



# Snap-together visualization: can users construct and operate coordinated visualizations?

CHRIS NORTH<sup>†</sup> AND BEN SHNEIDERMAN

*Human-Computer Interaction Lab & Department of Computer Science,  
University of Maryland, College Park, MD 20742, USA.*

*email: north@cs.umd.edu, ben@cs.umd.edu*

*URL: www.cs.umd.edu/hcil/snap/*

*(Received 2 February 2000, and accepted in revised form 31 May 2000)*

Multiple coordinated visualizations enable users to rapidly explore complex information. However, users often need unforeseen combinations of coordinated visualizations. Snap-together visualization (Snap) enables users to rapidly and dynamically construct coordinated-visualization interfaces, customized for their data, without programming. Users load data into desired visualizations, then construct coordinations between them for brushing and linking, overview and detail view, drill down, etc. Snap formalizes a conceptual model of visualization coordination based on the relational data model. Visualization developers can easily Snap-enable their independent visualizations using a simple API.

Empirical evaluation reveals benefits, cognitive issues and usability concerns with coordination concepts and Snap. Two user studies explore coordination construction and operation. Data-savvy users successfully, enthusiastically and rapidly constructed powerful coordinated-visualization interfaces of their own. Operating an overview-and-detail coordination reliably improved user performance by 30–80% over detail-only and uncoordinated interfaces for most tasks.

© 2000 Academic Press

## 1. Introduction

In the field of information visualization, researchers and developers have created many types of visualizations or visual depictions of information (Card, Mackinlay & Shneiderman, 1999). For example, to display hierarchical information, one can choose from outliners, hyperbolic trees (Lamping & Rao, 1996), treemaps (Shneiderman, 1992), fish-eye views (Furnas, 1986), etc. Each visualization has different strengths. For example, hyperbolic trees may be appropriate for deep unbalanced hierarchies, whereas treemaps are helpful when nodes have numerical attributes.

User-interface designers often coordinate multiple visualizations, taking advantage of the strengths of each, to create even more powerful information exploration environments (Shneiderman, 1998; Baldonado, Woodruff & Kuchinsky, 2000). This technique is particularly potent when the information is sufficiently complex to require different types of visualizations for different aspects or layers. A simple example interface is Microsoft's

<sup>†</sup> Now affiliated with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, USA

Windows Explorer, which employs three visualizations to browse hierarchical file systems: an outliner visualization of the folders, a tabular visualization of the files in the selected folder and a textual visualization of the details of the selected file including a miniature quick-view.

We define a *coordinated-visualization* user interface as a set of visualizations and a set of coordinations between them. Some common types of coordinations (North & Shneiderman, 1997) used in information-exploration interfaces are as follows.

- *Brushing and linking*: an exploratory data analysis (EDA) technique used when displaying a set of data items in multiple visualizations. When users select items in one visualization, those items are automatically highlighted in all the visualizations. A common example is brushing scatterplots (Becker & Cleveland, 1987).
- *Overview and detail view*: selecting an item in the overview navigates the detail view to the corresponding details. Items are represented visually smaller in the overview. This provides context and allows direct access to details. For example, web designers often add a table-of-contents frame to a large document. Users can then select a section title to scroll the main frame immediately to that section.
- *Drill down*: allows users to navigate down successive layers of a hierarchical database. Selecting a parent item in one visualization loads children items into another visualization, as in Windows Explorer. This enables exploring very large-scale data, by displaying aggregates in one visualization and the contents of a selected aggregate in another visualization (Fredrikson, North, Plaisant & Shneiderman, 1999).
- *Synchronized scrolling*: users can conveniently scroll through multiple corresponding data sets. Examples are alternate translations, music and information with summaries or annotations.

In the literature, the phrase “multiple views” is sometimes used to refer to only the brushing-and-linking coordination. Hence, we use “coordinated visualizations” to refer to the more general definition above.

While many coordinated-visualization interfaces have been implemented, two confounding problems arise. Firstly, the set of visualizations *and* coordinations needed in any given situation depends greatly on the following.

- *Data*: different data sets have different features and structure.
- *Tasks*: what does the user want to accomplish with the data?
- *Users*: there is tremendous variation between users in individual user preferences, experience levels, etc.

For example, while Windows Explorer is helpful for some users and tasks, system administrators may need different visualizations. Replacing the outliner visualization of folders with a scatterplot of the folders would enable administrators to quickly spot large old folders for archival (see Figure 1).

Secondly, the implemented visualization tools are typically not programmed to coordinate together. Hence, these alternate combinations usually require custom development. Researchers in our lab stumble over this problem often, and must constantly re-implement coordinations between new unforeseen combinations of visualizations. Unfortunately, this is a poor solution to the problem. Even with good component-based design, these hard-coded combinations are inflexible and difficult to construct.

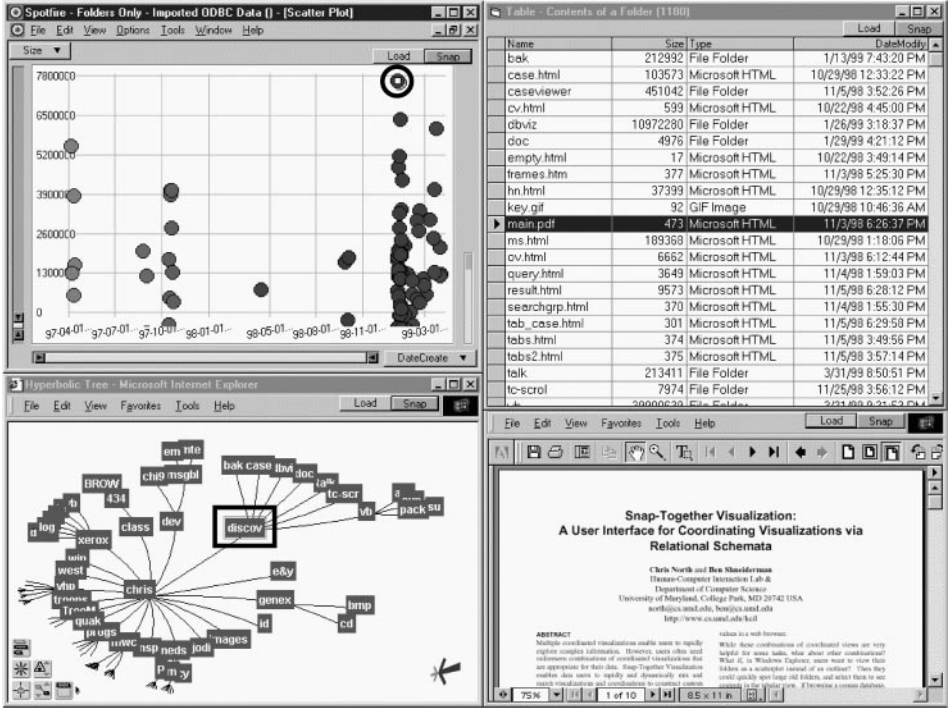


FIGURE 1. A coordinated-visualization interface, quickly constructed with Snap, for system administrators browsing file-folder structures. The scatterplot and hyperbolic tree display the folders, enabling users to examine size and date trends as well as hierarchical structure. Selecting a folder displays details of its files in the tabular visualization. Selecting a file displays it in the quickview.

Clearly, the number of needed combinations of visualizations and coordinations explodes exponentially, and implementation becomes intractable. Hence, the control of the choice of coordinated-visualization interface needs to be in the hands of the users. A lightweight mechanism is needed to allow end-users to easily “snap” individual visualizations together into custom combinations. This must not be a toolkit that requires programming, but a user interface.

Snap-together visualization (Snap) is a conceptual model, user interface, software architecture and implemented system developed to meet these needs. Snap enables data users to rapidly and dynamically mix and match visualizations and coordinations to construct custom exploration interfaces without programming. Snap is flexible in data, visualizations and coordinations. Snap focuses on (1) interconnecting the visualization tools created by researchers and developers in the field to (2) construct coordinated-visualization interfaces for rapid exploration and navigation of data and relationships.

Snap also provides a platform on which to study the use of coordinated visualizations in general. Do users understand coordination between visualizations? Can they construct their own coordinated-visualization user interfaces to support their tasks? Can they use coordination to their benefit? If there is a benefit, why and what are the key aspects of the coordinated visualizations that causes improvement?

In general, user-interface design requires significant expertise, but Snap puts some design capability in the hands of users. Can users essentially design their own user interfaces for information exploration by snapping together appropriate components?

This paper presents the Snap user interface and basic conceptual model and then reports on two studies on constructing and operating coordinated visualizations.

## 2. Related work

Coordinated-visualization systems for information visualization can be classified by their level of flexibility in data, visualizations and coordinations.

- (1) *Data*: users can load their own different data sets into the visualizations.
- (2) *Visualizations*: users can choose different sets of visualizations as appropriate for the data.
- (3) *Coordinations*: users can choose different types of coordinations between pairs of visualizations as needed for exploring or navigating relationships in the data.

Level 0 systems are not intended for flexibility. For example, Windows Explorer always displays the same data set (the hard-drive file structure), with the same visualizations and coordinations.

Most systems are level 1, flexible for data but not visualizations or coordinations. Users can load their own data, but are always presented with the same interface.

Level 2 systems include flexibility in choice of visualizations. EDA systems, such as Datadesk (Velleman, 1988), SAS JMP, EDV (Eick & Wills, 1995), and Spotfire (Ahlberg & Wistrand, 1995), display a data table in many different types of visualizations of users' choosing such as scatter plots or bar charts. All the visualizations are coordinated for brushing-and-linking, allowing users to relate data points across visualizations. In databases, Visage (Roth *et al.*, 1996) extends the brushing coordination to multiple tables by brushing across relational joins. However, users cannot establish a different type of coordination between two visualizations with these systems.

Level 3 systems include flexibility in the coordinations between visualizations. Some of these provide only one type of coordination but let users choose which visualizations to coordinate. The Apple Dylan programming environment (Duman & Parsons, 1995) lets users browse hierarchical object-oriented programs by splitting and linking frames so that selecting a folder in one frame displays its contents in the other frame (e.g. generalized Windows Explorer). Spreadsheet visualization (Chi, Barry, Riedl & Konstan, 1997) arranges many small 3D visualizations as cells in a 2D grid. Then, users can select a whole row or column of visualizations to synchronize their 3D navigation. DEVise (Livny *et al.*, 1997) allows users to select some different types of coordinations between visualizations. Users can synchronize panning and zooming of plots with common axes, and establish set operations between visualizations so that data in one visualization can be combined with data in another.

In scientific visualization, data-flow systems such as ConMan (Haeberli, 1988), AVS and IBM Data Explorer, and filter-flow systems such as Linkwinds (Jakobson, Berkin & Orton, 1994) also employ a form of dynamic linking, but for a different purpose. Users link a variety of modules to create custom data processing and viewing pipelines, much like pipes on the Unix command line. Snap coordinations transmit interaction rather

than data, and coordinations are bi-directional like constraints rather than pipes. Constraint-based tools such as ThingLab (Borning, 1986) are intended for specification of more complex interaction within a view.

Snap builds on these systems, borrowing Visage's information-centric approach making individual information objects the basis of coordination rather than 2D information-space axes as in DEVisé. Snap uses a drag-and-drop action similar to Apple Dylan to select visualizations to coordinate. However, Snap's coordination model, specification interface and ultimate purpose are unique. Snap allows users to construct a variety of common coordinations quickly and easily.

## 2.1. EVALUATION

Little work has been done to study and evaluate the use of coordinated visualizations. Several empirical studies have compared specific coordinated-visualization interfaces to other approaches such as fish-eye and detail-only visualizations for browsing hierarchies (Chimera & Shneiderman, 1994; Shneiderman, Shafer, Simon & Weldon, 1986) and large 2D spaces (Beard & Walker, 1990). In general, these studies indicate an advantage of coordinated visualizations over single detail-only visualizations. However, the studies did not determine why or what aspect of the coordinated visualizations caused improved performance. Was it the additional information displayed in the multiple visualizations or the interactive coordination between them?

Even less is known about users ability to construct such coordinated-visualization environments. Usability work on Apple Dylan (Dumas & Parson, 1995) indicates that once users were shown how to split and link its frames, they were able to remember it. While users were successful with Dylan's single type of data, visualization and coordination, will that carry over to a general coordinated-visualization environment? Can users grasp the notion of establishing different types of coordinations between different types of visualizations? Can users construct appropriate interfaces for themselves this way? Clearly, a deeper level of understanding about users and coordination is needed.

## 3. Snap-together visualization

### 3.1. USER INTERFACE

Snap is based on the relational data model. To explore a database, users first load and display relations (tables or query results) in visualizations. Then they coordinate the visualizations by specifying actions to tightly couple between the visualizations.

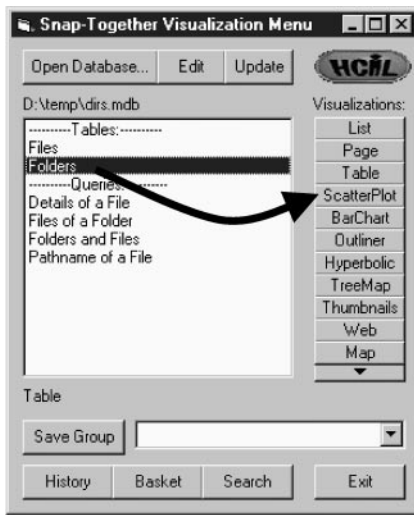
*3.1.1. Scenario: file-folders.* We first demonstrate step-by-step how Snap is used to construct the file-folder browser for system administrators as described in the example in the introduction (Figure 1). First, a database containing the folder and file information is opened with Snap. The Snap Menu window [Figure 2(a)] displays the list of data tables and queries in the database (left), as well as a list of available visualization tools (right).

To view the folders in a scatterplot, the data table containing folder information is dragged from the list and dropped onto the scatterplot button. A scatterplot window opens, loads and displays the folders as data points (using Spotfire (Ahlberg & Wistrand,

1995), a commercial scatterplot package) as on the left of Figure 2(d). Choosing folder’s “creation date” attribute for the X-axis and “size” attribute for the Y-axis establishes the desired view. Now it is easy to spot any large old folders in the upper left of the scatterplot.

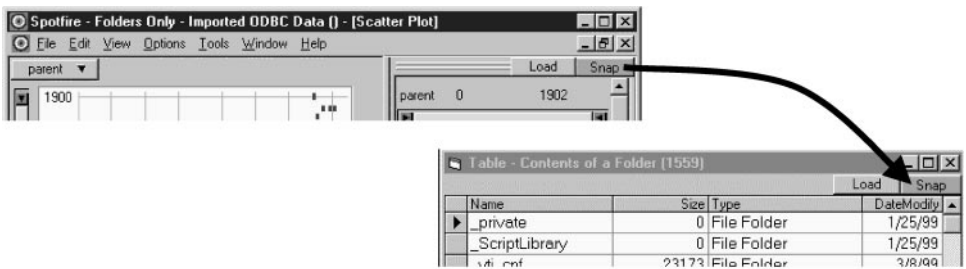
Of course, users need to see the files contained in the folders. In the Snap Menu window again, a query that extracts information about the files within a given folder is dragged from the list and dropped onto the tabular visualization button. This opens a tabular visualization window as on the right of Figure 2(d).

Now, the two visualizations can be coordinated for behavior similar to Windows Explorer. Snap automatically adorns each visualization window with a Snap button



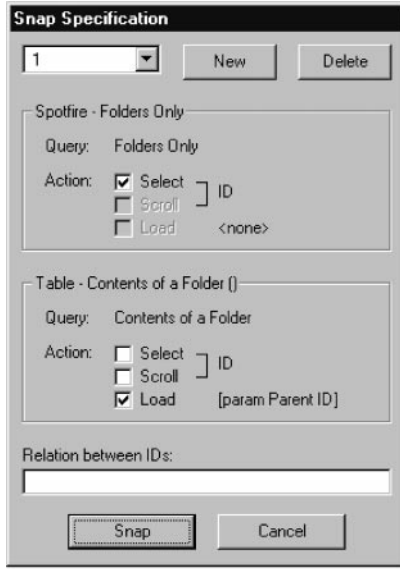
(a)

FIGURE 2(a). Snap’s Menu window displays the list of tables and queries in the database (left), and the available visualization tools (right). Dragging the folders table onto the scatterplot button opens a scatterplot visualization of the folders [as in Figure 2(d), left].



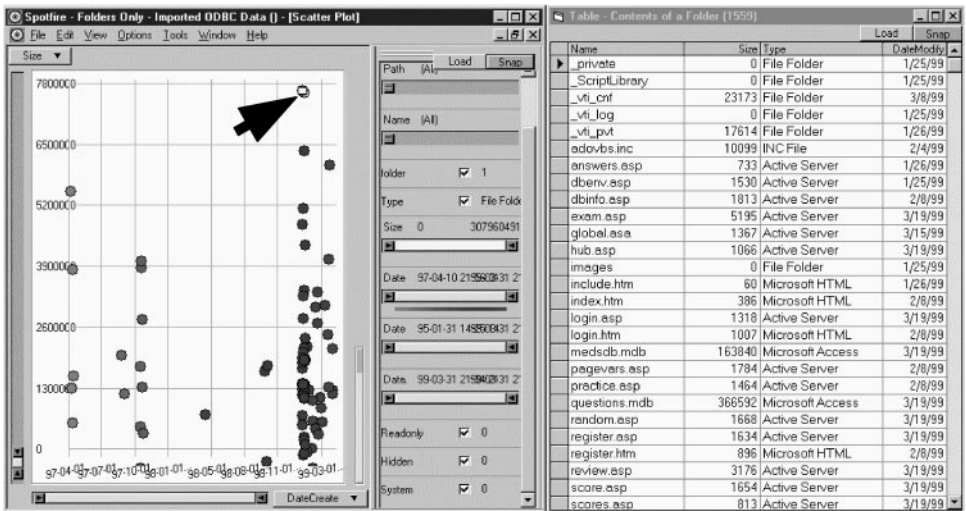
(b)

FIGURE 2(b) Each visualization window is adorned with a Snap button. Dragging the Snap button from the scatterplot window to the tabular window constructs a coordination between the visualizations and displays the Snap specification dialog.




(c)

FIGURE 2(c) With the Snap specification dialog, users choose the actions to tightly couple between the visualizations. Here, selecting a folder in the spotfire scatterplot should load the contents of the folder into the table visualization.



(d)

FIGURE 2(d) Construction of the coordinated-visualization interface is complete. Now, users can browse by simply selecting folders in the scatterplot to view their files in the tabular visualization, like Windows Explorer.

. To coordinate the visualizations, the Snap button is dragged from the scatterplot to the tabular visualization [Figure 2(b)]. The Snap Specification dialog [Figure 2(c)] then displays the available actions in each visualization that can be tightly coupled. Choosing the “select” action for the scatterplot and the “load” action for the tabular visualization specifies that selecting a folder in the plot should load and display the files in that folder into the tabular visualization.

Now, construction of the coordinated-visualization interface is complete [Figure 2(d)]. Users can browse by simply selecting folders in the plot to view their contents in the tabular visualization.

Additional visualizations could be added to further improve the interface (Figure 1). For example, if the context of the folders in the hierarchical structure is important, then users might load the folders into PARC’s Hyperbolic Tree (Figure 1, lower left). They could snap this to the scatterplot so that selecting a folder in either visualization would also select and highlight it in the other. To examine the contents of many files, users could snap a file viewer (Figure 1, lower right) onto the tabular visualization.

*3.1.2. Scenario: census data.* Snap is in use at the Census Bureau to construct visualization environments for analysts and to prototype interface variations for CD-ROM products. Census analysts have found the capability to relate data between maps and plots extremely helpful. As an example, Figure 3 is an interface constructed with Snap for exploring Census population data of US states (left) and counties (right). Users can explore from nominal, geographic and numeric perspectives using the textual lists, maps and scatterplots. Selecting Maryland reveals that it ranks very high in terms of income per capita and percent college graduates. Apparently, Maryland has two counties that are outliers with very high percentage of college graduates. Selecting the outlier county with the highest per capita income reveals that it borders Washington, DC.

### 3.2. MODEL OF VISUALIZATION COORDINATION

Snap’s conceptual model of visualization coordination is based on the relational data model. To explore a database, users can construct interfaces composed of coordinated visualizations based on the data schema. Snap establishes a direct correspondence between relational data concepts and user-interface concepts.

<i>Relational data model</i>	<i>User Interface</i>
Relation	→ Visualization
Tuple	→ Item in a visualization
Primary key	→ Item ID
Join	→ Coordination

In Snap, a visualization displays a relation (a table or query result) from the database. Coordination between two visualizations is based on the join relationship between their relations (see Figure 4). This is somewhat similar to RMM (Isakowitz, Stohr & Balasubramanian, 1995), which generates hyperlinks based on database relationships.

*3.2.1. Relations into visualizations.* When using Snap, users first load relations into visualizations (Figure 5). Generally, each tuple in the relation is depicted as an individual



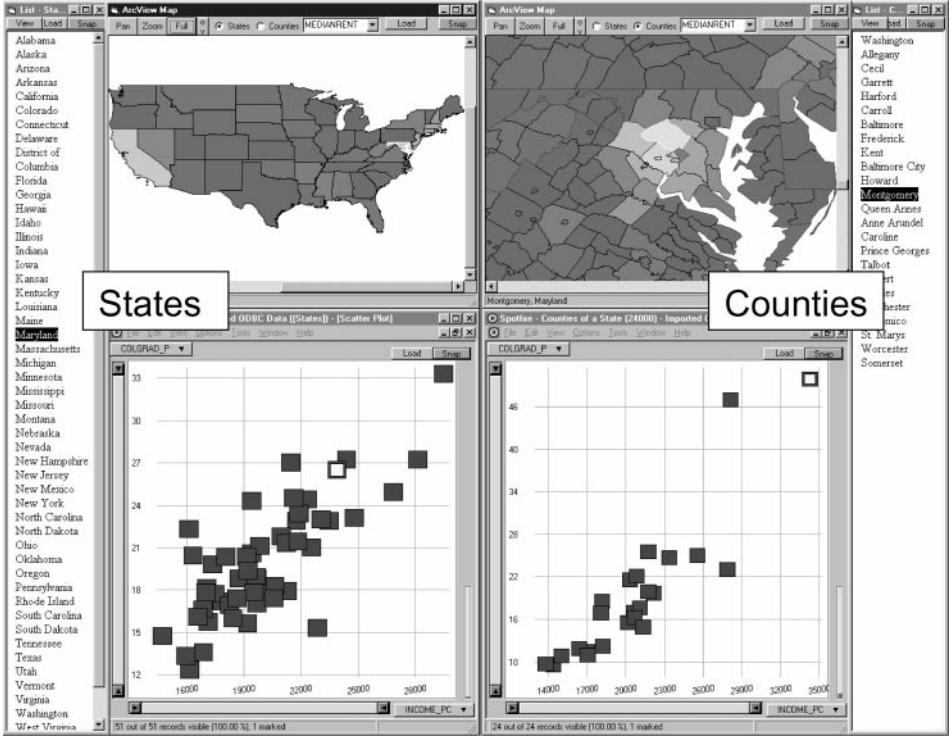


FIGURE 3. A coordinated-visualization interface for exploring Census data of US states and counties, dynamically constructed using Snap. Users can drill down from states (on the left) to their counties (on the right). Brushing-and-linking between the nominal, geographic and numeric perspectives enables users to relate the visualizations.

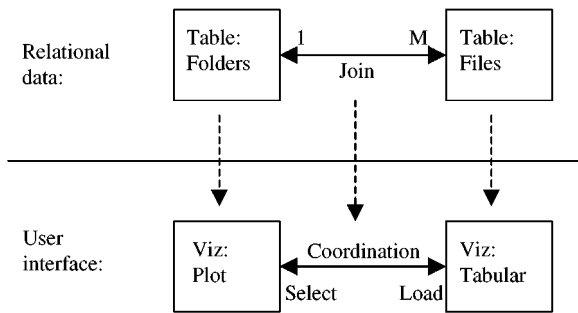


FIGURE 4. Snap’s model maps relational data concepts to user-interface concepts. This illustrates the file-folder example in Figure 2.

item in the visualization. For example, a scatter plot displays each tuple as a dot using two of its attributes as the coordinates. The relation must have a primary-key attribute to uniquely identify individual tuples. We assume that queries inherits a primary-key attribute from a base table.

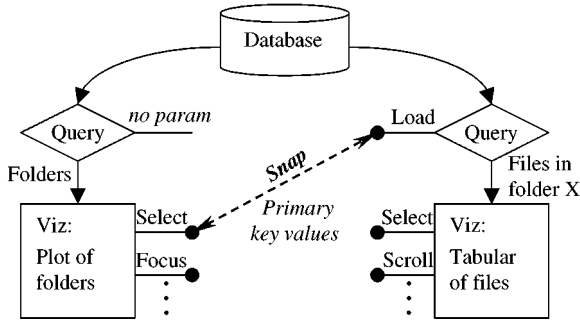


FIGURE 5. Components of Snap. Users first load queries into visualizations, then snap them together. Coordinations tightly couple actions between visualizations.

Each visualization supports a set of actions that can be performed on individual tuples. These actions allow users to indicate interest in a tuple. They can also be invoked by the system, using primary-key values to identify the tuple to act on. Each visualization tool publishes the set of actions it supports to Snap. Example actions include the following.

- *Select*: selecting a tuple to visually highlight it. For example, clicking on a dot in a scatter plot might color the dot bright yellow.
- *Scroll, zoom*, etc: navigating to a tuple to bring it to the center of view. For example, scrolling a list to bring an item to the top of the window.

When initially loading a query into a visualization, users may employ a parameterized selection query. The parameter value is used as selection criteria in the query. Different parameter values can be plugged in to select different subsets of a table. In this case, the visualization also has a **Load** action. Invoking this action and supplying a parameter value re-executes the query and loads the new results into the visualization. This enables a visualization to be used to display different portions of a large data table based on external input.

**3.2.2. Coordinating visualizations.** After loading relations into visualizations, users can then establish coordinations between the visualizations (“snap them together”). A Snap coordination tightly couples an action on items in one visualization to an action on items in another visualization. Then, when the user invokes the former, Snap automatically invokes the latter and vice versa. To establish a coordination between a pair of visualizations, users choose the actions in each visualization to tightly couple (Figure 5). Users are guided by the type of the join relationship between the relations in the two visualizations: one-to-one or one-to-many.

1. **One-to-one:** this relationship is often the result of displaying different projections of the same table in multiple visualizations. Examples of the common coordinations described in the introduction that are one-to-one are as follows.

- *Brushing-and-linking:*  
Join relationship: one-to-one

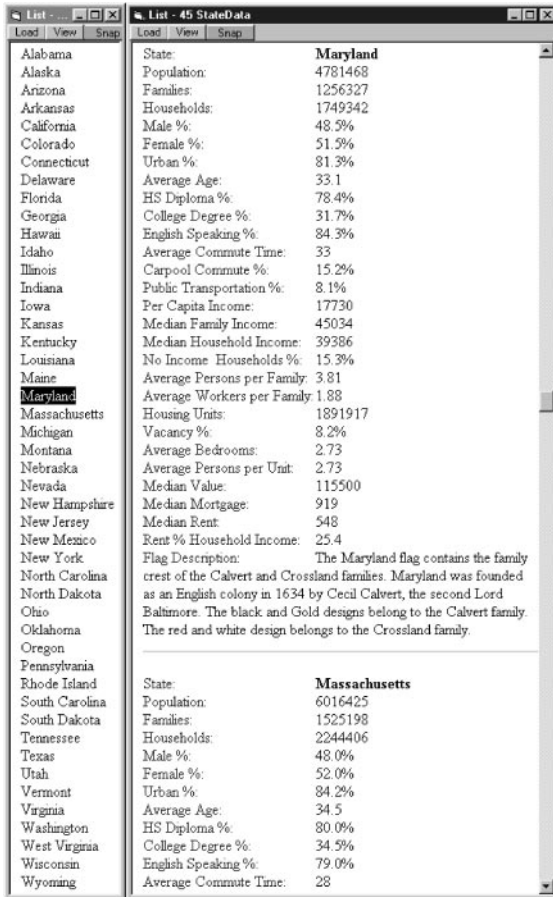


FIGURE 6. The user interface for Exercise 1 of the first study and also for the second study. A pair of textual visualizations with overview and detail-view coordination for browsing detailed census data of the US states.

Coordinated actions: **select** in visualization1 and **select** in visualization2. This links the primary-key value of the select action in visualization1 to the primary-key value of the select action in visualization2.

Operation: selecting an item in one visualization also selects (highlights) the corresponding item in the other visualization. For example, in Figure 1, selecting a folder in the Hyperbolic Tree highlights that folder in the scatter plot.

- *Overview and detail view:*

Join relationship: one-to-one

Coordinated actions: **select** in overview and **scroll** in detail view.

Operation: selecting an item in the overview scrolls (or more generally navigates) the detail view to the details of that item. Likewise, scrolling the detail view selects the currently viewed item in the overview. For example, in Figure 6, selecting a state from the list on the left jumps the scrolling detail view to the section about that state.

- *Synchronized scrolling:*

Join relationship: one-to-one

Coordinated actions: **scroll** in visualization1 and **scroll** in visualization2.

Operation: scrolling through a list of tuples in one visualization also scrolls to corresponding items in another visualization.

2. One-to-many: this relationship indicates a hierarchical structure between the relations. Each parent tuple in the first relation has many child tuples in the second relation. A common coordination for this relationship type is as follows.

- *Drill-down:*

Join relationship: one-to-many

Coordinated actions: **select** in parent visualization (*one*) and **load** in child visualization (*many*). The query in the child visualization is parameterized to select only those child tuples that are related to the parent tuple whose primary-key value is given as the argument. The primary-key value of the select action in the parent visualization is linked to the argument of the child query.

Operation: selecting an item in the parent visualization loads related items into the child visualization. For example, in Figure 1, selecting a folder in the scatterplot displays the files related to that folder in the tabular visualization. There is a one-to-many relationship in the database between folders and files. The “Files of a Folder” query loaded into the tabular visualization is essentially: “SELECT \* FROM Files WHERE Files.ParentFolder = ?”. Selecting a folder in the scatterplot binds its primary-key value to the “?” parameter and re-executes the query.

Snap coordinations are bi-directional, so that either action triggers the other. Users can also chain coordinations end-to-end. For example, users can establish brushing-and-linking across three visualizations. In Figure 1, selecting a folder in the Hyperbolic Tree also selects it in the scatterplot, which in turn loads files into the tabular visualization. After construction, users can save a set of coordinated visualizations as a group for later re-use or sharing.

### 3.3. SOFTWARE ARCHITECTURE

The Snap system acts as intermediary between visualization tools and handles all database access. Snap uses a very simple application programming interface (API) to communicate with visualization tools. Hence, researchers and developers can easily snap-enable their independent visualization tools. We propose this API as a standard, analogous to APIs for cut-and-paste capabilities in modern window systems, that would broadly enable this advanced functionality and greatly increase the applicability of many visualization tools.

Tools that have been enabled include a tabular visualization, a variety of textual list visualizations, an outliner, Spotfire scatterplots, Treemaps, Hyperbolic Trees, Internet Explorer, ArcView maps, image thumbnail browser, etc. Snap has been used with Microsoft Access and Oracle databases. Snap is implemented on the Windows platform, using COM to communicate with visualizations and ODBC for database access. When

using Microsoft Access databases, Snap uses Access's GUI to enable users to create and edit queries and manipulate the data schema.

For more details about the Snap system, see (North, 2000).

### 3.4. ADVANTAGES AND DISADVANTAGES

When constructing a coordinated-visualization interface, Snap has advantages (+) and disadvantages (–) over programming the interface by hand (hard coding the desired coordinations between visualizations):

Snap	Programming
+ Non-programmers (for enabled visualizations)	– Programmers only
+ Quick and easy	– Time consuming and difficult
+ Can make throw-away solutions for temporary or short-term needs	– Short-term needs go unmet
+ Interfaces are changeable on the fly	– Static, inflexible, slow turn-around
+ Can prototype many options	– Prototypes typically non-functional
+ Robust coordination model	– Prone to mistakes, inconsistencies
+ Guided by Snap model	– Design from scratch
+ Once enabled, visualizations are reusable in many different interfaces	– Visualizations hard-coded each time
– Potentially disparate visualizations	+ Package in custom user interface
– Bounded functionality	+ Custom functionality as needed

The Snap architecture is designed to use independent visualization tools. An alternate approach would be to fully integrate visualizations by custom implementing them within the context of the coordination system [as in Visage, DEVise, Spotfire, etc.]. Each approach has corresponding advantages (+) and disadvantages (–):

Independent visualizations	Integrated visualizations
+ Open system, others can easily add visualizations	– Closed system, only system developer can add visualizations
+ Reuses existing visualizations from the field	– Popular visualizations must be re-implemented within the system

+ Visualization development unaffected	– Visualizations must use designated structures
+ Visualizations can be used outside the system	– Visualizations only work within the system
+ Clean component-based design, visualizations insulated via API	– Potential inter-dependency complexities
+ Consistent coordination model	– Potential coordination inconsistencies
– Use only existing functionality of visualizations	+ Can add new functionality to visualizations
– Visualization user-interface inconsistencies	+ All visualizations implemented with same look and feel
– Potential performance hit	+ Potential performance boost from shared data structures, etc.
– Static coordination model	+ Can add advanced custom functionality for coordinating dynamic data, edits, etc.

#### 4. Empirical evaluation

Studying the use of Snap is important for the following two reasons.

- To evaluate the usability and benefit of the Snap system itself and discover potential user-interface improvements.
- To gain a deeper level of understanding about users' ability to understand, construct and use coordinated-visualization strategies in general. These are critically important issues for visualization researchers and user-interface designers.

The Introduction section of this paper concludes that users should be provided with the capability to construct coordinations between visualizations. This conclusion is valid only if the capability is both (1) usable and (2) beneficial. Hence, two separate studies were undertaken to evaluate these two distinct aspects of coordination:

- (1) *Construction*: can users successfully construct their own coordinated-visualization interfaces?
- (2) *Operation*: can users then operate the constructed coordinated-visualization interfaces to explore information beneficially?

##### 4.1. EVALUATION OF COORDINATION CONSTRUCTION

The first study uses Snap to examine the construction phase of coordination. The goal of this study is to determine if users can learn to construct coordinated-visualization interfaces and how difficult it is for users to construct them, in terms of success rate and time to completion and to identify cognitive trouble-spots in the construction process.

Can users grasp the concept of coordinating two independent visualizations together to form a unified browsing tool? What cognitive issues are involved, how much training is required, how do users' backgrounds affect performance and can relatively novice users construct powerful exploration environments in a short time? This study also reveals potential Snap user-interface improvements.

*4.1.1. Procedure.* We worked with six subjects on a one-on-one basis. Four of the subjects were employees of the US Bureau of the Census, three of whom were data analysts or statisticians, and one a programmer. The other two subjects were computer science students on campus.

First, background information was obtained from each subject concerning their occupation and experience with census data, computers, databases, Microsoft Access, visualization tools and programming.

Then, each subject was trained on Snap. The training program consisted of the following.

- (1) A quick demonstration of Snap by the administrator to give the subject an overview and motivation.
- (2) Review of relational database concepts including: tables, records, fields, primary keys, foreign keys; database query concepts including: projection, selection, sort, join; and Snap model concepts.
- (3) Detailed instruction on the use of Snap and Microsoft Access (Access is used to construct queries). The subjects walked through the construction of a few variations of coordinated-visualization interfaces for exploring census data. This demonstrated how to construct common types of coordinations.

Then, when confident to continue, each subject began the testing phase. Subjects were given a database of census data for the US states and counties. Testing consisted of three exercises in which subjects were asked to construct a coordinated-visualization user interface according to a provided specification.

*Exercise 1:* The first specification consisted of a printed screenshot of the desired user interface (Figure 6). This trial was designed to be fairly easy, to be similar to those constructed in the training, and to build confidence.

*Exercise 2:* The second specification was also a screenshot (Figure 7), but more difficult. It involved a one-to-many join relationship, so that selecting a state would display data for that state's counties.

*Exercise 3:* The final specification consisted of a textual description of the browsing task that the constructed interface should support: "Please create a user interface that will support users in efficiently performing the following task: to be able to quickly discover which states have high population and high Per Capita Income, and examine their counties with the most employees." This trial was designed to test if subjects could think abstractly about coordination, and if they could think in terms of task-oriented user-interface design. It was also designed to allow for potential creativity and variation in subjects' solutions.

Finally, subjects were given the opportunity to freely explore the system, describe problems with the Snap user interface and offer suggestions for improvement.

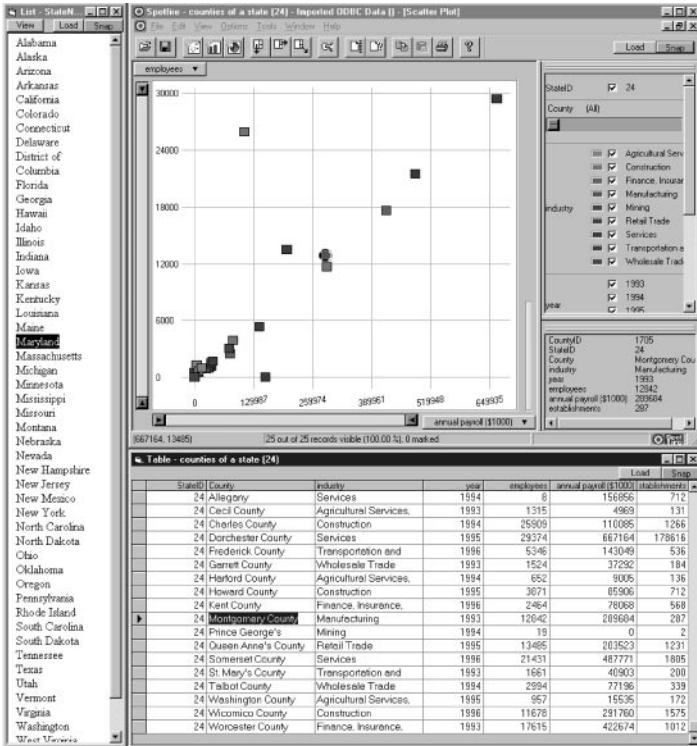


FIGURE 7. The user-interface specification for Exercise 2 of the first study. It uses a textual list, scatterplot and tabular visualization to explore census data for states and counties.

The following variables were measured.

- Subjects' background information.
- Learning time.
- Success (yes/no or how close to success).
- Time to completion.

This study also observed the following.

- Cognitive trouble spots (in training and test trials).
- Snap user-interface problems.

**4.1.2. Results.** From the background survey, none of the subjects except the Census programmer had experience with Access or SQL, and little exposure to relational database concepts. The Census analysts had significant experience with census data, but generally used flat files or spreadsheets. Each had experience with only basic visualization tools (e.g. Excel charts).

All the subjects completed the training phase in 30–45 min. They all were able to complete all three exercises, with occasional help from the administrators in using



Access's visual query editor. They accomplished Exercise 1 in 2–5 min, and Exercise 2 in 8–12 min. They spent 10–15 min on Exercise 3 until they were satisfied with their solution.

In general, the subjects were quick to learn the concepts and usage, and were very capable to construct their own coordinated–visualization interfaces. Several stated that they had a sense of satisfaction and power in being able to both (1) so quickly snap powerful exploration environments together and (2) with just a single-click effect exploration across several visualizations and see the many parts operate as a whole. They reported that it made exploration seem “effortless”, especially in comparison to the standard tools they are used to. As to the subjects' general reaction to Snap, they clearly showed enthusiasm. There may have been social pressure to respond positively, since the subjects knew that the administrator of the experiment was also the developer of the Snap system.

There was an interesting difference between the reaction of the data analysts and programmers (census programmer and computer science students). The programmers commented enthusiastically about the component-based programming approach and the ability to rapidly construct new interfaces. Whereas, the data analysts commented about being able to explore the data thoroughly and efficiently. They did not see it as construction, but as exploration.

In fact, the data analysts performed better than the programmers. They learned the database concepts quicker, completed the exercises quicker and constructed creative interesting new interfaces. Perhaps they were more motivated by the use of examples involving Census data. Even during the training, they were already trying variations of coordinations and exploring the data. Two pointed out various anomalies in the data. After finishing the exercises, these subjects each voluntarily stayed for an additional hour to discuss and try other examples. All four Census subjects expressed desire to use Snap in their work. In fact, a collaborative effort has since been undertaken.

An important result was the creativity and variation evident in the subjects' solutions to Exercise 3. Subjects were able to design user interfaces that made cognitive sense to their own perspective on the data. They used a mixture of visualizations including tables, scatterplots, lists and outliners. For example, while the expected design was two scatterplots with a drill-down coordination (one-to-many, select to load), one of the data analyst subjects augmented this design with a pair of lists for the state and county names (similar to Figure 3). The subject stated that this would help to see which state and county was currently selected in the scatterplots, and also allow for accessing states by name which would be difficult with a scatterplot alone. Another subject who preferred to see numeric values placed the counties in a table sorted by number of employees. One had even constructed an interface using the treemap visualization, which is generally considered a more advanced visualization difficult for novices. In addition to variation in user interfaces, subjects made use of the transitive property of coordination to coordinate visualizations in different pairings.

Overall, subjects did not have problems grasping the cognitive concept of coordinating visualizations. They were able to generate designs by visual duplication and by abstract task description. Results from Exercise 3 demonstrated that these users were able to design appropriate coordinated–visualization interfaces. These encouraging results indicate that users can handle a mid-level of design in which they piece together

pre-designed components to construct a larger design. Snap apparently finds a middle ground between usage (the realm of end-users) and design (the realm of experienced HCI practitioners) appropriate for these data-savvy users. This validates the primary benefit of the Snap capability, its flexibility.

The problems subjects did have were in manipulating the Snap and Access user interfaces. Creating queries was by far the most difficult part of the construction process for the subjects. Learning to use Access and its query editor is a challenge in such a short time.

*4.1.3. User-interface issues.* Understanding the basic underlying model of Snap was critical. However, the current Snap user interface and the form fill-in style of the Snap specification dialog does not reflect this model well. This study identified four major trouble spots in the interface.

- (1) The terminology of the snap-able actions “select” and “load” caused some confusion. It was not clear enough that these represented user-interface actions. Apparently, some subjects were confusing “select” with the database query sense of selection.
- (2) Constructing a drill-down coordination (one-to-many, select-to-load) was very laborious, and subjects sometimes got lost in the three-step process: writing the parameterized query, opening the query in a visualization and specifying the coordination.
- (3) When constructing interfaces of three or more visualizations, subjects sometimes forgot what coordinations they had constructed between visualizations. They had to recheck them individually.
- (4) When subjects were not quite sure what coordinations they should construct, they would often “just try stuff” and see how it behaves. A coordination debugging mode is needed to help them see how the tight-couplings propagate between the visualizations.

Redesigning the Snap user interface around an overview diagram would solve these problems. A node and link diagram could represent the visualizations as nodes and coordinations as links between them. This overview could become the primary user interface for constructing, editing, examining and debugging coordinations. Such a visual representation with direct-manipulation interaction would closely reflect the conceptual Snap model (a “visualization schema” analogous to the data schema). Hence, this would likely reduce users’ training time as well.

In addition, while the ability to create queries with Access enables more complex scenarios, it is a burden for common simple coordinations. Basing the Snap specification dialog on the database schema diagram would more closely match users’ mental model of the data. This would simplify constructing drill-down coordinations since Snap could generate the parameterized selection queries automatically. For projection queries, expanding the Snap Menu window to include attribute names would allow users to directly select desired attributes to load into visualizations. Together, these modifications would obviate the need to use Access to manually create queries in common cases. This could further reduce training time to almost nothing.

Also, window management is a serious problem. Subjects spent considerable amounts of time rearranging visualization windows on the screen into nicely tiled layouts. Others have proposed solutions to this general problem (see Kandogan & Shneiderman, 1997 for a review).

#### 4.2. EVALUATION OF COORDINATION OPERATION

The second study uses Snap to examine the operation phase of coordination. The goal of this study is to measure the benefit of coordinating visualizations as compared to simply using single or multiple uncoordinated visualizations. The visual feedback across coordinated visualizations could be distracting or disorienting for users. But if there is a benefit, what is its magnitude in terms of user task times and subjective satisfaction for browsing large information spaces?

While there are many possibilities, this study examines the overview-and-detail coordination. This coordination has two enhancements over the traditional single-visualization detail-only display.

- (1) *Overview*: a display enhancement that depicts the full breadth of the data in a compact form, like a table of contents.
- (2) *Coordination*: an interaction enhancement that allows users to select an item in the overview to scroll the detail to that item. Likewise, directly scrolling the detail highlights the current item in the overview.

Chimera's results (Chimera & Shneiderman, 1994) seem to indicate that overview-and-detail should perform better than detail-only. But, if so, which enhancement is the important factor that causes improved user performance? Is it (1) the information displayed in the overview or (2) the coordination between the overview and detail?

Hence, the purpose of this study is not to compare a coordinated user interface with the best alternative (see Section 2.1 for such studies). Instead, the purpose is to further understand coordination and its users. Specifically, why and how much does the capability to coordinate an overview improve over using the detail only, in the context of a single popular type of navigation (one-dimensional scrolling) for browsing tasks? What is the value or detriment of multiple visualizations that are not coordinated? What are users' reactions to these interfaces?

*4.2.1. Independent variables. User interface.* A simple textual user interface, constructed with Snap, for browsing population statistics of 45 of the US states from the Census Bureau's 1990 census. Subjects did not need to use the Snap user interface, but used instances of a simple textual visualization tool that were coordinated by Snap *a priori* to browse the data. The three treatments (see Figure 6) are as follows.

- (1) *Detail-only*: a single scrolling textual report of the states, in alphabetical order, and their data (Figure 6, right-most window).
- (2) *No-coordination*: the same visualization as detail-only, with the addition of a second textual visualization tiled on the left. The second visualization displays only the

names of the states as an overview. The visualizations are *not* coordinated (for example, as if generated by Access).

- (3) *Coordination*: the same visualizations as no-coordination, with the addition of an overview-and-detail coordination between them. In Snap, this tightly couples the overview's *select* action to detail's *scroll* action.

The no-coordination interface treatment is included for two important reasons. First, no-coordination will reveal which aspect of the coordinated-visualization interface approach is most critical: the multiple visualizations or the coordination. Second, no-coordination mimics many common systems such as Access, Excel, web browsers and other flexibility level 2 or below systems, which do not allow users to construct custom coordinations between their visualizations. No-coordination also occurs when using multiple visualization tools that were not originally developed to work together. Without Snap, users cannot coordinate them.

*Task.* A variety of browsing tasks, using a question and answer approach. The nine treatments are as follows.

- (1) *Coverage-yes*: "does the information include statistics about the state of Ohio?" where Ohio is included in the data.
- (2) *Coverage-no*: same as coverage-yes, but where the state is not included in the data.
- (3) *Overview patterns*: "how many states in the list begin with the letter M?"
- (4) *Visual lookup*: "what is the population of the sixth state from the bottom of the list?"
- (5) *Nominal lookup*: "what is the population of Georgia?"
- (6) *Compare-2*: "which of the following states has higher median family income: California or Washington?"
- (7) *Compare-5*: "which of the following five states has higher median household income: Florida, Texas, Louisiana, Alaska or Oregon?"
- (8) *Search for target value*: "which state has average commute time of 31?"
- (9) *Scan all*: "which state has the highest college degree %?"

The tasks are listed here in order from easy to difficult based on the experiment results. The actual order they were administered was: 5, 1, 6, 8, 3, 7, 2, 9, 4.

*4.2.2. Dependent variables. User performance time:* time to correctly complete each task, not including reading the task question.

*User subjective satisfaction:* subjects rated their satisfaction with each interface on a scale of 1–9 on four categories (with scales): comprehensibility (confusing to clear), ease of use (difficult to easy), speed of use (slow to fast), overall satisfaction (terrible to wonderful).

*4.2.3. Procedure.* The 18 subjects were students and staff from campus, and were paid \$10 to participate. A within-subjects design was used. Each subject used all three interfaces to perform all nine tasks. To avoid repetition, three different but similar sets of task questions were used. To counterbalance for potential order effects, all six possible permutations of interface order were each assigned 3 times. The three task sets were not permuted.

For each interface, subjects were first trained in its use and performed several practice tasks before beginning the timed trials. After finishing all three interface treatments, subjects then completed the subjective satisfaction questionnaire.

**4.2.4. Results.** Analysis of the data reveals a strong and interesting result. Figure 8 shows the mean user-performance times for each task and interface. A  $3 \times 9$  within-subjects ANOVA reveals that the user interface effect, task effect and interaction effect are all statistically significant at  $p < 0.001$ . Nine one-way ANOVAs reveal that user interface is significant for all nine tasks at  $p < 0.001$ .

Finally, individual  $t$ -tests between each pair of user interfaces within each task determine performance advantages. For tasks 1–3, the coordination and no-coordination interfaces are both significantly faster than the detail-only interface at  $p < 0.001$ , but not proven different from each other. Whereas, in tasks 5–9, coordination is significantly faster than both no-coordination and detail-only at  $p < 0.001$ , and the latter are not proven different from each other. However, while task 4 (Visual lookup) could be included in the second group of tasks, it may classify as an in-between case. For this task, coordination is significantly faster than the other two user interfaces at  $p < 0.005$ , but no-coordination is marginally significant over detail-only at the  $p < 0.07$  level.

First, coordination results in major improvement in user performance time over detail-only for all tasks. On average, coordination achieves an 80% speedup over detail-only for easy tasks and 50% for difficult tasks. The least improvement, about 33%, is in task 6 (compare-2). This task had the lowest interaction-time to thinking-time ratio.

The no-coordination interface results in a nearly binary pattern, and is likely the source of the interaction effect between task and interface (see Figure 9). For tasks 1–3, no-coordination performs faster than detail-only, and its averages are similar to coordination. In these tasks, subjects only needed the information in the overview to accomplish the task. Whereas, in tasks 5–9 the coordination interface is faster than no-coordination and the averages for no-coordination are similar to detail-only. In these

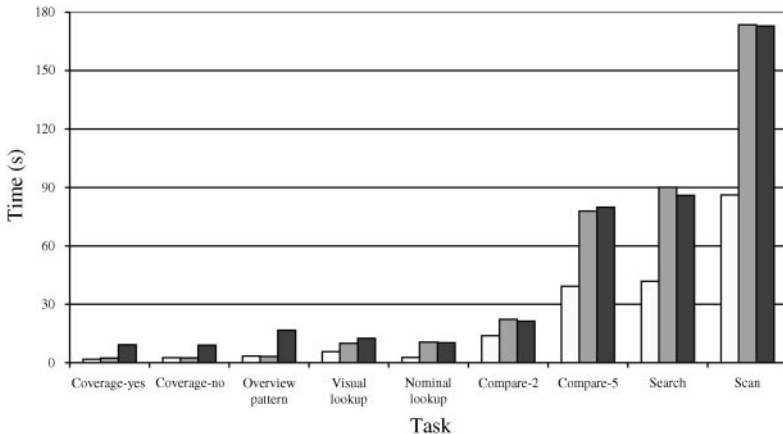


FIGURE 8. Average user-performance time for tasks in the second study. The coordinated interface has significantly faster performance in most cases: □ coordination; ■ no-coordination; ■ detail-only.

	Tasks	
	1-3	4-9
Slower group	Detail-only	Detail-only No-coordination
Faster group	No-coordination Coordination	Coordination

FIGURE 9. User-interface treatments in the second study, grouped by user performance in tasks. The faster groups are significantly faster than the slower groups at  $p < 0.005$ . The no-coordination interface results in a binary effect.

tasks, subjects needed to access the details of the data. Observing subjects' behavior as they performed these tasks revealed that when using no-coordination they tended to ignore the overview. The lack of significant difference between no-coordination and detail-only in these cases does not imply that they are necessarily the same. It is conjectured that they are the same due to the observation of the users. In any case, coordination is significantly faster than no-coordination for these tasks. Hence, in tasks where access to details is important, undoubtedly a majority in common applications, coordination is absolutely critical.

Task 4 (Visual lookup) might classify as an in-between case. With no-coordination, many subjects determined the name of the target state from the overview, then scrolled to it in the detail view. With detail-only, they scrolled to the bottom, then scrolled back up while counting, and sometimes lost track. Apparently, this is a case where simply having the contextual information of the overview was somewhat advantageous. Coordination was still a major improvement over both.

In fact, an important result is that coordination performance times for lookup tasks (4 and 5) are in the same extremely fast range as overview tasks 1-3. Whereas, no-coordination times drop to detail-only level performance. When looking up details, perhaps the most common task, coordination especially excels.

In general, overview-and-detail coordination greatly improved performance over detail-only scrolling. Clearly, a major advantage of the coordination is the ability to directly select a target in the overview to immediately locate its details. Whereas, the scrolling interfaces require careful searching while dragging the scroll bar thumb. Observing the subjects as they performed the tasks revealed that they were more likely to explore when using coordination. For example, in the Compare-2 and Compare-5 tasks, subjects were more willing to recheck their answers with coordination. With detail-only and no-coordination subjects spent extra effort to mentally alphabetize the five states to compare so as to minimize their scrolling effort. Several subjects reported verbally and on the questionnaire that scrolling was difficult. This is surprising since scrolling is a fundamental component of current GUI systems and perhaps the most common navigational method. The coordination interface could be considered an improved scroll bar that facilitates exploration.

*4.2.5. Subjective satisfaction.* With the satisfaction data (Figure 10), a  $3 \times 4$  within-subjects ANOVA indicates that user interface, subjective satisfaction category, and interaction effect are all significant at  $p < 0.001$ . One-way ANOVAs for each category

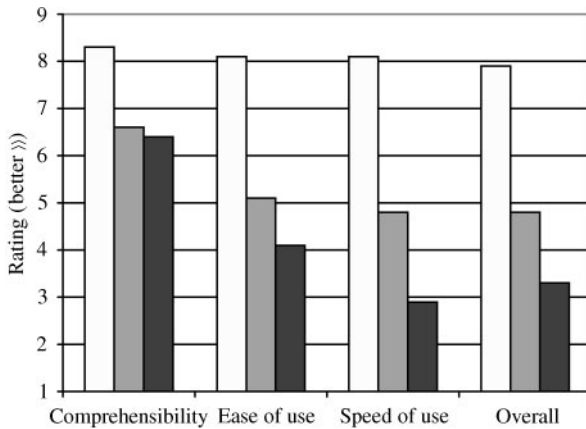


FIGURE 10. Average user subjective satisfaction in the second study. The coordinated interface rates significantly higher in all four categories: □ coordination; ■ no-coordination; ■ detail-only.

indicate that comprehensibility, ease of use, speed of use and overall satisfaction are all significant at  $p < 0.001$  level.

Analysing each pair of interface treatments within each category reveals that all pairs are significant at  $p < 0.001$  except: detail-only and no-coordination in ease of use are significant at  $p < 0.05$  and the same pair in comprehensibility are not proven different.

Coordination is a clear winner, gaining nearly twice the rankings of detail-only and no-coordination in ease, speed, and overall. On average, subjects ranked no-coordination 1–2 points higher than detail-only, except in comprehensibility they ranked about the same. While completing the survey, several subjects stated that no-coordination was only useful for the overview tasks.

**4.2.6. Answers.** Returning to the research questions: which factor is more critical, the overview information or the coordination? The answer is nearly binary. If only the overview information is needed, then naturally coordination is not necessary. But for the important cases where access to details is needed, then coordination is critical. What is the magnitude of the benefit? For the three most difficult tasks, the coordinated version cut tasks time in half. This study also reveals the importance of good overview design to enable common questions to be answered directly from the overview.

When first presented with the no-coordination interface, many subjects immediately attempted to click in the overview expecting the detail view to change, even when they had not yet seen the coordination interface. Hence, subjects were not distracted by this coordination, but rather they desired and expected it. They were visibly distraught when the interface did not behave as they hoped. They were clearly more satisfied with the coordination interface, as the subjective satisfaction data indicates. Subjects verbally expressed appreciation for the interactive coordination that sped their tasks.

### 4.3. COMBINED ANALYSIS

Combining the results from these two studies may indicate the breakpoint at which time savings during coordination operation surpass coordination construction time. In

Exercise 1 of the first study, subjects constructed the same user interface as was used in the second study for browsing tasks. The time cost of constructing the coordinated interface was about 2–5 min, while it saved about 0.6–1.5 min over the standard detail-only interface for the more difficult tasks. Hence, after just a few tasks, users are already reaping savings when constructing their own coordinated interface. Of course, it is difficult to factor in learning time and effects of sharing saved interfaces. Nevertheless, this simple analysis reveals that customized information visualization is within the grasp of data users.

## 5. Conclusions and future work

This paper validates the capability for data users to construct their own customized coordinated visualization environments for exploring data. The Snap conceptual model and software architecture demonstrate that such a capability is technologically feasible. The Snap user interface and two studies demonstrate that such a capability is both usable and beneficial.

The overview-and-detail coordination offered a 30–80% speedup over detail-only scrolling for all nine user tasks. While the uncoordinated overview was sufficient for overview only tasks, coordination was critical when accessing details. Data-savvy users successfully and enthusiastically designed and constructed coordinated interfaces of their own, showing creativity and variation. Users readily grasped coordination based on relational data schemas, although querying was problematic. These users are clearly ready for and strongly desire significantly more advanced tools than standard detail-only, uncoordinated or hard-wired systems. While these cognitive issues were examined within the Snap platform, we believe that these results will apply to similar coordinations and flexibility in other systems.

The implications for practitioners are as follows.

- User-interface designers should employ coordination strategies, such as overview-and-detail, in their designs to improve user performance and satisfaction.
- Designers of systems that display multiple visualizations (uncoordinated or flexibility level 2 and below) can implement Snap-like coordination construction concepts into them, advancing them to flexibility level 3.
- Developers of independent visualization tools and operating systems can integrate Snap APIs and architectures to enable universal coordination capabilities. For example, Snap concepts could be integrated into a data standard such as ODBC.

The design of each coordination is critical, and others need to be evaluated. Of particular interest are the drill-down and brushing-and-linking coordinations. In addition, these studies have identified major improvements to the Snap user interface for coordination construction based on cognitive issues. Additional work is needed to discover if user training requirements can be eliminated by basing the interface on data schema diagrams. Continued research is needed to explore coordination overviews, coordination guidelines, automated coordination suggestion and more.

This research was partially supported by funding from West Group and the US Bureau of the Census. Thanks to Kent Norman for advice on the second study. Thanks also to Ben Bederson, Dave Mount, Adam Porter and Allan Kuchinsky for comments.



## References

- AHLBERG, C. & WISTRAND, E. (1995). IVEE: an information visualization and exploration environment. *Proceedings of the IEEE Information Visualization Conference '95*, pp. 66–73.
- BALDONADO, M., WOODRUFF, A. & KUCHINSKY, A. (2000). Guidelines for using multiple views in information visualization. *Proceedings of the Advanced Visual Interfaces Conference 2000*.
- BEARD, D. & WALKER, J. (1990). Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour & Information Technology*, **9**, 451–466.
- BECKER, R. & CLEVELAND, W. (1987). Brushing scatterplots. *Technometrics*, **29**, 127–142.
- BORNING, A. (1986). Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, **5**, 345–374.
- CARD, S., MACKINLAY, J. & SHNEIDERMAN, B. (Eds). *Readings in Information Visualization: Using Vision to Think*. Los Altos, CA: Morgan Kaufmann.
- CHI, E. H., BARRY, P., RIEDL, J. & KONSTAN, J. (1997). A spreadsheet approach to information visualization. *Proceedings of the IEEE Information Visualization Conference '97*, pp. 17–24.
- CHIMERA, R. & SHNEIDERMAN, B. (1994). An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents. *ACM Transactions on Information Systems*, **12**, 383–406.
- DUMAS, J. & PARSONS, P. (1995). Discovering the way programmers think about new programming environments. *Communications of the ACM*, **38**, 45–56.
- EICK, S. & WILLS, G. (1995). High interaction graphics. *European Journal of Operations Research*, **81**, 445–459.
- FREDRIKSON, A., NORTH, C., PLAISANT, C. & SHNEIDERMAN, B. (1999). Temporal, geographical and categorical aggregations viewed through coordinated displays: a case study with highway incident data. *Proceedings of the ACM CIKM '99 Workshop on New Paradigms in Information Visualization and Manipulation*, pp. 26–34.
- FURNAS, G. (1986). Generalized fisheye views. *Proceedings ACM CHI Conference '86*, pp. 16–23.
- HAEBERLI, P. (1988). ConMan: a visual programming language for interactive graphics. *Proceedings of the ACM SigGraph Conference '88*, pp. 103–111.
- ISAKOWITZ, T., STOHR, E. & BALASUBRAMANIAN, P. (1995). RMM: a methodology for structured hypermedia design. *Communications of the ACM*, **38**, 34–44.
- JACOBSON, A., BERKIN, A. & ORTON, M. (1994). LinkWinds: interactive scientific data analysis and visualization. *Communications of the ACM*, **37**, 43–52.
- KANDOGAN, E. & SHNEIDERMAN, B. (1997). Elastic windows: evaluation of multi-window operations. *Proceedings of the ACM CHI Conference '97*, pp. 250–257.
- LAMPING, J. & RAO, R. (1996). The hyperbolic browser: a focus + context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, **7**, 33–55.
- LIVNY, M., RAMAKRISHNAN, R., BEYER, K., CHEN, G., DONJERKOVIC, D., LAWANDE, S., MYLLYMAKI, J. & WENGER, K. (1997). DEVise: integrated querying and visual exploration of large datasets. *Proceedings of the ACM SIGMOD Conference '97*, pp. 301–312.
- NORTH, C. (2000). *A user interface for coordinating visualizations based on relational schemata: snap-together visualization*. Doctoral Dissertation, University of Maryland, Computer Science Department. <http://www.cs.umd.edu/hcil/snap/> (May 2000).
- NORTH, C. & SHNEIDERMAN, B. (1997). *A taxonomy of multiple window coordinations*. Technical Report #CS-TR-3854. University of Maryland, College Park, Dept of Computer Science.
- ROTH, S., LUCAS, P., SENN, J., GOMBERG, C., BURKS, M., STROFFOLINO, P., KOLOJECHICK, J. & DUNMIRE, C. (1996). Visage: a user interface environment for exploring information. *Proceedings of the Information Visualization Conference, IEEE*, pp. 3–12.
- SHNEIDERMAN, B. (1992). Tree visualization with treemaps: a 2-d space-filling approach. *ACM Transactions on Graphics*, **11**, 92–99.
- SHNEIDERMAN, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (3rd Edn.). Reading, MA: Addison-Wesley.
- SHNEIDERMAN, B., SHAFER, P., SIMON, R. & WELDON, L. (1986). Display strategies for program browsing: concepts and an experiment. *IEEE Software*, **3**, 7–15.
- VELLEMAN, P. (1988). *The Datasheet Handbook*, Odesta Corporation.