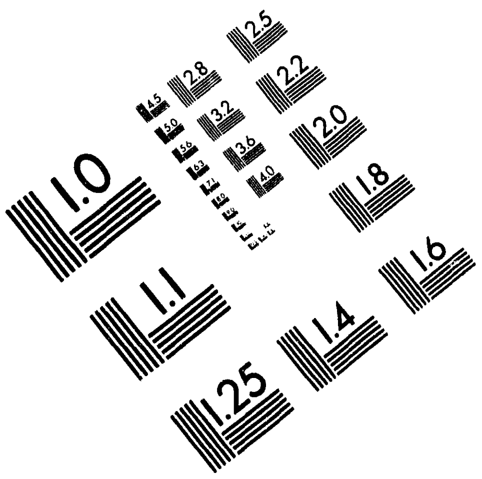
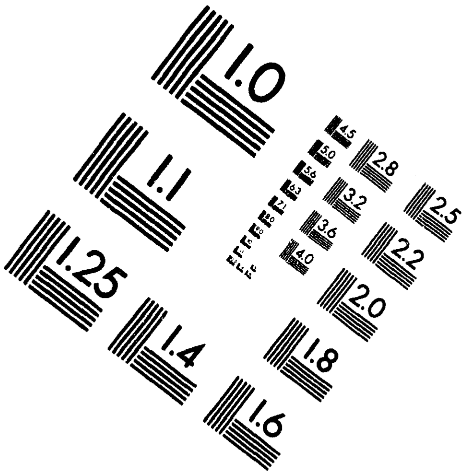
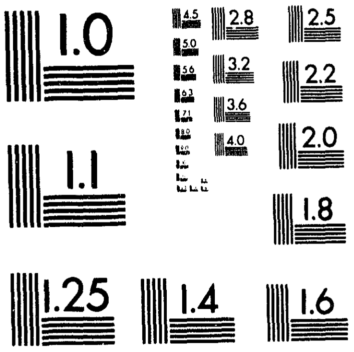
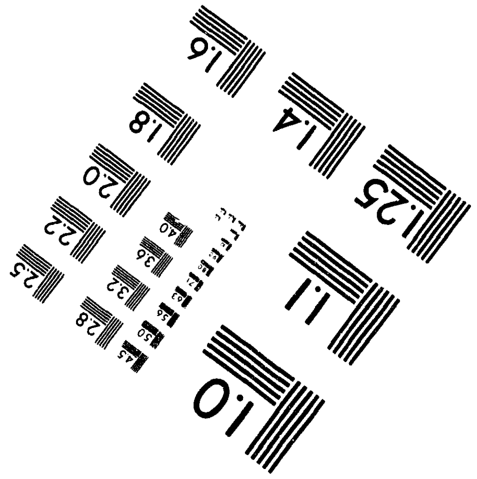
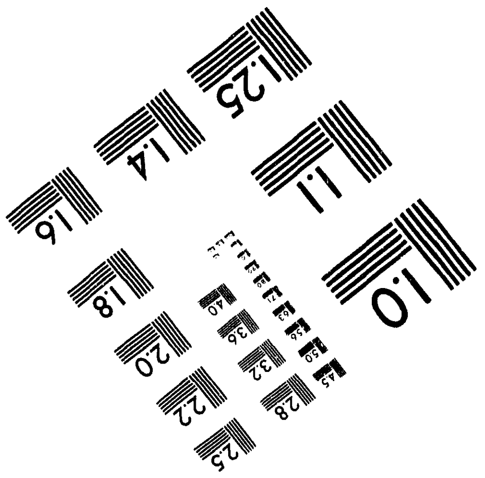



Centimeter
mm
Inches




1 of 1

Conf-940516---2

SAND94-0919C

John H. Ganter, GIS Specialist
Jonathan W. Cashwell, Task Leader
Transportation Systems Analysis (6641)
Sandia National Laboratories
POB 5800, MS 0718
Albuquerque, New Mexico USA 87185-0718
Telephone: (505) 844-1304
FAX: (505) 844-0244
Internet: jganter@ttd.sandia.gov

MAGENCO: A MAP GENERALIZATION CONTROLLER FOR ARC/INFO*

The Arc/Info GENERALIZE command implements the Douglas-Peucker algorithm, a well-regarded approach that preserves line 'character' while reducing the number of points according to a *tolerance* parameter supplied by the user. We have developed an Arc Macro Language (AML) interface called MAGENCO that allows the user to browse workspaces, select a coverage, extract a sample from this coverage, then apply various tolerances to the sample. The results are shown in multiple display windows that are arranged around the original sample for quick visual comparison. The user may then return to the whole coverage and apply the chosen tolerance. We analyze the ergonomics of line simplification, explain our design (which includes an animated demonstration of the Douglas-Peucker algorithm), and discuss key points of the MAGENCO implementation.

INTRODUCTION

One of the most common operations in cartographic generalization is line simplification. Line simplification is a form of filtering that matches the vertex density to display and processing requirements, resulting in appropriate aesthetic appearance, reduced storage needs, and faster display/analysis. To meet these needs, Arc/Info includes the GENERALIZE command in the ARC module (which permanently changes a coverage) and the WEEDDRAW command in the ARCPLOT module (which affects display only). Both use the Douglas-Peucker (D-P) algorithm (Douglas and Peucker 1973).

* This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

D-P seems to be a good choice. Even though many line simplification algorithms have been developed (McMaster and Shea 1993), a combination of fidelity to the original line and reasonable processing speed have made D-P arguably the most popular line simplification algorithm.

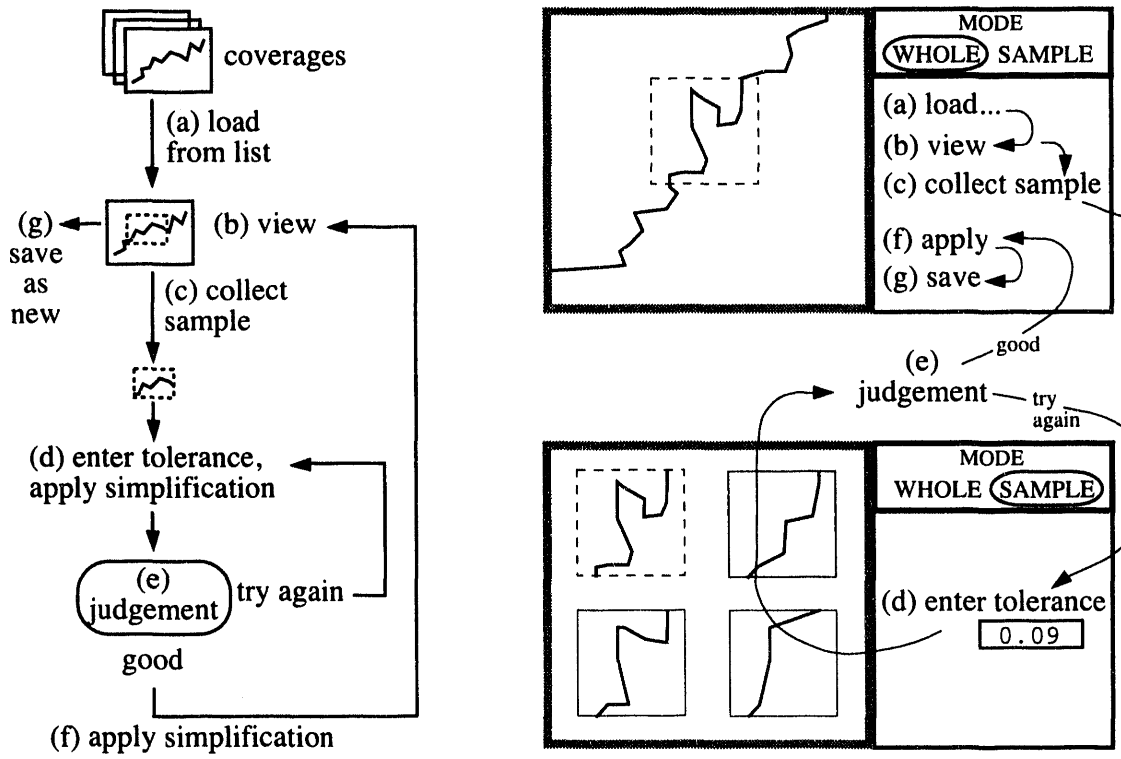


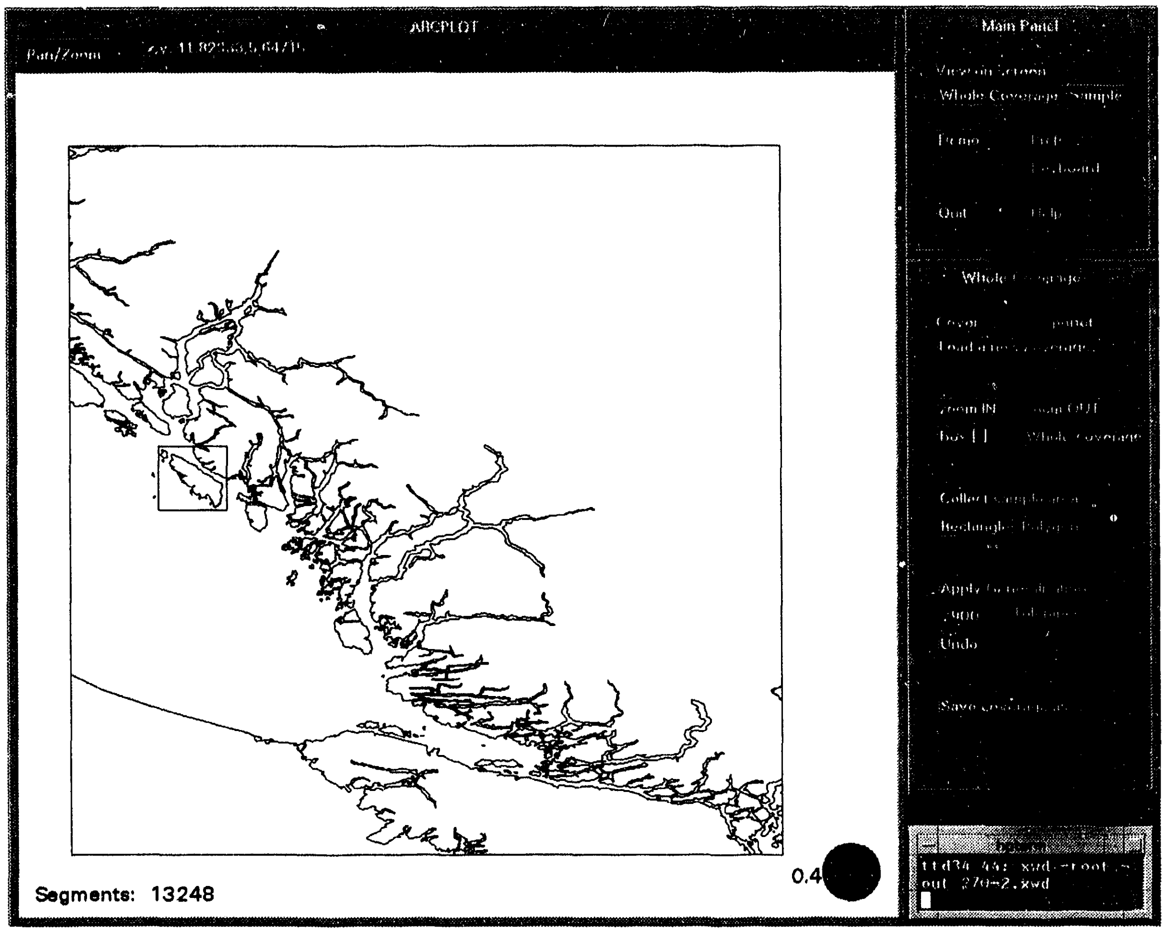
Figure 1: Task scenario shown as a flowchart (left) and translated into a design sketch of the GUI (right) for MAGENCO

The performance of D-P results from a holistic treatment of the cartographic line. The algorithm moves a 'floater' along the vertices, measuring the maximum orthogonal distance to each intervening point. The algorithm then decides whether to keep points by determining if they lie within a 'corridor' whose width is specified by a *tolerance* parameter. It is thought that the 'optimal' tolerance for a particular line coverage (or section thereof) can only be determined experimentally. This requires an iterative run-and-examine technique that WEEDDRAW and GENERALIZE do not directly facilitate.

By examining a typical work flow of generalization tasks, we designed an interface that minimizes user time and motion in determining the optimal tolerance and applying it to a complete coverage.

ANALYSIS OF USER TASKS AND DESIGN OF MAGENCO

Based on our experience with generalization and Arc/Info use, we developed a scenario for an imaginary user carrying out a line simplification task. It appeared that such a user would want to (Figure 1, left): (a) select and view any line coverage for which they had



```
ttd34 44: xwd -root >
out 270-2.xwd
```

Figure 2: Whole Coverage mode of MAGENCO (PONET Digital Chart of the World coverage used with permission of ESRI, Inc.)

file permissions; (b) zoom in and out of the coverage for detailed viewing; (c) select a sample area; (d) apply different tolerances to the sample; (e) reach a decision on the preferred tolerance; (f) apply the tolerance and simplification to the whole coverage; and (g) save the results as a new coverage.

We then attempted to translate this scenario into a graphical user interface (GUI) (Figure 1 right). The dominant feature of our design is a distinct split between (c) and (d). The GUI becomes 'bi-modal,' as it were, with two exclusive states: the user is forced to be in either 'Whole Coverage' or 'Sample' mode.

The Sample mode was partly inspired by the Map Generalization System (MGS) of Chang and McMaster (1993). MGS displays four views of a map and allows users to choose algorithms (D-P, Reumann, Lang, and Vectgen) and appropriate parameters to apply to each view. MGS responds with simplified lines and statistics such as angular and areal error. MGS simplifies in real-time, so that the user can watch what is happening. This is a distinct advantage of doing the simplifications in memory and using a modern, structure-oriented, display system (in this case the SunPHIGS package). Unfortunately, since WEEDDRAW and GENERALIZE are batch-oriented this is not possible in the present MAGENCO architecture.

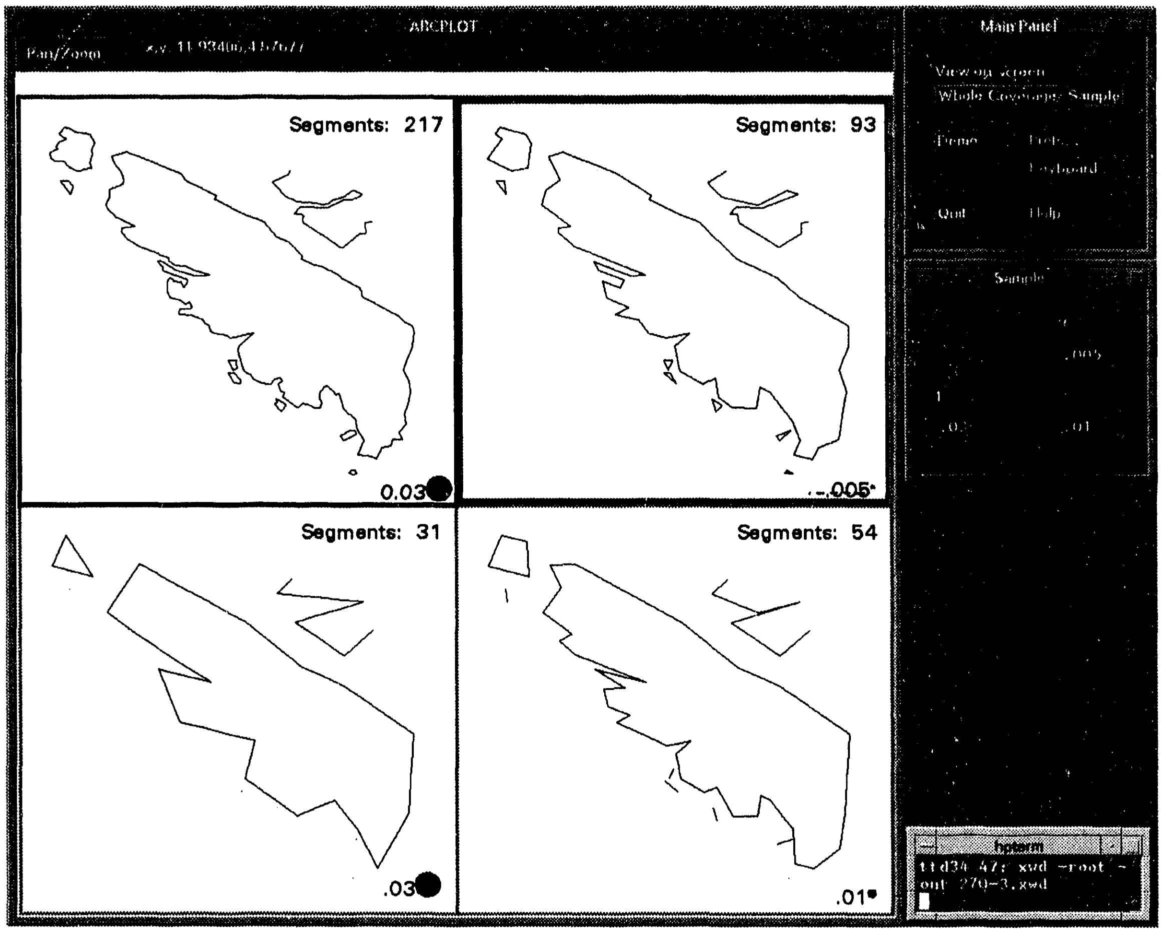


Figure 3: Sample mode, showing original sample and segment count (upper left). The three tolerances are applied with the panel at right, and the resulting displays and numbers of segments appear around the original. Note that the original sample has a blue scale ball, while the simplified displays have red scale balls that show the applied tolerance.

AN EXAMPLE SESSION

Before discussing the implementation of MAGENCO, we will proceed through a short example session to show the overall appearance and main features.

■ Figure 2: The *Whole Coverage mode*. The *Main Panel* (top right) is present in both modes, allowing mode switches and access to the D-P Demo, setting of user preferences, keyboard commands, the help facility, and Quit for exiting the program. Below, the *Whole Coverage* panel has a vertical sequence that corresponds closely to the (a-c) and (f-g) operations in the preceding discussion of design.

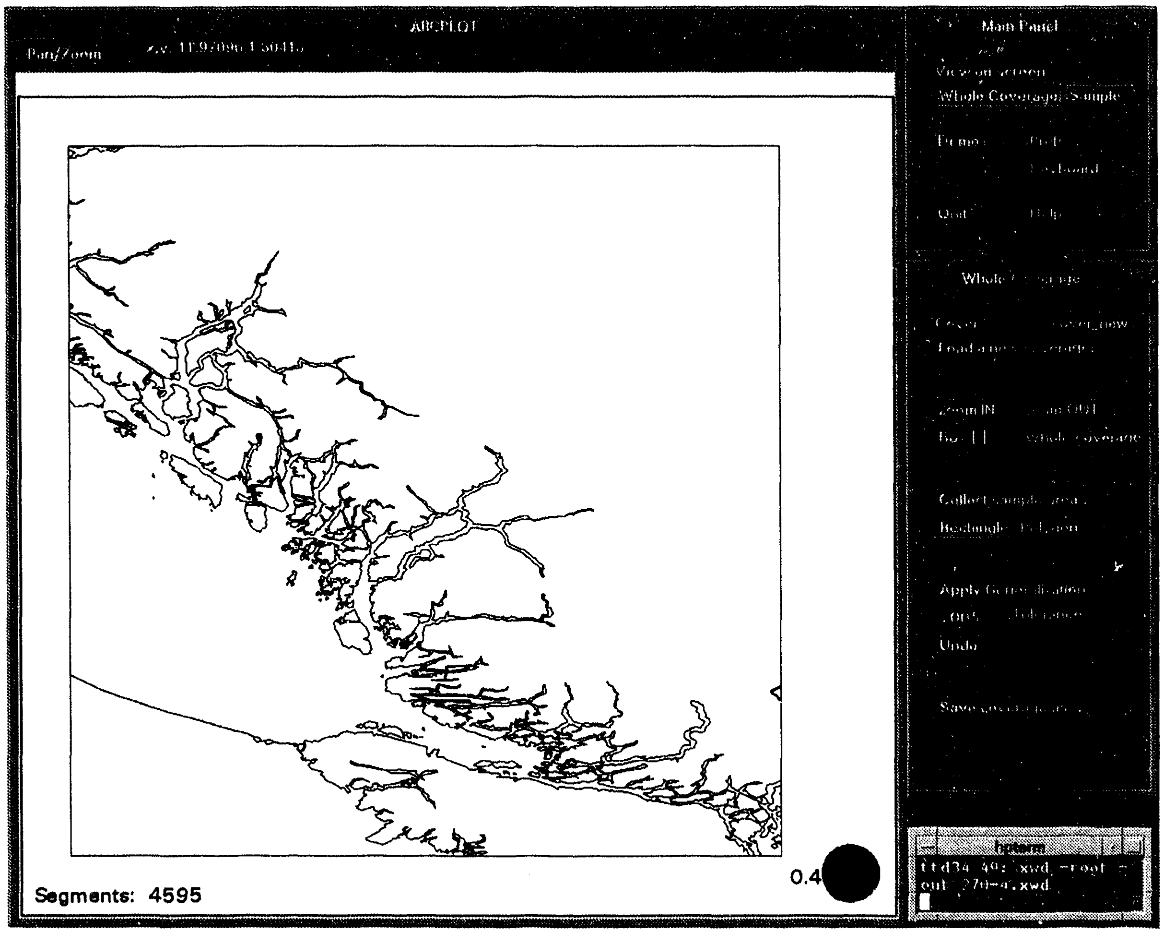


Figure 4: The user has returned to Whole Coverage mode and pushed the Apply Generalization button. The coverage has been generalized, although it is difficult to see at this scale. Note the reduced number of segments.

At the bottom left, the number of segments in the coverage is shown. At the right, a blue 'scale ball' shows 0.4 degrees of latitude/longitude (the units for this coverage). The scale maintains approximately the same size as the user zooms in and out of the coverage, with the number changing to report the size of the ball in map units. The advantage of a circular scale is that all directions are shown. Graticule lines that can be turned on and off are a possible future enhancement.

The user has collected a rectangular sample (island region at left) using the *Collect sample area...* button, and now depresses the *Sample* radio button on the *Main Panel...*

■ Figure 3: The *Sample mode*. The sample appears with blue border and scale ball at the upper-left, with three simplification displays surrounding it. Each display has the number of segments, and a scale ball (this one red) that shows the size of the tolerance. The tolerances are set on the panel at right, with the depressed button indicating the simplification display that is being processed (number 3 at the moment). This particular display also has a red highlight border drawn around it.

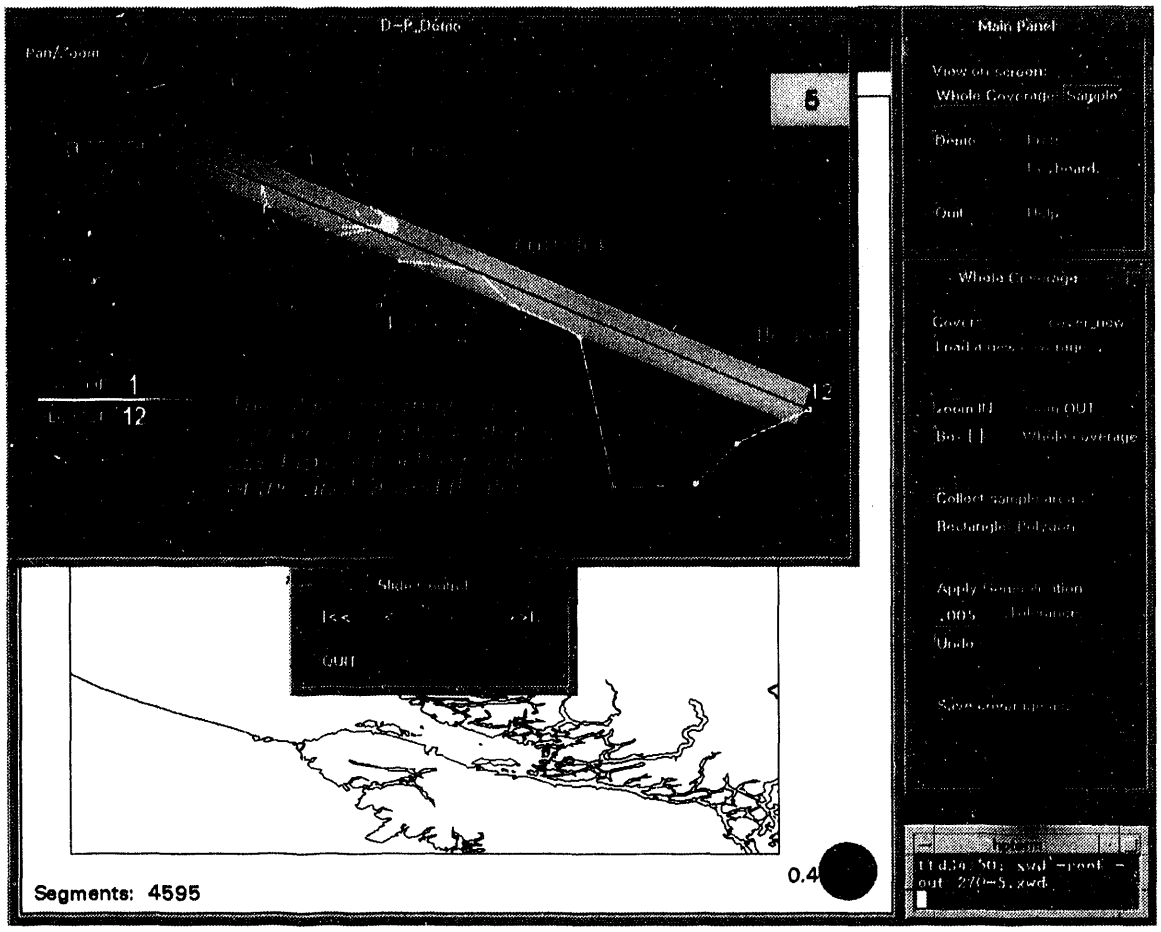


Figure 5: Slide 5 in the D-P Demo. The main mechanisms of the D-P algorithm are summarized in this slide; subsequent slides show the mechanisms in action as the number of points is reduced.

■ Figure 4: Once they have chosen a tolerance, the user switches modes back to *Whole Coverage*. The last tolerance that was selected in Sample mode (.005) now conveniently appears in the *Tolerance:* entry field under *Apply Generalization*. The user has applied the tolerance to the whole coverage, reducing the number of segments to 4595 (with minimal change in the appearance at this scale). The user can press *Undo* if they don't like the results or *Save coverage as...* if they do. Pop-up menus then allow entry of coverage name and convey overwrite warnings. At any point the user can restart the whole process by zooming and/or taking another sample.

■ Figure 5: In showing MAGENCO to those unfamiliar with computer-assisted cartography, we were at a loss to explain the workings of the D-P algorithm. The display of the tolerance scale ball seemed to precipitate this questioning, since its relationship to the simplification is not immediately obvious. We decided to develop a short 'slide show' that would explain, more-or-less, how the algorithm works. Slide 5 is shown, along with the 'VCR'-like control that advances the slides.

IMPLEMENTATION

MAGENCO is composed of about 40 modules and 1700 lines of code, for roughly 45 lines of code/module. The number of modules could be reduced somewhat by using the ROUTINE-oriented style of ArcTools programs (ESRI 1993). Modules communicate by some arguments and many global variables. Globals are used heavily to keep track of the many coordinates used by MAPLIMITS in drawing the sample displays, borders, statistics, and scale balls. They also provide the Boolean flags that are used to arbitrate between the control panels and maintain the statuses of sample displays, button positions, and the like.

A map of the modules (Figure 6) can be divided into three main regions:

(1) At top-center is the ever-present Main Panel (panel1.menu), with some mechanisms just below for switching modes. This involves erasing various areas with PATCHES, and re-drawing within MAPLIMITs;

(2) At the right side are modules to manage the sample displays and apply the chosen tolerances;

(3) At the left side are various view and sample-taking controllers, and nuisance routines for loading, saving and overwriting files. Furthest out on these branches are modules for drawing the whole coverage, the scale ball, and restoring the sample area box after the lines from RESELECT have gone away.

While somewhat complex to the casual reader, we have found such 'module maps' invaluable for communication and planning in development – especially when returning to code after a few months have elapsed.

D-P Demo 'slide show'

Our first thought was to write D-P Demo as an AML, so that the user could input tolerances and watch the animated results. This quickly conflicted with our desire to use multiple layers, highlights, etc. as explanatory aids. While cartographically broad, the ARCPLOT IGL graphics routines lack the depth and sophistication to make any sort of animation or layered drawing possible (or at least pleasant).

Instead we used Corel System's CorelDRAW 3.0 to create bitmapped graphics that are displayed through ARCPLOT IMAGEVIEW. Initially we tried using Microsoft PowerPoint to lay out the sequence of frames, but PowerPoint at 3.0 will not rotate rectangles (the D-P corridor) to arbitrary angles. After using the Windows Clipboard to transfer rectangles over, we settled on just using CorelDRAW. CorelDRAW exports compressed TIFF files in a flavor that Arc/Info likes, so this works fine. Some testing revealed that, not surprisingly, 75 DPI / 16 color TIFFs displayed fastest. The 24 frames of D-P Demo took a fair amount of work to create, but it was an interesting experience and future projects will be much easier. D-P Demo is also available as a stand-alone Windows application using the CorelSHOW runtime-player (shareware).

Storing and restoring user preferences

Users like to set preferences; in the case of MAGENCO this includes such things as line colors and thicknesses, location of scale balls and statistics, etc. There are a number of approaches to applying, preserving, and loading preferences, including complicated schemes like linked INFO tables that store metadata on variables and the values for these variables at various times (Ganter 1993). For this application, a simpler approach is to write and transform WATCH files (Figure 7).

```
/* PERSISTENT PREFERENCES EXAMPLE
/*   echo must be off
/*   messages must be on

/* set some preference globals
  &s .user$col1 blue
  &s .user$col2 red
  &s .user$wid 20

/* capture the preferences for Bill
  &WATCH user.bill
    &listg .user$*              /* list certain globals
  &WATCH &OFF

/* the preferences go away
  &dv .user$*

/* Bill wants his preferences back
  &s search [UNQUOTE 'Global:']   /* &lg adds this prefix
  &s replace [UNQUOTE '&set']     /* replace above with &set

&s pref_file user.bill
  &s read_file [OPEN %pref_file% status -READ]
  &s line [READ %read_file% status]

  &do &while %status% = 0         /* loop through the lines

    &ty The line is:    %line%                     /* raw line
    &s command [SUBST %line% %search% %replace%] /* fix the line
    &ty The command is: %command%                 /* a command results
    [UNQUOTE %command%]                           /* submit the command

    &s line [READ %read_file% status]
  &end

 &s null [CLOSE %read_file%]     /* always close files
 &lg                             /* the globals are back
```

Figure 6: Listing for a method to save certain global variable values to a file, then restore the values later. This technique is useful for saving user preferences, returning programs to their starting or ending points, ‘undo’ buttons, etc.

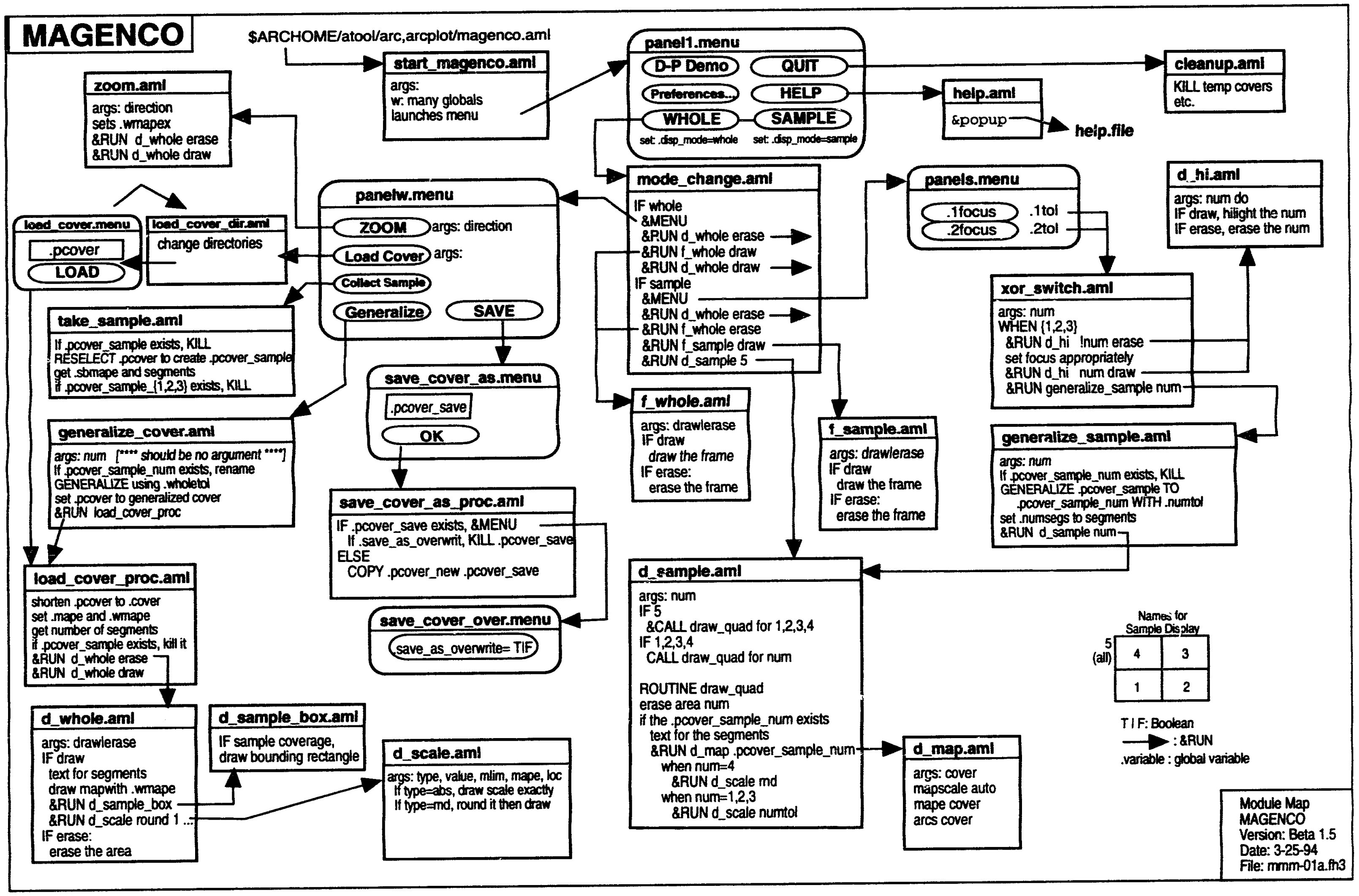


Figure 7: Module map for MAGENCO

OBSERVATIONS ON GUI DESIGN

The MAGENCO user interface is very simple, but its design is 'obvious' only in hindsight. Why is this so? Returning to Figure 1, the translation of a scenario into a GUI sketch, it is interesting to note several loose connections between the scenario and the design: (1) the whole display portion (left side) of the GUI is left unstated in the scenario narrative (i.e. it has no letters in parentheses); (2) most of the action centers around the action-oriented widgets (buttons, input fields, etc.), and; (3) the modal split, being an artifact of the design, is not reflected in the narrative at all.

The challenge in this task seems to come from the loose connection between natural-language descriptions and the visual 'syntax' of GUIs. Language narratives of a user task or group of tasks are often silent on matters such as state (e.g., what is the user *seeing* while they are *doing*?). Also, narratives do not specify how verbs (select, view, zoom) are to be accommodated through buttons and other widgets (e.g., *zoom* may be carried out with one button, while *save* is much more conditional and requires a sequence of menus, caveats, and decisions). The writer and reader of a narrative carry along with minds subconsciously filling in (or forestalling) the details of what is being described, but the GUI designer must create a tangible interface that matches (at least roughly) the underlying user conceptions while filling in the voids.

GUI design is an absorbing task that always seems remarkably easy and obvious in retrospect (if it is judged a success by users). The main challenge seems to be drawing back, recognizing, and examining spontaneous design decisions that are not optimal in the long term, as Essinger and Lanter (1992) point out:

> When you design an interface you are bound to make a series of assumptions about the user and what they will find usable. Being aware of these assumptions and being able to *test* them with real users is a very important feature of user-centered design. These is nothing wrong with making assumptions in your design. What becomes a source of difficulty is when you have the assumptions but don't *know* you have them, because in that case it is much harder to consciously test the design assumptions with the user community to see if they hold true

One example of this problem has already been discovered in MAGENCO. The first implementation of the *Sample* panel (see Figure 3) had a vertical row of controls because it was the simplest layout. Despite test users being assured that anything could be changed if they did not like it, none complained. After a break of a few weeks, the designer/programmer returned and had difficulty figuring out the relationship between the row of controls and the grid of displays – precisely the sort of *unnatural mapping* that we find on many stovetops! (Don Norman in Essinger and Lanter 1992).

CONCLUDING REMARKS

A MAGENCO improvement that comes to mind is the use of WEEDDRAW to speed display. The present method is to RESELECT and WRITESELECT in ARCPLOT, then run an &DATA block in ARC containing commands to create a new coverage and apply GENERALIZE. The use of WEEDDRAW seems obvious but how to obtain the statistics (number of segments after simplification) is not. At present we use the DSC$SEGMENTS system variable to obtain this data for a coverage. It may be desirable to give the user an option: fast simplification with only visual feedback or slower simplification with quantitative feedback.

We would also like to add support for the GISExpress (Kinn, Ofenstein and Gizzi 1993) ARCPLOT accelerator (if licensed on the user's system), but the present version ignores MAPLIMITS so it would be usable only in the Whole Coverage mode. GISExpress is so fast, however, that it might suffice to 'flicker' the original and simplified versions of a map, thus allowing the user another form of visual comparison.

MAGENCO seems intuitive and useful to preliminary users. We hope that further feedback will help to validate and improve the design decisions that we made in translating a simple scenario and narrative into a usable GUI.

AVAILABILITY

The beta version of MAGENCO is available at no charge under a test and evaluation license (for commercial users) or research license (non-commercial users) granted by Sandia Corporation as directed by the U. S. Department of Energy (DOE). Contact John Ganter for more information. We welcome any feedback that will help us to refine MAGENCO.

ACKNOWLEDGMENTS

David Dibiase provided a number of insights during the requirements and design stage. We thank Brad Godfrey and Hillary Minich for their suggestions on the manuscript.

DISCLAIMER

Portions of this work were carried cut in the course of the author's professional assignments. The opinions stated are those of the authors, and do not necessarily reflect those of Sandia National Laboratories, Sandia Corporation, or the U. S. Department of Energy (DOE). Trademarks are owned by their holders. Mention of commercial products does not constitute endorsement by any parties.

REFERENCES

Chang, Harry, and Robert B. McMaster. "Interface Design and Knowledge Acquisition for Cartographic Generalization." *Auto-Carto 11* (1993): 187-196.

Douglas, David H., and Thomas K. Peucker [now Poiker]. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Character." *The Canadian Cartographer* 10:2 (1973): 112-123.

Environmental Systems Research Institute, Inc. *ArcTools*. Redlands: ESRI,1993.

Essinger, Rupert and David Lanter. "User-Centered Software Design in GIS: Designing an Icon-based Flowchart that Reveals the Structure of Arc/Info Data Graphically." *Proceedings of the 12th Annual ESRI User Conference* III (1992): 245-256.

Ganter, John H. "Metadata Management in an Environmental GIS for Multidisciplinary Users." *Proceedings of GIS/LIS 1993* 1 (1993): 233-245.

Kinn, Gerald J., W. Thomas Ofenstein, and Sandro Gizzi. "Making Arc/Info Draw Faster: A New Approach." *Proceedings of the 13th Annual ESRI User Conference* I (1993): 497-502.

McMaster, Robert B., and K. Stuart Shea. *Generalization in Digital Cartography*. Washington, D.C.: Association of American Geographers, 1992.

* * *