INTERNATIONAL USER INTERFACE DESIGN IN COMPUTER-ASSISTED CARTOGRAPHY

Michael P. Peterson
Department of Geography / Geology
University of Nebraska at Omaha
Omaha, NE 68182

ABSTRACT

Most computer programs for cartographic applications were developed with little consideration for their use in other countries, thereby limiting their general usefulness. The major cultural component of a computer program is language. The introduction of graphical user interfaces has made possible the creation of less language-dependent programs. However, most current computer programs, even those developed within a window-menu environment, still contain words as the central part of the command structure. The use of graphical menu palettes to control the execution of a particular program for thematic mapping is examined. Various strategies are implemented for entirely removing the language component of the user interface. The major obstacle is the development of graphical symbols that are meaningful to people from different cultures. Standard symbols have already emerged for common graphical operations. The ability to interact with the program to learn the meaning of graphical symbols remains the most important aspect of the program interface.

INTRODUCTION

As computer-assisted cartography becomes more common, the way that people interact with computers to create maps is assuming a greater importance. An international aspect of the problem is the cultural component of a computer program. This component of a computer program for cartography is apparent in a variety of ways including different cartographic practices or different ways of expressing numbers. However, because a word-based command structure is normally used to direct the execution of a program, the most serious obstacle for the acceptance and use of a computer program in another culture is language.

As a result of initial developments in North America and the economic importance of the software industry to the United States, the english language has emerged as the most common language in computer programs. Most existing programs for cartographic applications use an english language command structure. Even programs written in other countries will often use the english language as the defacto standard for interacting with the computer. Presently, it is essentially impossible for a cartographer who is not familiar with the english language to create maps with the computer.

The overall objective of this research is to improve the human interface with maps by improving the user interface with computer mapping programs. The specific objective is to create a computer mapping program that would be usable by english and non-english speakers alike. This will be accomplished by designing and implementing a more international, graphical user interface based on icons for an existing computer mapping program. The graphical

nature of cartography creates a natural link to the design of graphical user interfaces. Before proceeding, it is important to examine the developments in the computer user interface and the way that computer programs are normally 'translated.'

1.0 THE HUMAN-COMPUTER INTERACTION

The interaction between man and computer has been of concern since the advent of computers but has been of particular concern since the introduction of the microcomputer in the early 1980's (Bass 1991, Bodker 1990, Bolt 1984, Carroll 1987, Card et.al. 1983, Gardiner & Christie 1987, Hamilton et.al. 1990, Myers 1988, Norman & Draper 1986, Shneiderman 1987, Thimbly 1990). An important concept in relation to this interaction is interface. The word 'face' indicates a junction or boundary between two completely different systems. An example of a face would be the boundary between two substances that do not mix, such as oil and water. If these two substances are placed in a container, a face or boundary forms between them. Interface refers then to the interaction across such a face. Interface is used to describe the set of rules and conventions by which one organized system communicates with another. Clearly, the inner-workings of a computer are vastly different from the intellectual constructs of the human mind. The user interface represents the bridge between these two systems.

In the late 1960's, Alan Kay, at the Xerox Palo Alto Research Center (PARC) began to pursue a particular user interface design based on a 'desk-top' metaphor. He envisioned the computer screen as an analog to a desk that includes a variety of documents and tools to manipulate them (typewriter, calculator, etc.). Kay's research at PARC eventually produced a number of revolutionary computers including the Star. Released in the late 1970's and priced at $16,000, the Star had a bit-map display, used a hand-held pointing device called a mouse, displayed information in separate windows, and supported pop-up menus that appeared on the screen in response to a click of the mouse. The Star represented a psychologically motivated approach to the user interface, emphasizing a consistent, well-thought-through model (Smith, et.al.,1982). It has changed how we think of user interfaces.

1.1 The Graphical User Interface. The Xerox Star implemented a graphical user interface (GUI) based on icons. Although the concept of GUI dates to the early 1970's at PARC, it was not until the mid-1980's with the introduction of the Apple Macintosh that the potential of this type of interface began to be realized. The Macintosh implemented the desk-top metaphor complete with icons, windows, dialogs and the mouse. Other features of the Macintosh included a high-resolution bit-map screen, object-oriented graphics facilitating the moving and resizing of graphic objects, and an interface to laser-printing through a page description language called Postscript™ (Apple 1985; Chernicoff 1985, Tagnazzini 1989). The Macintosh microcomputer has since earned the reputation for having an intuitive, easy-to-learn operating system. In fact, the term 'Macintosh-like' has now replaced the once popular term 'user-friendly' to describe an easy-to-use program. New operating systems for microcomputers, such as MS-Windows from Microsoft, now incorporate many Macintosh-like features such as pull-down menus, windows and dialogs.

While both the Star and the Macintosh represented major advancements

in making computers easier to use, creating a true graphical user interface has remained an elusive goal. Very little of the Macintosh interface is graphical. Only the representation of the files as icons and the graphical menu palettes that are implemented by some of the programs could be construed to be graphical. The remainder of the interface is still based on words - the menu items, the text within the dialogs, etc. The reason that computer programs still rely on words is not because of hardware or software restrictions in implementing a graphical user interface but because of a misconception that words are always superior to graphics in communicating a message. This attitude is particularly apparent in North America where the dominance of the english language makes it easy to ignore the existence of other forms of communication.

The use of graphical icons in traffic signs and for international events such as the Olympics is an established practice. Research has shown that there are clear advantages to an iconic interface in computer programs. Lodding (1983) asserts that, because people find images "natural," because the human mind has powerful image memory and processing capabilities, because icons can be easily learned and recognized, and because images "can possess more universality than text," iconic interfaces can "reduce the learning curve in both time and effort, and facilitate user performance while reducing errors." Gittens (1986) verifies the ease with which graphical attributes of icons such as style and color can be used to represent common properties of a collection of objects. As early as 1970, Easterby recognized the advantages for international use of symbolic displays over those that are language-based.

The international aspect of the user interface has become particularly important. The overseas market for computers and programs is expanding rapidly. In 1990, for example, Apple computer earned greater than half of its revenue from international sales. As a result, there is a need for software to be published in other countries. Apple Computer has responded by translating the Macintosh operating system into over 30 different languages.

1.2 Program Localization. The emphasis on a word-based command structure is evident in program localization. 'Localization' refers to the process of adapting a program to a particular country or language. At one time this involved the translating of all words in the input and output sections of the source code and the subsequent re-compilation of the program. Today, programs are split into so-called 'code' and 'resource forks.' The code fork is written and compiled in a traditional computer language such as C, Pascal or Fortran. If done properly, this part of the program does not contain any words. The resource fork, created with programs like Apple's ResEdit, contains the definition of the menus, windows and dialogs - in short, all components of a program that would need to be localized. A company interested in localizing a program will usually have a representative in a foreign country translate the menu items, window names and dialogs with a resource editing program. The source code itself is left unchanged. Figure 1 depicts the 'File' menu from the Apple Finder program in both english and german.

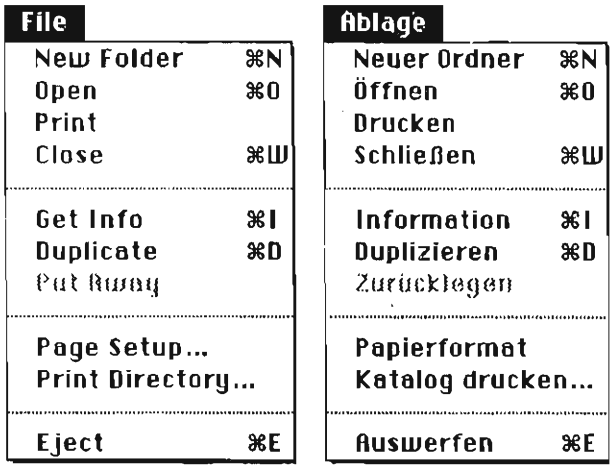

Figure 1. The Macintosh File menu in English and German

Localization is normally synonymous with translation. However, there are other aspects of the program that may need to be converted as well. For example, when writing a number in Germany, the meaning of the comma and the period is reversed. A number such as 1,234,567.89 would be written in Germany as 1.234.567,89. In China, commas are not used at all in numbers, e.g., 1234567.89. These formatting differences can usually be implemented within the resource fork as well so that the source code need not be altered.

In most cases, the localization process essentially involves the translation of an english language command interface into that of another language. It is generally not a process of creating a more graphical user interface that is less dependent on words. Recognizing the ability of graphics to communicate a message, it should be possible to create a more international user interface for a computer program, particularly for a graphics-oriented computer mapping program.

2.0 A GRAPHICAL USER INTERFACE FOR CARTOGRAPHY

2.1 The Program. This research to create a more international user interface is based on a computer mapping program that has been written by the author for the Apple Macintosh computer. The program, called MacChoro, has been designed for the creation of choropleth maps that use shadings to represent value by area. The program incorporates the Macintosh user–interface components including the use of icons, windows, menus, heirarchical menus, dialogs and the integration of the mouse as a pointing device.

Intended for the analysis of spatial data as well as the creation of maps for publication, the program integrates a spreadsheet, a variety of data classification options and the capability of creating map animations. Data is entered in the spreadsheet window and the map is displayed in the graphics

and reduced-view graphics windows. Six different methods of data classification are implemented: standard deviation, equal interval, quantiles, natural breaks, user-defined ranges and an 'unclassed' option. Once the map has been defined in memory as a series of polygon objects, the depiction of a new variable or classification is accomplished in under five seconds. For viewing individual maps at a faster rate, a map animation can be created by placing a series of maps in memory as bit-map representations. These maps may depict different variables and/or different classifications of the same variable. Animation sequences can be played back at speeds up to 60 per second and can be used to examine the effect of data classification or the change in the distribution of a variable over time. Animation sequences can also be saved to disk and later re-played.

Windows, menus and dialogs are the three major components of the user interface. Three separate window types are used in the program - a spreadsheet data window, a graphics window and an editor window (Figure 2). Pull-down menus contain the individual commands for the classification of the data and the drawing of the map, legend, bar scale and neatline. Two types of dialogs are used in the program. Option-dialogs are used in the program to control the various options. Specialized dialog items such as buttons, radio controls and scroll bars allow the mouse to be used to specify options. Message-dialogs serve to communicate a message to the user. This can be a simple error message such as "Printer not connected" to a sophisticated concept such as "This data classification procedure is not acceptable because the data are not normally distributed."

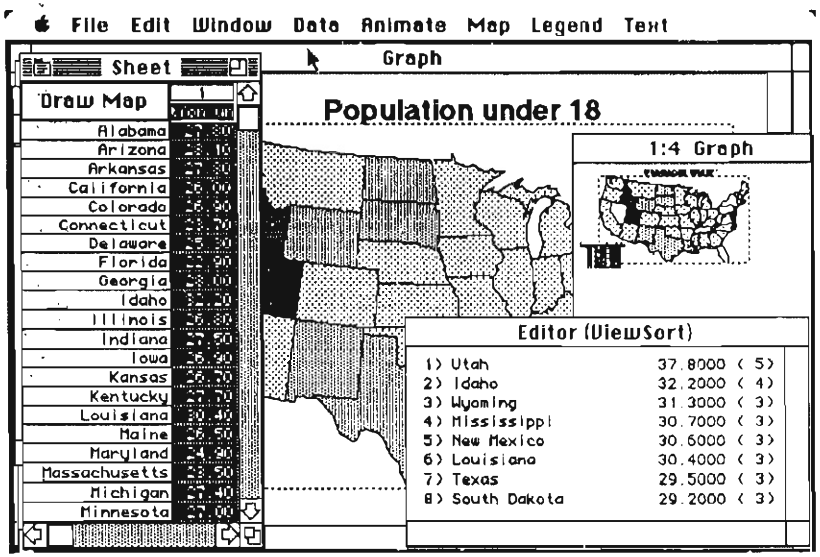

Figure 2. The Four Windows of the MacChoro Program

2.2 The Graphical User Interface. Maps themselves implement a graphical user interface in the sense that graphical symbols are used in the place of words to communicate a message. If graphic symbols can be used on a map to communicate what is often very complex information, then surely it

should be possible to use graphic symbols in a program to indicate program use and thereby create a more international user interface.

In creating a more international user interface for the MacChoro program, three aspects of the program had to be made graphical: 1) the menus; 2) the option-dialogs; and 3) the message-dialogs. Windows need not be made graphical because they do not contain text. The following is a description of how a graphical user interface was created for the program.

It was found that the interpretation of a graphical symbol or icons is much more dependent upon surrounding symbols than words are on surrounding words. With words, for example, it is possible to create a menu with a variety of dissimilar commands. A graphical menu must be more logically structured to facilitate the recognition of individual symbols. Therefore, it was not possible to simply substitute a graphical symbol for each command. Rather, the entire menu structure of the program had to be altered. For example, the previous 'Legend' menu contained items that would both draw and set legend, bar scale and neatline options. In the iconic version, the various drawing and option items were grouped together in separate palettes.

2.21 Converting Menus. Figure 3 depicts the palette menus used to draw items on the screen and the palette menu that is used to specify options. Previously, these menu items were in separate menus. In the menu palette version, a more logical ordering of menu items is necessary. The black triangle in the setup icons indicates that a dialog opens by selecting this item.

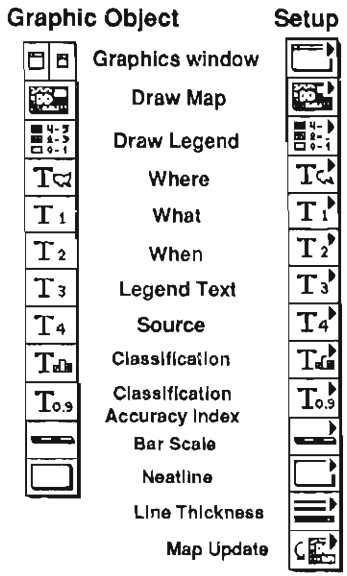

Figure 3. Palette Menus for the Graphic Elements

The original data classification menu and the corresponding palette menu are depicted in Figure 4. The palette menu includes more options at a savings of space. All possible number of classes are depicted along with an update map item in the upper corner.

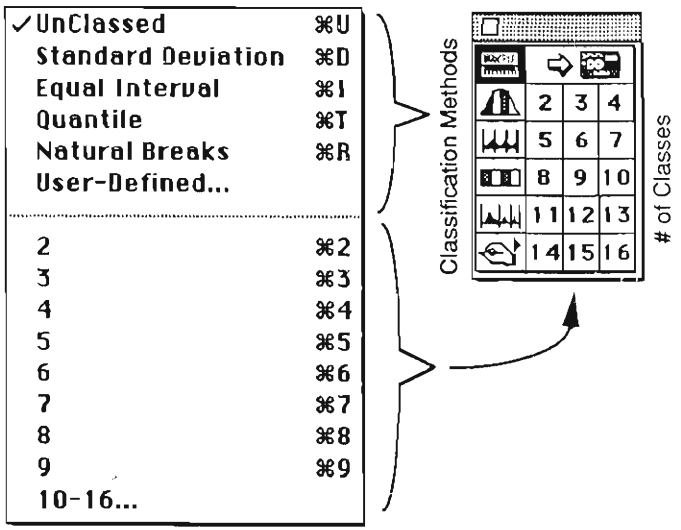

Figure 4. Text and Palette versions of the Classification Menu

2.22 Converting Option Dialogs. Figure 5 depicts the text and graphic dialogs used to control the depiction of the legend. The legend may be depicted as squares, rectangles or a histogram option in which the width of each shading box is proportional to the number of observations in that class. The type of legend desired may be selected from the upper part of the graphic dialog. Associated with the rectangle option is a scroll control to change the proportion of width to height. In a similar manner, a scroll control is associated with the histogram to change the length of the maximum histogram bar. In the lower left side of the graphic dialog are the options controlling the distance between the individual legend boxes. If the legend boxes are to be apart, then a scroll control can be used to alter the distance between legend boxes. The controls in the right-hand box control the distance between the legend boxes and the class-limit numbers. The font/style characteristics of the class-limit numbers may also be changed through another dialog accessed by clicking on the button below the scroll control.

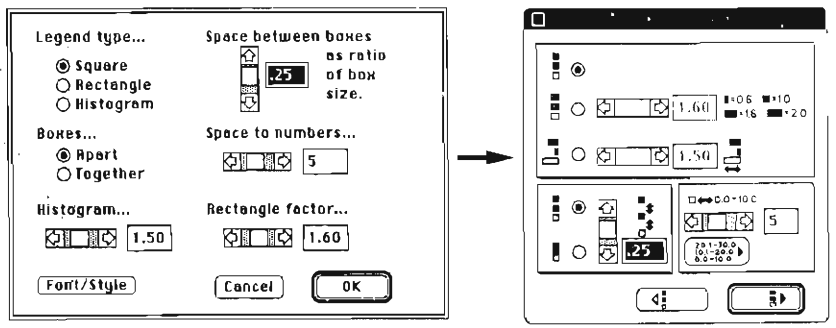

Figure 5. Text and graphical versions of the legend setup dialog.

2.23 Converting Message-Dialogs. Communicating a simple message with graphics can be easily done. To communicate a concept is more difficult. For example, the standard deviation classification option used in the program expects the data to be normally distributed. If it is not, a message is presented that warns the user that the classification method is not suitable but that the values could be made more normal if they were converted to their Log base 10 equivalents (Figure 6). To understand the message completely, one needs to know somethings about statitistics.

The best solution to the problem of communicating a message with a dialog is to write the program in such a way that such 'error' messages are not required. For example, it would be possible to check the data for normality before making the standard deviation classification an option for selection. In other words, lead the user away from mistakes before he has an opportunity to make them. In this way, you avoid the need for a dialog.

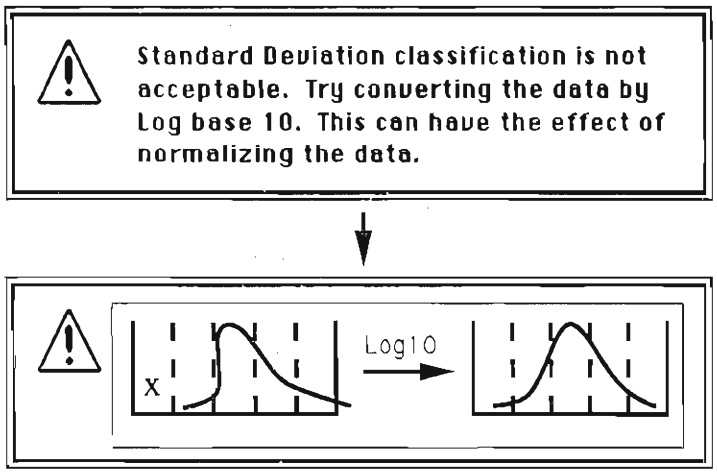

Figure 6. Text and graphic versions of a message dialog.

CONCLUSION

The microcomputer is becoming a more standard part of our lives. Its influence on how we use maps will certainly be extensive. To best serve the needs of a new generation of map users both in the United States and in other countries, cartographers must be involved in the design of more international user interfaces for computer mapping programs. This particular project is an initial effort in this area.

Language can be eliminated in most aspects of a program through a combination of graphical symbols and a sophisticated interface. The cartographic or statistical concepts, however, that underlie the program - data classification, for example - need to be explained with words. The meaning of graphical symbols that direct the program can be learned by trial and error, eliminating the use of a manual. This process is dependent upon the curiosity of the program user, a characteristic lacking in the more goal-

oriented adult population.

The most important observation is that the GUI has to be incorporated from the beginning. The writing of a program should begin with the interface design. This interface is then implemented in a graphic editing program where the symbols are designed and initially tested. In this way programs can be made both easier to use and more accepted in other parts of the world.

ACKNOWLEDMENTS

This research was partly supported by the German Fulbright Commission. The author wishes to thank Doris Karl of the Free University Berlin for her valuable suggestions.

REFERENCES

Apple Computer, Inc. (1987). *Human Interface Guidelines: the Apple Desktop Interface*, Reading, MA.: Addison-Wesley, pp. 144.

Bass, Len. (1991). *Developing Software for the user interface*, Reading, Mass.: Addison-Wesley Pub. Co.

Bodker, Susanne (1990). *Through the interface: a human activity approach to user interface design*, Hillsdale, N.J.: L. Erlbaum, pp. 169.

Bolt, Richard A. (1984). *The Human Interface: Where people and Computers Meet*, Lifetime Learning, Belmont, Ca.

Carroll, John M. (ed.). *Interfacing Thought*, MIT Press, 1987.

Card, Stuart K., Moran, Thomas P., and Newell, Allen. (1983). *The Psychology of the Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 469.

Chernicoff, Stephen. (1985). *Macintosh Revealed: Unlocking the Toolbox, Vol. 1*, Hayden Publishing.

Easterby, R. (1970). The Perception of Symbols for Machine Displays. *Ergonomics* 13(1), 149-158.

Gardiner, M. and Christie, B. (1987). *Applying cognitive psychology to user-interface design*, New York: Wiley, pp. 372.

Gittens, D. (1986). Icon-Based Human-Computer Interaction. *International Journal of Man-Machine Studies* 24, 519-543.

Hamilton, W. Ian et. al. ed. (1990). *Simulation and the user interface*, New York: Taylor & Francis, pp. 269.

Lodding, K. (1983). *Iconic Interfacing. IEEE Computer Graphics and Applications* 3(2), March/April 1983, 11-20.

Martin, James. (1985). *A Breakthrough in Making Computers Friendly: The Macintosh Computer*, Prentice-Hall.

Myers, Brad A. (1988). *Creating user interfaces by demonstration*. Boston: Academic Press, pp. 276.

Norman, Donald A. and Draper, Stephen W. (1986). *User Centered System Design*, Lawrence Erlbaum, Hillsdale, N.J.

Shneiderman, Ben. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishing.

Smith, D.C., Irby, C., Kimball, R., Verplank, W., & Harslem, E. (1982) Designing the Star user interface. *Byte*, 7(4), 242-282.

Thimbly, Harold. (1990). *User Interface Design*, New York: ACM Press, pp. 470.

Tognazzini, B. (1989). Achieving consistency for the Macintosh. In Nielsen, J., ed. *Coordinating user interfaces for consistency*. Academic Press, Inc., Boston, MA, 57-74.