Computer Corporation of America

575 Technology Square
Cambridge
Massachusetts 02139
617-491-3670

(12)

(9) Quarterly Research and Development Technical Report
1 Jun-31 Aug 79

(6) Spatial Data Management System

(12) 44

Contract Period Covered by Report: 1 June 1979 - 31 August 1979

(15) MDA903-78-C-0122, ARPA Order-3487

Computer Corporation of America

Report Author: (10) Mark Friedell
David Kramlich
Christopher F. Herot
Richard Carling

Research Division
Computer Corporation
of America

617-491-3670

The views and conclusion in this document are those of the authors and should not be interpreted as necessarily representing the official policies, express or implied, of the Advanced Research Projects Agency, or the United States Government.

(11) 31 Aug 79

387285

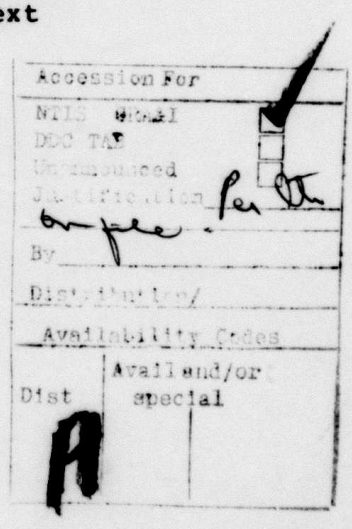
Sponsor: Defense Advanced Research Projects Agency
Office of Cybernetics Technology

ARPA Order Number: 3487

APRA Contract Number: MDA903-78-C-0122

Contract Period: 15 February 1978 - 30 November 1979

Table of Contents



1. Introduction

This technical report describes work completed on a prototype Spatial Data Management System (SDMS) during the quarter-year period ending August 31, 1979.

Section two, entitled "Graphical Display of Data," presents an overview of the icon creation mechanisms currently included in SDMS. Icon creation is the facility by which graphical representations of symbolic data are automatically produced. These graphical representations are referred to as 'icons.' Section two includes detailed discussions of the icon creation features added to the system ~~during this quarter~~: Secondary data a mechanism by which information in multiple relations or tuples may be presented in a single icon, and a subicons mechanism that allows the appearance of a portion of an icon to be described independently of its parent icon.

Section two also proposes extensions to icon creation to support the graphical representation of aggregate information. Section two shows how these extensions might be built upon the existing secondary data and subicon facilities.

Section three of this report describes the blinking and framing extensions to the Spatial QUEry Language (SQUEL). The blinking and framing features allow the combination of symbolic and graphical access to information by modifying the appearance of icons in response to users' queries.

Finally, section four describes two new enhancements to the text viewing facility in SDMS: an anti-aliased text font now being used to improve readability, and an improvement to the procedure for rapid transit through the text.

2. Graphical Display of Data

CCA's Spatial Data Management System incorporates a prototype implementation of icon creation a facility that automatically prepares graphical representations of data stored in a conventional database management system (SDMS employs the relational database management system INGRES [HELD STONEBRAKER WONG]). In the current implementation, icon creation produces separate graphical representations of each INGRES tuple being displayed. These graphical representations are referred to as icons.

Icon creation may be thought of as an Icon Class Description (ICD) processor. An ICD is a mapping rule which operates on symbolic data from the INGRES tuple being displayed to produce its graphical representation. As described in sections 2.1.1 and 2.1.2, an SDMS user may select among a repertoire of ICDs to best suit his information display needs. Typically, an ICD is written by the SDMS database administrator in Icon Class Description Language (ICDL). Section 2.2 describes the capabilities of ICDL and discusses the notion of an icon more fully.

A group of user-drawn graphical primitives is used by icon creation. These primitives are referred to as templates. Templates facilitate the automatic production of icons by serving as the basis for their construction.

Users produce icons through the DISPLAY and ASSOCIATE commands described in sections 2.1.1 and 2.1.2.

2.1 SQUEL Commands

SDMS terminal interaction is primarily composed of extended INGRES query language, QUEL. This extended QUEL is referred to as Spatial-QUEL or SQUEL. A terminal monitor (described in section 2.3.1) allows the user to compose SQUEL transactions, which are one or more SQUEL commands. The SQUEL monitor sends completed transactions to the SQUEL interpreter, which recognizes and executes all SDMS extensions to the INGRES query language. Two of these commands, DISPLAY and ASSOCIATE, result in the activation of icon creation. The DISPLAY statement is used to produce "snapshot" views of the current state of the database; these views do not change as the contents of the database change. The ASSOCIATE statement produces views which change as the symbolic data being represented changes. The ASSOCIATE and DISPLAY statements are described more fully below:

2.1.1 DISPLAY Command

The syntax of the DISPLAY command is:

DISPLAY NAMED display-name IS relation-name USING icd-name
{WHERE qualification}

When recognized by the SQUEL interpreter, icon creation is invoked for each tuple in relation-name meeting the qualification. If no qualification is present, icon creation is invoked for every tuple in relation-name. Icon class description icd-name is used. The icons created by this DISPLAY command can be referred to by display-name.

The ERASE command erases the icons created by a DISPLAY command. The ERASE command syntax is:

ERASE display-name

2.1.2 ASSOCIATE Command

The syntax of the ASSOCIATE command is:

ASSOCIATE relation-name USING icd-name

When recognized by the SQUEL interpreter, icon creation is invoked for each tuple in relation-name. Icon class description icd-name is used. A special semantic applies to all icons created by ASSOCIATE rather than DISPLAY command; changes to an associated tuple result in corresponding changes to its icon. Further, appending or deleting tuples to an associated relation results in the creation or erasure of the appropriate icons.

2.2 Icon Class Descriptions

Icon Class Descriptions (ICDs) explicitly define the graphical representations of symbolic data. ICDs describe the appearance of a rectangular area of a large two-dimensional plane, or Information space (Ispace), as a function of tuple [FRIEDELL SCHMOLZE] components. Such a rectangular area is termed an icon. Icons may be viewed

at several levels of detail. Consequently, ICDs specify an icon's appearance on each of these levels. These levels of detail are referred to as image planes (iplanes).

Tuple components are referenced in ICDs by the attribute names of the relation to which the tuple belongs (i.e. employee.name). ICDs use tuple components as a basis for selecting templates and modifying them by scaling, shading, and adding textual information. The location of an icon in an Ispace may also be specified as a function of tuple components.

ICDs may define a portion of an icon as a subicon. Subicon descriptions are analogous to "parent" icon descriptions (they chose templates, position and modify them, etc.). The subicon mechanism is used when more than the single main icon template is desired in an iplane. For example, the user may wish to display a database of ships such that the color of the ship's hull is indicative of the ship's fuel supply, and the shape of the ship's flag designates the ship's nationality. In this situation, an ICD that describes the appearance of the ship's flag as a subicon is appropriate; the subicon description should choose among templates on the basis of the nationality component of the ship tuple being displayed.

In general, each execution of an ICD by icon creation is meant to produce a graphical display of a single tuple, although a mechanism exists to utilize data from multiple tuples in constructing a single icon. This mechanism is referred to as "data from secondary tuples".

Data from secondary tuples is useful in constructing part of an icon, perhaps a subicon, as a function of data related to, but not part of, the primary tuple. For example, in viewing a database of ships, an SDMS user might require a military record synopsis of each ship's commanding officer. A subicon displaying information contained in the appropriate secondary tuple from a commanding officers relation could provide this synopsis.

2.3 Icon Creation Overview

This section is intended to convey the essential aspects of the implementation of icon creation and its environment.

The term "icon creation" refers to functionality which is, in fact, not provided by the icon creation UNIX process alone, but which is distributed throughout icon creation's multiprocess environment - an environment which invokes

icon creation and provides it with a number of services. The SDMS UNIX processes comprising the icon creation environment are: the SQUEL monitor, the SQUEL interpreter, the association processor, the integrity maintenance daemon, the icon manager, the modify process, and the INGRES database management system. These processes are shown in figure 2.1 in their relationship to the icon creation process.

Interprocess communication among these processes is accomplished through the use of pipes, files, the UNIX exec parameter passing feature, and a queue organized from a file and two locations in shared memory (treated as semaphores) (see LCBA in [BBN]).

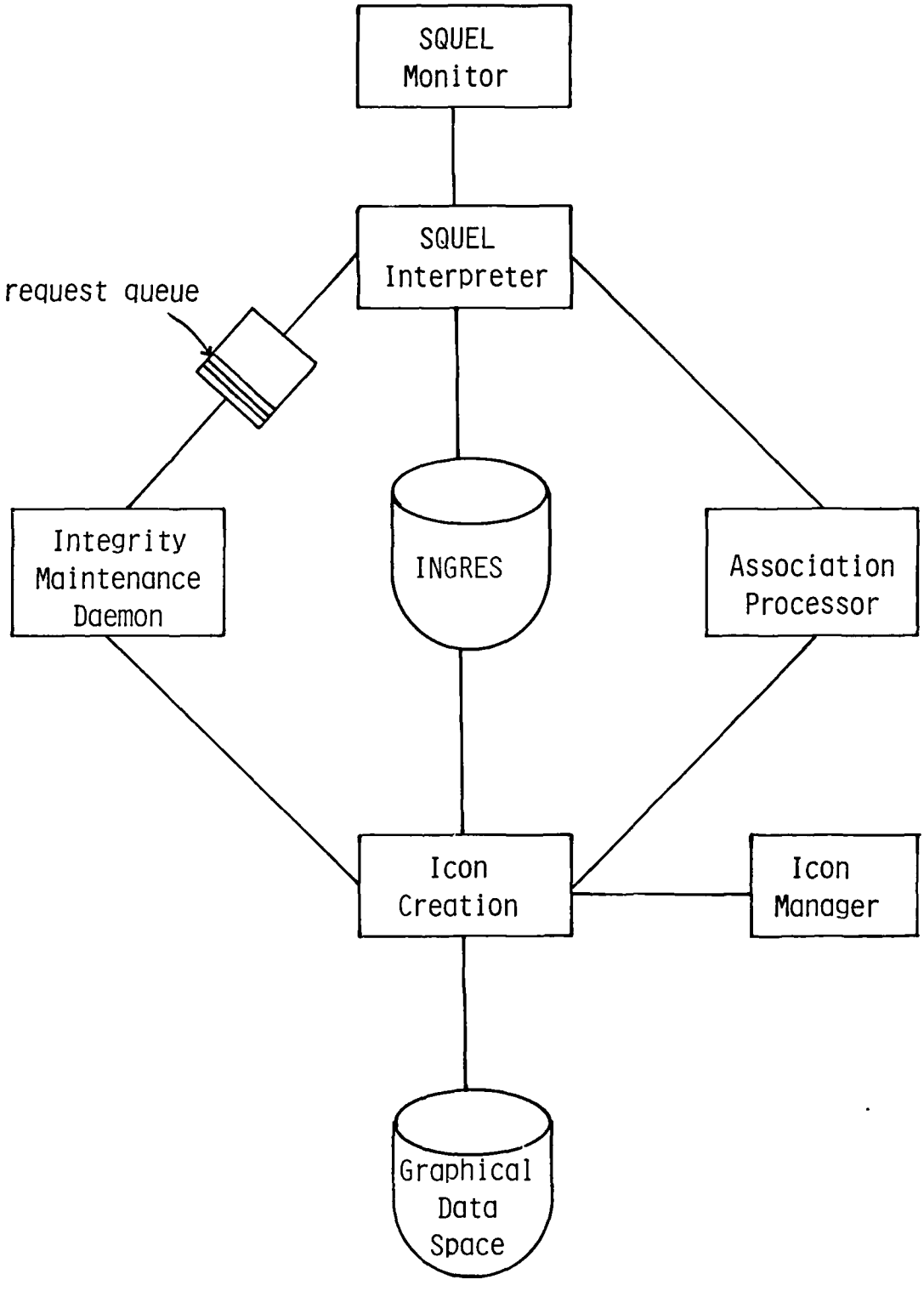
The following subsections discuss the major implementation aspects of the icon creation process and the processes comprising its environment.

2.3.1 SQUEL Monitor

The SQUEL monitor is the source of SQUEL transactions. A SQUEL transaction is one or more SQUEL commands. The primary role of the SQUEL monitor is to assist the SDMS user in preparing transactions from the terminal keyboard,

processes comprising icon creation

Figure 2.1



UNIX files, and previous transactions of the current SDMS session. Much of this primary role is performed by an incorporated screen-oriented editor, NED [BILOFSKY].

Each transaction of the current SDMS session is saved in a temporary file. These files are managed by a history list feature which allows previous transactions to be recalled, edited, executed, and resaved as a new transaction.

The SQUEL monitor considers characters from the terminal keyboard to be at either command or text level, much the same as many line-oriented editors do. Characters are at command level when they follow an escape character as the first character of a new line.

When the SDMS monitor recognizes the command to execute a transaction, it sends the name of a temporary file containing it down a pipe to the SQUEL interpreter where it is executed. The SQUEL monitor then sleeps on a pipe from the SQUEL interpreter waiting for a ready message.

2.3.2 SQUEL Interpreter

The SQUEL interpreter performs any operations required by the SDMS extensions to QUEL, then passes the remaining standard QUEL, if any, on to INGRES to be executed. The SQUEL interpreter recognizes SQUEL commands which require special processing by parsing all SQUEL transactions.

SQUEL commands which require special processing are either extensions to the QUEL repertoire (DISPLAY, ERASE, ASSOCIATE, BLINK, UNBLINK, FRAME, UNFRAME) or statements which may affect the graphical integrity of associations [HEROT et al A] [FRIEDELL] (APPEND, DELETE, DESTROY, REPLACE).

DISPLAY and ASSOCIATE commands, which require bulk creation of icons, cause the SQUEL interpreter to copy the appropriate tuples from the INGRES database into a regular UNIX file. The name of the user-specified ICD, the name of the database, and the name of the relation containing the tuples to be displayed are written down a pipe to the association processor.

APPEND, REPLACE, DELETE, and DESTROY commands affecting associated tuples require the addition, replacement, or deletion of icons. Requests to perform these operations are entered into the integrity maintenance daemon request queue.

2.3.3 Association Processor

The association processor sleeps on a pipe from the SQUEL interpreter waiting for a message that the bulk creation of icons is necessary (as a result of a SQUEL display or associate command). The message contains the name of the ICD to use, the name of the database, and the relation which the tuples are from.

One at a time, tuples are read from the file containing them, and icon creation is invoked for each.

2.3.4 Integrity Maintenance Daemon

The integrity maintenance daemon reads messages from its request queue. Messages are of two types: append and erase.

For an append message, sent by the SQUEL interpreter in response to a SQUEL append or replace command, icon creation is invoked in much the same way as the association processor invokes icon creation.

For an erase message, sent by the SQUEL interpreter in response to a SQUEL replace, delete, destroy, or erase command, icon creation is invoked in erase mode. Erase mode simply shades over the icon with the background color of the Ispace.

2.3.5 Icon Creation

The icon creation process is invoked either by the association processor or the integrity maintenance daemon and constructs exactly one icon per invocation.

Icon creation is implemented as a direct interpreter of Icon Class Description Language (ICDL). In order to access components of the tuple whose icon is being constructed, the tuple is passed as an argument to icon creation in the form of a table relating attribute names to tuple component values. Because this table is incorporated into icon creation's symbol table, Icon Class Descriptions (ICDs) may refer to tuple components by their

attribute names, much as any other name would be referenced in an ICD.

2.3.6 Icon Manager

A complete description of the icon manager can be found in a previous technical report [HEROT et al A]. This section is only a cursory description of one of the many services which the icon manager performs: the positioning of icons in Ispaces.

ICDs include positioning statements of the form:

POSITION IS (X,Y):

Without some positioning intervention icon collisions are likely to occur. In order to prevent collisions, icon creation consults the icon manager before installing an icon in an Ispace. The icon manager maintains a database of areas in the Ispace which are occupied. With this knowledge base, the icon manager adjusts, if necessary, the position of the icon such that collisions are prevented.

2.4 Secondary Data

2.4.1 Functional Description of Secondary Data

When an icon is being created, it is sometimes useful to access data related to, but not contained in, the associated tuple. Icon Class Description Language (ICDL) provides a mechanism by which such data can be accessed. This data is considered to be contained in secondary relations. While secondary data is often very useful, integrity maintenance mechanisms of SDMS cannot guarantee icon appearance updates in response to symbolic changes in secondary data [HEROT et al A].

The secondary data facilities mark a first step toward the provision of an entity description language. In effect, entities can be stored over multiple relations, as is often preferred when using a relational data model [CHAMBERLIN].

2.4.2 Implementation of Secondary Data

The secondary data facilities in ICDL introduce three new statements, two kinds of declarations, and a statement to perform retrieval operations. The first secondary data operation to take place must be the declaration of a secondary relation. The appearance of a secondary relation declaration in an ICD has two effects: One, an interface to INGRES from icon creation is established if it does not already exist. Two, a range variable over the relation with the same name as the relation is established within INGRES.

The syntax of a secondary relation declaration is shown below with ICDL reserved words in upper case:

SECONDARY DATA FROM RELATION relname {, relname}

After the "relation" reserved word is recognized by the ICDL interpreter, the comma separated list of relation names is shifted onto a stack. If the statement ends normally (i.e. by a semicolon), processing begins. First, a flag is interrogated to determine if the icon creation to INGRES interface exists. If not, that flag is set and

an EQUEL [YOUSSEFI] interface is established. Next, each relation name is popped from the stack and a check is made to determine if the relation exists within INGRES. If it does, a range variable is declared for it. If it does not exist, an error message is issued and icon creation aborts with a fatal error.

The other declaration associated with secondary data is the range variable declaration. This statement's syntax is shown below with reserved words in upper case:

RANGE OF rvar IS relname

After the "of" reserved word is recognized by the ICDL interpreter, the range variable name is shifted onto a stack. If the statement terminates correctly (i.e. after the "is" reserved word, the relation name, and a semicolon), an attempt is made to declare, in INGRES, the range variable over the relation. If the attempt should fail (probably because the relation does not exist), icon creation aborts with a fatal error status.

The statement which actually retrieves data from a secondary relation is shown below with all reserved words in upper-case:

RETRIEVE variable = rvar.domname [WHERE qualification]

Note that the range variable may be declared either explicitly, as in the range variable declaration, or implicitly, as in the secondary relation declaration. The qualification is optional.

As this statement is being recognized by the ICDL interpreter, the ICDL variable, the range variable, the domain name, and the qualification (if present) are shifted onto a stack. If the statement ends normally, processing of the statement takes place. First the type (integer or string) if the ICDL variable is determined from the icon creation symbol table. Based on the variable type, an EQUEL integer pointer or character pointer is set equal to the address of the ICDL variable. This pointer is used in conjunction with the range variable, domain name, and qualification, to compose an EQUEL INGRES query. This query is sent to INGRES through the icon creation to INGRES interface. Composition of the query at ICD execution time is achieved through the use of pointers in place of the range variable, domain name, and qualification, of an EQUEL retrieve statement. A statement in general form was preprocessed by the EQUEL preprocessor, and compiled and linked into icon creation. At ICD execution time, the pointers are assigned to the appropriate parts of the ICDL retrieve statement, and the EQUEL retrieve statement is executed. Since the ICDL

scanner includes a buffer manager which may or may not send the qualification to the ICDL parser in contiguous memory, setting the qualification pointer in the EQUEL retrieve statement to the "WHERE" reserved word may not produce the desired qualification. To compensate for this, the ICDL parser bundles [JOHNSON] each token in the qualification as it reads it. The qualification is unbundled in a memory location to which the EQUEL qualification pointer has been preset.

If no data is retrieved from the secondary relation, icon creation aborts with a fatal status. If more than a single value is returned, the first value is assigned to the ICDL variable, and a warning message is issued. If INGRES issues a fatal error message as a result of the retrieval operation, for instance, if the secondary relation had been destroyed, the message is forwarded to the SDMS user and icon creation aborts with a fatal status.

2.5 Subicons

2.5.1 Functional Description of Subicons

The subicon facility is used to introduce multiple templates in a single icon. Subicons are the appropriate tools to describe the components of the main, or parent icon, that vary in appearance as a function of some attribute being represented. Examples are: ships' flags as a function of their nationality, company logos of suppliers in an inventory database, and rank insignias in a military personnel database.

Subicons are often the natural way to describe the appearance of information ancillary to the prime topic of the icon. For example, information regarding the type of escort aircraft that may accompany a particular naval ship might be represented by a subicon which shows aircraft information disjointly from the ship information, but still within the ship's icon.

2.5.2 Implementation of Subicons

As described in the ICDL reference manual [FRIEDELL SCHMOLZE], Icon Class Descriptions include iplane blocks to describe the appearance of the icon at any single level of detail. Iplane blocks, in turn, may contain one or more subicon blocks to describe subicons at that level.

The general syntax (with ICDL reserved words in upper-case) is:

```
IPLANE arith-expr
        BEGIN
        iplane statements
        END;
```

where one valid iplane statement is:

```
SUBICON AT (arith-expr, arith-expr)
        BEGIN
        iplane statements
        END;
```

The effect is to translate the origin of all picture construction operations by the X-Y values designated in

the subicon block header. The domain of these operations is still limited to the area of the parent icon. Clipping all picture construction operations against the bounds of the parent icon enforces this limitation.

The subicon block is implemented using a recursive grammar to describe the nonterminal symbol shown above as "iplane statements." When the ICDL interpreter recognizes the reserved words "SUBICON AT" and the parenthetically enclosed X-Y offset, subicon processing begins.

First, the old icon origin and extents are stacked. Next, the new origin becomes the old origin translated by the values obtained from the subicon block header; the extents become the old extents minus the subicon header values. Finally, the ICDL parser recurses on the "iplane statements" grammar.

When the subicon block end is reached, the old icon origin and extents are popped from the stack.

2.6 Extensions to Icon Creation: Aggregate Data Display

This section describes a proposed extension to the data display capabilities of SDMS. The current display facilities are founded on the concept of entities (tuples which a.e graphically represented), entity icons (icons which are a product of icon creation rather than being hand-drawn), and mappings rules (icon class descriptions - see section 2.2) from entities to entity icons.

The course of this current research in the display of large databases suggests that a facility to display aggregate information in addition to specific entities would be useful.

One feasible approach would treat the display of data as a single function to operate on sets of tuples to produce "displays". Each element of this function would include a rule (an ICD-like definition) to map an element of the domain (a set of of tuples) into its image (a display). In effect, whenever a SQUEL associate command was issued, an element would be added to this function.

Replacing the current icon class descriptions, "display definitions" would produce a single image in an Ispace's top most iplane which conveys aggregate information. More detailed iplanes could be partitioned into areas representing individual tuples (or entities).

2.6.1 Specific Modifications to the Current Implementation

Much of the functionality necessary to achieve aggregate data displays is present in the current SDMS implementation. Aggregate data display implementation efforts could exploit combinations of the facilities supporting subicons and data from secondary tuples.

The mechanisms underlying the retrieval of data from secondary tuples return tuple components specified by ICDL qualifications. These mechanisms could be modified to return aggregate data specified by appropriate qualifications.

Assuming that "display definitions" are executed once, detailed iplanes which resolve portions of the Ispace as representing individual tuples could be constructed by retrieving each tuple through the data from secondary

tuple mechanisms and producing a subicon for each such tuple. A "display definition" execution flow control feature which looped for each tuple in the set of tuples included in the display would be needed.

3. Blinking and Framing

3.1 Function

Blinking and framing provide a mechanism to aid the SDMS user in locating specific icons which have been created by an associate. In this context, "retrieval" means temporarily modifying the selected icons in the graphical data space (GDS) to make them visually distinctive. This may be accomplished in one of two ways. The first is blinking, the second is framing.

The user causes a subset of icons from an association or display to blink by issuing a BLINK command in QUEL. The command specifies the relation whose icons are to be blinked and an optional qualification which will select some subset of the associated icons. The consequence of this command is to cause the selected icons to be blinked wherever they might be in the GDS. Blinking of the selected icons continues until an UNBLINK command is issued.

The user may frame selected icons in a manner analogous to blinking. There are FRAME and UNFRAME commands in SQUEL which are identical in syntax to the BLINK and UNBLINK commands, respectively. The effect of a FRAME command is to draw a blinking frame around the selected icons. Icons continue to be framed until an UNFRAME command is issued.

The implementation of blinking and framing has been changed from the description published in the last quarterly technical report [HEROT et al B]. The decision to change the fashion in which blinking and framing is implemented was based on an analysis of the computational load demanded by the proposed implementation. The details of the current implementation are described in the following section.

3.2 Implementation

The implementation of blinking and framing in SDMS affects three different processes. Those processes are: (1) the SQUEL interpreter, where the command is parsed; (2) the modification process, where the data structures necessary for blinking and framing are managed; and (3) the blinking process, where the actual blinking of the visible icons on the display is done.

This section will describe that portion of each of these processes which pertains to blinking and framing.

3.2.1 SQUEL

The SQUEL interpreter, once it has parsed a blink, frame, unblink, or unframe command, must create a list of icon-ids for the affected icons. This list and a command signifying what modification is to be made to the icons in the list are passed to the modification process. The mechanism for communication between SQUEL and the modification process is a set of pipes.

3.2.2 Modification Process

The modification process is responsible for several tasks. It must: (1) process icon lists arriving from the SQUEL interpreter; (2) manage a database of active modifications; and (3) modify the graphical representations of the icons affected.

When the modification process receives a command from the SQUEL interpreter, processing may take either of two

branches. Blink and frame commands take one path, unblink and unframe commands take the other.

3.2.2.1 Blink/Frame

To execute a blink or frame command, the modification process must ask the Icon Manager for detailed information about each of the icons in the list that accompanies the command. It must know the following about each icon:

- the I-space in which the icon is located
- the X and Y origin of the icon (universal coordinates)
- the width and height of the icon (universal coordinates)
- the color of the icon

This information about each icon, as well as the icon-id and modification to be performed (blink or frame), is inserted into the modification database. The modification database is implemented as a binary search tree, the icon-id serving as the sort key.

As each icon record is inserted into the mod database, the following processing takes place: For each iplane in the icon's I-space, compute the iplane coordinates and extents of the icon. Compute which tiles are affected in that iplane. For each affected tile in the iplane, compute the bounds of the area affected within the tile. Load the affected tile from disk, and for each pixel in the region to be modified, turn the blink/frame bit on if the value of the pixel is not equal to the color stored for that icon. The affected tile is then written back to disk.

When all tiles for a particular iplane have been modified, release the work buffers (in which the modified tiles have been loaded). This has the effect of updating the main display and navigational aid if the any of the modified tiles are currently visible.

3.2.2.2 Unblink/Unframe

When an unblink or unframe command is issued, the SQUEL interpreter passes the modification process a list of the affected icon-ids. The modification process must look up each of these icon-ids in the modification database. Processing then occurs in a similar fashion to blinking and framing described above. However, rather than turning the blink/frame bit on in each pixel, the blink/frame bit is turned off.

3.2.2.3 SDMS Termination

When SDMS is halted via a 'quit' command, the modification process is notified via a special signal. The modification process catches this signal, and invokes a special routine to 'clean up' the graphical data space. Any modifications which have not been undone by an unblink or unframe are cleared from the GDS before the modification process terminates. This insures that no icons are unintentionally left modified.

3.2.3 Blinker

The blinking process is responsible for changing the video lookup table (vlt) in the main and navigational aid displays at a regular interval. The effect of changing the vlt periodically is to cause those pixels on the screen which index that part of the vlt which is changing to appear to blink. N times a second (where N has been determined empirically) the blinking process toggles the portions of the vlts in the main and navigational aid displays which are indexed by the blinking colors.

The rationale for controlling the blinking in a separate process is that a more regular period between blinks can be obtained. If blinking were controlled by the Stager, the rate would be dependent on user motion, etc.

3.3 Data Structures

The modification process keeps a list of icons that have modifications outstanding on them. The list is actually implemented as a binary tree, the icon-id serving as the sort key. This allows rapid and efficient access to any modification record in the case of deletion. This list remains in core at all times. The following items are stored in each modification record:

- *icon-id*
- operation to be performed
- color to be modified (ignored for frames)
- X and Y origin of icon
- width and height of icon
- I-space of icon
- link to left subtree
- link to right subtree

4. Viewing Text in SDMS

As discussed in the previous QTR [Herot et al B], SDMS may be used to view online textual information such as reports, system documentation, etc. In the graphical data space, text documents are represented by special icons, termed "text ports." Zooming in on a text port displays the associated document -- the document appears on the center SDMS display, and a table of contents, if available, appears on the right display.

The user can scroll through the document by moving the joystick. The table of contents serves two purposes:

1. A cursor serves to locate the section of the document currently displayed.

2. The table of contents can be used for "rapid transit" to another section in the document.

Two enhancements have been made to the text viewing facility during the period covered by this report: an anti-aliased two-bits-per-pixel text font has been introduced to improve text readability; and the method of

rapid transit movement through the document has been improved.

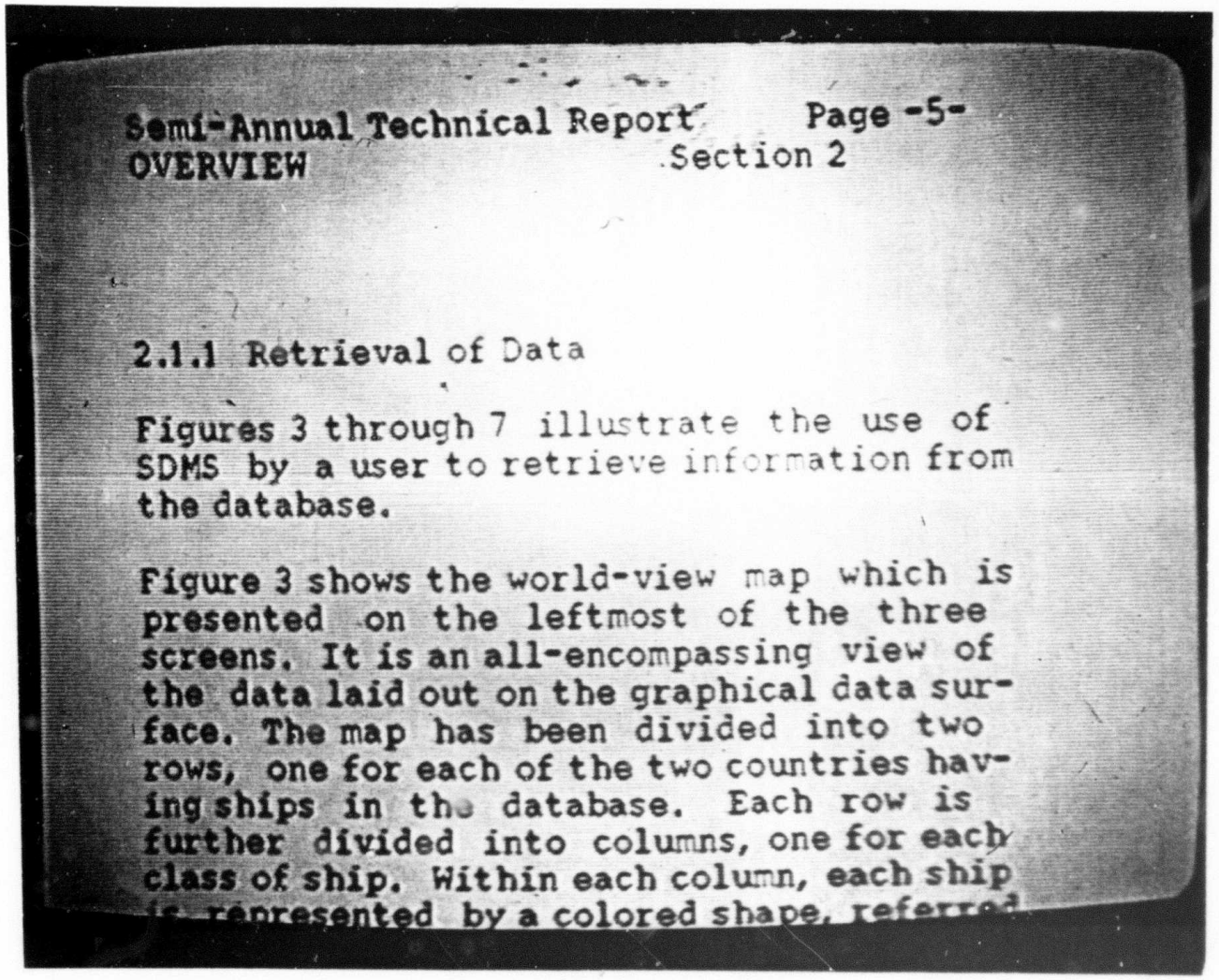
4.1 Anti-aliased Text Font

Within the context of computer graphics, the term "aliasing" refers to the jagged appearance of synthetic images. The effect is particularly noticeable in computer-generated images of smooth, curved lines and surfaces. Consequently, this problem is encountered when attempting to render images of smooth, easily readable text fonts.

Aliasing results from determining the color of an image area from a point sampling of the image's ideal representation. In practice, the human eye can be fooled into "seeing" smooth curved lines when each pixel in the synthetic image is shaded as a function of the appropriate area sampling of the ideal representation.

A text font which approximates this kind of shading function to two bits (four grey levels) per pixel is now used to view text in SDMS (see Figure 4.1).

Figure 4.1



4.2 Rapid Transit in Text

As already mentioned, if a text document has an associated table of contents, it will be displayed on the right monitor when the document is being viewed. A cursor on the table of contents indicates which section is currently displayed on the center screen.

To invoke rapid transit, the data tablet puck is used to position the cursor over the section of interest. When the top button on the puck is depressed, that section of the document is displayed on the center screen.

Along with the table of contents, the right monitor displays the following four text menu selections (see Figure 4.2):

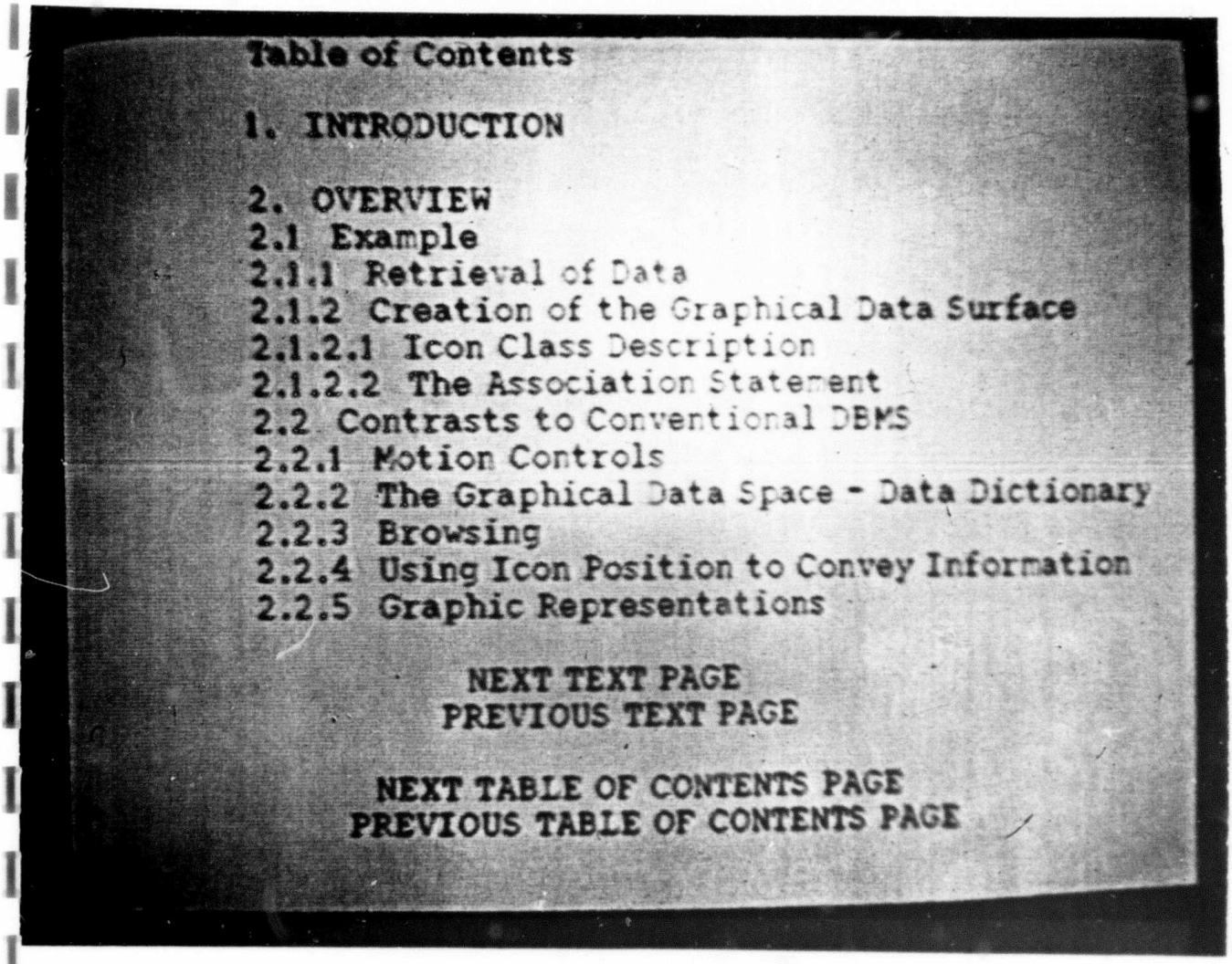
1. next text page

2. previous text page

3. next table of contents page -- this feature is used to page through the table of contents if it is too long to fit on the screen.

4. previous table of contents page

Figure 4.2



5. References

[BBN]
BBN extension of: Thompson, K. and Ritchie, D. M., "UNIX Programmer's Manual." Bolt, Beranek, and Newman, Inc., 50 Moulton Street, Cambridge, Massachusetts 02138.

[BILOFSKI]
Bilofsky, Walter, "The CRT Text Editor NED -- Introduction and Reference Manual." R-2176-ARPA, The Rand Corporation, Santa Monica, California 90406.

[CHAMBERLIN]
Chamberlin, Donald D., "Relational Data-Base Management Systems." ACM Computing Surveys Vol. 8 No. 1 (March, 1976).

[FRIEDELL SCHMOLZE]
Friedell, Mark and Schmolze, Jim, "Icon Class Description Language Reference Manual." Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[HELD STONEBRAKER WONG]
Held, G. D., Stonebraker, M. R. and Wong, E., "INGRES -- A relational data base system." AFIPS Proceedings Vol. 44.

[HEROT et a. A]
Herot, Christopher F., Carling, Richard T., Friedell, Mark, Kramlich, David and Thompson, John, "Spatial Data Management System: Semi-Annual Technical Report." Technical Report CCA-79-25, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139

[HEROT et al B]
Herot, Christopher F., Kramlich, David, Carling, Richard, Friedell, Mark and Thompson, John, "Quarterly Research and Development Technical Report, Spatial Data Management System, 1 March

1979 - 31 May 1979." Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[Johnson]
Johnson, Stephen C., "YACC -- Yet Another Compiler Compiler." in Documents for use with the UNIX Time-Sharing System, Sixth Edition, Bolt, Beranek, and Newman, Inc., 50 Moulton Street, Cambridge, Massachusetts 02138.

[YOUSSEFI et al]
Youssefi, K., Whyte, N., Ubell, M., Ries, D., Hawthorn, B., Epstein, B., Berman, R. and Allman, E., "INGRES Reference Manual -- Version 6.1." Electronics Research Laboratory, College of Engineering, University of California, Berkeley, California 94720.