Presented at an Advanced Study Symposium on
Topological Data Structures for Geographic
Information Systems, Harvard University,
Cambridge, MA, October 16 - 21, 1977

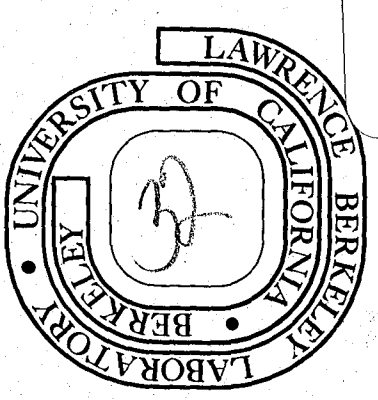
LBL-6490

INTERACTIVE DISPLAY OF POLYGONAL DATA

Peter M. Wood

October 1977

Prepared for the U. S. Department of Energy
under Contract W-7405-ENG-48



LBL-6490

INTERACTIVE DISPLAY OF POLYGONAL DATA

Peter M. Wood
Computer Science and Applied Mathematics Department
Lawrence Berkeley Laboratory
University of California
Berkeley, CA. 94720

October, 1977

ABSTRACT

The operations of interactive thematic mapping program are described which supports both quick display of retrieved data and detailed map design for printing. Emphasis is on the user view and map design functions such as data base arithmetic, definition of insets, shading algorithms, and on-line placement of text and figures.

Presented at An Advanced Study Symposium on Topological data structures for geographic information systems, Harvard University Oct. 16-21, 1977

TABLE OF CONTENTS

INTRODUCTION	1
USER VIEW AND SIMPLE MAP DESIGN	2
Input files	6
The setup phase	8
Simple map algorithm	10
MAP DESIGN FUNCTIONS	11
Data base arithmetic	12
Defining insets	13
Defining shades	19
Area shading by character	19
Area shading by lines	19
Line shading	22
Point shading	22
Adding titles and other information	25
EXTENSIONS	27
CONCLUSION	30
SAMPLE MAPS	30
REFERENCES	33

INTERACTIVE DISPLAY OF POLYGONAL DATA

Introduction

Interactive computer graphics is an excellent approach to many types of applications. It is an exciting method of doing geographic analysis when desiring to rapidly examine existing geographically related data or to display specially prepared data and base maps for publication. One such program is described in the following paper.

The interactive thematic mapping system called CARTE combines polygonal base maps with statistical data to produce shaded maps using a variety of shading symbolisms on a variety of output devices. A polygonal base map is one where geographic entities are described by points, lines, or polygons. It is combined with geocoded data to produce special subject or thematic maps. Shading symbolisms include texture shading for areas, varying widths for lines, and scaled symbols for points. Output devices include refresh and storage CRTs and auxiliary Calcomp or COM hardcopy.

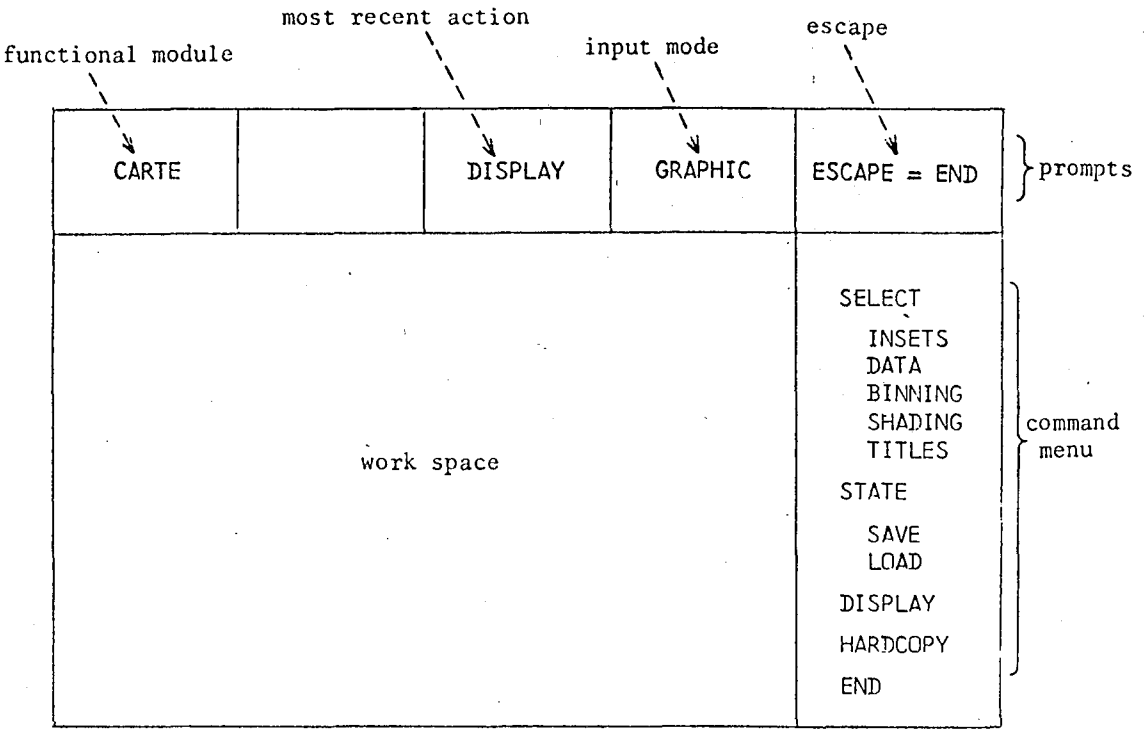
The system is designed to aid in the quick display of spatial data and in detailed map design. These two aspects of the program will be presented from the view of a user and then that of the underlying data structure and algorithms

which perform the various functions. The discussion will progress from those elements required to just make a map to those which support more sophisticated map design functions.

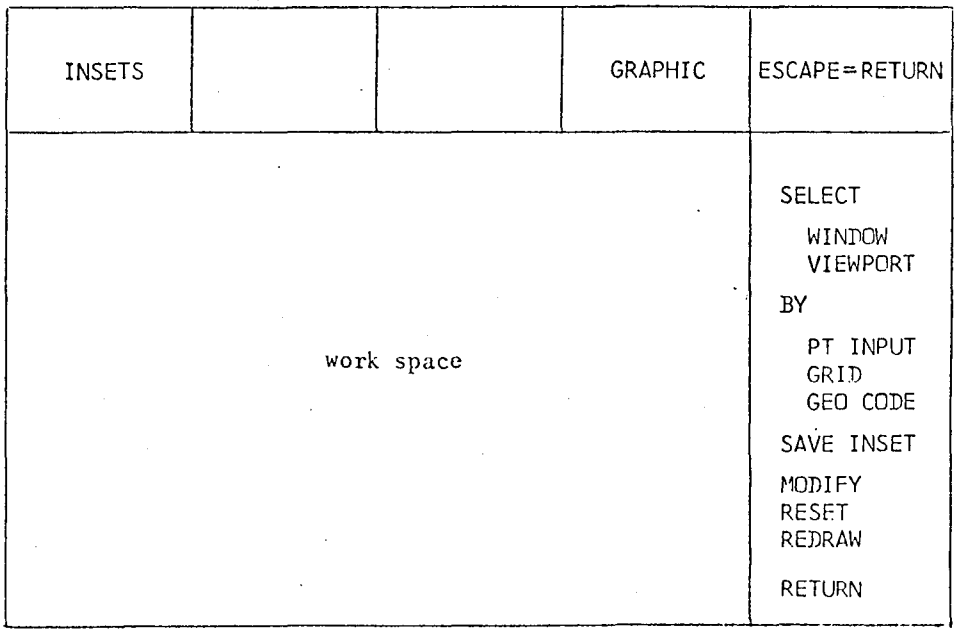
USER VIEW AND SIMPLE MAP DESIGN

When the user enters CARTE, he or she is confronted with a display much like those in figure one. Although initially overwhelmed, most users quickly adapt to the light pen menu format in which a command from the menu is selected for execution by a light pen or crosshairs. Many like the much reduced need for keyboard entries (in this system they are needed only for the specification of textual or numerical parameters, such as title text or character size).

One of the design goals was to allow an analyst to make one or more maps easily. Minimal labeling is needed as this mode assumes that the data and geographical area are familiar to the user. In batch mode programs, simple maps can be made with just a few directives. It is usually several hours or days before the user sees the results. With CARTE, a simple map can be made with a few (5) light pen hits and no keyboard entries. It can usually be viewed in just a few seconds. The computer cost is somewhat



Zero Level Commands



First Level Commands

Figure 1. The screen layout

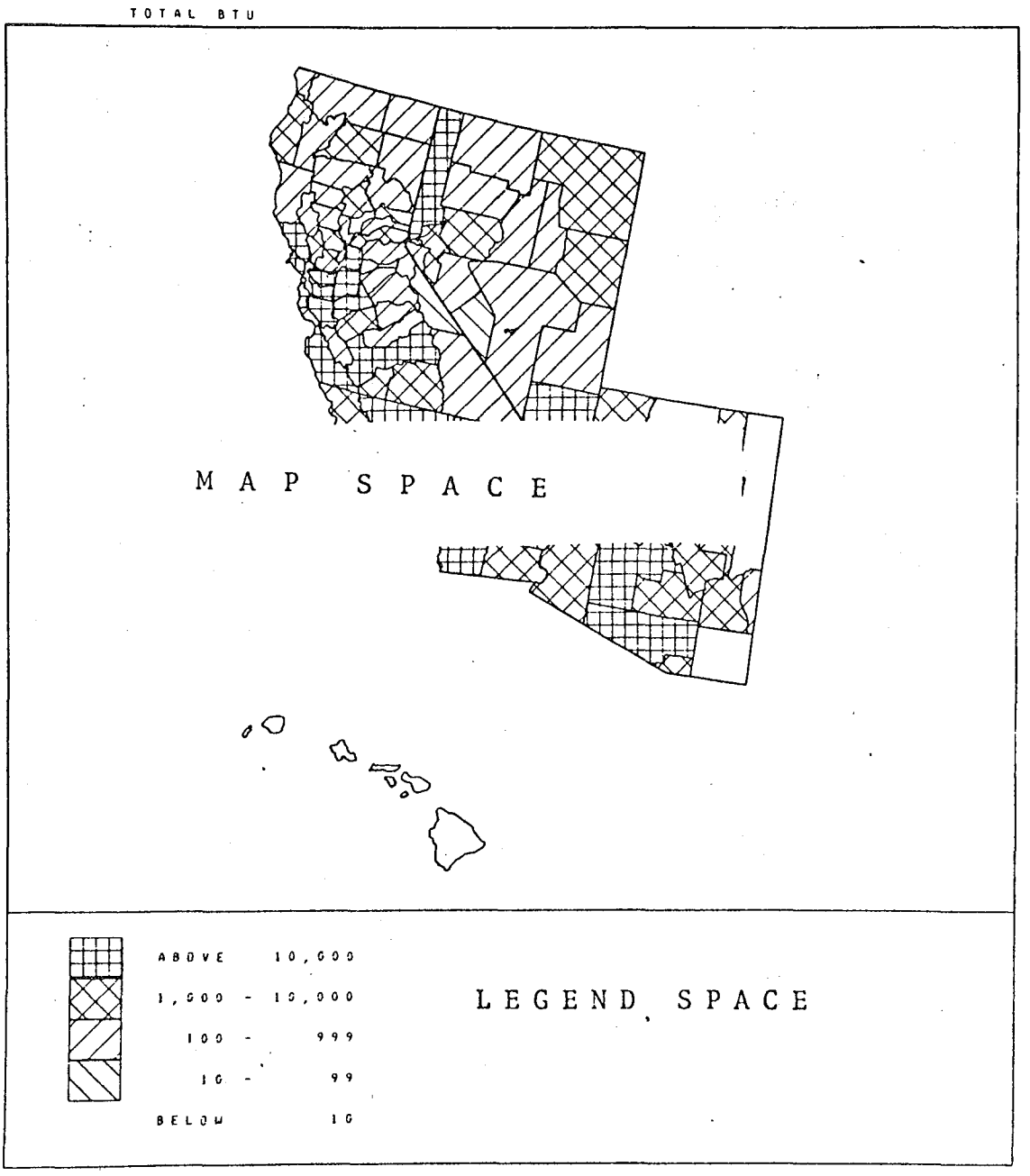


Figure 2. A simple map design

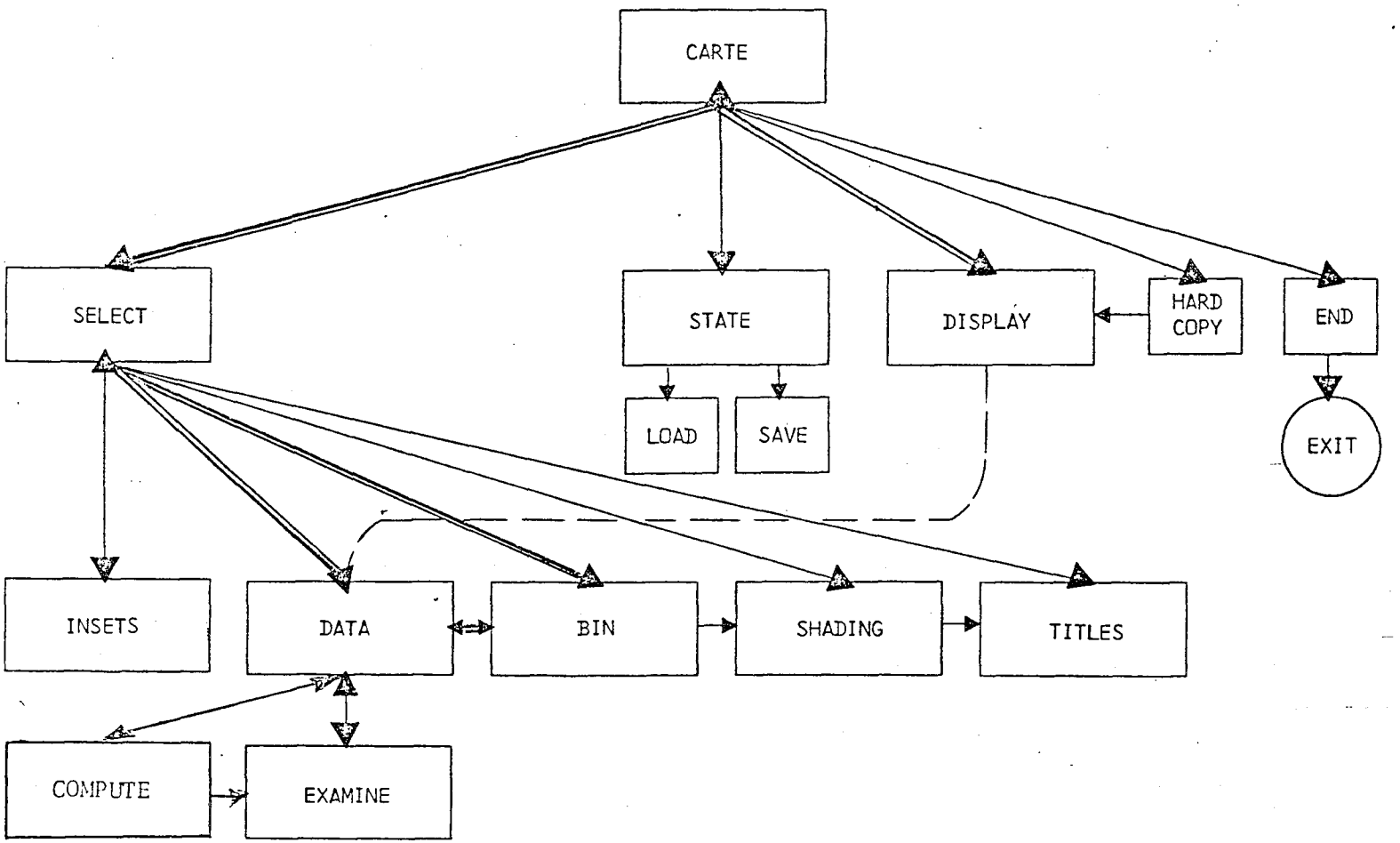


Figure 3. Possible flow of control

higher for interactive mapping, while the cost in a user's time is much lower. Figure 2 shows the format in which a quick map will appear.

Command flow possibilities are shown in Figure 3. The path followed in simple map-making is indicated by the double lines. Each of the commands at the first level or under SELECT is a module in the specific sense of being a separate subroutine with no parameters. Communication between these modules is by labeled common blocks and arrays stored on disk. Careful selection of the data structure elements to be fixed in the common blocks and of those to be stored on disk is needed to maximize program flexibility and minimize overhead. In CARTE any data structure element having a possible size greater than 64 words is made into a variable length array. Elements fitting this definition are the space for storing the points of a polygon (only one is in core at once), elements that are a function of the number of areas in the data or the number of areas on the mapfile, various workspaces, and the amount of supplementary identifying information (titles and figures).

Input Files

While CARTE proper is a stand alone subsystem for entity-driven mapping, it expects as input files which are the output of two other SEEDIS[1] subsystems, MAPEDIT and

DO BE DO. Together these three form the basis for the polygon-based mapping system in SEEDIS (Socio-Economic-Environmental-Demographic-Information System) at the Lawrence Berkeley Laboratory[2].

MAPEDIT[3] prepares a base map of outlines for mapping in a standard format called a nickel format. It consists of what may be called a bounded location list structure. It is a location list in the sense of consisting of geocode identifiers and all points for a single entity. It can be called bounded because it also has the points of the rectangle bounding the entity, as well as the centroid. These are important elements, simplifying several operations when making shaded maps. The advantages of this structure are that 1) an entity can be retrieved in just one binary FORTRAN read, 2) determination of its inclusion in an inset requires the comparison of only two already calculated points, and 3) certain types of shading can be done without reference to the actual points of the entity. Its disadvantages are well known in that 1) explicit adjacency information is not contained and 2) several inefficiencies occur because of duplicate boundaries.

DO BE DO prepares a data base for mapping so that all the values of a characteristic can be accessed also in a single FORTRAN binary read. This is the inverted file structure discussed in a previous report[4]. It allows

the mapping program to efficiently access the data base for all values of a characteristic for binning and mapping, for the names of all characteristics (a data base directory), as well as for data base geocodes and how they correspond to map geocodes.

The Setup Phase

It is these two files that are combined under user direction to make thematic maps. They are linked by the map directory as shown in figure four, during the setup phase of the program. The algorithm is:

1. Read data base id to identify data key records
2. Load data key records into core
3. Set map keys (geocodes) to match with data
4. Read the mapfile to get an entity record
5. Enter into the map directory:
   a. address of the entity on the map file
   b. type (point, line, or polygon)
   c. pointer to corresponding data area (result of key match)
   d. hierarchy indicator (shade only counties when states are present)
6. Save other available information as needed:
   a. Centroid and extent
   b. Names and geocodes
7. If more entity records go to 4

Pre-calculating the correspondence between map areas and data areas eliminates the need for key matching each time a map is drawn. The centroid is essential for certain types of shading and when insets are drawn. Let us first however, discuss the simple mapping algorithm. The data structure which is its base is diagrammed in Figure 4.

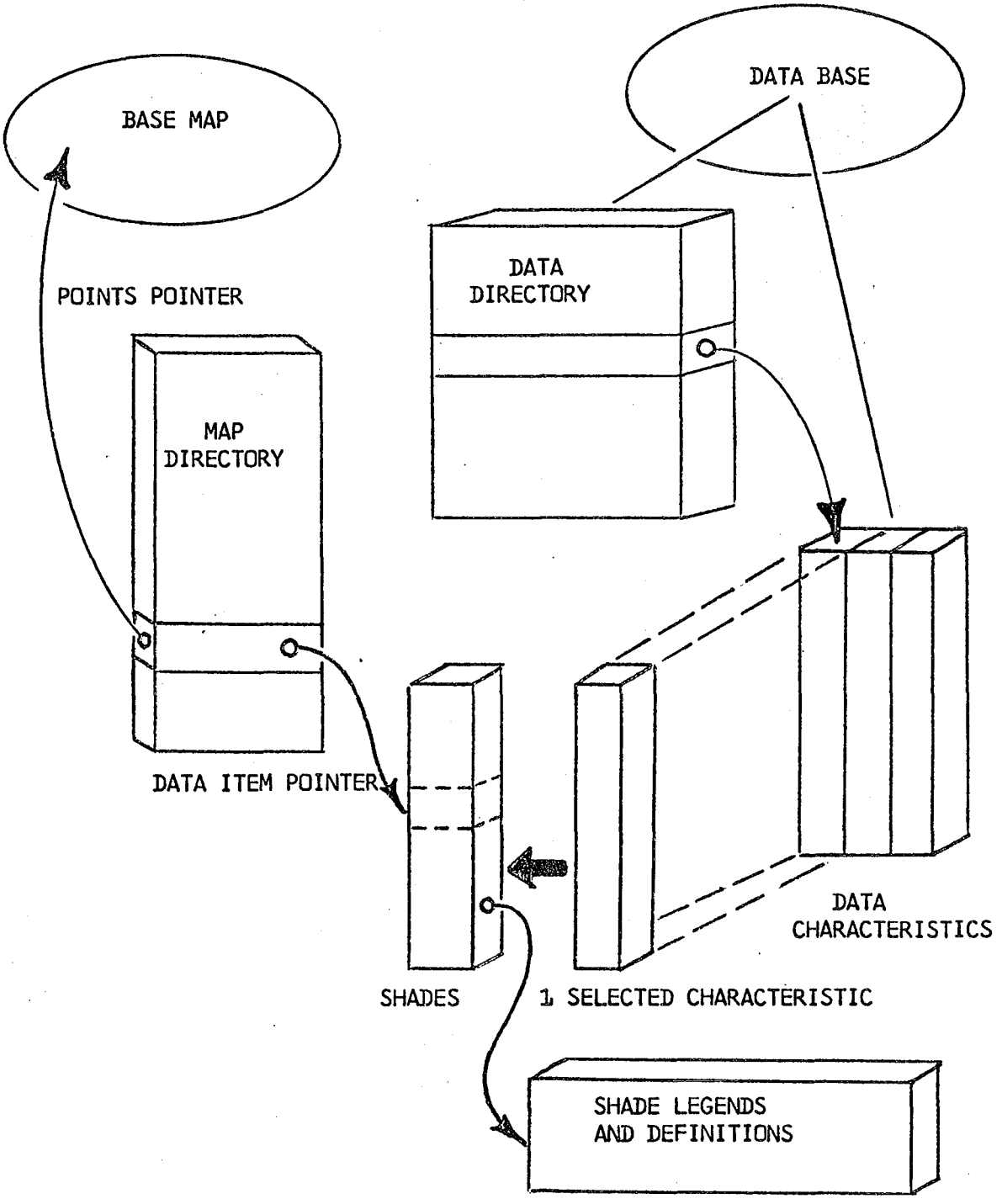


Figure 4. Basic Map Data Structure

Simple Map Algorithm

Here is the algorithm used in CARTE to generate a map. Light pen hits of the user are followed by the operations they initiate. The same algorithm is followed for all maps whenever these functions are performed, with additions of new data structure elements.

*** hits data select -- call DATA subroutine

1. Fetch data directory from disk
2. Display names of characteristics on screen (assign each to a rectangle)

* hits a characteristic name

3. Locate name from position of hit on screen
4. Copy information for that characterisitic to working data directory
5. Display name in selection list

*** hits done -- return data directory, call BINNING subroutine

1. Fetch selected data record from disk
2. Allocate work space for shades (one per area)

* hits generate intervals command

3. Bin the data, generating dividing values and shades for each area
4. Create legends for shades by calculating size of values, encoding with commas, and combining
5. Display legends and counts for each bin

*** hits done -- area shades saved on disk, data record returned

*** hits terminal display -- enter DISPLAY subroutine

1. Display title (characteristic name at top of screen)
2. Display legend for each shade
   a. generate and shade legend box
   b. display legend text
3. Display the map body
   a. Fetch area shades

3.
    b. Fetch map directory
    c. For each entry in map directory
        1. Set plotting limits
        2. Fetch and draw its points
        3. Look up its shade
        4. If non-zero, then shade it according to the corresponding shade definition
4. Return and wait for the next command

There are several algorithms for displaying a map. Looping on shade value is the technique for producing color separation negatives. It is not useful in this application as it requires two disk accesses per entity, one for the outline and one for the shading. Thus the two were separated. Looping in data area order was ruled out because more than one entity often occurs for one data value, e.g. the islands of Alaska. Looping on map entity is easier to program because one region corresponds to only one data area, and requires the points for a polygon to be fetched just once. In more complicated designs, when multiple insets are defined, an entity is plotted for all its insets before the next entry is examined.

MAP DESIGN FUNCTIONS

The secondary design goal allows the user to do much more than simply display given data with a given map. Arithmetic may be performed on data characteristics, insets may be made from the base map, shading symbolisms may be defined, and identifying information such as titles and legends may

be added to enhance the map's impact. Figure 12 gives the complete data structure elements of CARTE. Map 1 is an example of more complicated design.

Data Base Arithmetic

Performing calculations using data characteristics is a subfunction of data selection. It is useful for changing units or calculating percentage change between two characteristics.

The operation is invoked simply. Characteristics to be used as operands are selected, as if to be mapped (and in a sense they are). The COMPUTE command is selected, and the desired expression is entered. If successrul, the user views the result and then continues on to bin the data as with any other characteristic.

Evaluation of the expression is done by a module written by Peter Kreps of LBL. The module works with vector operands. The expression R2 - R1 means simply that for each area in the data set, record 1 is to be subtracted from record 2, where record 1 is the first data characteristic selected into the working data directory and record 2 the second. The expression is scanned for operators, scalar or vector operands. Operators are put on an operator stack, scalars are expanded to a vector and put on an

operand stack. A characteristic operand is first fetched through the working directory and then passed to the module to be put on the operand stack. Thus the operand stack is n x m words in length, where n = vector length and m = number of operands. At each stage syntax is checked for a valid arithmetic expression and, if possible, the stack is reduced by performing the indicated operation. When the expression has been completely processed, the final result may be called for and is returned in an array to the data module. Data base arithmetic can greatly enhance a user's ability to analyze data.

Defining Insets

Often some portion of the base map is either extraneous to the interests of the user or consists of many small areas that are indiscernible. One wants to zoom in on a portion of the base map, thus creating a window. The portion of the screen through which the window is viewed is a viewport. Together a window and viewport define an inset.

Inset definition has been generalized in CARTE to provide a multiple inset capability. This allows several disjoint areas to be displayed together as well as more detail of one area. Windows can be defined by graphically input geometry (i.e. two points) and geocodes, while viewports may be defined by graphic input or by user defined grid. This

makes it easy for a viewport to fill the entire screen or a quadrant exactly.

Data structure elements needed to provide this capability are minimal: arrays for the two points defining each window and viewport and a byte for the type of geocode selection (currently limited to all states or a specific state). The difficulties arise in ensuring complete definition of an inset.

A temporary window and viewport are defined. Information is kept on which parts of the inset definition have been filled in. As points are entered, old ones are replaced. The user is required to specifically save an inset and this can only be done if it is fully defined. The algorithm for inset definition is shown in Figure 5. Figure 6 gives some examples of inset.

Another difficulty is preventing them from distorting the map. This requires adjusting the window to fit the aspect ratio of the viewport.

Displaying the desired map outlines becomes a simple operation. The map directory and centroids are retrieved into core. Cycling thru the map directory, the centroid and extent of an entity are compared with those of the insets. For those windows the entity falls within by geometry and geocode the plotting limits are set and the points are drawn.

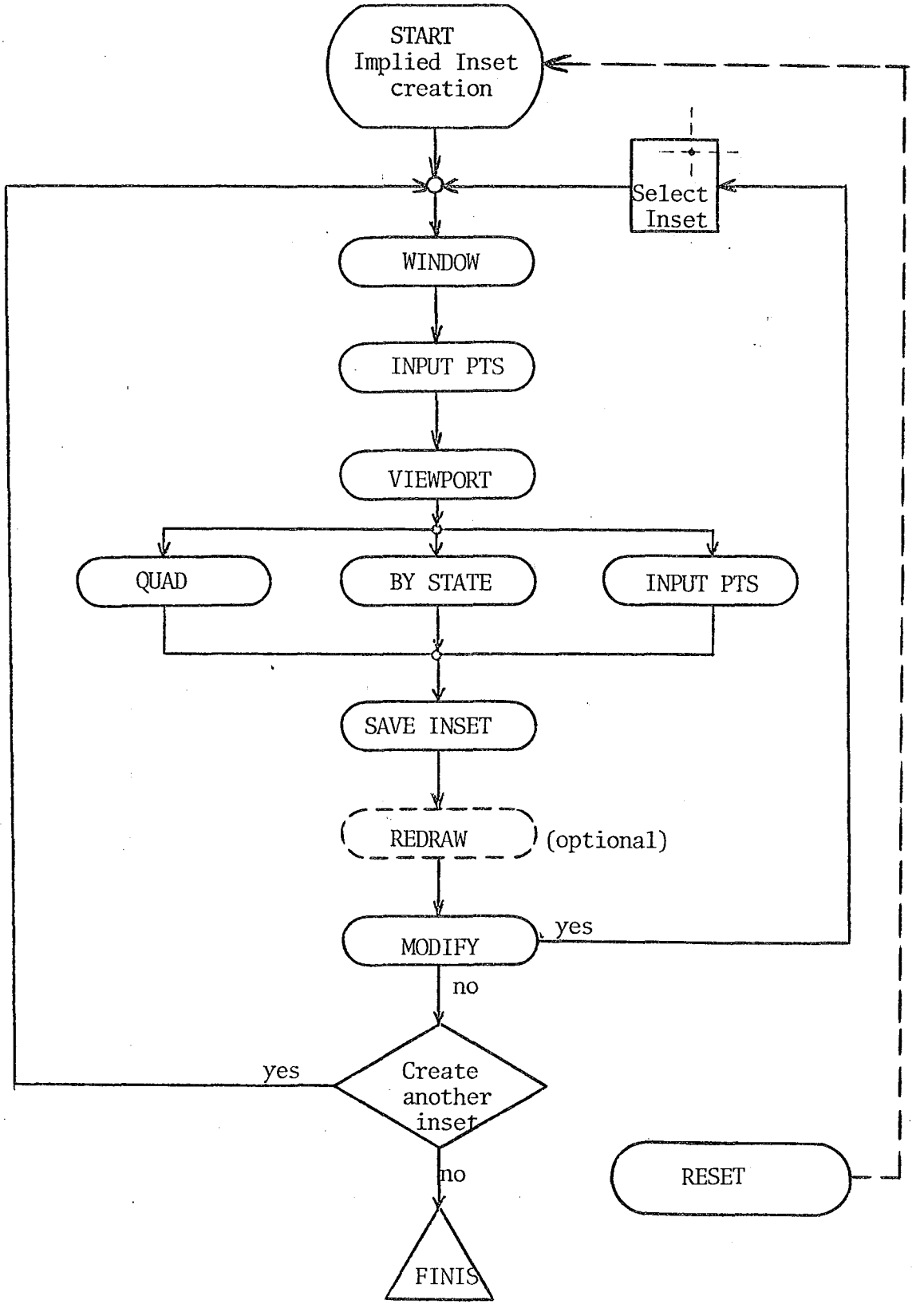


Figure 5. INSETS Flow Diagram

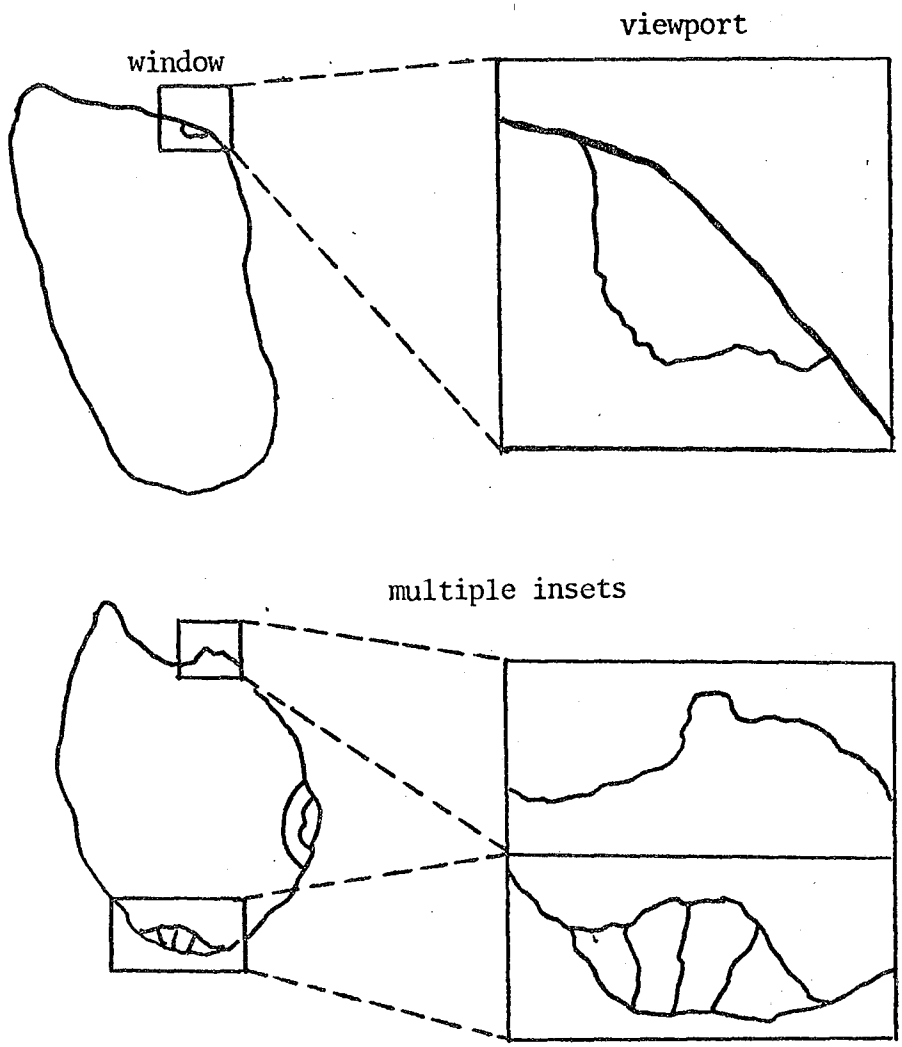


Figure 6. Examples of insets

The display algorithm requires at most npolys + 2 disk accesses and 2 x npolys + 2 x npts words of memory, where npolys = number of entities in the map directory and npts = maximum number of points. (This is for a CDC 6000 series computer where the four fields of the directory and the centroid and extent are packed into 2 words. On the other hand, each point of an entity is allocated 2 words of memory.) The process can be done either each time a map is made or once, but adding extra fields to the directory. This is not critical as the operation is I/O bound, and the largest portion is spent writing the display file. The largest proportion of central processor time is used by the generation of vector character titles and shade lines.

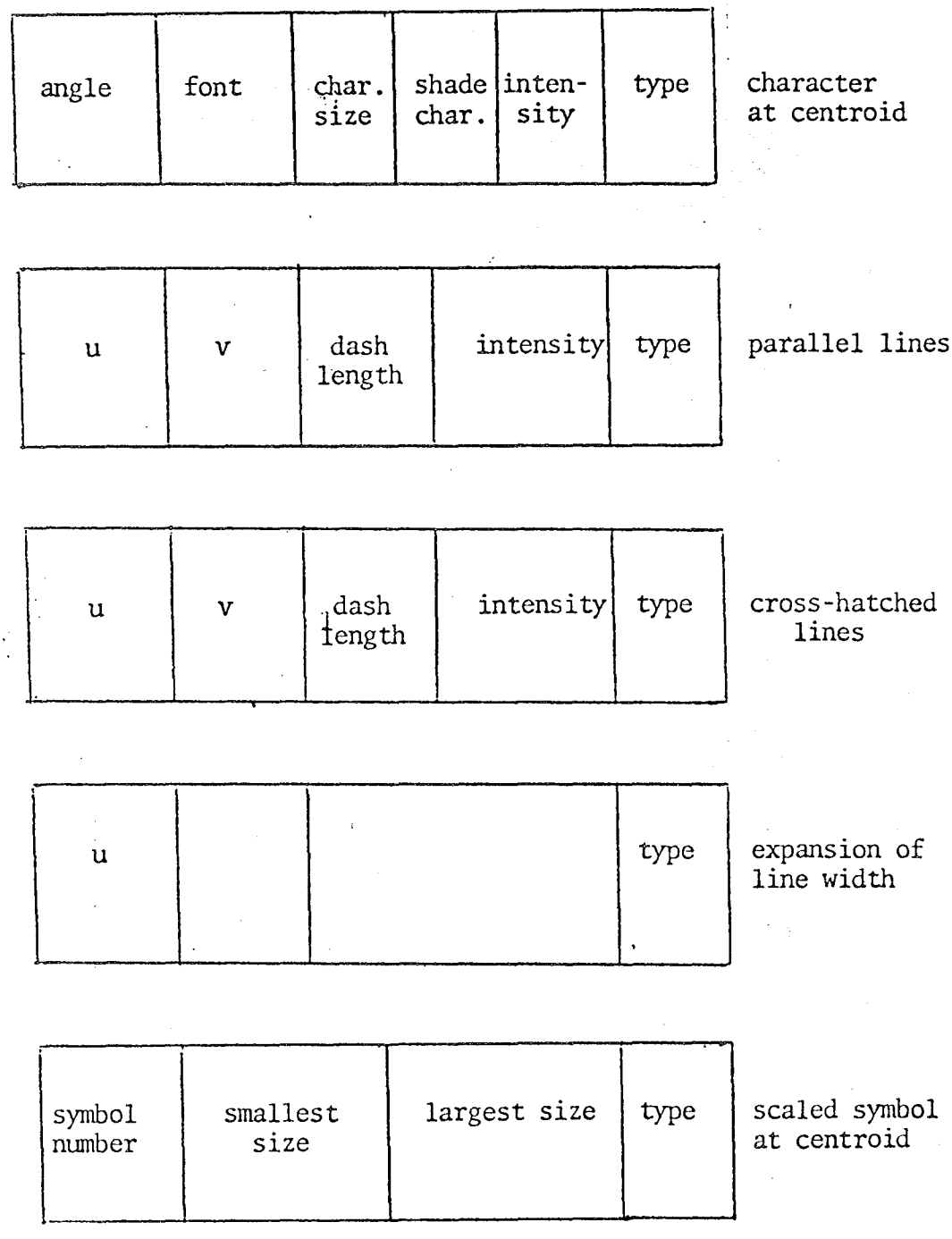


Figure 7. Parameters of the various shade types

Defining Shades

A user may define several shades, view them, and then select those preferred for use in the map. The parameters of each type of shading are diagrammed in Figure 7. These shade parameters, once defined, are packed into a shade definition word, and unpacked or set when used. Shades may be defined for the three types of entities: areas, lines, and points.

Area Shading by Character

Polygons can be shaded by a character placed at its centroid. This is useful particularly for refresh CRTs to avoid flicker problems. Since the centroids can be fetched by one disk access, a map can be shaded very quickly. The major parameters are the character itself and its size and intensity.

Area Shading by Lines

Polygons may also by shaded by using parallel or cross-hatched lines. Both these types have the same parameters. U and V in Figure 7 define a vector about the origin which determines shade line orientation and spacing. For cross-hatched shading, the shade algorithm is called a second time with a vector perpendicular to the original.

The shading algorithm differs from previous ones which generate all shade lines for a polygon in core. The memory required for this approach is unavailable to an interactive program. Thus a fast vector shading algorithm which minimizes the use of memory was developed by Harvard Holmes of LBL for interactive shading applications.

The method is as follows:

1. Rotate and scale the points so that the shade lines are horizontal and spaced one unit apart.
2. a. Initialize pointers to the points in a temporary array, K.
   b. Sort the pointers so that they select points in order of ascending y value.
3. Set initial y coordinate. Set pointer to beginning of K.
   a. Select points using K, between the last scan line and the next scan line. Include either or both of the associated edges if they extend beyond the next scan line.
      Insert the edge data into the edge intersection list.
   b. Transfer the x coordinates (from the edge list) and the y coordinate (from scan line value) back to the original space and output the shading segments.
   c. Remove completed edges and update others. If any edges are left, go to step 3a.
4. Empty the plot buffer
5. Rotate the points back to the original coordinates.

The algorithm is fast. Most of the time is spent in step 2b, sorting the points by y value. Combining cross-hatched lines with dashed lines gives some very interesting textures. Many of the resulting patterns resemble those of woven fabric.

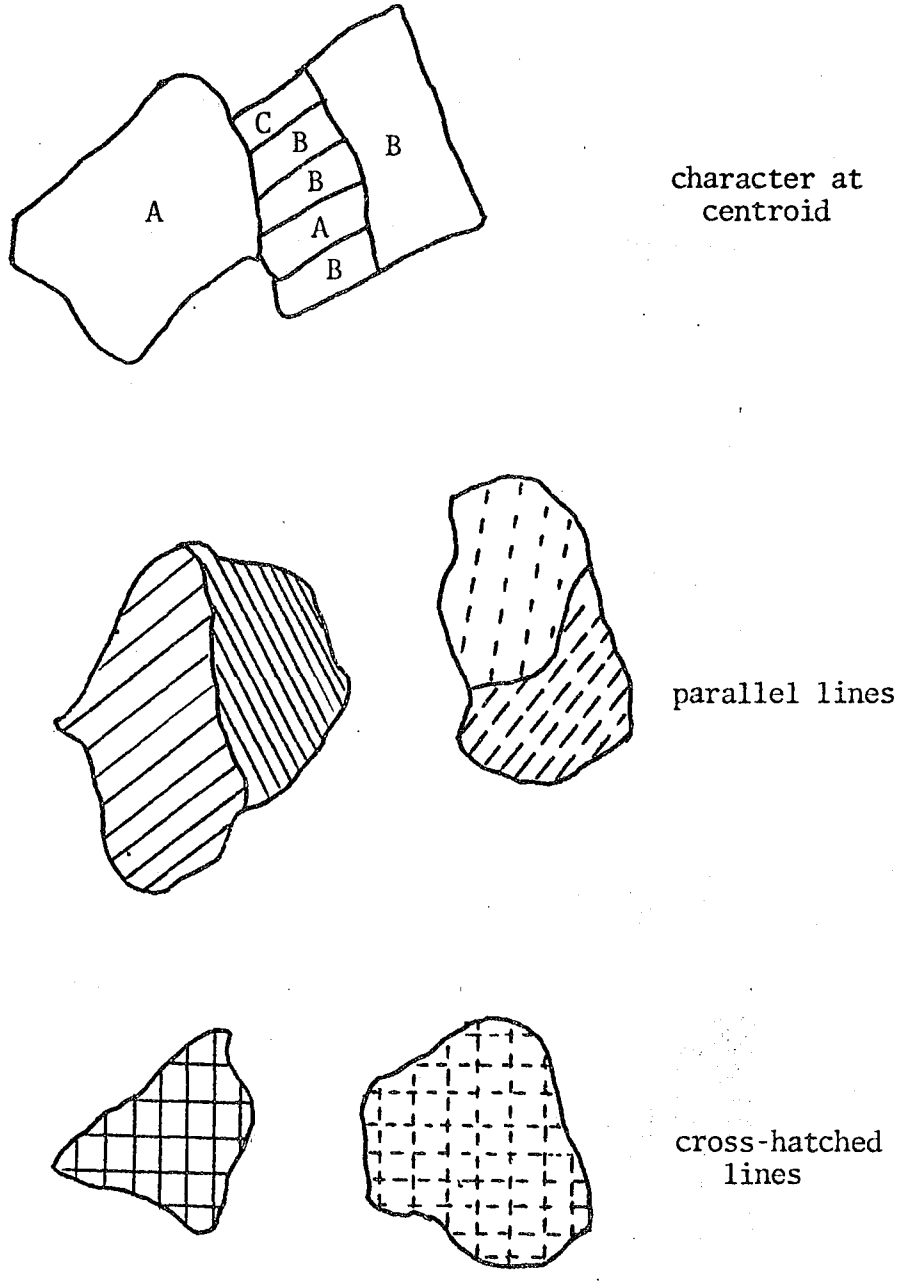


Figure 8. Examples of area shading

Line Shading

Line shading is done by expanding the line in width. This width is variable, being user-defined, and actually defines the width of a rectangle which is first constructed and then shaded to give the appearance of a solid line. The method is as follows: (see Figure 9)

1. Calculate x and y offset such that $p_3p_2 \perp p_1p_2$ and the distance from p_3 to p_2 = u/2.

   delx = u/2 * -cos(a) = u/2 * (-dx)/dis , dx = $x_2 - x_1$

   dely = u/2 * sin(a) = u/2 * dy /dis , dy = $y_2 - y_1$

   where u = user-defined width and dis = sqrt($dx^2 + dy^2$).

2. Create a closed rectangle from the old points p_1 and p_2 and the x and y offsets, balance about the old segment so two segments of opposite direction will meet correctly.
3. Draw and shade the rectangle.

Point Shading

Points are represented in interactive CARTE by symbols that are scaled by data value. This differs from the batch symbol shading where symbols are filled with a texture corresponding to data value[5]. Symbol points are drawn in by the user and a minimum and maximum scale defined, the minimum corresponding to the lowest bin, the maximum to the highest. Since the number of symbols can vary as well as the number of points per

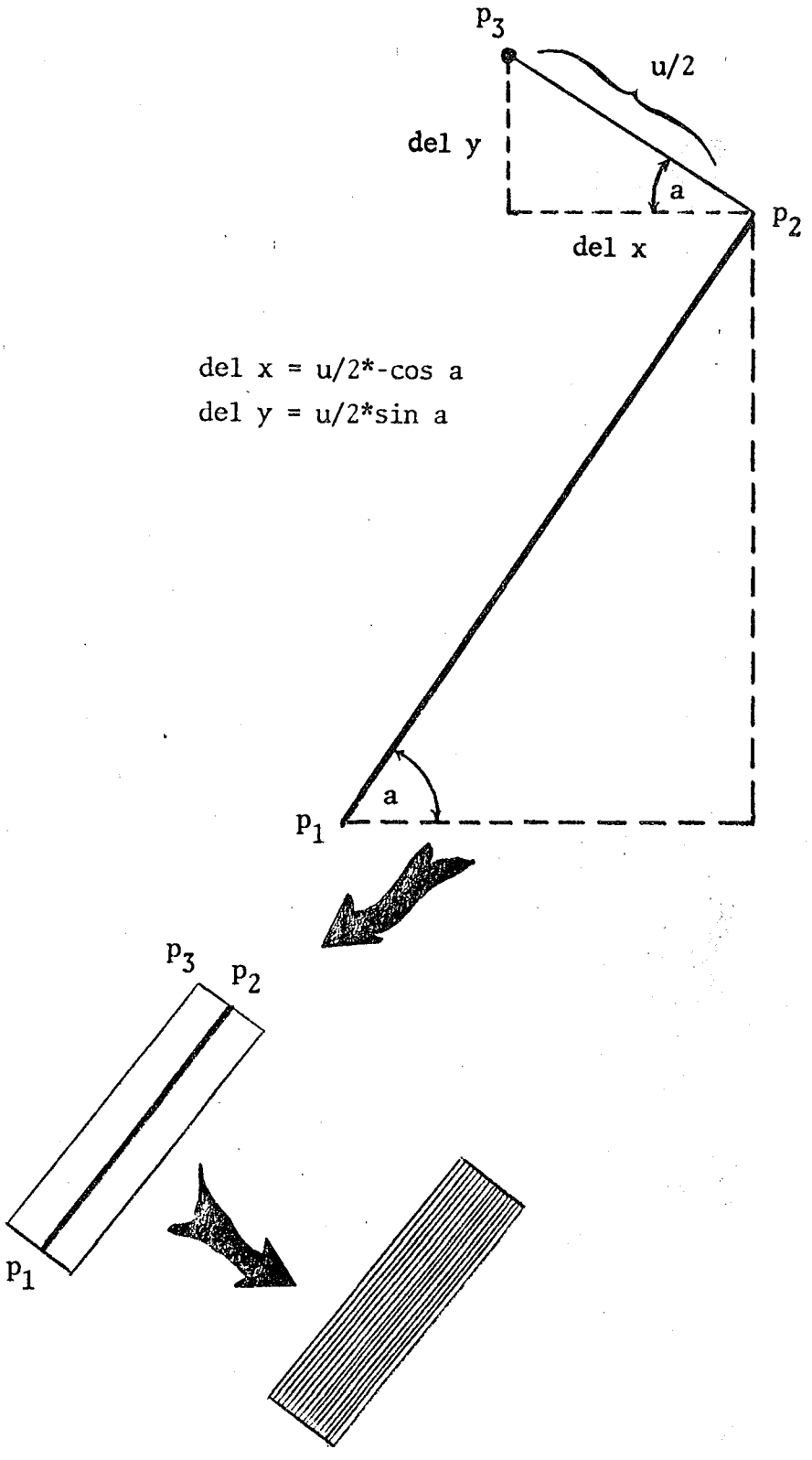


Figure 9. Steps in shading a line

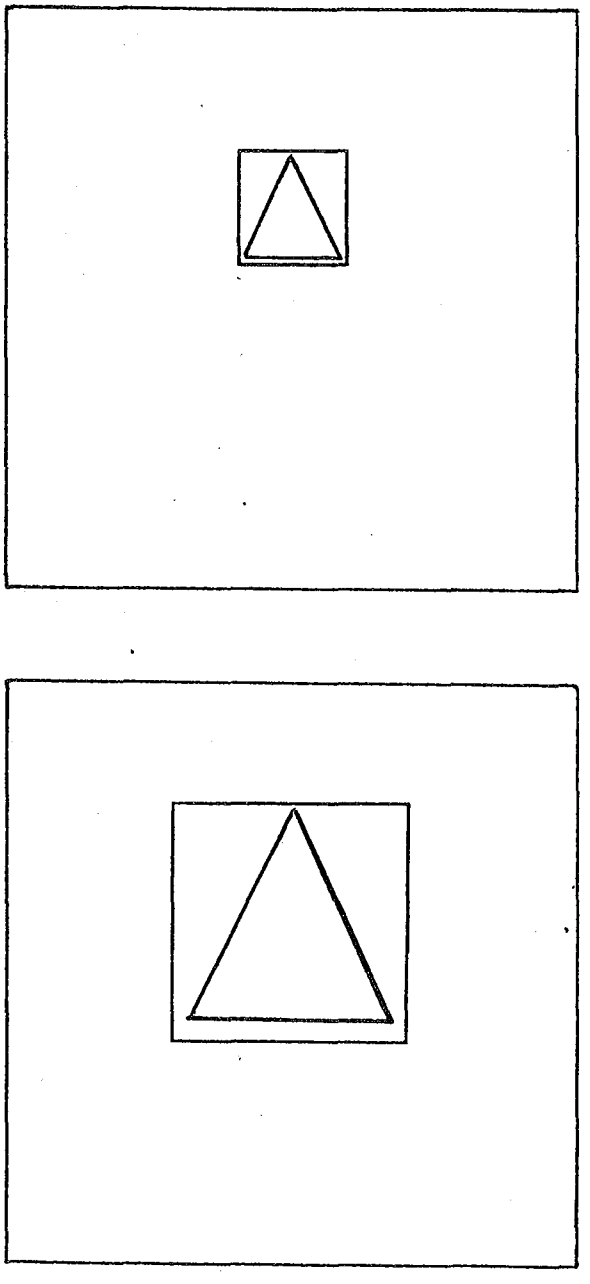


Figure 10. The method of point shading

symbol (there is no limit), the symbol definitions are stored with other identifying information in the title array.

When a symbol is to be drawn, its points are retrieved from the title array. It is scaled by defining a temporary window and viewport such that the symbol fills the window and the viewport is scaled by data value and centered at the point (which has been stored as the centroid in the map directory).

Adding Titles and Other Information

Adding identifying information can greatly enhance map impact. This includes user-defined titles, legends, and figures. The number and size of these elements varies widely from map to map. The information is stored in a dynamic array, with each element pointed to by a word in common. This data structure is diagrammed in Figure 11.

Titles can be in either hardware or vector characters. If vector characters are specified, one of eight fonts may be chosen and the character size becomes a continuous variable.

A legend is specified by two points and the type of entity. The rectangle defined by the two points is subdivided into smaller rectangles and the appropriate shade displayed in each.

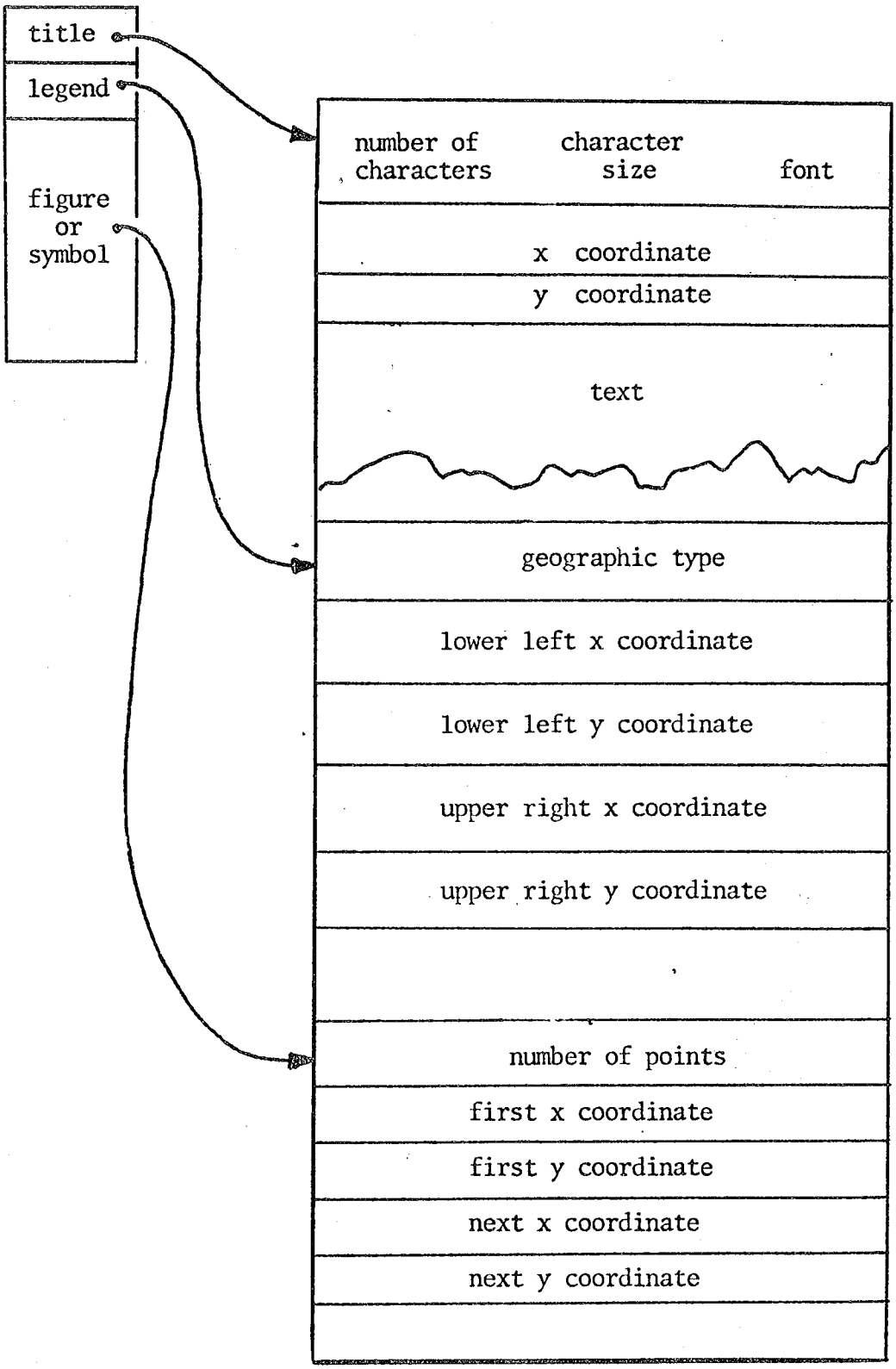


Figure 11. Data structure for added text and lines

Figures and symbols are simply a collection of points. Figures requiring more than one line can be described using several figures.

Any information of these types will be pointed to by a work in the control array. Its index number in this array is its name and is used to create, modify and delete these elements. The index number also determines the order in which the elements are drawn.

Extension

It may be useful to be able to tie identifying information to an inset. Then a simple redefinition of the inset would translate and scale all text and figures associated with the inset, as well as any geo-entities as in now done. An initial implementation was inconclusive as to the value of this capability.

Binding an inset explicitly to a characteristic adds another dimension in map-making capacity. It essentially provides the capability for multiple maps on one display, either side by side or as visual overlays. The value of this capacity is also not clear. The size of each component would be reduced, perhaps limiting legibility. The capacity constitutes the generalization of implicit data structure relations, requiring small data structure modifications,

while adding another dimension to the display.

Conclusion

The potential of interactive computer mapping is great. Algorithms have been presented here for the quick display of spatial data and the design of maps of publication quality. Substantial savings in people-time can be realized using interactive thematic mapping. The time between designing a map and viewing the result is reduced from hours to seconds. The value to those engaged in map production is clear. However, its main value in the future may be for quick displays of data to analysts, managers, and interested citizens.

Work performed under the auspices of the U. S. Department of Energy.

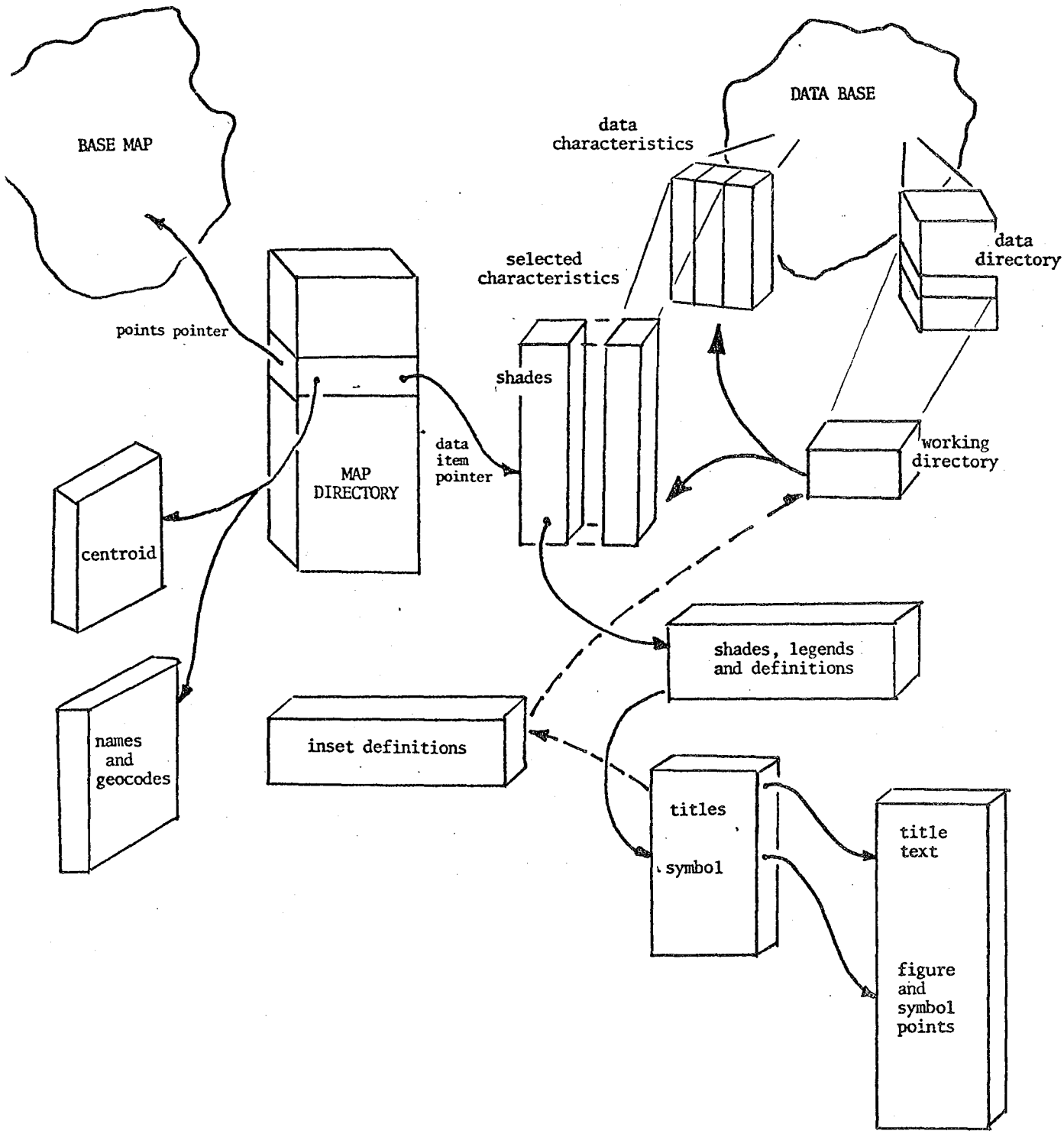
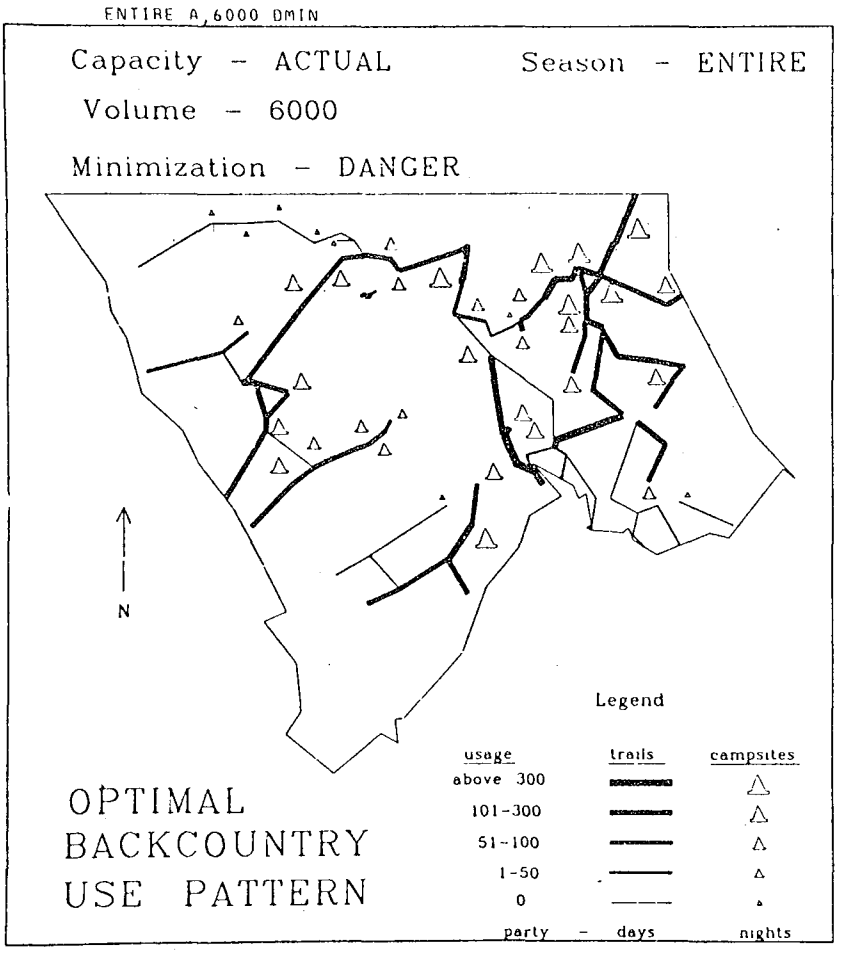


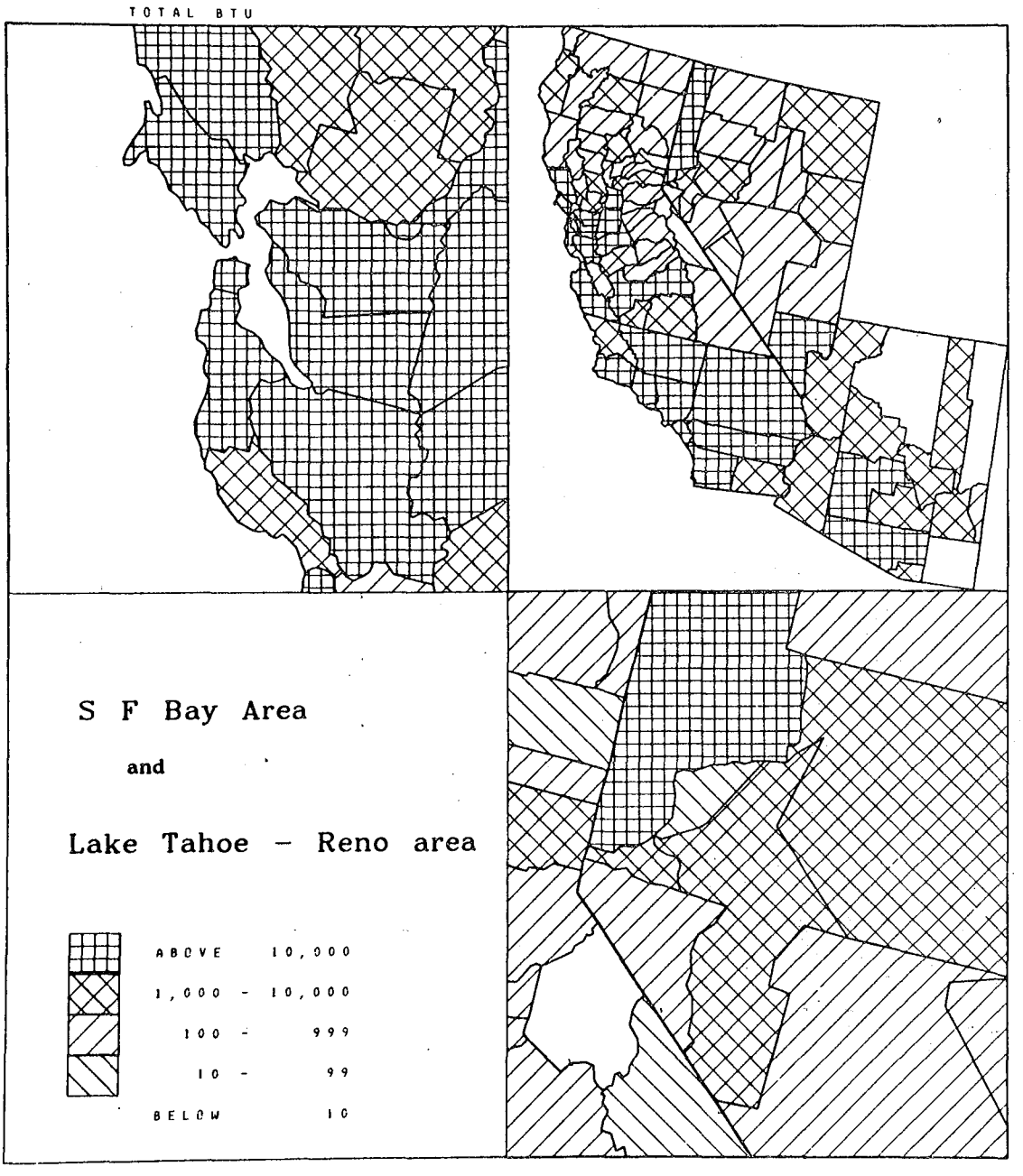
Figure 12. The major elements of the data structure



XBL 772-7595

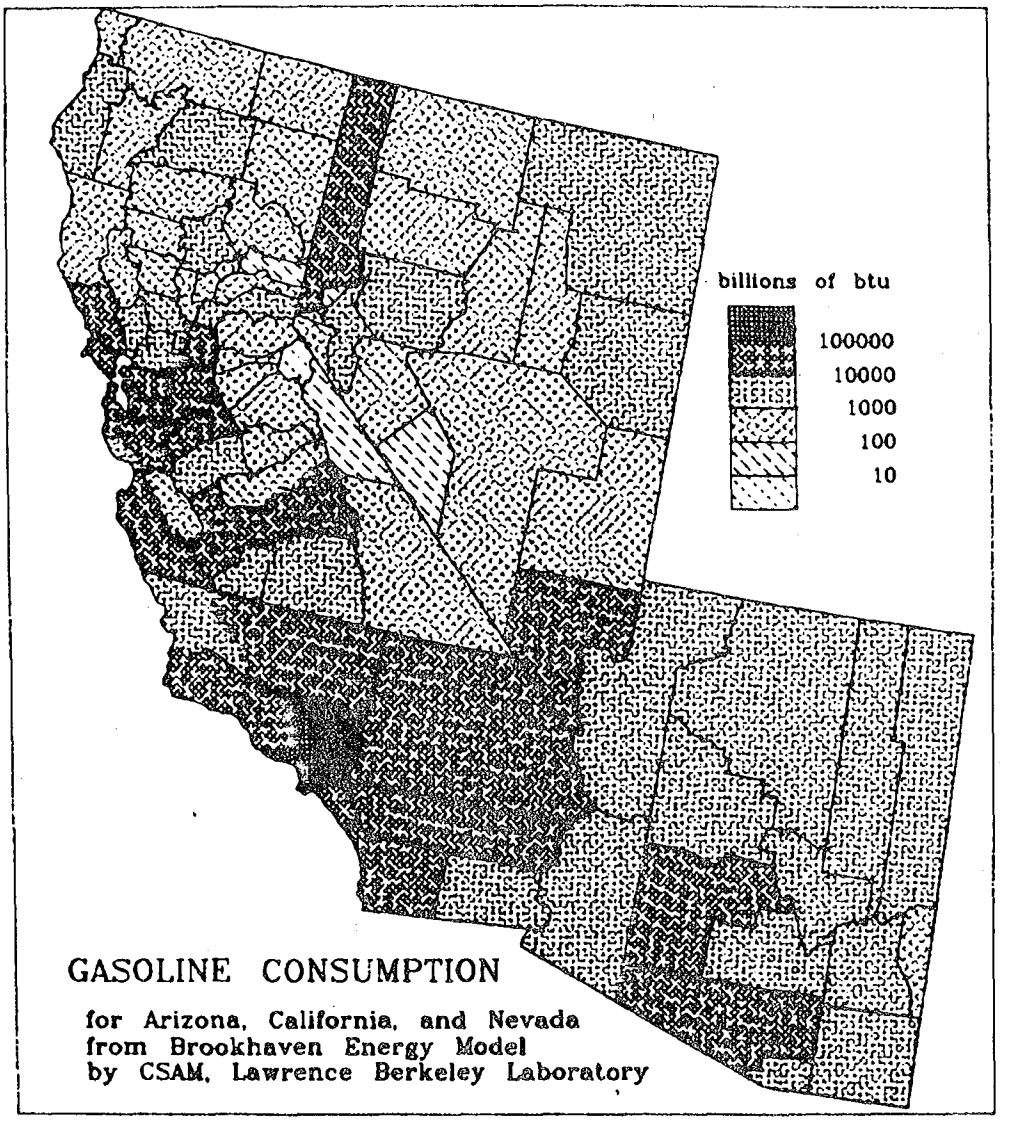
Output from a linear programming model is mapped using line and point symbolism. The design was done interactively and included designation of shading and placement and entry of titles and figures. The model minimized dangerous bear contacts for northern Glacier National Park.

Map 1. An example of complex map design.



XBL 776-9247

Map 2. An Example of Multiple Insets



XBL 772-7594

Map 3. Some Possible Shading Textures

REFERENCES

1. Austin, D.M., Kranz, S.G., and Quong,C. An Overview of the LBL Socio-Economic-Environmental Information System. LBL-3699. March, 1975.

2. Burkhart, B. and Wood, P. SEEDIS Workbook III, Interactive Polygon Mapping. LBL-6439. May, 1977.

3. Holmes, H.H., Austin, D.M., and Benson, W.H. The MAPEDIT System for Automatic Map Digitization. LBL-3072. August, 1974.

4. Wood, P.M. Interactive Thematic Mapping -- A Report. LBL-5362. October, 1976.

5. Wood, P.M. and Austin, D.M. CARTE - a thematic mapping program. LBL-3073. July, 1974.

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720