

# Visual Map Algebra: a direct-manipulation user interface for GIS

*M. J. Egenhofer and H. T. Bruns*

*National Center for Geographic Information and Analysis  
and*

*Department of Spatial Information Science and Engineering  
5711 Boardman Hall, University of Maine, Orono, ME 04469-5711,  
U.S.A., {max, tomb}@mcan1.maine.edu*

## **Abstract**

Geographic Information Systems (GISs) store, analyze, and present spatial data and information about geographic space and geographic phenomena. Virtually all aspects of a GIS have inherent spatial, graphical, and visual characteristics. While the database and analytical aspects of GIS have enjoyed considerable advancement in recent areas, a user's access to and interaction with spatial information has not. For such a highly visual system, GIS is often characterized by its distinctly non-visual user interfaces, where command-line and window-icon-menu-pointer (WIMP) user interfaces are most common, whereas visual, direct-manipulation user interfaces are rare. Direct-manipulation user interfaces based on metaphor offer increased usability for GIS. This paper extends the Geographer's Desktop, an innovative direct-manipulation environment for viewing data in a GIS, by integrating a new method for GIS Map Algebra operations. Used by planners, geographers, and other spatial scientists, Map Algebra facilitates the analysis of geographic phenomena. Historically, Map Algebra was performed manually by overlaying thematic map layers, a process that offers a rich source domain for user interface metaphors. *Visual Map Algebra* is a direct-manipulation query language that allows users to construct arbitrarily complex combinations of map layers by stacking iconic representations of thematic map layers onto an interface object called the computational platform. Users visualize such calculated map layers by moving them onto an interface object called the viewing platform that manages cartographic display parameters and is associated with a viewing window. Visual Map Algebra enables exploratory analysis by changing parameters of the overlay and immediately observing the outcome, and adding or removing map layers on the fly.

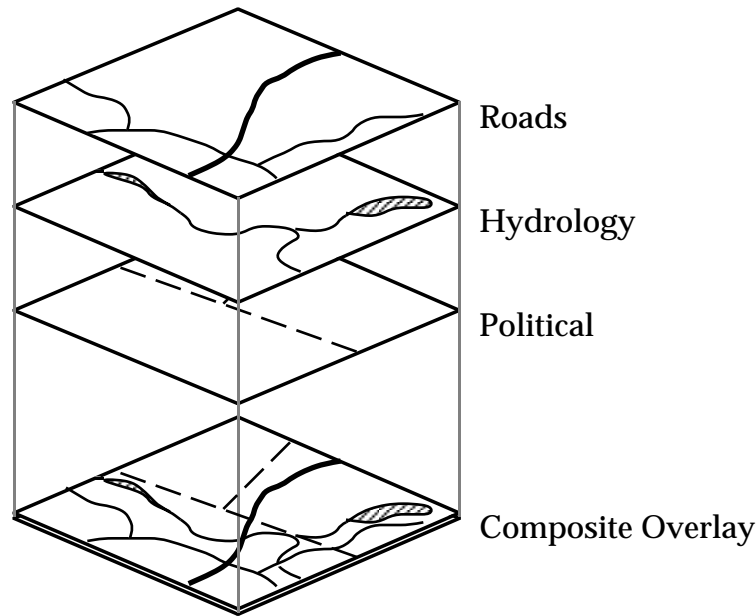
# 1 INTRODUCTION

Geographic Information Systems (GISs) store, analyze, and present spatial data and information about geographic space and geographic phenomena. GIS data are inherently visual, and graphical depictions of data's location—symbolized by some characteristic attribute—produce a powerful visual representation. GISs process spatial relations between spatial objects. Often, these relations are depicted visually as a fundamental and spanning set of spatial relations. While the data and processing of GIS have exhibited advanced visual characteristics, the access to these features has remained distinctly non-visual. The user interfaces for advanced GISs have so far been largely textual, and henceforth difficult to use. This paper presents the results of research to enhance the visual nature of GIS user interfaces, through the use of metaphor-based graphics and direct-manipulation. It focuses the design of a direct manipulation query language for GIS Map Algebra. This user interface was developed with an understanding of issues in two different domains: geographic information systems and human-computer interaction (HCI). GIS seeks to model spatial phenomena on a computer, whereas HCI seeks to model human behavior with a computer. A symbiosis of these disciplines brings together the necessary tools for effectively developing visual user interfaces for visual, spatial database systems. The direction here is primarily to bring HCI techniques to bear on GIS, in an effort to improve its usability. The reverse direction is conceivable, however. An understanding of how humans think and communicate about spatial phenomena is basic to human cognition, and may indeed have an impact on HCI.

Various aspects of the world can be captured in a GIS to create a model of reality. Some models consider space full of discernible objects, and measure their attributes and relations (Mark and Frank 1989). Others represent space by the continuous measurement of several different properties, or themes, over the same area (Goodchild 1992). The choice of themes and how they are combined is highly dependent on the purpose of the analysis and modeling. By treating each theme as a variable in an algebra, Tomlin (1983, 1990) has formalized the combination of themes in a Map Algebra. These combinations reveal information that was unavailable in any component theme. Spatial scientists use Map Algebra to explore relationships in the data, and gain an understanding of processes that exist in the physical world.

The techniques of Map Algebra are rooted in the history of GIS—for an overview see Chan and White (1987). The single most important concept is the *overlay* of thematic map layers. Computerized map overlay evolved from manual methods of thematic map overlay (Figure 1). Historically, map overlay was a strictly visual process, with limited analytical capabilities. Map Algebra is a completely generalized form of analytical map overlay, made possible by GIS technology. Layers of thematic data over the same geographic area are combined by some analytical operation, yielding a new thematic layer with integrated information. The analytical operations can be as simple as superimposing one layer over another, or any complex mapping expressed as a function or truth table over the values in the layers (Dorenbeck and Egenhofer, 1991).

Today, thematic map overlay in a GIS is a useful tool for market researchers, planners, social scientists, geologists, geographers, and other “spatially-aware” professionals; however, the true power of GIS Map Algebra is not fully exploited, because the implementations of GIS Map Algebra have not attained a level of usability to make the technology truly available to all that would benefit from its use.



**Figure 1** Thematic map overlay.

While window-icon-menu-pointer (WIMP) and flowchart user interfaces for Map Algebra do employ graphical displays, they do so only in a very limited sense. The visual characteristics of the user interface are not fully exploited, because the interaction has been restricted to manipulating user interface widgets—primarily menus and dialog boxes—to create command lines. To improve the visual interaction with spatial data, it is necessary to exploit the spatial characteristics of a user interface:

- The location of an interface object can be made as significant as its appearance.
- How the user articulates the pointing devices can have as much meaning as to what the user points.
- Tapping in on the semantics of movement and location can free the user from the difficult task of constructing command lines.

This paper presents the design of a *Visual Map Algebra*, which is high-level GIS query language that follows these guidelines. Visual Map Algebra minimizes not only the input from a keyboard, but also reduces the use of menu selections and dialog boxes, making it a truly direct-manipulation user interface. Aspects relating to the implementation of such a system, e.g., by translating Map Algebra expressions into an extended SQL syntax (Egenhofer 1991, 1994) have been discussed elsewhere. Also earlier work provides more detail about the limitations of traditional database query languages for geographic applications (Egenhofer and Kuhn 1991) and the motivation for this work (Frank 1992).

The remainder of this paper continues with a survey of different implementations of user interfaces for Map Algebra. Section 3 discusses the Geographer's Desktop as a user interface framework for interaction with geographic data. Section 4 describes the source metaphor for Visual Map Algebra and Section 5 introduces the computational platform as the user interface object where one performs Visual Map Algebra. Section 6 shows how the computational

platform is integrated into the Geographer's Desktop to allow a user to perform Visual Map Algebra operations, and in Section 7 we draw our conclusions and discuss future work.

## 2 IMPLEMENTATIONS OF MAP ALGEBRA

There is a complex relationship between analytical software, the user, and the user's ability to create complex models, solve problems, and make decisions (Turk 1993). Users possess certain task knowledge, spatial abilities, and problem-solving skills that are brought to bear in the user's work environment (Nyerges 1993). The role of the user interface is to provide a structure consistent with this environment. User interfaces for GIS Map Algebra generally fall into the following categories: command-line user interfaces; WIMP user interfaces; and graphical user interfaces.

### 2.1 Command-line user interfaces for Map Algebra

With a command-line user interface, the user types in Map Algebra expressions, one at a time, at a command prompt. The syntax of the commands is highly significant, and the order of the commands is also sometimes important. Table 1 shows examples of a Map Algebra operation to create a 50 unit-wide buffer around streams in various Map Algebra command languages. Arc/Info is a full-featured, command line GIS (ESRI 1990). The software has literally thousands of commands, from which a user must select in order to find those that will fulfill any given task. MGE Grid Analysis (Intergraph 1993), Arc/Info GRID (ESRI 1992), and OSU-MAP (Sandhu *et al.* 1987) are more direct implementations of Tomlin's Map Algebra.

**Table 1** Creating a buffer around streams in four different Map Algebra implementations

<i>Map Algebra command</i>	<i>GIS</i>
BUFFER streams streams_buf # # 50 # LINE	Arc/Info
streams_buf = EXPAND (streams, 50, LIST, 1)	Arc/Info GRID
streams_buf = FOCALPROXIMITY OF streams BY 50	MGE Grid Analyst
BUFFER streams BY 50 FOR streams_buf	OSU-MAP

Command-line systems proliferated before graphical user interfaces were available. Their syntax can be formally defined, making command interpreters and compilers relatively easy to create. Developing macros or programs is easily accomplished by grouping commands and control statements in files. However, command-line user interfaces have very poor visual and physical characteristics. Users interact only with text, which has several shortcomings: a screen full of text has a density of information that is too high for most users; typing is an interaction that is tiring and error-prone; and command-line user interfaces force users to be aware of command names and syntax. The number of commands is so great at times that a user spends more time thinking about command tools to use than thinking about the task at hand (Gould 1993).

## 2.2 WIMP user interfaces for Map Algebra

Window user interfaces for Map Algebra offer users a variety of menus and forms with which to interact. By selecting items from menus, clicking buttons and icons, and by filling in fields in a form, a user constructs a Map Algebra expression. MGE Grid Analyst (MGGA), Intergraph's Map Algebra module, offers the user a form that is filled out to construct a Map Algebra statement (Intergraph 1993), as does Map II, a GIS for the Apple Macintosh (Pazner *et al.* 1989). With each, the user interacts with fields to select the desired functions and data, and the system generates a Map Algebra expression in another field of the form.

WIMP user interfaces have a much stronger visual component than command-line user interfaces. By viewing menus, buttons, and icons, a user is made aware of possible choices. Pointing to selections replaces remembering and typing commands. The bit-mapped graphics alone contribute to the creation of a more pleasant environment for users. While WIMP user interfaces do reduce the amount of memorization of command names a user must do, the process of mapping a task to a suite of commands remains. The user interfaces mentioned above replace typing with button pushing. Disabling certain controls at certain times can prevent the user from generating syntax errors. But command lines, and their syntax, still exist. For GIS Map Algebra, WIMP user interfaces offer little significant improvement over command-line user interfaces.

## 2.3 Graphical user interfaces for Map Algebra

For GIS Map Algebra, graphical user interfaces based on flowcharts are the most prevalent. Flowcharts have long been an aid in programming, and provide a means to logically structure a task in a format close to a user's mental representation (Myers 1990). Graphical flowcharts exceed command-languages in exploiting the human vision process, which is optimized for multi-dimensional data. Flowchart user interfaces have much richer visual and physical characteristics than WIMP user interfaces.

Flowcharts have been used by GIS researchers to illustrate the structure of an environmental model (Tomlin and Berry 1979; Berry 1987; Laurini and Thompson 1992; Berry 1993). Lanter and Essinger (1991) developed a flowchart user interface to run on top of Arc/Info, with which GIS data is represented by icons, and functions by lines connecting the icons. Kirby and Pazner's (1990) user interface of MAP II has both data and functions represented by icons, connected by graphical "pipes." The Erdas Imagine Model Maker (ERDAS 1993) allows users to manipulate icons for data and functions for structuring image processing operations. However, the focus of these systems is on the visualization of flowchart user interfaces, not on the interactions that underlie their creation. The placement of icons has no semantic significance to the model being constructed. These flowchart user interfaces do not eliminate the need for the user to be aware of and conform to certain command line syntax. The visual and physical characteristics inherent to GIS Map Algebra may be used to create more appropriate user interfaces.

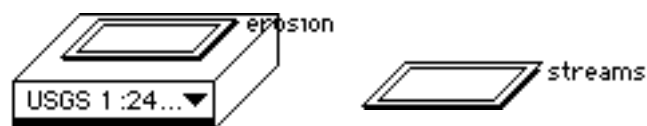
## 3 THE GEOGRAPHER'S DESKTOP

Direct manipulation is an appropriate interaction style for GIS Map Algebra. Historically, the process of map overlay was a visual, tactile operation. Direct manipulation, accompanied with a visual, graphical representation of data, fosters exploratory data analysis. Exploration creates a

dynamic, absorbing, and satisfying task environment. Users will become less aware of the existence of the user interface while becoming engrossed in their tasks. Direct manipulation data analysis is desirable for spatially-aware scientists performing Map Algebra in a GIS. Direct manipulation affords an empowering environment for spatially-aware scientists performing Map Algebra in a GIS.

Direct manipulation is a metaphor for touching and manipulating objects in our environment. This form of human-computer interaction became popular with full-screen text editors, like EMACS, and spreadsheets, like VISICALC, with which users could move directly to the text they wanted to edit and effect changes. Mapping movement or gesture to intent is very difficult. Mapping placement to command semantics, in a visual user interface, has more immediate, viable realizations.

The *Geographer's Desktop* (Egenhofer and Richards 1993b) is a visual, direct manipulation environment for interacting with spatial databases and viewing their content. Two fundamental, and related, metaphors form the basis of this user interface (1) thematic map layers are represented by map layer icons, providing a visual link with the source domain of the user's task and (2) the stacking of map layers is metaphorically mapped to direct manipulation in the user interface. The user sees and stacks map layer icons to express visualization and analysis functions. This is a natural mapping from the user's domain, where mylar map sheets are physically stacked on a light table to construct views. Manipulation of map layers is complemented by the *viewing platform* (Egenhofer and Richards 1993a), which is a metaphor for a light table (Figure 2). It facilitates GIS database query and visualization operations. To visualize GIS data, users place data layer icons onto the viewing platform. The generation of a map view is very visual, and the placement of icons has semantic significance. The result is a tightly coupled association between the execution and evaluation of a map view and database query.



**Figure 2** The Viewing Platform with a data layer on top, and another one off the platform.

### 3.1 Thematic map layer visualization

The icons are visualized as a perspective view of a map sheet. This perspective view gives the icons a 3D appearance, resembling actual map sheets. This metaphor is strengthened when several such icons coexist, and are stacked on top of each other (Figure 3). The visualization of the icons includes shadows, to enhance their 3D appearance.

Graphics can be placed on the icons to allude to the theme of the layer (Egenhofer and Richards 1993a). The design of the graphics depends on its contents. For some layers the graphics can be established by the system, but after some analysis, they would no longer be appropriate. It is also possible to put a graphic representation of the layer's data type on the icon (Lanter 1991). It is likely that a visualization of both theme and data type would be of use, helping the user select data layers for both viewing and analysis. Egenhofer and Richards

(1993b) divide the face of the icons into two parts, data area and map legend, to access respectively the database query and visualization parameters for each theme.



**Figure 3** Stacked map layers.

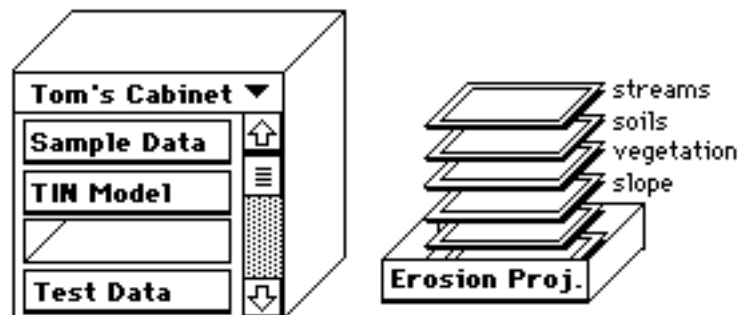
### 3.2 Thematic map layer interaction

Data layer icons respond to standard point-and-click, drag-and-drop interactions. The user places a data layer icon on any interface object, or on the data stack. The visualization of the layer icons changes in response to a user's manipulations. Layers highlight when selected with the mouse. An outline or shadow of the layer follows the cursor when it is being moved. If there is an error or an ill-defined function, the appearance of the icon changes to alert the user.

### 3.3 Thematic map layer organization

Data layers can also be grouped into classes by sharing some visual characteristic, such as color or texture, or grouped spatially in different stacks on the Geographer's Desktop. Employing graphical enhancements generally increases the size of the icons, and the separation between stacked icons. The appearance of stacked icons is crucial for maintaining the map overlay metaphor of stacked thematic data maps. If the layer icons must be vertically separated to reveal added graphics, the effect of this metaphor will diminish. In the visualizations presented here, no such graphics are used, and the data layers are distinguished only by their names.

Layers can be associated in another way, by project, and stored in a map cabinet (Figure 4).



**Figure 4** Map filing cabinet, with open drawer.

The user can select from the many drawers of the cabinet, opening by clicking on it with the mouse, and pulling it out. Once opened, the layers associated with that drawer appear stacked in the drawer and can be manipulated. Several drawers could be open at once, and layers can be moved between them, or multiple copies could be stored in different project drawers.

#### 4. VISUAL MAP ALGEBRA ON THE GEOGRAPHER'S DESKTOP

The viewing platform is a highly usable user interface for visualizing thematic map layers and combining them in a particular way. Egenhofer and Richards (1993a) chose *superimposition* as the operation to combine those layers that are stacked up on the viewing platform, i.e., the top layer overrides the value of the non-empty value in the bottom layer (Table 2).

**Table 2** The truth table for superimposition of layers *A* and *B* (*a* and *b* are values in the respective layers *A* and *B*)

<i>A</i>	<i>B</i>	<i>A on top of B</i>	<i>B on top of A</i>
empty	empty	empty	empty
a	empty	a	a
empty	b	b	b
a	b	a	b

While this combination is sufficient to do simple spatial analysis, it is far from the power Map Algebra provides with a much larger variety of analytical operations to combine map layers. For example, one may want to combine layers such that the result contains their intersection, determine the Voronoi regions, i.e., the neighborhoods around points in a layer, or create buffer zones (Table 3).

**Table 3** The truth table for the intersection of layers *A* and *B* (*a* and *b* are respective values in layers *A* and *B*; *c* is the value in the resulting layer) and the formation of a 50 ft buffer around the intersection of the two layers.

<i>A</i>	<i>B</i>	<i>A intersects B</i>	<i>50 ft-buffer (A intersects B)</i>
empty	empty	empty	} c if cell is less than 50 ft from a non-empty cell; otherwise empty
a	empty	empty	
empty	b	empty	
a	b	c	c

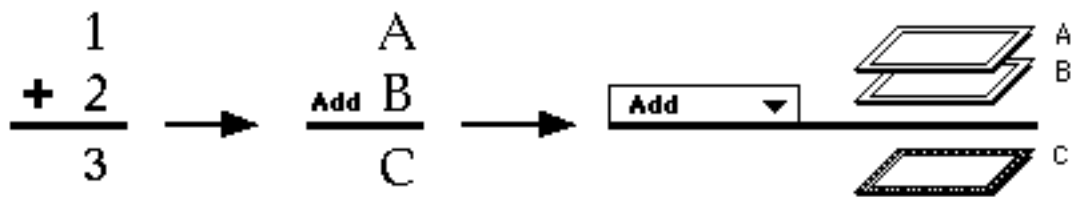
While both employ the map overlay metaphor, and direct manipulation of map layer icons, the differences between these operations make it awkward to use the viewing platform for computational overlay. It might be considered also to use the viewing platform for the selection

of the analytical operation to combine the layers; however, such a choice would have some serious drawbacks, because the viewing platform would now incorporate an additional functionality. Users may be confused as to what to choose—display parameters or computation formula. More importantly however, is that not every computational overlay results in something a user wants to be displayed graphically. Frequently, intermediate computational results may be constructed which in turn are used as input for the next calculation. With a viewing platform that handles the specification of the analytical formula as well as the display parameters, it would become difficult to implement intuitive mechanisms to use the result of one map overlay operation in another map overlay. To resolve these conflicts, it is necessary to introduce another concept into the Geographer’s Desktop in the form of a dedicated interface object where computational map overlay will be performed.

The design goal of this extension is (1) to perform the task of computational map overlay by using some computation metaphor and (2) to integrate with the existing concepts and structure of the Geographer’s Desktop, particularly the viewing platform, allowing users to exploit previously learned behavior, and allow users to elicit knowledge from the their task domain.

Three computation metaphors have been considered: (1) horizontal equations, (2) flowcharts, and (3) vertical equations. *Horizontal equations*, when read from left to right, are sentences in an abstract, mathematical language. The horizontal equation metaphor can be implemented directly with a command-line user interface when the equation language is completely defined. A language including GIS Map Algebra functions, then, creates a command-line user interface for GIS Map Algebra. Such a user interface, however, does not use visual map layers, or the visual and physical aspects of the map overlay metaphor. *Flowcharts* are a more visual computation metaphor, but they also do not employ the map overlay metaphor. Instead, they use a linking metaphor to establish connections between data and functions. A computation metaphor with which the link between data and function is accomplished with stacking is desirable.

A *vertical equation* generates expressions with operands in an arrangement similar to the stacking of layers onto the viewing platform. Making those operands thematic map layers yields a computation metaphor employing the map overlay metaphor. This is called the *addition line*, since the structure of vertical equations has its roots in the mathematical tool typically used to teach children addition, or used by adults when adding a column of numbers. This structure metaphorically evolves into the interface object for GIS Map Algebra, called the *computational platform* (Figure 5).



**Figure 5** Evolution from addition line for numbers to computational platform for layers.

The operands are placed above a line that separates the result of the computation. The operation also appears above this line, to the left of the operands. The line that separates the “inputs” from the “output” is called the platform because the inputs are stacked on top. The

platform can be thought to physically support the inputs. This is an application of the SURFACE image schema, a basic cognitive structure (Johnson 1987), where inputs are either on or off the platform. The strength of the metaphor lies in the fact that this is the same image schema used in the physical process of map overlay, as well in the viewing platform. The computational platform evolves from replacing the fixed numbers and symbols to a more generalized form, with variable names and a named operation. Finally, replacing the variable names with named icons representing thematic data layers, and by enclosing the operation within a pop-up menu, forms the final visualization of the computational platform.

## 5. THE COMPUTATIONAL PLATFORM

The interaction with the computational platform is consistent with the direct manipulation environment of the Geographer's Desktop. To create a complete Map Algebra expression requires several interactions: (1) creating a computational platform, (2) selecting a platform function, (3) choosing the layers to be combined, and if necessary (4) setting the parameters of the operation. To formulate a complete Map Algebra model will likely require the creation of several computational platforms, which can coexist on the Geographer's Desktop, be aggregated into a single platform, or be abstracted and represented by a new data layer icon.

### 5.1 Creating a computational platform

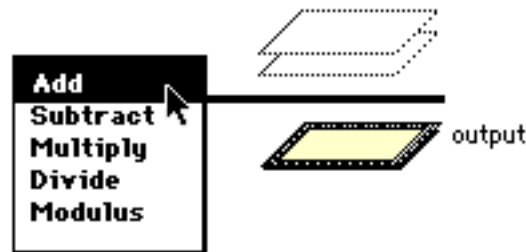
A new, empty computational platform consists of the horizontal line onto which layers will be stacked, and a pop-up menu for the selection of the platform parameters. To provide the user with guidance as to what to do with platform and layers, the computational platform features templates for input layers. These input slots have a dual purpose as they provide feedback to the user on where to put input layers and how many are needed, and they allow the user to specify, for multiple input functions, which layer is associated with which variable. Only for commutative operations, such as add and multiply, the order of the inputs on the platform can be ignored.

There are many ways to create a computational platform on the Geographer's Desktop. A platform can be created by selecting its icon from an interface tool box, or by activating a **Create Computational Platform** operation from a pull-down menu. A more natural way of making a new platform is achieved by drawing a horizontal line, much like drawing an addition line to add some numbers. This interaction can be mapped onto direct manipulations in the user interface, such that the user sketches a line representing the platform, and the system recognizes this gesture and compiles the line into a new computational platform. The sequence of creating a computational platform and assigning the layers is actually up to the user: one can first draw a platform and then stack the layers on top, or one could stack layers somewhere on the Geographer's Desktop and then draw a line underneath them.

### 5.2 Selecting platform function

Each computational platform is associated with one computational operation. To set the platform's operation, the platform has a popup menu, which allows the user to change its function at any time, even after the input layers have been added. The visualization of the

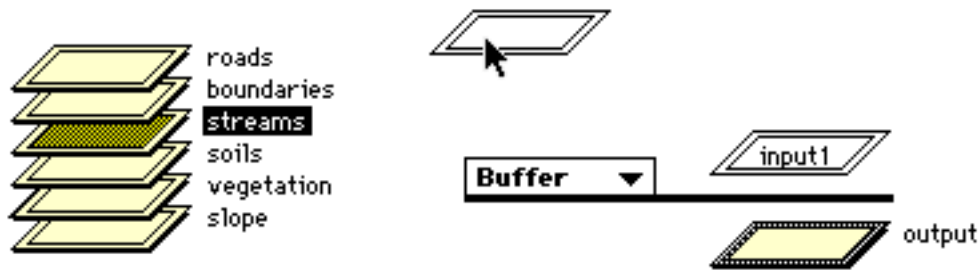
computational platform changes dynamically, as the user walks through the function menu, to provide additional information about the function. For example, the number of empty input slots on top of the platform changes to reflect the number of inputs required by the function currently being pointed at in the menu (Figure 6). Functions that require less layers than already placed onto the platform are dimmed and disabled for selection.



**Figure 6** Platform function popup menu.

### 5.3 Choosing layers as platform inputs

To add a data layer to a computational platform, the user drags and drops the data layer icon onto a computational platform (Figure 7). Rather than moving the original onto the platform, this action produces a link to the original (“alias”) so that the same layer can be used on different platforms at the same time. The use of links also allows changes in the original data set to be propagated through any subsequent analysis.

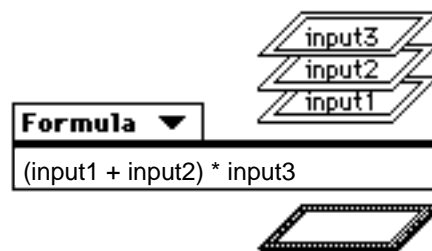


**Figure 7** Adding the streams layer to buffer platform.

When a layer appears over a platform, the bottom of the platform highlights, as does the next available input slot. When the platform highlights, the user may release the layer, and it will “fall” onto the platform. For platforms with multiple inputs, the user can guide the layer outline to the desired input slot, which highlights as the layer outline intersects it. Like moving individual layers onto a computational platform, multiple layers can be selected and dragged as a group. All manipulations to drag layers onto a computational platform are reversed to correct mistakes or make changes.

While input slots guide users, they also make it difficult to support functions with an indefinite number of inputs. On the viewing platform, a user can continue to add additional data layers at will, although the resulting graphical representation may begin to look cluttered. For associative operations, this behavior is also possible on the computational platform.

Another aspect of input slots becomes clear if customized functions are considered. Such functions may integrate different algebraic operations into a single Map Overlay operation and, depending on the layers' positions in the stack, different results may be obtained; therefore, input slots must be related with specific function variables. Figure 8 shows a computational platform visualized with a custom function beneath. The names of the variables in the formula expression match those in the input slots. When the user drags a data layer icon over an input slot, the name of the corresponding variable in the formula expression would highlight, reinforcing the connection between the input and the variable.



**Figure 8** Custom function with labeled input slots.

## 5.4 Setting platform parameters

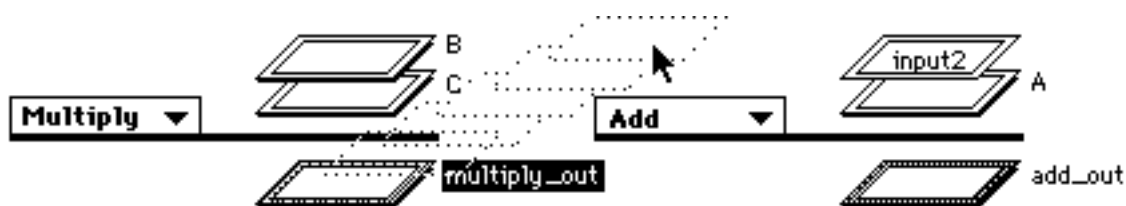
Besides the standard operations of intersection or union there are parametrized Map Algebra operations, which require a user to specify the overlay operation *and* its parameters. For example, making a buffer zone—a parametrized, unary operation—also requires the user to determine the extent of the buffer. While it could be convenient to choose from different buffer operations, each with a different extent such as 10m\_Buffer, 50m\_Buffer, 100m\_Buffer, it is still necessary to offer a generic buffer that can be customized and modified on the fly. Unfortunately, the specification of values does not fit with the selection of an item from a menu. Some operations may have a parameter, others may have several parameters, and many have none. Since the parameters of such operations vary considerably with the function and the data model, the interaction is consequently hard to generalize. A small number of parameters can be located near or in the computational platform. For example, parameters can be placed under the popup menu (Figure 9a), but then interaction with the popup would block the parameters, and the user would be unable to see them change dynamically with the selection function. If the parameters are built into an extended platform, but the platform starts to look less and less like its metaphorical origins (Figure 9b). Alternatively, setting parameters may be relegated to a dialog box that can be accessed by double-clicking the operation.



**Figure 9** Platform with function parameters (a) beneath and (b) in the platform bar.

## 5.5 Platform nesting and aggregates

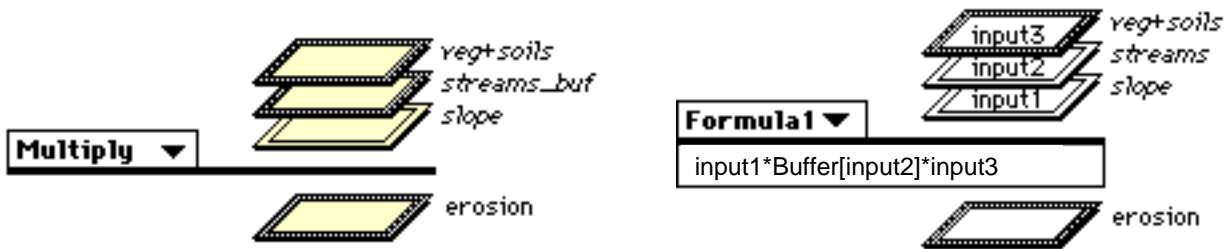
The appearance of the platform output layer icon is virtually the same as the data layer icons. This similarity affords the nesting of functions. A user interface for Map Algebra must have a mechanism allowing the output of one function to be used as the input to another. The computational platform accomplishes this by treating the output layer like any other data layer. Functions can be nested by dragging the output layer of one platform to the input slot of another. After creating and populating several computational platforms, the user has effectively created a formula or model. These platforms can be aggregated together to create a single, custom formula platform. This formula is then available in the function popup, and can be used repeatedly with other data. A simple method of creating a formula is by “example.” The user proceeds with the analysis in a step-by-step fashion, creating and nesting multiple platforms. Other methods of creating a formula are possible. Expert users can create an empty formula platform, and then type in the function in the box below the platform bar. Figure 10 shows how the expression  $A + (B * C)$  is produced. The **multiply** platform is created, and the layers **B** and **C** are placed on it. The **add** platform is then created, and the layer **A** is placed on it. Finally, the output layer of the multiply platform is manipulated in the same fashion as the other data layers: dragged and dropped onto the **add** platform.



**Figure 10** Using one platform's output as an input to another.

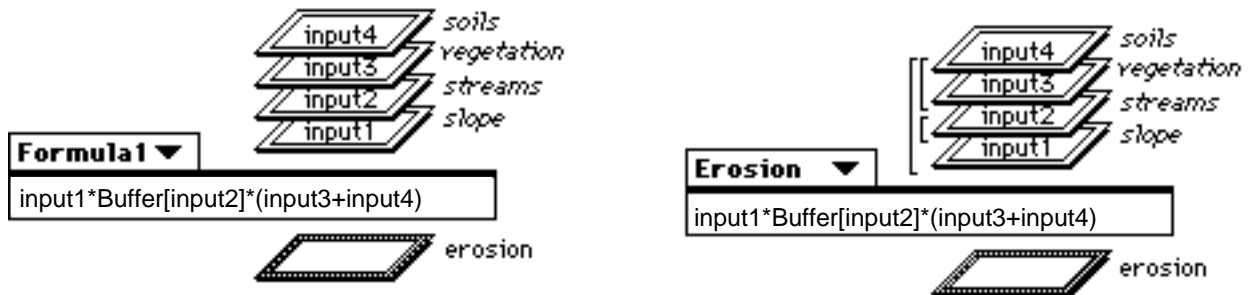
Reversely, compound layers may be expanded, either into multiple computational platforms or into atomic layers on a single platform with a complex platform function. Figure 11a shows a platform aggregate consisting of one data layer, and two platform output layers, added as inputs. Each of the platform output layers can be expanded, leaving their component layers in their place. In Figure 11b, the layer *streams\_buf* has been expanded: *streams\_buf* is replaced by the data layer *streams*, and the function **Multiply** becomes a formula, in which the **Buffer** operation is a part. Since the *streams\_buf* layer was the output of another computational

platform, which performed a buffer on streams, the computational platforms in Figures 11a and 11b are equivalent.



**Figure 11** (a) Platform aggregate and (b) partially converted to a formula.

Figure 11b shows the platform visualized with the formula expression. The variable names illustrate the relation between the inputs and the formula. Alternatively, the names of the actual data layers can be substituted into the formula, and then the input layer names would not be needed. Figure 12a shows the same platform further expanded to the lowest possible level, showing nothing but data layer icons. Once created, the user can assign a name to the formula, which will then appear as one of the choices in the function pop-up menu, allowing it to be used again easily. In Figure 12b, the function name appears on the platform, along with brackets to show the groupings between the functional elements.



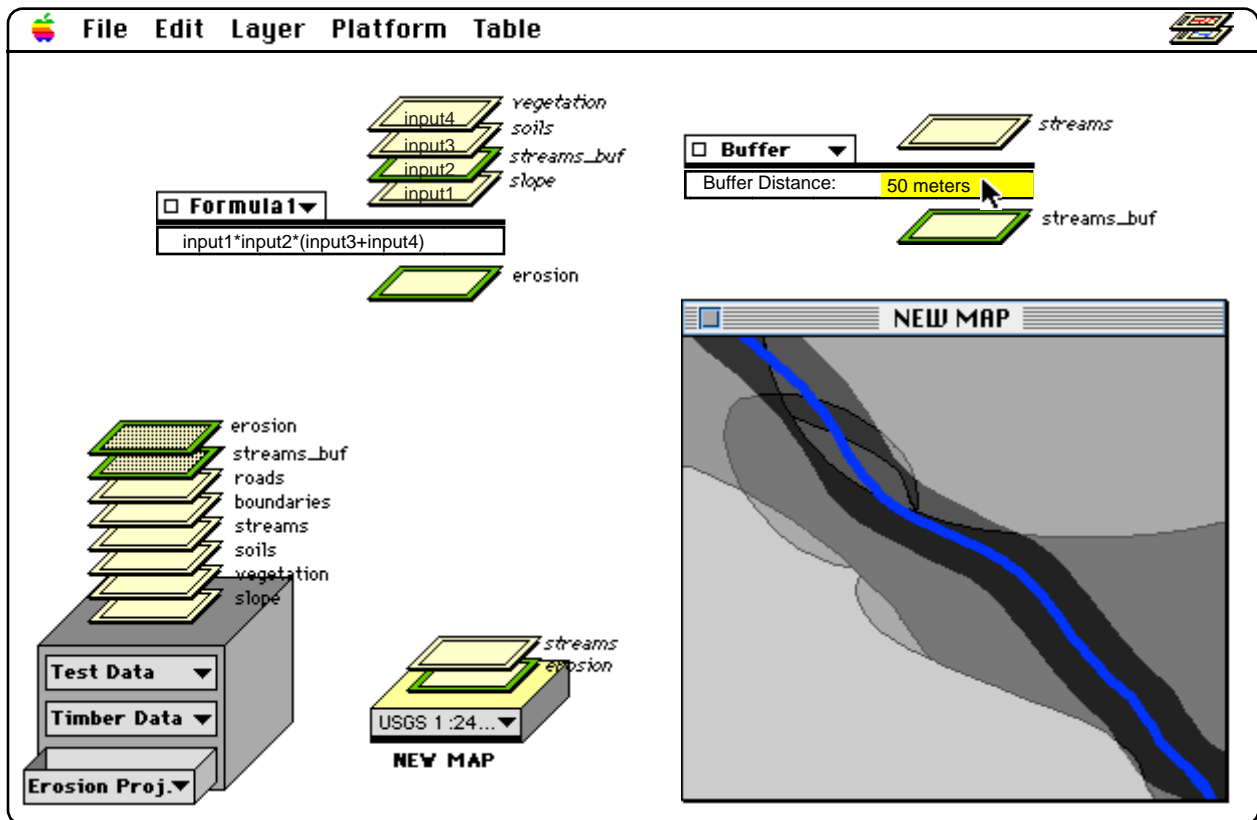
**Figure 12** Platform aggregate (a) reduced to custom formula with all data layer inputs and (b) with named custom formula and functional groupings visualized.

## 6 INTEGRATION OF THE COMPUTATIONAL PLATFORM ON THE GEOGRAPHER'S DESKTOP

The computational platform integrates well on the Geographer's Desktop. Both the viewing platform (Egenhofer and Richards 1993a) and the computational platform (Egenhofer and Bruns 1994) are based on the direct manipulation of thematic data layers. The appearances of the viewing and computational platforms are different, driven by their respective metaphors, because their functionality are also significantly different. These interface objects can coexist on the desktop, and layers can be moved from one to the other at will. Intermediate and end results of Map Algebra expressions can be visualized by placing the respective output layers on a viewing

platform, and with several viewing platforms open at the same time, users will be able to analyze visually different Map Algebra models at the same time and to compare them. If the Map Algebra model is then changed in any way, these changes propagate through the model and are immediately visualized. There is, however, some redundancy in the Visual Map Algebra, because combining several layers on the computational platform with an operation “superimpose” or “add,” and placing the resulting layer onto a viewing platform is equivalent to stacking the initial layers onto the viewing platform. Since the results do not contradict each other, no interaction problems are foreseen.

Figure 13 shows a snapshot of a Geographer’s Desktop. From the map layers available in the map cabinet (in the lower left corner), the user created with the Visual Map Algebra an erosion model on the computational platform in the left top, and a buffer zone around streams on the computational platform in the right top.



**Figure 13** An erosion model created and analyzed with Visual Map Algebra on the Geographer’s Desktop.

The results of the two Visual Map Algebra operations were placed on a viewing platform (bottom center) and the corresponding map view is shown in the window on the right. The user now analyzes the relationship between areas of high erosion potential and a flood zone by varying the width parameter of the stream buffer. Any change of this parameter is immediately propagated into the stream buffer layer and displayed in the viewing window.

## 6 CONCLUSIONS

GIS Map Algebra embodies a range of tools and techniques for spatial analysis. It is generally independent of data model, system type, or implementation. The investigation of GIS Map Algebra yields useful metaphors for user interface design. The map overlay metaphor and the addition-line metaphor formed the foundation for the design of the Visual Map Algebra. Map overlay has visual and physical components. The visual components are mapped onto the appearance and function of interface icons, and the physical component is mapped to the direct manipulation of these icons to form Map Algebra expressions.

A prototype mockup of Visual Map Algebra has been implemented on a Macintosh using Macromind Director as an animation tool. Frequent user feedback from such demonstrations was critical during its design. Several issues are subject to further investigations. Obvious steps are a prototype implementation on top of a GIS, systematic user testing, and performance comparisons with other Map Algebra implementations, or such alternatives as moveable filters (Stone *et al.* 1994). For a limited set of tasks, some analytical comparisons, using cognitive walkthrough analysis (Polson *et al.* 1992), have been made between Visual Map Algebra and a flowchart model (Bruns 1994).

Open questions for future work include:

- The integration of multiple representations into a single user interface. There are tasks like examining the lineage of a compound layer, which may be easier with a flowchart user interface than with nested computational platforms. How can the two user interfaces be integrated seamlessly?
- So far, (carto)graphic aspects have been deferred exclusively to the viewing platform, though an earlier version of layer icons integrated the data with their presentations (Egenhofer and Richards 1993b). Can one infer a reasonable graphical presentation of a compound layer from knowledge about the graphical presentation of the individual layers *and* the algebraic properties of the operation combining them?
- Can GIS operations that manipulate data be integrated with the Geographer's Desktop, without resorting to selections from pull-down menus and toolboxes?

## 8 ACKNOWLEDGMENTS

This work was partially supported by Intergraph Corporation and the National Science Foundation under grant number SBR-8810917 for the National Center for Geographic Information and Analysis. Max Egenhofer's work is further supported by NSF under grant IRI-9309230, Environmental Systems Research Institute, Space Imaging Inc., the Scientific Division of the North Atlantic Treaty Organization, and the Maine Mathematics and Science Alliance.

## 9 REFERENCES

Berry, J. (1987) Fundamental Operations in Computer-Assisted Map Analysis. *International Journal of Geographic Information Systems* **1**, 119–36.

- Berry, J. (1993) Cartographic Modeling: The Analytical Capabilities of GIS, in *Environmental Modeling with GIS* (eds. M. Goodchild, B. Parks, and L. Steyaert), pp. 58–74, Oxford University Press, New York.
- Bruns, H.T. (1994) *Direct Manipulation User Interfaces for GIS Map Algebra*. Master's Thesis, Department of Surveying Engineering, University of Maine, Orono, ME.
- Chan, K. and White, D. (1987) Map Algebra: An Object-Oriented Implementation, in *International Geographic Information Systems (IGIS) Symposium: The Research Agenda* (eds. R. Aangeenbrug and Y. Schiffman), Arlington, VI, Vol. 2, pp. 127–50.
- Dorenbeck, C. and Egenhofer, M. (1991) Algebraic Optimization of Combined Overlay Operations, in *Proceedings of Autocarto 10* (eds. D. Mark and D. White), Baltimore, MD, pp. 296–312.
- Egenhofer, M. (1991) Extending SQL for Cartographic Display. *Cartography and Geographic Information Systems* **18**, 230–45.
- Egenhofer, M. (1994) Spatial SQL: A Query and Presentation Language. *IEEE Transaction on Knowledge and Data Engineering* **6**, 86–95.
- Egenhofer, M. and Bruns, H.T. (1994) Bringing Maps onto the Desktop: GIS User Interfaces Beyond Menus and Buttons, *International IGUG News*, Fall 1994: 8–9.
- Egenhofer, M. and Kuhn, W. (1991) Visualizing Spatial Query Results: The Limitations of SQL, in *Visual Database Systems II* (eds. E. Knuth and L. Wegner), *IFIP Transactions A: Computer Science and Technology*, Vol. A-7, pp. 5–18, North-Holland, Amsterdam.
- Egenhofer, M. and Richards, J. (1993a) Exploratory access to geographic data based on the map-overlay metaphor. *Journal of Visual Languages and Computing* **4**, 105–25.
- Egenhofer, M. and Richards, J. (1993b) The geographer's desktop: A direct-manipulation user interface for map overlay, in *Proceedings of Autocarto 11* (eds. R. McMaster and M. Armstrong), Minneapolis, MN, pp. 63–71.
- ERDAS, Inc. (1993) *Model Maker Tour Guide*. ERDAS, Inc., Atlanta, GA.
- ESRI, Inc. (1990) *Understanding GIS: The Arc/Info Model*, Environmental Systems Research Institute, Inc., Redlands, CA.
- ESRI, Inc. (1992) *GRID Command Reference*. Environmental Systems Research Institute, Inc., Redlands, CA.
- Frank, A. (1992) Beyond Query Languages for Geographic Databases: Data Cubes and Maps, in *Geographic Database Management Systems* (eds. G. Gambosi, M. Scholl, and H.-W. Six), pp. 5–17, Springer-Verlag, New York.
- Goodchild, M. (1992) Geographic data modeling, *Computers and Geoscience* **18**, 401–8.
- Gould, M. (1993) Two Views of the User Interface. in *Human Factors in Geographic Information Systems*. (eds. D. Medyckyj-Scott and H. Hearnshaw), pp. 101–10, Belhaven Press, Boca Raton, FL.
- Intergraph, Inc. (1993) *MGE Grid Analyst (MGGA) Reference Manual*. Intergraph, Inc., Huntsville, AL.
- Johnson, M. (1987) *The Body in the Mind*. University of Chicago Press, Chicago, IL.
- Kirby, C. and Pazner, M. (1990) Graphic Map Algebra. in *Proceedings of the 4th International Symposium on Spatial Data Handling*, Zurich, Switzerland, pp. 413–22.
- Kuhn, W. (1992) Paradigms of GIS Use. in *Proceedings of the 5th International Symposium on Spatial Data Handling*, Charleston, SC, pp. 91–101.
- Lakoff, G. and Johnson, M. (1980) *Metaphors We Live By*. University of Chicago Press, Chicago, IL.

- Lanter, D. (1991) Design of Lineage-Based Meta-Data Base for GIS. *Cartography and Geographic Information Systems* **18**: 255–61.
- Lanter, D. and Essinger, R. (1991) User-Centered Graphical User Interface Design for GIS. National Center for Geographic Information and Analysis, Technical Report 91-6.
- Laurini, R. and Thompson, D. (1992) *Fundamentals of Spatial Information Systems*. Academic Press, New York.
- Mark, D. and Frank, A. (1989) Concepts of Space and Spatial Language. in *Autocarto 9*, Baltimore, MD, pp. 538–56.
- Myers, B. (1990) Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages and Computing* **1**: 97–123.
- Norman, D. (1987) Cognitive Engineering-Cognitive Science. in *Interfacing Thought—Cognitive Aspects of Human-Computer Interaction*. (ed. J. Carroll), pp. 325–36, MIT Press, Cambridge, MA.
- Norman, D. (1988) *The Design of Everyday Things*. Doubleday, New York.
- Nyerges, T. (1993) How do People Use Geographic Information Systems? in: *Human Factors in Geographic Information Systems*. (eds. D. Medyckyj-Scott and H. M. Hearnshaw), pp. 37–49, Belhaven Press, Boca Raton, FL.
- Pazner, M., Kirby, C., and Theis, N. (1989) *Map II Map Processor Tutorial*. John Wiley & Sons, New York.
- Polson, P., Lewis, C., Rieman, J., and Wharton, C. (1992) Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *International Journal of Man-Machine Studies* **36**, 741–73.
- Sandhu, J., Amundson, S., and Marble, D. (1987) *The Map Analysis Package*. The Ohio State University, Columbus, OH.
- Stone, M, Fishkin, K., and Bier, E. (1994) The Movable Filter as a User Interface Tool. in *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, Boston, MA, pp. 306–12.
- Tomlin, C.D. (1983) A Map Algebra. in *Proceedings of the 1983 Harvard Computer Graphics Conference*, Cambridge, MA.
- Tomlin, C.D. (1990) *Geographic Information Systems and Cartographic Modeling*, Prentice-Hall, Englewood Cliffs, NJ.
- Tomlin, C.D. and Berry, J. (1979) A Mathematical Structure for Cartographic Modeling in Environmental Analysis. in *Proceedings of the ACSM*, pp. 269–83.
- Turk, A. (1993) The Relevance of Human Factors to Geographic Information Systems. in *Human Factors in Geographic Information Systems*. (eds. D. Medyckyj-Scott and H. Hearnshaw), pp. 1–31, Belhaven Press, Boca Raton, FL.
- Weller, H. and Hartson, H. (1992) Metaphors for the Nature of Human-Computer Interaction in an Empowering Environment. *Computers in Human Behavior* **8**, 313–33.

check spacing  
check references  
check page size  
arrange for color (?)