A Prototype Spatial Data Management System

Christopher F. Herot
Richard Carling
Mark Friedell
David Kramlich

Computer Corporation of America
575 Technology Square
Cambridge, MA 02139

ABSTRACT

Spatial Data Management is a technique for organizing and retrieving information by positioning it in a spatial framework. Data is accessed in a Spatial Data Management System (SDMS) via pictorial representations which are arranged in space and viewed through a computer graphics system. These pictures can be created by an interactive graphical editor, allowing an SDMS to serve as a personal repository of diagrams, text, and photographs. Pictograms can also be generated from data in a symbolic database management system, allowing SDMS to be used as an interface to large, shared databases.

A prototype SDMS has been constructed which employs a set of color, raster scan displays driven by a large minicomputer. The user can create and examine data surfaces which are larger than the display screen, traversing a surface and zooming in and out to control the level of detail displayed. The prototype system provides a uniform mechanism for accessing a wide variety of data types in a manner which does not require the use of a formal command or query language.

Key Words and Phrases: Computer Graphics, man-machine interaction, database query languages, graphics languages.

CR Categories: 3.7, 8.2

1. INTRODUCTION

Spatial Data Management is intended to allow a person without formal training to access information in a computer. Previous approaches to this problem have involved replacing the usual formal database query language with some more friendly means of specifying the data to be retrieved. These approaches include letting the user express his query in natural language [1] [2], giving examples of the desired results [3], or describing the retrieval process graphically [4]. However, each of these techniques still requires that the user precisely specify the data to be retrieved. Composing such a specification requires a knowledge of the contents of the database and frequently requires some familiarity with its structure as well.

A Spatial Data Management System (SDMS) addresses this problem by organizing information in a much different manner. Whereas a conventional database management system organizes information in a number of abstract name spaces which are referenced by providing values to be matched, an SDMS organizes information in concrete two-dimensional geometric spaces which are more familiar to computer-naive users. The data is given a pictorial representation and placed on a surface which is viewed through a computer display. One such surface, representing a database of ships, is illustrated below. By tying the appearance and location of each picture to the semantic content of the data which it represents, the system enables the user to access information through the simple process of directing attention to the appropriate portion of the data surface.

Figure 1. Ship Database

The construction of a useful Spatial Data Management System requires the solution of two key tech-

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense under Contract No. MDA-903-78-C-0122. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

©1980 ACM 0-89791-021-4/80/0700-0063 $00.75

nical problems:

1. Providing a means by which a user can conveniently view a large data surface, and

2. implementing a mechanism for creating meaningful pictorial representations of symbolic information.

Two previous approaches to the first problem are described in [5] and [6]. This paper describes a prototype SDMS which marries some of the concepts of spatial organization developed in that early work with current database technology to provide a system which can be used to solve real problems of database usage.

2. THE PROTOTYPE SYSTEM

The prototype SDMS provides a variety of mechanisms for creating, viewing, and modifying a Graphical Data Space (GDS) consisting of a set of connected surfaces upon which the data is displayed. This Graphical Data Space is not restricted to containing the formatted data found in conventional symbolic databases but can include anything that can be represented in a computer graphics system, such as charts, diagrams, maps, photographs, and text. In this way, SDMS may serve as a uniform interface to a wide range of dissimilar datatypes.

2.1 The Graphical Data Space

One seemingly straightforward way of organizing data spatially would be to arrange everything on one large surface which would be displayed in its entirety. For most applications, however, such a display would not only be prohibitively expensive, but it would require the user to be constantly moving about a rather large room in search of data. In order to allow the user to remain seated and to make the system more practical, SDMS uses a graphics display as a window through which selected portions of a data surface may be viewed at various scale factors. By allowing this window to move constantly over the data surface, and by providing suitable auxiliary motion and position cues, the sensation of interacting with a large surface can be preserved. The user can thus construct a mental spatial model of the data surface which aids in locating information. This process is aided by the use of distinctive colors and shapes, such that locations may be recognized by their appearance.

The Graphical Data Space may be partitioned into multiple data surfaces which are linked together in a network. The user moves from one data surface to another by means of special pictures called ports. Activating a port causes its associated data surface to become the current data surface. The motion from one data surface to another is similar to that of menu-based systems such as ZOG [7]. While this ability provides a convenient mechanism for partitioning one's data, motion through a network is a foreign concept to most users, and in practice navigation through any appreciably sized network has proven difficult [6]. The ZOG system has addressed this problem by allowing extremely fast transitions between frames. While the results are promising, the technique is only practical for frames which are simple enough to display quickly. In SDMS, the hierarchy of spaces is used most often for separating unrelated information or for providing alternate views of related information while the spatial arrangement within a data surface carries most of the load for organizing information.

Ports can also be used to activate programs external to SDMS. This facility provides a general purpose escape mechanism through which the user can employ various facilities of the operating system such as electronic mail, text editors, or any other subsystem. Such subsystems can be written specially for SDMS, such as the text display program illustrated below, which provides a secondary display of a document table of contents in addition to the document itself.

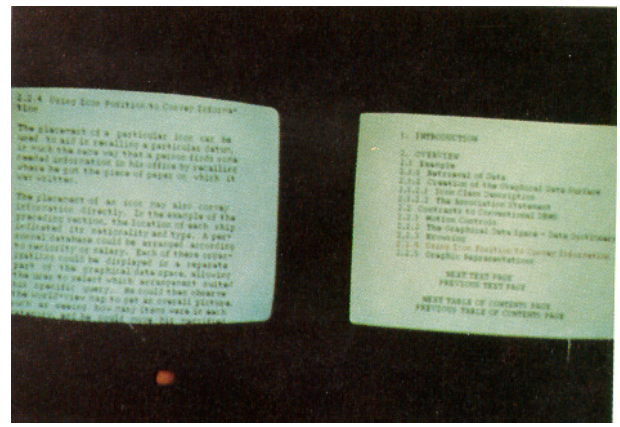

Figure 2. Text Display Program

2.2 The User Station

The user of SDMS sits at a display station like that shown below. The system provides three color, raster-scan displays, fitted with touch-sensitive digitizers. Additional user input can be performed through a three-axis joy stick, a data tablet, and a keyboard. The left-most of the three screens presents a "world-view" of the entire data surface. A magnified view of this data surface is simultaneously displayed on the main screen in the center. The location on the data surface of this magnified portion is indicated by a highlighted rectangle which appears on the world-view map. The two screens are shown below. The right-hand display is used for various other maps and for displaying a menu of commands for use with the graphical editor.

Figure 3. SDMS User Station

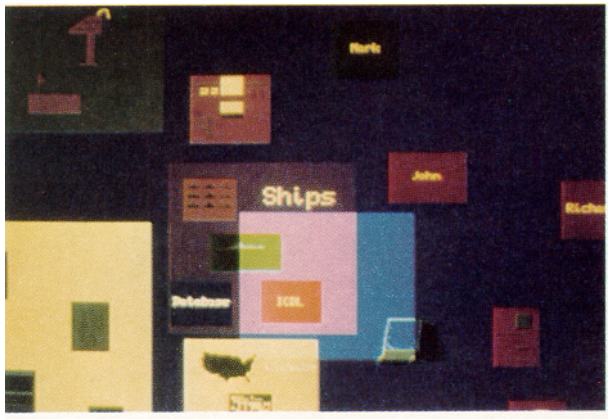

Figure 4. World-View Map

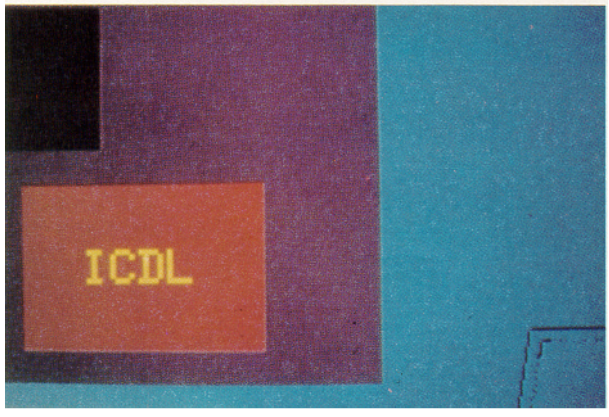

Figure 5. Main Display

2.3 Motion in the Graphical Data Space

The user can control which portion of the data surface appears on the center display by pressing on the joy stick shown in the user's left hand in the foreground of Figure 3. Pressing the joy stick in any given direction causes the user's magnified window to move immediately in that direction over the data surface. This motion is reflected in the corresponding motion of the highlighted rectangle on the world-view map, as shown below. The speed of motion is proportional to the pressure exerted on the joy stick. At the maximum speed the width of the center display is traversed in approximately two seconds.

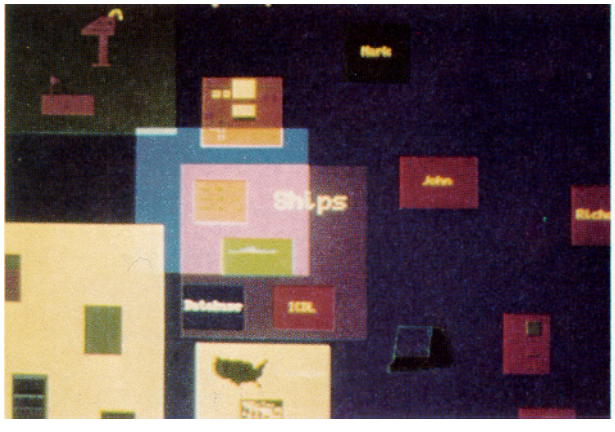

Figure 6. World-View Map

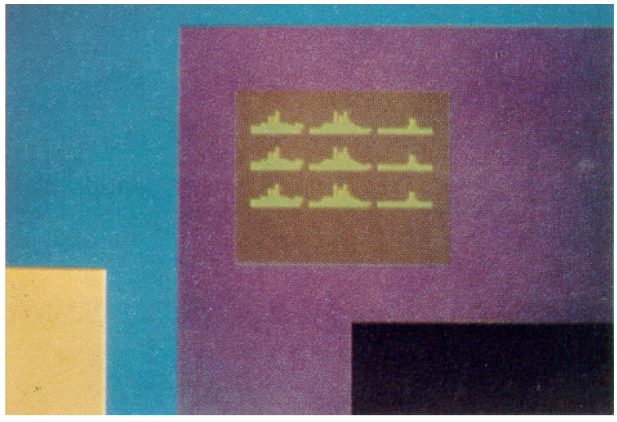

Figure 7. Main Display

A second type of motion which the user can perform is in a direction perpendicular to the data surface. Twisting the joy stick causes the picture to enlarge. This occurs in two stages. The first stage is done by zooming the display, a hardware technique which causes pixels to be replicated on the display, as shown below. The second stage depends on the user's position in the data surface.

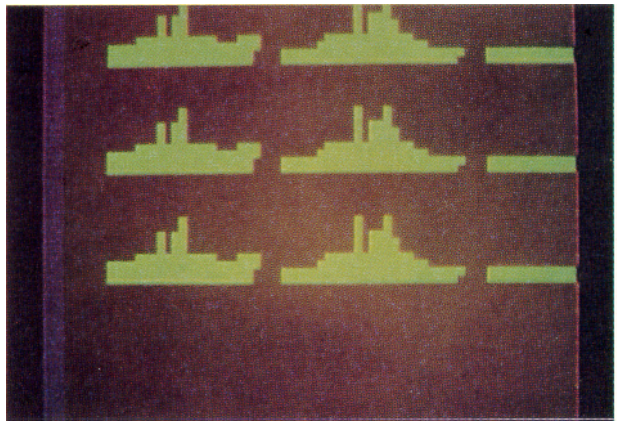

Figure 8. Zooming the Display

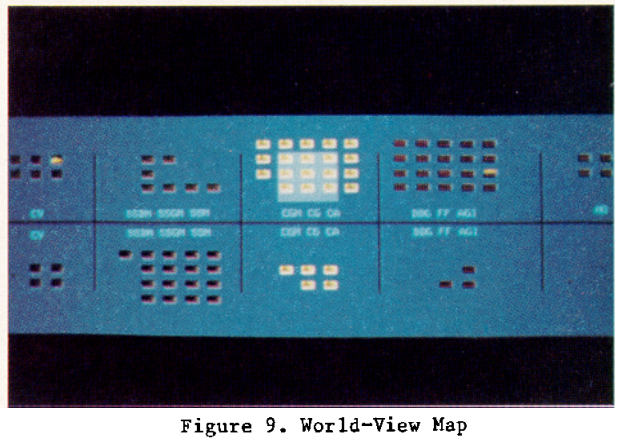

Figure 9. World-View Map

If he is positioned over a port, the port is activated, causing its associated data surface to become the current data surface. The result of zooming in on the "ships" port is shown below. Note that the world-view map has changed to be that of the new data surface.

Figure 10. Main Display

Some data surfaces are stored at several levels of detail. Zooming in on such a data surface causes the more detailed version to appear. The following figures show the result of zooming in on such a data surface.

Figure 11. Zooming In

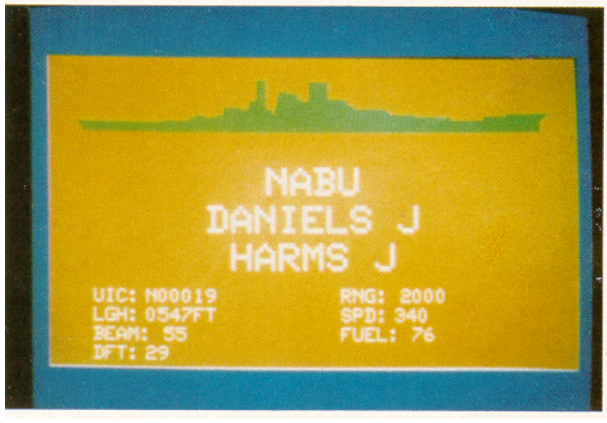

Figure 12. Detailed View of Data

2.4 Global Actions

There are four actions which the user can carry out regardless of his position in the graphical data space. These are:

1. selection of a navigational aid,
2. rapid transit,
3. using the interactive graphical editor, and
4. entering symbolic queries into the database management system.

The left-hand screen may be directed to display either of two forms of navigational aid. One of these is the world-view map, described above. The other is a map of the hierarchy of the data surfaces, illustrated below. Each data surface is represented by a labeled square, with the current data surface indicated by having its background color changed to yellow (the label is always visible). The lines connecting the squares show the possible routes (through ports) which can be taken from one data surface to another. The green squares indicate ports which connect to data surfaces already "owned" by another data surface. Thus, although ownership is hierarchical, the paths which can be taken through the various data surfaces form a network. The blue squares are "off-page-connectors" to other portions of the hierarchy map.

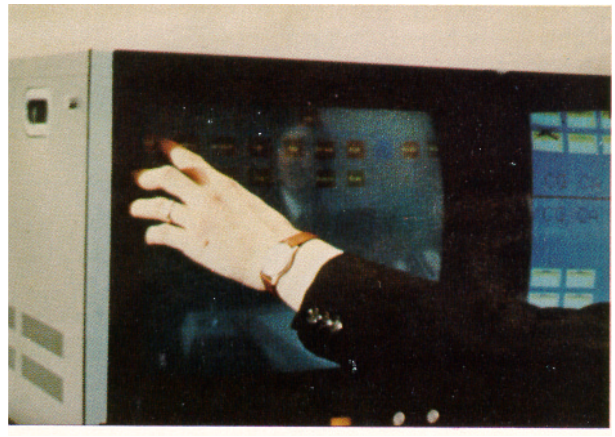

Figure 13. Hierarchy Map

Rapid transit is a mechanism by which a user may move from one point in the Graphical Data Space to another without traveling through the intervening space. Rapid transit is invoked when the user touches the navigational aid with his finger, as in the previous photograph. If a world-view map is displayed the user's window is moved to that location on the data surface. If the hierarchy map is displayed, the data surface which was pointed to becomes the current data surface. Rapid transit takes between one and three seconds. In order to provide immediate feedback to the user, the navigational aid is updated as soon as the screen is touched.

The graphical editor allows the user to annotate the data surface and add graphical data. It operates on the portion of the data surface currently displayed on the center screen. It is also used to edit the Graphical Data Space by creating data surfaces and ports. The editor is described in more detail in the next section.

Although a large portion of a user's need for access to information can be handled through the mechanisms described above, there are situations where the ability to precisely specify a query is required. In these cases, the user can employ a formal symbolic query language which is an extension of QUEL [8]. This language allows the user to define new data surfaces containing selected objects or to cause selected objects on an existing data surface to be blinked. The relationship between the symbolic and graphical modes of representation is discussed in more detail in the following section.

3. INPUT OPERATIONS

Graphical representations can be placed in the SDMS Graphical Data Space through either of two mechanisms:

1. Manual picture creation allows a user to create pictures with an interactive graphical editor.

2. Automatic picture creation generates a graphical view of a symbolic database under the control of a previously entered description.

The manual mode allows SDMS to function as a personal electronic workspace in which a user can store and retrieve maps, charts, diagrams, and photographs. The automatic mode allows SDMS to be used as an interface to a symbolic database which may be large, shared, and subject to frequent updates.

3.1 Manual Picture Creation

The SDMS prototype provides an interactive graphical editor which may be activated at any location in the graphical data space. Like many interactive painting programs [9] [10] it allows the user to select from a menu of commands which perform various graphical operations in response to coordinates input from a data tablet. The SDMS editor has the unique capability, however, of allowing operations which span an area larger than the display screen by allowing the user to move continuously over the data surface both within and between commands.

A person using the editor selects a function by touching with his finger a point on the menu which is displayed on the right-hand screen, shown in the following photograph. The functions include:

1. Geometric primitives such as lines, rectangles, circles, "ink" (continuous lines), and a random dot pattern which simulates an air brush,

2. graphical transformations such as translation and scaling,

3. attributes such as color and width,

4. a variable grid,

5. input digitized from a vidicon, and

6. a library of previously defined shapes.

The user performs these operations on the portion of the data surface currently displayed on the center screen. As he does this, the world-view map is continuously updated so that it always displays

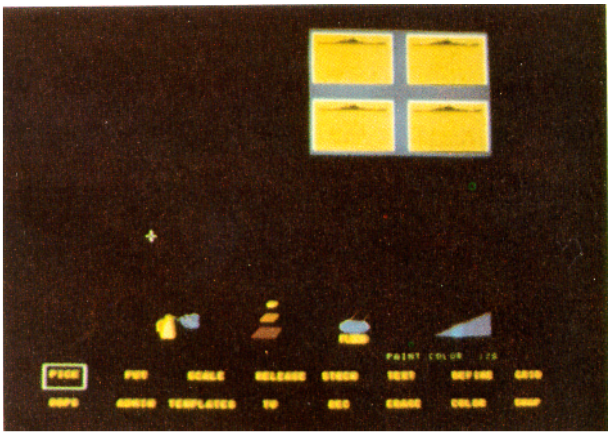

Figure 14. Graphical Editor Menu

a correct scaled-down version of the entire data surface.

An additional set of commands allows a user to create new data surfaces and ports and perform other operations described in the next section.

As with the joy stick and rapid transit operations, the user is given immediate feedback to every action. When a command is selected by touching the menu screen, a box is immediately drawn around the selected command, confirming its selection. If the command requires input of points on the data surface (as when drawing a line), a cursor appears on the center display. The position of the cursor is tied to the position of a puck on the data tablet on the table in front of the screen. Commands which require multiple points always provide some form of intermediate feedback. For example, the RECTANGLE command requires two points which define any two diagonally opposite corners of a rectangle. After the user has placed just one corner of the rectangle a white rectangle appears, with one corner fixed at the indicated position and one corner moving as the user moves the puck. When the user indicates the second point, the rectangle is drawn in the selected color.

The editor is described in more detail in [11].

3.2 Automatic Picture Creation

One of the most important uses of SDMS is as an interface to information stored in a conventional symbolic database management system. Such database may be large, shared with other users, and dynamic. While it would be possible to use the editor described above to manually create pictures for each entity in the database, this approach would hardly be practical for a database with hundreds, much less thousands, of entities. Furthermore, these pictures would have to be modified whenever the database was updated. Very often, such databases already exist in some symbolic database management system and, since they are shared with other users, can not be altered to fit into the SDMS framework.

To address these issues, SDMS provides a mechanism for defining a set of rules which determine how the data in a symbolic database should be expressed graphically. These rules form an icon class description and the pictures they create are referred to as icons. The icon class description used to create the pictures of the ship database shown on the preceding pages is shown below.

```
icon class cluster(r)
begin
maximum size is (110,60);
position is
   (case r.type begin
                "CV"     :800
                "SSN"    :1600
                "SSBN"   :1600
                "SSGN"   :1600
                "CGN"    :2500
                "CG"     :2500
                "CA"     :2500
                "DDG"    :3200
                "FF"     :3200
                "AGI"    :3200
                "AO"     :4000
                default  :1600
                end,     case r.nat begin
                         "US":1200
                         "UR":2000
                         default:2500
                         end );

      template icon  case r.type  begin
                                  "CV"     :carrier
                                  "SSN"    :sub
                                  "SSBN"   :sub
                                  "SSGN"   :sub
                                  "CGN"    :cruiser
                                  "CG"     :cruiser
                                  "CA"     :cruiser
                                  "DDG"    :destroyer
                                  "FF"     :destroyer
                                  "AGI"    :destroyer
                                  "AO"     :oiler
                                  default  :cruiser
                                  end;
      scale is r.beam*2 percent;
      color of region 1 is case r.ready begin
                                        "1":green
                                        "2":yellow
                                        "3":orange
                                        "4":red
                                        default: gray
                                        end ;
      attribute region r.nam from ( 5,16) to (70,28);
      attribute region r.ires from ( 5,28) to (70,40);
      attribute region r.conam from ( 5,40) to (70,52);
end;
```

Figure 15. Icon Class Description

A data surface is populated with icons in response to the user typing a display statement to the query language of the database management system. This statement specifies which entities are to be represented as icons. Through an optional qualification to the associate statement, the user may specify that only certain entities be selected, such as only U.S. ships with less than 20% fuel. For each entity which is selected, the system creates one icon by interpreting the icon class description with its variables bound to values of the particular entity being displayed.

The icons themselves are not stored in the symbolic database but are graphical interpretations of data contained in that database. This is in contrast to systems like [12] where a symbolic database management system is used to contain descriptions of the geometric primitives used to construct a picture. The approach taken by SDMS allows the system to display shared databases which do not originate as pictures in SDMS. The statements in an icon class description perform some graphical action and take arguments whose values are attributes of the entity being displayed.

The position statement determines the placement of the icon on the data surface. In the example, it maps the ship's type into x-coordinates and nationality into y-coordinates. Once an icon is created, further attempts to create icons at that location result in a nearby location being used.

The template statement specifies the shape of the icon by selecting among a set of pictures which have previously been drawn by the database administrator.

The scale statement specifies the size of the icon. In the example this is a function of the beam of the ship.

The color statement specifies the color of each ship according to its readiness.

Finally, the three attribute region statements place the values of the ship's name, its international radio call sign, and the commanding officer's name into the specified locations in the icon.

The use of templates allows the user to define complex icon shapes without requiring the use of an awkward graphics language. Different templates and icon class descriptions may be defined for each level of detail in the data surface, allowing explicit control over the scaling process. For instance, the ship data surface presents different amounts of text and different pictures of ships at each level of detail. The template for one type of ship is shown below. Note the four different templates, one for each level of detail in the ship data surface.

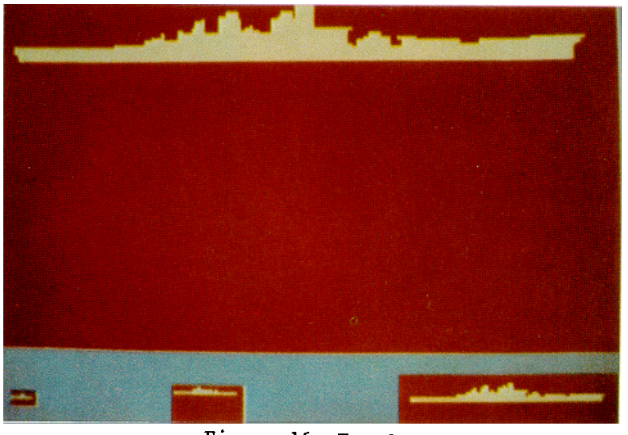

Figure 16. Templates

A template is the starting point for the generation of an icon. Its shape and color determine the shape and color of the icon. A template is divided into regions which are distinguished by the colors with which they are drawn. This allows arbitrary complexity in the shape of regions. The color of a region may be transformed by a color statement in the icon class description. For example, the colors of the ships themselves in the ship data surface are displayed as a function of the readiness attribute of that ship in the database.

4. IMPLEMENTATION

The SDMS prototype runs on a dedicated PDP-11/70 running a modified version of the UNIX operating system [13]. The machine has 1.25 megabytes of memory which is used primarily for manipulating the graphical data surfaces and staging them to the display. The operating system has been modified to allow user programs to map selected portions of their 16 bit address space into portions of physical memory which contain graphical data. The images are displayed on a Lexidata raster scan display system which refreshes three independent color

displays out of its own 480x640 frame buffers. The SDMS system runs in several parallel processes which communicate through shared memory and pipes (a low-speed interprocess channel which appears to a user program as an i/o device). The prototype configuration supports a single user. Although general purpose time sharing can be run simultaneously with SDMS, memory address space limitations preclude supporting more than one SDMS user simultaneously.

Data surfaces are stored on a moving-head disk as bit arrays known as image-planes. An image plane is partitioned into tiles which are rectangles of 64 lines, each containing 128 pixels. Thus each tile occupies 8K bytes, which is equal to the page size of the PDP-11/70. The ships database, shown earlier, consists of four image planes (to support zooming to four levels of detail) and occupies 36 megabytes.

A simplified process structure appears below. All of the processes shown run in the 11/70. The various input devices connect directly to the 11/70 where they are read by processes indicated on the diagram.

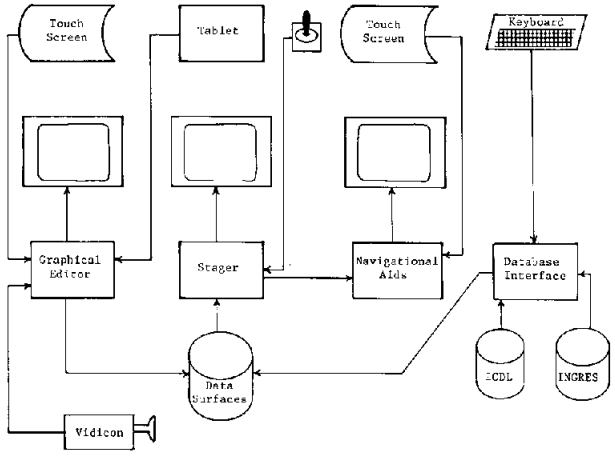

Figure 17. SDMS Process Structure

The stager is actually two processes which allow the user to move the display window over the data surface. One of these processes reads the bit-array representation of the data surface into the main memory of the PDP-11/70. This process endeavors to keep in memory all tiles currently on the screen plus sufficient extra tiles to maintain the user's current direction of motion. A second process sends portions of these tiles to the display, once again endeavoring to maintain a suitable margin of data just off-screen to allow for motion across the data surface. When the user presses on the joy stick, indicating a desire to move in a given direction, this process first checks to make sure that sufficient picture data is loaded into the display to start moving in that direction. If so, the display is commanded to begin its screen refresh at a new location, causing the picture to scroll. The program then determines whether new data must be sent to the display to maintain the margin around the displayed picture. This may require that additional data be read in from the disk as well. This process continues as long as the user holds down the joy stick or until an edge of the data surface is reached. This same program is responsible for updating the global variables describing the user's current position.

These two programs are also involved in zooming the display. When the user twists the joy stick, the display is commanded to change its scale factor. If the user is over a port or in a data surface which has multiple image planes, the stager must prepare to display the new, more detailed image plane. Fortunately, zooming the display reduces the storage required to contain the current image. The memory which remains is filled with the new image plane in anticipation of the user's continued twisting of the joy stick. Finally, if the user does continue to twist the joy stick, the next image plane, which has already been loaded into the display memory, appears on the screen.

The navigational aids process is responsible for displaying the current world-view map or the data surface hierarchy map. It alters the position of the highlighted rectangle as the global values of the user's current coordinates change. This process also listens to the touch sensitive digitizer mounted in front of the navigational aid display and directs the stager to perform a rapid transit operation if the user touches the screen.

The database interface is responsible for creating data surfaces in response to user commands and for blinking selected icons when so requested. It is also responsible for maintaining the consistency between the symbolic and graphical representations in the database. To this end, it monitors updates to the symbolic database. If such an update changes the value of an attribute which was used to create an icon, that icon is recreated with the new data.

The graphical editor allows the user to modify the graphical data space. It operates upon the same in-core tiles used by the stager. It also manipulates the system's description of the graphical data space, allowing the user to delete icons, designate pictures as templates, and create ports.

5. CONCLUSIONS

The initial informal evaluations by users who have used the system in our laboratory have been enthusiastic. Most people learn to use the controls in a matter of seconds and can create pictures almost immediately. Much of this success can be attributed to the small number of simple controls and the furnishing of immediate feedback to every user action. The touch sensitive digitizers simplify the process of directing user input to the appropriate display. They would have been used for all three displays (replacing the use of the data tablet in the graphical editor) if they had sufficient resolution and a means of sensing finger position (for intermediate feedback) before the user pressed on the screen.

The SDMS method of accessing data seems to be most appropriate for those situations where the user needs to browse through a database. Since the data to be retrieved need not be specified precisely, the user does not need to learn a formal query language or possess an intimate knowledge of the structure and contents of the database. On the other hand, SDMS is not especially well suited to those circumstances which require locating some number of entities in the database which meet some precisely stated criterion.

As was anticipated, the least successful aspects of the system are those which require keyboard input. The selection of entities which are to appear on a data surface requires the use of a database query language which is awkward at best. Likewise, the icon class descriptions require the services of

someone who not only knows the organization of the database and the various commands but has the talent to design a data surface which is useful and aesthetically pleasing.

A more rigorous evaluation of the system will take place during the coming year when the system is installed at several selected sites where it can be used under conditions more closely approximating those of the ultimate operational environment. This process will involve loading the system with real databases and adding process ports for some previously implemented decision analysis programs.

5.1 Future Work

The usefulness of the system could be greatly increased if graphic representations of databases could be created with less human intervention. An approach outlined in [11] would use profiles of users and their applications together with descriptions of the semantics conveyed by various representations to select the most appropriate representation for any particular circumstance.

The composition of complex queries could be accomplished by allowing the user to manipulate the graphical representations, creating a picture of the desired result in a form of graphical query-by-example. This might make the requisite logical thinking less painful for the user.

In terms of implementation details, two very specific goals will be pursued: creating icons more quickly and storing graphical data surfaces more compactly. A much faster icon creation would allow the icons to be generated on the fly as the user scrolled over the data surface. This would greatly reduce the storage requirements of the system and also eliminate the problems of maintaining the consistency of the two different methods of storage currently employed. One approach which will be attempted in the short term is to split the process of icon creation into two stages. The retrieval from the symbolic database will produce a "compiled" icon description. This description will be expanded as the user traverses the data surface. This approach would allow several users to examine the same graphical data space in an efficient and economical manner.

References

1. Hendrix, G., Sacerdoti, E., Sagalowicz, D., Slocum, J., "Developing a Natural Language Interface to Complex Data", ACM Transactions on Database Systems 3:2, June, 1978.

2. Woods, W., Kaplan, R., Nash-Webber, B., The Lunar Sciences Natural Language Information System, Bolt Beranek and Newman, Cambridge, Mass., June, 1972.

3. Zloof, M., "Query by Example", Proc. AFIPS 1975 NCC, Vol 44, AFIPS Press, Montvale, N.J.

4. McDonald, N., Stonebraker, M., "CUPID: The Friendly Query Language", Proc. ACM Pacific Conference, San Francisco, April, 1975.

5. Donelson, W., "Spatial Management of Information", Proc. ACM SIGGRAPH 1978, Atlanta, Georgia.

6. Bolt, R., Spatial Data Management, Architecture Machine Group, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.

7. Robertson, G., McCracken, D., Newell, A. The ZOG Approach to Man-Machine Communication, technical report CMU-CS-79-148, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October, 1979.

8. Stonebraker, M., Held, G., Wong, E., "INGRES - A Relational Data Base System", Proc. AFIPS, Volume 44.

9. Shoup, R., "Superpaint...the Digital Animator", Datamation, May, 1979.

10. Smith, A. Paint, Tech. Memo No. 7, Computer Graphics Lab, New York Institute of Technology, Old Westbury, NY, July,1978.

11. Herot, C., Carling, R., Friedell, M., Kramlich, D., and Thompson, J., "Spatial Data Management System Semi-Annual Technical Report," Technical Report CCA-79-25, June, 1979.

12. Weller, D., Williams, R., "Graphic and Database Support for Problem Solving", Proc ACM SIGGRAPH 1976, Philadelphia.

13. Ritchie, D., Thompson, K., "The UNIX Time Sharing System", Communications of the ACM, 17:7, July 1974.