

Exploratory Access to Geographic Data Based on the Map-Overlay Metaphor*

Max J. Egenhofer and James R. Richards
National Center for Geographic Information and Analysis
and
Department of Surveying Engineering
University of Maine
Orono, ME 04469, U.S.A.
{max, richards}@mecan1.maine.edu

Abstract

Many geographic information systems (GISs) attempt to imitate the manual process of laying transparent map layers over one another on a light table and analyzing the resulting configurations. While this map-overlay metaphor, familiar to many geo-scientists, has been used as a design principle for the underlying architecture of GISs, it has not yet been visually manifested at the user interface. To overcome this shortage, a new direct manipulation user interface for overlay-based GISs has been designed and prototyped. It is characterized by the separation of map layers into data cubes and map templates such that different thematic data can be combined and the same kind of data can be displayed in different formats. This paper introduces the conceptual objects that the user manipulates at the screen surface and discusses ways to visualize effectively the objects and operations upon them.

1 Introduction

Sophisticated user interfaces have employed metaphor to improve the user's level of understanding. Metaphors are concepts with which humans associate specific basic tasks, because they map from a source domain, which the user is presumably familiar with, to a target domain, which the user is trying to master [1]. Additionally, a metaphor can be extended to include concepts that are not in the source domain, but that improve the usability of the final product, without disrupting one's understanding of the metaphor [2]. In order for a metaphor to be effective for a particular target domain, there must be a morphism between the source and target domains [3]. At the user interface, metaphors can be employed to manipulate the target domain, without the need for the user to fully understand the underlying architecture. In human-computer interaction, the direct manipulation paradigm has proven to be an effective user interface design, as evidenced by the enormous popularity of Macintosh computers. The direct manipulation paradigm is the result of a metaphorical mapping from the way humans perceive haptic space—that is based on experiences of touch and movement through space—onto the way humans interact with digital data in a computer [4]. Examples of successfully used direct manipulation metaphors in user interface design are the desktop with its folders and the trash can [5, 6].

Despite the current dominance of graphical user interfaces that build on metaphor and utilize direct manipulation, many user interfaces to geographic information systems (GISs) are still abstract and command line based. This shortcoming is partially to be attributed to the conceptual complexity of geographic data, which presents an assortment of unique interaction problems [7]. Only recently, investigations of metaphors appropriate for dealing with geographic data [4, 8, 9] have influenced the design of some GIS and image processing user interfaces [10, 11]. The

* This work was partially supported by grants from Intergraph Corporation. Additional funding from NSF for the NCGIA under grant no. SES 88-10917 is gratefully acknowledged.

primary metaphor employed is “pan and zoom,” which allows the user to wander around in geographic space and to change the level of detail in the space displayed [9]. This metaphor is a first step in designing advanced GIS user interfaces as it provides an intuitive way of browsing through geographic space. Using the analogy of the complex desktop metaphor, zoom may correspond to “double clicking on a file icon to open a view on a document” and pan is “scrolling in a text document to see another piece.” It is necessary to incorporate such ways of viewing geographic data into a larger framework in which users can also explore the *thematic content* of a geographic database. The scope of this paper is the investigation of the larger setting, the *geographer’s desktop*, on which a variety of *spatial manipulations* are performed.

The source domain for the geographer’s desktop is borrowed from environmental planning and landscape architecture. Professionals in these disciplines use the “map overlay” paradigm, which is based on the idea of drawing a separate, single-factor transparent sheet for each theme, placing them over one another on a light table, and viewing the resulting integrated spatial information that would not have been visible on a single sheet. Here, we use this method of exploratory access to geographic information as the basis for the design of a direct-manipulation GIS user interface. Like the planner who puts transparencies on a light table, we envision that GIS users will select different kinds of geographic data resident in a GIS, and move them onto a viewing platform, where an integrated view of the data will be available. Complementary methods of achieving this kind of analysis involve spatial query languages [7, 12] to access data if users have explicit knowledge about values of objects and constraints among them; and browsing [13] if users navigate quickly through geographic or attribute spaces.

The remainder of this paper investigates the principles of designing a direct-manipulation user interface that employs the map-overlay metaphor at the screen surface, and presents one possible implementation of visualizing and interacting with the relevant objects in terms of a guided tour. It follows the design of a user interface based on *data cubes* and *maps* [14], which are representations of the separation of thematic layers into retrieval and presentation specific query operations [15]. The primary concern is with the visualization of the objects presented to the user at the screen surface and the implementation of direct-manipulation operations. One of the goals is to overcome some of the limitations discussed above by emphasizing that GIS user interface design should not just focus on the way people interact with computers, but should be based on how people go about problem solving when interacting with geographic information itself [16]. We will attempt this by proposing a direct manipulation language in which users manipulate objects rather than circumscribing the actions by constructing sentences. It is exactly this property that distinguishes our language from Kirby and Pazner’s [17] or Lanter and Essinger’s [18] iconic languages for map overlay. In our language, operations on geographic information are performed as actions rather than by structuring sentences with static icons.

The next section reviews previous approaches to designing user interfaces for overlay-based GISs. Section 3 details the objects which users deal with in our user interface and discusses the corresponding operations. Section 4 describes the visualization of the objects and their operations, concentrating on the methods to convey feedback to the user. Section 5 presents a guided tour through our system, visualizing the primary concepts. The conclusions in section 6 describe our prototype and discuss future research within the proposed framework.

2 Map Overlay

Map overlay consists of manually overlaying transparent “maps,” such that one can see different thematic layers in the same geographic extent [19]. The process involves drawing single-factor maps of different thematic information on transparent media at the same scale and over the same geographic extent. Once the maps are drawn they can be laid on top of one another in various combinations to examine the spatial relationships between different themes (Figure 1).

Disciplines that utilize the manual map overlay technique extensively include environmental planning, landscape architecture, and geo-sciences. See Chan and White [20] for a comprehensive review of the history of map overlay.

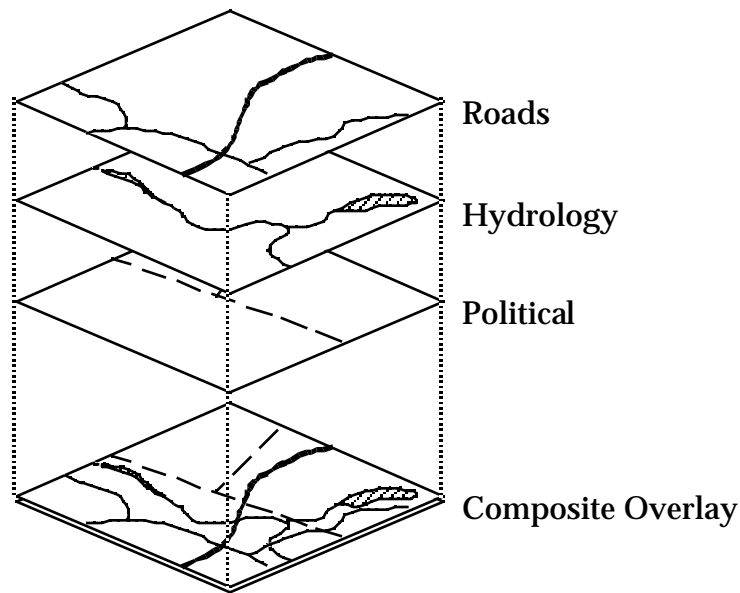


Figure 1: The overlay concept

2.1 Map Overlay in GIS

Map overlay has been used successfully as a design principle for geographic information systems by separating geographic data into thematic map layers [21, 22]. GISs that have been built on this concept have used such terminology as layers, coverages, and themes to embody this concept. An extensive discussion of the types of applications suitable for implementation in a GIS is available in [23]. A comprehensive discussion of GIS operations in terms of a “map algebra” is given in Dana Tomlin’s Map Analysis Package [24], which describes informally the kinds of manipulations that are usually done with map layers.

In addition to a simulation of the manual overlay process, the computerized map overlay process allows GIS users to perform other analytical operations, also called *computational overlays*. An example of a typical computational overlay is the creation of a buffer zone around sensitive ecological areas such as wetlands. This kind of operation takes as its input the layer containing spatial information about wetlands and a distance parameter specifying the size of the buffer zone. It returns a layer specifying the spatial extent of the area within the specified buffer distance around the wetlands. This is an established example of how the map overlay metaphor has been extended to include additional GIS operations.

The manual map overlay process includes the inherent connection between the geographic data and their graphical presentation. In a cartographic environment, graphical presentation is an important means to convey information [25]. Applied to the map layers, a new layer has to be drawn whenever some display parameters such as color, line width, or filling pattern, need to be changed. In a computational environment, this limitation can be overcome by separating the layer into a data component and a component with the visualization parameters [14, 15].

This method of access to geographic information is exploratory because: (1) users need only little knowledge about particular values of the data involved; and (2) users examine visually the integration of different kinds of spatial data, looking for interesting patterns, neighborhoods, or coincidence. For example, one may want to find areas that are threatened by land slides. To do so, one would overlay a layer of “soils” with a layer of “terrain steepness” to identify whether there is somewhere a particular area of an unstable soil type that coincides spatially with a critically steep slope.

2.2 Map Overlay in GIS User Interfaces

While the overlay metaphor has been used for the conceptual model and architecture of these systems, it has not been used effectively at the user interface level. Such a user interface would be highly desirable for a number of GIS user groups, as it supports the users’ familiar spatial concepts [26]. Traditionally, overlay based GIS have had command line interfaces requiring the user to learn a cryptic language and extensive vocabulary to operate the system. This can be very confusing for the user as the number of commands in a given package increases. For example, ARC/INFO contains over 1,200 different commands [27]. An advanced implementation of a GIS based on Tomlin’s MAP algebra offers a menu-based user interface [28]. In a further step, Kirby and Pazner [17] proposed to visualize the commands through icons for the objects involved *and* the operations upon them. Users formulate queries with direct manipulations on the icon *operations*, rather than executing the operations on the objects immediately. While these advanced interaction technique release the users from remembering and typing the commands, they still preserve the initial command structure of the underlying language and users have to think in terms of constructing “sentences.”

3 Data Cubes and Templates

As an alternative to the command-based language for map overlay, we propose a visual, direct-manipulation language based on the overlay metaphor. In this language, users perform actions much like the professional experts do in their familiar environment. They also receive immediate informational feedback about the status of operations being performed. This approach is superior to languages that describe operations verbally, because it reduces the “gulf of execution” (e.g., the extra effort required of users to successfully complete a task) at the user interface [29].

This section introduces the primitive user objects of our “map-overlay” language and describes the semantics of the operations that a user can perform with them. The separation of the “map layer” into *data* and their *display parameters* led to a language consisting of *data cubes* representing the data and *templates* that contain the display parameters for the data. Furthermore, the *viewing platform* is introduced as the location where overlays may occur.

3.1 Data Cubes

A *data cube*, or briefly *cube* (Figure 2), represents the spatial and attribute data of a single class, coverage, theme, or layer. This is similar to a table in relational algebra [30]; however, each cube has an inherent spatial location, orientation, and extent such that it may be considered a “spatial relation” to which also specific spatial operations apply [15, 31].

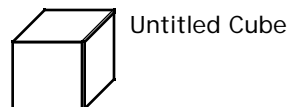


Figure 2: The data cube

It is important to note at this point that a cube is only a representation of data, without any knowledge about their presentation to the users. The structure of a cube was chosen for several reasons. One is that cubes are very basic to human understanding of spatial relationships, which are also probably acquired through bodily experience [1], and to graphic communication [32]. Consider that most humans as children experience learning through “building blocks,” and it becomes clear that cubes are appropriate as objects for spatial manipulations. Cubes “afford” stacking them [29] and, therefore, provide a strong clue to the operation users can execute on spatial data represented as cubes. A second reason is that cubes naturally convey the multidimensionality of the data that they represent. One can see that the minimum number of dimensions needed for storing GIS data is two for the spatial extent and a third for some kind of attribute information [14]. The different faces of the cube can also be used to suggest that data can take on multiple views [33]. For example, the data of a layer could be viewed geographically, statistically, or in tabular form such that each face of a cube represents a different analytical technique (Figure 3). Depending on the task at hand, the user may choose to explore the data in one or the other way. A third reason for using the cube structure is that it frees the naive user from needing to know how data is stored and retrieved from the database. Much like the object-oriented concepts of information hiding and data encapsulation [34], GIS users would see only a “black box.” They would know about the contents of that box, but would not have to understand the internal representation.

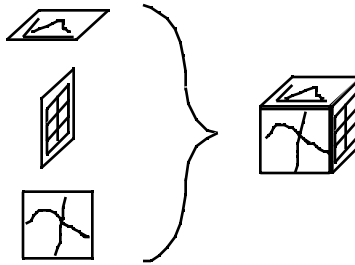


Figure 3: The multiple view concept applied to the data cube

Cubes over the same geographic extent can be combined by stacking them on top of one another, much like the layers that are put on top of one another in map-overlay, into a stack of cubes. It is assumed that the data of the cubes to be stacked cover the same area, with the same orientation and map-projection. Otherwise, more complex combinations apply, some of which will be covered later in this paper, others being disallowed due to meaningless semantics. For example, combining cubes that cover areas at different orientations, without accounting for them, is impractical.

A stack of data cubes associates with it the usual stack operations such as pushing a cube onto the stack (Figure 4a). In addition, a cube may be inserted into the stack at any place, not only on top of the stack (Figure 4b), and any cube of the stack may be removed without forcing the user to pop all cubes above it and to push them afterwards (Figure 4c). Likewise, the stacking order may be changed by reshuffling a cube to a new position without removing the cubes above (Figure 4d).

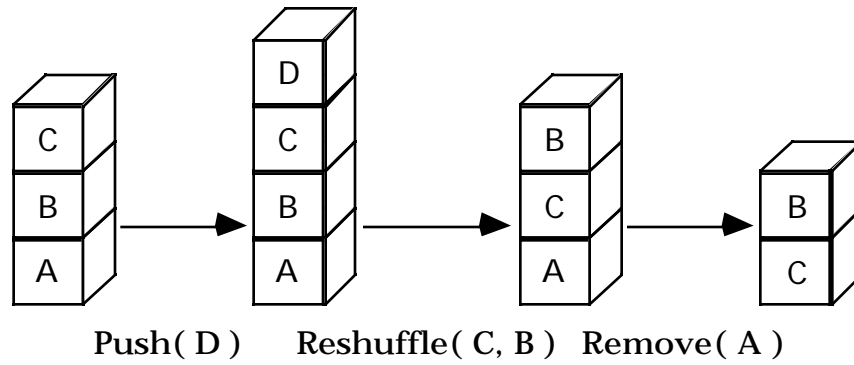


Figure 4: A sequence of typical operations on a stack of cubes and their results.

A stack of cubes can be “locked” into a *data set* (Figure 5) to reduce the user’s memory load and to facilitate better management of screen real estate. This operation is analogous to the desktop action of placing several files into a folder as opposed to merging them into one long file. A data set behaves like a cube, except that it cannot be included into another data set. In addition, a data set can be unlocked such that the constituent cubes appear as individuals.

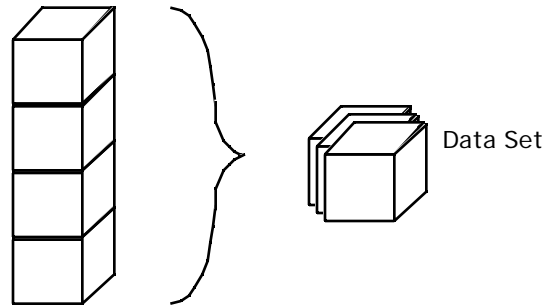


Figure 5: A stack of cubes can be 'locked' into a data set

3.2 Templates

A *template* is a rule set that describes how to render the data in a cube in the different types of views. Cubes are placed onto a template to apply the display specification to their content (Figure 6). A template is mandatory in order to view a cube. Without an appropriate template, a cube is merely raw data, unable to be explored by the user.

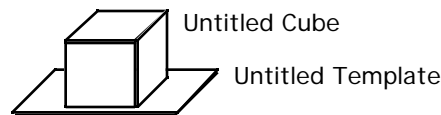


Figure 6: A cube stacked on a template

The kinds of templates applied to cubes are usually primarily graphical, though other types of templates could specify different views of the data, such as an alphanumeric table or a statistical chart (Figure 7). Rules for a graphical template may include what scale to set, what symbology to use, and how to resolve certain kinds of graphical conflicts. For example, a graphical roads

template may specify that roads in the associated cubes are to be drawn in red (color), as lines (object type), and at 0.4 mm (thickness); while a tabular roads template may specify listing state routes by order of increasing traffic use. These different kinds of templates can exist for a single data cube, enabling the data to take on these “multiple views.” Conversely, if a template specifies how to render a certain kind of layer (e.g., roads), it is applicable to any other cube of a different geographic extent that contains information about that same kind of layer.

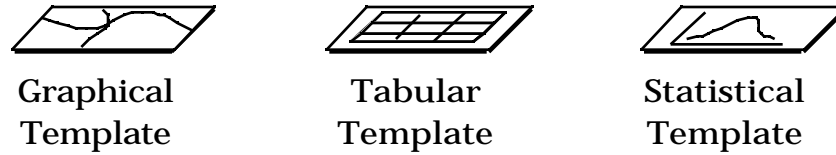


Figure 7: Templates to specify different views of data

In cases where no template is available for a certain kind of cube, a default template can be used to determine the initial rendering of these newly viewed cubes. For example, a default template might only contain the rule: “Draw all objects in black.” The default can be altered either by direct manipulation of the graphic symbols, tables, or statistical renderings, or in the abstract via dialog boxes. This is very similar to the use of paragraph settings in a word processor where default font, size and style are initially offered and changes can be made by making a selection and choosing, for example, a different font, or by accessing a dialog box to set the paragraph parameters in the abstract. These changes are recorded and applied in future instances where the rules are called upon for rendering.

Just as there are stacks of cubes, there are also stacks of templates as concise representations of combinations of templates. The major difference between a stack of templates and a stack of cubes is that the stacking order of templates affects the rendering of the data while the stacking order of cubes does not. Later we will see what effects this has on operations performed by the user.

Like a stack of cubes, a stack of templates can be “locked” into a *symbol set* (Figure 8). Symbol sets may contain only one kind of template, and not a variety of templates that specify multiple views. For example, there should not be a graphical template and a statistical template in the same symbol set; however, one could apply a graphical symbol set and a statistical symbol set to the same stack of cubes or data set. The concept of a symbol set allows users to create concise representations on the screen for such complex display specifications as a USGS 1:24,000 topo sheet.

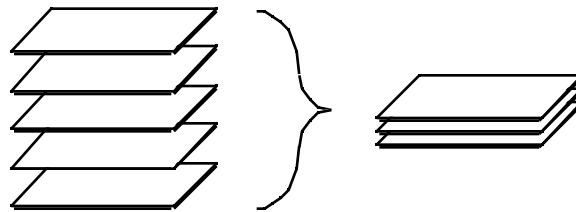


Figure 8: A stack of templates can be 'locked' into a symbol set

3.3 Viewing Platform

Initially, cubes and templates are located anywhere on the geographer’s desktop. Users select the cubes and templates of interest with a direct-manipulation device and combine them. When

manipulating objects on the geographer's desktop, users must have a way of distinguishing between actions intended for "house cleaning" (i.e., moving things around), and actions meant to initiate operations upon the objects. The *viewing platform* is the object that enables users to differentiate these two fundamentally different kinds of actions. It is associated with a set of viewing windows in which the multiple views of the data may be displayed concurrently. A viewing platform is a vital ingredient of the map-overlay metaphor as it represents the "light table" in the source domain. Users place cubes and templates onto the platform in order to see the contents of the cubes as rendered by the appropriate templates (Figure 9) or remove them from the platform to erase them from the current view.

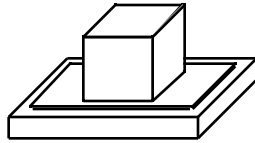


Figure 9: A cube and template stacked on a viewing platform

New viewing platforms can be created on demand so that users may simultaneously compare several different overlays. Each platform can have multiple "hot linked" windows that correspond to the multiple views of the data. In a hot linked window system, different views of the data are displayed concurrently in different windows. Direct manipulation queries can be performed in one window with the results being shown in all. For example, imagine a data set rendered graphically as a map and statistically as a scatter plot. A user may be curious about outliers in the scatter plot, and may select them via a standard direct manipulation technique. The objects selected are highlighted in both the scatter-plot window and the graphical map window, showing immediately if there is any geographic coincidence to the outliers in the statistical rendering. This gives users tremendous flexibility in exploring the data and seeing the results of those explorations, similar to other methods recently promoted in visual spatial analysis [35, 36].

4 Visualization and Feedback

A crucial step in the successful implementation of this user interface is the visualization of the concepts discussed thus far. This section will describe how to present cubes, templates, and platforms as natural and intuitive objects. These primitives are visualized as icons on the geographer's desktop, which embodies standard direct manipulation user interface guidelines. Typical direct manipulation techniques such as selection, dragging, and individual naming apply to all of the icons. The icons are dynamic since they change their appearances when they are in different states. As operations are performed by dragging cubes and templates around the geographer's desktop, users must encounter *feedback* so that they can determine if the operations are producing the desired results. Conceptually, one can clearly separate feedback that is normally associated with direct manipulation from feedback that results from the semantics of the language. *Direct-manipulation feedback* is governed by standard WYSIWYG interface design guidelines [5]. For example, when the user selects a group of cubes and templates those objects will highlight so the user understands that any actions performed will be executed on those highlighted objects. If the user then selects a menu, only those menu items whose operations could be applied to the selected objects at that time will be available. Alternatively, if the user decides to drag these objects, their grayed outlines will follow the mouse, indicating to the user that the objects are currently in the process of being moved. *Semantic feedback* gives for each operation an immediate effect about its outcome. For example, the attempt to stack two cubes over different geographic areas is rejected by pulling the cube off the platform, much like repelling magnetic forces. Combining direct-manipulation feedback and semantic feedback produces the object states necessary to keep users informed of what is happening at any given moment of interaction and ensuring that they have obtained the desired result.

4.1 Data Cube Visualization

A data cube is visualized as an icon representing a 3-dimensional cube and labeled with a unique cube name. Placing a cube on top of a template will initiate the rendering of the data in the cube as specified by the rules in the template. The cube's faces visualize the different states during the overlay, providing feedback to the user. A cube may be in two distinct states with respect to a corresponding template: it may be *viewable*, i.e., it is placed on the platform together with a template such that the template can render the cube; or the cube may be *unviewable*, i.e., it is either inactive (not placed on the platform) or on the platform, but without a corresponding template. We have selected to display these two different states through changes of the cube icon. A viewable cube displays a color icon on its face to indicate which view is currently available for the user to explore, while an unviewable cube icon reflects no information about the cube's display status. Orthogonal to these two view stages, a cube may be in different direct manipulation stages. These manipulation stages are (1) unselected, (2) selected, and (3) selected and being dragged. For these states, the usual Macintosh conventions of regular, highlighted, and outlined icons, respectively, have been adopted. Combining these orthogonal view stages gives all of the possible states that a cube icon can take on (Figure 10).

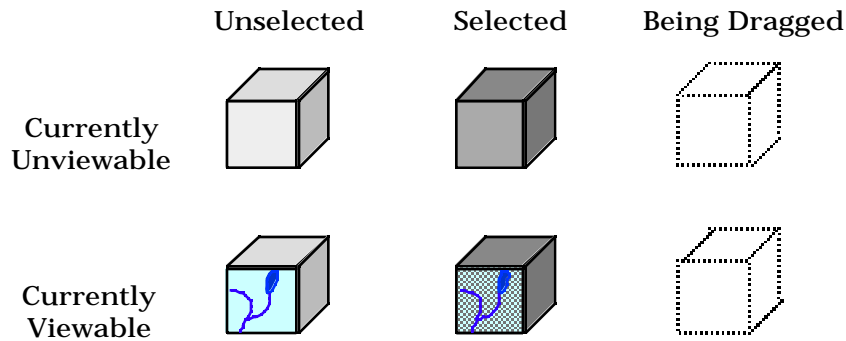


Figure 10: Visualization of the data cube in different stages of feedback

Prior to combining cubes and templates on the platform, templates that correspond to a particular cube may be identified by holding the SHIFT key down when selecting a cube or template. As long as the key is pressed down, all templates that could render that cube on the platform would also highlight. This operation works correspondingly to identify all cubes that apply to a particular template. If a template were selected with the SHIFT key, all cubes that could be rendered by that template would also be highlighted

As meaningful combinations of cubes and templates are stacked on the viewing platform, feedback is provided in two ways: (1) the results of operations are shown in the appropriate viewing windows, and (2) visual clues are given by the icons themselves. Initially, data cubes are gray in tone, indicating that the data exists, but that it is not viewable without the aid of a template. When a cube is dragged and placed on an appropriate template that specifies a particular rendering, a color icon on one of the cube faces will appear to tell users that they have performed a meaningful operation. If the template happened to be inappropriate for the particular cube, then the face of the cube will stay unchanged, i.e., gray. In this way, the user can see immediately which cubes are appearing in the view window and which need a template to be seen.

As the stack of cubes grows and is altered by users, it is automatically cleaned up after each manipulation. For example, when removing a cube, the gap in the stack is closed; when adding a

cube into the stack, space is inserted at locations to which the new cube was moved. Automatic performance of these tasks by the system frees users from having to worry about continually neatening up the geographer's desktop. It also ensures that the visualization presented to users is always consistent so that users do not have to bridge a mental gap between the changing style of how information is shown on the screen and their own concept of how the information should appear.

4.2 Data Set Visualization

Data sets are *aggregates* of *cubes*. We have selected a visualization of data sets that suggests a multitude of cubes, lined up one behind the other. Its similarity to the cube icon reflects that the users can execute the same operations they can do with cubes. On the other hand, data sets contain more than one cube, which is immediately observable from the icon. Like the cube, the faces of a data set also reflect its viewability, but with an additional stage necessary for partially viewable sets.

Users can make data sets by selecting several cubes and performing a "create data set" operation on the cubes. Once the data set is created, users can put additional cubes into the set by dragging them onto the set's icon so that it highlights (much like the trash can), and releasing the mouse button. Double clicking on the data set restores the cubes into the state in which they were previously.

Like data cubes, data sets can also be visualized in the different stages of direct manipulation. There is an additional stage of viewability, as a data set could be *partially viewable*. This would be true if there were fewer templates onto which the data set is stacked than are necessary to display all of its cubes. A data set that is partially viewable displays its corresponding face in gray. Combining the data set's two orthogonal view stages results in visualizations shown in (Figure 11).

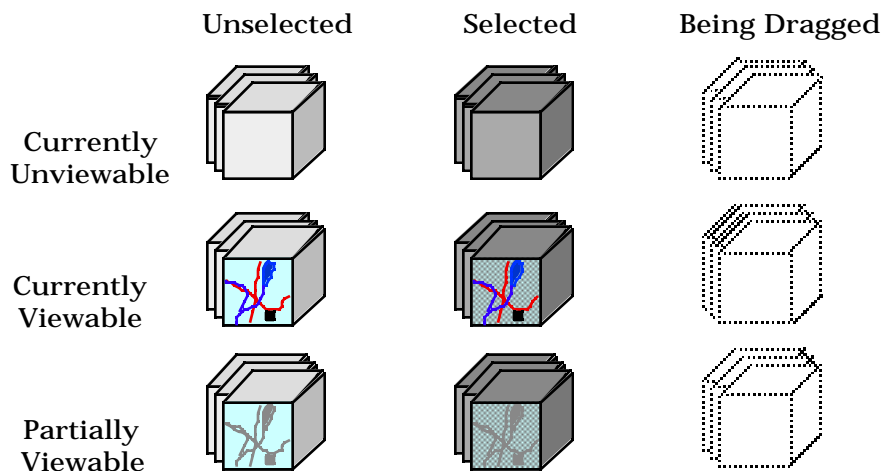


Figure 11: Visualization of the data set in different stages of feedback

4.3 Template Visualization

Templates are thought of as a rule set for rendering a certain kind of cube as a particular view. Hence, their icons must convey to the user what kind of data the template can render. Some examples of different template visualizations in the various direct manipulation stages are given in (Figure 12).

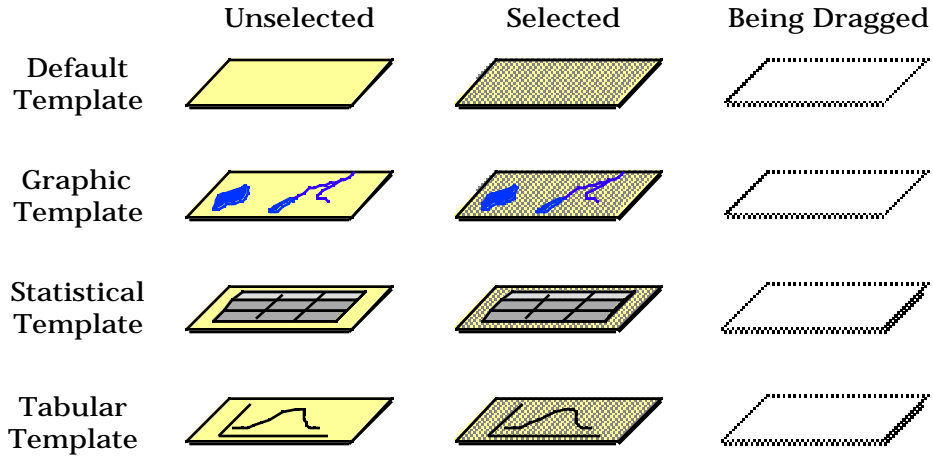


Figure 12: Visualization of the template in different stages of feedback

There can be additional information about which kind of cube the template is to render as well as what kind of view the rendering will specify. Thus, there are many different graphical template icons to indicate, for example, that a template rendering roads is different from a template rendering rivers (Figure 13). This is also true for certain other kinds of views, i.e., a statistical scatter-plot template might have a different icon than one specifying a bell curve.

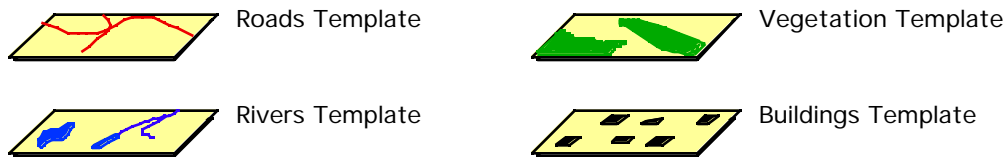


Figure 13: Visualization of different graphical templates indicates what kind of data cube each template can render

4.4 Symbol Set Visualization

Just as the data set icon indicates an aggregation of cubes, the visualization of the symbol set indicates that it represents several templates. There are, of course, indications about what kind of rendering the symbol set will specify, and these appear on the top face of the icon. Some examples of different symbol set visualizations in the various stages of direct manipulation are shown in (Figure 14).

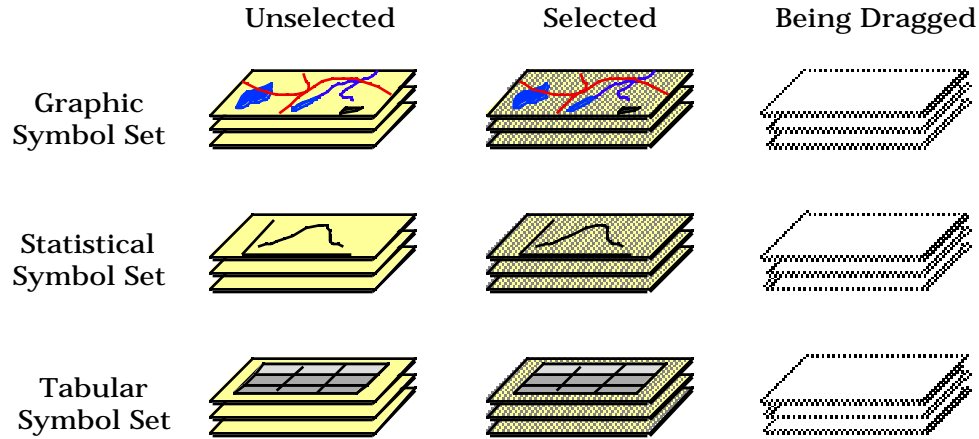


Figure 14: Visualization of the symbol set in different stages of feedback

4.5 Platform Visualization

Viewing platforms enter the usual direct manipulation stages and need to provide other (i.e., semantic) feedback to users as well. There is a need for the viewing platform to provide semantic feedback as a means to differentiate between meaningful operations and “house cleaning” actions. To reduce errors by users, it is necessary to provide this feedback *before* the intended action is performed.

During typical interaction, users can drag a group of objects and can drop them onto the viewing platform by releasing the mouse in the zone above it. The platform will attract the objects, similar to magnetic force, and stack them neatly on top of itself. Hence the platform must provide feedback to users telling them when the release of the mouse will result in the desired actions, i.e., before the mouse button is actually released. It does this by highlighting when the mouse moving the selected objects is in the zone above the platform. The feedback is followed by some induced activity. This active feedback is very similar to that of the trash can on the Macintosh desktop, which highlights when a file is dragged over it, absorbing that file when the mouse button is released.

The visualization of the platform in these different stages (Figure 15) may at first seem contradictory, because the two stages “not ready to accept objects being dragged” and “ready to accept objects being dragged” have some of the same icons. There are additional components in the interaction process, however, that prevent users from being confused by this. Namely, users know by sense of touch whether or not they are holding down the mouse button and dragging objects around on the screen. Thus, a user seeing the selected platform but not actively dragging objects on the screen should not be fooled into believing that the platform is in the “ready to accept objects being dragged” stage.

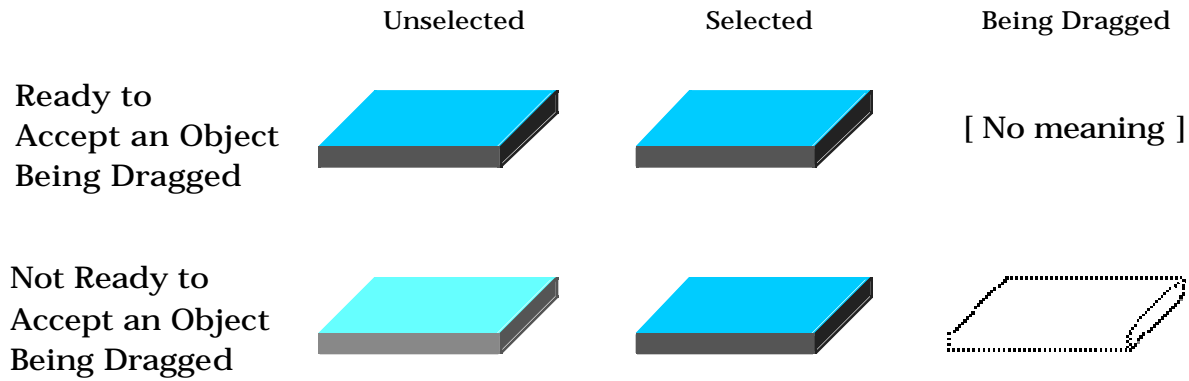


Figure 15: Visualization of the viewing platform in different stages of feedback

5 A Guided Tour Through an Overlay User Interface

Based on the design above, a map overlay user interface has been prototyped. The design process we followed began by first generating a number of “thumbnail sketches” of interface scenarios and proposed objects. It then became necessary to experiment with the actual media that would be used (i.e., the Macintosh screen), and a subsequent set of thumbnail sketches were drawn with the application Canvas . From these drawings of icons, algebraic specifications of the interface objects and their operations were written. Next, an animated mockup of the interface was created with Macromind Director . By viewing the animated mockup and comparing it with the algebraic specifications, we were able to determine how accurate the formalization was, and subsequent tweaking of both the mockup and the specifications made them compatible. Finally, a working prototype was developed in the visual, object-oriented programming language Prograph .

Our guided tour through a prototype overlay interface consists of five screen snapshots detailing various points during a simple interactive session. It is important to remember that this prototype is one visualization of the concepts discussed, and that there could be other successful (but different) visualizations as well. We present a scenario in which the user wants to explore some of the attributes associated with Maine’s ski areas and compare them with their geographic locations.

The user begins by launching the application and opening an *object window* (entitled “Data Window”), in which users directly manipulate the icons that represent the retrieval and presentation components of geographic information. The user then creates a viewing platform in the object window and loads into that window several cubes and templates. The user names the viewing platform “Maine Ski Areas”, and specifies that it be associated with it the two *view windows* on the right. The top window is where geographical rendering takes place, and therefore it has the platform’s name plus the suffix “.map” as its title. Similarly, the bottom window is for tabular rendering and it is appropriately titled “Maine Ski Areas.tab.” The three cubes loaded by the user include Maine Roads and Maine Boundaries, which are included to provide needed contextual information for a geographic frame of reference; and Maine Recreation, which contains information about the ski areas in Maine that the user is interested in. The four templates loaded by the user specify how to render the cubes in the view windows. Three graphical templates, Roads, Boundaries, and Recreation, determine how the three cubes are to be rendered in the map view window; while the tabular template, Recreation, specifies rendering

the Maine Recreation cube as an alphanumeric table in the tab view window. The screen as it appears to the user at this stage in the interaction is showing in (Figure 16).

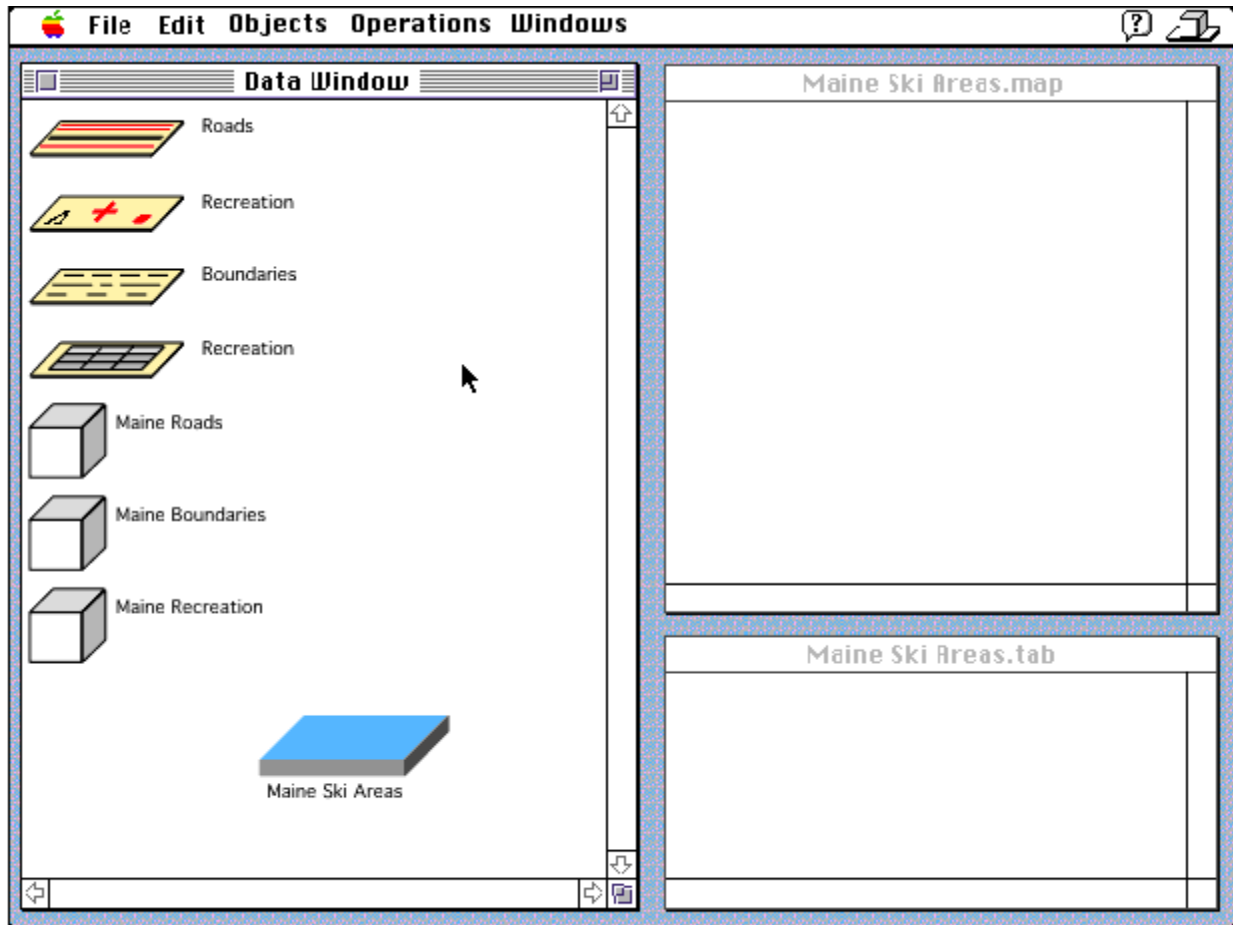


Figure 16

The user now wishes to see the geographic locations of ski areas in the state of Maine, and moves some cubes and templates onto the viewing platform. First the three graphic templates are moved, and then two of the cubes, Maine Boundaries, and Maine Recreation, are also added. When a user moves an object onto a viewing platform, an alias of that object appears on the screen in that position. This happens so that the system can efficiently handle multiple stacks of objects that utilize several of the same cubes and templates, without having to duplicate the information stored in each of the objects. This also prevents inconsistencies from arising when a user has several copies of the same information and needs to update. When the operation is complete, the user immediately receives feedback that it was successful, as is indicated by the colored graphic icons on the face of the two cubes and the rendering of the data in the map window as specified by the templates (Figure 17).

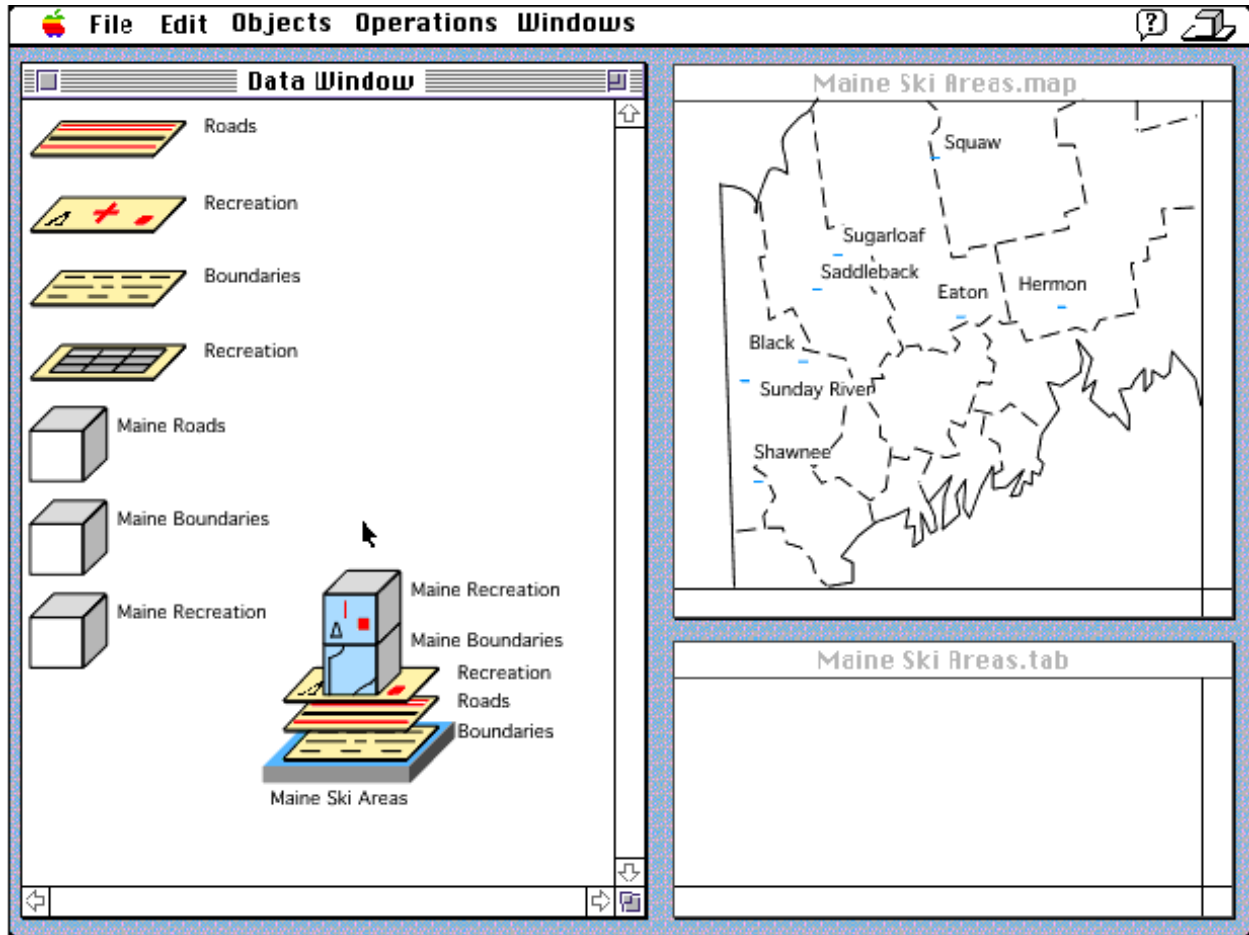


Figure 17

Satisfied with the success of the system thus far, the user decides to add the remaining cube and template to the platform. The user selects these two objects and drags them into a position where the platform can take them and stack them neatly on top of itself. As the objects are being dragged, the platform highlights when the cursor enters the zone directly above the platform (Figure 18), thus informing the user that releasing the mouse at this time will result in the desired action.

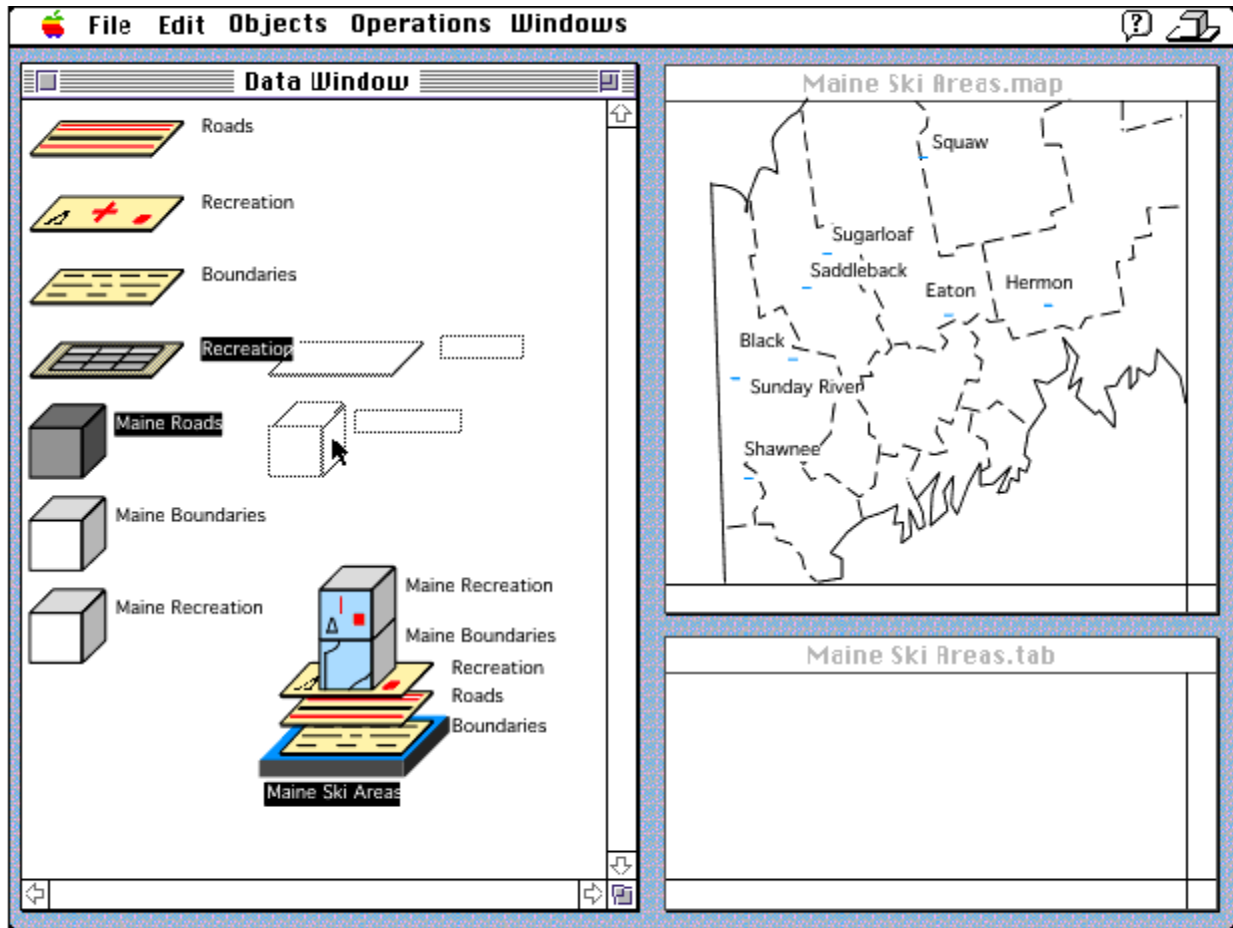


Figure 18

When the mouse is released the data in Maine Roads is added to the map view and the tabular view becomes active, displaying all of Maine's ski resorts and some of their attribute data. The Maine Recreation cube, located on the platform, indicates that this additional view of its data is available by showing a colored tabular icon on another side of the cube (Figure 19, left). Next, the user decides to readjust the order in which templates are stacked because it is affecting the readability of the map view. Because the order in which templates are stacked affects the rendering of the data—when the layers are drawn in the view window, the back to front drawing order of the layers follows the bottom to top stacking order of the templates—the user needs to reshuffle the templates so that boundaries will not be drawn over the recreation areas, which the user is most interested in. After making this adjustment, the user decides that s/he is only interested in the geographical location of ski areas with a vertical drop greater than 2,000 feet, and a quick scan through the list in the tab window reveals that there are only two. Curious to see if there is a geographical similarity in the locations of these two resorts, the user selects them in the tab window, and immediately the results of the selection are also shown in the map window (Figure 19, right). As a result of this selection, the user can clearly see that the two ski areas in question are indeed very close to one another.

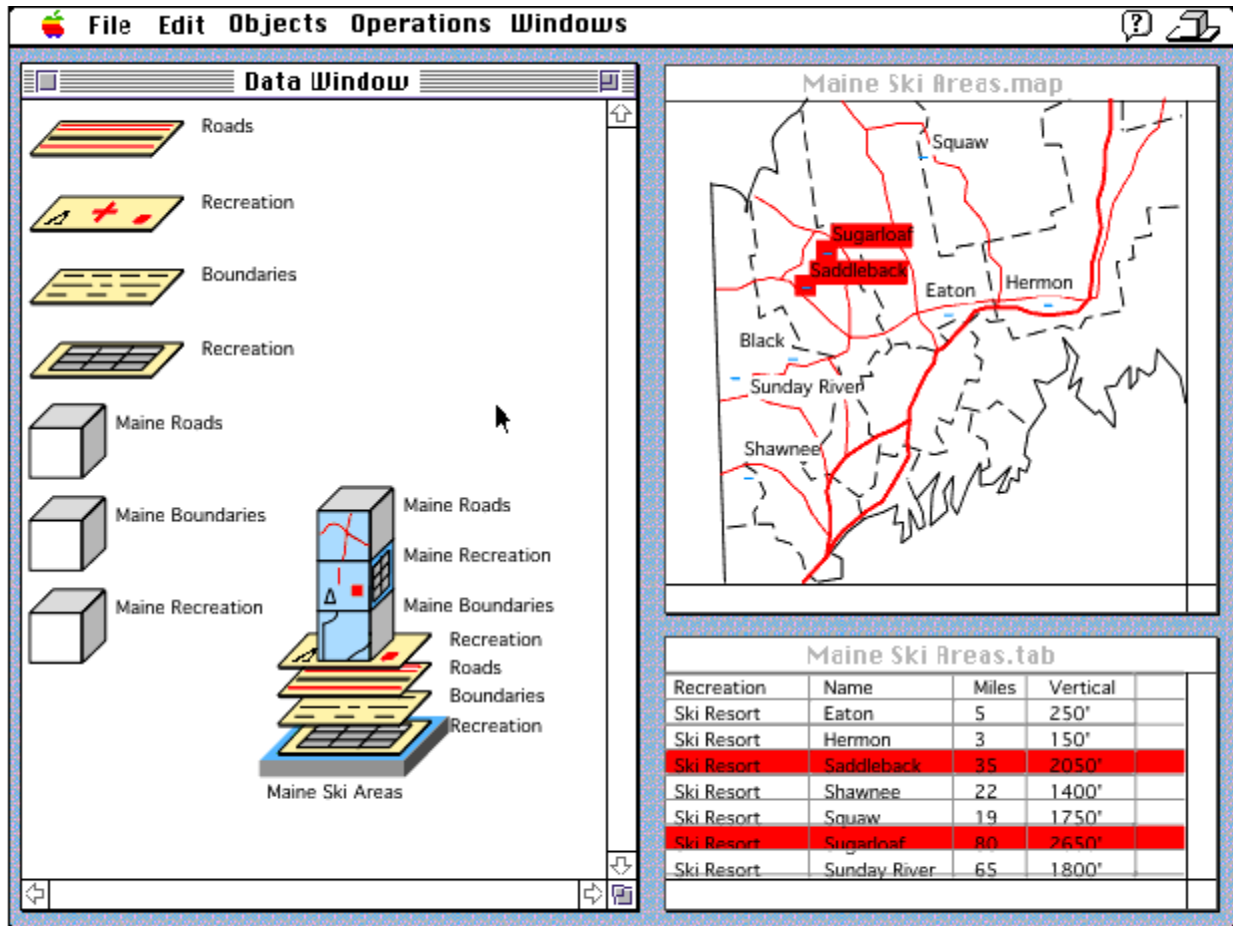


Figure 19

After some additional exploratory interaction, the user inadvertently removed the two Recreation templates from the viewing platform. A bit confused as to why the recreation data is no longer available in the view windows, the user needs only to glance at the stacks on the viewing platform (Figure 20) to discover what the problem is. It is obvious from the lack of colored icons on the face of the Maine Recreation cube that it currently unviewable, and adding the proper templates to the stack would surely solve the problem. The user then imagines that as a stack of cube grows in size this obvious visual feedback will become very valuable in making problems easier to identify.

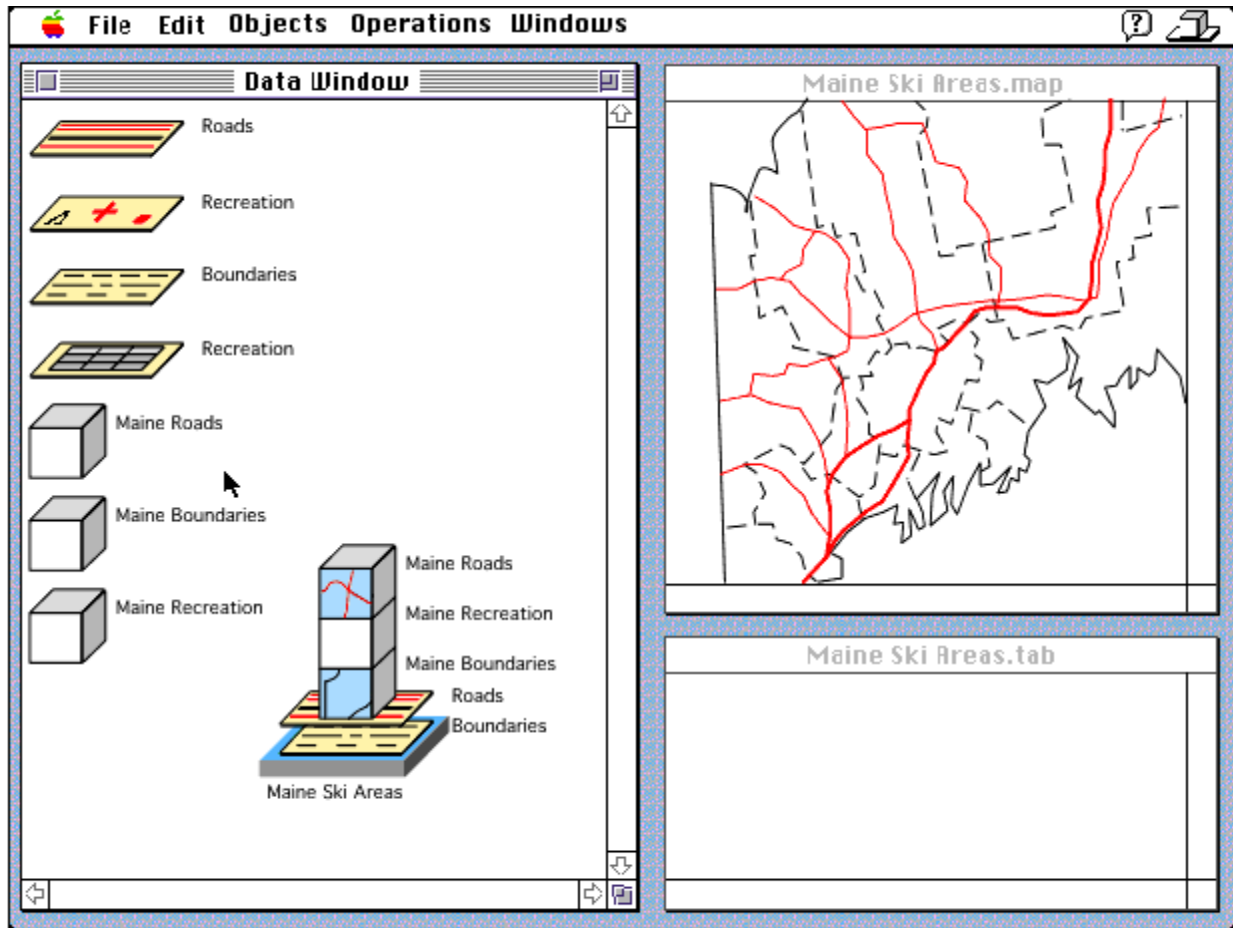


Figure 20

6 Conclusions and Future Work

We have presented the design of a visual language for access to geographic information based on the map-overlay metaphor. The major components of the language are the data cube, a representation for geographic data; the template describing their display parameters; and the viewing platform as the location where users combine cubes and templates to explore geographic data.

The visualization of the objects and their operations in terms of an iconic language was discussed and one particular implementation was presented. Appropriate feedback about the semantics of the language and the state of direct manipulation were identified as critical components of this user interface. Emphasis was placed on a user interface appearance that is consistent with the Macintosh desktop metaphor. The geographer's desktop is close to the user's conceptual model of map overlay as it is easy to evaluate—we continually tell the users what actions are possible—and easy to execute—we always tell the users if the system is in the desired state.

Five stages of an implementation have been completed: first, the objects and their operations were formally specified in terms of algebraic formalizations. Second, thumbnail sketches of the objects and some typical operations were drawn with paper and pen. Third, icons were designed, using a drawing program on a Macintosh. Fourth, a mockup of the geographer's desktop was implemented as a canned movie. This allowed us to receive feedback from potential users and to

identify design problems in the visualization and interaction, prior to making an actual implementation. Finally, a prototype of a subset of the geographer's desktop was implemented on a Macintosh using Prograph, simulating the GIS interface through data stored in files.

The geographer's desktop is a novel and promising user interface for geographic information systems. The ideas presented comprise only a framework, which needs extensions in order to become practical. A number of these issues are the subject of ongoing or future research:

- Currently, the only overlay operation offered is an OR applied to the cubes. Other computational operations, such as creating a buffer zone or adding and averaging several layers, must be incorporated into this simple framework of stacking cubes and templates.
- Extend the stacking of cubes to allow users to assemble cubes over disjoint areas.
- Design a language to describe and manipulate templates.
- Resolve conflicts about rendering data if two templates with contradicting rules for the same cube are applied.
- Apply other direct-manipulation techniques, such as pan and zoom, to the map viewing window.
- Link the user interface implementation with a GIS.

7 Acknowledgments

A number of our colleagues have contributed considerably to this project. Thanks to all of them. Particularly, Andrew Frank developed the original idea of "data cubes and maps" and Todd Rowell helped with the implementation of the prototype.

8 References

1. G. Lakoff & M. Johnson, *Metaphors We Live By* (1980), University of Chicago Press, Chicago.
2. J.M. Carroll & R.L. Mack (1985) Metaphor, computing systems, and active learning. *International Journal of Man-Machine Studies* **22**, 39-57.
3. W. Kuhn & A.U. Frank (1991) A Formalization of Metaphors and Image Schemas in User Interfaces. In *Cognitive and Linguistic Aspects of Geographic Space* (D.M. Mark & A.U. Frank, eds.) Kluwer Academic Publishers, Netherlands, pp. 419-434.
4. D.M. Mark (1992) Spatial Metaphors for Human-Computer Interaction. In: *Proceedings, 5th International Symposium on Spatial Data Handling*. Charleston, SC, pp 104-112.
5. Apple (1987) *Human Interface Guidelines: The Apple Desktop Interface* Addison-Wesley, Reading, MA.
6. D.C. Smith, C. Irby, R. Kimball, B. Verplank & E. Harslem (1982) Designing the Star User Interface. In *BYTE* **7**, 242-282.
7. M.J. Egenhofer (1990) Interaction with Geographic Information Systems via Spatial Queries. In: *Journal of Visual Languages and Computing* **1**, 389-413.
8. M.D. Gould & M. McGranaghan (1990) Metaphor in Geographic Information Systems. In: *4th International Symposium on Spatial Data Handling*, Zurich, Switzerland, July, pp. 542-550.
9. J.P. Jackson (1990) Developing an Effective Human Interface for Geographic Information Systems Using Metaphors. In *GIS/LIS Technical Papers.*, Denver, Colorado, March, pp. 117-125.
10. ESRI (1991) *Exploring Your World Just Got a Whole Lot Easier* Environmental Systems Research Institute, Inc., Redlands, CA.

11. M. Ehlers, G. Edwards & Y. Bédard (1989) Integration of Remote Sensing with Geographic Information Systems: A Necessary Evolution. *Photogrammetric Engineering and Remote Sensing* **55**, 1619-1627.
12. A.U. Frank (1982) MAPQUERY: Database Query Language for Retrieval of Geometric Data and its Graphical Representation. In: *ACM SIGGRAPH* **16** (3), pp. 199-207.
13. E. Clementini & P.D. Felice (1992) Towards an Interaction Level for Object-Oriented Geographic Database Systems. In: *Proceedings: 20th Annual Computer Science Conference*. Kansas City, Missouri, March, pp. 33-40.
14. A.U. Frank (1991) Beyond Query Languages for Geographic Databases: Data Cubes and Maps. In: *Proceedings: International Workshop on DBMS for Geographic Applications*. Capri, Italy, May, pp. 74-86.
15. M.J. Egenhofer (1991) Extending SQL for Cartographic Display. *Cartography and Geographic Information Systems* **18**, 230-245.
16. D.M. Mark & M. D. Gould (1991) Interacting with Geographic Information: A Commentary. *Photogrammetric Engineering and Remote Sensing* **57**, 1427-1430.
17. K.C. Kirby & M. Pazner (1990) Graphic Map Algebra. In: *4th International Symposium on Spatial Data Handling*. Zurich, Switzerland, July, pp. 413-422.
18. D. Lantner & R. Essenger (1991) User-Centered Graphical User Interface Design for GIS. In: Technical Report 91-6, National Center for Geographic Information and Analysis.
19. J. Tyrwhitt (1950) Surveys in Planning. In: *Town and Country Planning Textbook* (edited by the Association for Planning and Regional Reconstruction), The Architectural Press, pp. 146-178.
20. K.K.L. Chan & D. White (1987) Map Algebra: An Object Oriented Implementation. In: *International Geographic Information Systems (IGIS) Symposium: The Research Agenda*. Arlington, Virginia, November, pp. 127-150.
21. M.P. Goldberg, A. H. Schmidt & N. R. Chrisman (1979) Integration and Analysis of Multiple Geographic Data Bases: An Application of ODYSSEY. In: *Mapping Software and Cartographic Data Bases*. Harvard Laboratory for Computer Graphics and Spatial Analysis, Cambridge, MA, pp. 81-97.
22. D.J. Peuquet & D.F. Marble (1990) ARC/INFO: An Example of a Contemporary Geographic Information System. In: *Introductory Readings in Geographic Information Systems* (D.J. Peuquet & D.F. Marble, eds.) Taylor & Francis, London, England, pp. 90-99.
23. J. Dangermond (1983) A Classification of Software Components Commonly Used in Geographic Information Systems. In: *Design and Implementation of Computer-Based Geographic Information Systems* (D. Peuquet & D.F. Marble, eds.) Taylor & Francis, London, England, pp. 30-51.
24. C.D. Tomlin (1983) A Map Algebra. In *Proceedings of the 1983 Harvard Computer Graphics Conference*. Harvard Laboratory for Computer Graphics and Spatial Analysis, Cambridge, MA.
25. J. Bertin (1983) *Seminology of Graphics* The University of Wisconsin Press, Madison, WI.
26. T. Smith & A.U. Frank (1990) Very Large Spatial Databases: Report from the Specialist Meeting. *Journal of Visual Languages and Computing* **1**, 291-309.
27. ESRI (1991) *Understanding GIS, The ARC/INFO Method*. Environmental Systems Research Institute, Inc., Redlands, CA.
28. M. Pazner, K.C. Kirby & N. Thies (1989) *MAP II: Map Processor - A Geographic Information System for the Macintosh*. Wiley & Sons, New York.
29. D.A. Norman (1988) *The Design of Everyday Things* Doubleday, New York.
30. E. Codd (1970) A Relational Model for Large Shared Data Banks. In: *Communications of the ACM* **13**, 377-387.

31. R. Gutting (1988) Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In: *EBDT '88 - International Conference on Extending Database Technology* Venice, Italy.
32. W.J. Bowman (1968) *Graphic Communication*. Wiley & Sons, New York.
33. T.R. Henry & S. E. Hudson (1990) Multidimensional Icons. *ACM Transactions on Graphics* **9**, 133-137.
34. O.J. Dahl & K. Nygaard (1966) SIMULA - An Algol-based Simulation Language. *Communications of the ACM* **9**, 671-678.
35. J. Haslett, G. Wills & A. Unwin (1990) SPIDER - an interactive statistical tool for the analysis of spatially distributed data. *International Journal of Geographic Information Systems* **4**, 285-296.
36. W. Stuetzle & A. Buja (1991) *Visualization of Quantitative Data* University of Washington Instructional Media Services, Seattle, WA, Video.