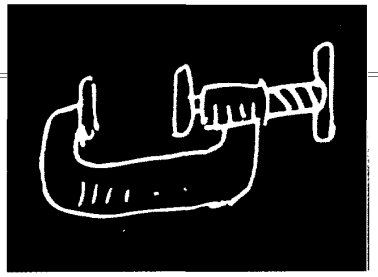


Image-Browser Taxonomy and Guidelines for Designers

CATHERINE PLAISANT, DAVID CARR, and BEN SHNEIDERMAN
University of Maryland

◆ *In many applications users must browse large images. Most designers merely use two one-dimensional scroll bars or ad hoc designs for two-dimensional scroll bars. However, the complexity of two-dimensional browsing suggests that more careful analysis, design, and evaluation might lead to significant improvements.*

The one-dimensional scroll bar is a well-established fixture in contemporary graphical user interfaces. For example, in word processors one-dimensional scroll bars help users navigate long documents. Without a scroll bar users must remember their position and use some command to jump within the document (for example, "173,193p" to display lines 173 through 193). Scroll bars let users move through the document incrementally and by jumps, and they indicate the current position of the screen. This visual feedback probably reduces memory and cognitive load.

Although all one-dimensional scroll bars have a common core functionality, their individual features and operation differ substantially. But because users quickly accommodate these differences, research on scroll bars is limited.[1-2]

Building on user familiarity with one-dimensional scroll bars, many designers simply use two one-dimensional scroll bars when the application requires independent control over the horizontal and vertical directions, as in panning a map. This is effective if users frequently move in a single direction by small increments of less than one screen.

But in many cases this solution is inadequate:

- In painting and drawing programs, the image is often much larger than a screen, redisplay times are long, overviews are needed,[3] zooming is desirable, diagonal panning is required, or multiple detailed views are needed.

0740-7459/95/$04.00 © 1995 IEEE

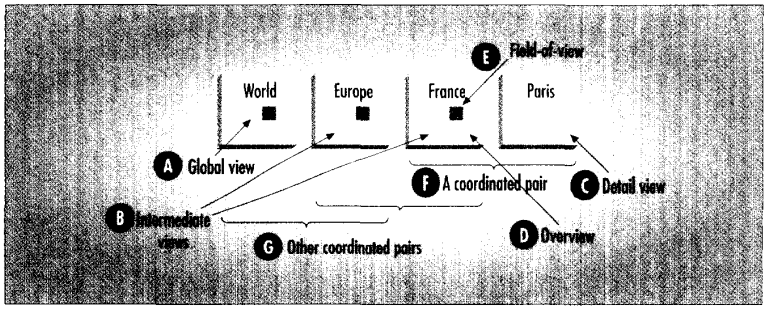


Figure 1. *Illustration of some terms used to describe browsers. First, a global view* ***(A)*** *gives a view of the entire universe that can be explored. One purpose of the global view is to give a sense of what information will be in the image — and what is not. For example, a world atlas global view might show a map of the earth, telling the user that this "world" doesn't include the moon or the galaxy. An intermediate view* ***(B)*** *of the world map would include maps of Europe and France. The detailed view* ***(C)****, also called the local view, shows a part of an overview* ***(D)****, usually magnified. The level of detail required depends on the task to be performed. The zooming factor describes the level of magnification between two views. The field-of-view* ***(E)*** *indicates on the overview the location and shape of the coordinated detailed view. Taken together, an overview and detailed view are called a coordinated pair* ***(F)****. In a coordinated pair, both the overview and detail are shown, letting users keep a sense of context while they view detail. Several coordinated pairs can provide a hierarchy of views, in which the detailed view of one pair becomes the overview of another pair* ***(G)****. The global view is often used as an overview in a coordinated pair. But if the maximum zooming factor is too large, an intermediate view is called for.*

(See Figure 1 for short definitions of some key terms we use in this article.)

- In geographic information systems, users browsing a world map may want to see detailed views of a country, county, or city. The world map provides a helpful — possibly necessary — overview, and the system must then support a zooming factor of 1 to 10, 1 to 10,000, or even 1 to 10,000,000. In addition, users may want to follow the route of a river, border, or highway (diagonal panning), compare two harbors (multiple detailed views), or simultaneously view highways and population-density maps (related views).
- In medicine, a doctor may need to see a full spinal X-ray and close-ups of vertebra pairs (an overview and multiple detailed views) or to examine a tissue boundary (pan a detailed view).
- In large applications such as power distribution, telephone networks, system administration, transportation systems, and chemical plants, managers typically use an overview diagram to monitor the system and detailed views to focus on anomalies. Some problems can be solved with local information only, but other problems require multiple detailed views or an understanding of the big picture.

All these situations call for browsing in two or more dimensions, and their requirements suggest that more careful analysis,[4] design, and evaluation might lead to significant improvements. Indeed, our exploration of existing 2D browsers has led us to identify many features and a wide variety of tasks performed with the browsers. Here we introduce an informal specification technique to describe 2D browsers and a task taxonomy, suggest design features and guidelines, and assess existing strategies. We focus here on the tools to explore a selected image and so do not cover techniques to browse a series of images (via, for example, a radiology workstation that shows dozens of images) or to browse large-image databases (via thumbnails or graphical searches, for example).

BROWSER SPECIFICATION

When we began to explore browsers, we found it difficult to even discuss our findings because there was no adequate method to describe browser features. This led us to expand a sketching technique, DMsketch (direct-manipulation sketch),[1] being developed in our laboratory. We had created DMsketch to help designers exchange and record ideas more quickly and clearly than a formal specification language.

Originally, DMsketch included icons to represent single clicking, double clicking, dragging, and so on. But this detail is too low-level for our purposes, so we extended DMsketch to show the major differentiating characteristics of browsers. With DMsketch, designers can informally specify

- a browser's most significant graphical elements,
- the interrelation among those elements, and
- the most important possible user actions.

DMsketch is based on a technique from both Scott Hudson and Shamin Mohamed's graphical specification of layout constraints in the Opus system.[5] Hudson and Mohamed introduced the idea of graphically representing a constraint on the layout of a user interface. They used an arrow to represent the presence of a constraint, which is a hidden equation. The layout designer views the equation by pointing at the constraint arrow.

However, we believe that equations do not convey meaning as clearly and quickly as a few specialized graphics. Moreover, equations cannot specify that an area in one window will be viewed in greater detail in another. In specifying browsers, we are not so much concerned with the details of interface operation at the keystroke level as we are with the relationships among windows.

Primitives. Figure 2 shows a few primitives used in our notation. As we describe browsers in this article, we will add new primitives and define composite objects as necessary.

- *Movement constraint.* The movement-constraint operator in Figure 2a specifies that the object at its tail is

movable. If the arrow is horizontal, it is movable in the horizontal direction. If vertical, it is vertically moveable. An object without a movement-constraint operator attached is not moveable. The movement of the object at the tail is limited to be within the context of the object at the head of the arrow.

♦ *Proportional size constraint.* The proportional size-constraint-operator in Figure 2b joins two objects by its circle end points. The proportional size constraint forces the two joined objects to maintain the same relative size. For example, a line whose maximum length is four might be joined to a line whose maximum length is two. If the longer line is shortened to two, the smaller line is automatically shortened to one. This constraint operates both ways, so changing the shorter line changes the longer as well.

This operator is not confined to lines; 2D objects and movement-constraint operators may also be joined. We characterize the existence of such bidirectional links between user-interface elements by the concept of *tight coupling*.

♦ *Field of view.* The field of view encloses an area of an image and is displayed on the window that contains the image. They define a clipping rectangle for the image in its underlying representation. This means that images enclosed by a field of view do not automatically become coarser as they are magnified. This happens only when the maximum resolution of the underlying representation is reached.

The contents of the field of view are projected into a new window, which is identified by an arrow that points from the source field of view to the destination window.

Figure 2c shows the generic field of view; Figure 2d shows a generic field of view constructed by defining two points that represent its corners. This rectangle is typically defined by a "mouse down, drag, mouse up" operation. The field of view in Figure 2e is similar to one in Figure 2d, except that the point defines the center of the field of view instead of one corner. The field-of-view operator in Figure 2f represents a window that is always the same size and is defined by one point. The field of view in Figure 2g represents a view with several magnifications available, and Figure 2h shows a field of view with a shape that matches the destination window.

♦ *Fitted projection.* The symbol in Figure 2i shows that the image within the field of view is projected to a window that the arrow points to.

Composite objects. To simplify the specification, we defined composite objects, gave them their own symbol, and used them in subsequent specifications. For example, the object in Figure 3a specifies a standard coordination between an overview and detailed view of fixed sizes, as illustrated in Figure 3b.

In defining this composite object, we add the convention that unless otherwise specified all objects presented will be of fixed size. In Figure 3b, the left window is the source view of some image. As indicated by the movement-constraint operators, this field of view can move both horizontally and vertically. The image it encloses is projected on a second window, which has scroll bars. The horizontal and vertical scroll bars are linked to the field of view by the movement constraints. Thus, moving the field of view will not only change the image in the second window, it will change the scroll bar positions as well. And moving a scroll bar will change the position of the field of view and modify the projection dis-

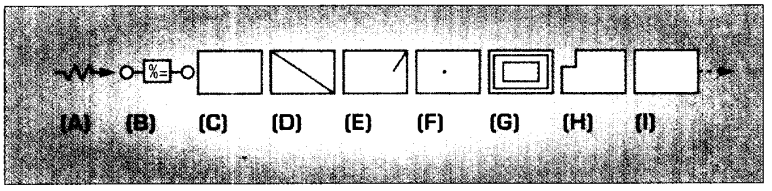


Figure 2. *DMSketch primitives.* ***(A)*** *Movement-constraint operator;* ***(B)*** *proportional-size constraint operator;* ***(C through H)*** *six variations on the field-of-view operator;* ***(I)*** *fitted projection.*

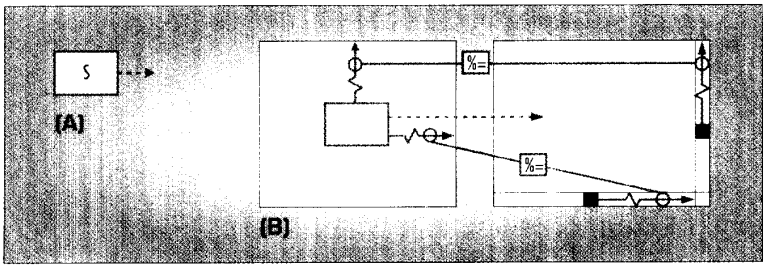


Figure 3. *(A) Composite object for our recommended standard coordination between two fixed-size windows;* ***(B)*** *browser specification.*

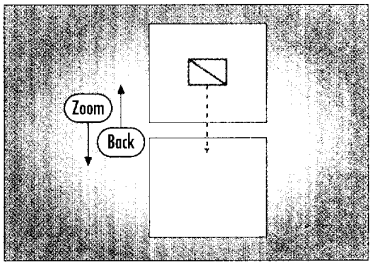


Figure 4. *Zoom command specification.*

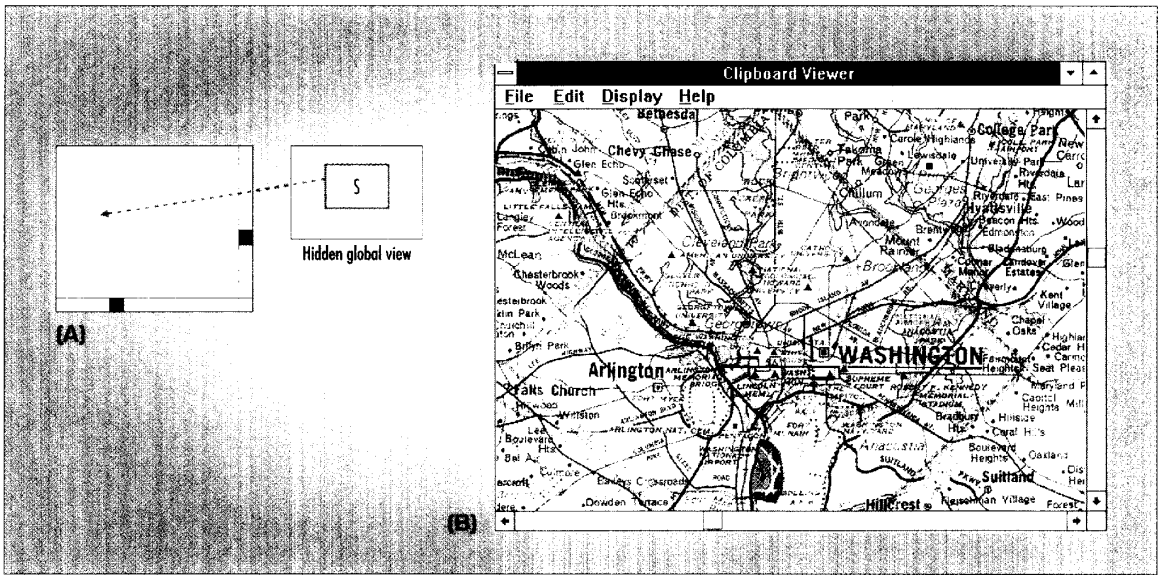


Figure 5. *Single-view browser.*

played in the second window.

This is our recommended standard coordination for fixed-window browsers and its symbol (the "S" in Figure 3a means "standard"). It is used frequently in specifying browsers. (Note that if the windows were resizeable, the shape of the field of view would have to be coupled to the shape of the detailed view.)

Commands. Currently, DMsketch provides only a rudimentary way of describing commands. As Figure 4 illustrates, the "before" and "after" state of the interface are shown with the command name associated with a directional arrow.

MULTITUDE OF BROWSERS

Our review of existing systems reveals great diversity in the design of 2D browsers. Here we review some classic techniques and their variations.

Detail-only browser. This method, which is used in X Windows, Microsoft Windows, and the Macintosh user interface, is the most common. The user is presented with a single window that can be panned both horizontally and vertically over the detailed view of the image. Figure 5a shows this in our notation; Figure 5b shows a common example.

This technique is easy to implement but it is satisfactory only when the zooming factor is relatively small or if it is unnecessary to see the global view. For example, if the zooming factor is two, you can see a quarter of the image at once, so there is not much navigation required to see everything. However, if the zooming factor is much larger, navigation is difficult. Imagine looking at a map that details all of Europe at street level. With this technique, it could take you some time to realize that the view you are seeing is Brussels, when what you really wanted was to find your way around Paris.

Single window with zoom and replace. This technique, common to many CAD/CAM and geographic information systems, presents a global view of the entire image. The user marks a rectangular area which is magnified and replaces the original image. Again, this technique is easy to implement. Also, because it handles navigation separately, it uses the screen space efficiently because users work on the detailed view with all the screen space. However, the context switch can be disorienting.

Figure 6 illustrates three variations of this technique. Figure 6a shows the simplest; its major drawback is that users must return to the global view every time they want to adjust the zoomed view. The variation in Figure 6b solves this problem by letting the user scroll the detailed view, and the variation in Figure 6c adds additional levels of magnification.

Of course the first two methods can be combined (global view, zoom, replace, scroll, as in Aldus PageMaker; or scroll, with option to zoom out to global view or in to more detail, as in MacPaint).

Single coordinated pair (overview-detail). Many 2D browsers are variations on our standard coordinated pair. These browsers combine displays of the overview and a local magnified view. The most common screen layout, shown in Figure 7a, reserves a small part of the screen for the global view, but others use windows of equal size, shown in Figure 7b, or reserve the large part of the screen for the global view, as shown in Figure 7c.

Tiled multilevel browser. These browsers combine global, intermediate, and detailed views, as the specification in Figure 8a shows. The global view is related to the intermediate view using our standard coordination, as is the intermediate to the detailed view. Figure 8b shows a sample application. In this technique, moving the global view or scrolling the intermediate view updates both views. Similarly, scrolling the intermediate view or the detailed view updates both.

Free zoom and multiple overlap. This is

a common design for applications running on fast platforms with large screens. Figure 9 shows the specification. Users are free (but required) to specify, move, reshape and delete every window as they wish. Any side-by-side comparison is possible.

The overview of the entire image is always presented first. The user must mark an area in the current view (top frame) and the boundaries for a new window (bottom frame). The system then creates the window and projects the marked area into the new window, which overlaps the source window. Both windows are linked to the undisplayed global view (not shown). Because there is no coordination between the views, the user has two independent browsers at different magnifications.

This design is flexible, but users must spend a significant amount of time managing the display because windows constantly obscure one another.

Bifocal view browser. A variant of the classic overview-detail browser is the bifocal browser,[6] specified in Figure 10. This browser uses a magnifying glass metaphor: It places a zoomed image on top of the area in which the

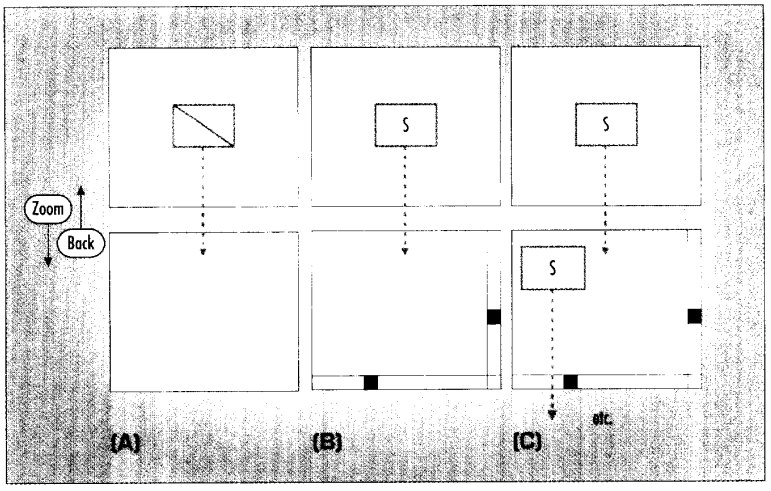


Figure 6. Three variations of zoom-and-replace. (A) Zoom only; (B) zoom then scroll; (C) zoom with additional levels of magnification.

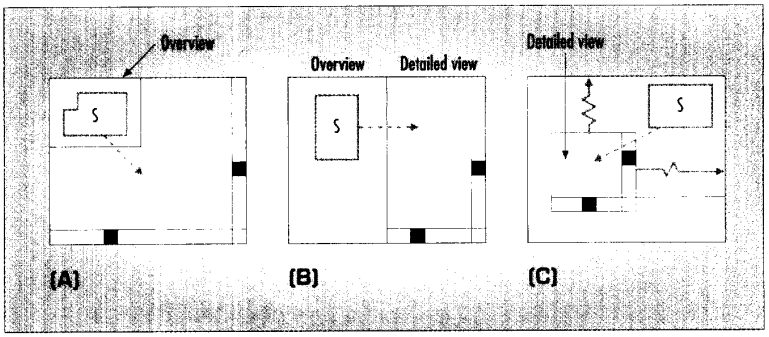


Figure 7. Single coordinated pairs. (A) Fixed small overview; (B) Overview and detail of equal size; (C) Moveable small detailed view.

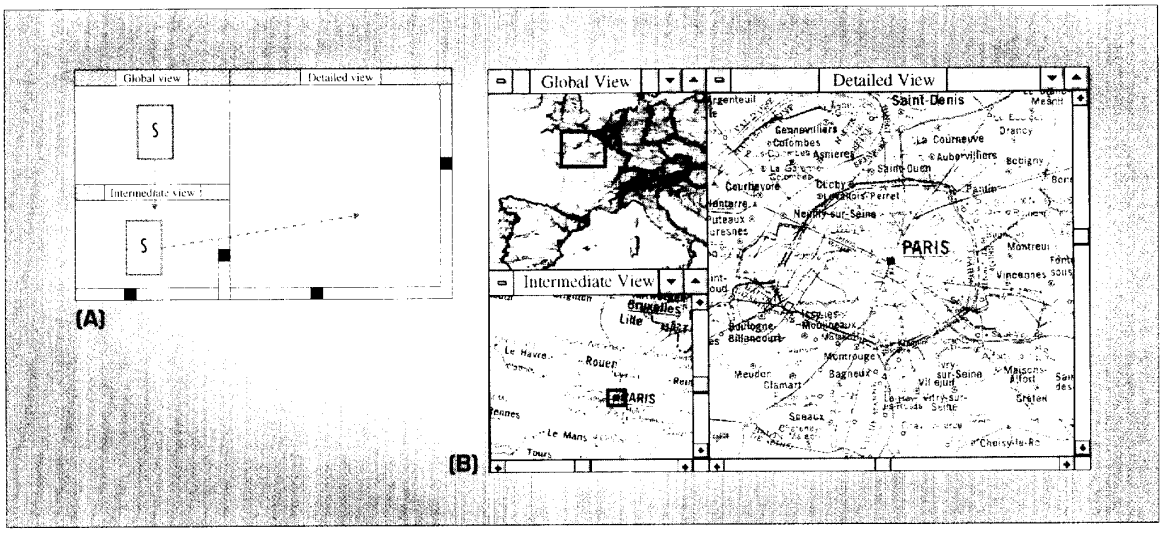
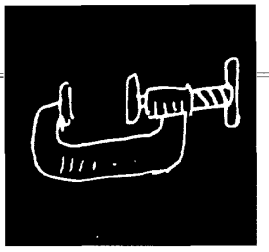


Figure 8. (A) Specification of three-level browser with global, intermediate, and detailed views; (B) sample application.



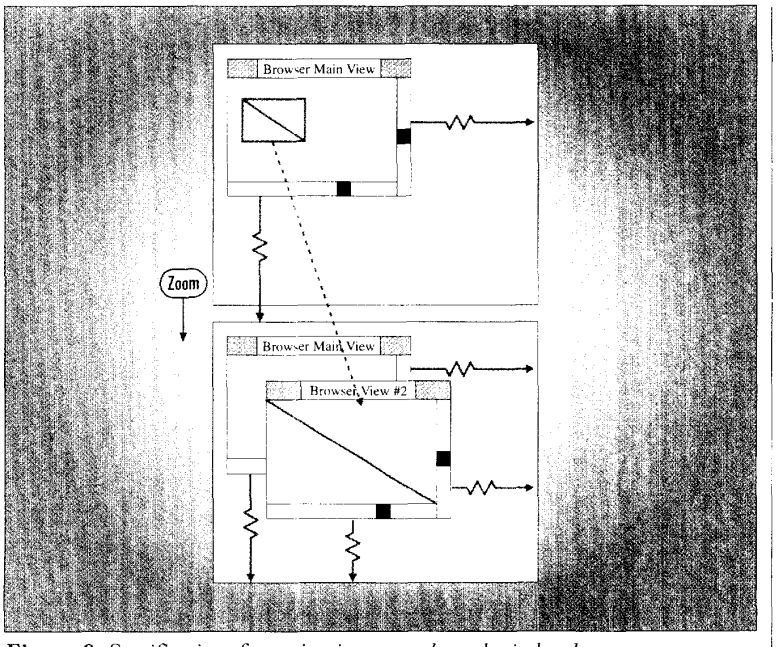


Figure 9. Specification of zooming in an overlapped window browser.

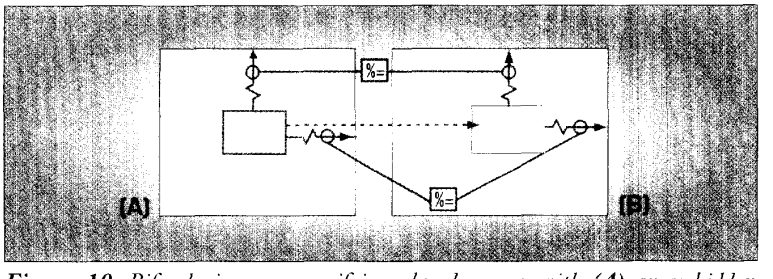


Figure 10. Bifocal view or magnifying-glass browser, with (A) areas hidden under detail view and (B) areas visible to users.

magnified object is located, thereby covering the neighboring objects.

Fish-eye view. An interesting extension of the bifocal view is the fish-eye view,[7] illustrated in Figures 11a and 11b. This browser distorts the magnified image so that the center of interest is displayed at high magnification, and the rest of the image is progressively compressed. In this way, it uses a single view to show a distorted global view, so no zooming or scrolling is required, but users must specify the focus of the magnification. However, distortion can be severe, especially with large images.

For example, Figure 11 is a small, hierarchically clustered telephone network with four levels of substations. The map in Figure 11b is a fish-eye view of San Francisco (the size of each color-coded area is made proportional to the 1980 white male population).

TASK TAXONOMY

We have identified five classes of tasks users accomplish with image browsers. Applications must often provide for different types of tasks, but usually one task is either performed repetitively or gets first priority because of safety requirements.

Image generation. When users draw or paint a large image or diagram, their attention is on a small part of the image but they often need to step back to look at the entire image. With a painting program a painter might concentrate on the drawing of a face, then return to view the entire scene. With a CAD/CAM program a boat designer might spend an hour drawing the bow of a boat then check the overall shape of the hull. Here units and sizes are often important. When a large document is automatically digitized by a scanner, progress is shown on a view of the whole document, but the refining work will be done in the few areas of the image that need retouching.

For image generation, an overview is important, but most of the time is spent at a detail level. Users tend to be experts.

Open-ended exploration. A tourist explores a remote city by navigating a map and accessing information on the local attractions. An adventure game player moves quickly around an imaginary space to become familiar with it. In both scenarios, the space is unknown to the user, so it's easy to get lost. The overview of the space being explored is not always complete or even available because it is explored for the first time.

In these applications, navigation must be fast and the user interface quickly mastered.

Diagnostic. An example of this special case of exploration is a pathologist who explores a digitized sample of tissue at low or high resolution, or a VLSI circuit specialist exploring a

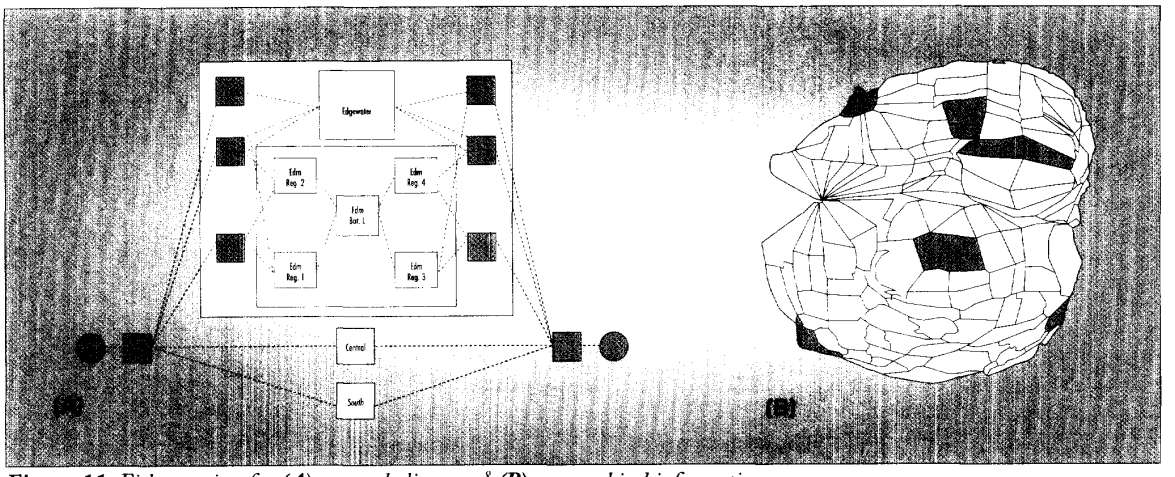


Figure 11. *Fish-eye view for* ***(A)*** *network diagram;[8]* ***(B)*** *geographical information.*

magnified view of a circuit. In these applications, panning speed and complete coverage is crucial because users spend most of the time panning the image and looking for patterns. The coverage must be complete or the wrong diagnosis can result. On the other hand, a complete automatic scan can also lead to boredom and errors, so the application must let the user save important locations for later review. Several browsers might be needed to compare cases.

Navigation. Here users more or less know the environment, but need to know how to get around. A delivery-truck driver uses a geographic information system to get directions. In this case, a global view must show the current position to provide context and point at the destination. Then the relevant information is presented at the minimum magnification level necessary to view the route. Zooming and panning occur only occasionally.

Monitoring. Here users must keep an eye on everything and always have information status on the entire system they are monitoring. Examples include the management of a large network, the central monitoring of the security or temperature of a large set of buildings, and the monitoring of a production plant. When a problem occurs, the user must be able to allocate some attention to local aspects while still watching the overview. Multiple views can be associated with a given problem that should be globally saved or retrieved, because the number of windows can become very large. Window management is an important issue: an overlapping window can hide important changes in another window.

BROWSER TAXONOMY

Figures 12 and 13 present a taxonomy of the points of comparison we have identified among image browsers. Figure 12 shows presentation aspects; Figure 13 operation aspects. We separated static and dynamic presentation aspects and manual and automated operations.

Static presentation. Under this category, we classified single- and multiple-view techniques, but hybrid browsers are common. For example, a global view can be provided along a second window that functions as a zoom-and-replace, the field of view of the global view always providing feedback about the size and location of the zoomed area. Another nice hybrid is the free overlapping of multiple chained pairs of coordinated views.

OUR BROWSER TAXONOMY SEPARATES STATIC AND DYNAMIC PRESENTATION ASPECTS.

Single-view browsers. These browsers dedicate all the screen space to a single view. They are very efficient when panning is limited and are the most commonly used browsers when display space is scarce. Appropriate when the task requires users to concentrate on the part of an image that fits on the screen; they are inappropriate when users must compare several distant parts of an image.

We identified three variations of the single-view browser:

- *Detail-only*: Does not support zooming, only panning. The default for most windowing systems if the image is larger than the window, but seems to work well only when the entire image is not much larger (magnification of four or less) than the view. These are common for image generation because most work is done at the detail level. They are not appropriate for monitoring.
- *Zoom-and-replace*: More appropriate as the difference in size between the entire image and the detailed view increases and navigation becomes more difficult. Some do not offer panning, which can be annoying because users must zoom out and zoom in to adjust the detailed view. Some zoom-and-replace browsers do not update the hidden overview as the detailed view is panned, which causes confusion when zooming out. This large family of browsers is appropriate for image generation and diagnostics if the display-update speed is sufficient.
- *Fish-eye*: Gives detail and context in a single view but severely distorts the image and requires constant reorientation. Distortion is a severe problem in applications in which size and geometry are important. These

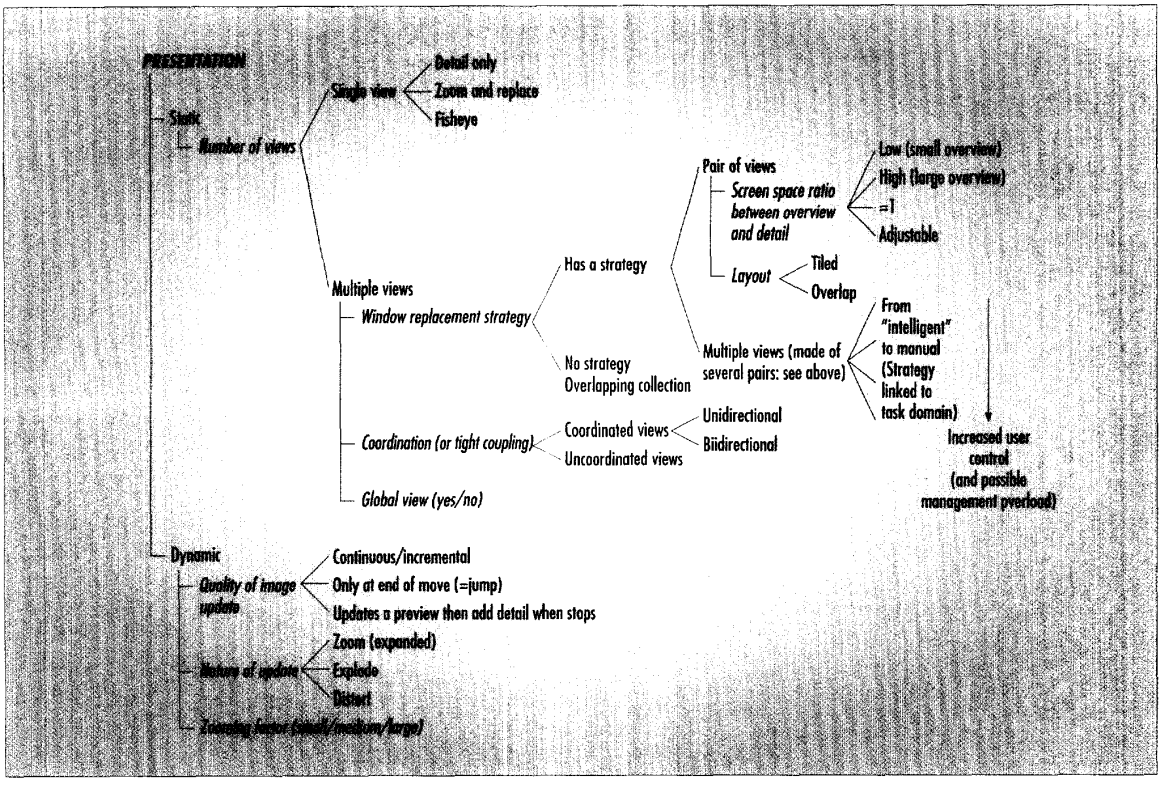


Figure 12. *Browser taxonomy for presentation aspects.*

browsers seem more appropriate for viewing abstract representations such as network diagrams, in which the view can be tailored for a user or a task but does not change constantly. Although transformations are complex and computationally demanding,[7] these browsers can be very effective (for hierarchically clustered networks,[8] for example). Designers should remember that fish-eye views can be inappropriate when fidelity to standard layout is important. The map in Figure 11b, for example, was rejected by epidemiologists because they could not compare it with the many other maps they are trained to memorize (such as maps of diseases).

Three techniques modify the fish-eye view: graphical distortion of the image, filtering to remove unwanted objects from the focus, and abstraction to replace blocks with symbols. Fish-eye views resemble domain-specific layout programs because they allow interactively generated custom layouts.

Multiple-view browsers. These browsers display several views. They are used when it is important to view details and context simultaneously, when fish-eye distortion is not appropriate, when parallel viewing is required for comparison, or when the display speed is insufficient to allow continuous zooming and panning.

We identified three important considerations when designing multiple-view browsers.

♦ *Window-placement strategy:* Too often, designers rely on window managers to handle the overlapping and resizing of windows. Although it is easier to implement a multiple-view browser without a window-placement strategy, we believe such browsers are more difficult to use. Research has shown that managing the overlapping windows can take considerable effort and time for the users.[9,10] We believe designers should provide an automatic window-management strategy that limits the need to move and resize windows incessantly. Many simple strategies (like the classic overview-detail pair) are available; researchers are investigating more complex strategies.[11] The more elaborate strategies are likely to be task-dependent, and designers would benefit from research into guidelines and tools for the specifying and customizing of window-management strategies.

For now, the standard overview-detail pair described in Figure 3 is easy to implement and addresses many users' needs. We recommend it for all tasks. The paired views should have the same shape (boot-shaped in Figure 7a; rectangular in Figure 7b).

We compare systems using this technique by the ratio of the screen space devoted to the overview and detailed views (the SSROD — screen space ratio: overview, detail.). This ratio should be a function of the task. For example, drawing or open-ended exploration requires a large detailed view, monitoring requires a large overview, navigation requires an overview and a detailed view of similar size, and an application that includes different tasks requires an adjustable ratio.

In addition to the SSROD, these systems can be compared according to view layout. Tiling windows frees the user from managing the views. Overlapping windows gives more flexibility but forces the user to do

more management. Some systems provide the specification in Figure 7a: overlapping windows that cannot be moved and that block access to part of the overlapped image (an early version of Publishers Paintbrush, for example). Of course, designers should avoid this.

♦ *Coordination:* The amount of coordination between views can be nonexistent (there is no overview), unidirectional (moving the overview updates the detailed view) or bidirectional (unidirectional, plus scrolling the detailed view updates the overview). We believe there are many opportunities for beneficial coordination. Indeed, our standard overview-detail pair in Figure 3a includes a bidirectional coordination: Moving the field of view in the overview updates the detailed view. Similarly, panning the detailed view should update the overview. This is an example of bidirectional tight coupling between two views.

♦ *Global view:* A global view shows the entire information space and allows quick access to any part. Just as a table of contents is required in print, a global view is required when browsing an image larger than one screen. This overview can be made simple and attractive for novice users or for public-access information systems. For experts, the global view should be as detailed as permitted by the display. Dense global views provide experts with direct access to details that would otherwise require several zooming operations (even if these global views appear unreadable to others!).

Dynamic aspects. Under this category, we classified the smoothness of the screen update when the image is panned or zoomed, the nature of the update, and the zooming factor.

♦ *Quality of the update:* A fast, smooth, and continuous image update makes navigation and exploration natural and simple, even over relatively long distances. It lets users concentrate on their tasks, not on the navigation tool. At one end of this spectrum are the "fly-over" interfaces that are possible only on fast hardware. At the other end are the slow, jumpy updates that can be disorienting, if not dizzying. Smooth scrolling plus rapid and continuous zooming are the secrets of success for single-view browsers.

♦ *Nature of the update:* An area that is zoomed can be simply expanded (similar to the way a camera zooms in) or "exploded" to reveal an internal structure not apparent in the overview (such as zooming in on a network node to reveal the internal structure of a node that was represented as a simple rectangle). Explosion is regularly used for hierarchical or hierarchically clustered data sets.[8] It simplifies the overview, but it can cause disorientation because the image is always changing. When using an explode zoom, designers should consider what subset of information appears on the overview. This is especially important for monitoring applications, in which alarms should be visible on the overview. In addition to expansion and explosion, the zoomed image can be distorted, as is the case in fish-eye browsers.

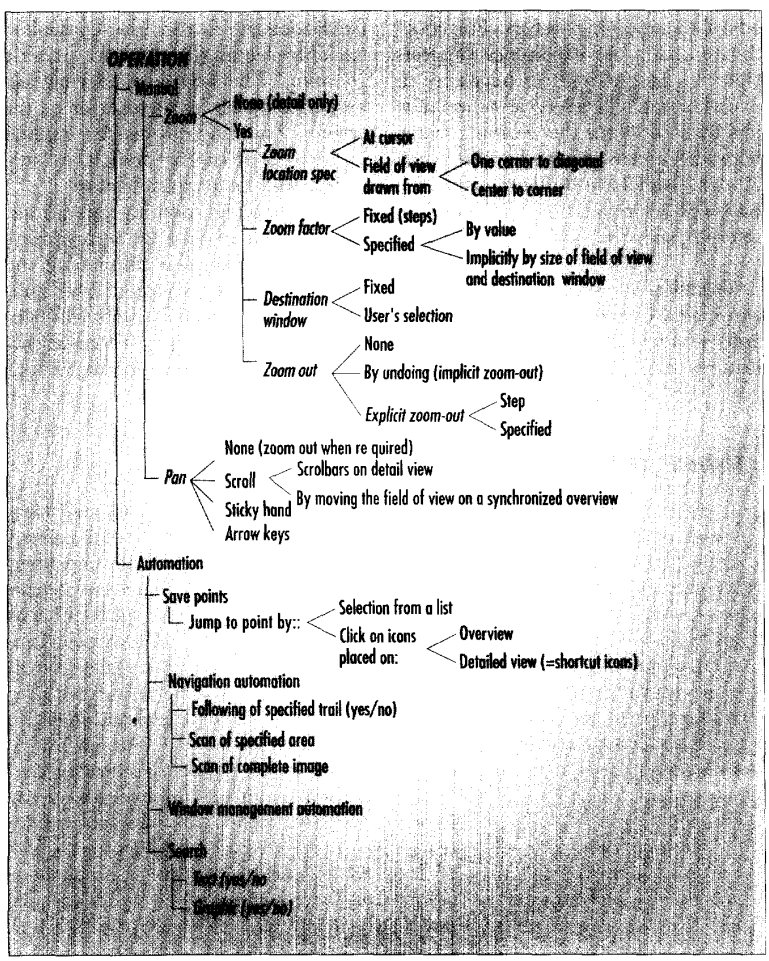
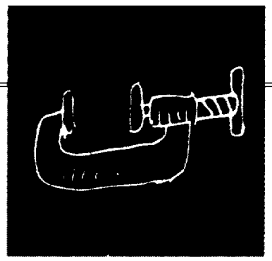


Figure 13. *Browser taxonomy for operation aspects.*



Again, designers can combine techniques. For example, when a user selects an object for zooming, the neighborhood around the object can be expanded and the object itself exploded to show its interior structure.

♦ *Zooming factor:* The zooming factor is the level of magnification between two views. Zooming factors can be fixed or specified. Fixed factors are set by the designer. This is a delicate task that requires designers to compromise between speed of access to details and the preservation of context information. No validated guidelines exist; designers must rely on usability testing with real users and tasks to adjust the zooming factors. For coordinated pairs, our experience suggests that the magnification between an overview and a detailed view should be less than 20. Once the zooming factor between screens gets to be more than 20 to 1, users have difficulty using the overview for navigation[12] and perhaps intermediate views are called for.

Operation. We separated manual and automated operations. Figure 13 lists and classifies all the techniques and features we found. Under the manual operations category, we classified zoom and pan techniques. Under the automated operations category, we classified saving, navigating, window-management, and searching techniques.

Manual operations. Browsers support two principal manual operations, zooming and panning. Panning and zoom can be readjusted simultaneously by redrawing the field of view or adjusting its size, placement, and even aspect ratio.

♦ *Zooming:* Users specify a zoom location by the cursor location or by drawing a field of view on the overview. Fixed-size rectangles specify a fixed zooming factor; user-controlled variable rectangles specify variable zooming. The new detailed view can be placed either by the user or by the system. Zooming out can be implicit by undo or it can be explicit, by step or by zoom-factor specification.

In specifying zooming operations, designers must find the appropriate compromise between complexity and flexibility. Browsers intended for public access or occasional users will benefit from simple designs (zooms at cursor location and fixed zooming factor), while expert users will demand more control over zooming. It is unrealistic to implement every possibility in a single system. Instead, designers should carefully study the tasks to be accomplished. For example, if size is important or if measurements are to be done on the image, specifying the zooming factor by its value (200 percent, for example) is more important than giving control of the field of view.

♦ *Panning:* We observed three panning implementations. Scrolling is the most common, usually accomplished with vertical and horizontal scroll bars. When an overview is present, scrolling can be accomplished by moving the field-of-view indicator in the overview. The second way to implement panning is to use a "sticky hand," which grabs the picture when the mouse button is pressed (first used in MacPaint). The picture then follows the cursor until the mouse button is released. The sticky-hand metaphor is appropriate only when a real-time image update is possible, however. The third panning method is the use of arrow cursor keys.

We think most systems should provide some general panning. Only in

TOWARD THREE-DIMENSIONAL BROWSERS

So far, three-dimensional spaces generated and browsed on computer screens tend to be either small spaces (in which there is a limited need for navigation), exploratory adventure games (in which being lost is a feature), or based on some pseudonatural navigating interface (going somewhere or flying through).

These applications are not so much for drawing or constructing, but for viewing what was created with other 2D tools. Many 3D applications are really 2D navigation (on the ground) or 2D with layers (in a building). There are more possible manual operations than just zoom and pan. More complex systems are like flight simulators, which use flight instruments and tools for navigation.

The basic browser in these cases is a zoom-and-replace browser (also resembling the fish-eye view with perspective). Three-dimensional browsing is based on the "natural" navigation skills of the users (approaching, turning around). Overviews are sometimes provided, either as classic 2D overviews (a user in virtual reality, for example, can grab a virtual street map), or in three dimensions.

For monitoring applications that require the overview to be complete and always visible, three-dimensional overviews must either be automatically rotated so that all faces are periodically exposed or flattened into multiple 2D slices.

In comparison, 2D image browsers provide much more functionality than the traditional "natural" browsing of 2D printed images because they can provide things like multiple windows, coordinated views, save points, and automations.

As 3D "spaces" become more widespread, we can expect the invention of better 3D navigation and exploration tools.

rare cases should designers have to disable detailed-view panning to avoid confusion between similar parts of an image. For example, it might make sense to restrict panning by moving the field of view when a user must closely examine a single vertebra of a spinal X-ray, because panning may lead to confusion as to which vertebra is currently on the screen.

Automated operations. It becomes difficult for users to concentrate on their task if there are too many potential browsing actions. We identified four categories of operations that designers should consider automating.

- *Save points*: Similar to setting bookmarks in text, marking points on an image can speed up image browsing. Locations of interest can be saved to allow rapid return to those saved points. Eventually, written or spoken comments can be saved with the location. This process of saving and retrieving points can greatly speed navigation and diagnostics (as when a second opinion is sought).
- *Navigation*: Direct-manipulation techniques can automate some navigation. An area can be marked on the overview to be systematically explored. A trail can be drawn on an overview and followed automatically, with the possibility to stop, explore locally, and resume the trail. Those techniques can be useful for diagnostics or even for simple exploration tasks. The areas already explored can also be shown on the overview to verify that all important parts of the image have been explored. Macro commands can be offered. For example, if users must analyze a series of similar images, they can mark the start point of a typical exploration sequence, which is then executed with a single command.
- *Window management*: When multiple views are used, designers can automate window placement. Fixed positioning of windows is, of course, an extreme example of such automation. The coordination of the field of view and the detailed view is another example. But more elaborate strategies can affect the sizing and placement of windows. For example, windows can become icons when unused for a certain time, or resized according to their estimated level of interest.[11] Synchro-nized windows can help users compare multiple images which they can pan simultaneously and then close simultaneously.[1]
- *Image search*: The automatic identification of image features is a growing field of interest based on the large body of work in computer vision on feature extraction and on similarity measures. A lot of work has recently been devoted to image retrieval, but similar techniques could be used to navigate within a large single image. For example, users may want to search a map for switching yards, a spinal X-ray for the location of each vertebra, and so on. Of course, a simpler search can be done on the text in the map. Such feature extraction might let designers adapt the browser to the task using content information. Multiple detailed views can be created automatically or panning speed can be adjusted according to the presence of features of interest (for example, the panning of a state map would be tailored to slow down when a switching yard is visible on the screen, or to jump from yard to yard).

DESIGNING AN IMAGE BROWSER INVOLVES MANY CHOICES, BUT WE HAVE ONLY VERY LIMITED GUIDELINES.

Many automations are possible. Research is needed to determine the benefits of such automations (or even in some cases to prototype and implement them). In general, automated operations are likely to be task-dependent and found only in specialized browsers.

Research is also needed into how to let users specify the automated operations they need. This topic borders on the more general topic of programming the user interface.

As this taxonomy suggests, designing an image browser involves many choices. Improved design based on controlled experiments could improve speed, error rates, and subjective satisfaction. But we have only limited guidelines, and few of those have been validated. We must prototype and test new automations. Techniques allowing users to specify the needed automation should be investigated. The multiple-view browsers will indirectly benefit from an increased attention to the design of window managers and of coordinated window-placement strategies.

The many options, features, and parameters we have described show the complexity of image-browser interfaces. The goal is to design the simplest tools that fit the task. In some cases, this might mean avoiding a browser entirely! Browsing is rarely trivial. Before evaluating the details of an image browser, designers should consider larger screens (or even multiple screens) and denser representations that do not require zooming and panning. Pixels on the screen are precious and effort should be made to display as much information on the screen as the task and user population will permit. Elegance and readability are important for public access, while speed of use should be the goal for expert users who need less zooming, less panning, fewer automated functions, and dense screens.

Image-browser design is a lively topic. If zooming and panning cannot be avoided, the tasks and the user population should drive the selection of the browser characteristics. Usability testing remains a requirement because of the still small number of validated guidelines. Beyond flat-screen browsing, novel features for three-dimensional browsers have yet to be invented. ◆

ACKNOWLEDGMENTS

We thank all the members of the Human-Computer Interaction Laboratory for their helpful suggestions. Partial support for this research was provided by Johnson Controls and OCLC and by NSF under grant NSF-D-CDR 8803012 and NSF-EEC 94-02384.

REFERENCES

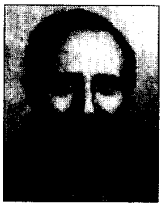
1. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 2nd ed., Addison-Wesley, Reading, Mass., 1992.
2. R. Chimera, "Value Bars: An Information Visualization and Navigation Tool for Multi-attribute Listings" (demonstration summary), *Proc. CHI 92: Human Factors in Computing Systems*, ACM Press, New York, 1992, pp. 293-294.
3. D. Beard and J. Walker II, "Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces," *Behaviour and Information Technology*, No. 6, 1990, pp. 451-466.
4. Y.K. Leung, "Human-Computer Interface Techniques for Map-Based Diagrams," in *Designing and Using Human-Computer Interfaces and Knowledge-Based Systems*, G. Salvendy and M. Smith, eds., Elsevier, Amsterdam, 1989.
5. S. Hudson and S. Mohamed, "Interactive Specification of Flexible User Interface Displays," *ACM Trans. Information Systems*, July 1990, pp. 269-288.
6. R. Spence and M. Apperley, "Database Navigation: An Office Environment for the Professional," *Behaviour and Information Technology*, No. 1, 1982, pp. 43-54.
7. M. Sarkar and M. Brown, "Graphical Fish-eye Views of Graphs," *Proc. CHI '92: Human Factors in Computing Systems*, ACM Press, New York, 1992, pp. 83-92.
8. D. Schaffer et al., Navigating Hierarchically Clustered Networks Through Fish-eye and Full-Zoom Methods," *ACM Trans. Information Systems*, 1995, to appear.
9. S. Bly and J. Rosenberg, "A Comparison of Tiled and Overlapped Windows," *Proc. CHI 92: Human Factors in Computing Systems*, ACM Press, New York, 1986, pp. 101-106.
10. S. Davies, K. Bury, and M. Darnell, "An Experimental Comparison of Windowed vs. Nonwindowed Operating System Environments," *Proc. Human Factors Society 29th Ann. Meeting*, Human Factors Society, Santa Monica, Calif., 1985, pp. 250-254.
11. D. Funke, J. Neal , and R. Paul, "An Approach to Intelligent Automated Window Management," *Int'l J. Man-Machine Studies*, No. 38, 1993, pp. 949-983.
12. C. Plaisant, D. Carr, and H. Hasegawa, "When An Intermediate View Matters: A 2D-Browser Experiment," Tech. Report CAR-TR-645, Center for Automation Research, University of Maryland, 1992.

Catherine Plaisant is assistant research scientist at the Human-Computer Interaction Laboratory of the Center for Automation Research at the University of Maryland. Her work centers primarily on graphical user interface design and evaluation. She has worked on projects involving hypertext, touchscreens, public-access information services, and information visualization for complex systems.

Plaisant received a general engineering degree from l'Ecole Nationale Supérieure des Arts et Métiers in Paris and a Docteur Ingénieur from the Université Pierre et Marie Curie in Paris.

David Carr is a PhD candidate at the University of Maryland and a senior software engineer at RMS Technologies at the NASA Goddard Space Flight Center, where he does research on specification of user-interface software. His research interests include software tools for user-interface design and human-computer interaction.

Carr received a BS in computer science from Michigan Technological University and an MS in computer and communications science from the University of Michigan.



Ben Shneiderman is a professor of computer science, head of the Human-Computer Interaction Laboratory, and a member of the Institute for Systems Research, all at the University of Maryland at College Park. He organizes an annual satellite television presentation, "User Interface Strategies," is the author of *Software Psychology: Human Factors in Computer and Information Systems* (Little Brown, 1980) and *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (Addison-Wesley, 1992), and the editor of *Sparks of Innovation in Human-Computer Interaction* (Ablex, 1993).

Shneiderman received a PhD in computer science from the University of New York at Stony Brook.

Address questions about this article to Plaisant, A.V. Williams Building, University of Maryland, College Park, MD 20742 (plaisant@cs.umd.edu), or see http://www.cs.umd.edu/projects/hcil/