# Navigation and Coordination Primitives for Multidimensional Visual Browsers[*]

*Allison Woodruff, Alan Su, Michael Stonebraker, Caroline Paxson, Jolly Chen, and Alexander Aiken*
*Department of Electrical Engineering and Computer Sciences*
*University of California at Berkeley*
*Berkeley, CA  94720  USA*
*email: tioga@postgres.berkeley.edu*

*Peter Wisnovsky and Cimarron Taylor*[†]
*Illustra Information Technologies, Inc.*
*1111 Broadway, Suite 2000*
*Oakland, CA  94607  USA*

## Abstract

This paper describes extensions to the Tioga flight-simulator browsing protocol presented by Stonebraker et al. (1993a). These extensions allow users to navigate a multidimensional data space using sophisticated zooming capabilities. This design also allows users to move easily between different multidimensional spaces. Tunneling between different data spaces is shown to be a substantial generalization of hyperlinks in a hypermedia system. Finally, our design provides for the coordination of multiple browsers. This preserves context and allows users to explore multiple paths simultaneously.

In concert, these extensions incorporate the functionality of many information management paradigms as well as introducing new constructs. These powerful mechanisms for relating data provide users with great flexibility. For example, users can create magnifying glasses which show an enhanced view of the underlying data.

## Keywords

Data browsing, hypermedia, magnifying glasses, user interfaces, visual databases, visual programming, wormholes, zooming.

---

[*] Supported in part by CNRI grant #M1717 and by NSF grant #FD94-00773.
[†] Presently with Automation Consultants Group, 650 California Street, 26th Floor, San Francisco, CA  94108.

## 1 INTRODUCTION

The design of user interfaces for database systems is an area in need of more attention (Stonebraker et al., 1993c). Existing database user interfaces are often unfriendly and difficult for nonexperts to use. Common database interfaces include textual programming languages or forms-based interfaces oriented towards business applications.

In Stonebraker et al. (1993a), we presented Tioga, a new paradigm for user interaction with a database management system (DBMS). Tioga is motivated by the needs of scientific DBMS users in the SEQUOIA 2000 project (Stonebraker and Dozier, 1992; Stonebraker et al. 1993b). Tioga uses the **boxes and arrows** notation popularized by scientific visualization systems such as AVS (Upson et al., 1989), Data Explorer (Lucas et al., 1992), and Khoros (Rasure and Young, 1992). Tioga improves upon these systems by providing sophisticated data management using the POSTGRES DBMS (Stonebraker and Kemnitz, 1991). In the Tioga programming model, boxes represent user-defined database queries or browsers, and edges between boxes represent flow of data. Although a limited number of boxes has currently been implemented, additional boxes may be programmed by users. Nonexperts build visual programs called **recipes** by interactively connecting boxes together using a graphical user interface. Current recipes include a photographic 35mm slide library and a geoindexing system. The underlying data manager is able to optimize and efficiently execute recipes.

The purpose of a recipe is to specify the data to be visualized, access the data through a database management system, and then locate the data in a multidimensional browser display. Figure 1 shows a typical recipe as constructed by a user. The recipe includes two browsers, viewer1 and viewer2, to display the data generated by the recipe. The default Tioga browsing paradigm allows users to visualize data results in a multidimensional space. Users navigate through their data using a flight-simulator interface. (Additional browsers may be implemented by advanced users.) Figure 2 shows a browser displaying objects in a latitude/longitude viewing space that contains California. One object, a digitized 35mm slide, has been selected and displayed.

In the default Tioga browser, the user chooses two dimensions to be displayed on the screen. Remaining dimensions appear as **sliders** which restrict the objects in the display to those which have values matching the constraints indicated by the sliders. The original navigational interface allows the user to pan over the two dimensions of the display or to zoom by enlarging a certain portion of the display. Clearly, more sophisticated navigation is desirable.

In Stonebraker et al. (1993a), we explored the basic constructs of Tioga and provided a query execution model. Chen et al. (1993) expanded the Tioga model to interface to foreign systems and provided a notion of

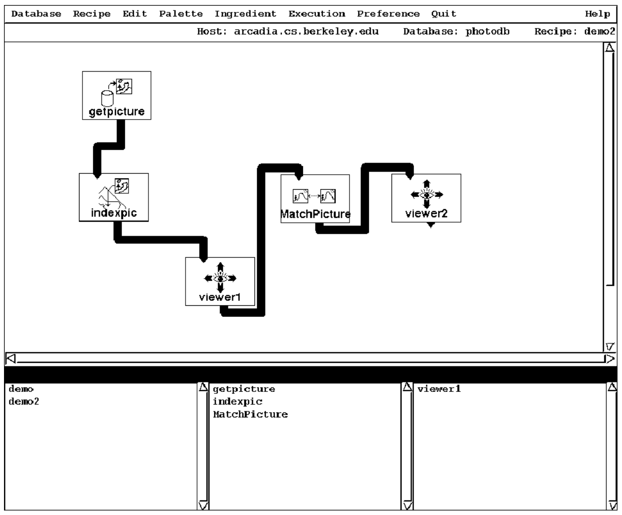
Database Recipe Edit Palette Ingredient Execution Preference Quit Help
Host: arcadia.cs.berkeley.edu Database: photodb Recipe: demo2
getpicture
indexpic
viewer1
MatchPicture
viewer2
demo
demo2
getpicture
indexpic
MatchPicture
viewer1

**Figure 1**
A Tioga Boxes and Arrows Diagram.

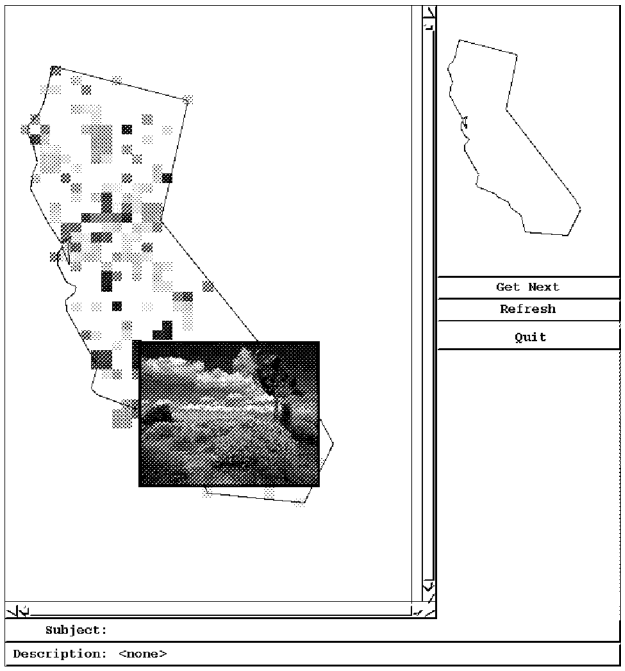

**Figure 2**
Data Displayed in the Tioga Browser.

transactions for the Tioga environment. Woodruff et al. (1994) introduced mechanisms to support navigation in multidimensional space.

In this companion paper, we extend these ideas and present three mechanisms which can be combined to support both novel, powerful constructs and constructs of proven worth. These extensions include:

- *enhanced detail.* Our system must be able to provide enhanced detail as a result of a zoom operation. For example, the Kodak PhotoCD representation for digital images supports five different resolution formats, ranging from a full size 2K by 3K by 8-bit color image to a 128 by 192 by 8-bit abstract (Eastman, 1992). A user would like the ability to see abstracts on the screen and then zoom in to view the images at a higher resolution. A similar feature was provided by SDMS (Herot, 1980), but it was hard-coded into that execution engine. Hence, retargeting SDMS required a considerable amount of customization.
- *movement to different multidimensional spaces.* Enhanced detail implies a change in perspective within a multidimensional space. Users also want the ability to switch to a new multidimensional space. For example, a user could zoom in on a map of Berkeley to find the Computer Science building. Additional levels of detail could yield documents corresponding to Computer Science technical reports. These documents should be displayed in a different context than the latitude/longitude coordinates appropriate for the map of Berkeley. When a document is being viewed, a further zoom could yield the image of the author or the layout geometry of his or her office. Again, a different multidimensional space should be used.
- *coordination of multiple browsers.* Our system must support multiple levels of detail in the same display. For example, it should be possible to place a magnifying glass on a portion of the display and have a zoom operation performed only for the objects under the glass. The remaining objects in the display should serve as a context for the magnified data and should not change. Because the objects in the magnifying glass are shown with enhanced detail, this function is considerably more complex than simply changing the number of pixels used for display. For example, support for magnifying glasses requires that browsers be allowed to share windows.

In the rest of this paper, we explore our design in detail. Specifically, in Section 2 we define a zoom capability that allows enhanced detail. We proceed in Section 3 to define wormholes that allow users to change multidimensional spaces. We turn in Sections 4 and 5 to our design for coordination of multiple browsers. We present the execution model in Section 6. In Section 7 we discuss potential applications, and in Section 8 we summarize our findings.

## 2 SHOWING ENHANCED DETAIL

To eliminate clutter in the display and to orient the user, data should have different representations when seen from different distances in multidimensional space. Intuitively, we wish to extend Tioga with the possibility of **zooming** into data to display more detail about screen objects. Our notion of zooming is semantic in that it involves changing the data objects being displayed, as opposed to simple graphical zooming. To support this functionality, we are extending the original Tioga browsing protocol.

In our design, data objects can have different abstracts that are produced by separate recipes. An **elevation map** relates these recipes to each other by specifying which recipes are valid at what distance (elevation) from the object. The elevation map is used to control the invocation of different recipes as the user zooms in and out through the data space. Therefore, if the user zooms into the elevation range of a different recipe, the recipe providing input to the browser is changed.

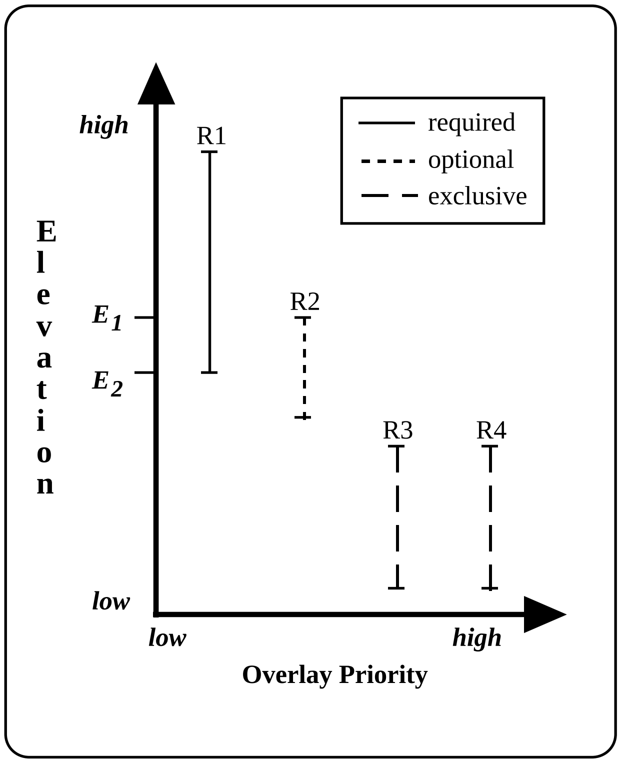

**Figure 3**
An Example Elevation Map.

Figure 3 shows an elevation map containing four recipes, R1 ("State outline"), R2 ("Census tracts"), R3 ("Rivers"), and R4 ("Highways"). R1 produces output for the browser at high elevations. When the user zooms to elevation $E_1$, R1 and R2 are both valid. Further inward at $E_2$, R1 stops output and only R2 may display data. Further zooming can display output from recipes R3 and R4.

To support this behavior, we begin by associating with any browser in any recipe an elevation range over which the browser displays data from this recipe. A browser is associated with a multidimensional coordinate system as noted above. In this presentation, we assume N dimensions which we denote $A_1$, ..., $A_N$. We add an N + 1st dimension, designated **elevation**, which is used to indicate the user's perspective. This does not represent a physical elevation, but is rather a logical representation of a user's viewing distance from the N-dimensional space.

The original Tioga implementation displays two user-selected dimensions, $A_X$ and $A_Y$, on the screen. In this browser, the user can change the range of these dimensions by resizing the window. The range is adjusted proportionally to the change in window size. Note that resizing the window has no effect on elevation, as Figure 4 illustrates. Assume that the user's initial position in a displayed dimension is $ELEV_I$ with viewing angle $\theta_I$, as shown in $TRIANGLE_I$. Adjusting the window size while remaining at a constant elevation is analogous to changing the user's viewing angle, as shown in $TRIANGLE_W$.

In our new design, the user is also allowed to adjust the elevation of a browser. When the user zooms to $ELEV_Z$, $\theta_I$ remains constant, resulting in $TRIANGLE_Z$. Because $TRIANGLE_I$ and $TRIANGLE_Z$ are similar, $RANGE_Z$ can be recalculated as follows:

$$RANGE_Z = RANGE_I * (ELEV_Z / ELEV_I)$$

Adjusting the window size or zooming may select the same range for display. However, the two operations may have very different results. Adjusting the window size does not change the recipe providing input to the browser. Conversely, zooming may place the user in the elevation range of a different recipe. In this case, the recipe providing input to the browser is changed as specified by the elevation map.

The data provided by each recipe must be supplied to the browser in a common, multidimensional coordinate system. When more than one recipe at a given elevation may provide output to the browser, the elevation map also specifies the overlay priority of the recipes, which is shown on the horizontal axis. When conflicts occur in allocation of display space, objects from recipes with higher priority are visible on the screen in preference to those from recipes with lower priority.

In addition to specifying the elevations at which recipes are to be invoked and their overlay priority, the elevation map also contains a collection of semantic restrictions on the display of recipe output. Each recipe in an elevation map can be:

- *required*. In this case, recipe output must be displayed if the browser is at an elevation within the elevation range.
- *optional*. In this case, when the user enters the elevation range of the recipe, he or she is prompted as to whether the output from this recipe should be visible. This behavior occurs if the user zooms into the elevation range from above or zooms out from below. At any given elevation range, a mechanism allows users to turn on or off optional recipes valid at the current elevation. Using this interface, a user can change his or her mind about seeing (or not seeing) the objects from optional recipes.

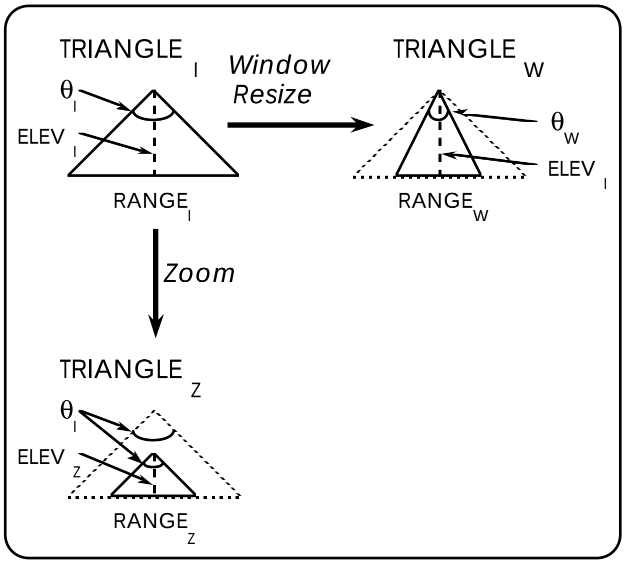

**Figure 4**
Changing Viewing Angle and Elevation.

- *exclusive*. A user can specify a radio button behavior for recipes that are valid at common elevations. With this behavior, at most one of the recipes can be activated, and the user is presented with a menu of radio buttons to indicate which one.

These semantics are illustrated in Figure 3. At higher elevations, R1 displays the outline of California. Upon zooming to $E_1$, the user has the option to see also the census tracts in the state as output by R2. At the next transition point, $E_2$, the outline of the state is no longer visible and the census tracts are optionally visible. Further zooming shows either the rivers or the highways of California (as generated by R3 and R4), but not both.

Previous work has identified the merits of zooming capabilities (Gorlick and Quilici, 1994; Perlin and Fox, 1993). We contribute a dynamic, visual mechanism for specifying the behavior of objects when viewed from different distances. Elevation maps allow a user to define easily the semantics of the zoom operation, assuming that all recipes produce data in the same multidimensional space. We now turn to a mechanism for changing from one multidimensional space to another.

## 3 CHANGING MULTIDIMENSIONAL SPACES

Enhanced detail implies a change in perspective within a multidimensional space. Users also want the ability to move to related multidimensional data spaces for new perspectives on the data, a functionality similar to that provided by hyperlinks (Conklin, 1987). Consider the following example in which Tioga presents information about the residents of Berkeley. Initially, the application displays a map of Berkeley. Zooming inward gives more detail about geographic objects, culminating with the outline of each individual residence. At this point, the user may invoke a new type of browser, defined for each residence, that displays an image of the people living there. Requesting detail on the residence therefore causes a different multidimensional space to be explored.

When a user changes to a new multidimensional space, objects have a spatial relationship that is unrelated to the relationship in effect on the near side. This behavior should be distinguished from a zoom operation where the same spatial relationship is present before and after the zoom. Therefore, we denote this operation of changing from one multidimensional space to another as **tunneling** through a **wormhole**, to differentiate it from zooming. A wormhole is a connector between two disparate data spaces, and tunneling is the process of travelling through a wormhole.

To construct a wormhole, the user must specify the following information:

- *the wormhole location*. We associate the wormhole with objects displayed by some recipe in some browser. Hence, the location is indicated by the three-tuple: (recipe name, browser name, query).
- *the new application that should be run on the other side of the wormhole.*
- *a* ***tag*** *associated with the wormhole*. Because there may be multiple wormholes for a given object in a given recipe, we require the tag field to allow a user to specify which wormhole should be followed.

In practice, a user-interface gesture indicates the user's desire to tunnel through a wormhole. The user then chooses a wormhole from the list of tags associated with the object(s) selected. At this point, a new application is invoked taking the object(s) selected as a parameter(s). In the example given above, the object identifier of the house would be passed to the new application to allow it to display only the people living there. It is also possible to define the wormhole over a collection of object identifiers. Specifically, we allow an arbitrary function to identify the objects for which the wormhole is defined. One such function could be:

```
retrieve (House.oid)
where House.architect = "Wright"
```

in which case the wormhole would be defined only for houses designed by Wright.

## 4 SLAVING AND CLONING BROWSERS

It is difficult for users zooming and tunneling in single browsers to explore multiple paths simultaneously. Additionally, when users are zooming and tunneling in multidimensional space, it is difficult for them to maintain a sense of context. A related problem may be observed in traditional systems such as graphics packages, help systems, and hypermedia systems. In many of these systems, users must repeatedly zoom in and out or backtrack. Somewhat more sophisticated systems provide limited browser coordination in overview-detail browser pairs (Plaisant et al., 1995). In the next two sections, we introduce mechanisms for browser coordination which support simultaneous exploration of multiple paths through multidimensional space and display context to the user.

Browsers in the same recipe can be independent of each other. In this case, movement in one browser does not affect the others. This behavior is appropriate when the browsers are displaying independent objects.

On the other hand, one browser can be constrained to another in a master/slave relationship. In this case, whenever the user changes the master's position in N-dimensional space, the slave's position in M-dimensional space automatically changes as well. More specifically, during the recipe definitions, the user defines a function, `LOC_CALC`, that translates requests in the master into requests to the slave:

```
LOC_CALC(N-space-region) → M-space-region
```

When the master is moved to $\text{REGION}_1$, the slave will be instructed by Tioga to move to `LOC_CALC(REGION1)`. Both browsers must recalculate their visible rectangles and issue commands to retrieve the objects which will be displayed. Further, when the master changes elevation, the slave must automatically change elevation at the same time. Usually, slaved browsers will be constrained to have the same elevation as their master; however, we allow the user to specify optionally a second function, `ELEV_CALC`:

```
ELEV_CALC (elevation) → elevation
```

In this way, the slaved browser can be constrained to operate at a second elevation that is a function, `ELEV_CALC`, of the elevation of the master.

There are four interesting ways in which slaved browsers may be constrained:

- *Slaved browsers may display data from different regions in a multidimensional space.* For example, a user may need to examine all areas 5 miles north of a pipeline to ensure that a toxin carried by the pipeline has not affected these areas. In Tioga, the pipeline could be viewed in one browser. A second browser could be slaved with a function `LOC_CALC` that mapped an N-space-point to a new point 5 miles north.

- *Slaved browsers may display the same data seen from different elevations.* For example, the slave could show a detailed representation of what is seen in the display of the master. In effect, the slave would show the data of the first browser through a magnifying glass.
- *Slaved browsers may display different data from the same points in a multidimensional space.* For example, one browser could show color-coded precipitation data for an area while its slave displayed temperature data for the same area.
- *Slaved browsers may display related data in different multidimensional viewing spaces.* For example, a master browser could show a map of California while a slaved browser displayed a list of all seismographic sensor stations located in the area shown in the master browser. As the user looked at different parts of California, the slaved browser would automatically adjust the list of stations.

The four examples above assume the existence of two browsers in the same recipe. If only one browser exists, it may be more convenient to **clone** it rather than to place explicitly a new browser in the recipe graph. Cloned browsers can be slaved or independent. The examples above demonstrate possible uses for slaved clones. For instance, to display the same data at different levels of detail, the user could clone and slave a viewer to act as the magnifying glass.

Independent clones are appropriate in other situations. For example, it is easy for a user to become disoriented when navigating in a multidimensional space. Hence, the user would like to mark the position of something of interest and return to it at a later time. Cloning a browser allows one browser to remain stationary at the object of interest while the second one continues to browse the multidimensional space.

To support this functionality, we propose the following design. The original browser becomes the **originator** and the new browser becomes the **clone**. The originator and the clone have exactly the same elevation map. During the cloning operation, the user must specify if the clone is to be slaved to the originator, and if so must specify `LOC_CALC` and `ELEV_CALC`.

If the originator and the clone are independent (i.e. not slaved), then the second browser is initially assigned the same elevation and spatial location in multidimensional space as the first browser. The joystick of either browser can be moved arbitrarily, and the displays of the two browsers will typically diverge. As a result, cloning subsumes bookmarking features found in other systems.

If the clone is slaved to the originator, then the clone is constrained to operate at an elevation and location offset relative to the originator; these offsets are determined by `LOC_CALC` and `ELEV_CALC`. In this case, the slaved browser operates at an elevation of

```
ELEV_CALC (elevation of originator)
```

and at a location given by

```
LOC_CALC (N-space-region of originator)
```

If `LOC_CALC` is the identity function, and `ELEV_CALC` specifies a zoom, then the clone will provide an automatic magnifying glass, without requiring the user to perform the zoom manually.

Using slaving and cloning, a variety of offset displays and magnifying glass effects can be constructed. The next section describes mechanisms that allow two browsers to share the same screen area, thereby permitting a true magnifying glass effect to be implemented.

## 5 SHARING WINDOWS

We would like to allow the user to place one browser inside another. For example, the user could place a **magnifying glass** browser inside another browser displaying a map of California. This is analogous to reading a map of California with the aid of a physical magnifying glass. Adding this functionality to Tioga allows users to combine the usefulness of effects such as Magic Lenses (Bier et al., 1993) with the strength of a DBMS. To provide this functionality, we require that browsers be allowed to share their windows. If two browsers share the same multidimensional space, then it is permissible for them to share a browser window. Note that since a clone and its originator automatically share the same multidimensional space, they are able by definition to share a window.

When two browsers are declared to share a window, they are referred to as the **outer** browser and the **inner** browser. Each browser has separate slider bars that determine the content of its display. The outer browser uses the complete window to display its objects. The inner browser magnifies a portion of the data shown in the outer browser. The display of the inner browser is then overlaid on the display of the outer one, thereby creating a single composite display which can be rendered on the screen by the window manager.

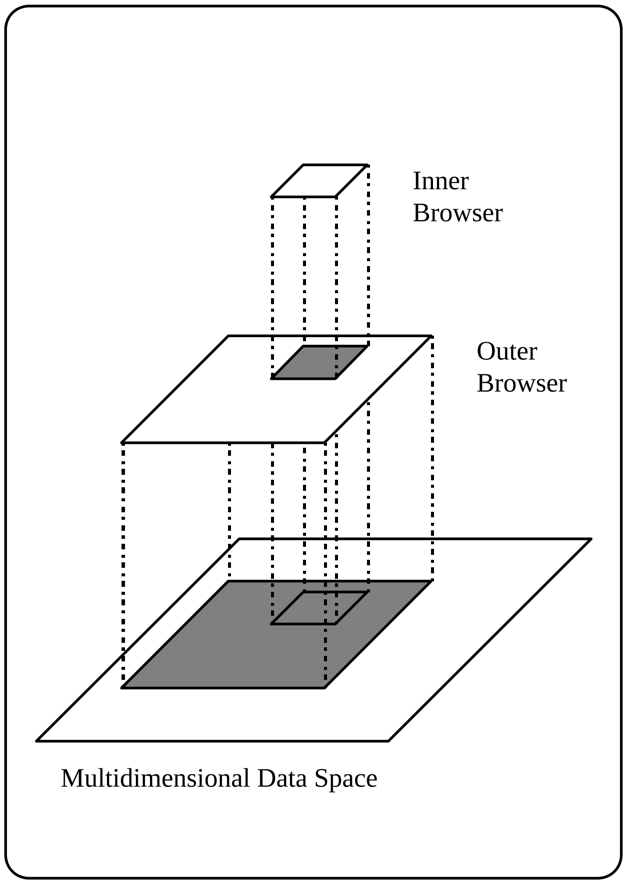

**Figure 5**
Viewing Regions of Nested Browsers.

To create this composite, Tioga finds the center of the inner browser's current viewing region, and then locates this point in the outer browser's window. Next, Tioga positions the inner browser's viewing region at this location within the outer browser. There are three cases of interest:

- *case 1*: the inner viewing region is completely inside the outer viewing region. In this case, the two regions are overlaid and displayed.
- *case 2:* the inner viewing region is completely disjoint from the outer viewing region. In this case, the inner viewing region cannot be seen, and only the outer viewing region is visible.
- *case 3:* the inner viewing region overlaps the outer viewing region. In this case, Tioga must clip the inner viewing region and then overlay the two displays as above.

These behaviors can be illustrated in the following examples. The user can clone a browser and run both the originator and the clone in a common window. A zoom operation on the inner browser will allow the user to observe a magnifying glass effect, whereby one browser is providing a detailed blowup of the region displayed by the other. Figure 5 shows a magnifying glass (the inner browser) positioned above a window (the outer browser) which is in turn positioned above a map (the multidimensional space).

Consider the case in which the inner and outer browsers are independent. In this situation, the position and elevation of each browser can be changed independently. Moving the inner browser will change its position above the map, and therefore, its contents. Because it moves independently, moving it will also change its position relative to the outer browser. This will allow the user to magnify various areas of the map. If the user moves the magnifying glass completely out of the region displayed by the outer browser, then case 2 above applies and the detail will not be shown. If the magnifying glass is partly outside this region, then case 3 above applies and only a portion of the inner browser will be displayed. Similarly, if the user moves the outer browser, the content of the inner browser will not automatically change, although its position relative to the outer browser will change.

To achieve different behavior, we can slave the clone to the originator. Since the inner browser is slaved to the outer browser, moving the outer browser will change the area displayed in both browsers. However, the position of the magnifying glass within the outer browser will remain constant. Moreover, if the user zooms the outer browser to blow up the map of interest, say to a map of a specific county, then the magnifying glass will also zoom.

The definitions presented above may be recursively extended so that an inner browser may in turn serve as an outer browser. This allows an arbitrary number of browsers to share a window, with a pairwise inner-outer relationship between them. Thus, a hierarchical collection of browsers can be defined. For example, a user may choose to have a magnifying glass on top of another magnifying glass, providing even more detail.

## 6 EXECUTION MODEL

The original Tioga executor considered the execution of single recipes. However, zooming requires that multiple recipes execute concurrently to supply data to a browser. We have extended our design to include an abstract manager which controls the execution of recipes and coordinates their communication with browsers.

In a naive implementation of zooming, all browsers in a recipe would be displayed when the user zoomed to the elevation at which the recipe became valid. However, different recipes in an elevation map may contain different types and numbers of browsers. Therefore, the naive implementation would result in arbitrary browsers appearing and disappearing as the user adjusted elevation. To provide a more intuitive semantics, we introduce the notion of a **base** recipe which defines the visual environment which the user can explore. The visual environment is made up of one or more browsers which appear in the base recipe. Each of these browsers is associated with an elevation map that specifies which recipes may provide data to it.

Execution begins when a base recipe is passed to the abstract manager. The abstract manager examines the elevation maps of all browsers in the base recipe to determine the set of all recipes which may be accessed through the visual environment. The abstract manager then starts an executor for each of these recipes, as well as a process for each browser in the base recipe. The abstract manager establishes communication channels between executors and browsers. These are distinct from the dataflow edges which exist in the recipe graphs. At any given time, a browser has a communication channel to every recipe in its elevation map which is currently active (i.e. valid at the current elevation). The browser uses these channels to request data from the executors. When the user adjusts elevation, the abstract manager updates the communication channels between the browsers and executors, adding and deleting connections as appropriate.

Since abstracts allow movement in a single multidimensional space, the appearance of new browsers might be disorienting. However, since tunneling by definition involves viewing a new multidimensional space, we allow the recipe invoked by a wormhole to serve as a base recipe and start a new visual environment. The invocation is similar to a fork operation; both the original browser and the new browser continue to exist as independent processes, each with a base recipe.

## 7 APPLICATIONS

In the preceding sections, we have defined semantics for zooming, tunneling, and the coordination of multiple browsers. The resulting model unites constructs which have proved useful in many types of information management systems.

For example, the reader can readily observe that wormholes are a substantial generalization of hyperlinks in a hypermedia system. A traditional hyperlink is a wormhole in which there is a prespecified object or collection of objects on each side of the hole. Wormholes allow a run-time specified set of objects to be on each side. Further, classical hypermedia systems only provide a two-dimensional display space composed of the X and Y screen dimensions. The Tioga paradigm extends such systems by supporting the display of objects in user-specified dimensions.

Similarly, abstracts generalize the notion of layers as used in many Geographic Information Systems (GISs). In many such systems, data is stored in separate layers representing categories such as species distribution, land use, or political boundaries. Layers may be viewed independently or displayed together in one window. Abstracts provide a mechanism for the application developer to define the levels at which each layer will be visible. Further, while some GISs support browsing in 2 or 3 dimensions representing the physical dimensions of the Earth, they do not incorporate browsing in arbitrary multidimensional spaces. Such browsers could be used to provide additional data. For example, selecting a city might offer the option to tunnel to a browser containing people objects displayed according to the dimensions age, income, and consumption of a product.

Scientific visualization systems could also take advantage of the constructs supported in the Tioga model. Many existing scientific visualization systems provide users the option of browsing through data in multidimensional space. However, many scientists would like to perform data lineage queries in which they trace the history of the operations that have been performed on their data. Slaved browsers could be used to simultaneously display different versions of data. In this case, navigation in one browser would cause the slaved browser to automatically display the corresponding data.

The model presented in this paper incorporates multiple concepts of proven worth. This model can be used both to enhance existing systems and to create new information management applications.

## 8 CONCLUSIONS

In this paper we have detailed mechanisms that support navigation in multidimensional space. We first outlined a zoom capability that allows users to view data at different levels of detail. We next introduced the concept of wormholes. Tunneling through a wormhole allows users to view their data in the context of a new multidimensional viewing space. We then proposed mechanisms for linking browsers together as well as for creating new browsers. We detailed the behavior of multiple browsers sharing a portion of the screen. The resulting data management and browsing model is a generalization of many existing systems. In combination, the ideas presented not only add functionality to existing paradigms but support new constructs such as magnifying glasses as well. We are currently extending the original Tioga system to support the features discussed in this paper.

## 9 REFERENCES

Bier, E., Stone, M., Pier, K., Buxton, W., and DeRose, T. (1993) Toolglass and Magic Lenses: The See-Through Interface. Proceedings of SIGGRAPH 93, Anaheim, California.

Chen, J., et al. (1993) Extending a Graphical Query Language to Support Updates, Foreign Systems, and Transactions. SEQUOIA 2000 Technical Report 93/38, University of California at Berkeley.

Conklin, J. (1987) Hypertext: An Introduction and Survey. *Computer*, 20:9, pp. 17-41.

Eastman Kodak Company (1992) Programmer's Guide for UNIX Systems. *Kodak PhotoCD Access Developer Toolkit.*

Gorlick, M. and Quilici, A. (1994) Visual Programming-in-the-Large versus Visual Programming-in-the-Small. Proceedings of the 1994 IEEE Symposium on Visual Languages, St. Louis, Missouri.

Herot, C. (1980) Spatial Management of Data. *ACM Transactions on Database Systems,* 5:4, pp. 493-513.

Lucas, B. et al. (1992) An Architecture for a Scientific Visualization System. Proceedings of the 1992 IEEE Visualization Conference, Boston, Massachusetts.

Perlin, K. and Fox, D. (1993) Pad: An Alternative Approach to the Computer Interface. Proceedings of SIGGRAPH 93, Anaheim, California.

Plaisant, C., Carr, D., and Shneiderman, S. (1995) Image-Browser Taxonomy and Guidelines for Designers. *IEEE Software*, 12:2, pp. 21-32.

Rasure, J. and Young, M. (1992) An Open Environment for Image Processing Software Development. Proceedings of the 1992 SPIE Symposium on Electronic Image Processing.

Stonebraker, M. and Kemnitz, G. (1991) The POSTGRES Next-Generation Database Management System. *Communications of the ACM*, 4:10, pp. 78-92.

Stonebraker, M. and Dozier, J. (1992) SEQUOIA 2000: Large Capacity Object Servers to Support Global Change Research. SEQUOIA 2000 Technical Report 91/1, University of California at Berkeley.

Stonebraker, M., Chen, J., Nathan, N., Paxson, C., and Wu, J. (1993a) Tioga: Providing Data Management for Scientific Visualization Applications. Proceedings of the 1993 VLDB Conference, Dublin, Ireland.

Stonebraker, M., et al. (1993b) The SEQUOIA 2000 Architecture and Implementation Strategy, SEQUOIA 2000 Technical Report 93/23, University of California at Berkeley.

Stonebraker, M., et al. (1993c) DBMS Research at a Crossroads: The Vienna Update. Proceedings of the 1993 VLDB Conference, Dublin, Ireland.

Upson, C., et al. (1989) The Application Visualization System. *IEEE Computer Graphics and Applications*, 9:4, pp. 32-40.

Woodruff, A., Wisnovsky, P., Taylor, C., Stonebraker, M., Paxson, C., Chen, J., and Aiken, A. (1994) Zooming and Tunneling in Tioga: Supporting Navigation in Multidimensional Space. Proceedings of the IEEE Symposium on Visual Languages, St. Louis, Missouri.

## 10 BIOGRAPHIES

**Allison Woodruff** is a Ph.D. student in Computer Science at the University of California at Berkeley. She received the Bachelors degree in English from California State University, Chico in 1987. She received the Masters degree in Linguistics in 1989 and the Masters degree in Computer Science in 1993 from the University of California, Davis. She has worked as a student assistant and as a geographic information systems analyst at the California Department of Water Resources. Her research interests include user interfaces for databases and implementation issues for visual languages**.**

**Alan Su** is a Bachelors degree candidate majoring in Electrical Engineering and Computer Science at the University of California at Berkeley. He plans to enroll in the Computer Science Ph.D. program at the University of California at San Diego in the Fall of 1996.

**Michael Stonebraker** is a professor of Electrical Engineering and Computer Sciences at the University of California at Berkeley, where he has been employed since 1971. He was one of the principal architects of the INGRES relational database management system which was developed during the period 1973-77. Subsequently, he constructed Distributed INGRES, one of the first working distributed database systems. Then, he turned his attention to developing a next generation DBMS, POSTGRES, that can effectively manage not only data but also objects and rules as well. POSTGRES formed one of the cornerstones of the SEQUOIA 2000 project, the DEC flagship research project of the 1990s for which Dr. Stonebraker has served as co-project director. SEQUOIA 2000 is striving to build a new computer environment for Earth science researchers and encompasses networking, storage, DBMS, and visualization activities. Currently, he is focusing on the DBMS support for visualization environments and on next-generation distributed DBMSs.

Dr. Stonebraker is a founder of INGRES Corp (now the INGRES Products Division of ASK Computer Systems), the founder of Illustra Information Technologies, Inc., a past chairman of the ACM Special Interest Group on Management of Data, and the author of many papers on DBMS technology. He lectures widely, has been the keynote speaker at several recent conferences, and was the winner of the first ACM SIGMOD innovations award in 1992.

**Caroline Paxson** is a Ph.D. student in Art History at the University of California at Berkeley. She received the Bachelors degree in German Literature from Harvard. She received the Masters degree in Computer Science from the University of California at Berkeley in 1993.

**Jolly Chen** is a doctoral candidate in Computer Science at the University of California at Berkeley. He received the Bachelors degree in Computer Science and Masters degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1990. His research interests include query optimization and user interfaces for databases. Jolly Chen is a student member of ACM.

**Alexander Aiken** received his Bachelors degree in Computer Science and Music from Bowling Green State University in 1983 and his Ph.D. from Cornell University in 1988. He was a Research Staff Member at the IBM Almaden Research Center before joining the Berkeley faculty in 1993.

**Peter Wisnovsky** has worked in the database industry for the last six years, for the Sharebase Corporation (formerly Britton-Lee) and Illustra Information Technologies, the leader in object-relational database technology. At Illustra he has worked on the Object Knowledge database visualization system and its visual programming language development environment, the Project Editor. He is currently working on interface technology at Illustra. He received the Bachelors degree in Computer Science from Princeton in 1989.

**Cimarron Taylor** is Chief Scientist at Automation Consultants Group. While at the University of California at Berkeley from 1987 to 1990, he contributed to the query executor, access methods and transaction system of POSTGRES. After graduating, he went to Scopus Technology where he turned Scopus's workflow prototype into a commercial product. He joined Illustra when it was founded in 1992 to build client software tools. He developed and implemented the compilation algorithms in the Object Knowledge Toolkit to translate visual programs into Illustra's dialect of SQL3. He is now responsible for the information architecture of ACG's manufacturing enterprise automation system.