

# Visage: A User Interface Environment for Exploring Information

Steven F. Roth, Peter Lucas, Jeffrey A. Senn, Cristina C. Gomberg,  
Michael B. Burks, Philip J. Stroffolino, John A. Kolojechick & Carolyn Dunmire<sup>1</sup>

MAYA Design Group, Inc.  
2100 Wharton Street  
Pittsburgh, PA 15203  
412-488-2900  
lucas@maya.com  
<http://www.maya.com>

Carnegie Mellon University  
School of Computer Science  
Pittsburgh, PA 15213  
412-268-7690  
steven.roth@cs.cmu.edu  
<http://www.cs.cmu.edu/~sage>

Army Research Lab  
Aberdeen Proving Ground  
MD 21005  
410-278-5937  
cdunmire@arl.mil

<sup>1</sup>Roth & Kolojechick are affiliated with Carnegie Mellon, Dunmire with ARL, and the others with MAYA Design Group.

## Abstract

*Visage is a prototype user interface environment for exploring and analyzing information. It represents an approach to coordinating multiple visualizations, analysis and presentation tools in data-intensive domains. Visage is based on an Information-centric approach to user interface design which strives to eliminate impediments to direct user access to information objects across applications and visualizations. Visage consists of a set of data manipulation operations, an intelligent system for generating a wide variety of data visualizations (SAGE) and a briefing tool that supports the conversion of visual displays used during exploration into interactive presentation slides. This paper presents the user interface components and styles of interaction central to Visage's information-centric approach.*

**Keywords:** Visualization, exploratory data analysis, graphics, user interface environment, human-computer interaction

## 1. Introduction

There has been a great deal of research recently on techniques for supporting the exploration and manipulation of information. This research has produced new ways to visualize information, new techniques for interacting with visualizations to manipulate information, and new tools for supporting the creation of visualizations. Much of this work has been quite general in producing visualization techniques applicable to diverse data (e.g. quantitative, symbolic/relational, geographic, temporal) and interaction techniques that can be combined and applied across many different styles of visualization (e.g. 1,4,6,7,8,9,10,11,12).

This work has begun to produce several general purpose analysis tools, each with its own strengths. The Table Lens [11], for example, is a dynamic spreadsheet environment for exploring large, multidimensional data sets with techniques

for focusing attention on subsets while viewing the rest as context. Its strengths include techniques for rapidly creating and viewing the relations among new data attributes. Another evolving analysis package is IVEE [2], which provides the ability to rapidly create multiple dynamic query sliders to filter data. A third example is the SAGE system [12], whose central feature is rapid design of visualizations that integrate multiple attributes.

Taken together, these systems illustrate a fundamental user interface design question: how can we use the complementary features of different visualization and analysis tools in a coordinated way. Even for just these three systems, how can we create new attributes in one, filter the same data with another, and visualize the resulting subsets with a third? More generally, what user interface approach would enable people to easily move and combine interesting subsets of information across the isolating boundaries imposed by different applications?

Of course, the coordination problem is not unique to these tools. Most people who work with large amounts of information also use custom applications. For example, in domains like transportation scheduling and tracking (which we have been using as a test case), analysts use one system to generate and display airplane schedules, another for tracking the location of cargo in transit, and a third for managing warehouse inventory and requisition handling. The interfaces to these applications each have useful visualizations but no mechanism to explore relationships among the different data they portray. For example, there is no way to explore the relations among the locations where supplies are stored, the people who order them, and when they are scheduled to be shipped by air.

These problems suggest the need for a user interface environment for people who work in information-intensive domains - an electronic workspace for people who explore and analyze large amounts of data daily. Such a workspace must provide several key capabilities.

First, it requires user interface techniques that enable information to be selected and combined from multiple application interfaces, visualizations, and analysis tools.

Second, it must enable rapid generation of visualizations to integrate information from these diverse sources. The value of integrative visualizations is obvious. However, because the combinations of information that people will create are often unpredictable, it is not possible for software developers to create every visualization in advance. Therefore, an effective workspace must provide tools by which users can create new visualizations as needed without great effort or skill.

Third, consistent user interface techniques are needed with which people can filter, control level of detail, navigate, and create new information wherever it is displayed.

Fourth, an effective environment should make it easy for people to share and communicate their results in collaborative settings, where they must iterate between analysis and presentation activities frequently.

In order to address these needs, we are developing an approach within a software environment called Visage. Our goal is to incorporate basic information exploration components within a new user interface paradigm. This paper describes several key elements of Visage.

**1. A consistent information-centric user interface paradigm.** As the name implies, this paradigm strives to provide users with greater direct contact with objects that represent information they need to view and manipulate to perform their work. In this paradigm, information is represented as first-class objects that can reside and be manipulated in visualizations, application user interfaces, on desktops, in briefing materials, or anywhere else people elect to place it. It is ultimately concerned with usability (i.e. it is *user-centered*), in that it seeks to reduce the complexity and restrictions created when people cannot access information directly and instead must face the mechanics of running and coordinating applications and working with file system metaphors.

**2. Dynamic visualization generation.** In order to provide integrative views of information, we are incorporating work on SAGE, a knowledge-based automatic graphic design tool [12]. This approach provides rapid generation of visualizations customized to users' immediate data exploration tasks.

**3. Interactive information manipulation.** These include tools for:

- finding and interactively partitioning, filtering, and selectively combining subsets of data on which to focus,
- controlling the level of detail with which this information is viewed using *drill-down* and *roll-up* techniques (*drill-down* commonly refers to the process of segmenting or breaking down aggregated data along different dimensions to create a larger number of smaller aggregates; *roll-up* commonly refers to the process of merging detailed data into aggregates that summarize their attributes), and
- assembling, laying out, and *interactively* presenting information to others.

## 2. An Example

In order to convey Visage's basic styles of interaction, it is useful to consider a detailed example. The example is based on one of the applications of this approach that we are pursuing to facilitate next generation logistics tracking and planning systems. These government systems are being developed to access and analyze information about the location, quantities, status, transportation, distribution, consumption and other properties of equipment and supplies and the people who need them worldwide.

Figures 1 and 2 contain an outliner style of table that is one of many displays we created in the Visage environment. It is one way to provide a hierarchical perspective on tabular data and is useful for this example because it illustrates drill-down and roll-up capabilities in a familiar way. The same techniques are applicable to other approaches to displaying hierarchical data (e.g. [8, 9]).

	total_supply	echelon
ARMY	20156.3	FA
22MD.M	3165.86	DIV
53MD.M	2681.8	DIV
SUPPLY REQ	186.81	TA
parent_unit	100.89	FA
subord_unit		
supported by		

Figure 1: Drilling down organizationally

Starting from any point in an object-oriented database for a logistics exercise, users are offered a menu of alternative dimensions along which they may drill down. In Figure 1, a user has already drilled down from an object representing the Army Corp to its five subordinate units and has selected one division (53rd Division) to drill down

further organizationally. This occurs by selecting the *subordinate unit* relation from the pop-up menu that is dynamically attached to the 53rd. The result is the more detailed organizational breakdown (the highlighted text) in Figure 2 (also see blue highlighted text in color plate).

This drill-down process could also occur across different relations or links from any of these objects. For example, it is possible to drill-down from an object representing an Army unit to the equipment it possesses (e.g. trucks, generators, stoves). One could then drill-down further from one type of equipment to the parts or supplies it requires (e.g. from trucks to truck parts) and then drill-down from one supply-type to all the warehouses that have it in stock. This is a process of turning a network or web of database objects into a convenient hierarchical breakdown for analysis purposes.

On the right side of the outliner in Figure 1, users can select any attribute of the objects in the hierarchy that they want to have displayed, like the weight of supplies a unit requires, its echelon, the number of people in the unit, etc. These attributes can be taken directly from the database or dynamically created as derived attributes using a scripting language. In either case, as the hierarchy is expanded, the values for these attributes are added with it. The dynamic drill-down and expression of attribute values is a fundamental operation in Visage that can occur in every display. In the example, a user has drilled-down just organizationally to a group of Army units of interest.

An important operation in our implementation of the information-centric interface approach is the ability to drag objects representing information among visualizations and application interfaces throughout the Visage environment. For example, in order to display graphically some of the attributes of the subordinate units of the 53rd Division, one simply drags a copy of the unit names from the outliner to an empty bar chart display in Figure 2 (the moved units are indicated by the translucent text).

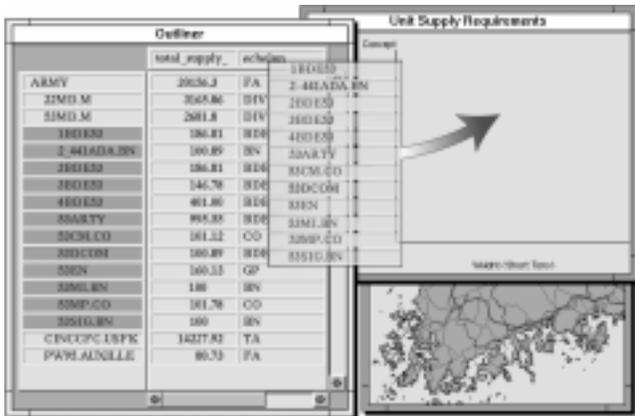


Figure 2: Dragging objects across displays

In this case, a bar chart shows the weight of supplies that these units require as bars in Figure 3 (some are already selected for subsequent copying and dragging to a map and appear darker in Figure 3 and red in the color plate). Each display shows some attributes by default or users can select other attributes from a menu attached to the chart. The menu of attributes is constructed dynamically from the objects that are dropped into the display (i.e. the attributes of Army units in the example). Visualizing unit supply weights in a bar chart makes it easy to select those needing the most supplies - those with the longest bars. These are painted by the user and appear dark gray in Figure 3.

It's now possible to check the locations of just these units on a map, perhaps to determine the locations where supply warehouses should be established. Units are transferred to a mapping application with a similar drag and drop operation suggested by the translucent bars in Figure 3.

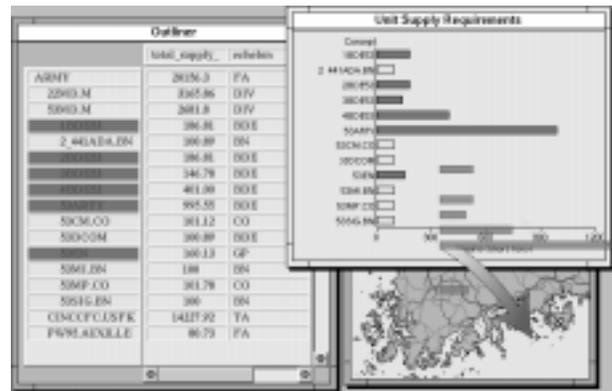


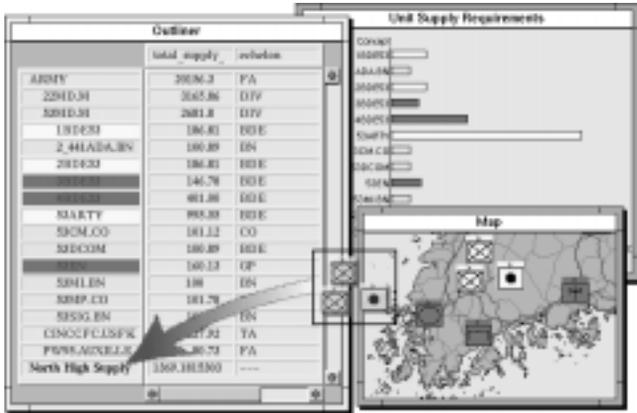
Figure 3: Dragging units with the largest quantities (dark bars) from a chart to a map.

The map application is a product called MATT which was developed independently of Visage by Bolt, Beranek and Newman and coordinated using its program interface. This is an example of one of the primary goals of this work, to explore how to cement together separately developed analysis tools into what the user will experience as an integrated work environment. By default, a map display shows the longitude and latitude attributes of objects and uses military symbols associated with unique identifiers of units and other objects.

The map display can be used to further focus attention, for example, by painting yellow (light gray in Figure 4) the subset of units that occur close together in the center of the map (perhaps to identify a region where large quantities of supplies will be needed). Notice that painting [4,6,8,10] an object in one display causes it to be similarly colored in all other displays. Together, the three displays in Figure 4 show who the selected units are, how many supplies they need, and where they are located. In the color plate, the

blue, red, and yellow units are all subordinates of the 53rd Division. The red and yellow ones need the most supplies. Yellow ones are Northern units of special interest to a user.

Finally, this small subset of units can be rolled-up (i.e. aggregated) into a single object and named by a user, in this case, "North High Supply Units." It appears in the bottom of the outliner, and its attributes are the appropriate totals for the units it aggregates. The new aggregate can be treated as a single object for new drill-down operations. For example, it is possible to drill-down along a new dimension to the supply types needed by the aggregated units.



**Figure 4: Dragging copies of objects back to the outliner and composing them into new data aggregates**

Notice that this approach enables users to compose a complex query through a series of drill-down, drag, paint and roll-up operations. The last aggregate represents "units of the 53rd Division that need the 'most' supplies and are located in a small region to the north". No abstract query needed to be constructed.

In summary, this example illustrates the most important aspect of Visage's information-centric approach: operations are directly applied to graphical objects representing information and not through the mechanics of running applications. For example, bar chart and mapping programs are invoked by moving graphical objects to displays rather than by running programs and performing export/import procedures (the typical data sharing mechanisms used in spreadsheets and other tools). Also, painting operations are applied directly to objects that are coordinated in multiple displays via a common underlying database. Similarly, drill-down and roll-up operations are performed directly in any display because of a shared object representation. Later, we will describe how these display themselves can be similarly copied and dragged into presentation slides for briefings.

Finally, what is not illustrated here is the role of dynamic graphic design for creating composite visualizations dynamically that integrate multiple types of information. The outliner and bar chart displays in this example were created using a scripting language which will be described in detail later.

### 3. Background: Towards an Information Centric User Interface Architecture

The VISAGE user interface paradigm takes an aggressively *information-centric* approach to the presentation of information to the user. The information-centric approach may be thought of as the next logical step along the path from *application-centric* architectures to the modern *document-centric* approach. The distinctions among the three approaches hinges on differences in the "basic currency" through which the users interact with the system.

In application-centric architectures, the basic currency is the file. The file system is completely exposed to the user and a somewhat detailed understanding of its workings is a prerequisite to the productive use of the system. Moreover, although files in the file system are the basic unit of information, the files themselves are of little use to the user. To access the information in their files, users must rely on "applications" to fetch and display the information from the files on their behalf. In this regard, applications are like remote manipulator arms in nuclear power plants--users are not allowed to "touch" their data, except indirectly via various special-purpose tools. Each application has its own user interface which defines what kinds of files people can manipulate and what they can do with them.

With the introduction of graphical user interfaces and the desktop metaphor, files became concrete visual objects, directly manipulable by the user, storable on the desktop or in folders, and--to a limited extent--arrangeable by users and software in semantically meaningful ways. But the *contents* of those files were still out of direct reach of the user.

The advent of document-centric interface paradigms has introduced many positive changes into this story. In this world, the basic currency is no longer the file but rather the *document*--an entity with some potential meaning in the user's world-outside-the-computer. The role of the application is subordinated (and perhaps ultimately eliminated) in favor of component architectures whose interactions with the user are focused on direct manipulations of documents. Documents may be kept on the desktop in addition to files and may be directly activated and manipulated via drag-and-drop operations. Documents may serve as containers for other documents, enabling natural modes of grouping and attaching information together in meaningful units. Some extremely document-

centric interfaces (e.g. Workspace, [3], Web Forager [5]) permit the spatial arrangement of large numbers of documents, enabling effective visualizations of the relationships among them. The application of dynamic query techniques in a document-centric world enables visual search paradigms. In document-centric interfaces, users can almost literally "get their hands on" their documents.

The information-centric approach in Visage simply represents a natural continuation of these trends. Visage abandons the primacy of the document wrapper as the central focus of user interaction in favor of the *data element* as the basic currency of the interface. Rather than limiting the user to files and documents as targets of direct manipulation, Visage permits direct drag-and-drop manipulation of data at *any* level of granularity. A numeric entry in a table, selected bars from a bar chart, and a complex presentation graphic are all first-class candidates for user manipulations, and all follow the same "physics" of the interface. The object oriented nature of this approach is clearly is not unique to Visage and indeed was introduced and explored in Smalltalk and other systems (e.g. 10). Our work addressed the user interface issues raised in using this approach throughout an information visualization and exploration environment.

#### 4. Visage Environment Main Components

The Visage interface environment strives to minimize the number of fundamentally different kinds of objects that must be understood by the user. As a first approximation, Visage may be thought of as having only two basic object types: *elements* and *frames*.

The term *elements* (more properly called "Visual Elements") refers to any atomically-manipulable graphical object in a Visage display. Examples of elements are bars in a bar chart, the text label of an axis, a point in a scatter chart, or a numeric value in a spreadsheet-like cell. Each visual element corresponds to an object in an underlying database. Note that this relationship is one-to-many: each element is associated with exactly one database object, but the same object may be represented by multiple elements. Some elements are atomic, but others are compound, i.e., made up of multiple elements bound together into a single entity. An example of the latter might be a cluster composed of a dot representing a city on a map, a text label naming it, and a gauge telling its population.

In a literal sense, all Visage displays are made up exclusively of collections of elements arranged to form the display. For example, the bar chart in Figure 3 is not a discrete "picture", but rather an arrangement of elements which can be broken apart by the user and separately manipulated. As the illustration shows, this makes it easy

for the user to select some bars from the display (either removing them or--as in the case shown--duplicating them) and drag them to some other display. This ability to directly drag small units of data forms the basis of the information-centric approach to interface design described above.

Hints of this approach may be found in a few existing interfaces. For example, recent versions of Microsoft Word support the ability to drag selected text from one place in the document to another--thus bypassing the often criticized invisible clipboard as a mechanism for moving data around within an application (however text cannot be dragged onto the desktop or into other applications). The Macintosh system provides transparent drag-and-drop, which Netscape uses to enable images to be dragged from web page displays onto a desktop (though they are immediately hidden within files). Likewise, several visualization tools support representing data objects graphically and provide filtering, painting of linked displays, and related operations [4,6,8,10]. In Visage, these capabilities are promoted from special-purpose features to capabilities that can be used everywhere in the environment. It becomes part of the "basic physics" of the interface, empowering the user to directly perform unique actions that might otherwise require knowledge of numerous specialized interface "features."

*Frames*, the second basic object type, serve as pasteboards for elements. Strictly speaking, frames are themselves elements, but are sufficiently distinct in the user's model of the interface as to warrant separate treatment. Like windows in traditional GUI designs, frames provide a grouping function for related elements as well as a frame of reference for their arrangement. Unlike windows, however, frames are lightweight objects, easily created and destroyed, frequently manipulated by the user, and are themselves subject to the entire repertoire of direct-manipulation actions available for other elements (duplication, drag & drop, dynamic scaling, etc).

Another major function of frames is to serve as anchor points for *scripts*. The Visage user interface is highly scripted. Beyond the processing of basic user events, such as mouse-dragging and clicking, much of the high-level behavior of the system is controlled by user-accessible script rather than hard-coded methods. Although scripts may be attached to any element, most of the scripts of a typical interface are associated with frames. In the illustrated examples, it is the script of the "Bar Chart" frame that causes data dropped on that frame to be displayed as horizontal bars of certain lengths and locations. Similarly, scripts of the map frame cause the same data to be displayed in iconic form arranged by latitude and longitude. In this way, scripted frames may be composed to form highly

customized applications environments, tailored to a particular user's needs.

The scripting environment provides another essential feature in support of the Visage's data exploration operations. Although the underlying database being explored may have many data values directly given, many other such values typically need to be derived in a very situation-specific manner. For example, in a transportation scheduling application, the database may contain attributes of a commodity such as gross weight and package weight. The user, however, may require a display of net shipping weight, which is not directly given. Visage allows the definition of scripts that compute these "derived attributes." Once defined, these scripts make available to the user data indistinguishable from that directly given in a database. Indeed, the total supply weight attribute in Figure 1 is a detailed script that reduces much database access and calculation to a single attribute. It is attached to a Visage data object representing the class, military unit, and is invoked when an instance of this class receives a message to return the value of this attribute. The script traverses relations between a unit, its subordinates, the supply quantities they possess and accesses attributes of supply classes to retrieve their individual weights to be accumulated.

Finally, scripting is the basis for delivering to the user the drill-down and roll-up data navigation features described above. For instance, our logistics application supports a "regrouping" operation which, in a single step, decomposes a supplied unit into a long list of its required inventory items and then regroups them by the supply points that provides those items. Such operations form the basis for very powerful incremental data navigation and summarization which would require complex queries in conventional database systems. The Visage scripting language is similar to HyperTalk and contains language features tuned to data navigation and aggregation functions (e.g. for stepping across links among objects, iterating over object sets and accumulating sums of values of quantitative attributes).

Collections of specialized frames are typically gathered together to form a coherent, highly-tailored work environment. Such environments may be augmented by scripted behaviors that add useful global features to the environment at large. An example of this is the fact that the painting of elements in our logistics environment is globally coordinated across all frames of the interface, thus greatly enhancing the user's ability to identify related information across displays. Similarly, dynamic query tools [1] are included in the environment, permitting the interactive control of the visual attributes of the elements of the display according to parametric aspects of the database.

People may add sliders to frames to select a subset of objects and then drag the subset to other frames to focus on different attributes. DQ sliders are visual elements with scripted behavior. Dropping one on a visualization in a frame causes it to collect all the attributes of data objects currently displayed in the frame. Users may then select an attribute from these for performing filtering operations.

Frames and their contents may be freely scaled, either by direct manipulation or by script. A given frame may be shrunk to thumbnail size for temporary storage, or expanded to full-screen during a presentation. This capability enables the efficient and flexible usage of the available screen real estate. Scale control is also a useful feature in the creation of briefing slides.

## 5. Incorporating Briefing Tools

We are exploring the use of the basic Visage tools to provide a simple briefing or "slide show" application which is completely integrated with the rest of the environment. As analyses are performed, text and graphics can be captured and saved in special frames called 'slides.' A slide is simply a frame with special scripts designed to make it easy to "paste up" other frames and elements for visual presentations. A user simply "drags and drops" the desired frames onto a slide frame, where they are scaled appropriately. Text annotation can be added to the slide using Visage text elements. Slides created in this way can be accumulated in a "slide sorter" frame, which has special scripts making it easy to sequence a presentation by simple drag operations (similar to popular commercial presentation packages, like PowerPoint). In the color plate, an outline, map, and chart have been dropped in a slide, which has been augmented with text and dropped into the slide sorter (the latter illustrated in Figure 5).



Figure 5: Slide sorter frame

The slider sorter has controls for initiating the sequential display of each slide at full-screen size. Thus, the briefing function has been seamlessly integrated with those of data exploration and analysis. Note that elements on the slide do not lose their separate identity--they are still fully-functional interface objects that can be dragged among displays, used for painting, dynamic query, drill-down and roll-up operations right in the briefing.

The ability to perform these information analysis operations during a presentation raises some interesting

opportunities and challenges. First, it provides the ability to answer questions by modifying the graphic. By copying slides in the briefing, it is possible to both preserve the original presentation as well as keep a record of changes created in response to questions. Another challenge which we have not addressed is the dynamic modification of briefing slides to reflect changes in the underlying database. This problem has been addressed partially by current approaches to application embedding and linking. We are currently implementing constraint mechanisms whereby changes can either be propagated or blocked from affecting briefing slides.

## 6. SAGE Automatic Graphics Generation

The impetus for the Visage project originated with a project on a related system called SAGE [12], whose goal was to provide ways to generate data visualizations automatically. SAGE is a knowledge-based presentation system that designs displays of combinations of diverse information (e.g. quantitative, relational, temporal, hierarchical, categorical, geographic). We have explored the use of SAGE to automatically assume the burden of visualizing information for other applications and for computer-supported graphic design. In the latter case, users interactively specify some or all of the graphical choices that SAGE must consider in generating displays of data. They do this by creating rough sketches of the graphics they want using a draw-like interface (called SageBrush) or by using a portfolio interface to browse previously created graphics to find some relevant to current needs (called SageBook).

SAGE's inputs include sets of data, a characterization of the properties of the data and the tasks that the graphics should be designed to support, and an optional set of design specifications, expressing a user's preferences for how the data set should be visualized. SAGE's output consists of one or more coordinated sets of 2D information graphics that use a variety of techniques to integrate multiple data attributes within a single display. SAGE integrates multiple attributes by representing them as different properties of the same set of graphical objects (e.g. color-coded bars), by assembling graphical objects into clusters that function as units to express data (e.g. gauges and text within nodes of networks), and by aligning multiple charts and tables with respect to a common axis.

The SAGE work has been successful for creating hundreds of integrative graphics. Our previous work developed an architecture within which basic data manipulation operations can be applied to coordinated combinations of SAGE graphics (e.g. painting, dynamic queries, and aggregation). However, to be useful in data-intensive environments, it was necessary to develop the

kinds of information-centric data manipulation approaches that are described here, especially the ability to: transfer objects across displays, turn displays into briefing materials, coordinate painting with application interfaces, and provide drill-down and roll-up operations for every object.

In order to achieve this, we separated SAGE's knowledge-based design engine from the process of rendering its graphics. Thus SAGE has become a "visualization design server", which provides specifications of its designs to be rendered in Visage. The Visage renderer interprets SAGE designs, renders them in Visage frames, and populates them with visual elements whose appearance is specified by the SAGE design. Once rendered in this fashion, visualizations are subject to all the Visage operations.

Integrating the two systems combines the strengths of Visage's information-centric manipulation operations with SAGE's graphics repertoire. We have also successfully added scripts to SAGE visualizations to perform special data manipulation operations and user interface functions for applications. This appears to be a promising combination of visualization creation technology and user interface development tools.

A parallel research and development effort is continuing to incorporate new design knowledge for creating 3D, animation and other visualization techniques in SAGE - especially for large data sets. As these are added to SAGE, we will explore the implications of applying current operations within the Visage environment to these new techniques.

## Acknowledgments

The authors would like to acknowledge the contributions of Noah B. Guyot, Hugo Cheng, Kenichiro Tanaka, Michelle Vincow, Joseph Mattis. This work was supported by contracts from ARPA and Army Research Laboratory.

## References

1. C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. *Proceedings CHI '94 Human Factors in Computing Systems*, ACM, April 1994, pp. 313-317.
2. C. Ahlberg and E. Wistrand. IVEE: An environment for automatic creation of dynamic queries applications. *Conference Companion, CHI '95 Human Factors in Computing Systems*, ACM, May 1995, pp. 15-16.
3. J. M. Ballay, Designing Workspaces: An Interdisciplinary Experience. *Proceedings CHI '94 Human Factors in Computing Systems*, ACM, April 1994, pp. 10-15.

4. R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, May 1987, vol. 29, no. 2, pp. 127-142.
5. S. K. Card, G. Robertson, and W. York. The WebBook and the Web Forager: An information workspace for the world-wide web. *Proceedings CHI '96 Human Factors in Computing Systems*, ACM, April 1996, pp. 111-117.
6. S. G. Eick and G. J. Wills. Navigating large networks with hierarchies. *Proceedings Visualization'93*, IEEE, October 1993.
7. K. Fishkin and M. C. Stone. Enhanced dynamic queries via movable filters. *Proceedings CHI '95 Human Factors in Computing Systems*, ACM, May 1995, pp. 415-420.
8. J. Goldstein, S. F. Roth, J. Kolojechick, and J. Mattis. A framework for knowledge-based, interactive data exploration. *Journal of Visual Language and Computing*, no. 5, 1994, pp. 339-363.
9. B. Johnson and B. Schneiderman. Treemaps: A space-filling approach to the visualization of hierarchical information structures. *Proceedings Visualization'91*, IEEE, October, 1991, pp. 284-291.
10. J. A. McDonald. Painting multiple views of complex objects. *ECOOP/OOPSLA '90 Proceedings*, Oct. 1990, pp. 245-257.
11. R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. *Proceedings CHI '94 Human Factors in Computing Systems*, ACM, April 1994, pp. 318-322.
12. S. F. Roth, J. Kolojechick, J. Mattis, and J. Goldstein. Interactive graphic design using automatic presentation knowledge. *Proceedings CHI'94 Human Factors in computing Systems*, ACM, April 1994, pp. 112-117.
13. L. Tessler. The Smalltalk Environment, *Byte Magazine*, 6, 8, Aug 1981, pp. 90-147.