THE PHILOSOPHY AND REQUIREMENTS OF COMPUTER-AIDED GRAPHIC DESIGN IN CARTOGRAPHY

Rupert Essinger
Environmental Systems Research Institute
380 New York Street
Redlands, California, 92373

Rupert Essinger studied at Oxford and Glasgow Universities and the State University of New York at Buffalo. He currently works in system support at ESRI in Southern California.

Abstract: This paper investigates the philosophy of computer-aided graphic design in cartography and details the software requirements for a CAD workstation for map design. It is argued that the most important function of this software is to enable the cartographer to experiment fully with the design of symbols while they are displayed on a map. Only in this way can map symbols be designed in their true visual context and be judged graphically against the other symbols that surround them.

GRAPHIC DESIGN

Automating the graphic design process has received relatively little attention in computer cartography. Several authors have identified the benefits of computer-aided design (CAD) tools for the cartographer (Moellering 1977; Bickmore 1980; Joyner 1982; Monmonier 1982, ch.8; Marble & Calkins 1985, draft). However, there has apparently been no attempt to detail how computer-aided cartographic design software might function.

Graphic design is an artistic, expressionistic and experimental activity. The nature of the creative grapnic design process is that the designer is, in effect, the first consumer of the design product. The designer predicts the visual needs of the map's audience and continually improves the graphic design in response to his own perception of it. In this way the graphic designer carries out intuitively the perceptual tasks of the future audience. Compared with the map's final audience, the cartographer probably has sophisticated map reading abilities, but the skill of the map designer is precisely that of being able to control the interplay of line, color, pattern and detail to solve problems of perception and visibility, and so create graphics that will facilitate communication at all levels.

THE NEED FOR COMPUTER-AIDED CARTOGRAPHIC DESIGN

The underlying philosophy of CAD is that by liberating designers from distracting and tedious tasks, such as hand drafting, they can devote more time to the truly imaginative processes of creativity. This should lead to better design and greater efficiency (Negroponte 1975).

In traditional cartography the graphic designer's ability to experiment with design, to devise and review different approaches to the line, color and pattern, is limited both by the available technology and the typical production environment. Lawrence Fahey (1972) says "the practice of map design demands that the designer always keeps traditional map conventions in mind," while trying "to produce better maps by developing and refining new ideas" (p.132). He continues, "experimentation on different designs is also helpful, but unfortunately most cartographers find little time to spend on such pursuits" (p.135).

Real graphic experimentation involving all the symbols on a map would usually necessitate impossible amounts of redrafting. Of course there is no limit to the amount of prior pencil sketching and rough working out that can take place before the scribing begins. However these rough drafts cannot replicate the fine detail and exact color of the map symbols that will be used in thê final product, and are probably most useful for planning sheet layout. It is usually only when we can review the actual proof of our map that we can assess the success of our design decisions, especially where finely detailed symbols are involved, but there are limits to the design changes that can be made at the color proof stage. Indeed, in many cartographic environments the proof is meant mainly for error-checking, with time and resources militating against elaborate reworking. In order to experiment fully with symbol size, pattern and color for any map, we would have to produce an inordinate number of proofs.

Instead, the cartographer tends to rely on in-house graphic styles, design experience with similar maps, and visualization from the paper compilation. Visualization, like experimentation, is an inherent part of the design activity, but it does present problems in map-making, which depends on the establishment of visual hierarchies among a mass of medium and fine detail. Because they have enormous variation and complexity, these fine patterns tend to be hard to visualize.

Moellering (1977) says that compared to the manual map production cycle, interactive computer graphic methods provide a much richer creative environment, because the results of design decisions can be displayed rapidly on a CRT. Experimentation in design will be encouraged by the ability to view the impact of changes in color and pattern on the whole map graphic. In this way individual map symbols can be designed in their true visual context and judged graphically against the other symbols that surround them. Such CAD software would provide cartographic designers with an unprecendentedly powerful set of creative tools.

THE REQUIREMENTS OF COMPUTER-ASSISTED DESIGN IN CARTOGRAPHY

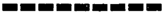
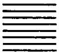
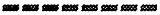
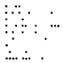
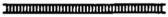
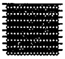
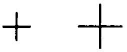
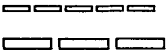
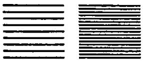
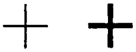
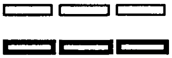
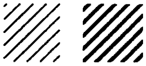
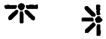
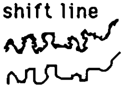
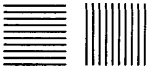
Two assumptions have been made about the geographic database used in the proposed CAD software. Firstly, it is assumed that geographic data are stored and retrieved separately from their symbolic representations as graphics. Compilation involves pulling geometric spatial data out of existing databases, data that as yet has no map symbolization. Secondly, geographic features in the database can be divided into a number of sets according to their descriptive attributes. Features within one set can be displayed on the map with the same symbology. The basic design function is, then, the assignment of symbols to sets of features through a process of interactive choice and review. All the final design choices made during the session can be stored in a design specification file. This file can be used to create a series of maps by linking it to different parts of the database, and so can be seen as a filter through which geometric data is sent for graphic display. Symbol designs are not stored with in the database itself because the cartographer wants to be free to assign different symbols to features as different map designs demand, and to be able to experiment with a variety of symbols during the interactive session.

It is likely that some of the interactive design work will be carried out on a map displayed at a scale larger than that of the intended output product. This is because the resolution of the color CRT may make it difficult to see the results of fine design work at the same scale as that of the intended final reproduction, especially when the map features many convoluted lines. Similarly, many maps created at this workstation will have to be designed in patches because it may not be possible to display the whole map on the relatively small CRT screen at a scale that is useful to the designer. As the design process entails assigning symbols to sets of features, it is possible that we could design an entire map just by viewing the results of our choices on one patch from that map. The map design could then be reviewed by displaying it patch by patch on the screen so that the effect of symbol selection can be seen on areas of the map that were not viewed while the choices were being made. Furthermore, intelligence could be engineered into the system so that the computer alerts the designer to patches on the map where features are unusually bunched together and present special problems to the designer.

Table 1 shows the required interactive graphic functions for designing map symbols. The first function in Table 1 enables the cartographer to devise basic patterns for symbols and enter these directly into the system by drawing either on a tablet or on a reserved part of the screen. These customized symbols will complement any basic start-up palette of symbols available to the user whenever a session is begun. Once the customized patterns have been entered, it should be possible to permanently save them so they become part of the start-up symbol palette.

Any symbol selected from the start-up palettes or defined by the user can be given further design changes by using the graphic manipulation functions in Table 1. For example, if we need to create an area symbol that consists of a black dotted boundary line enclosing an area filled with yellow and green stripes we might first enter the dotted line pattern by drawing a small

Table 1 Functions for designing cartographic symbols

FUNCTION	POINTS	LINES	AREA FILLS
Design and store in symbol library:	Point symbol	Line pattern 	Area fill pattern 
Select line color of:	Point symbol outline	Line pattern 	Area fill pattern 
Select fill color of:	Area inside point symbol	Area between double line 	Area inside existing fill pattern 
Enlarge or reduce:	Point symbol 	Line pattern 	Area fill pattern 
Thicken or thin linework in:	Point symbol 	Line pattern 	Area fill pattern 
Symbol location manipulations	Rotate and shift point symbol 	Simplify and shift line 	Rotate area fill pattern 
Terrain treatments	Spot heights	Contouring	Hillshading and hypsometric layering
Thematic treatments, eg, devise class intervals for:	Proportional symbols	Proportional symbols	Choropleth bands

portion of it in a special place on the screen. When we then use the line thickening function, the lines representing the boundaries of the areas that are to receive this symbol will slowly grow wider on the screen until we signal happiness with their weight. They will also appear in the color we have selected. The yellow stripes can be defined by devising an area fill pattern of yellow parallel lines. Their spacing is defined by the enlarging function, their line weight is chosen by the thickening function we have already seen and the pattern can be rotated too. The area is filled with green color. This will be overwritten by the yellow stripe pattern thus producing alternate green stripes too. Symbol design is thus seen as a process of continual refinement as we develop symbols and review their visual performance on the map display.

FEATURE-BASED DRAWING

We have seen that computer-aided cartographic design software should let the cartographer experiment fully with the design of symbols while they are displayed on a map on the screen. The most basic function, then, of this software, is to enable the map designer to draw map features on the screen display with certain symbols and then redraw any of them any number of times with different symbols. In this way parts of the map image can be redesigned if any of the choices of symbols prove graphically unsatisfactory when reviewed on the screen. The cartographer should be able to select any one of the various features on the map display and redraw that feature with a different symbol design at any time during the session.

This level of freedom is essential if the designer is to have full interactive control over the detailed visual hierarchies characteristic of cartographic products. For example, one of the first features we draw on the map display may be a city, which will be drawn using a particular area-fill symbol. We may then proceed to draw various line features on the map, like rivers and roads, that go through this city, only to find that the initial choice of area-fill symbol for the city provides a poor visual background for the line features. Assuming that we do not wish to change the design of the line symbols, we must be able to redraw the city with a new area-fill symbol to improve the figure-ground relationship. Furthermore, when we redraw the city symbol, the system should preserve the existing line symbols on the screen. Redrawing a background feature should not cause existing foreground features to be overwritten and thereby erased.

In this way the cartographer can experiment with a number of different area-fill symbol designs for the city without disturbing the line symbols that have already been designed and drawn on the map. All that will happen on the screen is that the urban area symbol will be altered where it is not masked out by the roads and rivers. We can call this system of drawing a feature-based system, because the cartographer manipulates the image by assigning symbols to features and because the display is updated in a way which preserves the existing features on the map.

Note that when this system redraws a symbol X on the screen it not only preserves any symbols that have been placed over that symbol X, it also preserves any symbols that symbol X itself overwrote. For example, we might have used a wide line symbol to show roads going through the city but now decide to redraw these roads with a narrower symbol. In this case once the roads have been redrawn on the map with the new, narrow line symbol, the system should automatically redisplay the city area-fill symbol in those places along the edge of the now narrower road where the city area-fill symbol had previously been overwritten by the wide road symbol. Similarly, the user must be provided with an erase function so that any feature can be completely removed from the map display. Erasing a feature X from the map should not erase features that have been already been drawn over feature X, nor should it erase any parts of features which were themselves hidden when feature X was drawn. Erasing feature X will automatically make any parts of features overwritten by X visible again on the map.

In the feature-based drawing system, when we draw a feature on the map for the first time it would overwrite (but not irretrievably erase) any existing features that coincide with in any locations on the map. This graphic protocol seems to makes good design sense because it encourages the cartographer to design the area symbols that form the background before designing the fine planimetry and text found in the foreground. There will then be an existing map background with color and pattern against which the symbols for the foreground planimetry and text can be designed, instead of having to do this delicate design work earlier in the session when those foreground symbols would hang against a void with no way of gauging their eventual needs of contrast and visibility. And of course, at any stage we can change any of the existing symbol designs, whether foreground or background.

Feature-based drawing poses some implementation problems. In most computer graphics systems drawing something on the screen overwrites (and therefore erases) anything that is already displayed in the same position on the screen. This situation arises because the simplest way of driving a raster display terminal is to keep the screen frame buffer updated with just one set (the latest set) of pixel values. For our purposes this situation is not satisfactory because, in the example already given, redrawing the urban areas with the new symbol design in order to provide a better background for the line features would automatically overwrite the existing line features, and we would need to draw them all again. Even when drawing a feature on the map for the first time, in which case the symbol overwrites any parts of existing symbols that coincide it, this overwriting should not be permanent. It must be possible for us to erase that new feature, which will then cause those parts of existing symbols that it overwrote to be redisplayed.

We can represent the requirements of this system conceptually as a series of layers, somewhat similar to a set of transparent mylar overlays as might be found in manual cartography. Each layer represents a discrete set of features on the map. When a new feature is drawn on the map display on the screen, that layer becomes the top-most in the series, it is unobscured on the screen yet it obscures all the layers beneath wherever they share map locations with it. When one of the layers is redrawn with a new symbol the layers above and beneath that layer are preserved. In fact, the key to implementing the feature-based drawing system of drawing is to replicate this layer structure with bit-planes in the screen frame buffer (that part of addressable memory used for driving the display). Each set of features drawn on the screen with the same symbol can be represented on a separate bit plane. Each bit-plane stores the pixel locations needed to display the set of features by giving those locations the bit-value 1. Whenever a feature is added to the map, a new bit-plane is set-up and stored, as it were, on top of the existing series of bit-planes. Whenever a feature is erased or redrawn, the bit-plane for the feature is accessed and the bit values are changed to represent the new screen locations needed for the feature (if the feature is erased, all the locations in the bit-plane are given the value 0). Updating the display is then a matter of reading the layers of bit-planes. For any pixel location on the screen, the upper-most (most recent) bit-value of 1 in the bit-plane series, indicates that that pixel must be used to display the feature represented by that bit's bit-plane. The color of that pixel can then be determined by referencing a separate color look-up table that matches features to symbol colors.

The precise nature of the visual hierarchy on a map is determined by the geographic data being mapped. For example we might have roads overwriting rivers wherever they cross because most roads do go over rivers, whereas the interrelationship between roads and railway lines is rather less predictable. Handling roads and railway lines within our feature-based drawing system, which draws by overwriting unless the feature has already been drawn, may be difficult. Where the road crosses over the railway, the railway symbol will have to be overwritten, and where the railway crosses over the road the road symbol will have to be overwritten, and in whichever order these sets of lines are drawn on the map, these crossing interrelationships will not be accurately portrayed. The solution to this will depend on the level of detail in the geographic database. In many databases crossing points between lines in two networks will be entities themselves and will be represented in the data. We could then, for example, divide our road and railway features into two sets; those that are crossed over and those that cross over, and then access and draw the former set before the latter set.

Feature-based drawing enables the designer to experiment with the symbols for one set of features without disturbing the other symbols on the map. Handling drawing in feature-based layers as described here enables us to experiment fully and interactively with the design of symbols at all levels in the visual hierarchy to an extent that is not possible with manual cartographic techniques.

REFERENCES

Bickmore, D.P. 1980. Future research and development in computer assisted cartography. In The Computer in Contemporary Cartography, ed. D.R.F. Taylor, pp. 235-249. John Wiley and Sons.

Calkins, H.W., and Marble, D.F. 1985. Draft. The structure of an an automated cartographic system. Geographic Information Systems Laboratory, SUNY at Buffalo.

Fahey, L. 1972. An approach to map design. Proceedings, ACSM 32nd Annual Meeting. pp. 132-141.

Joyner, S.L. 1982. Automating color map design: a designer's viewpoint. Unpublished Masters Thesis, Department of Geography, SUNY at Buffalo.

Moellering, H. 1977. Interactive cartographic design. Proceedings. ACSM 37th Annual Meeting. pp. 516-530.

Monmonier, M.S. 1982. Computer-assisted cartography: principles and prospects. Prentice-Hall.

Negroponte, N. ed. 1975. Reflections on computer aids to design and architecture. Petrocelli-Charter, N.Y.

Robinson, A.H. 1975. Map design. Proceedings, Second International Symposium on Computer-assisted Cartography, pp. 9-14.

The views expressed here are the author's and do not necessarily reflect those of his company