A GENERALIZED CARTOGRAPHIC DATA PROCESSING PROGRAM

Denis White
Laboratory for Computer Graphics and Spatial Analysis
Graduate School of Design, Harvard University
Cambridge, MA 02138 USA

ABSTRACT

The Harmony program, developed in 1980 and 1981, is a general computer tool for modifying, transforming, and displaying cartographic data. It is a companion program to the Harvard Odyssey system and shares many conventions with that set of programs. As a program managing sequential files of cartographic data, it provides for the creation of new fields in one or more output files by combining one or more input fields with arithmetic, relational, or logical operators in arbitrarily complex formulas. Set membership evaluation, ordinal ranking, numerical recoding, and various geometric transformations including map projections may be included in the formula expressions. One or two input files may be synchronized on one or more key fields in order to aggregrate or disaggregrate data. Records for output may be selected by an arbitrarily complex formula expression. As a display program, Harmony allows the description of the color, pattern, and size of data symbolism by the same formula mechanism used in data management. An adjunct program called Picture provides a two dimensional figure drawing capability for preparing titles, graphic scales, north arrows, and similar map elements.

INTRODUCTION

As more geographic information is recorded, processed, and mapped with the assistance of computers, programs with a wide range of capabilities are more and more needed. This paper describes a program which acts upon cartographic data like a programmable calculator acts upon numbers. It has a language in which the cartographic elements and their attributes are the objects to be manipulated and various mathematical, logical, and geometric operations are the functions that do the manipulation. Other approaches to cartographic data processing similar to this are Berman and Stonebraker (1977), Burton (1979), Cox and others (1980), Tsurutani and others (1980), and Frank (1982). The program, called Harmony, was developed in conjunction with the Odyssey System (White 1979, Chrisman 1979) with which it shares a number of features. Harmony is built upon a conceptual model of cartographic information, explained in the first section of the paper, using an entity-relationship model of data (Roussopoulos and Yeh 1984).

Subsequent sections of the paper describe the capabilities of the program and the language used in the user interface. The final sections show examples of the use of Harmony and discuss supporting programs.

CONCEPTUAL MODEL

The fundamental structural notion of the cartographic model of Harmony is the decomposition of the landscape into named (or numbered) cartographic elements with the geometric form of points, lines, or areas. As such this program, and all others like it, differ radically from most image processing approaches to cartography which process and map continuous spatial fields of information without identifying individual features. Point, line, or areal features in Harmony are the entities of the cartographic model and they are associated with one another by a set of relationships (or relations). Relations are defined in a general way and include the relations between a feature and its name, and a feature and its location, as well as more sophisticated relations such as that between a feature and its topological neighbors. The model consists of the classes of entities and their relations. Below is a table of some of the relations used in cartographic data processing.

TYPE OF RELATION NAME OF RELATION	DESCRIPTION OF RELATION
Identification Relations	
Name	Entity's "name is " (e.g., geocode)
Location	Entity's "location is" (coordinate position)
Spatial Relations	
Scale	Entity "contains" another entity
Order	Entity "precedes" another entity
Boundary	Entity "bounds" another entity
Interior	Entity "has in its interior" another entity
Intersection	Entity "intersects" another entity
Congruence	Entity "is congruent to" another entity
Dimension	Entity's "spatial dimension is"
Distance	Entity's "distance from" another entity "is"

Table 1. A Taxonomy of Spatial Relations.

The mathematical background for the relations described in this table are discussed in White (1981). Note that the relations of dimension and distance are only two of a potentially large number of geometric measures which might be used. The spatial relations defined in this table plus other non-spatial relations form the basis for the organization of cartographic information in Harmony. The unit of organization is a collection of like

entities, such as a group of point, line, or area features, along with a set of relations that are constant throught the collection. These collections are stored in the computer as files and each individual (record) in the collection has the same set of relations (fields) as every other individual.

Each field in a file has a code indicating the nature of the relation that is stored in the field. One of the fields is designated as the "primary" field and is presumed to contain the names or identifiers of the point, line, or area features that are contained in the file. Following are a list of the numerical codes used to indicate various relations:

Primary Field Types	Interior Topology
0 Points	42 Adjacent Chains
1 Lines	43 Adjacent Polygons
2 Areas	45 Connected Chains
Scale	46 Connected Polygons
11 Containers	47 Separated Nodes
12 Containees	48 Separated Chains
Order	49 Separated Polygons ("holes")
21 Precedessors	Geometry
22 Successors	50 (X,Y) Coordinate Locations
Boundary Topology	51 (X,Y) Minima
31 Nodes clockwise	52 (X,Y) Maxima
32 Chains clockwise	61 (X,Y) Centroids
33 Polygons clockwise	81 Perimeters/Lengths
34 Beginning Nodes	82 Areas
35 Ending Nodes	91 Distance from Point
36 Left Polygons	Non-Spatial Attributes
37 Right Polygons	101 Binary
	102 Nominal
	103 Ordinal
	104 Interval
	105 Ratio

Table 2. Types of Relations in Harmony.

When Harmony prints out the field descriptions of a file, the column labeled "Rel" indicates which relation of those above is represented by each field of the file. A special code of -1 is used to indicate the field which contains the number of coordinate pairs in a line or area boundary description (relation number 50). The Odyssey file management system (Morehouse 1978) used by Harmony allows for one variable length field as part of a record. A count field (code -1) always accompanies a variable length field.

Harmony is a sequential file processing program with three processing "modes". Each mode uses one or more input files and produces one or more output files or a set of output graphics commands. The three modes are sorting, copying, and drawing. Sorting must be explicitly invoked by the user in order to prepare files for further processing such as aggregation. One input file is sorted to one output file. Copying is the basic mode for file transformations and may be performed from one or two input files (two input files must be synchronized, i.e., sorted on the same feature) to any number of output files. Drawing is from any number of input files to a single file of graphics commands or directly to a graphics device. The command for the appropriate action,

```
sort;
copy;
draw;
```

is normally given after the relevant files have been opened and the file descriptions have been modified, or after symbolism for drawing has been specified. (See Dougenik (1978), Broekhuysen and Dutton (1983), for a discussion of the Odyssey command language generator used for this program.)

Input files are opened by specifying their internal Harmony file number and their external system file name,

```
open input:file filename:'name';
```

Output files are also identified by internal file numbers and external file names, but in addition have optional qualifiers of text, aggregated, or graphics. Text files are written in character format rather than in binary. A file which is opened as aggregated sets up the aggregation mechanism for the subsequent copy operation. The text and aggregation options may coexist. Mutually exclusive from both of these is the graphics option which prepares the file to receive graphics commands. The final qualifier for output files is the input file with which to pattern the output file's fields. Each output file is initially patterned after an existing input file such that it has the same number of fields, each with the same characteristics as the template input file. The statement opening output files is,

```
open output:file filename:'name'
                 text
                 aggregated
                 graphics
                 like template:file;
```

where the default is binary, non-aggregated, non-graphics, and with a template like input file number one.

The three commands for managing the fields of files are the construct, remove, and modify commands. To construct a new field for an output file the template field from one of the input files must be indicated. The new field is then constructed with the same characteristics as the template field. Any field of an output file (or an entire output file) may be eliminated from processing with the remove statement. The modify statement is one of the most important in Harmony. With it, fields of output files are set up according to the desired contents on output. Most of the functional capabilities of Harmony are in fact initiated by specifying how each output field is to be made. The qualifiers to the modify command are the field's label (which appears in file descriptions as a documentary assist), the data type (which may be integer, real, or character in codes of 1, 2, or 3), the word count (in computer words appropriate to the data type), the relation code (as described above), and, finally, the "formula" for the field. The syntax for these commands are:

```
construct output:(file,field) like template:(file,field);
remove    output:(file,field);
modify    output:(file,field) label:'label'
                              datatype:code
                              wordcount:count
                              relation:code
                              formula:'formula';
```

FORMULAS

The formula mechanism is one of the distinguishing features of Harmony and provides the program's flexibility and generality. Each field of each output file is built during the copy operation according to its formula. Formulas consist of variables, operators, and functions arranged in a conventional infix notation. The variables are fields of the input files. The operators are the arithmetical, relational, and logical operators found in most algebraic programming languages. The functions are some of the common mathematical functions (trig, log, exp) and plus some special ones. The membership function evaluates the presence or absence of its first argument in the set specified by its second argument. The ranking function returns the position into which its first argument would sort in the ordered table specified by its second argument. The conversion function returns the position in which its first argument is found in the conversion (recoding) table specified by its second argument. The transformation function specifies in its second argument one of twenty coordinate transformations (including ten map projections) to be applied to its first argument. The aggregation function specifies in its second argument one of six (last value, number of values, sum, mean, minimum, maximum) aggregation methods to be applied to its first argument. The

conditional function returns its second argument as a value if its first argument is true, otherwise it returns its third argument. The formula language supports any level of nesting of variables, operators, and functions as long as the total length of the formula does not exceed a predeclared maximum (288 characters).

The implementation of the formula mechanism consists of a preparation stage, before file processing, in which all formulas are compiled from their infix representation to a more compact and efficient postfix representation. As each record of each output file is prepared, each field's formula is evaluated in its postfix form to obtain the contents of the output field. The variables (the current contents of the designated input fields) are combined by operators and transformed by functions as specified in the formula. The program performs type conversion (all numerical values are processed as real numbers) but no type checking or word count checking.

The formula mechanism is also used to select records for an output file. Each output file may optionally have a formula which is evaluated for each record to determine whether to include the record in output. Selection formulas must evaluate to a logical value (true or false). In the drawing mode, Harmony also uses formulas to specify the color, pattern, and size of symbolism to represent each entity of each input file to be mapped. With the various operators and the ranking function and the conversion function, any type of value classification which does not require preprocessing all values (such as quantiles or nested means would, for example) can be performed. The relevant statements in Harmony are:

```
select  output:file by formula:'formula';
color   input:file  by formula:'formula';
pattern input:file  by formula:'formula';
size    input:file  by formula:'formula';
```

Many of the coordinate transformations require parameters which are specified in separate statements in Harmony. The membership sets, and ranking and conversion tables are also specified separately. These statements are:

```
make center:(xcenter,ycenter)
     radius:distance
     parameters:(one,two)
     resolution:distance
     angle:degrees;
make membership:number table:(mem1,mem2,...memN)
     ranking:number    table:(break1,break2,...,breakN+1)
     conversion:number table:(code1,code2,...,codeN);
```

Further statements are used to specify the graphics device, viewport, and

data window for maps, and to specify the fields to be used in sorting. The status of any field or file and the values of various parameters may be displayed; the codes for relations, aggregration types, devices, graphics patterns, and transformations, and the symbols for operators may be reviewed with help commands. Finally, the program supports a scripting or command file feature in which long sequences of Harmony instructions may be prepared with a text editor and subsequently read into the program either interactively or in a batch mode.

AGGREGATION AND DISAGGREGATION

Harmony has a general capability for aggregation and disaggregation. In the aggregation process, data from subsidiary geographic units are grouped together in some way and assigned to the parent unit. The simplest form of aggregation uses one input file sorted on the field containing parent identifiers. The output file normally consists of a primary field of parent units and one or more attribute fields constructed with formulas using aggregation functions. These formulas accumulate the data for one parent unit while that unit's subsidiaries are being processed in the input file, and then store the aggregate data in the output file when a new parent appears in the input. An additional input file may be used if, for example, attributes of the parent units are needed in the aggregation formulas or are to be transferred to the output file from an existing parent level file.

In the disaggregation process, data from a set of parent geographic units are distributed in some way to subsidiary units. For this process two input files are needed: a parent file with aggregate data, and a file of subsidiary units with attributes to be used in the disaggregation formulas (such as the subsidiary units' areas or boundary lengths, for example). Harmony creates the output files as subsidiary unit files with fields of disaggregated data calculated by formulas taking a parent attribute and spreading it to subsidiaries according to some ratio of subsidiary attribute to parent attribute, or, for nominal data, simply copying the parent attribute. As with its other features, Harmony allows any kind of disaggregation which can be expressed in its formula language, but requires the user to "formulate" the result.

FORMULA EXAMPLES

A good way to understand how formulas are used to accomplish cartographic data processing goals is to look at some examples. Many applications using coordinate locations begin with a map projection from spherical coordinates to planar. The first transformation upon a data set of points or lines normally has a formula like:

```
'TRAN (F(1, -1), 411)'
```

where this formula would be specified for the output field containing the coordinates. (When the coordinates are contained in a variable length field, as they would in a "chain" file of polygon boundaries, the field number is always designated as -1). The formula specifies that the transformation function indicated by the second argument (an Albers projection *411) is to be applied to the first argument, the variable length field (of coordinates) of the first file. Subsequent processing might calculate areas of polygons, lengths of lines, or, for points, distance from a prescribed point with similar formulas, changing the second argument of the transformation function to specify the appropriate transformation.

A class of operations often using long and nested formulas is the selection of individual geographic units based on one or more criteria. Simple selection can be made from membership tables by first assigning members to the table and then using a formula like:

```
'MEM (F(1, 1), 1)'
```

to chose only those input records for output processing whose first field value is a member of table number one. Such a formula can be arbitrarily complex including the logical operators of AND, OR, and NOT to refine the selection. A similar use of formulas is assigning suitability classes to geographic units based on a number of criteria. Nested conditional functions can create a quite complicated formula, for example:

```
'IF (F(1, 1) > 1, IF (F(1, 2) = F(1, 3), 2, 1), 0)'
```

where the meaning is "if (file 1, field 1) is greater than one then if (file 1, field 2) is equal to (file 1, field 3) then assign the value of 2, otherwise if condition one is true but two is false assign the value of 1, otherwise if condition one is false assign the value of 0." Any degree of arithmetical, relational, or logical complexity can be handled this way.

Mapping with equal interval value classifications is done with a formula like:

```
'(F(1, 1) DIV 10) + 1'
```

that subdivides the range of values in (file 1, field 1) by intervals of 10. Classification by arbitrary intervals can be done by the ranking function or by nested conditionals. Each input file has its own formulas for determining the color, pattern, and size of symbolism of its values.

SUPPORTING PROGRAMS

Two additional programs support the activities of Harmony. The Picture program (White 1983) provides a two dimensional figure drawing capability that allows points, lines, and areas to be drawn as symbols, in outline form, or to be filled with colors and patterns. Text annotation can be created in several fonts. Titles, keys, north arrows, and other map

elements are constructed with this program to accompany the symbolized geographic units drawn by Harmony. A program called Global converts data into the correct format for Harmony. Odyssey files can be read directly into Harmony.

CONCLUSION

The Harmony program uses a "higher level" language for cartographic data processing that provides for the specification of the results of cartographic operations in a mostly non-procedural manner. The success of this abstraction of cartographic processes is based on the validity, generality, elegance, and completeness of the underlying conceptual model of cartographic entities and their relationships.

REFERENCES

Berman, R. R., and Stonebraker, M. 1977. GEO-QUEL: A system for the manipulation and display of geographic data. Computer Graphics 11(2):186-91.

Broekhuysen, M. and Dutton, G. 1983. Conversations with Odyssey. Proceedings, Auto-Carto VI, ed. B. S. Wellar. Ottawa.

Burton, W. 1979. Logical and physical data types in geographic information systems. Geo-Processing 1(2):167-81.

Chrisman, N. 1979. A many dimensional projection of Odyssey. Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

Cox, N. J., Aldred, B. K., and Rhind, D. W. 1980. A relational data base system and a proposal for a geographic data type. Geo-Processing 1(3):217-29.

Dougenik, J. A. 1978. LINGUIST: A table driven language module for Odyssey. Proceedings of the First International Symposium on Topological Data Structures for Geographic Information Systems, ed. G. H. Dutton. Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

Frank, A. 1982. MAPQUERY: Data base query language for retrieval of geometric data and their graphical representation. Computer Graphics 16(3):199-207.

Morehouse, S. 1978. The Odyssey file system. Proceedings of the First

International Symposium on Topological Data Structures for Geographic Information Systems, ed. G. H. Dutton. Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

Roussopoulos, N., and Yeh, R. T. 1984. An adaptable methodology for database design. IEEE Computer 17(5):64-80.

Tsurutani, T., Kasahara, Y., and Naniwada, M. 1980. ATLAS: A geograhic database system. Computer Graphics 14(3):71-7.

White, D. 1983. A graphics system for instruction in computer graphics. Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

White, D. 1981. A taxonomy of space-time data relations. Presented at the Princeton Conference on Computer Graphics and Transportation Planning, Princeton, N. J.

White, D. 1979. Odyssey design structure. Harvard Library of Computer Graphics, 1979 Mapping Collection, II:207-15.

Short Biography
Denis White
January 1985

Denis White is a Research Associate in the Laboratory for Computer Graphics and Spatial Analysis, and a Lecturer in Landscape Architecture, at the Harvard University Graduate School of Design where he has been for ten years. His research and teaching interests are computer assisted cartography, the design and implementation of geographic information systems, and computer modeling for environmental design and planning. He has a BA in computer science from the University of Wisconsin, graduate credit in geography from the University of Oregon, and a MA in geography from Boston University.