THE UNIVERSITY OF HULL

Cartographic Information Systems Research Group
C.I.S.R.G.

Hon. Co-ordinator:
Dr. M. Visvalingam
Department of Computer Science
The University
HULL HU6 7RX

Tel. (0482) 465295/465951
Telex 592530

Discussion Paper Series Editors: Dr. M. Visvalingam
Dr. M. E. Turner

C.I.S.R.G. DISCUSSION PAPER 2

Problems in the Design and Implementation of a GKS-Based User Interface for a Graphical Information System

M. Visvalingam

Paper presented at GKS Review Workshop (Disley, 25-27 September, 1987), organised by the Eurographics Association.

September 1987

CONTENTS

ABSTRACT

State-of-the-art Graphical (mainly Geographical) Information Systems (GIS) began life in the early 1970s and are now well established on multi-user computer systems. They support the capture, storage, retrieval, mathematical manipulation and display of spatially referenced data. Current GIS do not support the interactive exploration nor the validation of multivariate data. We have researched the potential offered by the ICL PERQ graphics workstation and its implementation of the Graphical Kernel System (GKS) for the development of a GIS that would also facilitate interactive data exploration and concept refinement.

The design of the GIS confined the use of GKS to the user interface process. Whilst the concept of input classes in GKS is welcomed, nevertheless the current provisions are inadequate for the style of interaction required in a GIS. The categories of input classes are neither exhaustive nor mutually exclusive and merit a revision of concepts and functions. The functions of the user interface were consequently distributed between two co-operating processes and the use of GKS was limited to that responsible for output.

The confinement of GKS to one process within the design enabled the abstraction and clarification of the common functions that a GIS layer immediately above GKS should provide and the identification of a uniform, consistent and effective user interface.

One of the main requirements proved to be for a block manager to provide many of the functions of a window manager. The concept of a graphic block is familiar to graphic designers and cartographers but is missing in GKS where single level segment storage does not facilitate its expression. The block concept enables a collection of segments to be manipulated as a whole without sacrificing the facility for manipulating individual segments independently. A graphic tool may write to several blocks and many tools may

output to one block. The position paper elaborates upon the attributes, implementation, significance and outstanding problems of the concept of the block, which should be incorporated within GKS.

The concept of a key is a further higher-level output device used in graphic communication. The generation and interpretation of drawings is eased by the use of a well-defined specification which assigns a consistent meaning to syntactic elements. The specification, which defines the objects to be displayed and the conventions for their symbolism, may remain implicit or be made explicit through a key. Graphic symbolism relies ultimately upon the same output primitives as identified in GKS. However, there is a many-to-many relationship between keys and output primitives. The position paper describes a convenient mechanism, presently within the user interface, for definition and use of keys for multiple symbolism, for parameterizing the details of symbolism and for experimentation with creative symbolism. There is scope for deriving a simple model of output. The kernel need only cater for a single primitive, point sets, a configurable output device called the key, and a data record for additional control.

1. Background

The Graphical Kernel System (GKS) is the international standard for graphics (Hopgood et al, 1984). This paper reports some GKS-related aspects of research undertaken with the support of an SERC grant (GR/C/30474) entitled "Graphical Information Systems on Personal Workstations" during the period 1983-86. When this work commenced, powerful UNIX-based single user workstations with high resolution A4 bit-mapped screens were just becoming widely available. This grant provided an ICL Perq 1 workstation with 1 Mbyte memory and essential software, including PNX (the ICL Perq version of the UNIX operating system) and the ICL/RAL level 1b implementation of GKS (Gallop and Osland, 1985).

State-of-the-art graphical (mainly geographical) information systems (GIS), which began life in the early 1970s and which are now well-established on multi-user systems, support the capture/acquisition, storage, retrieval, mathematical manipulation and display of spatially referenced data. GIS is a fast growing field, particularly in the USA from where most current software comes. A UK Committee of Enquiry into the handling of geographic information set up in 1985 under the chairmanship of Lord Chorley reported in May 1987 (Department of the Environment, 1987). This committee identified lack of awareness of the capabilities of modern GIS as the major factor inhibiting effective use of geographical information. However, we believe that the powerful GIS of today with their wide range of facilities are consciously rejected by many.

Many end-users perceive current GIS as over blown or too high-powered for their purposes. Instead, both large and small users are building applications based directly on database management systems (DBMS), computer-aided design (CAD) systems and/or other products of information technology (IT). Some previous users of powerful command-driven mapping packages are already showing preference for systems which encourage a return to near manual cartography, practised on the versatile medium of the graphics screen rather than on paper. The ability to program and design by seeing and pointing to functions, objects and attributes and the capacity for highly interactive programming within an affordable environment is attractive. For example, AutoCAD and the Apple Macintosh are now widely used for incorporating stylised statistical diagrams and maps in documents designed to impress non-technical decision makers. Usability appears to be triumphing over GIS functionality.

Our research has addressed the urgent need to develop more flexible architectures based on modern IT for packaging and delivering GIS technology to end-users. To be successful, a GIS must provide both facilities for interactive design, to support near manual practices, as well as the inherently more powerful command-based connections to applications for the benefit of sophisticated users. The architecture must also support the derivation of simpler systems and must create an environment which will encourage the growth of specific applications through the addition of user-identified or user-created processes. This will enable users to start with a small configuration which can be gradually extended in a systematic way within a well-defined environment as, and when, required.

Furthermore, state-of-the-art GIS do not provide facilities for cross-checking and validation of multivariate data. This remains the users' responsibility to be undertaken outside the context of the GIS. These systems do not provide any support for interactive exploration of multivariate data. Graphic workstations offer considerable potential for cross-referencing elements in different views and thereby for gaining a better appreciation of the nature of the phenomena described by multivariate data. Our appraisal of the importance of data exploration and visualisation for concept refinement is set out elsewhere (Visvalingam and Kirby, 1984; Visvalingam, 1985). In the SERC-funded project, we investigated alternative architectures for GIS and the scope offered by graphic workstations, such as the ICL Perq, for the display of multivariate data in map and diagrammatic form.

The Chorley Committee of Enquiry endorsed the need for standards such as GKS in the area of geographic information handling and supported their promulgation by BSI and ISO (Department of the Environment, 1987, p. 95). However, recent trends in the behaviour of potential GIS users and our experience of GKS indicate that some concepts in GKS must be revised or extended in order to support and surpass the kind of interaction which end-users of bit-mapped display systems have come to expect.

This paper is concerned with three issues relating to the user interface, namely input classes and modes of interaction, the BLOCK for structuring pictures and the KEY for easing symbolisation. Hitherto, we have been largely concerned with identifying and researching some implications of these concepts, i.e. in formulating what is required rather than in how they should be provided within or outside the kernel.

2. Architecture of a Graphical Information System

GIS may be used with any type of spatial data; geographic data forms a subset of such data. The details of the architecture and functions and our GIS will be presented elsewhere but it is necessary to sketch the main ideas here so that the reasons for the suggestions relating to GKS are clear.

2.1 User Requirements

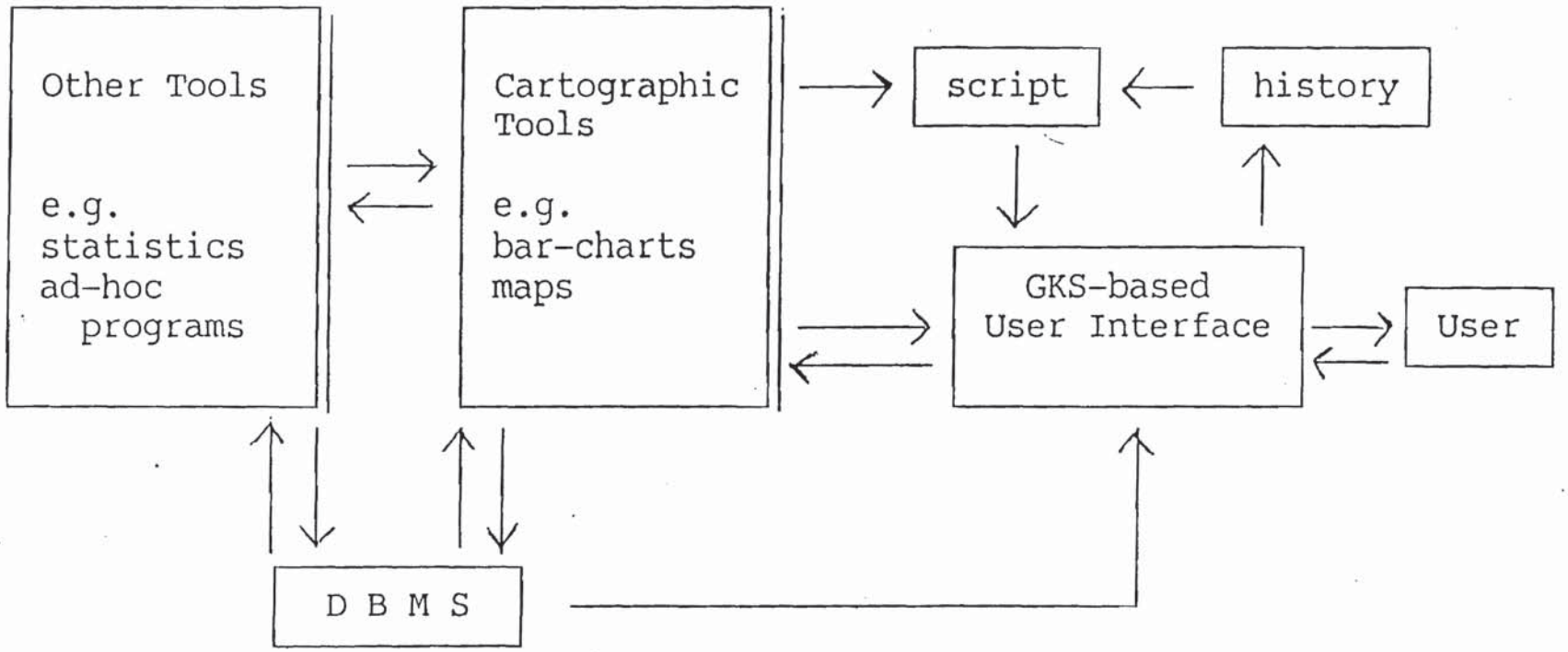
The GIS will allow a user to establish the nature of the relationships between data and information by visualisation of the data. Graphics will be used as an instrument for information processing. To effect this, the user will need to portray different views of data characterising the same entities, in the form of a variety of maps, statistical and network diagrams and tables. The user selects the entities, the relevant attributes and the views for display. He can then interactively explore the nature of these entities by pointing to them in one display and observing their occurrence and relationships to other entities in other displays, and, if need be, check the corresponding data records in displayed tables.

2.2 A simplified model

Figure 1 shows a simplified model to make clear the GKS related issues. The design provides for a hybrid user interface (UI) which supports direct interaction with a pointing device, as in a sketch-pad system, as well as a keyboard-based command-driven system. The UI also has other clients and accepts commands and/or data from nested command scripts, a DBMS and cartographic tools. Command sequences may be captured in history files in a user- and system-readable format.

The system design isolates the use of GKS to one process, the GKS-based UI, for several reasons. On the Perq 1, it took over eight minutes to compile and link-edit the simplest of the GKS demonstration programs which draws one straight line across the window. The restriction of GKS to the UI meant therefore that other parts of the system could be developed more speedily and without an in-depth understanding of GKS. Certain users of GKS have already

Figure 1: A simplified model of the GIS



said that their initial enthusiasm has been dampened by its over-complexity (Rase, 1987). The UI was therefore endowed with some minimal intelligence and suitable checks and defaults to ensure robust and sensible performance in the event of incomplete or erroneous input.

By forcing all access to GKS through one process, we hoped to abstract and clarify the common functions that a user interface layer immediately above GKS should provide and identify a uniform, consistent and effective user interface.

This design offers a further advantage. Cartographic tools, which are activated by the UI at the request of a user, are only necessary for picture generation. They are not required for experimenting with picture design. If the picture is properly structured, it is possible to change the aspects of pictures and transform segments to enhance the impact of the graphic using the UI alone. Also one-off graphics can be generated using only the UI by saving templates for standard graphics and including the variable elements interactively.

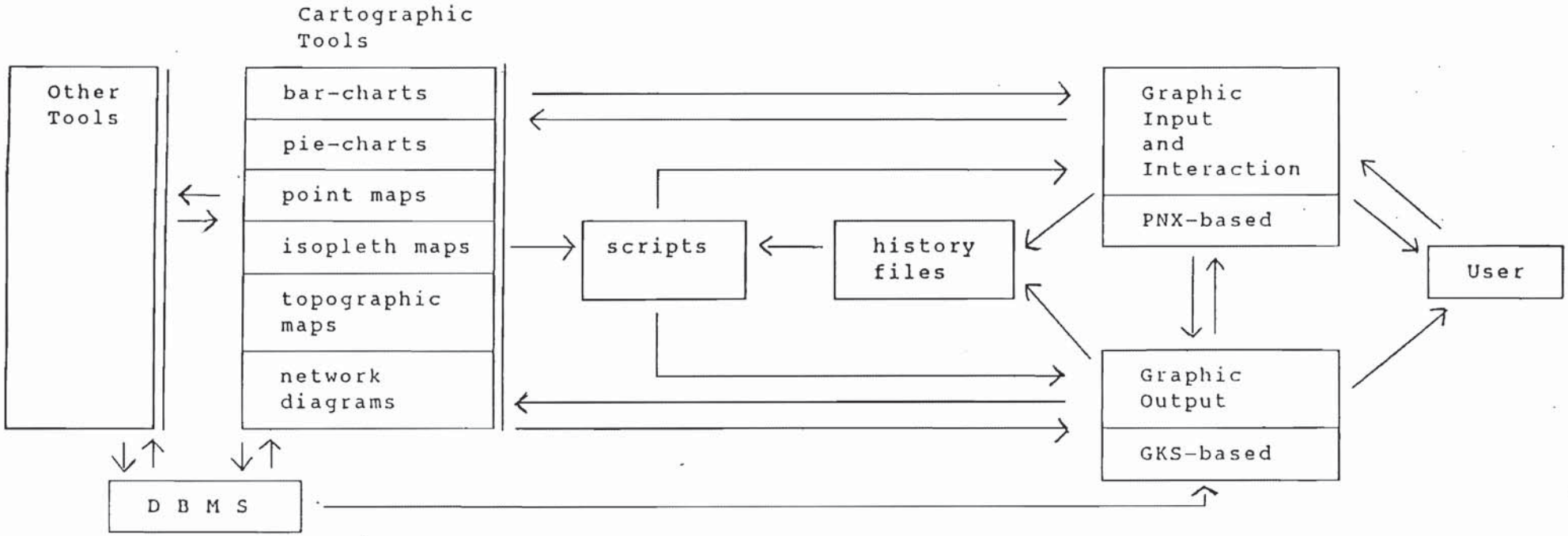
With sufficient experience, it may be possible to locate the GKS-based UI in hardware for speed. Without acceptable response times, there will be a continuing need to escape to machine dependent features, particularly for feedback during user interaction, defeating a prime reason for instituting a graphics standard.

3. GKS Revisions and Extensions: Some Proposals

3.1 Input

With minor exceptions, the GKS standards for output proved satisfactory for the GIS but the provision for input was a matter for concern. Consequently, the functions of the UI were distributed between two co-operating processes, namely UI-1 and UI-2, as shown in Figure 2. This more complicated model was forced by deficiencies in the GKS standard and in the level 1b implementation on the Perq, which only supports the REQUEST mode of input. It may therefore be no more than a temporary expedient. UI-1 is responsible for process management and for providing graphic interaction techniques absent in level 1b Perq GKS. UI-2 is responsible for picture generation.

Figure 2: A pragmatic model of the GIS



The GKS standard defines input classes and modes of interaction but not the styles of interaction which should be supported. These are left to the implementor. Until versatile interaction techniques are available by statute across a wide range of workstations, it will be necessary to isolate and locate interaction and input functions within a separate process such as UI-1. We therefore support the efforts of UK experts towards the formulation of configurable input devices which will allow several input measures to be associated with a trigger. This will enable the joint use of LOCATOR and CHOICE input as in the ICL Perq **spy** editor and facilitate the design of a forms-device for users who grow out of their initial over-enthusiasm for interaction.

There is some inconsistency in the way GKS treats the different input classes. The categories of input classes are neither exhaustive nor mutually exclusive and merit a revision of concepts and functions. It is possible to argue that STROKE is a set of LOCATORs and that STRING is a set of a configuration of CHOICE which allows one key input. Their separate existence within the standard is presumably justified and accepted on the grounds of convenience and because they allow the inclusion of suitable styles of interaction. For example, STROKE items may be LOCATEd with rubber-banding.

Yet, CHOICE and PICK are treated as singletons. This makes CHOICE an inconvenient device for specifying a selection such as "motorway **or** trunk road". It cannot be used for graphic input of selections of the type "0 to 14 year olds" on a list of 5-year age categories. Given that graphics act as pointers to data, is it possible to arrive at some abstraction which eases the use of direct manipulation techniques in formulating database queries?

PICK allows a set of one or more separately named output primitives within segments to be picked by direct interaction. In many GIS applications it is necessary to pick several elements within an area defined by an irregular polygon, i.e. to extract a spatial subset. This spatial subset is not the feature subset named by PICKIDs. The inclusion therefore of the concept of a set of PICKs within GKS will provide greater scope for an appropriate and natural style of interaction for multiple identifications based on different criteria.

To summarise, we believe that GKS should include provisions for the input of sets of measures and treat singletons as a special case for convenience. GKS

must also allow the simultaneous triggering of a set of nested input measures in order to provide greater scope to designers of user interfaces. The question of speed of interaction must also be addressed.

3.2 BLOCKs and GROUPs of BLOCKs

The concept of a graphic BLOCK is familiar to graphic designers and cartographers but it is missing in GKS where the single level segment storage does not facilitate its expression.

BLOCK management functions could provide many of the facilities provided by a window manager for structuring and organising the display. A BLOCK is a collection of segments, which could be manipulated as a whole without sacrificing the facility for manipulating individual segments independently.

To the user, a BLOCK is more than just a collection of segments. It allows the user to design a complex picture using the full Normalised Device Co-ordinate (NDC) space of the display and then shape, size, orientate and position it within NDC space. It is an independent virtual device within the NDC space. The user is able to define viewports, windows, segments and their attributes within each BLOCK, as if it was a separate NDC layer.

The main concepts relating to BLOCKs are outlined below; not all of them have been tested.

There is always a default BLOCK, which has the name BLOCK Ø. This extends over viewport Ø. It cannot be deleted, collapsed, re-defined or geometrically transformed by the user. This means that the concept of the BLOCK may be transparent to novice users, who do not require the BLOCK functions.

All segments and output primitives belong to a BLOCK. The BLOCK concept utilises existing GKS concepts and it does not require changes to the latter for its existence. Neither does it detract from the requirements for device independence but it has implications for the scope of bundles. The user may create/delete a BLOCK and give it a textual or iconic label, which could be displayed immediately above its frame. Collapsing a BLOCK causes segments associated with the BLOCK to become invisible; the label remains on display to

remind the user of its existence. The collapsed BLOCK may be expanded to view its contents and interact with them. The user may also re-define a BLOCK, which is just another way of scaling and positioning it.

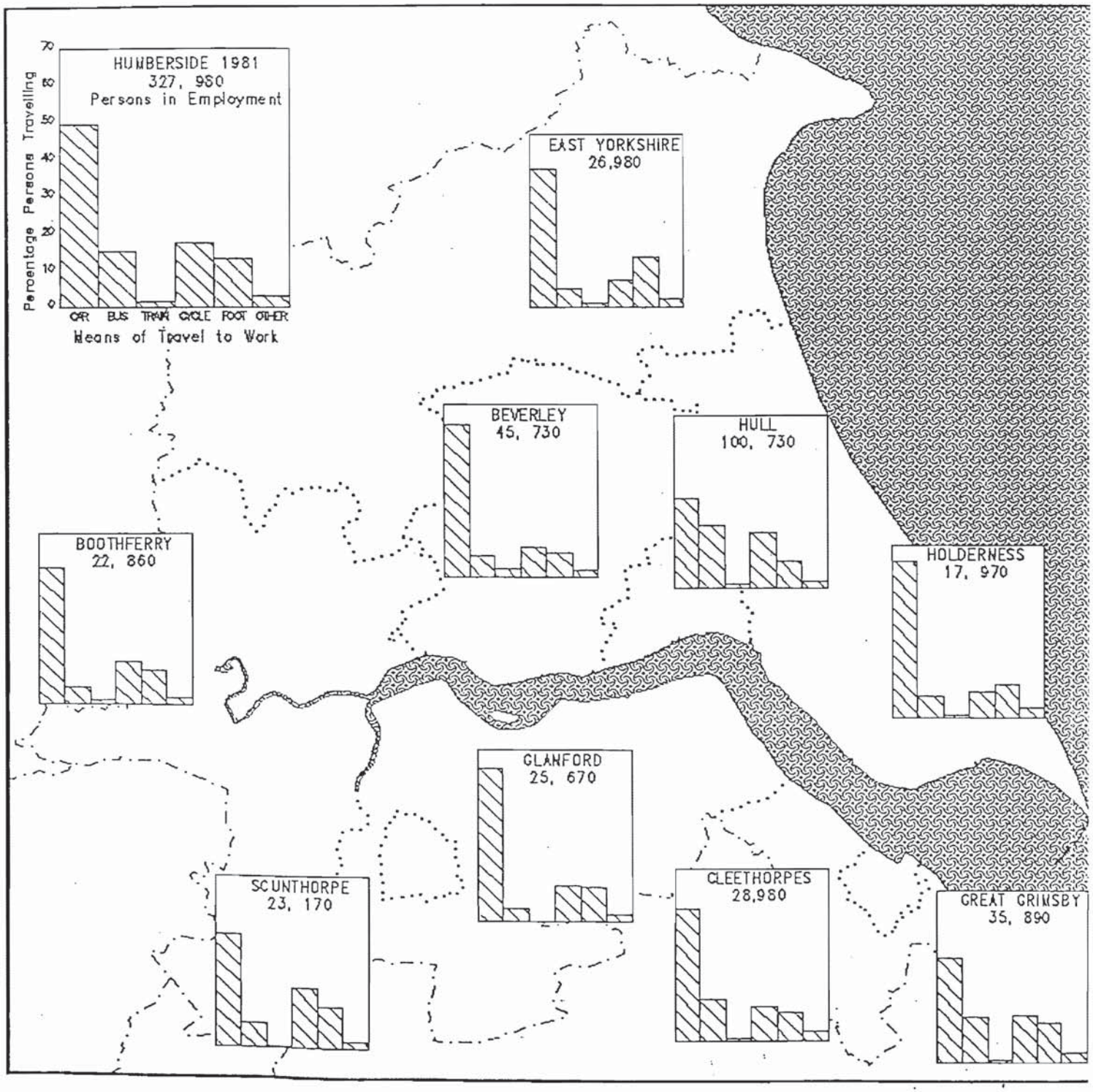
A BLOCK must be selected to receive output primitives and segments and user input. Each segment therefore may belong to only one BLOCK. Consequently, creation of a new BLOCK or selection of another existing BLOCK causes the currently open segment to be closed. It is, however, possible to append output primitives and/or segments to existing BLOCKs at any time. The output primitives belonging to a BLOCK may appear outside the spatial extent of a BLOCK. A BLOCK has a clipping rectangle associated with its defined/transformed extent. BLOCK Ø is always clipped. For other BLOCKs, clipping may be disabled; in which case, if the output primitives were also not clipped at the time of their creation, they can extend outside the BLOCK. This is necessary since some graphics associated with a BLOCK, especially text and also keys, are often located outside the BLOCK. For example, the data source for a display is usually cited below the BLOCK. Note that the histogram for Humberside in Figure 3, which also acts as a key, uses primitives outside the BLOCK.

Within the present level 1b Perq GKS, clipping poses a problem. Since the clipping rectangles associated with output primitives in Workstation Dependent Segment Storage (WDSS) are not transformed by segment transformations, all BLOCKs and primitives are unclipped by us at present. Otherwise they may become invisible after geometric transformations. This problem can be overcome when Workstation Independent Segment Storage (WISS) becomes available since it would be possible to re-define the viewport of the clipping rectangles.

It must also be possible to clear a BLOCK, i.e. delete all segments within the BLOCK without deleting the BLOCK itself. The user may also manipulate the contents and attributes (appearance) of the BLOCK. The segments form the durable contents of the BLOCK and their names are internal to the BLOCK. Thus, it is possible to use the segment name, 1, for the titles in all histogram BLOCKs in Figure 3. The appearance of this map can therefore be improved through a consistent specification which requests that the segments containing the title lines for each of the BLOCKs be moved down a fraction.

Segment attributes, like segment names, are also relative to the BLOCK. Thus, the priority of segments is made relative to the BLOCK and changing the priority

Figure 3: Map designed with eleven BLOCKs



of a BLOCK automatically changes the absolute priority of the segments associated with it. This requires that the implementation provides a sufficient number of segment priorities. The segment is visible only if its BLOCK is visible and detectable if the BLOCK is detectable. However, it is possible for a segment to be invisible when the BLOCK is visible and to be undetectable when the BLOCK itself is detectable.

It must be possible to PICK a BLOCK.

The BLOCK has additional segments, which may only be created by the system. These include the polyline for a frame, the text for its name and the fill area primitive for its spatial extent. The user should be allowed to determine the aspects of these objects. For example, specify the line style and colour for the frame and make it visible or invisible. The user will decide whether the BLOCK is opaque, transparent, and maybe, even translucent. If it is not transparent, the attribute of its background fill area may be defined by the user.

Transparent BLOCKs are useful for producing complex pictures generated by more than one tool. A simple climatic chart, of annual variations in rainfall, temperature and pressure, consists of a bar-chart and line graphs, which can be produced by two specialised tools. The use of a consistent default scheme for labelling rectilinear graphs ensures that only the segments annotating the Y-axis in one BLOCK has to be moved, for example from the left of the display to the right, to achieve the required design.

Highlighting a BLOCK does not involve the highlighting of its contents. Instead, the attributes of the frame, name or areal extent could be altered to visual cues reserved for this purpose. The currently selected BLOCK needs to be highlighted by the system. If users are allowed to highlight BLOCKs directly, this could result in confusion. Since the user has control over all non-reserved attributes of BLOCK-related entities (such as the frame, name and areal extent), it should not be necessary to provide an additional facility for users to highlight BLOCKs. Consequently, a segment may be highlighted even if its BLOCK is not selected/highlighted.

In general, all operations may apply to any BLOCK and not necessarily to the currently selected BLOCK. A BLOCK must be selected only to receive output

primitives and segments. However, some actions are not permitted on the default BLOCK as indicated in Figure 4.

Each BLOCK has access to all available viewports, since the latter are not tangible objects but merely serve to map objects described in World Co-ordinates (WC) onto the NDC plane and transform positional input back into WC. Viewports can be defined and re-defined very quickly in GKS. Thus, it is possible to keep a list of viewports for each BLOCK. This means that the number of BLOCKs is independent of the available number of viewports. The maximum number of available viewports per BLOCK will be one less than the implementation maximum. One viewport has to be reserved for the BLOCK definition, in the same way that normalisation transformation Ø is reserved for the unit square of the visible NDC space in GKS. The viewport of the BLOCK is defined when it is created by the user either interactively or through commands.

There must also be a facility for saving and re-using BLOCKs within and between terminal sessions; BLOCKs therefore have consequences for metafiles and WISS. BLOCKs provide a useful concept for generating instances of complex symbols. This requires the facilities associated with WISS. By symbols we mean transformable pictures whose attributes, such as colour or fill area symbolism, may also be changed.

BLOCKs also provide the scope for using templates, which may be regarded as complex symbols whose constituent output primitives may also be changed. The histograms in Figure 3, for example, are instances of templates. The concept of the BLOCK therefore provides an extremely powerful facility for generating complex pictures.

The use of BLOCKs necessitates the articulation of the concept of a set of BLOCKs, which may be subjected to the same set of operations. This set may be called a GROUP. It is a further level in the structuring of pictures. For example, the appearance of Figure 3 could be improved if the BLOCKs for districts (i.e. a subset of all the BLOCKs) which are all of the same size, could be reduced slightly by the same factor. This requirement illustrates the deficiency of the current CHOICE device, which limits the scope for user specification of the members of a GROUP, given a menu of BLOCKs.

Figure 4: Valid user actions on BLOCKs

Actions	BLOCK 0	Other BLOCKs
Highlight	-	-
Create	-	X
Delete	-	X
Collapse	-	X
Expand	-	X
Redefine	-	X
Name	-	X
Select	X	X
Clear	X	X
Clip	-	X
Transform	-	X
Copy	X	X
Save	X	X
Load	X	X
Pick	X	X
Group	*	X
Change Priority	X	X
Change Visibility	X	X
Change Transparency	X	X
Change Detectability	X	X
Change Name Attributes	-	X
Change Frame Attributes	X	X
Change Background Attributes	X	X

X : Valid user actions
- : System controlled actions
* : May be grouped for non-reserved actions

It must be noted that, in our GIS, UI-2 is independent of the process manager UI-1 and of the cartographic tools. Whereas a process is associated with a window in windowing systems, there is no such fixed connection between BLOCKs and client processes. This connection is dynamic and is made by the user at run time. UI-2 is itself spawned by UI-1 and it therefore belongs to the same process group as the other co-operating processes with which it communicates. Consequently, a process may write to any BLOCK and many processes may write to one BLOCK, albeit in a regulated way.

Not all ideas have been tested. Nevertheless, experience to date has already identified a potential problem relating to the scope of bundles. The GIS must support local adjustments to the design and the appearance of displays using the UI alone. Such retroactive changes can only be made if the attributes of segments were defined to be workstation dependent, i.e. if they were bundled. Within GKS, bundles are local to workstations. The BLOCK is a virtual device and within our present implementation of UI-2, bundles are global to BLOCKs. This unfortunately means that a great deal of attention needs to be devoted to the way in which client processes access and use bundles. The re-definition of a previously used bundle by another process can cause unexpected and potentially confusing results. The system itself needs to reserve a few bundles for its use; for example, for highlighting a BLOCK. Since segments are local to BLOCK, bundles similarly could be made local so that they may be manipulated in a compatible way.

However, local bundles have their disadvantages. For example, the histograms in Figure 3 use globally defined bundles. Thus, the shading of all histograms can be consistently changed from a hatch style to solid fill by changing the specification of just one bundle. This would not be possible if bundles were local, but local bundles can be conveniently managed given the concept of a GROUP.

It is possible to resolve these conflicting requirements within a GIS context if a sufficient number of bundles were made available by the implementation. However, if GKS is to include the BLOCK and GROUP, then attention must be paid to the scope of bundled attributes. GKS overcomes similar conflicting requirements of segments with local WDSS and global WISS. The WISS 'device' permits, for example, the re-definition of viewports and their associated

clipping rectangles. Could the use of bundles be resolved through a similar 'device'?

3.3 The concept of a KEY

The key, as portrayed in atlases and Ordnance Survey maps, is a further higher-level output device used in graphic communication. The generation and interpretation of formal drawings is facilitated by the key, which provides a well-defined specification for graphic symbolisation and thus a consistent meaning to syntactic elements in a display. This specification may remain implicit as in some one-off drawings or be made explicit through a legend, which may apply to a series of maps (Harley, 1975). The legend consists of graphic keys and associated textual descriptions; it identifies the classes of mapped objects. Each instance of a class is visually identified as belonging to a specific class by its graphic symbolism, as typified in the key.

The graphic symbolism ultimately relies upon the same output primitives as expressed in GKS. However, the GKS primitives may themselves be regarded as depictions or forms of the basic primitive, point sets.

GKS includes six output primitives. Cell array and the generalised drawing primitive exist for purposes of convenience and efficiency. GKS also relates text to the point specified in the text primitive. In reality, text may relate to points (such as cities on a small-scale map), lines (such as named roads) or to areas (such as counties). This is a conceptual rather than a positional relationship and automatic name placement by a computer remains a challenging task because the location of the text often cannot coincide with the location of the object for reasons of legibility, aesthetics etc. Consequently, within our GIS, the text primitive is treated as a facility for linking text to an arbitrary user-specified point. Text is thus viewed as a complex point symbol.

Polymarkers, polylines and fill area may also be regarded as graphic depictions of points, which are the only locational primitives. The kernel need only feature one locational output primitive, point sets. Within this simplified model, markers, lines, areas, text, curves etc. become high-level forms (attributes) for depicting points. This merely extends the existing nested

description of attributes within GKS. For example, the colour of a polyline, is already set by an index at present.

A simplistic version of this model was tested using the concept of KEYs with four primitives (polymarker, polyline, fill area and text) and the GKS aspect and attribute model.

KEYs specify the graphic symbolism to be used for sets of points and their detailed attributes. There is a many-to-many relationship between KEYs and output primitives. The same primitive can be used as a component of many KEYs; for example, a fine outline may be used to depict areal entities such as woodland, built-up land and types of social areas identified by cluster analysis. Each KEY may also use a number of primitives, i.e. it may consist of multiple symbolism. The extent of woodland may be depicted by a fine outline and the attribute, woodland, may be indicated by suitable fill area symbolism. In grid square mapping, cells with data are often marked by a square, enclosing some other symbol indicating the attributes of the area (Kirby and Visvalingam, 1982). It is also customary to show sample data points on line graphs and time series if the sampling framework is not continuous, regular or systematic.

Consequently, a bundle may be regarded as a rudimentary form of a KEY. It is one primitive of KEY. Bundles are used to define the graphic rendering of output primitives. KEYs may be regarded therefore as configurable models or 'devices' for the display of classes of objects. Graphic symbolisation and design will be eased if there was a convenient mechanism for modelling KEYs.

A KEY is defined by means of a KEY name, such as an integer, and a list of definitions, the elements of which have to be concatenated with the & character for multiple symbolism. These elements comprise GKS primitives and a list of valid local or global attributes and their values.

Figures 5a and 5b illustrate the use of local and global attributes for multiple symbolism; commands are included for purposes of illustration.

Figure 5b indicates that some of the hatch styles currently provided by implementors is redundant, since it is easy to achieve the desired effect with multiple symbolism. The user cannot define hatch styles, which have some advantages over patterns in that hatch patterns may be overlaid. Hatch styles

Figure 5: Use of multiple symbolism

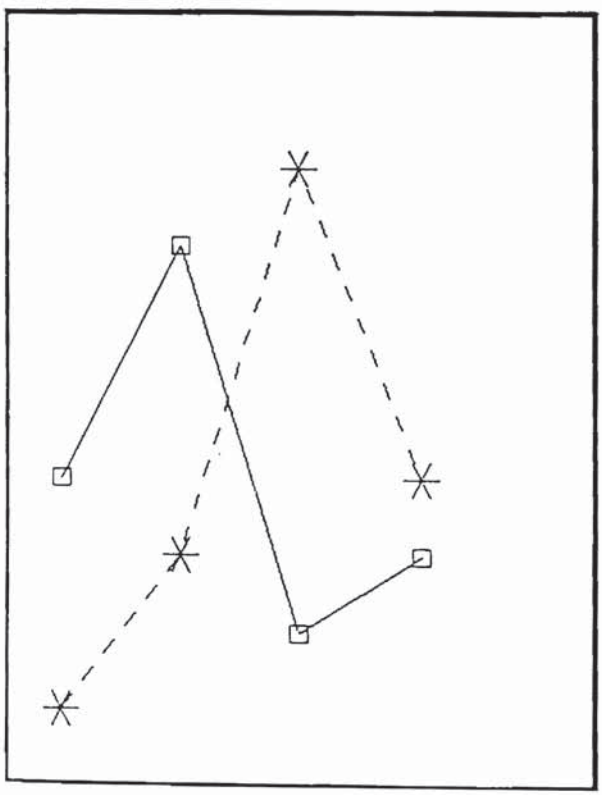
Key for 5a

```
#Key  :Define 1  ^Line bundle 1 & ^Marker bundle 3
#Key  :Define 2  ^Line bundle 2 & ^Marker bundle 2 size 2.0
```

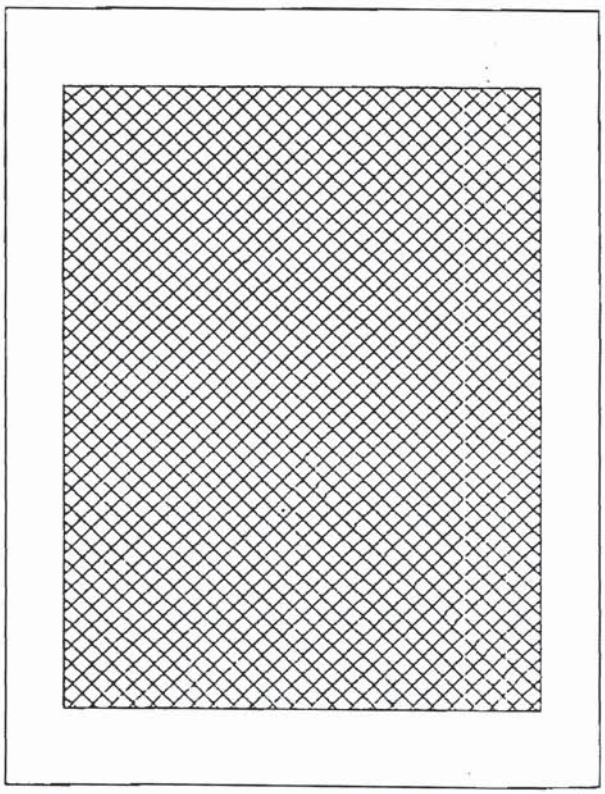
Key for 5b

```
#Key  :Define 3  ^Area style 3 hatch 3 & hatch 4 & style Ø
```

5a



5b



also are not transformable, thus the user cannot alter the density of lines. Given the scope for multiple symbolism, implementors should be more helpful in the hatch styles they offer. Only vertical, horizontal and the two diagonal hatches are required. Sets with a different spacing of these hatches and/or different line styles for hatches are more useful than some of the other patterns on offer at present, some of which are aesthetically unacceptable for use in cartography.

The use of pre-defined multiple symbolism is sufficient for objects which belong to discrete classes. Even data which is inherently continuous (i.e. unemployment rates) is sometimes pre-classified so that the map reflects areas of very high, high, average, low and very low incidence or severity. However, there are a number of schemes for the delimitation of class intervals and some distributions vary perceptibly with a change in class intervals, creating a subjective view of the phenomena being displayed; i.e. a view of an artefact rather than of reality. With the availability of high resolution colour displays, there is an increasing use of class-free mapping; proportional symbolism is used for displaying the distribution of phenomena which occur along a continuum. There are a number of parameters which may be attuned to reflect a change in magnitude (Bertin, 1983). The value of a hue may be modified. In monochrome displays, the point data may be displayed as markers, lines, areas or other symbols whose size, width, area, perceived volume or grey tone varies on a continuous scale.

Often the data has to be 'recoded', statistically transformed and/or weighted prior to mapping. It was decided that such data modelling was the responsibility of Service Tools and that UI-2 need only provide a facility for proportional mapping. This is accommodated within the definition of a key as shown in the following command.

```
£Key :Define 8  ^Marker bundle 1 %size
```

This allows the size of the symbol to be specified later together with the locational data for the marker. The specification of any GKS individual attribute can be deferred in this way.

UI-2 also allows the definition of in-filled objects. The '*' operator is used to request a set of symbols ranging from the smallest representation of a given

symbol, namely a dot, to one of the requested size in stated increments. If the step size is not given, UI-2 will produce an in-filled symbol.

The facility to define keys in command scripts and to load and use them as required, provides considerable scope for experimenting with graphic symbolisation. For instance, figure-ground relationships are often established by omitting background details where they are likely to obscure or detract from substantive objects. Rase (1987) was critical that GKS does not provide a facility for shielding areas. In proportional symbol mapping, the legibility and aesthetic appearance of the map is improved by shielding an area slightly larger than the symbol, creating a halo effect. The combined use of multiple symbolism and parametrised attributes can be used, as shown below, to create the desired halo effect.

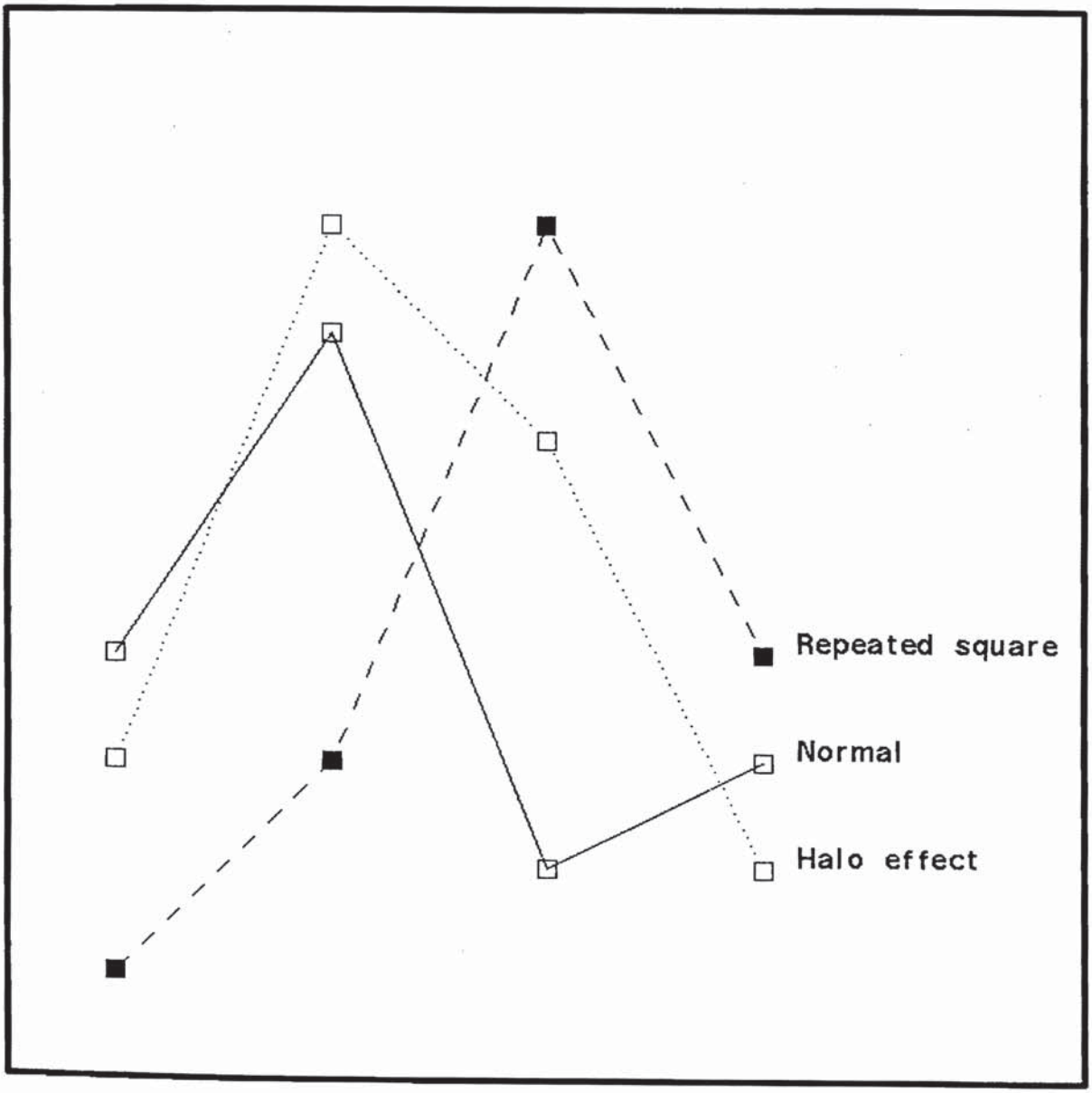
```
£Key  :Define 10  ^Marker bundle 3 * &
                  ^Marker bundle 4 %size &
                  ^Marker bundle 5 %size

£Comment    POSITION      KEY     SIZE and INCREMENT    SIZE for    SIZE for
£                                   of infilled symbol    bundle 4    bundle 5

£Data       0.5  0.5    Key 10     5.2        0.2          5.0          5.0
```

Key 10 requests multiple symbolism. Marker bundle 3 is an in-filled square in the background colour; bundle 4 is a square in the foreground colour and bundle 5 an asterisk in the foreground colour.

The data statement specifies that all markers be placed at position (0.5, 0.5). The effect is that an area slightly larger than the required foreground symbols is 'rubbed out'. If the co-ordinates of several points precede the literal, Key, in the £Data command, then the same key will be used for each of these points (see Figure 6). We are able to use the same mechanism for outputting text; for example, the symbol PO for every post office and T for telephones.

To summarise, the concept of a key provides a flexible and convenient mechanism for displaying the basic locational primitive, point sets. The concept needs extension and a more elegant expression. The articulation of the key could be

Figure 6: Repeated symbolism and halo effect



extended to include symbols. Some related issues, for example the scope of keys, needs to be clarified.

4. Conclusion

Usability of computing systems has become one of the more important factors to end users. They can see, compare, judge and choose for themselves the interfaces offered by competing systems. GKS must therefore ensure that application programmers are not limited by inadequate models and functions but are instead assisted by the existence of appropriate higher-level concepts and functions.

GKS is inadequate in the standards it provides for effecting input. This forces the use of device-specific features, which is contrary to the aims of the standard. Many applications will also appreciate the inclusion of some high-level abstractions, such as the BLOCK, GROUP and KEY.

Acknowledgements

I am grateful to Dr. G.H. Kirby, co-investigator of the SERC-funded project, and to past final year undergraduates in Computer Science, especially Ms. C.A. Mattin and Mr. A.K. Roberts, for their help in shaping and expressing the ideas presented in this paper.

References

Bertin, J., 1983, "Semiology of Graphics, Diagrams Networks Maps", translated by W. J. Berg, The University of Wisconsin Press, Wisconsin.

Department of the Environment, 1987, "Handling Geographic Information", Report of the Committee of Enquiry, Chairman: Lord Chorley, H.M.S.O., London.

Gallop, J. R. and Osland, C. D., 1985, "Experiences with implementing GKS on a Perq and other computers", Computers and Graphics, 9(1), 9-17.

Harley, J. B., 1975, "Ordnance Survey Maps a descriptive manual", Ordnance Survey, Southampton.

Hopgood, F. R. A., Duce, D. A., Gallop, J. R. and Sutcliffe, D. C., 1984. Introduction to the Graphical Kernel System G.K.S., Academic Press, London, 3rd edition.

Kirby, G. H. and Visvalingam, M., 1982, "The representation and use of multivariate information in graphic form", in Greenway, D. S. and Warman, E. A. (eds.), Eurographics 82, North-Holland, Amsterdam, pp. 149-154.

Rase, W. D., 1987, "The evolution of a graduated symbol software package in a changing graphics environment", International Journal of Geographical Information Systems, 1(1), 51-65.

Visvalingam, M., 1985, "Concept refinement in social planning through the graphic representation of large data sets", in Shackel, B. (ed.), Human-Computer Interaction - Interact '84, North-Holland, Amsterdam, pp. 281-286.

Visvalingam, M. and Kirby, G. H., 1984, "The impact of advances in Information Technology on the cartographic interface in social planning", Miscellaneous Series No. 27, Dept. of Geography, University of Hull.