INT. J. GEOGRAPHICAL INFORMATION SYSTEMS, 1987, VOL. 1, NO. 1, 51–65

The evolution of a graduated symbol software package in a changing graphics environment

WOLF-DIETER RASE

Bundesforschungsanstalt für Landeskunde und Raumordnung,
Postfach 20 01 30, 5300 Bonn 2, F.R. Germany

Abstract. The paper describes the developmental history of a software package for computer-assisted drafting of graduated symbol maps. Given the machinery prevalent 10 years ago, the first version was implemented on a mini-computer with vector-oriented devices, such as mechanical plotters and storage tube terminals. The programs provide functions for plotting graduated symbols of various forms, legends and line drawing. To improve map legibility the hidden areas in cases of overlapping symbols are removed. The visual separation of the symbols is enhanced by a halo effect, a small gap between the symbols, or the symbols and map background. New hardware required the reprogramming of the package to exploit the extended capabilities of new computers and graphic devices, including colour and hardware area fill. To increase flexibility, portability and maintainability, the graphic standard GKS (Graphical Kernel System) has been used as a device-independent software interface for the third generation of the package.

1. Introduction

The output from geographical information systems (GIS) will often be in graphic form and one factor determining the usefulness of such output will be the effectiveness of those graphics in conveying the information to users. Many of the maps output from such systems are choropleth maps in which administrative areas are shaded according to the value for each area, but there are many distributions in which graduated symbols will be a more appropriate method of representation. This paper describes the history of development of a software package for computer-assisted drafting of such maps. As well as cartographic aspects, some aspects of its implementation and the environments of hardware and software are discussed, broadening the scope beyond pure programming.

2. Graduated symbols in thematic mapping

The graduated symbol, also known as a proportional symbol (Dent 1985), is considered to be the appropriate technique for mapping the spatial distribution of absolute values at discrete points. The point can also serve to represent position for non-point references, such as administrative areas. The size of the symbol is proportional to the value of the variable associated with the location, whether in length, area or volume (figure 1).

In addition to size, other geographic variables, such as brightness, colour, texture or orientation, may be used to form the graduated symbol. The graphic variables can be either combined to increase the redundancy of the information, or used separately to portray several independent variables in a multivariate map. For the sake of readability, however, the number of variables should be limited; not more than three is a good rule of thumb. A recent discussion of the use of graduated symbols in computer-assisted thematic mapping can be found in Haldrup (1985).

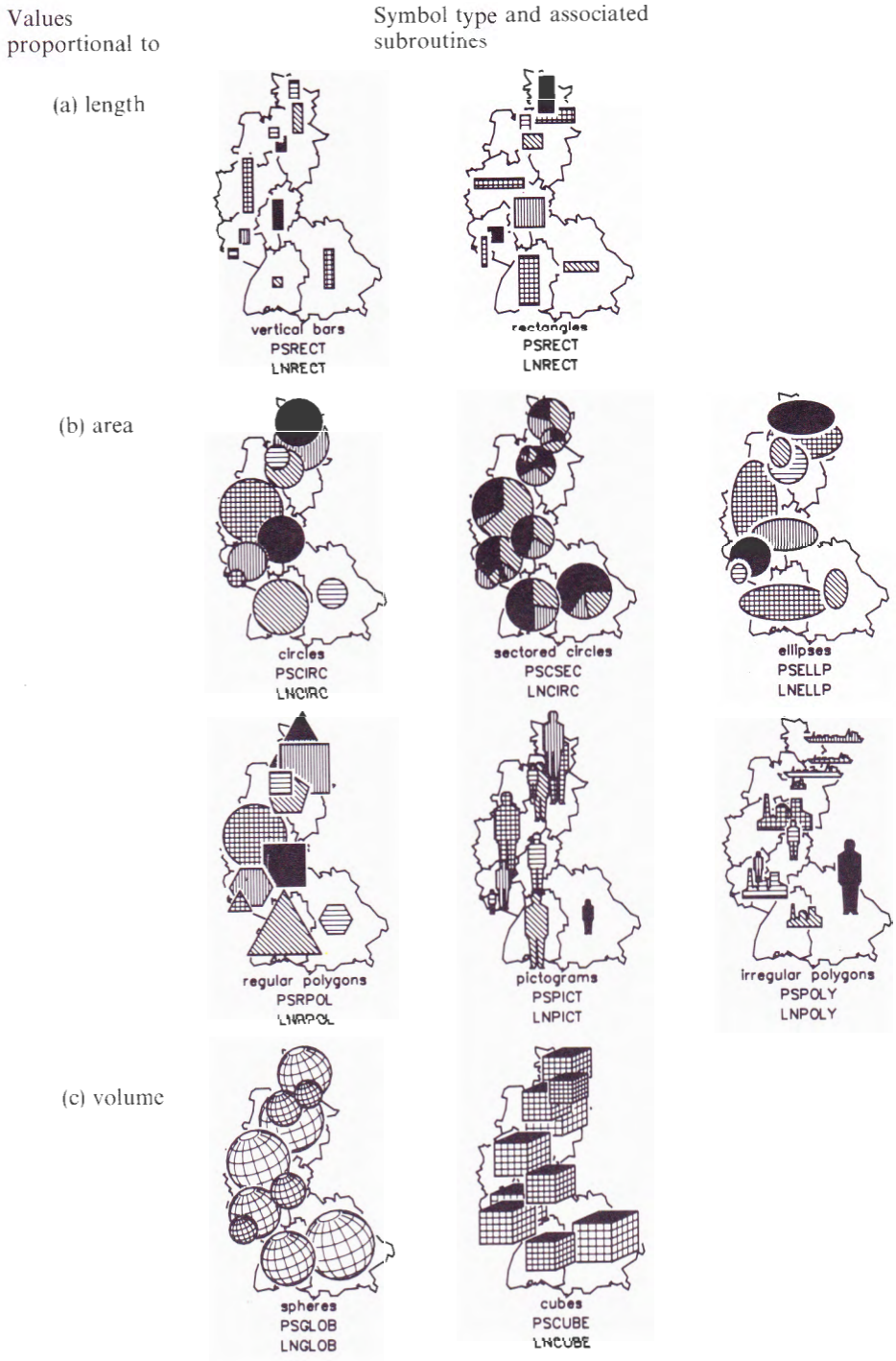

Figure 1. Graduated symbols and names of associated sub-routines.

3. Objectives of computer-assisted drafting of graduated symbols

The computer-assisted production of graduated symbol maps seems to be a simple task because the geometric construction of the symbols is well-defined. The formulae to calculate the coordinates of circles, ellipses, regular polygons and other forms can be found in most textbooks on elementary analytical geometry. To achieve a cartographic product comparable to hand-made maps, however, it is necessary to provide additional features besides the drafting of the symbols.

(1) Because the placement of symbols on the map is largely fixed, overlapping cannot be avoided. The simulation of an arrangement in three dimensions (stacking of the symbols in the z axis) enhances the legibility. To improve the impression of a perspective scene the hidden parts of the symbols should be eliminated.
(2) Readability is further enhanced if a small gap is left between overlapping symbols, or between the symbols and the map background (for example, situation, boundary lines). This small gap is known as a halo, and is standard practice in graphics and cartography.
(3) Sometimes it is necessary to modify slightly the position and orientation of certain symbols, for example, if parts meaningful for the perception of the map are hidden by other symbols.

The cartographic requirements of map readability and legibility are complemented by economic considerations for the development and the application of the programs.

(1) To keep unit production costs low the programs for drafting graduated symbols should make moderate use of computer resources.
(2) A user-friendly interface is an important factor for minimizing production costs, in respect of both program development and production.
(3) The overall development costs, including design, programming, testing and maintenance, should be distributed over a long lifetime for the program and a large number of users, again to minimize unit cost. Portability is important for the painless transfer not only to other computer systems, but also to subsequent generations of hardware within the organization, for the average lifetime of software is much longer than that of hardware.
(4) The software must, therefore, be sufficiently general and flexible to cover various application environments, computer systems and graphic devices; no modifications should be necessary to implement the software on a computer system different from that of the developer.

During the past 10 years, software for computer-assisted drafting of graduated symbol maps has been developed and used at the Federal Research Institute for Regional Planning (BfLR). Its development taught us some lessons about the lifetime of software, portability and standardization which are described in the following history of development.

4. General remarks on removal of hidden surfaces and haloes

4.1. *Hidden surfaces*

Elimination of hidden surfaces is a common problem in computer graphics. Various algorithms have been devised to give efficient and natural displays of complex three-dimensional (3D) scenes (Sutherland *et al.* 1974). The general case of the removal

of hidden surfaces in the space available to the user (maintaining the resolution of the input data) is rather complex and time-consuming. Perspective transformations, shearing and other peculiarities have to be coped with, along with shading, illumination and shadowing. Shortcuts are possible by reducing the information to the resolution of the picture space, and by exploiting special features of the graphic output devices, for example, opaque or transparent overplotting on raster devices, or specialized processors for 3D graphics. Increased efficiency is bought by increased device dependence, and subsequently by reduced portability and flexibility.

The removal of hidden surfaces is much easier with graduated symbols because some of the special cases causing so much trouble in a general solution can be ruled out from the beginning.

(1) The symbols have zero thickness and zero distance in the third dimension. Viewing transformations from 3D space to the picture plane (and vice versa for certain algorithms for hidden surfaces) is unnecessary.
(2) Shearing is impossible because the plane of the symbol has the same orientation as the picture plane.

Another dimension of complexity can be eliminated by reducing the problem of hidden surfaces to one of hidden lines. With vector-oriented graphic devices, area fill for graduated symbols is simulated by hatching. The points of interaction of a hatch line with an overlapping circle or polygon are easier to calculate than the intersection lines of two polygons and the subsequent recomposition of the resultant visible polygon.

4.2. *Halo effect*

The halo, the small gap between crossing lines or overlapping surfaces, enhances the visual separation of the elements in a picture. The shadow-like appearance of the halo stimulates the impression of perspective and depth in the z axis. It is used, for example, to improve images of wire-frame models in construction, architecture and other applications (Appel *et al.* 1979, Akman 1981).

Again the special case of proportional symbols is much easier to handle than the general case: a temporary enlargement of the upper symbol during the output of the lower symbol or line is sufficient. Practical experience has shown that a halo with variable width is superior to a halo with fixed width. The variable width is set proportional to the area of the upper symbol.

5. The first generation for vector devices

5.1. *The algorithms for the removal of hidden lines*

The first generation of software for graduated symbol mapping was developed for a 16 bit mini-computer (PDP-11/45) and vector-oriented output devices, such as a mechanical x–y plotter and storage cathode ray tubes. As mentioned above, the area fill for graduated symbols is simulated by hatching and the removal of hidden surfaces is reduced to the removal of hidden lines. The algorithm works as follows:

(1) Calculate minimum and maximum of symbol in y direction. Set y value for first hatching line to the minimum value.
(2) Calculate intersections of hatching line with symbol outline. Assign intersection code of '0' to intersection (x) values.

(3) Determine overlapping symbols. Calculate intersections of hatching line with overlapping (upper) symbols. Assign an intersection code of '1' to left intersection (entry), and a code of '-1' to right intersection (exit).
(4) Sort intersections (x values and intersection codes) in ascending order of x value.
(5) Plot intersection line as long as the sum of current intersection codes equals 0. Do not plot intersection line while the sum is greater than 0.
(6) Increment y value of hatching line by the distance of the hatching lines. Go to (2) until y value of hatching line is greater than the maximum y value.
(7) Repeat from (1) until the last symbol is plotted.

The best way of determining overlapping symbols depends on the type of symbol. For circles the well-know distance condition is used (see table 1, figure 2), and rectangles are even simpler. For polygons the comparison of the enclosing rectangles narrows the potential candidates to a sufficiently small number for an intersection to be calculated analytically.

To improve the visual appearance, especially if much overlapping occurs, the y value of the first hatching line should be corrected to modulo DIST in step (1) to ensure a common starting point of the hatching pattern for all symbols. Some other improvements and shortcuts may be added, for example an additional condition in step (3):

(3 *a*) If the left intersection value of an overlapping symbol is smaller than the left intersection value of the overlapped symbol, and the right intersection value of the overlapping symbol is greater than the right value of the overlapped symbol, go to (6) (line completely hidden).

Table 1. Example of an application of the algorithm to overlapping circles.

S	x value	Code
1	0·5	0
2	7·5	0
3	2·8	1
4	6·2	−1
5	1·9	1
6	4·1	−1
7	3·2	1
8	5·2	−1

Sorted in ascending order of x.

			Sum	Action
1	0·5	0	0	line start
5	1·9	1	1	line end
3	2·8	1	2	
7	3·2	1	3	
6	4·1	−1	2	
8	5·2	−1	1	
4	6·2	−1	0	line start
2	7·5	0	0	line end

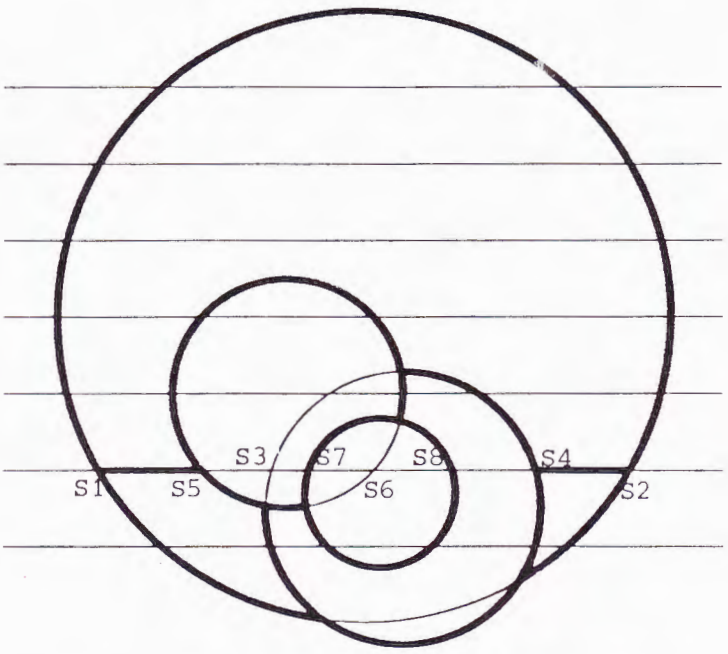

Figure 2. Results of the algorithm applied to overlapping circles.

If the hatching line is not horizontal, the rotation of the coordinates has proved to be the most efficient solution. Of course, the coordinates of intersections must be re-transformed before plotting. Whereas the calculations of intersections for circles and ellipses are trivial, polygons have to be checked edge by edge. Precautions are necessary for irregular polygons because more than two intersections per hatching line are possible.

The spheres are depicted by parallels and meridians resulting in circles and rotated ellipses in the picture plane. To intersect these curves an iterative approach, a relative of the bisection method (Gruenberger and Jaffray 1965) proved to be superior to a purely analytical solution.

The outline of the symbols is treated in a similar fashion. Again the intersections of two circles can be calculated directly. The coordinates of intersections are sorted in ascending order of angles, with the associated intersection codes. The intersections of polygon outlines have to be calculated by comparison edge by edge. The removal of hidden lines is done in the same way as with the hatching lines.

5.2. *The graphic interface*

When the program was being developed, only device-dependent software was available for interfacing to output devices. The capabilities of these devices were rather restricted. Thus the software interface consisted of only two functions: draw vector and change pen. Although no standard has been used, the modification for graphic devices other than plotter or storage tube terminals was not a major problem. The sub-routine calls could be easily emulated or replaced in the source code. In this respect the objective of portability was met, but not exactly in the sense it is now understood.

5.3. *Limitations*

The small addressable space of the mini-computer (64 kbytes) on one hand and an acceptable throughput for floating point operations on the other imposed some bias on the use of computer resources.

(*a*) The program had to be optimized for minimal requirements. As a trade-off the speed of execution was not optimal, but it was still sufficiently low for the most common symbols and to allow interactive use.
(*b*) The software was written in Fortran, but language features not defined in the Fortran-66 language standard, such as 16 bit integers, were used to keep requirements of memory low.
(*c*) The functions were implemented as a set of sub-routines, not as a main program.

The creation as a sub-routine library improved the portability, but hampered unrestricted use. To draw a map the user had to write his own program to read in the data and call the symbol and other functional sub-routines. A fair amount of programming skill, experience of data processing and time was required, resources which are quite scarce among cartographers in a production environment.

6. The supermini generation

The 32 bit superminis with a much larger addressable space and virtual memory capabilities, in our case a DEC VAX-11/780, opened new perspectives for cost-efficient computer-assisted mapping. Accordingly, a second version of the package emerged. The programs were optimized for execution time, because memory was no longer the limiting factor. Some deficiencies and minor problems were corrected and new features and functions, which had proved to be necessary for production, were added. The software interface remained unchanged. Only vector devices with their limited capabilities for area fill could be used.

A general main program was planned, but after the revision of the sub-routines had been completed, it was postponed. One reason was the advent of inexpensive colour raster devices, the other the emerging graphic standard GKS. It became obvious that both developments would have a considerable influence on computer graphics. Thus a complete revision of the sub-routines was necessary before a main program with an elaborate user interface or even an interactive design system would be feasible.

7. New developments in graphics hardware and software

7.1. *Extended requirements*

From experience with the routines for graduated symbols in actual production it became evident that a new version of the software needed improvement in several respects.

(*a*) Some additional cartographic functions, for example, for legends, were required.
(*b*) The programs should be able to exploit the extended graphic capabilities provided by new output devices, such as different styles of line and area fillings, with default or user-defined colours and patterns available.
(*c*) Portability and maintainability should be enhanced.

The simple two-function graphics interface was no longer sufficient to fulfil the new requirements and had to be replaced. The coding of the sub-routines was not in accordance with the principles of modern software technology, for example, in the excessive use of 'go to' statements. In the intermediate version for the supermini, the use of new language constructs in Fortran-77 had been deliberately avoided, because compilers for Fortran-77 were not generally available at that time. This restriction is no longer valid and some of the 'spaghetti code' is to be replaced by 'if–then–else' constructs.

7.2. *The graphic standard GKS*

The only way to provide extended graphic capabilities and maintain portability is to use a generally accepted standard for interfacing graphic devices. The Graphical Kernel System (GKS) was chosen because it was the first graphic standard adopted by important national and international standards organizations, among them ANSI, DIN and ISO. The 'virtual workstation' liberated the user from the constraints of a specific real device. GKS provides the desired additional output functions for area fill and styles of line. The elaborate functions for input and picture manipulation furnish excellent tools for interactive map design and manipulation (Enderle *et al.* 1984).

Our first enthusiasm about GKS was dampened progressively when we applied it on our graphic software, including the graduated symbol routines. We came to realize that such a general system is much more complex than the basic software for Calcomp plotters or the PLOT-10 sub-routines for Tektronix terminals with which we were familiar. There are so many traps that a newcomer to GKS cannot avoid falling into at least one. An application-specific shell would help, but this shell has to be a standard, again to guarantee portability.

Defining a standard is one thing, implementing it another. The overhead was so large with a validated GKS implementation that we decided to revert to our own GKS emulation for production work. Originally the emulation was intended to be an educational tool with a very restricted range of functions. A selection of functions below what is specified in the 'minimal' level of the ANSI standard is interpreted and transformed into calls to the device-specific basic software. This solution is certainly not elegant, but it is economic and we hope that more efficient implementations will be available in the future. The suppliers of GKS have learned their lessons, for example, that Fortran is not the ideal language for implementing system functions. More and more of the functions now executed in software will be provided by the workstations, thereby relieving the host of elementary work.

7.3. *Raster devices*

As a consequence of advances in the introduction of integrated circuits, especially for microprocessors and memory chips, graphic devices based on raster technology became less expensive. For economic reasons, but also because of their superior performance, they are replacing the vector devices progressively, except in special markets. With a device-independent interface such as GKS the applications programmer can ignore the specifics of new hardware and leave the support for the new devices to the implementation of GKS.

However, raster CRTs, inkjet and electrostatic plotters can simplify considerably the removal of hidden surfaces in graduated symbols. The symbols are sorted topologically in the z axis according to cartographic practice. Plotted in the right sequence from background to foreground, the smaller symbols overlie the larger

symbols further in the background. For the reasons mentioned above, shearing is not possible. Complicated algorithms for hidden lines are no longer necessary, since plotting objects is sufficient to mask any underlying object.

On the other hand, very many vector devices are still in use and these cannot be neglected. Even the new graphic terminals, although internally raster devices, expect data in vector format; for example, all the look-alikes of the Tektronix 4010 series. The objective of portability is not met if the software is restricted to a specific type of output device. Although raster devices would simplify the removal of hidden surfaces and save execution time, device independence has a higher priority.

7.4. *Segments in GKS*

The Graphical Kernel System not only provides a standardized device-independent interface for input and output. The more advanced features allow the subdivision of the picture into parts called 'segments'. A segment can be 'picked' directly by a 'pointer' without special book-keeping in the application program. Segments can be moved, copied or deleted independently of the rest of the picture. Highlighting segments may be specified to attract the attention of the user, or as an acknowledgment (receipt) for a successful pick operation. Priorities can be assigned to the segments for input, and the visibility of the output in the case of overlapping segments.

If each graduated symbol is defined as a segment, interactive manipulation could be effected quite easily. A symbol is located directly by the pick function. The segment can be moved without regeneration of the complete picture, provided the device supports that function. The assignment of priorities of visibility is a restricted but adequate form of removing hidden surfaces if the segments are moved interactively.

As usual, reality is rather different from theory. One problem is the generation of haloes which cannot be implemented in a general way by GKS functions. Less obvious is the fact that segment priority by GKS functions does not necessarily include the removal of hidden lines for vector devices. Thus most implementations of GKS do not support segment priority, for good reasons. The general case of removing hidden lines is costly, both in programming and in execution. For some cases of segment overlay singular solutions do not exist.

8. The third generation of the graduated symbol software

8.1. *Guidelines for implementation*

The previous discussion has shown that the third generation of software for graduated symbols should exploit the new developments in hardware and software, but there has to be some compromise.

(*a*) The Graphical Kernel System is used as a device-independent software interface for graphic output, including area fill with colours and patterns.

(*b*) Segmentation of GKS should be allowed, but its actual activation should be left to the user. If the specific implementation of GKS does not support segmentation, or the main program is restricted to graphic output, the programs should function in the usual way.

(*c*) The removal of hidden surfaces can rely solely neither on the use of raster devices, nor on the segment priority feature of GKS. The algorithm for vector devices must still be a part of the software to maintain device-independence and portability.

(*d*) To enhance portability and flexibility the Fortran-77 language standard is used, including 'if–then–else' clauses.

The use of a programming language other than Fortran was never considered to be a serious alternative. Most of the code of the PDP-11 version needed only minor modification. The conversion to Pascal or ADA would have been very expensive and, in any case, implementations of GKS for these languages are still rare.

A general application program with an elaborate user interface and all the functions necessary to produce maps without the need for the users to write their own programs is the next logical step after the sub-routines have been completed.

8.2. *Implemented functions*

To draw a map using graduated symbols, at least four groups of functions are necessary to obtain a satisfying result: graduated symbol plotting, line drawing with removal of parts hidden by symbols, plotting of legends for form, size and filling of symbols, and lettering. The last group is not specific to graduated symbol maps and is not considered further in this context, nor are other graphic and geometric information on the map, such as location, histograms or scattergrams. Table 2 gives a summary of the functions that have been implemented together with their sub-routine names. Each form has its associated routines for drawing lines and legends (figure 3).

Table 2. Implemented functions.

	Sub-routine name for plotting		
Type of symbol	Symbols	Lines	Legends
Circles	PSCIRC	LNCIRC	LGCIRC
Sectored circles	PSCIRS	LNSIRC	LGSIRC
Sectored circles, extended	PSCSEC	LNCSEC	LGCIRC
Ellipses	PSELLP	LNELLP	LGELLP
Rectangles	PSRECT	LNRECT	LGRECT
Regular polygons	PSRPOL	LNRPOL	LGRPOL
Irregular polygons	PSPOLY	LNPOLY	
Pictograms	PSPICT	LNPICT	LGPICT
Spheres	PSGLOB	LNCIRC	LGGLOB
Cubes	PSCUBE	LNCUBE	LGCUBE
Area full legend			TYPLEG

(*a*) (*b*) (*c*)

Figure 3. Examples of symbols, line drawings and size legends. (*a*) Symbol; (*b*) line drawing; (*c*) size legend.

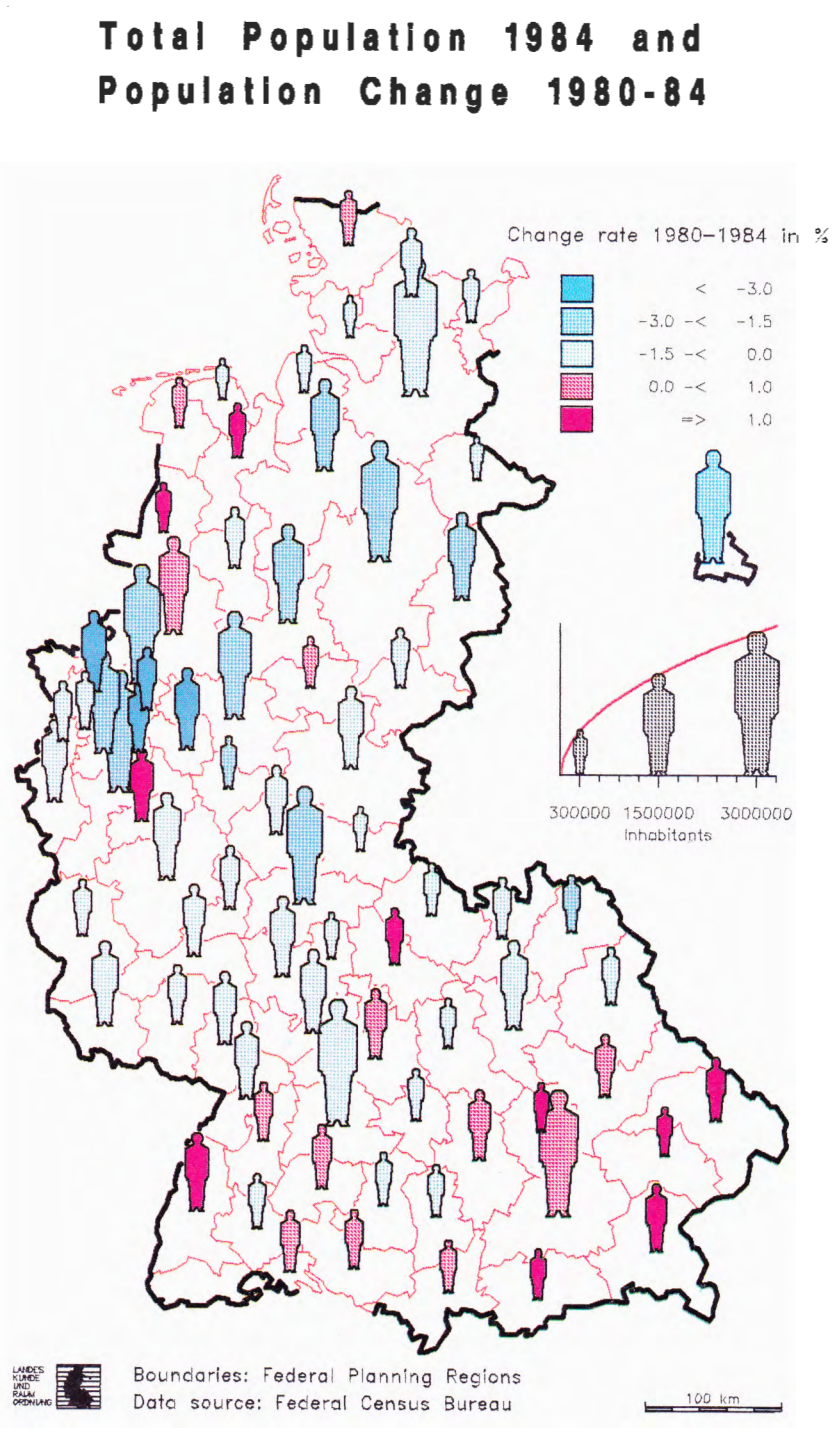

Figure 4. An example of the use of pictograms.

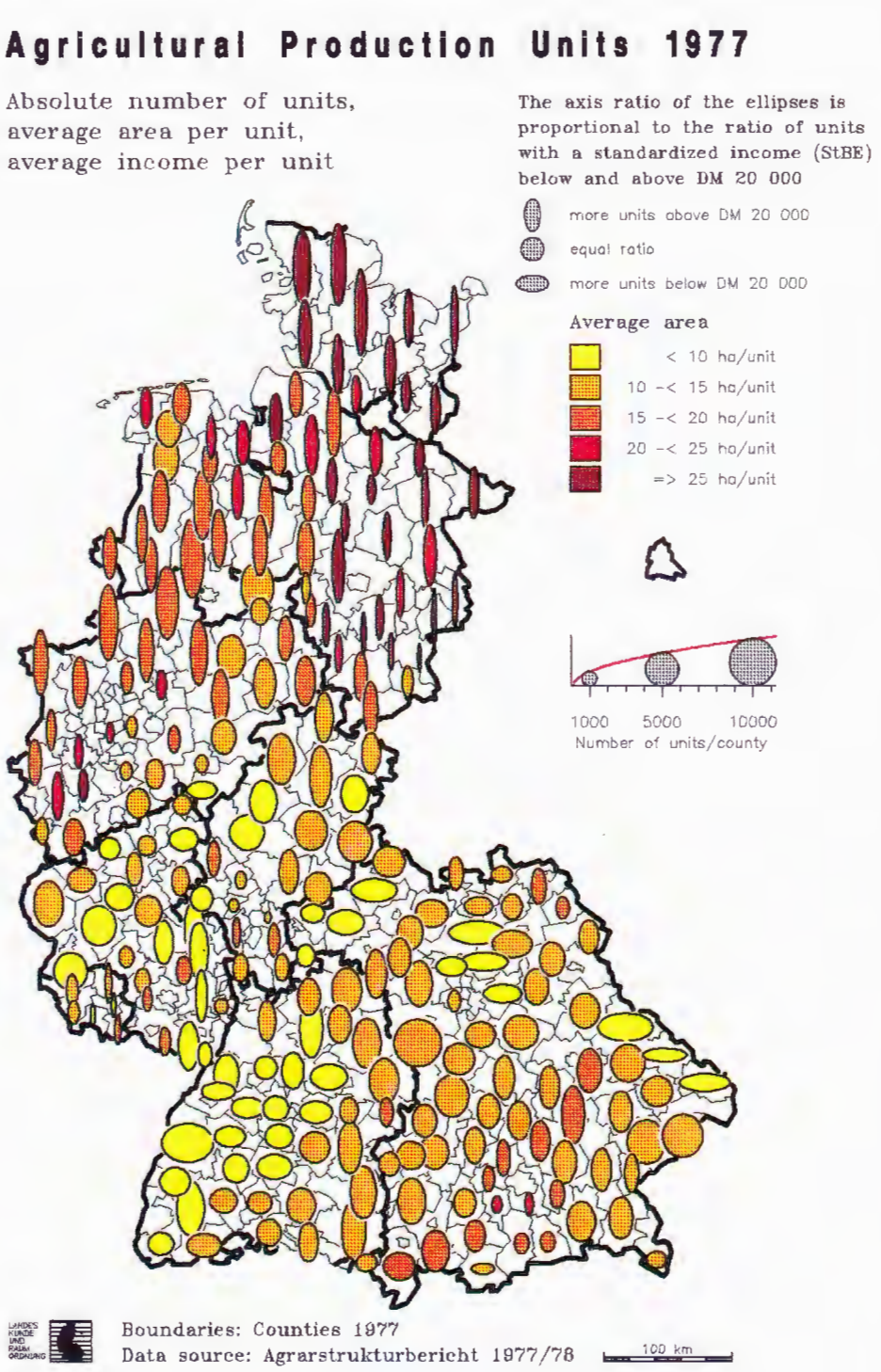

Figure 5. Trivariate map with ellipses.

In practical applications, circles, full and sectored, are the most commonly used symbols. Although other forms proportional to area convey the same information, circles are preferred. The human eye appears to perceive smooth forms as more pleasing than angular ones, although it is a large field for psychological speculation. Two different sub-routines for sectored circles are provided, a simple one with all the sectors having equal radii, and an extended one with individual sector radii, sector displacements and radial subdivisions of the sectors.

Pictograms are symbols of arbitrary shape scaled proportionately to the variables to be mapped. The calling sub-routine supplies the outline coordinates of the pictogram and the area the symbols should cover on the map. The sub-routine scales the coordinates to the size requested, and positions the symbols at the points on the map specified by the user. The coordinates of irregular polygons, however, must be specified in map space, including proper scale and position. (For an example, see figure 4.)

The symbols representing volumes should be used only on data with an unusually extended range. The visual representation works best with areas, so that the use of (pseudo-)volume symbols may result in misleading interpretations of the map. As might be expected from the experience with circles, spheres are preferred to cubes.

A function depicting pillars with a rectangular base will be one of the next extensions. The creation of pillars, as well as other new symbols, is relatively easy. The sub-routines in the table call a number of common functions for area fill and plotting outlines which can be used for symbols not yet contained in the library.

Normally one or two variables are represented on a map by the size and filling of the symbols. Style of line and colour of symbol outline may be used to define a third classed variable. The number of variables can be extended by using additional graphic variables, for example by varying the ratio of axes of ellipses, the number of edges for regular polygons and the outlines of irregular polygons.

Varying form and orientation by changing the axes of ellipses is an unusual form of representation, because manual drafting of ellipses is too costly to be practical. But the unusual attracts attention, and that is the first step in persuading the end user to have a closer look at the map. In this case, computer assistance is not only an economic advantage; it also provides a novel tool for transmitting the message more efficiently (Rase 1980) (figure 5). As might be expected, the smoother forms of ellipses make them more acceptable than rectangles.

It is important to stress again that multivariate maps should not be too complicated. Too many variables confuse or even embarrass the reader, and perception may be hampered or even prevented completely. In most cases less is more, depending, of course, on the message and the potential recipients.

8.3. *Options*

A set of options allows the user to control the output and the visual appearance of the graduated symbols. If applicable, the options are common for all types of symbol (Rase and Steffens 1986).

8.3.1. *Topological sort*

The original sequence of the symbols as supplied by the user is kept. The sequence is changed only if symbols overlap. Then the larger symbols are plotted before the smaller. If the user insists on a specific sequence, a field can be pre-set with sequence numbers. The same field contains these numbers after execution of the program. These

sequence numbers, either pre-set or generated by the program, can be used to assign priorities of visibility where segments are being used.

8.3.2. *Symbol filling*

Hatching is the most general option for filling areas. It can be used with vector or raster output devices. The hatching angle, distance and offset can be specified by the user. Besides styles of lines and colours provided by the GKS implementation, the user may select two arrangements of dashed lines (with variable lengths of dash), or up to 14 small symbols (markers) placed at the intersection of a rectangular grid. The grid and the small symbols can be rotated through any angle. Area fill with colours or user-defined patterns is restricted to raster devices and by the specifics of the GKS implementation.

8.3.3. *Outline plotting*

In accordance with GKS conventions, colour and the style and width of line of the symbols are set by calling sub-routines before the execution of a sub-routine. A line index is transferred to the sub-routine, either the same index for all symbols, or an individual index for each symbol. The latter feature allows the additional representation of a type variable.

8.3.4. *Halo*

For vector devices halo generation is performed during the removal of the hidden lines by a slight enlargement of the upper symbol(s). On raster devices (the removal of hidden surfaces by overplotting), a white surface with the same but enlarged outline is plotted before the symbol is plotted. The width of halo can be either fixed or proportional to the area of the symbol.

8.3.5. *Positioning of symbols*

The symbols may be either centred on the reference points, or 'standing'. Standing means that the lowest point or the base line of the symbol is equivalent to the reference point for the areal unit. The man-like shapes in figure 4 'stand' on the reference point.

8.3.6. *Segmentation*

In the Fortran binding of GKS, segments are identified by numbers. If the caller requires segmentation he supplies the number for the first segment. The routines increment the number for each segment (symbol) and return the last value. The range of segment numbers can be used for picking and other segment-related operations in the calling program.

Finally the user has to specify whether he wishes to use a vector or raster device for output. The inquiry could also be done by a GKS function, but for special effects the parameter has to be set explicitly.

8.4. *User interface at the BfLR*

The sub-routine package is the most general interface for the application of graduated symbol mapping. The environment of the individual user (for example, in respect of the specifics of access to locational and statistical data (data banks); the computer, its operating system and the attached graphic devices; the installed implementation of GKS; and the established conventions and procedures for interactions between programs, control and graphical output) has not been taken into

consideration. The sub-routines can be used in *ad hoc* main programs, or incorporated into existing mapping systems. Both earlier and current versions have been implemented successfully at other institutions.

As mentioned above, average users cannot be expected to supply their own main programs to plot the symbols, legends, boundary lines and other features on maps. A general way must be offered to produce a map with some form of control facility. In our case, we took the command language of an existing choropleth mapping program and extended it for graduated symbols.

The first version of this program, called PROKAR (Schmalenbach 1986), has been in general use for several months. As a first step it provides the most commonly used symbols, together with functions for plotting legends, histograms, boundary networks, lettering and locations consisting of points, lines and areas. The program will be extended gradually to include symbols not yet available. An interactive version is planned, but it will certainly not be developed in the near future.

9. Conclusion

The sub-routine package described in this paper provides an efficient set of tools for the computer-assisted production of graduated symbol maps. To distribute the development cost over a long lifetime for the software and a large number of applications, special attention has been paid to standardization. The Fortran-77 language standard and the graphic standard GKS are intended to provide the flexibility, portability and ease of maintenance necessary to satisfy both cartographic and economic requirements.

References

AKMAN, V., 1981, Halo—a computer graphics program for efficiently obtaining the haloed line drawings of computer aided design models of wire-frame objects. User's manual and program logic manual. Rensselaer Polytechnic Institute Image Processing Laboratory, Troy, New York.

APPEL, A., ROHLF, F. J., and STEIN, J. A., 1979, The haloed line effect for hidden line elimination. *Computer Graphics*, **13**, 151.

DENT, B. D., 1985, *Principles of Thematic Map Design* (Reading, Massachusetts: Addison-Wesley).

ENDERLE, G., KANSY, K., and PFAFF, G., 1984, *Computer Graphics Programming. GKS—The Graphic Standard* (Berlin, New York: Springer-Verlag).

GRUENBERGER, F., and JAFFRAY, G., 1965, *Problems for Computer Solution* (New York: John Wiley).

HALDRUP, K., 1985, Generation and editing of proportional point symbols on a microcomputer (DEC 'Rainbow' 100). Thesis submitted to the International Institute for Aerospace Surveys and Earth Sciences (ITC), Enschede.

RASE, W.-D., 1980, A family of subroutines for plotting graduated symbol maps. *Geo-Processing*, **1**, 231.

RASE, W.-D., and STEFFENS, L. J., 1986, *PROSYM. Unterprogramme für die Zeichnung von proportionalen Grössensymbolen, Version 3.0, Benutzerhandbuch* (Bonn: Bundesforschungsanstalt für Landeskunde und Raumordnung).

SCHMALENBACH, I., 1986, *PROKAR V1.0, ein Programmsystem für die computerunterstützte Zeichnung von Karten mit Proportionalsymbolen, Benutzerhandbuch* (Bonn: Bundesforschungsanstalt für Landeskunde und Raumordnung).

SUTHERLAND, I. E., SPROULL, R. F., and SCHUMAKER, R. A., 1974, A characterization of ten hidden-surface algorithms. *Computing Surveys*, **6**, 1.