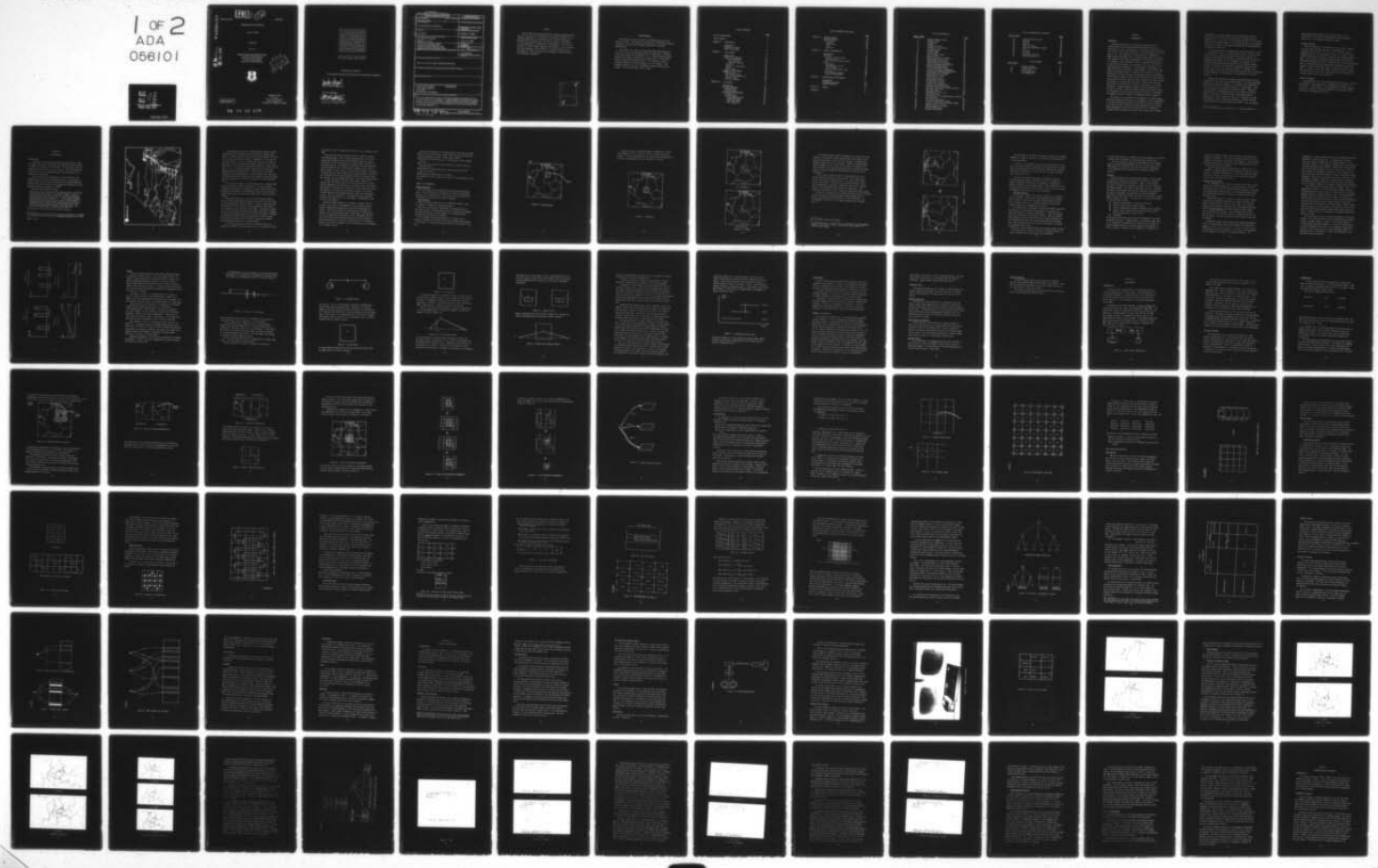
AD-A056 101 MITRE CORP BEDFORD MASS F/G 14/5
GEOGRAPHIC DATA DISPLAY, (U)
JUN 78 D E BELL F19628-77-C-0001
UNCLASSIFIED MTR-3315 ESD-TR-77-362 NL

1 OF 2
ADA
056101



AD A056101

AD No. ____
DDC FILE COPY

LEVEL II (12)

ESD-TR-77-362

MTR-3315

GEOGRAPHIC DATA DISPLAY

BY D.E. BELL

JUNE 1978

Prepared for

DEPUTY FOR DEVELOPMENT PLANS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

DDC
RECEIVED
JUL 10 1978
F



Approved for public release;
distribution unlimited.

Project No. 7090
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-77-C-0001

78 07 06 010

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

Jack Segal

JACK SEGAL, GS-14
Project Engineer/Scientist

FOR THE COMMANDER

W. SMITH, Colonel, USAF
Director of Advanced Planning
Deputy for Development Plans

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ESD-TR-77-362	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): GEOGRAPHIC DATA DISPLAY	5. TYPE OF REPORT & PERIOD COVERED
6. PERFORMING ORG. REPORT NUMBER: MTR-3315	
7. AUTHOR(s): D. E. Bell	8. CONTRACT OR GRANT NUMBER(s): F19628-77-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, Box 208, Bedford, MA 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 7090
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Development Plans, Electronic Systems Division, AFSC, Hanscom Air Force Base, MA 01731	12. REPORT DATE: JUNE 1978
	13. NUMBER OF PAGES: 100
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMPUTER SOFTWARE
COMPUTER GRAPHICS
DATA BASE
DISTRIBUTED PROCESSING
GEOGRAPHY

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report addresses the problem of providing a geographic background for the overlay of real-time C^3 information. A general design is presented which includes automatic control of detail, selection of display features and an overlay capability for event data (real-time data). A brief description of the current implementation of the design is included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

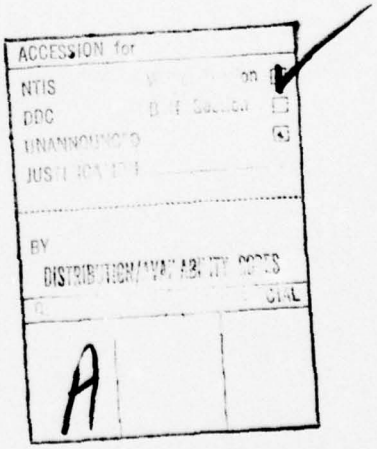
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

78 07 06 010
235 450

PREFACE

Project 7090, Operations/Intelligence Techniques Experimentation, has as an objective the development of automated techniques for the effective application of intelligence data. Because information derived from intelligence data is often positional, data display on a map background is a major mode of presentation. The focus of this part of Project 7090, Geographic Data Display, is the provision of a map background for the display of positional data over the wide range of scales necessary for the analysis of various classes of intelligence information.



ACKNOWLEDGMENTS

This report has been prepared by The MITRE Corporation under Project No. 7090. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

R.H. Bullen, Jr., and H. A. Bayard made major contributions to this report in the initial planning phase of the effort. Similarly in the final stages of documentation, M. J. Brooks supplied valuable editorial and textual criticism of the report itself. The initial burden of typing this material fell to P. M. Antonuccio; final, iterative typing of the final document, to C. A. Sullivan. Both responded beautifully to the unreasonable demands inherent in this document. My sincere thanks to all of them.

TABLE OF CONTENTS

		Page
LIST OF ILLUSTRATIONS		5
LIST OF TABLES		5
SECTION I	INTRODUCTION	7
	BACKGROUND	7
	REQUIRED SOLUTION	9
	PLAN OF THE REPORT	9
SECTION II	A GDD CONCEPT	10
	INTRODUCTION	10
	ISSUES TO BE ADDRESSED	14
	System Interfaces	14
	User Functions	14
	External Interface	20
	Features	21
	Parallel Vs. Monolithic	22
	Detail	25
	System Speed	32
	SUMMARY OF THE CONCEPT	32
	TECHNICAL GOALS	33
	GDB Data Structure	33
	GDB Storage and Retrieval	33
	Optional Detail	33
	Parallel Features	34
SECTION III	A GDD DESIGN	35
	INTRODUCTION	35
	GDB DATA STRUCTURE	36
	Neighborhoods	37
	Data Structure	45
	GDB STORAGE AND RETRIEVAL	49
	Introduction	49
	Storage Schemes	51
	Pseudo-Matrix Method	51
	Speed-Over-Space	53
	Space-Over-Speed	55
	Cost Comparison	63

TABLE OF CONTENTS (Concluded)

		Page
SECTION III	OPTIONAL DETAIL	65
	PARALLEL FEATURES	65
	DESIGN PRECIS	68
	Location	68
	Translation	69
	Zoom	69
	Selection	69
SECTION IV	CURRENT IMPLEMENTATION	70
	INTRODUCTION	70
	DATA BASE	70
	Content of the Data Base	70
	Internal Format	71
	The Creation of Detail Levels	72
	SYSTEM	72
	Environment	72
	Implementation Itself	74
	Logic Package	78
	Location, Translation, Zoom	78
	Selection	82
	Map Management Package	90
	Areas for Improvement	91
SECTION V	EXTRAPOLATION OF THE DESIGN	93
	INTRODUCTION	93
	INCREMENTAL EXPANSION	93
	QUANTUM EXPANSION	94
SECTION VI	SUMMARY	96
REFERENCES		97

LIST OF ILLUSTRATIONS

Figure Number		Page
1	Operation Market-Garden	11
2	Display Window	15
3	Translation	16
4	Zooming	17
5	Feature Selection	19
6	Schematic of a Telescope	26
7	An Example Image	27
8	Blurred Image	27
9	Resolved Discs	28
10	A Line-Segment Object	28
11	Region R Twice	29
12	Region with Stronger Primary	29
13	Overlapping Detail Levels	31
14	General GDDS Organization	35
15	Superfluous Map Calculations	38
16	Window Straddling Neighborhoods	39
17	Overlapped Neighborhoods	40
18	Shared Neighborhood Blocks	40
19	Locus of the Window Centerpoint	41
20	Translation Neighborhood Management	42
21	Zoom Neighborhood Management	43
22	Single GDB Data Structure	44
23	Neighborhood with Hub	47
24	Block Naming Scheme	47
25	Neighborhoods and Hubs	48
26	Pseudo-Matrix Storage	50
27	Slanted Matrix Storage	52
28	Storage of a Neighborhood	53
29	Speed-Over-Space Map Storage	54
30	Secondary Storage Using Vertical Strips	56
31	Data Base Index Entry	57
32	Data Block Index	58
33	Neighborhood of Hub (i,j)	58
34	Four DBI Entries	59
35	The Overlap of the Neighborhoods of Hub(i,j) and Hub(i,j+1)	60
36	Data Base Index Index	62
37	The Overall Management Scheme	62
38	A GDB's Pointers to the Universal Index	66
39	Optional Detail Pointers	66
40	GDBs Pointing to the Index	67
41	System Configuration	73
42	Physical Environment	75

LIST OF ILLUSTRATIONS (concluded)

Figure Number		Page
43	Layout of Function Keys	76
44	Translate	77
45	Zoom In	79
46	Zoom Out	80
47	Shift of Context	81
48	Functional Breakdown of GDDS	83
49	Menu	84
50	Selection	85
51	Deletion	87
52	Rescinding Selection	89
53	Rescinding Deletion	89

LIST OF TABLES

Table Number		Page
I	Storage Tradeoff	24
II	Display Calculation	24
III	Access Time	24
IV	Data Manipulation	24
V	Cost Comparison	64

SECTION I

INTRODUCTION

BACKGROUND

Effective utilization of intelligence information in a C^3 context requires that the data-gathering and data-utilization functions be in tune. Specifically, the timeliness and perishability of a class of data should correspond to the time required to put that data to use. Two examples will illustrate this correspondence. Radar systems that gather data on moving aircraft must have displays that operate in real-time: responsiveness of the display system is at a premium. Since geographic detail is relatively unimportant to an air situation display, it is provided, if at all, only for high level geopolitical boundaries. Ground situation intelligence as practiced heretofore provides a second example. In this case, high quality cartographic detail is required of the display, but rapid processing of the data has been less important because traditionally the situation changed relatively slowly and because the data gathered on the ground situation could have been relatively old (hours to days old) before it was available for display. Thus the use of manual plot boards was formerly well suited for the display of the ground situation. As these examples illustrate, where a homogeneous set of data is involved, a special purpose display system can be devised to accommodate the special characteristics of the data, maximizing, for example, speed of display or cartographic detail as required.

Two aspects of this simplified intelligence cycle have changed, affecting the proper utilization of intelligence. First, improvements in the data gathering function have made data available in minutes or seconds that have traditionally been available in hours or days. These improvements cast doubt on the ability of the associated display systems to handle the new data rates, doubt that is indeed

justifiable. The second change is the perceived need to "fuse" data from diverse sources and to display the combined data on a single system. This fusion of data requires the best of all possible worlds: the time-sensitiveness of radar-like systems, the cartographic detail of plot boards, and a general flexibility in the manipulation and display of input data.

The starting point in considering the altered intelligence cycle is the possible adaption of current display systems. The most time-efficient display systems are radar-like systems. However, the dearth in such systems of cartographic capabilities and the system's consequent inapplicability to, for example, the display of the ground situation argue against adaption of air situation display systems. Conversely, the excellent cartographic features of plot boards cannot be used in real-time contexts. In addition, both display systems are relatively rigid, affording very little flexibility in user alteration of the display or in the addition of new data sources.

Attempts have been made to combine the responsiveness of radar displays with high quality cartographic detail, as in the Bunker-Ramo hybrid system called the BR-90*. On the BR-90, a map background is provided by a map slide back-projected onto a radar display. This is an improvement, but intrinsic limitations remain. Additional inputs cannot be accommodated directly; the selection of maps is limited by the size of the slide tray; no map manipulation is possible; and context changes are very slow. Generally, the same inflexibility found in radar-like systems and in manual plotting systems remains in hybrid systems like the BR-90. Thus, it appears

*Further information about the BR-90 can be found in Reference 5.

that current display systems will not adapt easily to new requirements of flexibility, responsiveness, and cartographic support in a C^3 intelligence utilization context.

REQUIRED SOLUTION

What is required is an environment for the selective display of various real-time data sources on a map background. In the remainder of this document, we will refer to this environment as a Geographic Data Display Environment or GDDE. We intend to supply a GDDE by defining a new system - Geographic Data Display System or GDDS - that performs the Geographic Data Display (GDD) function. The general characteristics of an acceptable GDDS have been highlighted above. A GDDS should provide appropriate background maps and the ability to overlay arbitrarily chosen classes of input data on the maps. In order to cope with volumes of input data and to perform the GDD function in real time, a GDDS must react very rapidly both to input stimuli and to user interaction.

PLAN OF THE REPORT

The remainder of this report addresses the GDD function. Section II introduces a GDD concept which is expanded in Section III to a full design. A description of an implementation of a GDDS comprises Section IV. Section V discusses extensions to the current effort, and Section VI summarizes the report.

SECTION II

A GDD CONCEPT

INTRODUCTION

In Section I, we defined our problem as the design of a digital GDDS with a general background map capability and the ability to display an arbitrary set of overlays. In this section, we will elaborate the basic issues in such a design. From that discussion we will derive a list of technical goals that collectively circumscribe our GDD concept.

We will motivate our discussion of our GDD concept by considering the use to which our GDDS might be put. In particular, we will review the World War II operation called Market-Garden and determine what characteristics the planning and supervision of that operation would have required of a GDDE.[1*]

Operation Market-Garden utilized

> three and a half divisions--the U.S. 82nd and 101st, the British 1st Airborne and the Polish 1st Parachute Brigade. The airborne forces were to seize a succession of river crossings in Holland ahead of [the] troops, with the major objective being the Lower Rhine bridge at Arnhem. . . . The surprise airborne attack would open a corridor for the tanks of [Field Marshal Montgomery's] British Second Army, which would race across the captured bridges to Arnhem, over the Rhine and beyond. Once all this was accomplished, Montgomery could wheel east, outflank the Siegfried Line, and dash into the Ruhr.[1**]

The basic plan of the operation is pictured in Figure 1.

---

*The information presented here on Operation Market-Garden is abridged and excerpted from Cornelius Ryan's A Bridge Too Far (New York, 1974).

**Ibid., p. 88.

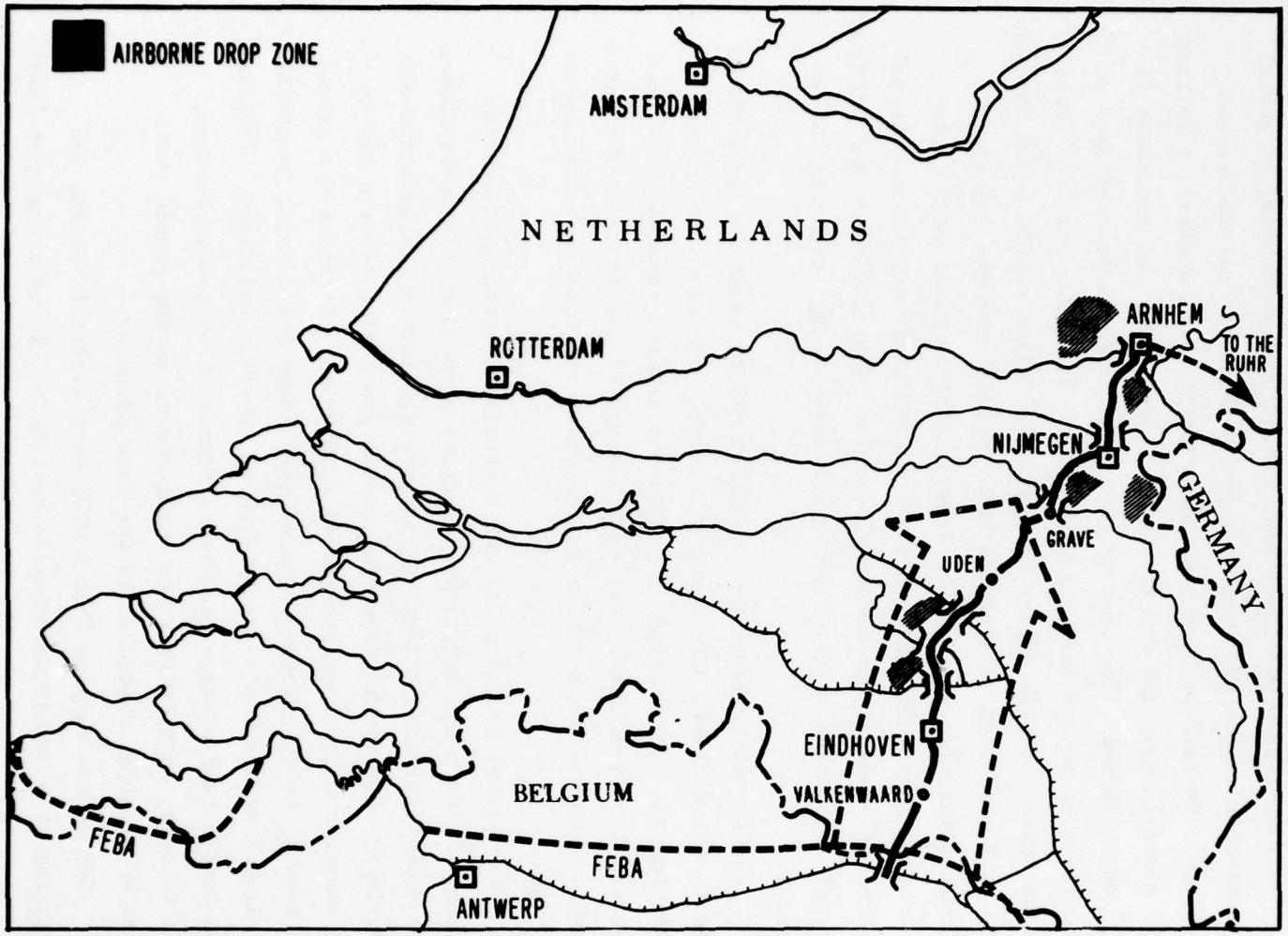


Figure 1. Operation Market-Garden

The operation was bipartite, Operation Market being an aerial paratrooper attack and Operation Garden being an armored ground advance. Market entailed a massive airlift from England to Holland. Its overall objective was to secure a passageway for the tanks involved in Garden. The foci of Market were the various bridges on the route up to and including the bridge at Arnhem over the Lower Rhine River. The objective of Operation Garden was to break through the F. E. B. A. in Holland and move rapidly into the important German industrial region, the Ruhr. The operations were interdependent in the following sense. Garden's advance was predicated on Market's suecessful occupation of the important roads and bridges; relief for Market units would be provided by the advancing Garden forces.

The top-level planning for Market-Garden dealt in large terms: fleets of aircraft, divisions, and corps. The scope of Market planning was aeronautical, covering the several hundred miles from England to mid-Holland. The tank advance of Garden covered about a hundred miles along a single main roadway which crossed five major bridges.

Divisional planning for the operation covered much smaller areas. Primary and sometimes secondary drop zones had to be chosen for Market paratroops (shown in Figure 1). Terrain, vegetation and proximity both to the objective and to population centers had to be considered in the selection of drop zones. Planning the ground action of the paratroops required very large scale maps, including at Arnhem street-map features such as specific buildings. Planning the Garden tank advance required information on roads, bridges, and terrain, particularly the locations of marshy ground which were substantial obstacles to the tank advance.

Let us now suppose that a GDDE were available during the planning of a Market-Garden-like operation. To be useful--at least

as useful as a set of charts--what would have to be available in the GDDE?

There would be first the obvious need to be able to display and manipulate maps. Some of these maps would have to cover the large area encompassed by the airborne Market operation; others would have to cover the much more restricted areas of interest of the ground battles. The planning of different component operations required different sets of map features--bridges, roads, canals, dikes, marshes, forests, and city streets and buildings, for example. The GDDE would have had to supply these various features, preferably in a selective way so that the planners could have used only those map features that they deemed important to their particular operation.

We have concluded so far that a GDDE needs the ability to display maps over a wide range of scale and the ability to present various combinations of map features at the discretion of the user. Also needed is the ability to overlay force structures--the location and movement of friendly and enemy military units. The overlay of enemy positions and strengths requires input from intelligence sources. Hence the GDDE must be integrated with other planning and monitoring adjuncts such as intelligence, reconnaisance, order of battle, and sensors.

Operation Market-Garden, for a variety of reasons, did not go well. As a result, the management of the operation involved rapidly changing plans improvised to deal with unexpected situations. The use of a GDDE in the monitoring of a dynamic situation, such as Market-Garden became, would require the same flexibility in manipulation and display of both cartographic and military data that we have mentioned under severe time constraints: system responsiveness to the user's needs would be of a paramount importance so that the commander would be helped rather than hindered in his command function.

This brief description of a representative use to which our GDDE might be put has illustrated several issues that must be addressed in the development of a GDDS. These issues include:

- the system's interfaces, both to the user and to other command systems;
- the fact that a number of map features are needed to make the displays useful;
- the problem of providing useful, complete maps over a wide range of scale;and
- the system's responsiveness--its "speed."

Our resolution of these issues is the topic of the remainder of this section.

ISSUES TO BE ADDRESSED

System Interfaces

The first issue we will address involves the GDD system interfaces. We are defining how the GDDS will fit into an operational setting. We will begin with its interaction with a user and conclude with its interaction with other C^2 systems.

User Functions

There are four user functions that a GDDS must provide. They are location, translation, zoom and selection.

Location is the process of displaying a certain area of the map with overlays. Location would be used to initialize a planning or monitoring display for Market-Garden or to change contexts radically from one position to another.

If one considers the displayed portion of the map to be a square of variable side situated on the map as shown, then location is the process of centering the display "window" on any point of the map.

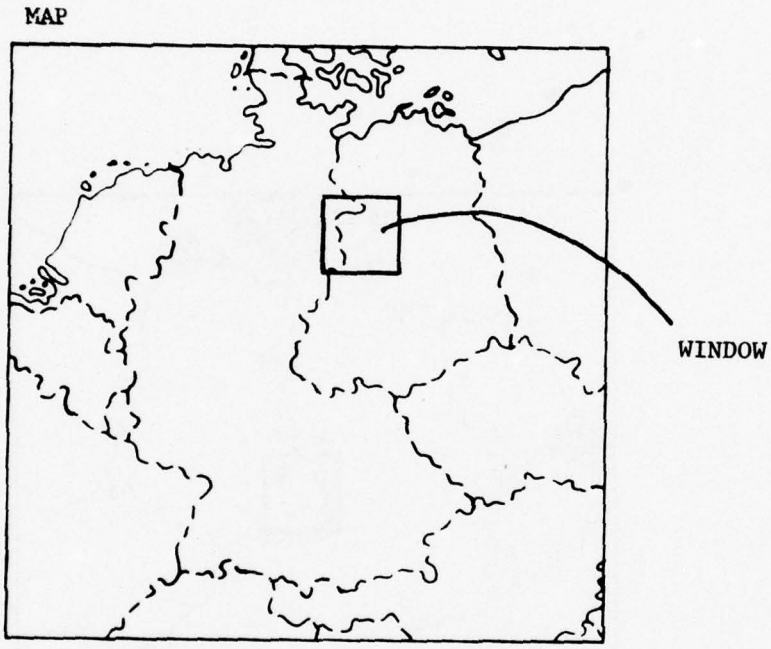


Figure 2. Display Window

Translation moves the display window incrementally. Translation might be used to follow the action in Market-Garden monitoring. A cursor superimposed on the window specifies what point will be the centerpoint after the translation, as shown.

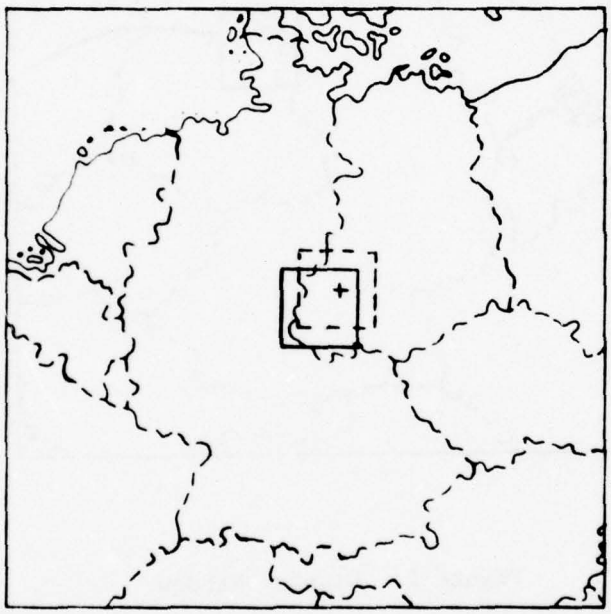
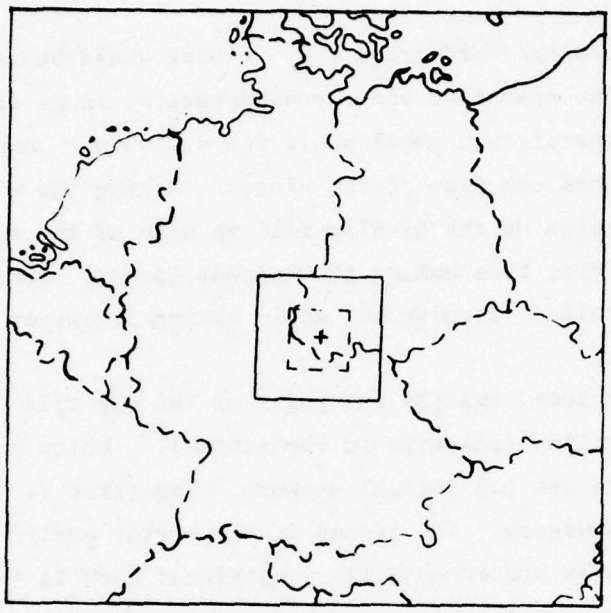
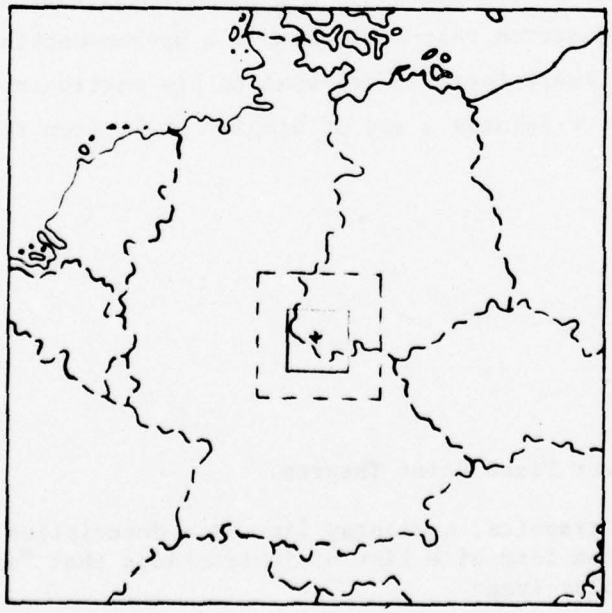


Figure 3. Translation



(a) Zooming In



(b) Zooming Out

Figure 4. Zooming

In the Market-Garden context, zooming would be used to fit the display to the operation under consideration, large scale for company-level operations, small scale for division - and army-level. Zooming changes the size of the window. Making the window smaller makes the images on the display fill up more of the display than they did before, thus making them appear larger. Making the window smaller is called "zooming in" while making it larger is called "zooming out."

After a zoom, exactly one point of the map will be in its original position (relative to the window).* Which point should it be? There are two logical answers. The first is the center point of the window. The second is the cursor position. Clearly the two methods are essentially equivalent: each is the concatenation of two translations and a zoom of the other type. For purposes of discussion, we will assume the centerpoint remains fixed.

Selection is fundamentally a parameter-altering function. Selection is the user function that would enable a Market-Garden user to display only those features relevant to his particular use. Selection adds or deletes a set of display items from the "display list."**

---

* By the Brouwer Fixed Point Theorem.

** In computer graphics, a display list is a description of the graphics display in the form of a list of instructions that "draws" the desired display image.

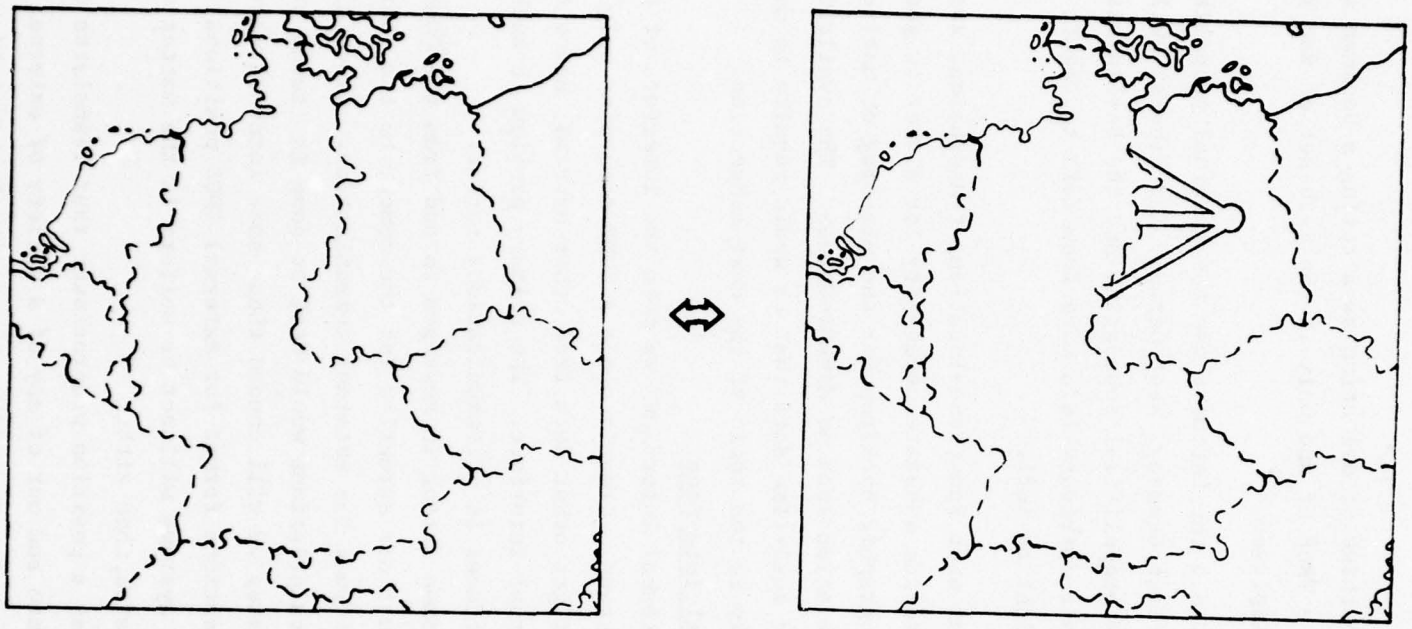


Figure 5. Feature Selection

One can think of selection as altering a Boolean array DISPLAY: DISPLAY (i) = TRUE if and only if the i-th set of map features is currently displayed.

In Figure 5 the leftmost map has external geopolitical boundaries only (East Germany, West Germany and others) while the rightmost one has geopolitical boundaries and the Berlin air corridors. Adding the air corridors is a move from left to right; deleting them from right to left.

The four user functions-location, translation, zoom and selection-provide adequate capability for a user to generate a useful cartographic backdrop for the planning or monitoring of a military operation such as Market-Garden. The availability of the archival and real-time data that he would require to use the system intelligently is the topic of the next subsection.

External Interface

By "external interface" we mean the interface of the GDDS to other C^3 systems. Clearly since a GDDE maintains only geographic data it must get other data from other systems. Hence the need for an external interface. The primary problem in such inter-system interfaces is a communications format.

The common factor in messages to and from the GDDS is position. We will direct our attention for the moment to the choice of a positional format for external communication. Since all internal positional calculations would best be done in terms of internal map coordinates we will choose these same internal coordinates as the communications format for external GDDS positional messages. Since other systems will not be uniform in the matter of internal map coordinates, then either

- GDDS has a position preprocessor that translates GDDS internal form into and out of any of a variety of external formats, or
- the communications link provides such translation.

In either case there is a necessity for the development of conversion routines that will operate on positional data in real time.

The total external interface between GDDS and other command systems is application-dependent and cannot be specified further. When systems are connected, there must be careful consideration of the interface and its inclusion of conversion routines for position.

Features

As was mentioned in the Market-Garden example, a large number of diverse map features are potentially useful in the planning and monitoring of military operations. Many of these features are static and should be available in a GDDS. But which features? The answer depends on the specific application at hand. Planning the Market airlift might have required coastlines, user-specified drop zones, topography, and enemy anti-aircraft emplacements. Planning the Garden tank assault would have required roads, marshes, dikes, bridges, and canals. The total list of features that might be required would include:

- geopolitical boundaries, internal and external;
- places (cities, towns, military bases);
- lines of communication (roads, railroads, rivers, canals);
- special lines of demarcation (like the Berlin corridor, military districts, air routes); and
- topography.

The features listed above would provide a background for the user performing some task. A Market planner, for instance, might place on his display map representations of enemy forces to assess alternate dropping patterns, calculate distances from take-off to drop to construct a schedule for his squadron, or review his re-

sources in an Order of Battle file to allocate them optimally to achieve his objective. This other information will be the "foreground" on his display, the focus of his attention. The foreground data must be able to be overlaid on the background map data which the user has selected for himself.

Clearly then the general requirement of a GDDS is for the capability to manage and display an indeterminate number of display features. Specific requirements of a particular GDDS--the number and type of features required--are totally application-dependent and will not be treated further here.

Parallel Vs. Monolithic

The requirement that a user be allowed to select which feature data will be displayed admits two general solutions. The first method would be to store the various features separately, combining the required data at display time. The second method would store all the data monolithically, deleting the unwanted feature data at display time.

The parallel method of storage will require separate storage locations, but probably not more total storage, since the same number of points will be stored in either case. The recall of display will be driven by a Boolean array (the DISPLAY array mentioned in User Functions above) which specifies which features will be used and which ignored. The calculation will require little time, but the several accesses to secondary store will impede the system. The more features desired, the more access time is required.

Monolithic storage could be effected by tagging each display point or by maintaining a feature-keyed index to the data base. Tagging a datum so that its feature is included will require extra storage space, the amount depending on the size of the tag (which in turn depends on the number of features available). The

maintenance of a keyed index will require only additional internal delimiters, a small storage cost. In either case, a Boolean array will figure in display but at a different point. In all monolithic schemes, all the data is accessed after which deletions are made based on the array DISPLAY. Access time is constant and equal to the time required to access all feature data. There is also calculation time needed for deletion. In the tagging case, every point must be checked no matter how many features are being displayed. Using a key, the deletion process is used only on the nondisplayed features. Hence with a feature index, the sparser the display the more extensive the deletion process: calculation time is inversely related to the number of features being displayed.

The trade-offs parallel and monolithic (tagged or keyed) are summarized in Tables I - IV. Total storage is roughly constant. The determination of which features to display (that is, checking DISPLAY) requires a constant amount of time. The parallel method requires additional accesses for each feature to be displayed; thus the total access time increases as more features are displayed. The monolithic methods require a single access and thus have a constant access time whatever the display clutter. Time to manipulate data (that is, delete points) is constant near zero for the parallel method; the monolithic methods require some calculation time, a constant amount with tagging and a decreasing amount with keying, as shown.

We have decided to pursue a parallel development, for several reasons. One is the obviation of extra calculation time, helping our goal of rapid response. Another is the cleaner design of storing separate features separately. Lastly, the complications of storing features monolithically, not to mention the problem of adding to a monolithic data base, seemed formidable indeed, engendering in us a strong desire to reduce the complexity of the problem by using parallelism.

Table I.

Storage Tradeoff

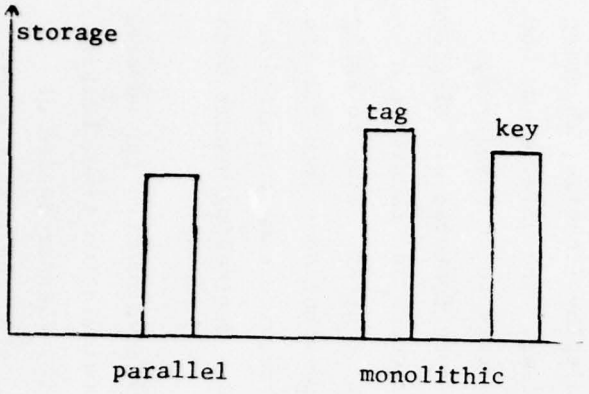


Table II.

Display Calculation

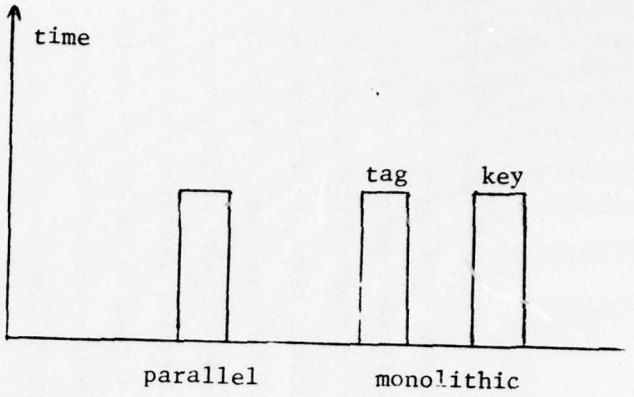


Table III.

Access Time

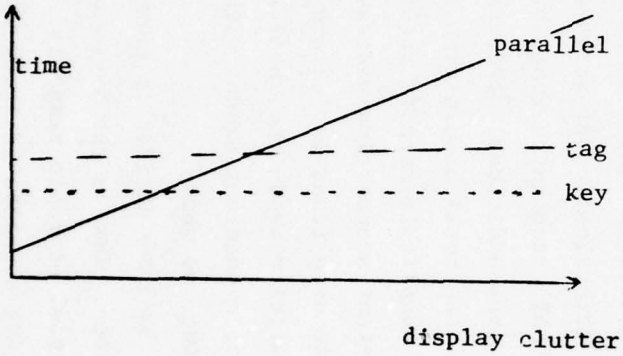
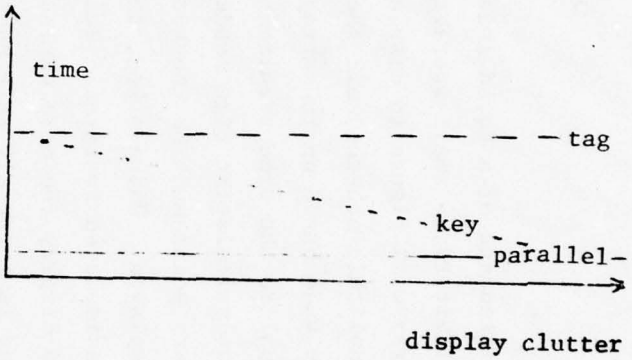


Table IV.

Data Manipulation



Detail

One of the difficulties with current map background systems is the lack of detail control. Over a wide range of scale, the amount of detail provided varies substantially and in some cases the map background disappears altogether. We want our GDDS to supply a map background that always provides a frame of reference without cluttering up the display. To this end, we must determine how to provide a map background that maintains an appropriate amount of detail on its display.

In general, a detail is a constituent part of a thing - a nose is a detail of a face. More specifically, a detail of an image is an easily identifiable and seeable part of the image. For our purposes, a "detail" will be a resolvable detail of a map image.

Justification for the inclusion of the adjective "resolvable" follows from the pseudo-optical nature of a GDD system. A GDD replaces maps of various kinds. A map can be considered the visible image of actual countryside, overlaid with artificial boundaries, symbols, and various alphanumerics. Hence, one can view a GDD as creating an image of some reality (a line map) from some fixed position using optical devices such as telescopes.

This analogy of a GDD to a telescopic system can be very instructive. A telescope produces an enlarged image using two elements (for our purposes, two lenses) placed in tandem (see Figure 6). Conceptually, the image is produced in two steps: the primary lens bends the light rays forming an intermediate "primary image," which acts as an object for the eyepiece.

Additional enlargement of the image is accomplished by changing the eyepiece. However, this sort of magnification cannot increase the detail that can be seen:

"...increasing the size of the image by increasing the power of the eyepiece does not increase the amount of detail that can be seen, since it is impossible by magnification to bring out detail which is not originally present in the primary image."[2]

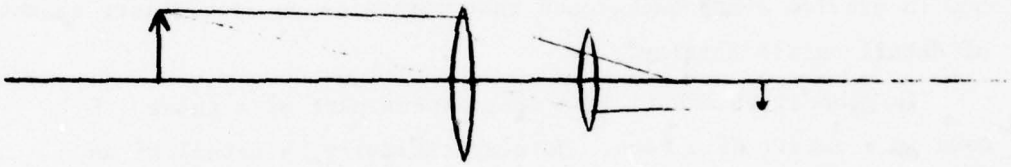


Figure 6. Schematic of a Telescope

Thus, the detail that can be seen is dependent on the quality of the primary image and the magnification of the eyepiece.

The basic map data in a GDDS provides the primary image. Software zoom provides magnification. Thus, zoom capability on a single map is like many eyepieces for one telescope. If the size of the field of view varies over a wide range, one would be advised to use several primary lenses (that is, several maps), each capable of providing a primary image with detail.

How does one decide what detail is appropriate in a given range of use? The answer again is pseudo-optical.

Consider two discs of radius r separated by a distance d.

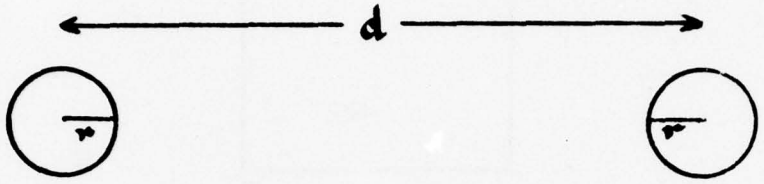


Figure 7. An Example Image

As long as d > 2r, it is theoretically possible to differentiate the two discs with an optical system. However, if the primary images of the discs are point-sources with respect to the eyepiece of a telescope, then diffraction causes a blurring of the two images. Thus, in this case, if the image separation is less than the amount of blurring, the two discs will not be resolvable (that is, differentiable) and will appear as shown:

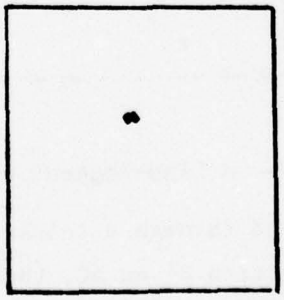


Figure 8. Blurred Image

If a more powerful primary lens could resolve the two discs, then the image would be as shown in Figure 9:

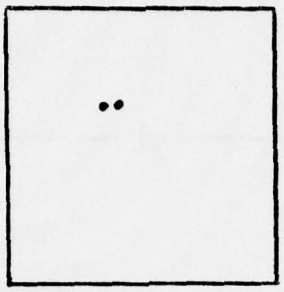


Figure 9. Resolved Discs

Any display medium has a granularity--plots on matrix, raster-scan devices for example, can be no more accurate than the image of one point. Attempting to display two spots with their centers separated by a distance of one half-point will result in a blurred image: those two points cannot be resolved by the device.

How will we relate resolvable points to details "appropriate" to a certain magnification? Consider the image of the two line segments as shown in Figure 10:

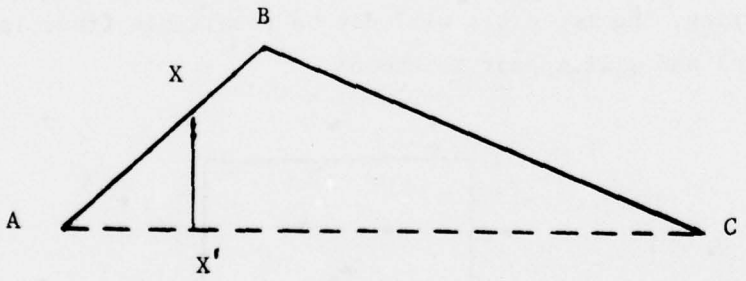


Figure 10. A Line-Segment Object

If when the image is viewed through a telescope, no point X is resolvable from its projection X' on AC, then the two segments are indistinguishable from the line AC. Suppose AB and BC are part of the boundary of a region R. The detail of the corner B is not resolvable in this system. Similarly, the inclusion of the triangle

ABC within R is not noticeable. Hence, replacing AB and BC with AC in R's image, no matter what power eyepiece (that is, no matter how much magnification is used), will not alter R's appearance noticeably.

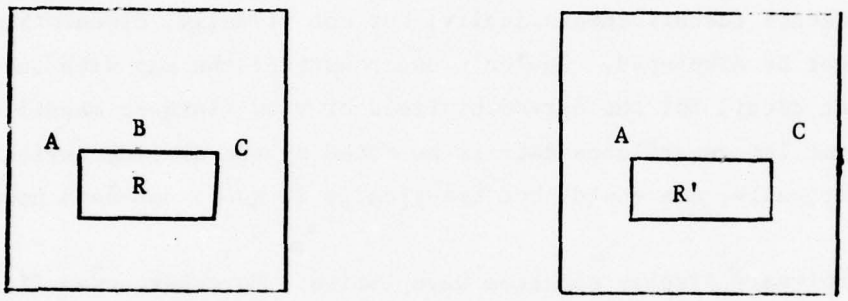


Figure 11. Region R Twice

Using a telescope with a more powerful primary lens can make the corner B resolvable and the inclusion of ABC noticeable.

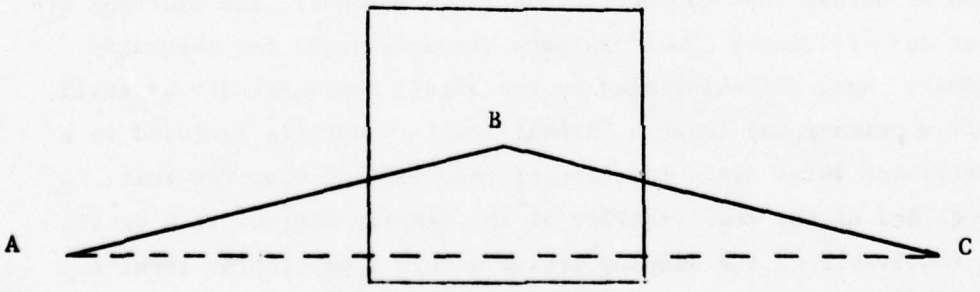


Figure 12. Region with Stronger Primary

Clearly, the resolution of a display device will limit the noticeable detail in an exactly analogous way.

To sum up, a detail in a GDD is an optical detail (corner, edge, and so forth) that is resolvable on the display medium.

The question arises whether one should use separate maps for different fields of view. We have just determined that unresolvable detail (detail theoretically, but not visually, discernible) will not be displayed. Couldn't one construct the map with sufficient detail for the narrowest field of view (largest magnification) and just let superfluous details be muted by the display device? Theoretically, one could, but practically it would not be a good idea.

Software display packages have limits. Moreover, even if the number of points that can be manipulated is large, the calculation time necessary to process L points when only M << L are useful is indefensively wasteful. Furthermore, the blurring from which telescopic images suffer will be present in an image created from too many points, resulting in a noticeable loss of clarity.

The resolvability of the display medium of a GDD limits the size of detail that can be represented. Moreover, the blurring effect and efficiency considerations strongly argue for separate primary maps, differentiated by the detail represented. We shall call a primary map image a "detail level." Details included in a particular level are a function of the field of view for that level and of the resolvability of the display medium: if a detail is resolvable on the display medium within a particular level's field of view, then that detail is included in that level.

There is one further aspect of detail control to be mentioned. Its importance stems from the fact that at a single scale a given feature can be more or less important to a user. For a monitor of Operation Market, for example, political boundaries could be very unimportant faced with an aerial attack on his squadron. We

can easily imagine that a user might want to mute but not eliminate some feature of his map, political boundaries in our example. To deal with this need, we have decided to provide an optional amount of detail to the user. That is, for any display window, the user can see more detail or less detail, as he choses. The provision of this option is accomplished by overlapping the ranges of adjacent levels.

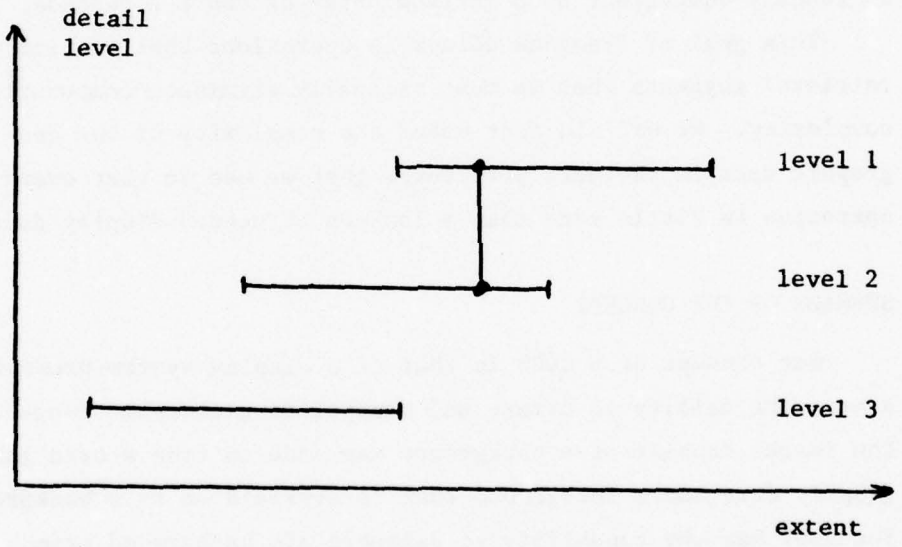


Figure 13. Overlapping Detail Levels

As shown in Figure 13, a user viewing the display window marked could see more detail by displaying the level 2 map and less detail by displaying level 1.

System Speed

Use of a GDDS in an operational setting requires very rapid response to user requests. In fact, as we mentioned in Section I, part of the stimulus for our effort is the inadequacy of current systems to provide rapid enough response in real-time C^2 contexts.

We want to provide the quickest possible response, that is, immediate response. In terms of a human user, immediate response is roughly equivalent to a maximum delay of about 2 seconds.

This goal of 2-second delays in operations that require data retrieval suggests that we must virtually eliminate computational complexity. We will in fact embed the complexity of our geographic data in the data structures that we use so that every operation is little more than a look-up of needed display data.

SUMMARY OF THE CONCEPT

Our concept of a GDDS is that of a display system presenting a user the ability to create and manipulate geographic images. The images consist of a background map made up from stored geographic data and a foreground that is overlaid on this background. The user has the capability to assemble his background from a list of available features by selecting those he wants to see. His selection affects a Boolean array DISPLAY which controls which of a set of parallel data bases is used in creating the background. The user can locate his display anywhere on the map, adjust the window by translating, and change the scale by zooming. Each user request causes a calculation of the new centerpoint and the length of the window (the "side") desired. If neither a scale threshold nor a lateral motion threshold has been passed, the current display is adjusted. Otherwise, a new display from the appropriate detail level is accessed for each feature currently being displayed.

These displays are merged to form a single background. The foreground data is overlaid on the background and the display is refreshed. Target response time is under two seconds.

TECHNICAL GOALS

The above description of our concept of a GDDS presumes that several problems are soluble. We will discuss these problems, phrasing each problem as a technical goal that we set for ourselves.

GDB Data Structure

The choice of a data structure for a geographic data base (GDB) is particularly important in the design of a GDD system, because of the severe time constraint involved (target response of two seconds). The specification of a data structure which accommodates both detail levels and geographical proximity with very rapid data access is one of the technical goals that one faces in the design of a GDDS.

GDB Storage and Retrieval

Closely related to the selection of a data structure for GDBs is the specification of a storage scheme which allows quick retrieval of data from secondary store. Several schemes are possible, the method chosen being a function of tradeoff between time and space. The description and analysis of various storage and retrieval schemes is a second major technical goal.

Optional Detail

As was mentioned in the Detail subsection above, we want to allow a user to control the amount of detail that he sees in a particular display. The exact mechanism for providing detail is a third technical goal of a GDD design.

Parallel Features

A fourth technical goal is the realization of parallel features using parallel GDBs and the Boolean array DISPLAY. This task includes the coordination of the several GDBs in reaction to a user request to alter the display.

These four technical goals provide the basis for our development of a GDD design from our GDD concept.

SECTION III

A GDD DESIGN

INTRODUCTION

In this section, the GDD concept of the previous section will be expanded into a full GDD design. The design begins with a functional breakdown of a GDDS given below. The remainder of the section treats the four technical goals identified at the end of Section II in the framework of the general organization of the GDDS.

Our GDDS has three main components: a Display Package (DP), a GDD Logic Package (LP), and a Map Management Package (MMP). The DP provides a means of creating and controlling a display of the GDDS data. The MMP manages the storage and retrieval of GDBs on secondary storage. LP embodies the internal logic of our GDDS design and connects the DP and MMP. LP determines logically what data are needed from secondary store and requests it from MMP. MMP finds the required data and passes it (possibly via LP) to the DP for display.

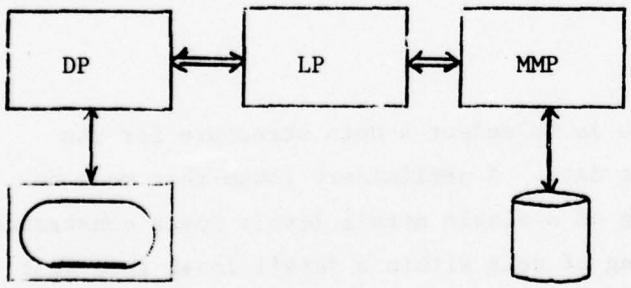


Figure 14. General GDDS Organization

DP will not be discussed further in this report: it is a graphics handler and its precise nature is peripheral to the design of our GDDS.

The primary concern of MMP development is the handling of geographic data. Ways to organize the data, both logically and physically, have to be enunciated. In the process, the interface between MMP and LP -- the manner of referencing a GDB -- will be defined. The logical organization of the geographic data is the specific concern of GDB DATA STRUCTURE, as is the MMP/LP interface. Alternatives for the physical organization of geographic data are presented in GDB STORAGE AND RETRIEVAL with an analysis of their respective merits and shortcomings.

Much of the internal logic of the system was presented in Section II. The topics outstanding--provision of optional amounts of geographic detail and independent map features--are covered in OPTIONAL DETAIL and PARALLEL FEATURES, respectively. A synthesis of the various constituent factors of the design is presented in the context of user requests in DESIGN PRECIS. This description completes the definition both of LP and of the overall GDD design.

GDB DATA STRUCTURE

The problem here is to select a data structure for the storage of geographic data. A preliminary issue that must be faced is the handling of a single detail level: space constraints require a partitioning of data within a detail level into what we call "neighborhoods." After the neighborhood concept is introduced, the development of a GDB data structure will proceed constrained by two further boundary conditions that were specified in the previous section.

Neighborhoods

In the preceding section, we argued that a detail level must have neither too much nor too little resolvable detail. However, creating an appropriate detail level for one range of scales can induce difficulties in other levels.

Consider the example indicated below.

Top level	has	N_1 points
.	.	.
.	.	.
.	.	.
Bottom level	has	N_K points

The top-level display requires N_1 points in the display list. The next level down, having more detail, requires more points, N_2, and so on down the line. We get

$$N_1 < N_2 < . \ . \ . < N_K.$$

If N_1 is chosen close to the maximum number of points that can be handled on the display list, the lower levels will exceed the display list's limit. On the other hand, if N_K is chosen close to the limit, N_1 will be too small to provide an adequate representation.

The problem results from manipulating an entire map no matter how small the window being displayed is. Our remedy is to abandon that procedure. We do so with no loss, since translations and zooms on the small window shown do not require the manipulations that are routinely performed on the entire display list. All that

is required is the appropriate calculations done on some small neighborhood of the current window (the shaded area in Figure 15). Thus we are led to the idea of splitting up the map into "neighborhoods."

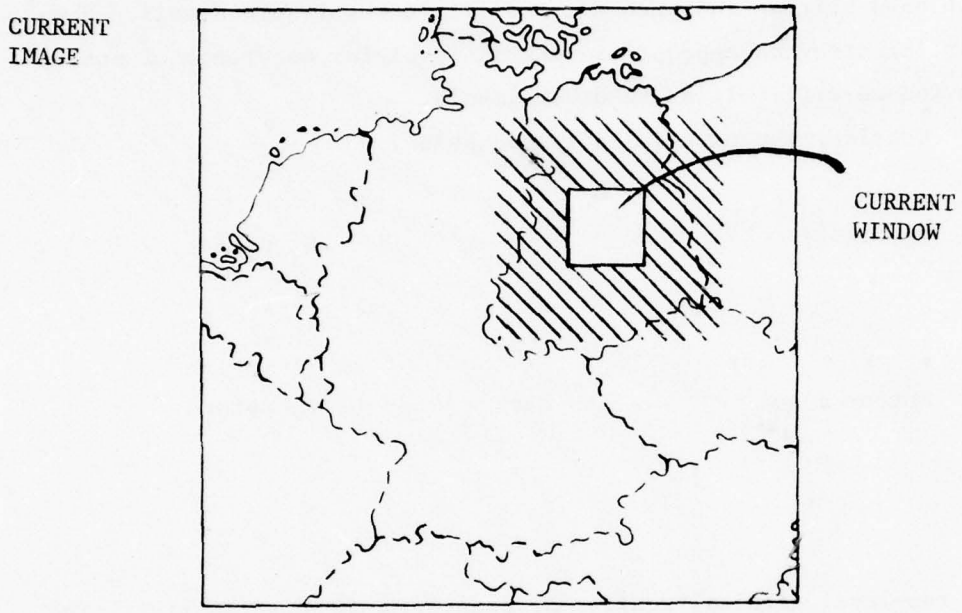


Figure 15. Superfluous Map Calculations

The appropriate neighborhood is manipulated locally, with neighborhood swapping being done only when required. (The analogy of paging in computer memory management is enlightening.)

The neighborhood approach relieves the space problem we mentioned. In particular, the display-list limit is now applied not to the entire map but to the "current neighborhood." With care, adequate detail can be provided at all scales without overloading the display list.

The management of neighborhoods poses another problem: specifically, how does one display a window straddling the edge of two neighborhoods(as in Figure 16)?

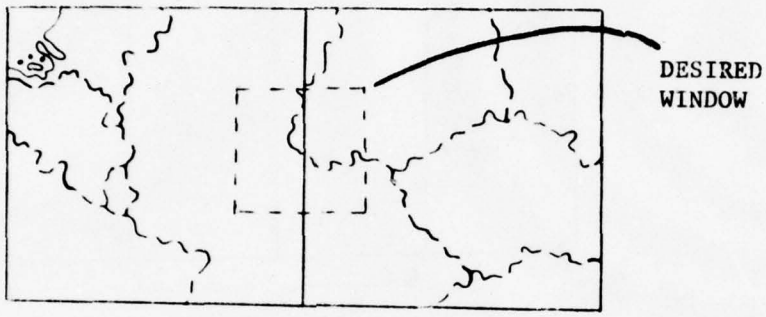


Figure 16. Window Straddling Neighborhoods

The answer lies in not allowing neighborhoods to be too discrete: any window must lie entirely within some neighborhood of its scale. One way to assure this is to overlap neighborhoods as shown:

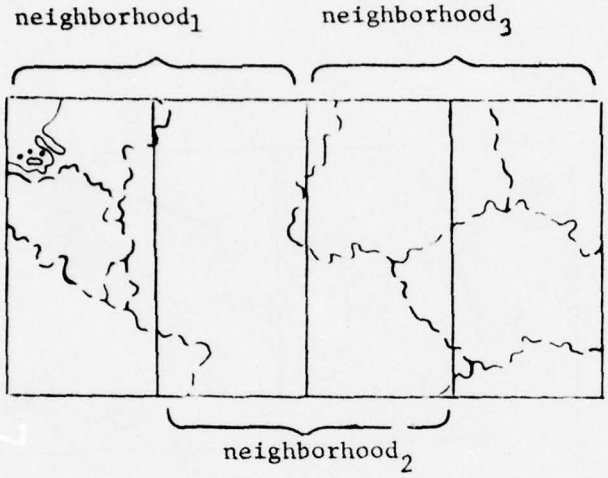


Figure 17. Overlapped Neighborhoods

This scheme solves the problem at the expense of storing all feature data with a particular amount of detail twice. Another scheme which doesn't require multiple storage constructs neighborhoods from smaller map blocks. In this scheme, there are (conceptually) overlapping neighborhoods whose common points are "shared" and stored in mutual blocks, as illustrated in Figure 18.

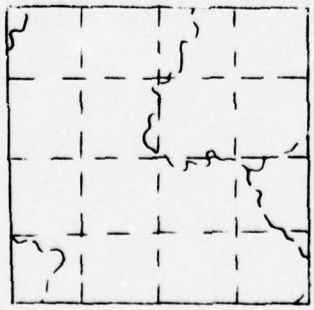


Figure 18. Shared Neighborhood Blocks

In sum, our initial GDD concept provides maps with varying amounts of detail. The restrictions imposed by the display lists are handled by dividing each map into blocks and constructing a neighborhood that fits into the display list from a set of contiguous map blocks.

Translation and zooming under the assumptions of detail levels and neighborhoods is relatively straightforward. The current window is always centered inside the dashed box in Figure 19.

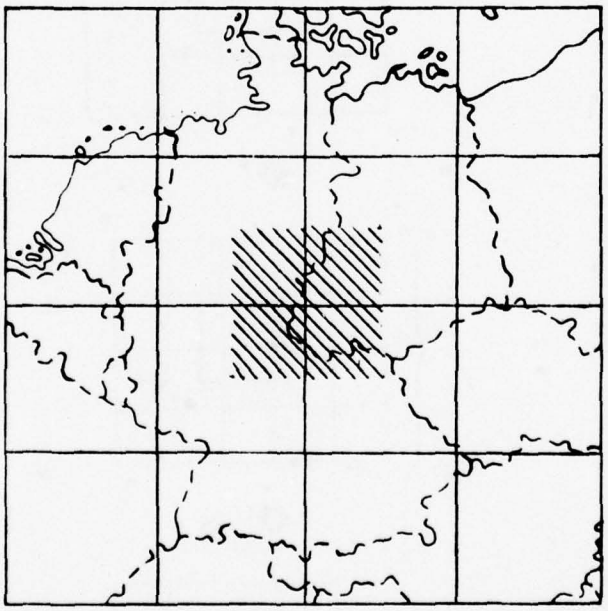


Figure 19. Locus of the Window Centerpoint

A translation that moves the centerpoint of the window outside that box causes a neighborhood to be constructed using some new and some old map blocks, as shown in Figure 20.

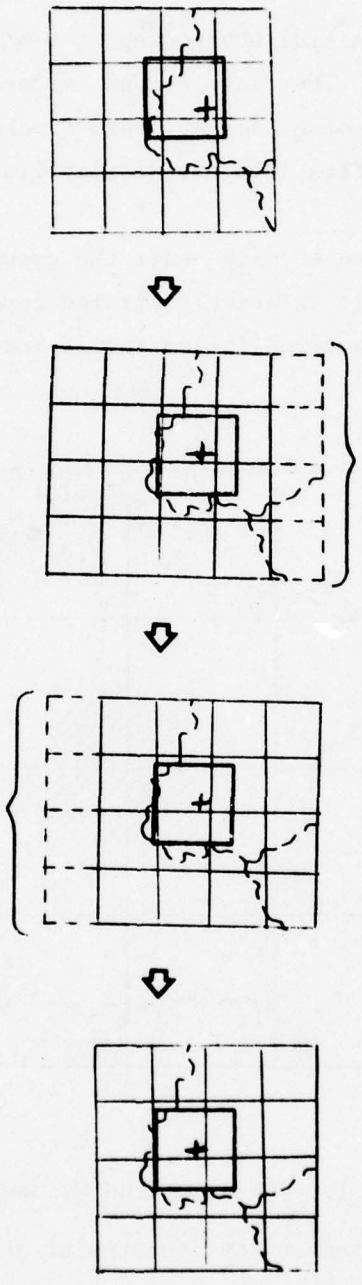


Figure 20. Translation Neighborhood Management

A zoom that changes the detail level causes a neighborhood of a lower (or higher) level to be constructed about the new centerpoint, as shown in Figure 21.

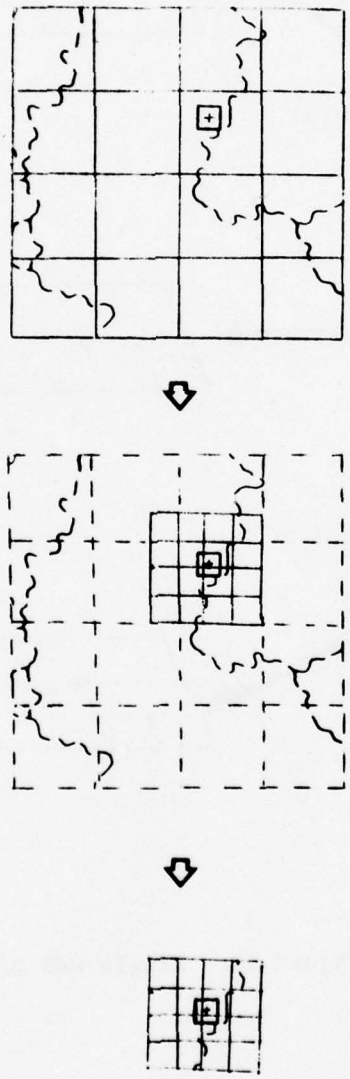


Figure 21. Zoom Neighborhood Management

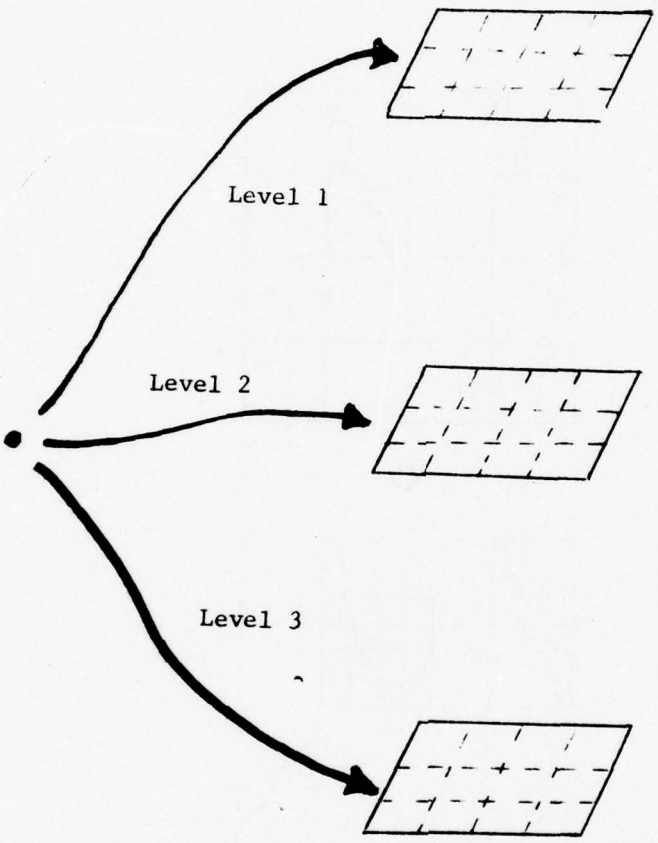


Figure 22. Single GDB Data Structure

Location and selection are less directly related to the concepts of detail level and neighborhood. Location causes a neighborhood to be constructed from the proper detail level (based on the scale desired) about the indicated centerpoint. Selection alters the list of displayed features, adding or deleting features from the display as appropriate.

Data Structure

The development of GDB data structure must take into account the three boundary conditions developed earlier. The boundary conditions are

- the use of separately maintained detail levels for a single GDB;
- the parallel handling of independent GDBs; and
- the use of map blocks to create neighborhoods for a detail level.

The first condition suggests that detail levels of a GDB can be organized hierarchically, and the second, that each GDB will have its own detail-level hierarchy entered via a common "universal" index. The last condition requires a 2-dimensional organization within detail levels to simplify the process of neighborhood creation.

The overall data structure for a single GDB can be pictured as in Figure 22. Data in the GDB is specified both by level and by position. There will be a similar structure for each GDB that will be provided.

How does one index this GDB data structure? That is, what parameters determine the blocks of data required? From the user's point of view, the basic parameter is that portion of the geographic area he wishes displayed, the "window." The desired window can be specified by two opposite corners or by a centerpoint

and the side of the window. We will use the second set: the pair (C,S) where C is the centerpoint and S is the side of the square window centered at C that is to be displayed.

How can (C,S) be used as an index to the data structure? The magnitude of S determines a detail level through a series of comparisons:

if ($high_1 \geqq S \geqq low_1$) then level = 1;

if ($high_2 \geqq S \geqq low_2$) then level = 2;

.

.

.

if ($high_n \geqq S \geqq low_n$) then level = n.

The point C determines a set of data blocks for the designated level that depicts the area around C containing the window specified by (C,S). We will let a geographic point (x,y) determine an index which we will call a "hub." For this discussion, a "neighborhood" of data blocks will be 16 blocks arranged in a 4 x 4 square and the neighborhood's hub is the center of the area. Each point (x,y) is associated with the hub nearest it: in Figure 23, the points within the dashed square are associated with the marked hub.

To summarize, the index into our data structure is a pair (C,S), where C is a point in the geographical area and S is a positive number. S determines a hierarchical (detail) level. C determines a hub which specifies a neighborhood of 16 blocks which form a 4 x 4 square about the hub, covering C.

The next consideration is the LP/MMP interface. This will take the form of a naming scheme for the blocks of a detail level. Because of the 2-dimensional nature of the underlying data, we will use 2-dimensional indices.

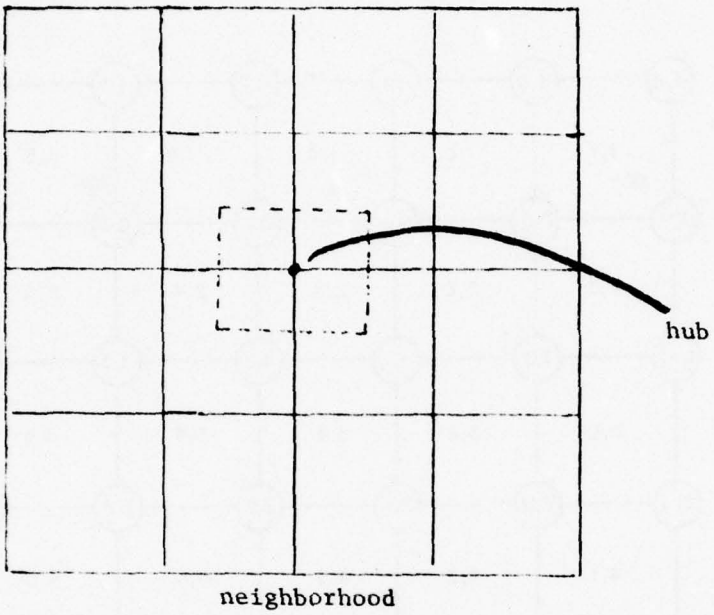


Figure 23. Neighborhood with Hub

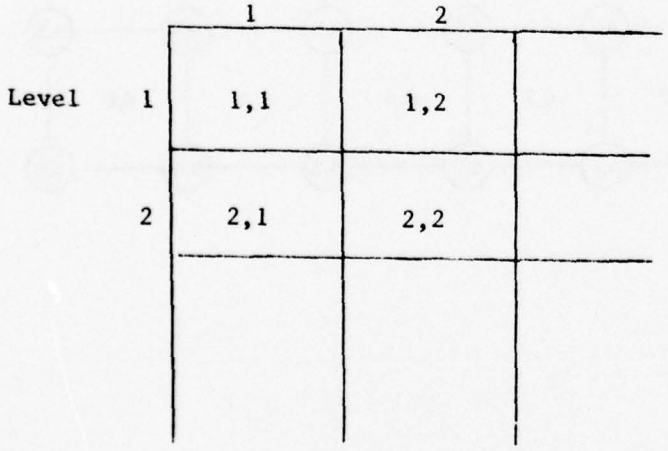
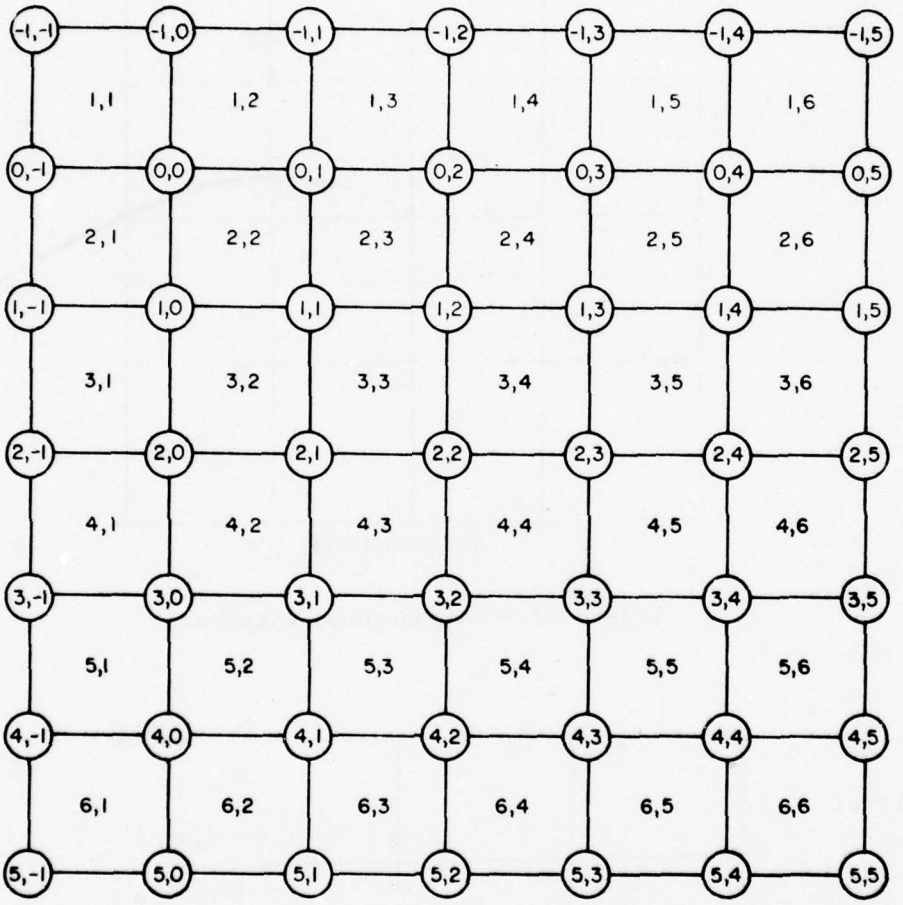


Figure 24. Block Naming Scheme



IA-48,837

Figure 25: NEIGHBORHOODS AND HUBS

We propose a 2-dimensional i-j identification of blocks using integers starting at some arbitrary block, as shown in Figure 24. For convenience, we will give a hub the same name as the upper left-hand block in its neighborhood as shown in Figure 25. With this convention hub (i,j) specifies all the blocks in the neighborhood. For us, the neighborhood of hub (i,j) is

Bl(i,j)	Bl(i,j+1)	Bl(i,j+2)	Bl(i,j+3)
Bl(i+1,j)	Bl(i+1,j+1)	Bl(i+1,j+2)	Bl(i+1,j+3)
Bl(i+2,j)	Bl(i+2,j+1)	Bl(i+2,j+2)	Bl(i+2,j+3)
Bl(i+3,j)	Bl(i+3,j+1)	Bl(i+3,j+2)	Bl(i+3,j+3)

The need for a new neighborhood display is handled as follows:

- from a new (C,S), a level is chosen from the value of S;
- from C, hub(i,j) is calculated; and
- the neighborhood of hub(i,j) is assembled and used for the display

GDB STORAGE AND RETRIEVAL

Introduction

MMP has one main function: to store leveled, geographical data so that it can be retrieved with the input of the parameters level and "place." One obviously wants to make the storage and retrieval functions as elegant and simple as possible and to minimize the number of secondary storage accesses. Secondary functions of the MMP will involve the maintenance of internal data bases.

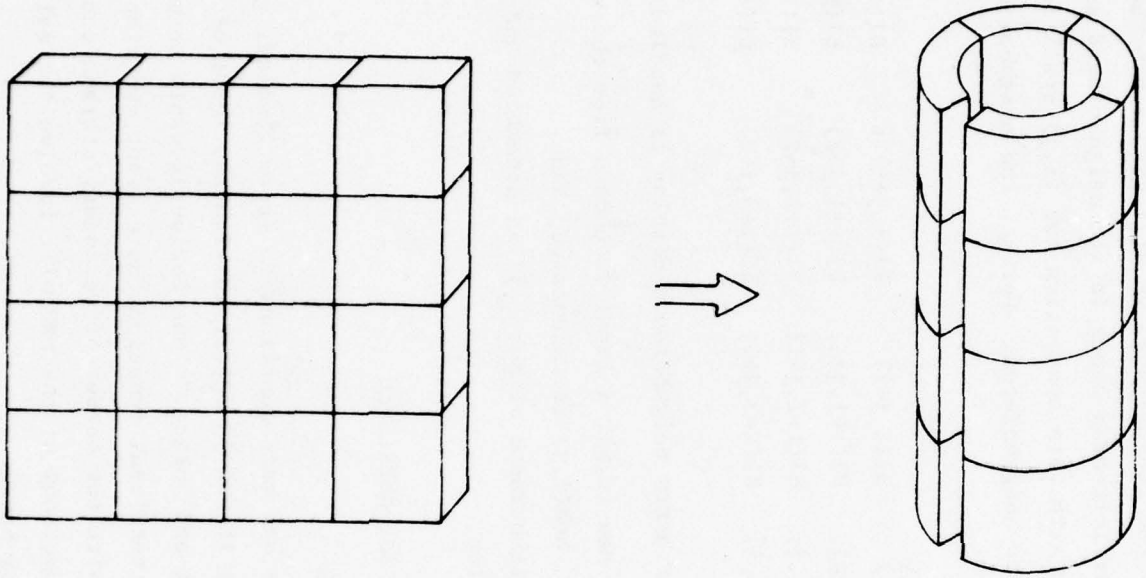


Figure 26: PSEUDO-MATRIX STORAGE

Since the index into a single GDB is a hub rather than a particular map block, we need a storage and retrieval system for a single level of the map. Nothing further is required of MMP since the ability to move between levels entails an external calculation of a new hub (by LP) and a neighborhood retrieval (by MMP), all of which is identical to an initial display procedure.

Storage Schemes

Several storage schemes can be devised to handle the MMP requirements. We will discuss three: a pseudo-matrix method where the storage structure reflects the block structure of a level; a space-over-speed method which optimizes space conservation primarily and speed enhancement secondarily; and a speed-over-space method which optimizes speed enhancement primarily and space conservation secondarily.

Pseudo-Matrix Method

Since the map itself is divided neatly into a matrix of cells, it is particularly tempting to try to organize the stored data in the same way so as to be able to employ "matrix access." An example will illustrate the point. A cylinder on a random access device is very much like a two-dimensional array. It is, in fact, a cyclical array. If the original structure for a map could be duplicated on the storage medium as shown in Figure 26, then retrieving an n x n block of geographical data could proceed by reading n successive blocks on n separate heads. If the device does not allow parallel reads then by "slanting" the data base, the illusion of parallel reads could be gained by switching read heads. An illustration of this idea is given in Figure 27.

4,5	4,6	4,7
5,5	5,6	5,7
6,5	6,6	6,7

is stored as

4,5			4,6			4,7		
	5,5			5,6			5,7	
		6,5			6,6			6,7

and is read by three heads in sequence

Figure 27: Slanted Matrix Storage.

Unfortunately, none of these methods is practical unless the total secondary storage required for each map block of a given level is nearly constant. Without that condition holding, the number of storage blocks needed per map block will vary from 0 to many, rendering useless the clever schemes of this sort we might devise. In particular, many map blocks in the data base for a particular kind of display item will be empty. Hence we direct our efforts towards a storage scheme which will allow us to flag null map blocks at the index level and to use exactly as much secondary storage as necessary for each nonempty map block.

Speed-Over-Space

This method minimizes response time by using as few accesses as possible: specifically, every change of neighborhood requires a maximum of two accesses, one to an index and one for the data themselves. Within that constraint, we minimize space requirements by sharing common blocks between adjacent neighborhoods.

The basic idea of reducing block retrieval to one access is to arrange the storage of a neighborhood's blocks linearly as shown.

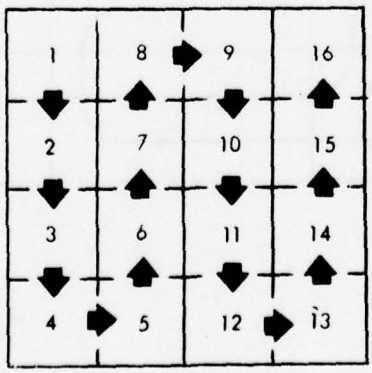


Figure 28. Storage of a Neighborhood

Figure 29: SPEED-OVER-SPACE MAP STORAGE

IA-48,843

With an m x n map and neighborhoods of k x k blocks, separate storage of every neighborhood requires the storage of 12 x (m-3) x (n-3) map blocks (k^2 for each neighborhood, (m-3) x (n-3) neighborhoods). One can save a great deal of space by arranging the neighborhoods so that common blocks can be shared. Requiring an odd number of columns, one can use the system as shown in Figure 29, with the storage of $(m+1-k)\lceil k(n-1)+1\rceil + (k-1)$ map blocks.

Since our entire map is stored linearly, the data base index for this scheme requires two fields - the address of the first block and the length (the number of data blocks required). The data base index in this case can be of any form since one reference to the index and one access to the data suffices in all situations. Hence the data structure used for the data base index will be chosen to minimize cost.

For purposes of comparison, we will discuss here the general cost characteristics of the Speed-Over-Space method. The time cost is optimal: two data accesses, one to the data base, are necessary and sufficient. The cost of storage space has two aspects, the storage of the data and the storage of the DBI. As was mentioned earlier, a map divided into m x n map blocks and using neighborhoods of k x k blocks requires the storage of

$$(m+1-k)\ \lceil k(n-1)+1\rceil+(k-1)$$

map blocks, approximately kn(m-k) map blocks. The DBI requires one entry per neighborhood, each entry consisting of an address and a length for a total of 2 x (m-k) x (n-k) integers.

Space-Over-Speed

This method uses as little secondary storage as possible; specifically, every block of data is stored exactly once. Within that constraint, we have attempted to minimize response-time by

minimizing the number of disk accesses required to retrieve an entire neighborhood.

First, we will reduce the number of accesses to secondary storage if we store our map blocks so that several of them can be retrieved in a single access. This suggests that a map of a given level could be stored in horizontal or vertical strips, with no a priori advantage of one scheme over the other. We will use vertical strips for a reason to be discussed later. Thus our map

1,1	1,2	1,3	
2,1	2,2	2,3	
3,1	3,2	3,3	

will be stored as the sequences

(1,1), (2,1), (3,1), ... ;

(1,2), (2,2), (3,2), ... ;

and so forth,

in order.

We picture our secondary storage as shown*:

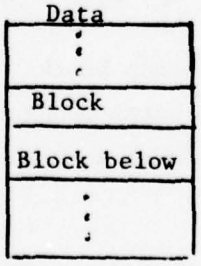


Figure 30. Secondary Storage Using Vertical Strips

*Each entry here may consist of several physical blocks of data; we picture the several physical blocks as one logical block.

To call up desired blocks will require a data block index. The index that we propose will contain four pieces of information:

- the address of the beginning of the data for the given map block (Address);
- the number of data blocks needed to retrieve the given map block only ($N^{(1)}$);
- the number of data blocks needed to retrieve the given map block and the three map blocks directly below it ($N^{(4)}$); and
- a code word that specifies which of the vertical map blocks are null and which are not (code).

We picture an index to our data base as shown:

Address	$N^{(1)}$	$N^{(4)}$	code

Figure 31. Data Base Index Entry

How shall we store the data block index? We propose to store it "horizontally": the address for the map block (i,j) directly precedes that for(i,j+1), and so forth (see Figure 32).

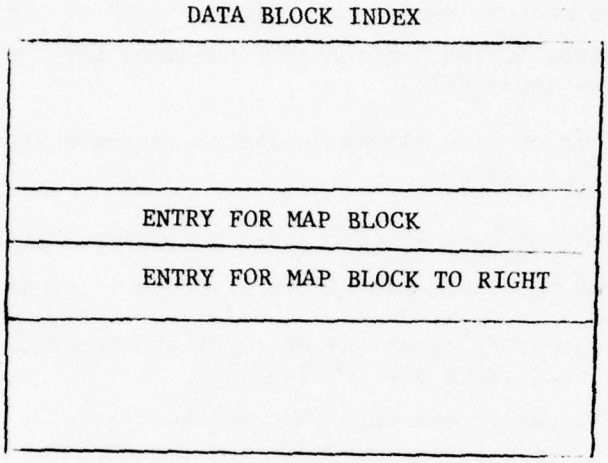


Figure 32: Data Block Index

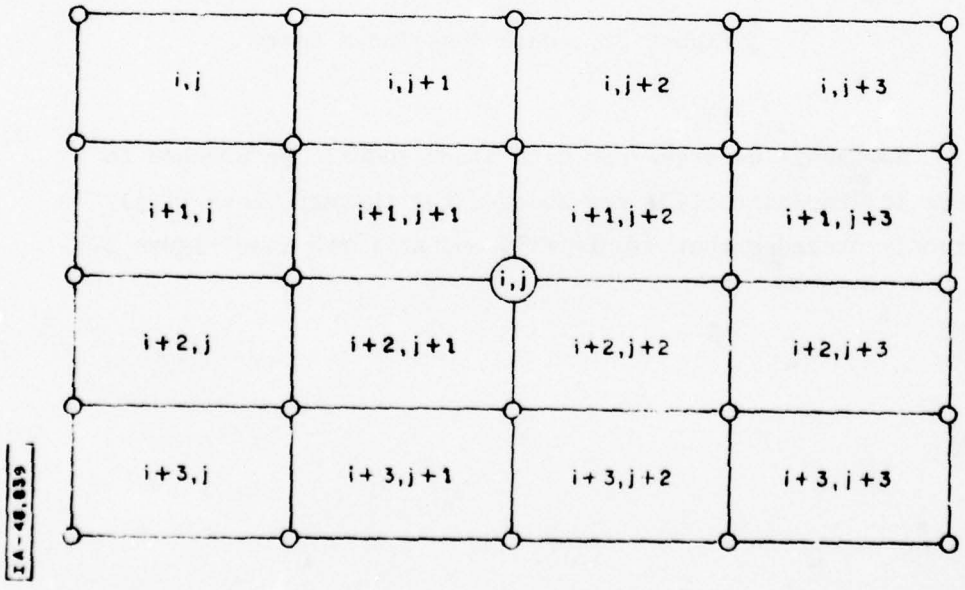


Figure 33: NEIGHBORHOOD OF HUB (i,j)

Consider the implications of storing data vertically and indices horizontally. Suppose it is necessary to call up the neighborhood of hub (i,j) at some level (shown in Figure 33). An access to the entry for map block (i,j) in the data base index and to the next three brings the following data into core:

$Address_{i,j}$	$N^{(1)}_{i,j}$	$N^{(4)}_{i,j}$	$code_{i,j}$
$Address_{i,j+1}$	$N^{(1)}_{i,j+1}$	$N^{(4)}_{i,j+1}$	$code_{i,j+1}$
$Address_{i,j+2}$	$N^{(1)}_{i,j+2}$	$N^{(4)}_{i,j+2}$	$code_{i,j+2}$
$Address_{i,j+3}$	$N^{(1)}_{i,j+3}$	$N^{(4)}_{i,j+3}$	$code_{i,j+3}$

Figure 34. Four DBI Entries

Four additional calls--

- at $Address_{i,j}$ for $N^{(4)}_{i,j}$ data blocks;
- at $Address_{i,j+1}$ for $N^{(4)}_{i,j+1}$ data blocks;
- at $Address_{i,j+2}$ for $N^{(4)}_{i,j+2}$ data blocks; and
- at $Address_{i,j+3}$ for $N^{(4)}_{i,j+3}$ data blocks--

will retrieve the entire square of sixteen blocks. Thus a maximum of five accesses to secondary storage will suffice to generate an entire neighborhood. One could embed the logic for determining data accesses from neighborhood requests in MMP or in LP. One of the refinements that can be employed should be considered before that assignment is made.

Our first consideration will explain why we decided to store our data vertically. In our situation, displaying Europe, East-West translations should predominate over North-South translations, because of the relative geographical locations of the major protagonists in Europe. Hence, in many cases, a translation can cause the necessity to move from hub(i,j) to hub(i,j+1). Now the neighborhoods to these two hubs differ in only two vertical strips.

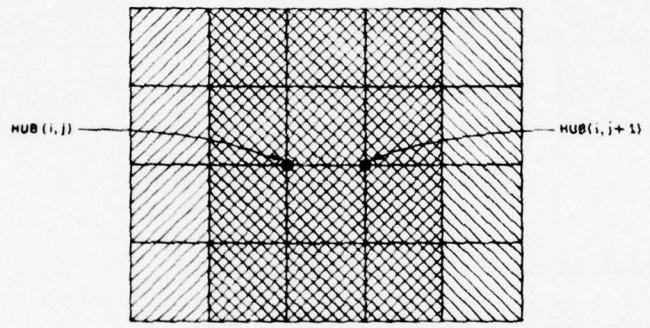


Figure 35 THE OVERLAP OF THE NEIGHBORHOODS OF HUB (i,j) AND HUB (i,j+1)

The cross-hatched blocks in Figure 35 need not be retrieved. Since it would be possible to delete blocks (m,j), $i \leq m \leq i+3$, using DP functions, the data manipulation is reduced to retrieving the four map blocks (m,j+4), $i \leq m \leq i+3$. With vertical organization, this can be accomplished with one access to the data (and possibly one to the data base index). If the map management package handles the generation of accesses, then it must recognize that a request for the neighborhood of hub(i,j+1), when the current hub is (i,j) implies a right accretion and a left deletion. Moreover, either the map management package must cause a DP delete

(requiring knowledge of the item name and location) or it must pass the command to delete a strip to the GDD Logic Package. If the LP handles the generation of accesses (in terms of squares, strips, or map blocks), then it can itself adjust the DP image when required. In sum, it seems better to have the basic map block access decisions made by the application package and the data access decisions made by the map management package.

A second improvement is suggested by our assumption that horizontal translations will predominate. If some of the data base index were kept in main memory, then one access would be eliminated for many horizontal translations. Initially a table of about eight entries centered on the currently-displayed neighborhood should suffice. Then only horizontal translations out of the current neighborhood will require an access to secondary storage.

Most of the considerations for the map management problem are handled. The main exception is entry to the data base index (DBI). The index at the application level is the pair (C,S), centerpoint and side of the display window. This is transformed to a detail level n and a hub(i,j). Thus the data-base-index-index (DBII) is directly describable as a tree as shown in Figure 36. The value at (n,i) is a base address in the data base index such that offset j is the entry for map block (i,j) at level n.

Consider the overall scheme (Figure 37). Our data structure for leveled geographic data in neighborhood blocks uses vertical data, a horizontal index, and a hierarchical index-index.

We conclude this discussion with cost estimates for the Space-Over-Speed method. Here the space cost for the storage

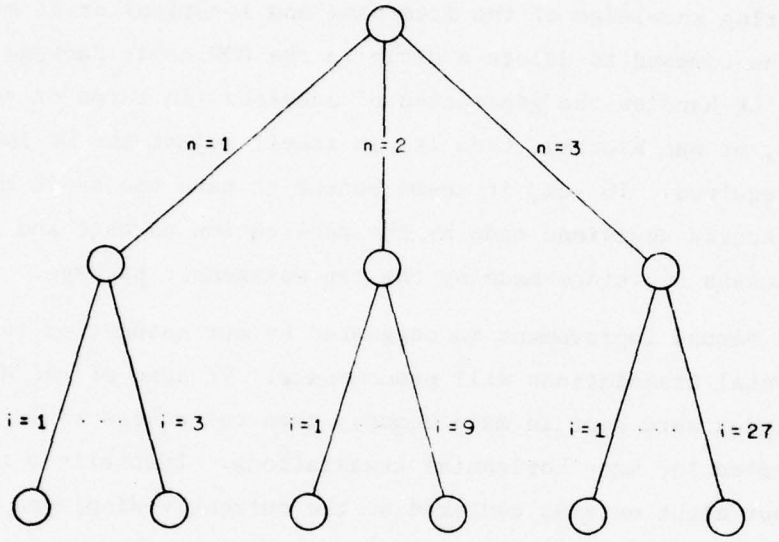


Figure 36: DATA BASE INDEX INDEX

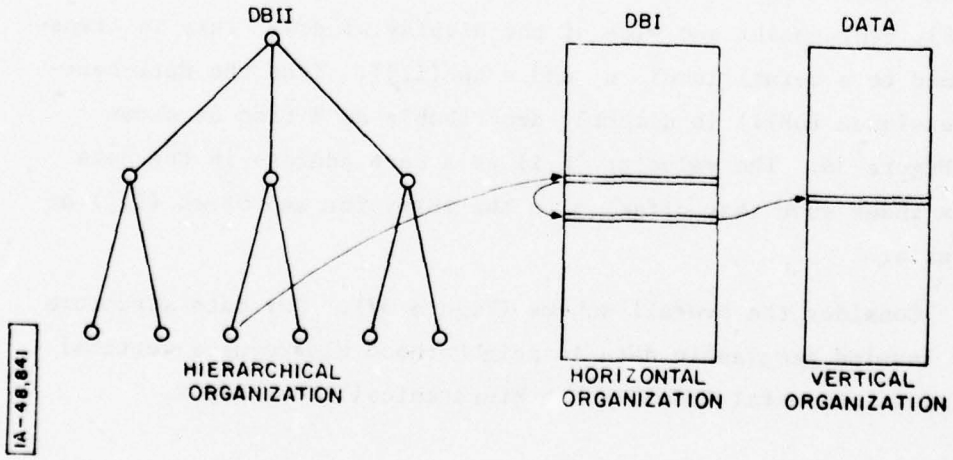


Figure 37: THE OVERALL MANAGEMENT SCHEME

of data is optimal: each map block is stored once. The storage of the indices is more complicated than for the Speed-Over-Space method. The DBII for Space-Over-Speed can be stored in a list with about as many entries as nodes in the tree for the data base. Specifically, the number of entries a DBII for a map requires is

$$[1 + 2 \times (\text{Number of levels}) + (\text{Total number of horizontal strips at all levels})].$$

Each entry is an integer. An entire DBI requires approximately as many entries as the sum of the number of map blocks and vertical strips. Each entry requires four integers, one each for the address, $N^{(1)}$, $N^{(4)}$, and the code. Thus a DBI requires a total of $4(m + n)$ integers. The time cost is $k + 1$ data accesses, one to the DBI and k to the data itself (one for each vertical strip in the neighborhood). No access to the DBII is counted since the DBII is presumed to reside in main memory.

Cost Comparison *

To conclude this topic, let us compare the Speed-Over-Space and Space-Over-Speed methods with respect to cost. The cost values for the two methods are summarized in Table V, where, as before, neighborhoods consist of k x k groups of blocks and the detail level is divided into m x n map blocks.

The major price paid for Speed-Over-Space is a roughly k-fold increase in the storage requirements for the basic data. The price paid for Space-Over-Speed is a (k + 1)-fold increase in data retrieval times. Clearly the decision rests squarely on the relative importance and availability of adequate secondary storage capability.

---

*The comparison is on the basis of one detail level of one feature. The full difference will reflect the values here multiplied by the total number of detail levels in all the features provided.

Table V.

Cost Comparison

	Time (Device Cycles)	Storage	
		Blocks	Index(words)
Speed-Over-Space	1	$(m+1-k)\left[k(n-1)+1\right]+(k-1)$	$2(m-k)(n-k)$
Space-Over-Speed	$k+1$	mn	$4(mn+n)$

OPTIONAL DETAIL

The provision of an optional amount of detail is to be accomplished by overlapping the ranges of scale for the various detail levels. The ranges themselves are specified by a set of "thresholds," ordered pointers into the range of scales as shown in Figure 38. For the data base pictured, no data is displayed when INDEX > P_0; level 1 data is displayed when $P_0 \geq$ INDEX > P_1; level 2 data is displayed when $P_1 \geq$ INDEX > P_2; and so forth. Level m data is displayed for all index values below P_{m-1}. Overlap of detail ranges is accomplished by providing two sets of pointers for each GDB: a "more detail" set and a "less detail" set. The pointers in Figure 39 represent the desired overlap of detail. Between P_i^+ and P_i^- a user can choose more detail (P^+ pointers) or less detail (P^- pointers) as he desires.

PARALLEL FEATURES

The major conceptual advantage of maintaining parallel features is that the process of managing one feature need be replicated once for each feature maintained and the results amalgamated. The internal handling of any one feature is identical to that of any other feature. The only potential stumbling block is the coordination of the GDBs.

Coordination of GDBs is accomplished using a single index into the several GDB data parallel structures (Figure 40) and the Boolean array DISPLAY. For those GDBs currently displayed, the appropriate data blocks are retrieved and the desired display is assembled.

For the sake of completeness, it should perhaps be mentioned that the division of independent GDBs into the same levels of detail has been considered and rejected. The only advantage would be the elimination of parallel calculations of detail

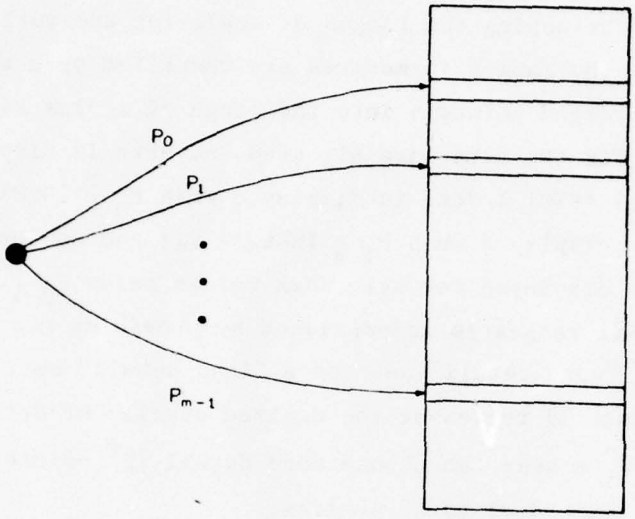
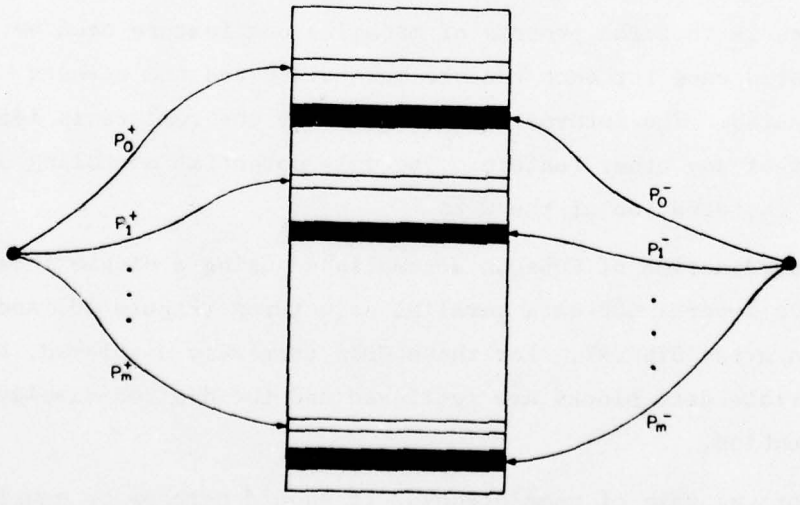
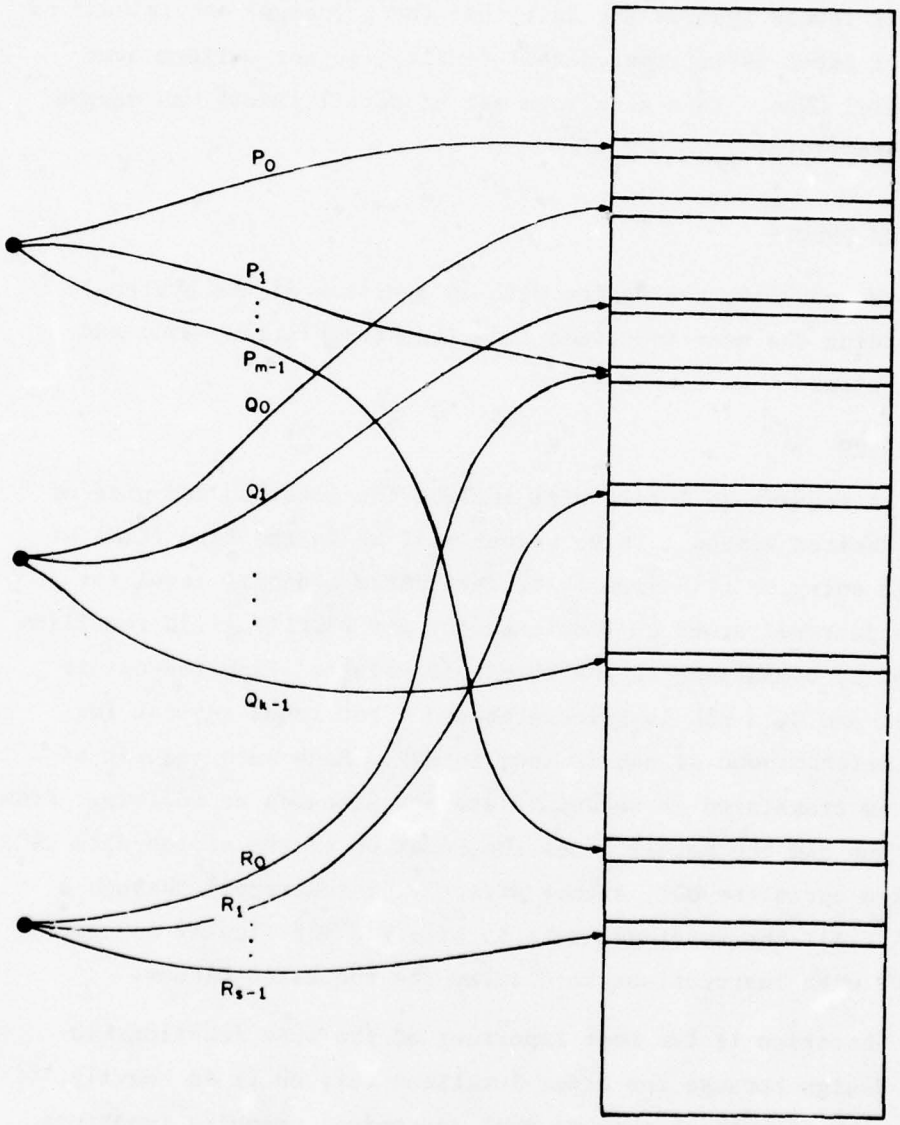


Figure 38: A GDB's POINTERS TO THE UNIVERSAL INDEX



IA-48,846

Figure 39: OPTIONAL DETAIL POINTERS



IA-48,845

Figure 40 GDBs POINTING TO THE INDEX

level and neighborhood. However, forcing all GDBs into the same detail levels ignores the fact that the principal determinant of detail level definition, detail density, is not uniform over parallel GDBs. Thus a uniform set of detail levels was deemed inappropriate.

DESIGN PRECIS

We conclude the design with an overview of the system in providing the user functions location, translation, zoom and selection.

Location

A request to locate will include the location and size of the desired window. These values will be in the form (C,S) at their entry to LP. From S, LP determines a detail level for each feature turned on (indicated by its DISPLAY field equalling TRUE) by comparison to the threshold values. From the detail level and C, a hub is calculated and a retrieval request for the neighborhood of hub is sent to MMP. Each such request of MMP is translated to secondary storage accesses as follows. From the hub and the detail level,the location of the stored data is looked up in the DBI, either directly or indirectly through a DBII. All the retrieval data is prepared for display and sent to DP with instructions to display the requested window.

Location is the most important of the user functions in our design because the other functions rely on it so heavily. The other functions operate much as typical graphics functions unless a threshold is passed. In that case, the new display parameters are used to construct a new display from fresh data retrieved and displayed in the way location does.

Translation

A translation request contains the point that is to be the new centerpoint C' (marked by the cursor). The translation parameters Δx and Δy are calculated from the current centerpoint C and the new one. If the new hub is the same as the old one, a request is sent to DP to translate by the amount $\Delta = (\Delta x, \Delta y)$. If not, the location actions are repeated using the window parameters (C',S), where S is the current side. This calculation can be shortened by retaining the current detail level of each displayable feature.

Zoom

A zoom request includes a zoom factor ξ and the coordinates of a visible point that is not to move(marked by either the cursor or the centerpoint). From the fixed point, the zoom factor ξ, and the current centerpoint C, a new centerpoint C' is calculated. The desired window is $(C', \xi S)$. For each currently displayed feature, ξS and S can be in different levels and C and C' can have different hubs. If either condition holds, the parameters $(C', \xi S)$ are used as in location.

Selection

A selection request specifies some alteration to the array DISPLAY. The appropriate field i of DISPLAY is altered (a la location) to add $feature_i$ data (in the case of an addition command) or to delete $feature_i$ data (in the case of delete).

For each of the four user functions, very little calculation is done--usually a few comparisons dominate the calculation time--and the bulk of the request handling is table look-up and secondary storage accesses. From a design standpoint, the design thus satisfies our criteria of rapid response. It also provides the user functions we specified for a digital map background system. The design is complete.

SECTION IV

CURRENT IMPLEMENTATION

INTRODUCTION

The proof of a paper design for a computer-based system is a working implementation. There is proof of our GDD design in the form of a working GDD system. In this section we will describe the current state of the implementation, concentrating on the system data base, the structure of the system, and the nature of the user interface.

DATA BASE

Our GDD design presumes the availability for digital geographic data for each class of feature that is desired. It assumes further the ability to create a set of detail levels from a fixed digital source. The satisfaction of all these assumptions is an arduous task indeed which is described at length elsewhere (see ESD-TR-76-360, "Geographic Data Base Development,"[3]). We will mention here the content of our current data base; the internal format of the data base and the related problem of selecting an appropriate map projection; and the creation of detail levels.

Content of the Data Base

Currently we have two parallel GDBs, rivers and geopolitical boundaries, covering Europe from Iceland to Finland to Turkey to Portugal. The boundary data base has three detail levels and the river data has two. This data was garnered from preexisting digital sources: the geopolitical boundaries from World Data Bank I (WDBI) and the rivers from World Data Bank II (WDBII).* WDBI

*WDBI is a digital map data base produced by IBM Federal Systems Division and available from NTIS on magnetic tape (PB-2231178). WDBII will be available shortly from the same source.

consists of two digital files totaling 110,000 geographical points and two indices into the data base by region and type of feature. WDBII is an expanded version of WDBI containing 6,000,000 points and covering rivers, canals, lakes, and railroads in addition to improved coastline and boundary data.

Internal Format

Our internal form of the map consists of sequences of points that define boundaries and rivers. The coordinates of the points are pre-projected onto a plane and stored in displayable x-y form.

Our decision to store Cartesian coordinates rather than latitude and longitude values was based on our desire to eliminate as much computation as possible from the real-time response to a user's requests. This decision required us to choose a map projection that could cover the European sector well.

The map projection we chose is a secant conic with two standard parallels. This projection suits our purposes because it has no distortion along meridians, no distortion along the standard parallels, and little distortion (about 1%) between the standard parallels. Altogether this projection is particularly well suited to regions that are limited latitudinally and extensive longitudinally and are neither polar nor equatorial: this specification fits Europe admirably. In addition, both the projection and its inverse are computationally simple and the meridians and parallels have simple forms.

Most other map projections were far worse than our projection in distortion over the European region (see ESD-TR-76-360 [3]). The ones that distorted less tended either to be computationally complex or to have complex meridians and parallels. The best compromise projection is the one we chose.

The Creation of Detail Levels

Data from WDBI and WDBII represents a single level of data, that of the largest scale. The creation of detail levels requires substantial processing of the data base to filter out unwanted detail from the data base.

The filtering of details involves ordering the points of the data base with respect to their relative importance in the representation of the feature. After the points are ordered, as many useable points as can be managed are retained in the detail level of a particular scale.

As indicated in the discussion of detail in Section II, the ranking of points should reflect their resolvability on display media. Thus detail filtering algorithms utilize the deviation of a point from its chain's general direction. The specific algorithms we used for the creation of our detail levels are discussed at length in ESD-TR-76-360.[3]

SYSTEM

Our current implementation of a GDDS includes the pertinent features of a full design. Our implementation has two parallel data bases--boundaries and rivers--at various data levels. The user functions of location, translation, zoom and selection are provided, although not in full generality in the first and last cases. In this subsection we will discuss the developmental environment for our GDDS, the implementation itself, and areas for generalization. The user interface is covered in the next subsection.

Environment

Our GDDS was developed on the multi-minicomputer configuration as shown in Figure 41.

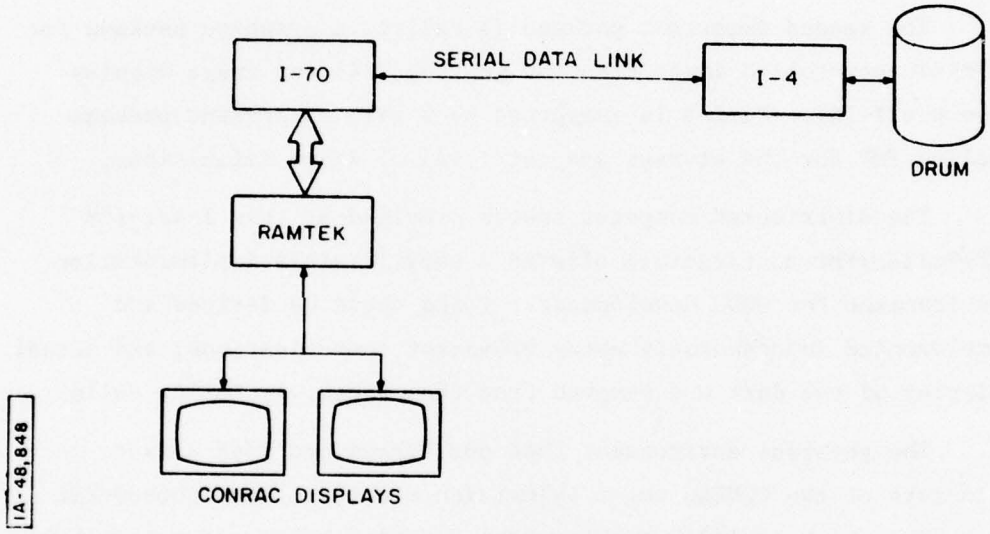


Figure 41: SYSTEM CONFIGURATION

Within this configuration, several software packages were available to facilitate system development.

The first is a general-purpose message processor called MP. This package is co-resident on the I-4 and I-70 processors and provides for broadcast-oriented communication between tasks on the two processors and for task scheduling based on the priority of messages received.

The second important package is Pallet, a graphics package for FORTRAN controlled image creation (on the I-4) and image display (on the I-70). Pallet is supported by a file management package called FMP for the storage and retrieval of image definitions.

The distributed computer system provided by this I-4/I-70/MP/Pallet/FMP architecture offered a very flexible implementation environment for GDDS development. Tasks could be defined and implemented independently using broadcast communications, and actual display of the data was removed from the system via Pallet calls.

The physical environment that our system provides a user consists of two CONRAC color television monitors, an alphanumeric keyboard which includes twenty-eight function keys, and a trackball for the control of the cursor (see Figure 42). One of the monitors is used for map display and the other for feature selection. The trackball and function keys are used to control the system on behalf of the user. The layout of function keys is shown in Figure 43.

Implementation Itself

Our GDDS follows the design of Section III very closely. The actual display of data is delegated to Pallet, while the function of the Logic Package (in Section III) is performed by the main implementation programs. The Map Management Package is implemented as a special-purpose data base management package for static geographic data. A detailed description of the implementation is

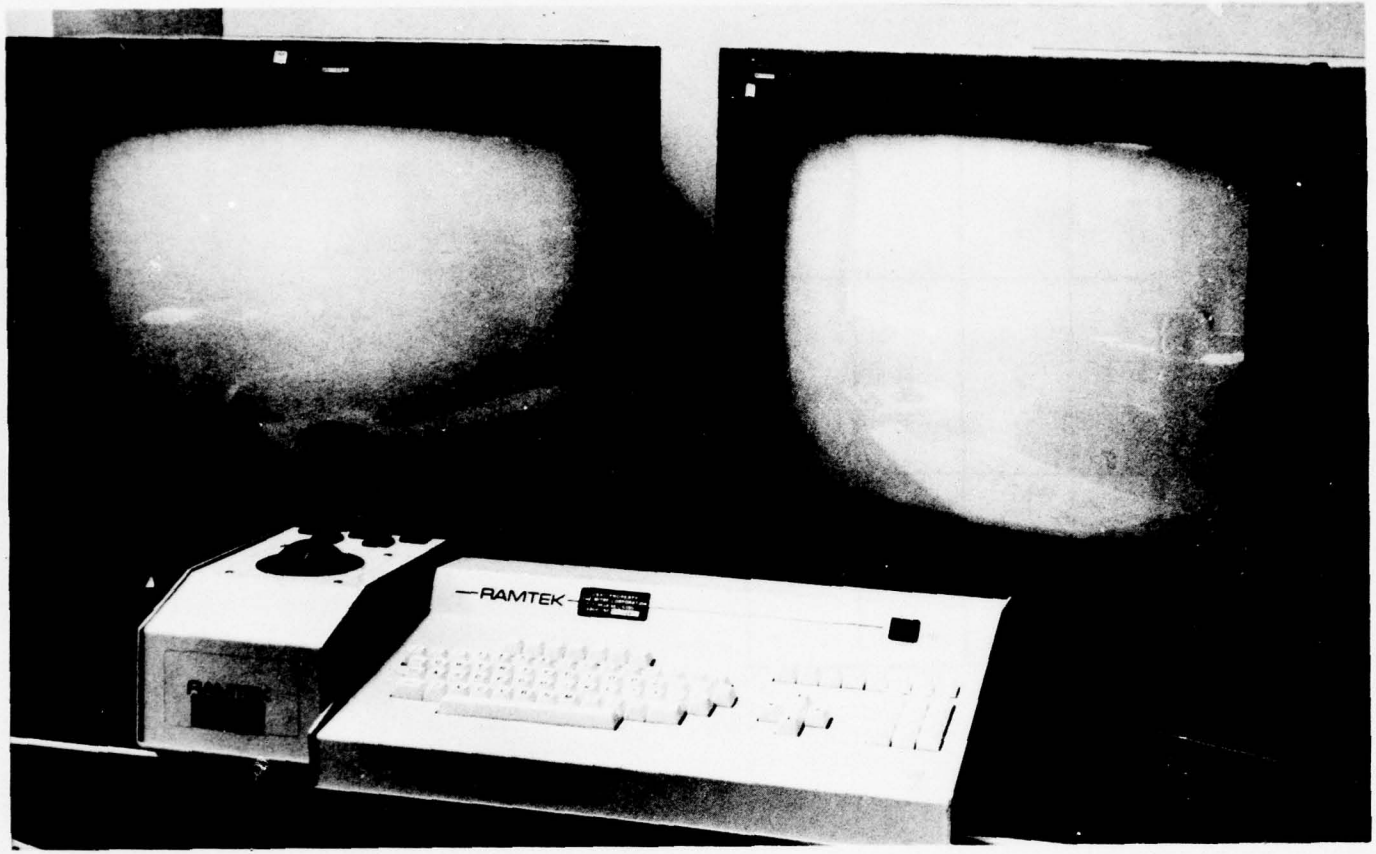
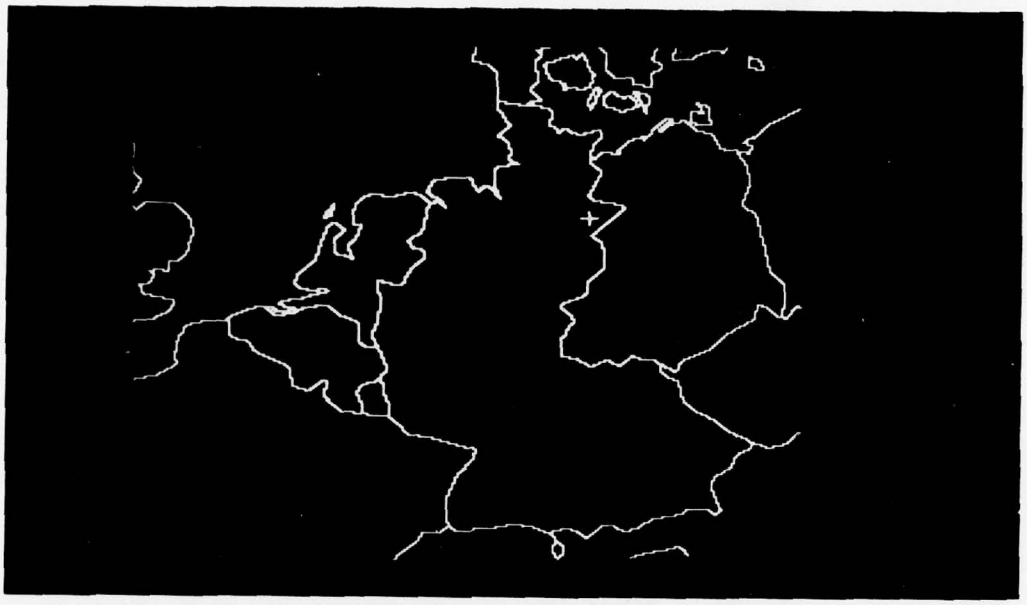


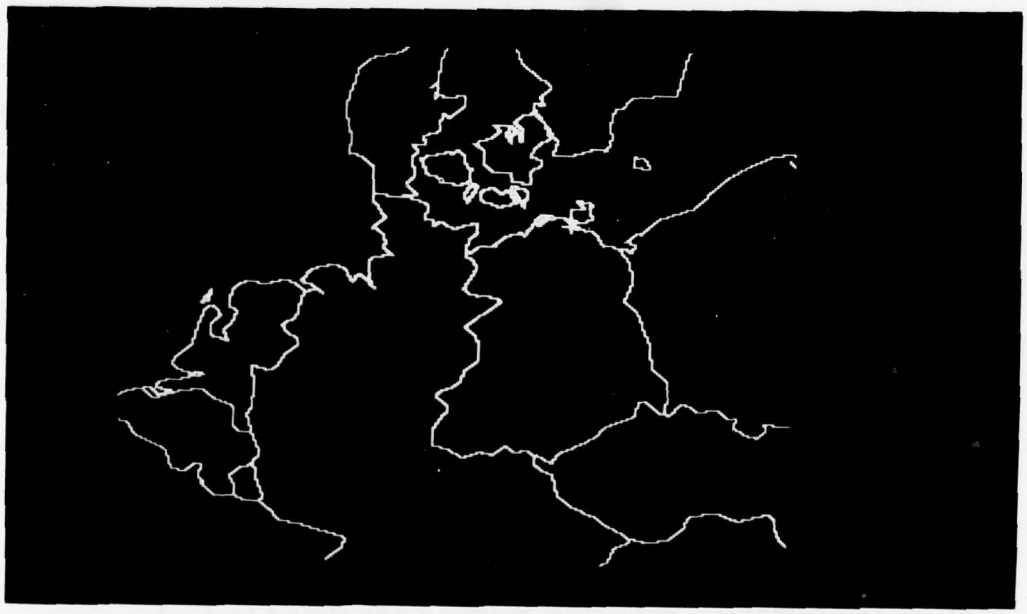
Figure 42. Physical Environment

	ZOOM IN	ZOOM OUT
TRANSLATE		
MENU	SELECT	
AUTO	NORMAL	SPECIAL

Figure 43: Layout of Function Keys



a. Before



b. After

Figure 44. Translate

given in ESD-TR-77-137,[4] "Geographic Data Display Implementation." Let us briefly look at points relating the implementation of these two packages.

Logic Package

The main function of the Logic Package (LP) is the realization of the user functions of location, translation, zoom, and selection. Our view of LP will then be in terms of these functions.

Location, Translation, Zoom

Three of the user functions--location, translation, and zoom -- are handled directly and obviously. Since re-location is not yet provided, location is now an automatic initial display that cannot be affected by the user. That is, the user initializes the system and location at a particular centerpoint and scale happens automatically. Translation is done by positioning the cursor (via the trackball) at the new centerpoint desired and pressing the TRANSLATE key (see Figure 44). Zooming is accomplished in a similar fashion. The point to be zoomed about is identified with the cursor and the appropriate key, ZOOM IN or ZOOM OUT, is pressed (see Figures 45, and 46). In both cases, a shift of context (new neighborhood or new detail level) causes an immediate reaction followed by a secondary update as shown in Figure 47.

In more detail, when a zoom or translate function key is hit, a message is sent to LP. The translate and zoom tasks of LP are activated in response to their associated messages and respond by sending a translate or scale message back to Pallet, causing an immediate response to the user based on the current data. In addition, a message containing the current extent and centerpoint in both projected map units and absolute device coordinates is broadcast by the translate and scale tasks to any other tasks (on the I-4 or I-70) listening for it. Specifically, this message is received by the LP subtasks which determine and handle data



a. Before

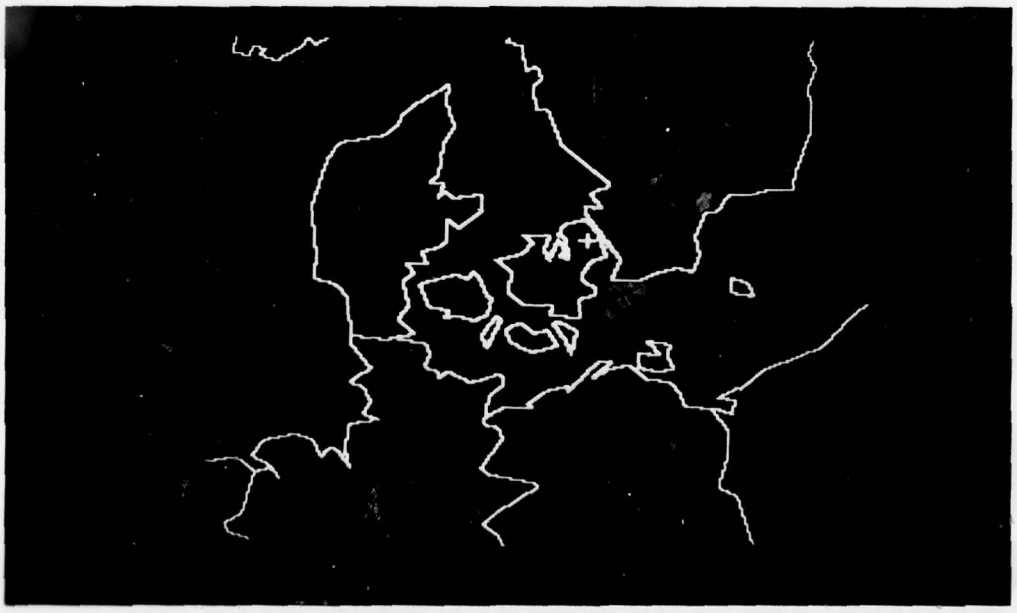


b. After

Figure 45. Zoom In



a. Before



b. After

Figure 46. Zoom Out



a. Before



b. During



c. After

Figure 47. Shift of Context

exceptions and by other tasks which allow additional geographic dependent background and foreground information to be updated. Figure 48 pictures the functional breakdown of the LP.

This global broadcasting of the map's scale and centerpoint provides the only necessary communication between the LP tasks controlling the background display and the foreground tasks displaying operations/intelligence information. Such an implementation provides excellent modularity. We can develop only one task at a time and in the future can add an undefined number of tasks which can communicate with the background task.

Selection. The selection function is somewhat more complicated than the other three. Selection is accomplished using function buttons and a feature selection menu on the second display. The ability to change the set of displayed features is affected by the current mode of the system. Therefore, we will first discuss the modes of our GDDS.

The user has the choice of operating in one of three modes--AUTOMATIC, NORMAL, and SPECIAL. In AUTOMATIC mode the user has no control over features selected for display. Feature selection and detail levels are controlled automatically as a function of predetermined scale values and the current scale of the displayed map. NORMAL mode allows the user to select or delete features to tailor the display to his needs. Detail levels are automatically adjusted to the current scale as in AUTOMATIC mode. SPECIAL mode allows the user to zoom the current display without an automatic adjustment of detail levels, though features may still be selected and deleted as in NORMAL mode. NORMAL mode is the usual mode. AUTOMATIC mode locks in the full set of features for use. SPECIAL mode is used to examine the data base itself by checking its detail level thresholds and its resolution near the thresholds.

IA-48,849

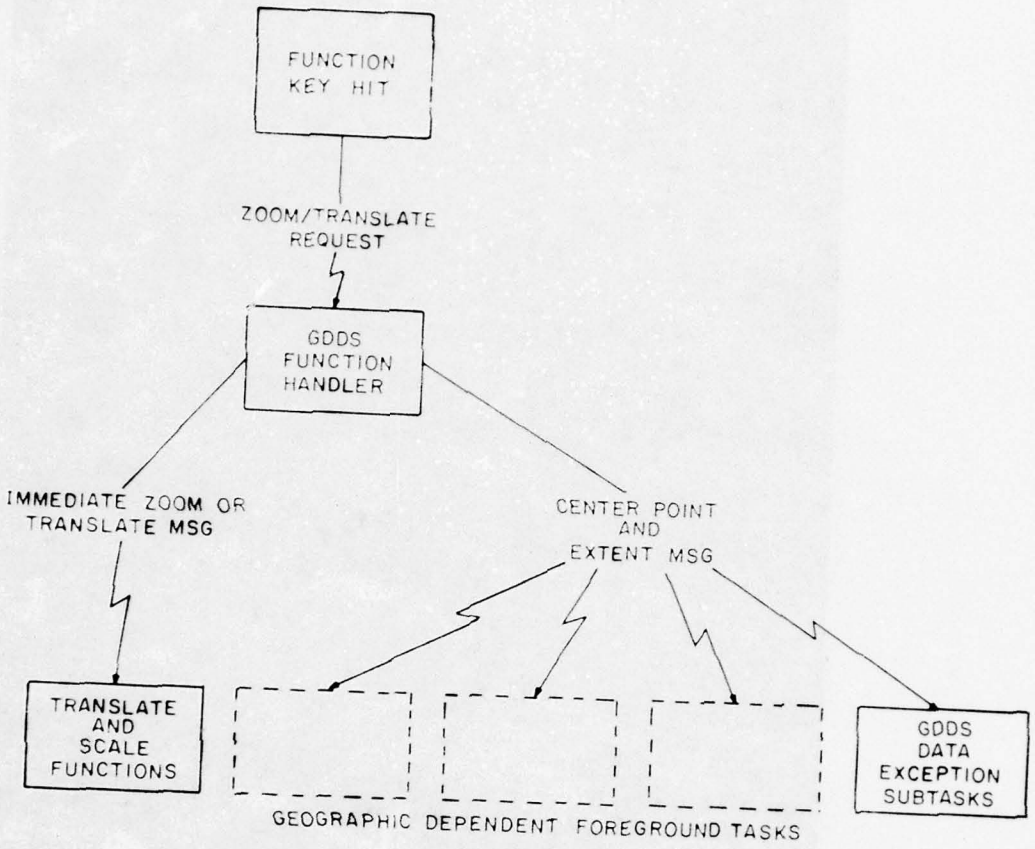


Figure 48: FUNCTIONAL BREAKDOWN OF GDDS

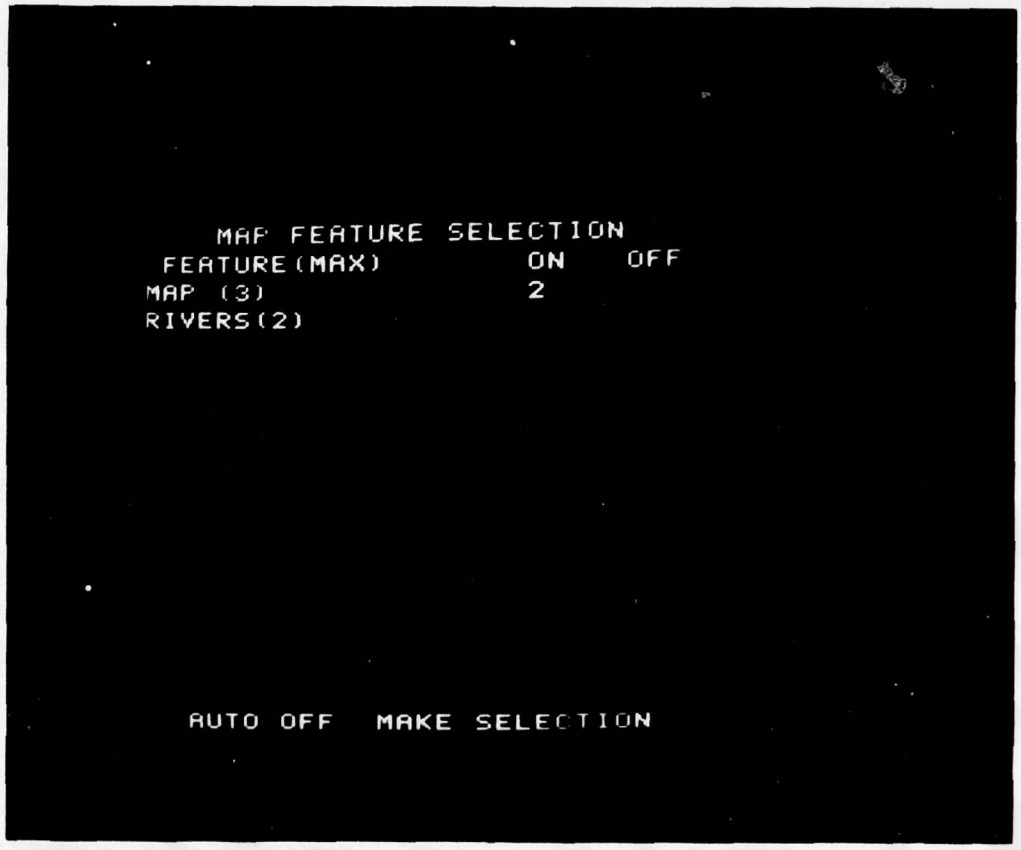
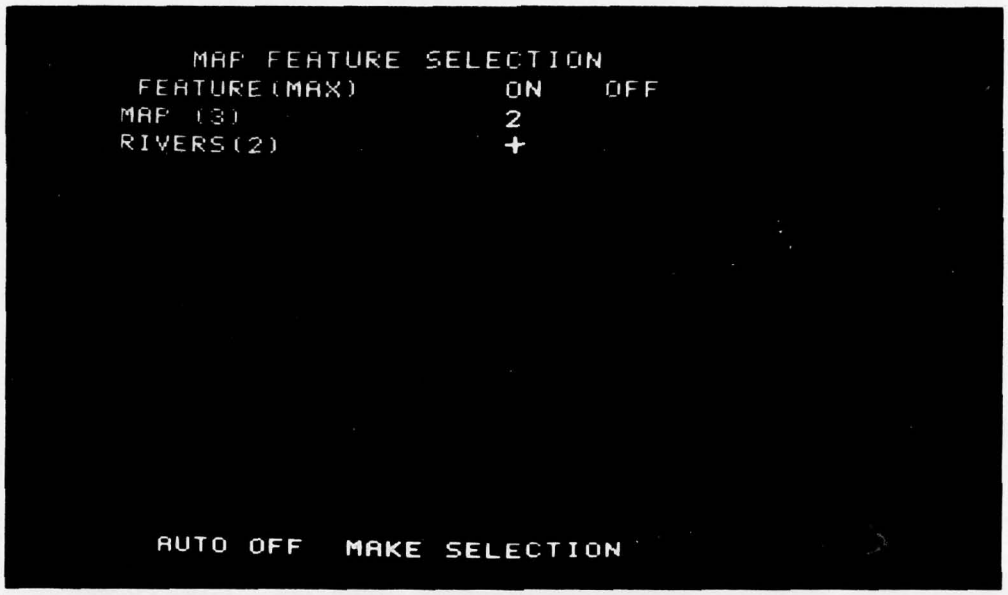
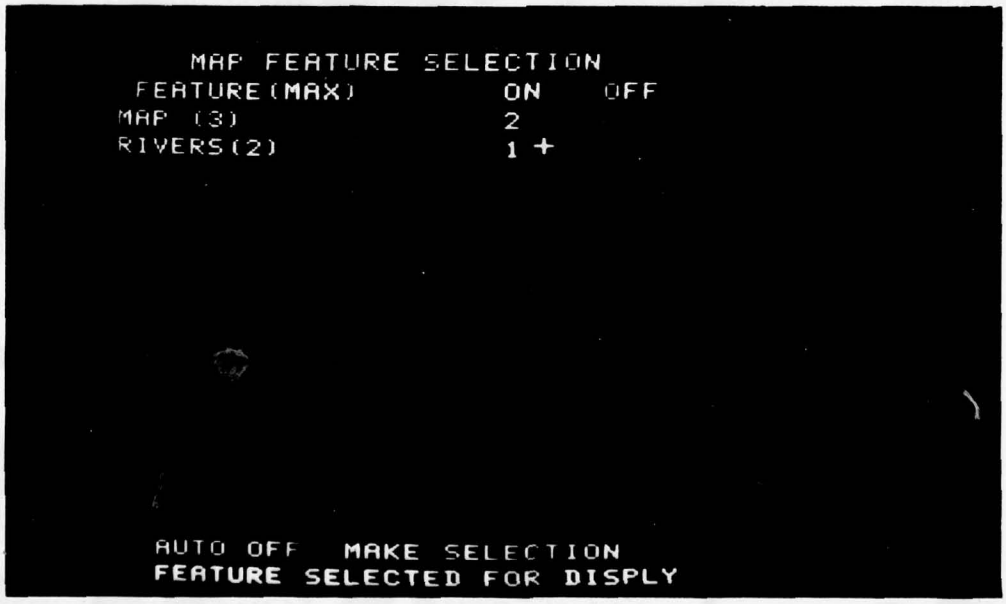


Figure 49. Menu



a. Before



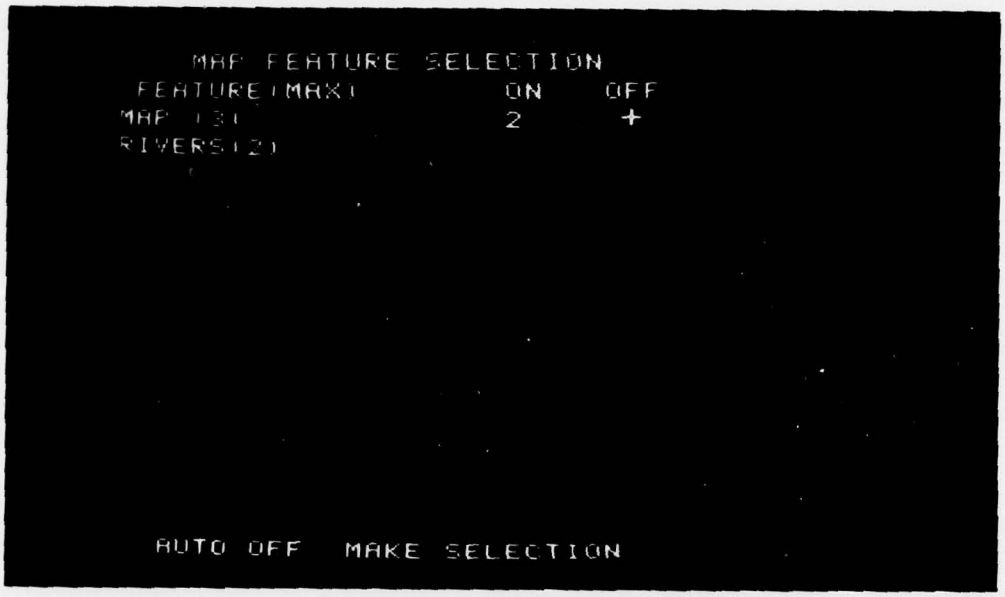
b. After

Figure 50. Selection

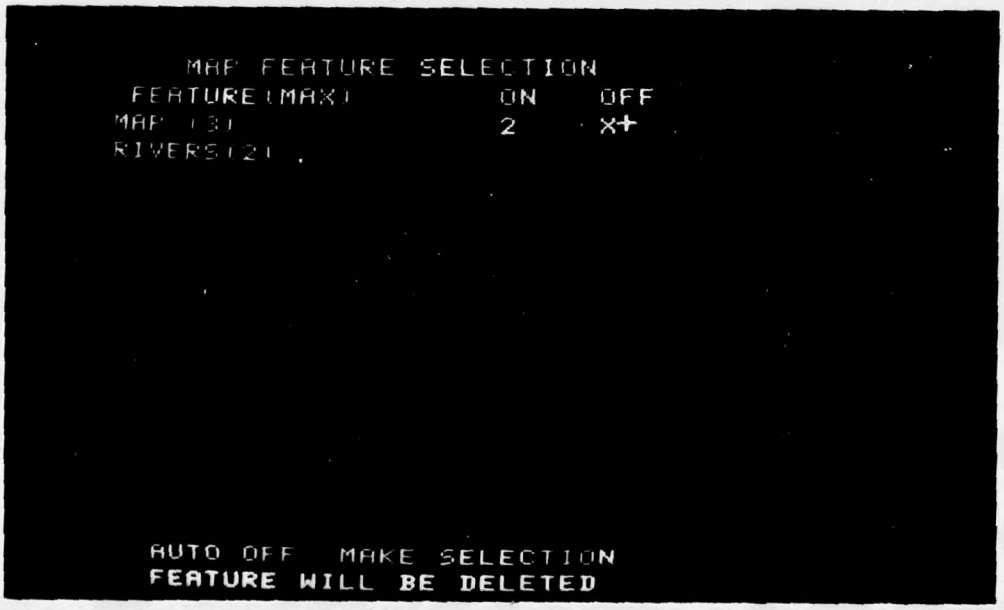
Function keys control the mode for operation and the mode in turn controls the features selection process. In AUTOMATIC mode, no feature selection is allowed: a status report is the only feature selection function available in AUTOMATIC mode. In NORMAL and SPECIAL modes, feature selection is allowed: the feature selection process is initiated by pressing the MENU key. Figure 49 shows the menu format. In the ON column are red numbers indicating the currently displayed detail level of each feature; the ON field opposite a feature not displayed will be blank. By positioning the cursor in the ON or OFF column and hitting the SELECT function key, the user can pick new features for display or delete currently displayed features. If the user selects a new feature, the system will respond by placing a yellow number indicating the detail level at which the feature will be displayed. A yellow X will be placed in the OFF column opposite a feature to be deleted. The selection and deletion functions are pictured in Figures 50 and 51. A capability is provided for macro expansion of feature names. Thus the names in the menu can be category names that are expanded to the individual features in the category.

Appropriate error messages appear in the system response line of the menu if a currently displayed feature is selected for display or a non-displayed feature is selected for deletion.

The user has the ability to change his entries: if the user selects a feature and then changes his mind, he can make an entry in the OFF column to negate the effect of the ON column entry. The reverse is also true; if the user originally wanted to delete a feature, by selecting to display it, the feature will remain on the screen (see Figures 52 and 53). When the user finishes selecting features for display and deletion, he presses the MENU key again. The options he has chosen are entered in an internal "user interest" vector and the display is immediately updated to reflect the new



a. Before



b. After

Figure 51. Deletion

user interest vector.

We continue this discussion with an elaboration of the interrelation of the mode, the user interest vector, and the display. As we mentioned the mode of operation is controlled by the function keys AUTOMATIC, NORMAL, and SPECIAL.

AUTOMATIC mode is entered by pressing the AUTOMATIC key. The display does not change, but the user interest vector is changed to include all available features. On the next zoom request the display will be updated to include all the features.

NORMAL mode is entered by pressing the NORMAL key. Again the display does not change now: updating takes place at the next zoom request. The user interest vector is set to the features currently displayed.

SPECIAL mode is entered using the SPECIAL key. Nothing changes at this time, but the current detail level of each displayed feature is effectively fixed. The user interest vector is set equal to the collection of currently displayed features.

In summary, the selection function normally operates in the following way. A user, desiring to add or to delete a feature, enables feature selection by pressing the MENU button. The menu is displayed, showing the current level of every displayed feature. If the current mode is AUTOMATIC, the System Status Message near the bottom of the menu will state "NO FEATURE SELECTION IS ALLOWED." In this case, the NORMAL button is pressed, the mode changes to NORMAL, and the Status Message changes to "AUTO OFF MAKE SELECTION". If a feature needs to be deleted, the cursor is positioned in the OFF column, even with the feature. This done, the user pushes the SELECT key, causing a yellow "X" to be placed in the OFF column. The map display does not change. To add a feature, the cursor is placed in the ON column opposite the feature to be added and the



Figure 52. Rescinding Selection

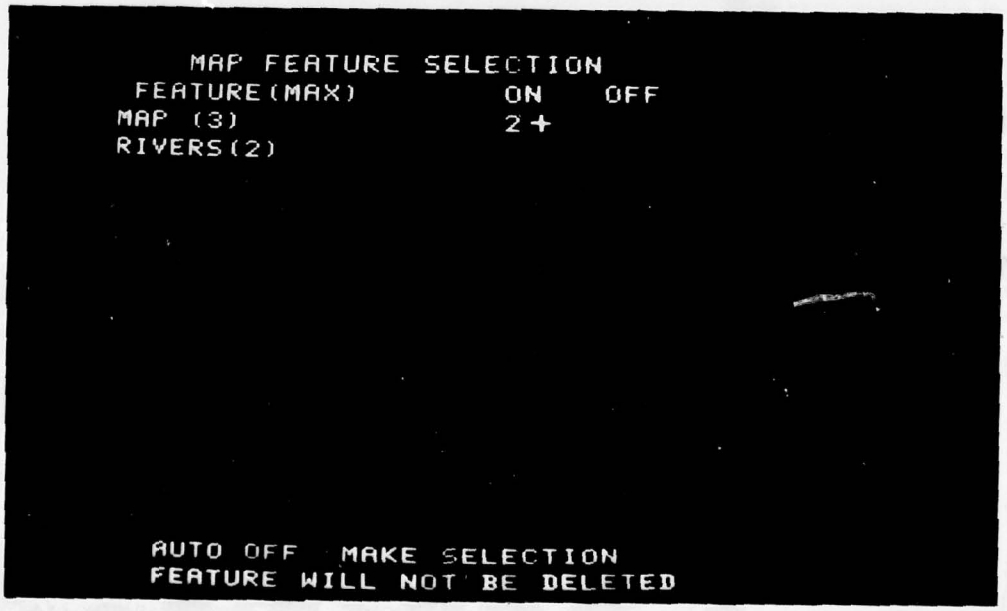


Figure 53. Rescinding Deletion

SELECT button is pushed. A yellow detail-level number appears on the menu and the map display remains unchanged. When selection is completed, the MENU button is pushed again and the map display is altered appropriately.

This implementation of selection does not match the description in Section III. The main difference, of course, is the use of a special selection "state" for gathering a set of selection requests. Altering the user interface for selection is an important area for improvement which is discussed later in this section.

Map Management Package

The development of a special-purpose map management package was needed to realize the Space-Over-Speed map data management scheme. The normal data management used by Pallet, implemented in FMP, required that each map block be stored and retrieved as a separate image. The ability to store and retrieve efficiently both blocks and columns of blocks as single images as required in the Space-Over-Speed scheme did not exist.

The system developed implements closely the Space-Over-Speed scheme described in Section III. Data is stored by section as a sequence of points for each line (boundary, river, and so forth) in that section. To accommodate Pallet in forming images with these sets of points, each point is stored with a move-draw bit (the low order bit of the mantissa of the Y-coordinate). Each start of line point in the block is a move; all the following points in a line are draws. Now every section starts with a move point. Thus to form an image per column the GDDS sets up in a buffer an image header followed by a line header, and retrieves and stores in the buffer the points for four sections (which were stored contiguously). Thus the data base is indexed by section but an entire column is retrieved at once and formed into an image.

The above discussion centered on data bases consisting of lines. The same scheme will work for point-oriented data bases such as latitude/longitude grid points or cities. In this case, the GDDS sets up an image header followed by a point header instead of a line header. In either case, the data base is simply a list of points stored in column order.

This system appears to provide a maximally efficient storage mechanism. Since the index provides a drum block address and a byte address within that block, drum storage can be packed with no wasted space. Sections of a data base containing no data require only an index entry but no storage either on drum or in the Pallet tree structure. If a section is empty, its index entry points to the next section in the column containing data and a length to retrieve the remaining sections in the column. If all four sections of a column are empty, the index will contain a zero length, a flag for that column will be set indicating that there is no node on the display tree for that column.

Areas for Improvement

Our GDDS implementation is not fully general in all respects, although no important facet of the design is unrepresented.

Our selection of available features is not general because our feature data base is incomplete. We need to include other feature data -- roads, places, and so forth. The inclusion of new features will not be difficult: the system can handle any number of parallel GDBs, although it now has only two to manage. One of the features we definitely intend to include is terrain data, contour lines and the like, in some form.

The current user interface is not operationally optimal. In particular, the necessity to go into a feature selection state to

alter the mix of displayed features is undesirable; the development of a selection capability using a medium other than a menu is needed. In addition, the SPECIAL mode is operationally superfluous.

Two nonstandard user functions are not yet available: crash zoom and relocation. Location now occurs only on initial start-up and the user cannot control the scale or center-point of the initial display. Zooming currently uses a fixed zoom factor $\zeta = 1.5$. However, crash zoom and location at an arbitrary center point and scale involve a change of context indistinguishable from the change of context involved in a zoom between detail levels, as far as the neighborhood construction process is concerned. Thus the inclusion of these two user functions involves only user-interface programming.

Provision of an interface between our GDDS and external DBMS's following the outline found in Section II's External Interface has not yet been made. However, this addition will require only a limited effort. To allow GDDS users to solicit DB information about a display item requires an item identification procedure and a request generator for the appropriate data base system. To allow external processors to alter a display based on their information requires the same display structure required for a dynamic foreground overlay. Both these capabilities can be provided in straightforward fashion.

The use of foreground overlays on the geographic background provided by GDDS is well understood. We have, in fact, used our leveled map with an air situation demonstration during the GDD development. The dynamic nature of such foreground data recommends storing the data hierarchically in a display tree. Hence, the overlay of foreground data will rely on FMP for storage and Pallet for display and will require no substantive additions to our current systems.

SECTION V

EXTRAPOLATION OF THE DESIGN

INTRODUCTION

In a sense, our view of GDD is limited. Our view is that of a theatre (or lower) C^2 center. Much of our design is unaffected by that provincialism, but some of the implementation decisions made might be difficult to generalize. We will discuss here the problem of enlarging the scope of our GDD implementation both incrementally and in quantum amounts.

INCREMENTAL EXPANSION

First we consider incremental expansion of the map data base--for example, extending the eastern boundary of the map to Moscow. Such an incremental change need not alter the original map data; each added block adds and changes a few entries in the various indices and adds data to the basic data base, but the general structure of the system remains constant.

Incremental expansion in another direction - in particular, north or south - would pose a different set of problems having to do with our display projection. Remember that our choice of map projection was influenced substantially by the geographic area of interest - its extent and its latitude. Incremental expansion outside the acceptable bounds for this projection might require a different projection. This fact recalls the tradeoff of speed for generality that we accepted in storing our map in projected form. Clearly a projected map image can be displayed more rapidly since the projection step for each point is not present. However, the necessity for reprojecting could be handled much more easily in a design where projection has already been integrated into the system. This problem recurs in quantum expansion of the

AD-A056 101 MITRE CORP BEDFORD MASS F/G 14/5
GEOGRAPHIC DATA DISPLAY, (U)
JUN 78 D E BELL F19628-77-C-0001
UNCLASSIFIED MTR-3315 ESD-TR-77-362 NL

2 OF 2
ADA
056101



END
DATE
FILMED
8-78
DDC

GDDS, so we will defer further discussion until we have covered that topic.

QUANTUM EXPANSION

Generalization of our system to cover the world instead of just Europe would be a problem of a different scale. Our system becomes the system module in charge of local operations. A radical shift of center point (from Berlin to Angola) would require:

- a change of the current map, or
- the real-time generation of an appropriate map for Angola.

These two choices represent the two extremes in a spectrum of possibilities. At one end, a selection of basic maps - blocked, projected, and stored - would be provided to the user. A user would select the best map for his purposes and display. The problem is to select the right maps to cover all contingencies. At the other end of the spectrum, the total system will maintain a "current map" (blocked, projected, and stored) and a full data base for the world. If that full data base is stored by latitude and longitude, a change of focus would require a rapid processing of the archival material to create a new current map to cover the new area of interest. The capability to create a new set of GDBs from central archives requires that we be able to do in near-real-time what we have done on this project in the last year:

- block the area of interest;
- project a map using an appropriate projection;
- filter details;
- divide the map into blocks;
- store the map blocks; and
- create indices to the map.

This capability is beyond us right now, but it is an obvious extension of our work to date. Its feasibility in some context should

be investigated in the near term.

A more realistic scheme for map management for a world-wide GDDS lies between the two extremes. This scheme would isolate a small number of likely areas of interest (say, Europe, Korea, and the Middle East) and prepare blocked-projected-stored maps of these areas. Any other area of interest (like Angola) would be covered by generating an appropriate map.

A last possibility for handling a radical shift of attention uses an archival data base not stored in latitude/longitude. The idea is to use a moderately faithful projection of the world for rapid displays of some exotic area. This world wide map would not be nearly as accurate as a custom map of the area, so that the immediate use of the raw data would parallel the generation of a custom map of the area of interest. This generation would involve:

- the inverse projection of the archival data to latitude/longitude form;
- the choice of an appropriate map projection; and
- the same map creation process as before.

A projection that shows promise for archival purposes is known variously as the Fisher and the Fuller projection. It projects the sphere gnomonically onto an enclosing tangent icosahedron, yielding 20 equilateral triangles. By joining these triangles appropriately a reasonable map about any region of the world can be constructed. Moreover, the relative computational simplicity of the gnomonic projection makes the recovery of latitude/longitude values from the projected map easy enough.

One aspect of the investigation of the feasibility of schemes should be an estimation of the processing time required to create a custom map. Based on the time required for our current method, a processing time of about an hour seems to be a reasonable figure. The determination of substantially longer or shorter times would be of considerable interest and the most realistic estimates should be ascertained.

SECTION VI

SUMMARY

A GDD design for the management of geographic data has been completed. It includes automatic detail control, user selection of display features, and the capability to overlay event data on the geographic background. The design manages different features as parallel data bases and reduces the display functions to table look-ups. Target response time is two seconds. A working GDDS exists which demonstrates detail control and user selection from a set of two features. In the near term there will be added an overlay capability, additional feature data, terrain data, and the ability to communicate with other C^3 data base management systems.

REFERENCES

1. Cornelius Ryan, A Bridge Too Far, (New York, 1974).

2. F.A. Jenkins and H.E. White, Fundamentals of Optics, 3rd Edition (New York, 1957).

3. Anne M. Molloy, "Geographic Data Base Development," ESD-TR-76-360, Electronics Systems Division, AFSC, Hanscom AF Base, Mass., January 1977 (ADA037116).

4. David H. Lehman, "Geographic Data Display Implementation," ESD-TR-77-137, Electronics Systems Division, AFSC, Hanscom AF Base, Mass., June 1977 (ADA044621).

5. Programming Reference Manual for Message Console, AN/FYQ-45(BR-90), The Bunker-Ramo Corporation, 1 April 1967.