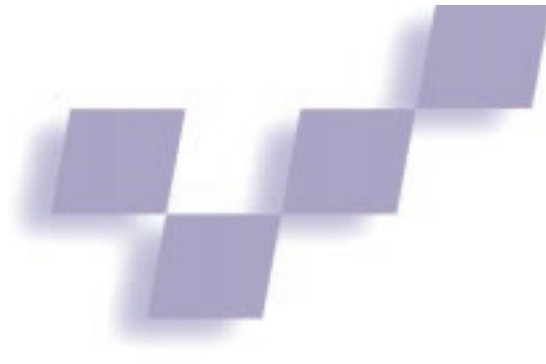


Information Appliances and Tools in Visage



John Kolojechick and Steven F. Roth
Carnegie Mellon University

Peter Lucas
Maya Design Group

Analysts benefit from a common set of operations and ways to integrate information across applications. Visage provides both basic tools and specialized information appliances to help them.

Most people who work with large amounts of information use special-purpose applications, each with its own customized visualizations, operations, and interfaces. In domains like transportation scheduling and tracking (which we use as a test case), analysts employ one system to generate and display shipping schedules, another to track the location of cargo in transit, and a third to manage warehouse inventory and requisition handling. Each

application provides operations and visualizations appropriate to the specific data analysis tasks it supports. They rarely provide a general mechanism for exploring relationships among the data used in a suite of applications. As a result, users of such “stove-pipe” systems frequently cannot explore important relationships in their data. For example, it is very difficult for transportation planners to use existing systems to explore the relationship between the locations where supplies are stored, the people who order them, and when they are scheduled to be shipped.

Lack of a common set of operations and ways to integrate information across multiple applications continues to be a problem for research on general-purpose tools and techniques for exploring and manipulating information. This research has focused on new ways to visualize information, new techniques for interacting with visualizations to manipulate information, and new tools to support the creation of visualizations. Some of this research has led to the development of general-purpose analysis tools, each with its own strengths. For example, the Table Lens¹ 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 data attributes. Another

evolving analysis package, IVEE,² rapidly creates multiple dynamic query sliders to filter data. A third example, the SAGE system,³ facilitates rapid design of visualizations that integrate multiple attributes.

Taken together, these systems prompt some fundamental design questions: How can we use the complementary features of different visualization and analysis tools in a coordinated way? What user interface approach would enable people to easily move and combine interesting subsets of information across the isolating boundaries imposed by different applications? These questions 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. Ideally, such a workspace would provide several key capabilities that enable users to

- select and combine information from multiple application interfaces, visualizations, and analysis tools;
- rapidly generate visualizations that integrate information from these diverse sources; and
- manipulate information, that is, filter it, control its level of detail, navigate through it, and add to it whenever it is displayed.

A workspace for supporting these information analysis tasks must also support varying levels of user expertise and user tasks. At one extreme, it must support specific analysis and reporting tasks performed frequently for the same data and purpose (such as checking weather conditions or inventory levels of a part). Analysts might perform these tasks to answer just a small set of frequently asked questions. However, skilled data analysts might also perform these tasks to answer the same frequently asked questions but in the context of other detailed, exploratory data analyses. Therefore, a workspace for information analysis must support both types of users for two types of analysis:

- Routine information reporting, in which users frequently seek answers to the same analysis questions

with slight variations in form. They can be supported with special-purpose interfaces that have a minimal, simple set of controls.

- Exploratory analysis, in which users must iteratively ask both routine and unanticipated questions about unique combinations of information. This requires interfaces with greater flexibility and power of expression, which typically require more expertise to operate.

We have been developing a workspace called Visage⁴ to address both types of users and analysis tasks. Our goal is for expert users to move easily between routine tasks and exploratory data analysis as situations demand. Furthermore, recognizing the inherent complexity of data analysis, we want to design a system accessible to occasional users that simultaneously provides power and flexibility for expert users. Ideally, such a system will support a gradual shift to increasing expertise so that occasional users of routine interfaces can perform more detailed analyses if needed.

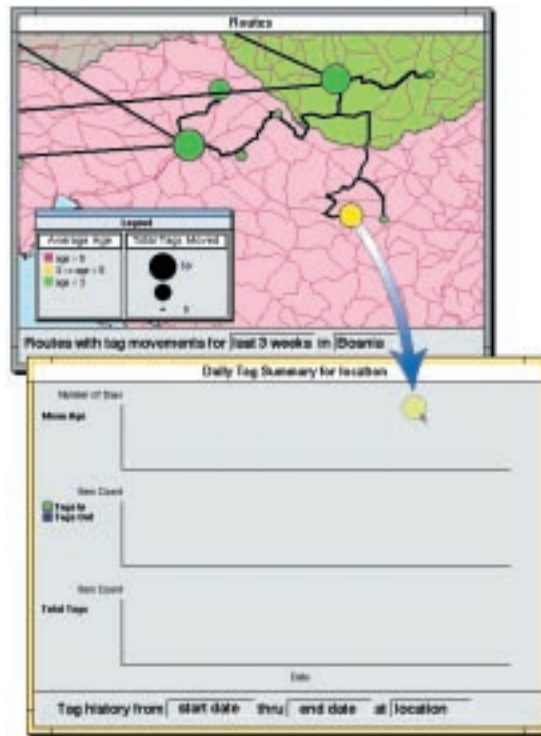
Visage addresses these needs by providing a consistent interface to a wide variety of data manipulation and visualization primitives. Plus, it lets users combine these operations to create *appliances*—special-purpose applications capturing multiple steps of a routine data analysis process in a single, simple interface. From a user's perspective, appliances are distinct applications. In Visage, however, the user can flexibly integrate them within the workspace because of a shared data space, drag-and-drop operations for transferring information among appliances and more general basic tools, and a common set of data manipulation operations available in every interface.

We used Visage to create a coordinated suite of basic tools and specialized information appliances. A scenario illustrates Visage's user interaction styles. We'll also discuss the information-centric design underlying Visage, its basic components, and the rationale for providing primitive operations, power tools, and appliance-like interfaces within the same environment.

Moving among appliances and tools

A scenario best illustrates the variety of interaction styles encountered in Visage applications. The following example is based on an application we are developing for the US Army to analyze a shipment tracking system database.

The Army fills orders for supplies requested by its units throughout the world. Supplies for each order are grouped in shipments. The shipment tracking database contains information about each shipment's contents, priority, origin, and destination. Attached to each ship-



1 A map for a selected region shows elements representing collections of shipments passing through various staging areas in a transportation network. Air and ground routes connecting staging areas are indicated with thick black lines. A collection of shipments from one staging area is being dragged to another frame below it for more detailed analysis.

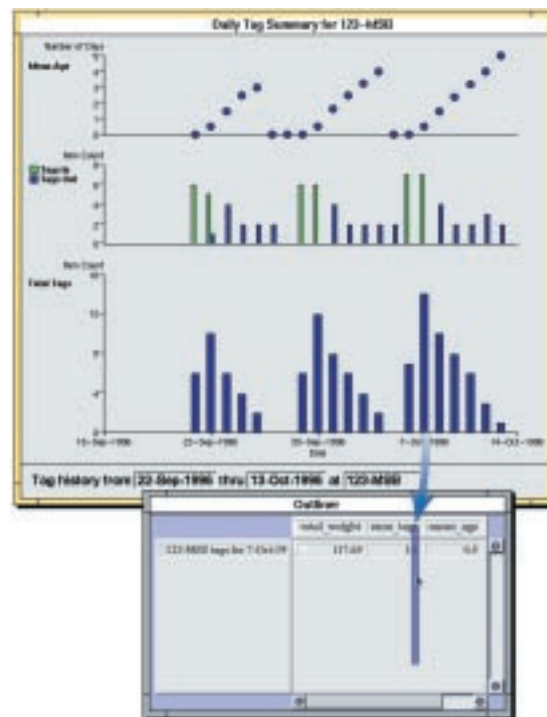
ment is a transmitter, called a *tag*, that reports the shipment's location. As each tagged shipment arrives at a staging area, an entry is automatically entered into the database.

Sometimes a user wants to locate an individual shipment. More often, though, analysts want to monitor the whole transportation and distribution system's performance. Users search out problem areas and either route shipments around them or assign additional resources (such as trucks and personnel). Analyses often consider both current and historical information about the location, movement, quantities, status, and priority, and related information about shipments and the units that requested them. Analysts frequently need to check the status of an intermediate staging area in a distribution network, asking questions like, "How long have shipments been waiting at a particular staging area?" "How much cargo is heading from this area to other areas?" "What's the rate at which a staging area has been handling cargo (receiving and moving it out)?"

Figure 1 shows two Visage frames. The top frame contains a map showing part of a shipping network; the bottom frame is an aligned chart currently showing no data. Frames are information displays that provide different views and arrangements of data. All the interfaces and visualizations in this example are contained in Visage frames. Frames provide operations that are universal to the Visage environment as well as specialized to the unique task that the frame was designed to support. We discuss frames in detail later.

The user creates the "Routes" frame at the top of Figure 1 by dragging an empty instance from a palette of commonly used frames, then selecting a region of the shipping network and a time interval from a menu within a *caption interface* attached to the bottom of the

2 The populated Daily Tag Summary frame shows historical information for the tags arriving, departing, and remaining daily at a specified staging area. The data within the frame can be specified using the caption interface or by dropping data on it from other frames. A collection of tags remaining at the staging area on October 7th are dragged to an Outliner frame for further analysis.



data. This appliance takes as input a staging area and a range of dates for which information should be shown. For each day in the range of dates, the frame shows several attributes for the specified area (see Figure 2). The horizontal axis indicates date. The bottom chart shows the number of shipments remaining at the staging area at the end of each day. The middle double bar chart shows the number of new tagged shipments arriving and leaving the staging area each day (green signifies arriving and blue departing shipments for that day). In the top chart, the points show the mean age of shipments at the specified staging area at the end of each day (where age equals the number of days that a shipment has been at the staging area).

This example illustrates a convenient operation for populating frames with data. A subset of data represented by a graphical element is copied and dragged from the

frame. In this case, the user selects the region “Bosnia” and the time interval “last 3 weeks.” The caption interface responds by extracting the relevant data from the database and forming collections of shipments passing through each staging area in the specified time interval. The resulting collections appear as circles on the map. The area shows the total number of tagged shipments that have moved through a staging area in the specified time interval. The color shows the average age of tags, defined as the number of days that a tagged shipment remained at a staging area before it was moved elsewhere. The links show the shipment routes between staging areas (straight lines indicating air movements and irregular lines indicating truck routes).

Used in this way, the routes frame is an *appliance* with two simple controls (date and location menus). It gives users an overview of the state of part of the shipping network during a particular time period. The caption at the bottom of the display serves two purposes. It reminds the user what data appears in the frame, but it also represents an interface to specify inputs for populating the frame with data.

A key user interface feature in Visage is the ability to drag graphical objects representing information among visualizations and other application interfaces in the workspace. For example, a user noting the large number of aging shipments indicated by the large yellow circle (which represents a supply handling area called the “123rd-MSB”) can call up more detailed data by dragging a copy of the circle to an empty “Daily Tag Summary” frame (the operation suggested by the blue arrow in Figure 1).

As with the initial display, the “Daily Tag Summary” frame is a specialized appliance that extracts data from a database, aggregates it in a prescribed way, and creates a set of aligned charts showing a particular view of the

Routes frame into the Daily Tag Summary Frame. The caption interface uses the underlying data to compute a date range and identify a staging area of interest for input to its data reorganization script. In this case, performing the drag and drop operation is much easier than selecting the corresponding staging area and time interval from the menus in the caption interface. Hence, the user can remain focused on the analysis task at hand.

Examining the Daily Tag Summary frame, the analyst first notices a strong periodicity in the data. Each week, a large number of shipments arrive on two consecutive days (indicated by the three pairs of green bars in the middle bar chart) and are delivered to their subsequent destinations during the remainder of the week (indicated by the blue bars in the middle bar chart). It appears to the analyst that this staging area (the 123rd-MSB) is operating near its capacity because each week’s incoming shipments barely move out before the next week’s shipments arrive. The analyst also notes that the number of shipments processed early each week seems to be higher than those processed on later days.

To gain more visibility into the apparent periodic decrease in throughput, the user decides to analyze the most recent week’s shipments. This is accomplished by dragging a bar from the bottom-most chart to the *Outliner* frame, also shown in Figure 2. This bar represents all tagged shipments that remain in the 123rd-MSB on October 7th.

The *Outliner* provides a hierarchical, spreadsheet-like interface that enables the user to view multiple attributes of the data textually while performing *drill-down* and *roll-up* operations.⁵ (Drill-down commonly refers to the process of segmenting or breaking down collections of data along different dimensions to create a larger number of smaller collections; roll-up commonly refers to the process of merging detailed data into col-

lections that summarize their attributes.) On the right side of the Outliner in Figure 2, the user selects the attributes of the objects in the first column to display, such as the total cargo weight, number of tags, and mean age of the shipments contained in each collection. These attributes can be taken from the database or created dynamically as derived attributes using a scripting language.

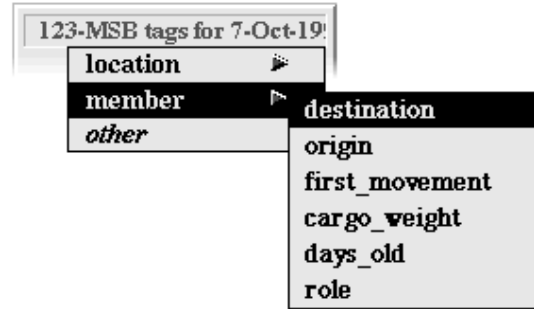
Starting from the aggregate containing all the tagged shipments located at the 123rd-MSB on October 7th, the analyst considers a menu of alternative dimensions along which to drill down (illustrated in Figure 3). Choosing the attribute called *members* would drill down to all the shipments represented by the aggregate (that is, the shipments in this set). There are too many to view individually, so the user chooses to *recompose* or reorganize this set of shipments into new collections, grouped by the shipment's "destination" attribute. The result is a set of collections, each containing all tagged shipments going to a particular destination, shown highlighted and indented in the Outliner frame in Figure 4. Such dynamic drill-down and roll-up capabilities are a fundamental operation in the Visage environment and can occur in every display.

A flexible tool, the Outliner requires awareness of these basic operations as well as some awareness of the underlying object's structure. It thus demands more experience from users than the two previous frames, though they also provide the same drill-down operations on their elements in addition to their simpler caption-like appliance controls.

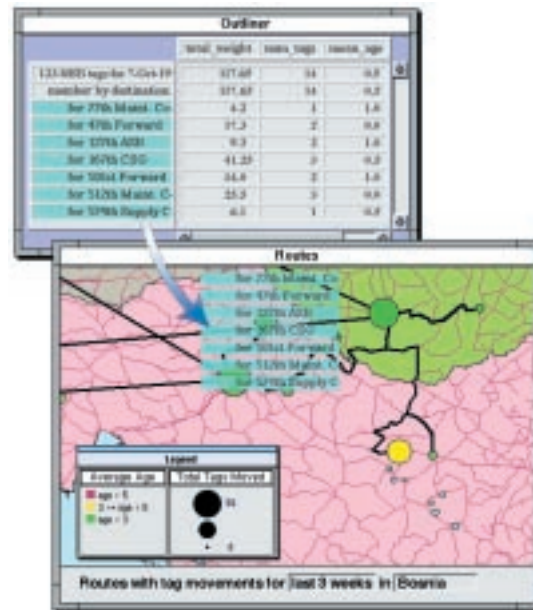
In Figure 4, the analyst drags the seven newly created collections of shipments to the original map frame (shown in Figure 1) to view them at their subsequent destinations. Recall that the resulting seven points represent a reorganization of part of the shipments represented by the yellow circle, categorized by their destination. In other words, the seven circles represent the destinations of shipments originating at the yellow circle.

The circles form two clusters south of that location. The analyst selects an orange *painting* color and paints one cluster of shipment destinations on the map by dragging a bounding box around them. Note that the corresponding items in the Outliner display are also painted orange (see Figure 5). Painting is another fundamental operation in the Visage environment, coordinated across all frames.

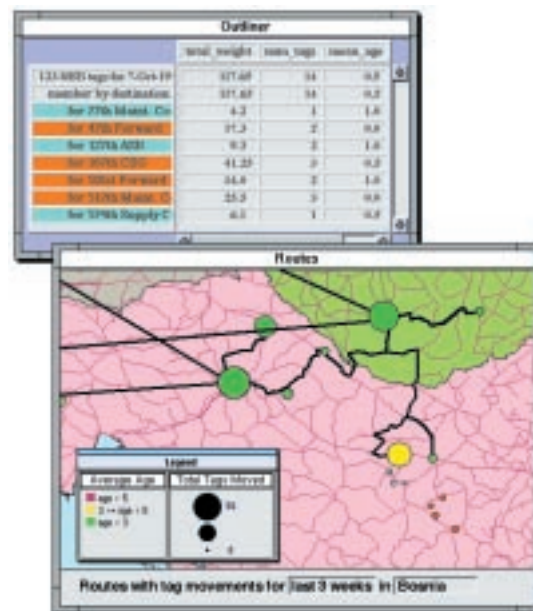
From these two coordinated views of the data, the analyst gets a clearer picture of the problem. The 123rd-MSB supplies several quite distant destination units that represent the majority of actual cargo weight shipped. This explains the drop in the throughput after the first day—trucks take a long time to deliver these shipments and return for more cargo. A possible solution is to reroute some shipments to these destinations through another staging area (the smaller green circle) or to assign the 123rd-MSB additional truck crews.



3 In the Outliner, the user performs a *recomposition* operation on the members of the collection representing tagged shipments located at the 123rd-MSB on 7-Oct-1996. In one step, these tagged shipments are reorganized into new collections based on their destination.



4 Newly recomposed collections of shipments are dragged to the original Routes frame, where they are shown at their destinations.



5 A cluster of elements is painted orange in the Routes frame, causing elements representing the same concepts to be painted in the Outliner.

This example illustrates several important aspects of the Visage environment:

- Information is represented by first-class graphical objects that users manipulate directly and can copy and transfer across frames.
- Painting operations are applied to objects and coordinated in multiple frames via a common underlying object database.
- Drill down, roll up, and other general-purpose operations can be performed directly on objects in all frames, again through the common underlying object representation.
- General-purpose operations are available for experienced users; special-purpose appliance-like interfaces are available in some frames for routine tasks and for occasional users. The latter may use appliances to obtain basic information without using any of the drag, drill-down, or other operations.

Visage goals

In designing Visage we aimed to incorporate basic information exploration components within a consistent user interface containing specific key elements:

1. A uniform object-oriented data space. The Visage object representation provides a common medium for user interactions and data sharing among applications with interfaces implemented in the Visage environment.
2. A consistent information-centric user interface paradigm. As the name implies, this paradigm strives to give users greater direct contact with objects representing 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, desktops, briefing materials, or anywhere else people elect to place it. It is *user-centered*.
3. Interactive information manipulation. Tools include those for
 - finding and interactively partitioning, filtering, and selectively combining subsets of data;
 - controlling the level of detail for viewing information using drill-down and roll-up techniques; and
 - assembling, laying out, and interactively presenting information.
4. Dynamic visualization generation. To provide integrative views of information, we are incorporating work on Sage, a knowledge-based automatic graphic design tool.³ This approach rapidly generates visualizations customized to users' immediate data exploration tasks.

The information-centric approach

The Visage user interface paradigm takes an aggressively information-centric approach to presenting infor-

mation to users. By information-centric we mean that data presented in any frame can be manipulated with every basic Visage operation. Thus, users can drill down, highlight, and copy information whether represented as a graphical element in a visualization, a briefing slide, a user interface to an application, or even as an icon on a desktop.

This approach makes it possible to select or paint objects in one area of a Visage environment and cause coordinated painting of corresponding graphical elements in all other frames in the environment. In other words, Visage permits a one-to-many mapping of data to visual objects.

Information-centricity and the corresponding ability to coordinate and drag information between user interfaces for different applications arises from Visage's use of a shared data repository. Conceptually, information is stored as data objects with unique identifiers that multiple applications can reference when appropriate. The data realm is then mapped to a graphical realm permitting coordinated access to the common underlying repository in a uniform way from all applications and visualizations. This common data repository and consistent mapping to graphics enable the graceful coordination of special-purpose, customized appliances with more primitive operations performed throughout Visage frames.

We see the information-centric approach as the next logical step along the path from application-centric architectures to the modern document-centric approach. The distinctions among the three approaches hinge on differences in the *basic currency* through which 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 are the basic unit of information, they themselves are of little use to the user. To access the information in them, users must rely on *applications* to fetch and display the information on their behalf. In this regard, applications resemble remote manipulator arms in nuclear power plants—users may not touch their data except through special-purpose tools. Each application has its own user interface defining 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. Users could manipulate them directly, store them on the desktop or in folders, and—to a limited extent—arrange them in semantically meaningful ways. But the contents of those files remained out of the user's direct reach.

The advent of document-centric interface paradigms 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 application's role is subordinated (and perhaps ultimately eliminated) in favor of component architectures whose interactions with the user focus on direct manipulations of documents.

Documents may reside on the desktop, where the user can directly activate and manipulate them via drag-and-drop operations. Documents may serve as containers for other documents, enabling natural modes of grouping and attaching information in meaningful units. Some extremely document-centric interfaces (such as Workscape⁵ and Web Forager⁶) permit the spatial arrangement of large numbers of documents, enabling effective visualizations of the relationships among them. Applying dynamic query techniques in a document-centric world enables visual search paradigms. In document-centric interfaces, users can almost “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. 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 interface “physics.”

The object-oriented nature of this approach clearly is not unique to Visage and indeed was introduced and explored in Smalltalk.⁷ Work done more recently on ARK⁸ (Alternate Reality Kit) and Self⁹ have applied similar principles to simulation and programming environments. Such object-focused environments eschew the use of separate views and tools to make the objects themselves the focus of attention.

The design principles of *directness* and *liveness*¹⁰ are central to Self, Workscape, and Visage. Operations are always performed directly on representations of the objects themselves rather than through some alternative representation or tool. Objects respond to user actions or the influence of other objects in an active and animated fashion.

However, there are important differences. To emphasize the concreteness of objects, both Self and Workscape use a one-to-one mapping of objects or documents to their interface representations. In Visage, it is often desirable to examine multiple perspectives on the same object. For example, in Figure 5 the user examines the same objects in the Outliner and on the map. Visage’s focus on information visualization makes it essential to allow multiple simultaneous views of the same object. Unfortunately, this makes the objects somewhat less concrete. Ubiquitous coordinated painting reinforces the underlying connections between graphical elements representing the same objects.

Main components of Visage

The Visage interface environment minimizes the number of fundamentally different kinds of objects users must understand. Objects exist in either the data realm or the graphical realm.

Object-oriented data representations

Visage uses an object-oriented representation for data in its database. From a user’s perspective, Visage has two

basic object types: *concepts* and *collections*. Concepts are simply buckets of attribute-value pairs, with values being either literals such as integers, strings, and dates or links (relations) to other concepts. Collections—themselves represented as concepts—are distinguished because users are encouraged to view them as groupings (aggregates) of other concepts.

Visage employs a much simpler object model than traditional object-oriented systems—focused less on the encapsulation and representation of *verbs* and more on *nouns*. In other words, rather than focusing primarily on the encapsulation of program code as methods (that is, the verbs of the environment), Visage represents information in a simple, universally accessible form (nouns). Rather than hiding data details, it makes the contents of objects available for examination and use by all applications and users.

In the graphical realm, Visage has only two basic object types: *visual elements* and *frames*. The term visual elements (or simply elements) refers to any atomically-manipulable graphical object in a Visage display. Examples of elements include 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 a concept in the database. Note that this relationship is one-to-many: each element is associated with exactly one concept, but the same concept may be represented by multiple elements.

Data manipulation with frames

Frames serve as pasteboards for elements. Strictly speaking, frames are themselves elements, but 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 subject to the entire repertoire of direct-manipulation actions available for other elements (duplication, drag and drop, dynamic scaling, and so forth). Indeed, as an element a frame is represented by a single underlying concept, which typically serves as an aggregate (collection) of all data concepts represented within the frame and all graphical elements contained in it (for example, the elements representing the data concepts, graph axes, table columns, map guidelines, and other visualization components).

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.

In Visage all visualizations consist exclusively of collections of elements arranged by the frame to form the display. For example, the chart in Figure 2 is not a dis-

Rather than hiding data details, Visage makes the contents of objects available for examination and use by all applications and users.

crete picture, but rather an arrangement of elements that a user can break apart and manipulate separately. In this sense, Visage frames are not so much drawn as composed of arrangements of elements. As the illustration shows, this makes it easy for a user to select bars from a frame and drag them to another frame. This ability to directly drag small collections of data forms the basis of the information-centric approach to interface design described above and enables the coordination of appliances and tools.

Visage needed to support a range of users and tasks. A continuum of interface types emerged along the tool-appliance spectrum: primitive tools, power tools, and appliances.

Hints of this approach appear in a few existing interfaces. For example, Microsoft Word supports 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. On some platforms, Netscape lets users drag images from Web page windows onto the desktop (though they are immediately replaced by icons and hidden within files).

Likewise, several visualization tools support representing data objects graphically and provide filtering, painting of linked displays, and related operations.¹¹⁻¹⁴ Visage promotes these capabilities from special-purpose features to capabilities provided everywhere in the environment. They become part of the basic physics of the interface, empowering the user to directly perform simple actions where other systems require the use of complicated or inconsistent interface features.

Scripting for customization

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.

The scripting environment delivers to the user the drill-down and roll-up data navigation features described above. 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 resembles HyperTalk or Visual Basic and contains language features tuned to data navigation and aggregation functions (such as for stepping across links among objects, iterating over object sets, and accumulating sums of values of quantitative attributes).

Scripting provides a mechanism for specifying *derived attributes*, an essential feature supporting Visage's data exploration operations. Although the underlying database being explored may have many data values directly given, many other 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.

Finally, the process of rendering a visualization is the responsibility of a frame's script. For both Sage-generated and hand-built frames, the frame's scripts manage creation and arrangement of elements within a frame. When elements are dropped into a Visage frame, its script identifies the concepts represented by the elements, determines those appropriate to the frame, and directly maps these concepts to new visual elements of a type appropriate to the frame (such as text for an Outliner table or bars for a chart). These new elements are then arranged in the style the frame specifies.

Although scripts may be attached to any element, many of a typical interface's scripts are associated with frames. In the illustrated example, the script of the "Daily Tag Summary" frame performs multiple data processing and visualization operations. A collection of tagged shipments dropped on the frame is broken into multiple collections based on the date each shipment arrives at the staging area. Summary attributes are computed for each of these collections. Finally, graphical elements are created and spatially arranged within the frame. In this way, a number of such scripted frames can be created to form highly customized application environments, tailored to a particular user's needs.

Tools and appliances in Visage

Any complex user interface design inevitably trades off between interface techniques that maximize flexibility and access to the system's power and those that optimize the system's approachability by novice and casual users. Typically, the more general a device, the more difficult it is for untrained users to achieve a high level of usability. Conversely, the more specific (special-purpose) a device, the easier it is to deliver the device's functionality without exposing the user to complexity. The reason for this trade-off is obvious: A special-purpose device needs to ask a user fewer questions to determine exactly what is required of it, whereas a general-purpose device—being more flexible—must somehow make its entire repertoire of operations available to a user.

You can, of course, attempt to mitigate this trade-off in many ways. For example, functionality can be layered, with advanced features hidden from a beginning user. Novice and expert modalities may be added to the system in an attempt to accommodate both user groups. Shortcuts may be added to permit experts to bypass tedious novice interfaces. Products may be shipped in one version for casual users, with a separate professional product for the power user.

Such attempts, however, often fail. Simplified products are often seen by the market as "dumbed down" (a process the *New York Times* termed "Bobbing," in reference to Microsoft's Bob interface). On the other hand, users often fail to make the sometimes abrupt leap from novice to expert mode in products so equipped. In contrast, a skillful designer will not only provide for both modes of usage, but will also explicitly design an incre-

mental path from novice to expert usage. Moreover, an ideally designed interface would support frequent and casual switching between novice and expert modes.

In Visage, we addressed these needs in the context of a single data exploration and visualization environment that provides functionality in two distinct but seamless modes. The first mode is represented by a set of general-purpose capabilities delivered to the user as tools. Like real tools, they are very general and flexible, but require the expert's skilled hand to be used effectively. They are *composable*—they can be combined in unique ways to perform sequences of operations. However, such composition requires knowing how to select the right tools and sequence to achieve each information analysis goal.

The second mode is embodied in an open-ended set of scripted appliances specialized to meet the needs of specific users, tasks, and contexts. Appliances are much less general than tools, but they tend to be easy to learn and use because of their specificity. Appliances tend not to be composable—the results they generate cannot typically serve as input to other appliances. These properties produce both ease of use and inflexibility in most systems. In Visage, we have attempted to achieve flexible composition by supporting transfer of information among appliances and tools, and direct application of tools in specialized appliances.

The techniques described here generalize an approach pioneered by computer spreadsheet programs. In typical use, spreadsheets are more than just applications. They are media for power users creating special-purpose analysis appliances—for use in very specific situations by very specific user communities. This ability to let a few power users support a larger community of end users without expert programmers in the loop made spreadsheets the first enormous success in the personal computer market. Spreadsheets also demonstrated the value of providing a continuum of interfaces, ranging from primitive basic operations to so-called power tools and macros, and customized programs.

Similarly, in Visage we needed to support a range of users and tasks. The tasks varied in how frequently they were performed and whether they were performed in relative isolation of each other or flexibly combined and coordinated. A continuum of interface types emerged along the tool-appliance spectrum. We refer to these as primitive tools, power tools, and appliances.

Primitive tools

Primitive operations in Visage—available throughout the entire environment—are applied to elements in all frames. A brief overview here describes their relation to more appliance-like interfaces. (Details appear elsewhere.⁴) Primitives include elemental data manipulation, filtering, and visualization operations:

- *Navigation or drill-down* involves traversing a link from a data object represented by a visual element to related objects in the database.
- *Drag-and-drop filtering* controls the set of objects and attributes shown by removing them from one frame and/or combining them from multiple frames to another frame.

- *Creating collections* involves creating a new collection from a group of selected concepts and summarizing the attributes of those concepts (for example, means, totals, or ranges of the collection).
- *Coordinated painting* means that one element painted a certain color engenders the same color on all elements showing the same underlying concept. Painting in Visage is globally coordinated across all frames, enhancing a user's ability to identify related information. Painting is composable with other operations: painted elements can be copied and/or dragged as a group to other frames.
- *Dynamic query* controls objects shown in a frame with a dynamic query slider or other in-place interface to adjust the scope of information viewed. Dynamic query tools¹² provided in Visage permit interactive control of elements' visual properties based on attributes of underlying data (typically visibility, though sliders can control color and other properties as well). Analysts may add sliders to frames to select a subset of objects, then drag the subset to other frames for other analyses. Likewise, a subset can be aggregated into collections and summarized.
- *Visualization creation* involves selecting the frames of reference, the visual elements to appear within them, and the mappings from data to the visual elements' properties. SageBrush,³ an interface for designing new visualizations, provides a set of primitive operations for selecting and composing graphical objects (such as axes, tables, networks, bars, points, links) and for picking properties of these objects to encode data attributes (such as position, color, size, or shape). The resulting visualizations are subject to all the above operations.

Power tools

Power tools typically consist of a sequence of primitive operations structured to let a user perform a frequently needed task more quickly. Some are tailored to a particular analysis domain, others are general operations that simplify what would otherwise require numerous repetitive steps. Power tools are still composable operations and tend to be available in most Visage interfaces. Examples include

- *Recomposition*—grouping objects (retrieved by navigating across a relation) by common values of a specified attribute. Recall from the example, the elements of the “October 7” aggregate were recomposed by the destination attribute. Without recomposition, a user would list all the elements of a collection (the shipments), select all that share a particular value for “destination” (all shipments going to New York, for example), and create a new collection composed of just those elements. Repeating this step for each unique value in destination yields the same result as the recompose operation.
- *Scripted derived attribute*—a new attribute whose value is determined by executing script code.
- *Scripted navigation paths*—the creation of shortcut relations (paths) among objects. For example, a unit makes a requisition, which is filled by a tagged ship-

ment, which is currently at a location. A scripted navigation path can be added to the unit concept to represent this path directly as a single link called the `order_location` if this is a frequently traversed set of links.

- Visualization reuse—the storing and reuse of a visualization by creating a “stationery pad” from which a user may tear off new visualizations. Thus, once a visualization has been specified through basic design operations, it becomes available as a power tool for making new visualizations of the same type.

In Visage, power tools such as recomposition menus are available anywhere an element appears—in the Outliner, on a map, or in a chart. Although Visage’s recomposition mechanism provides a general capability to reorganize collections in flexible ways, note that users must understand the database structure to use this mechanism effectively. Recomposition requires knowing the names of the relations and attributes that will

be used in the operation. Furthermore, analysts must recognize the recomposition tool as a means to achieve their analysis goals.

In contrast, when recomposition operations are captured within an appliance-like interface, the appliance builder specifies these options, freeing the user from knowing the specifics of the database structure and enabling users to communicate at a much higher, but less flexible level. For example, the Routes appliance collected shipments passing through specified staging areas during the specified time period. The user didn’t need to know the specific objects and relations used to define those sets.

Appliances

Appliances are specialized tools that perform operations specific to an application domain, a user, or a particular analysis. Appliances typically allow the user to perform an analysis task without requiring knowledge of the structural details of the database. Appliances are usually frames with a small number of specialized interface controls. Examples include the caption interfaces at the bottom of the Routes and Daily Tag Summary frames described in the example.

Just as recomposition rolls together a sequence of primitive navigation and composition steps, appliances encapsulate operations by rolling together a sequence of primitive and scripted operations. Appliances can also capture operations not expressible using primitives because they require visualization operations or data manipulations that can only be accomplished in script. An example might be an appliance that performed a clustering algorithm, generating a collection for each cluster of values.

Operations encapsulated within an appliance can be parameterized by simple interface controls. For exam-

ple, the controls on the caption on the Daily Tag Summary frame (in Figures 2 and 3) let the user specify a staging area and date range. The appliance uses these values as parameters to a script that filters the data to appear in the frame.

In addition to specifying data input to appliances using special-purpose interfaces, users can drag and drop subsets of objects onto their frames. In the example, the Routes frame pulled the data directly from the database in response to the user selecting a staging area and date range. The Daily Tag Summary frame used the object dropped by the analyst to determine which data to extract and visualize.

Primitives and power tools realize much of their power through their ability to compose multiple operations into expressive queries. In contrast, appliances are generally not as composable because they do not provide access to the intermediate steps leading to the results they generate. They are therefore not customizable beyond those parameters that may be changed in the interface.

However, well-constructed appliances still follow the information-centric paradigm. They enable the user to drag elements representing results to other frames, add dynamic query sliders to filter data, and perform drill-down operations.

Conclusions

Combining different interaction styles allows for more effective user interface environments for data exploration. The ability to dynamically combine appliances with primitive operations lets users choose among alternative processes for achieving analysis goals based on personal expertise. More experienced users can choose more powerful tools as appropriate to their tasks, while novice users can still perform analyses by selecting appliances.

As system designers, we try to observe and anticipate the steps our users perform frequently during their analyses. We build appliances to support these steps, enabling people to perform them quickly and easily. We further support users by making it easy to shift from routine data access and report generation to exploratory data analysis.

By holding steadfastly to the information-centric approach, we compound the usefulness of our interfaces. Elements can be dragged from each frame to eliminate irrelevant data, to focus on subsets in new ways in other frames for further analysis, or to categorize subsets of data from the original frame by painting them in the context of attributes viewed in a new frame.

The universal availability of general-purpose tools and primitive operations in the Visage environment enables users to analyze data in ways that developers cannot anticipate. Because users aren’t limited to just the appliances we provide, they can at any time consider alternative steps that better serve the analysis at hand. ■

References

1. R. Rao and S.K. Card, “The Table Lens: Merging Graphical and Symbolic Representations in an Interactive

The universal availability of general-purpose tools and primitive operations in the Visage environment enables users to analyze data in ways that developers cannot anticipate.

- Focus+Context Visualization for Tabular Information," *Proc. CHI 94 Human Factors in Computing Systems*, ACM Press, New York, 1994, pp. 318-322.
2. C. Ahlberg and E. Wistrand, "IVEE: An Environment for Automatic Creation of Dynamic Queries Applications," *Conf. Companion, CHI 95 Human Factors in Computing Systems*, ACM Press, New York, 1995, pp. 15-16.
 3. S.F. Roth et al., "Interactive Graphic Design Using Automatic Presentation Knowledge," *Proc. CHI 94 Human Factors in Computing Systems*, ACM Press, New York, 1994, pp. 112-117.
 4. S.F. Roth et al., "Towards an Information Visualization Workspace: Combining Multiple Means of Expression," *Human-Computer Interaction*, Vol. 12, No. 1-2, Lawrence Erlbaum, Mahwah, N.J., in press.
 5. J.M. Ballay, "Designing Workscape: An Interdisciplinary Experience," *Proc. CHI 94 Human Factors in Computing Systems*, ACM Press, New York, pp. 10-15.
 6. S.K. Card, G.C. Robertson, and W. York, "The WebBook and the Web Forager: An Information Workspace for the World-Wide Web," *Proc. CHI 96 Conf. on Human Factors in Computing Systems*, ACM Press, New York, pp. 111-117.
 7. L. Tessler, "The Smalltalk Environment," *Byte*, Vol. 6, No. 8, Aug. 1981, pp. 90-147.
 8. R.B. Smith, "Experiences with the Alternate Reality Kit, An Example of the Tension Between Literalism and Magic," *Proc. ACM CHI+GI 87 Conf. on Human Factors in Computing Systems and Graphics Interface*, ACM Press, New York, 1987, pp. 61-67.
 9. B.W. Chang, D. Ungar, and R.B. Smith, "Getting Close to Objects: Object-Focused Programming Environments Published in Visual Object-Oriented Programming," M. Burnett, A. Goldberg, and T. Lewis, eds., Prentice Hall, Greenwich, Conn., 1995, pp. 185-198.
 10. J.H. Maloney and R.B. Smith, "Directness and Liveness in the Morphic User Interface Construction Environment," *Proc. ACM UIST 95 Symp. on User Interface Software and Technology*, ACM Press, New York, 1995, pp. 21-28
 11. S.G. Eick and G.J. Wills, "Navigating Large Networks with Hierarchies," *Proc. Visualization 93*, IEEE Computer Society Press, Los Alamitos, Calif., Oct. 1993, pp. 204-211.
 12. C. Ahlberg and B. Shneiderman, "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays," *Proc. CHI 94 Conf. on Human Factors in Computing Systems*, ACM Press, New York, 1994, pp. 313-317.
 13. J. Goldstein et al., "A Framework for Knowledge-Based Interactive Data Exploration," *J. of Visual Languages and Computing*, Vol. 5, No. 4, pp. 339-364.
 14. J.A. McDonald, "Painting Multiple Views of Complex Objects," *Proc. ECOOP/OOPSLA 90 Conf.*, ACM Press, New York, pp. 245-257.



John Kolojechick is a researcher at the Intelligent Visualization and User Interfaces Lab at Carnegie Mellon University. His research focuses on developing techniques and tools to help people more effectively organize, understand, and exchange information. Current projects include construction of improved human-computer interfaces for information manipulation and the use of AI techniques to automatically generate effective visualizations. He is a member of the ACM and the IEEE Computer Society.



Steven F. Roth directs the Intelligent Visualization and User Interfaces Lab at Carnegie Mellon University. His research is on techniques, environments (such as Visage), and automated systems to support information visualization and exploration. The Sage project has focused on systems that enable automatic and user-directed design of data visualizations. Related research on AutoBrief is developing systems that automatically generate explanations and summaries of patterns and changes in quantitative and relational data using coordinated text and graphics.



Peter Lucas is president of Maya Design Group, a product design consultancy he cofounded in 1989. He holds a PhD from Cornell University, where he studied cognitive psychology, education, and psycholinguistics. He was also a Sloan Fellow in cognitive science at Carnegie Mellon University. He is a member of the IEEE Computer Society, the Industrial Designers Society of America, and the American Association for the Advancement of Science.

Readers may contact Lucas at Maya Design Group, 2100 Wharton St., Pittsburgh, PA 15203, e-mail lucas@maya.com, Web site <http://www.maya.com/>. Contact Kolojechick and Roth at Carnegie Mellon University, School of Computer Science/Robotics Institute, 5000 Forbes Ave., Pittsburgh, PA 15213, e-mail jake@cs.cmu.edu and roth@cs.cmu.edu. Their Web site is <http://www.cs.cmu.edu/~sage/>.