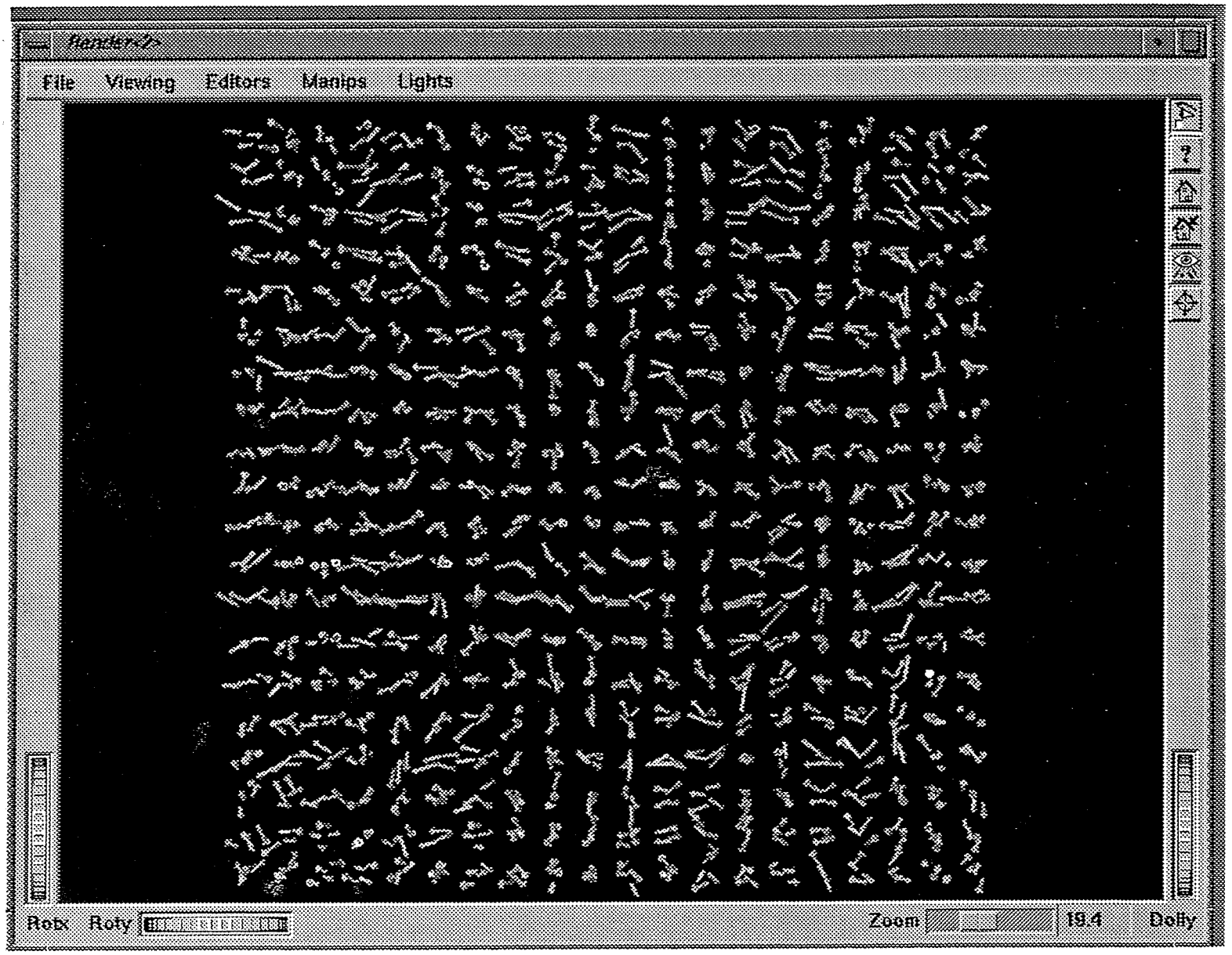
Renderer2>
File
Viewing
Editors
Manips
Lights
Rotx
Roty
Zoom
19.4
Dolly

Fig. 10

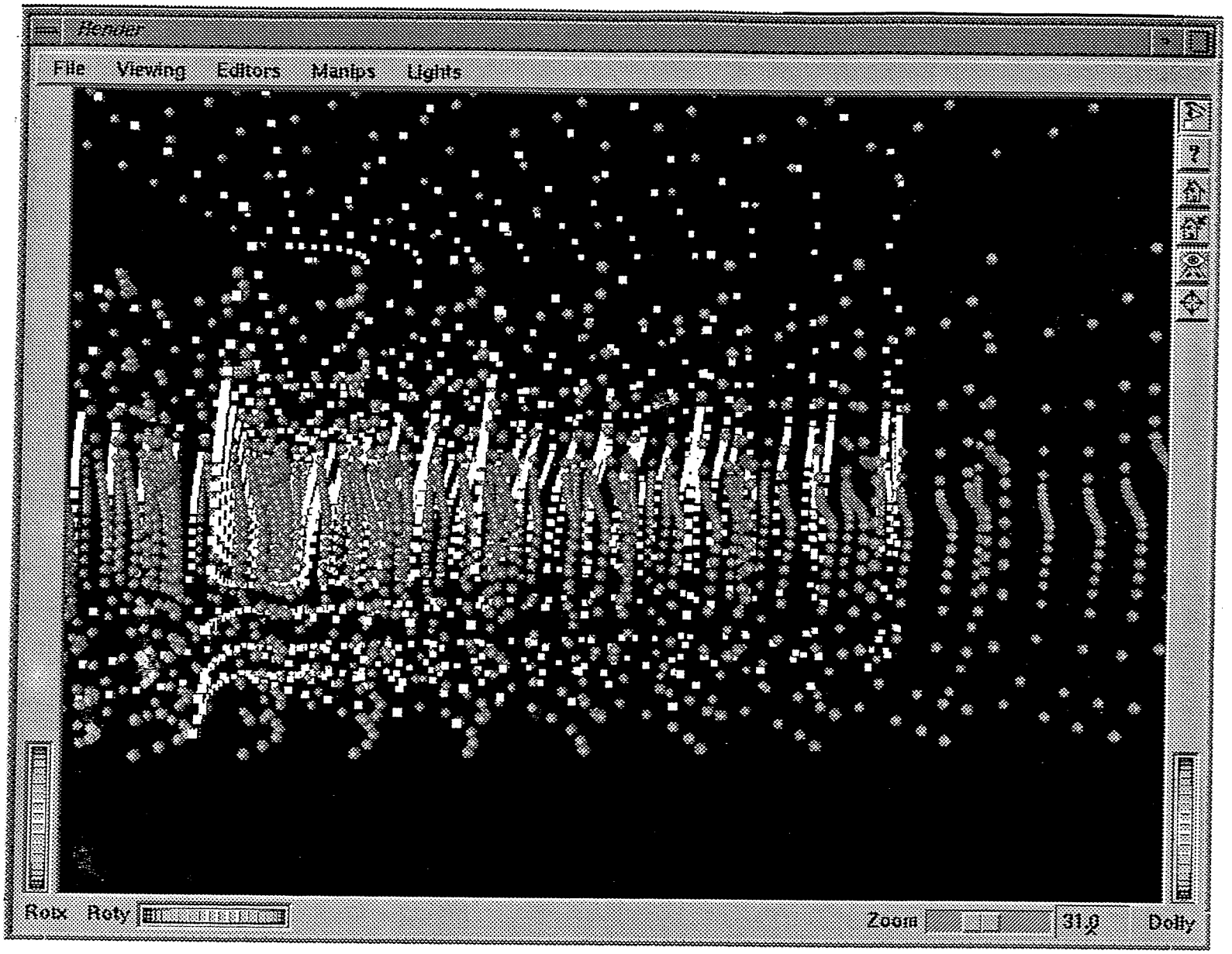
Render
File
Viewing
Editors
Manips
Lights
Rotx
Roty
Zoom
31.9
Dolly

Fig. 11

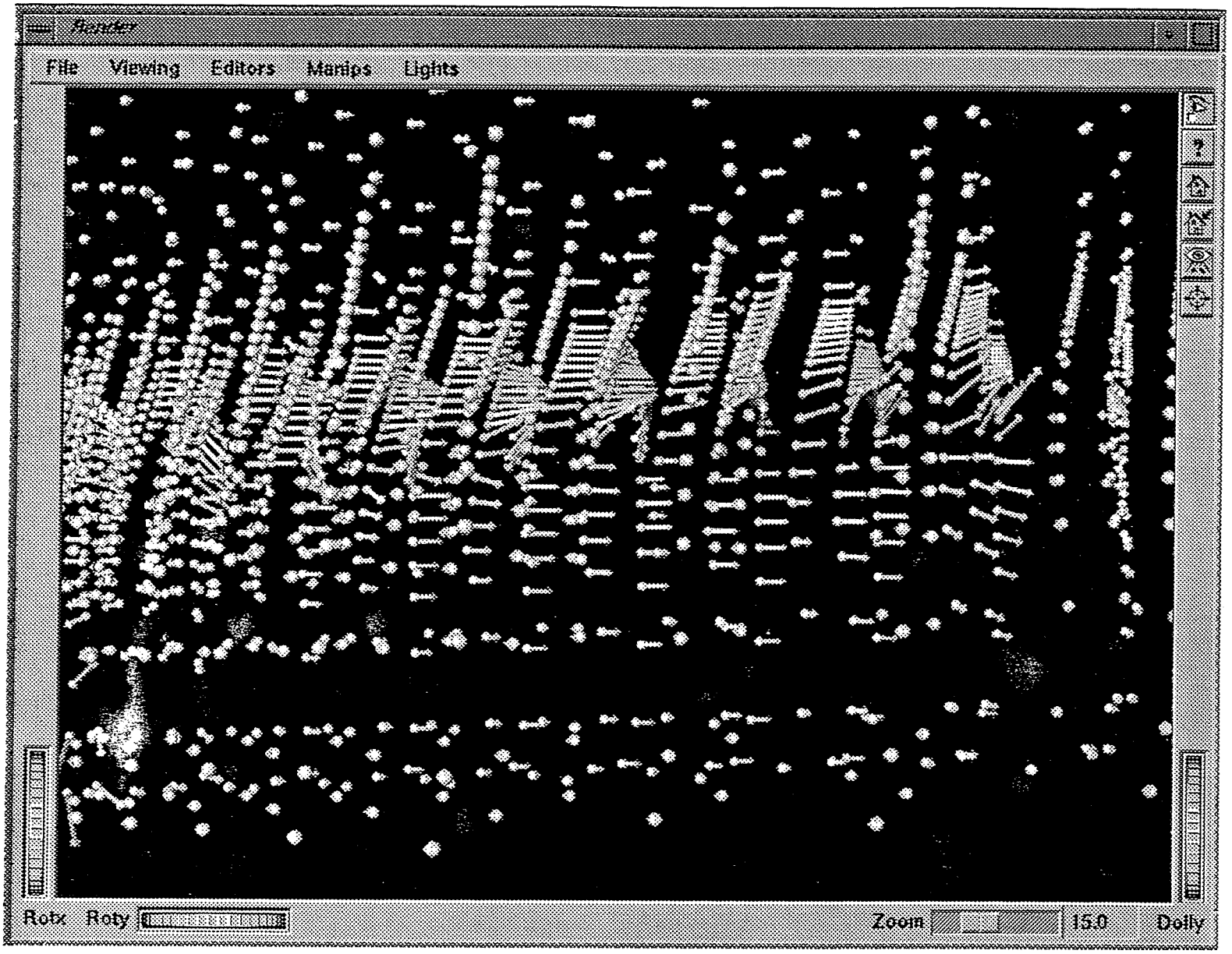
Renderer
File
Viewing
Editors
Manips
Lights
Rotx
Roty
Zoom
15.0
Dolly

Fig. 12