Towards the Inclusion of Graphics in an Electronic Journal system

Nicholas. S. Hall.
Department of Computing and Computer Science,
University of Birmingham.

Abstract

This paper aims to give a brief summary of several methods avialable for reproducing diagrams on computer. The degree to which these methods are useful when applied to various types of diagram likely to be found in journals is discussed.

In the first section, several examples are given of how an electronic journal system could allow a user to perform browsing actions. This is used to indicate various limitations of the BLEND system leading to a description of proposals for the current study to compare methods for representing diagrams in an electronic journal. The second section describes a diagram classification carried out with a view to selecting some suitable example graphics from archaeological texts. Methods for storing and regenerating the various types are discussed and simple examples given.

Finally, examples are given which illustrate the storage space required for different diagrams. Leading on from this, the feasibility of holding catalogues of text and diagrams on CD-ROM are briefly considered.

Introduction

This paper describes some preliminary work being carried out at Birmingham to investigate various methods, and problems associated with providing graphics for electronic journal systems such as BLEND, (Birmingham and Loughborough Electronic Network Development).

The discussion is approximately divided into two parts. In the first of these, the aims and original proposals for the current study will be described. Here it is intended to provide a brief overview of how a prospective user might interact with a, "browsing", facility in some computer based electronic journal. The constraints this imposes on how we should treat graphics in such a system include, minimizing reproduction times and all that this involves, minimizing storage requirements, while at the same time providing high quality images/diagrams. Clearly, some of these conflict and there are obviously trade offs and compromises that the current study is intended to address.

In the latter part, various ways of representing graphical information will be discussed in the context of an electronic journal. I shall also discuss the merits of different methods for representing several types of diagram. These methods have been used as a guide to very loosely classify diagrams in terms of their complexity. Part of the present study is to investigate the applicability of several different representation techniques for various diagrams. Thus, the diagram classification has been used to direct the choice of sample illustrations from a number of archaeological journals. Examples are given which illustrate relative storage space requirements for different types of computer based images. In this context, I shall briefly consider the possibility of storing catalogues of diagrams and text on CD-ROM.

Browsing actions within an electronic journal

This is a brief and sketchy introduction to what we might want the passive side of an electronic journal system to provide. First it will be helpful to consider how a passive user may wish to use an electronic journal. In this context let us assume a "passive user" will be one who only reads the material held on a system, he is not an author, and does not modify any of the documents. What actions is this prospective electronic bookworm likely to want to perform?

A user may want to browse through a number of articles or papers held on the system, he may wish to skip through a number of abstracts in order to find the paper of interest. A reader might know that a certain paper lies within a particular journal, or he may want to browse through any of a number of journals held within the system. Such documents would in the general case contain text and also diagrams, the latter may range in content over a wide spectrum, from simple line illustrations to photographs.

While browsing through a journal what do we do? We might be looking for a section containing a certain type of diagram, perhaps this is connected with a heading in the text. We might not have a very detailed picture of the illustration in mind. A person would usually flip through the pages until he notices a diagram or picture that seems to match his mental image. Because the reader might be flicking quickly through the pages this match need not be a close one. In fact generally we do not give ourselves time to see any one of the diagrams in great detail. However, when something useful is noticed, the reader can stop at that page and extract the information contained in the diagrams and text to whatever level of detail is desired or possible.

If an electronic journal is set up on some computer system, there should be software to accommodate these "browsing" actions. It must allow us to flick through text and/or diagrams in a similar fashion. In fact, it might even tailor the way it presents the information to suit a particular reader's style of browsing.

Without dwelling on the details for the moment, we might want to search for particular diagrams. For example, there may be a need to search for diagrams that contain a named item. This is analogous to the manner in which one might use the computer to search for a certain string in a file containing ASCII characters.

Implications of Browsing functions for Software and Hardware

(A) Content

Documents containing graphics and text need to be held on a computer system in some manner.

(B) Possibility of searching

Some means should be provided where by not only strings of text can be searched and manipulated generally, but also diagrams. How do you arrange for searching on diagrams? Although the current study does not directly address this issue, it has been briefly mentioned here to illustrate that additional problems exist and that these will have a bearing on how graphical material is represented.

(C) Response time

It is necessary for a system to very quickly display pages of text, and with equal speed, any diagrams those pages may contain. It has to be quick, a user will very soon become frustrated with a system whose response time is appreciably slower than the rate at which he wishes to progress through a document. This is a very important point, the success of an electronic journal at achieving this goal may determine whether it gets wide usage, and, this in turn will eventually decide how cost effective it is.

(D) Presentation of Information on an Electronic Journal

Actually how the information should be presented is another very important consideration. For example, from a purely text point of view, the "concept" of the "page" seems to be crucial, it is an accepted unit for the display of information. It imposes an expected quantization onto the physical structure of a document. This is something that is subconsciously used to to measure the amount of work we have to do in order for example to have covered a given topic.

Surveys and experiments have shown[5] that from a computerized text point of view, people prefer to have pages of material "zapped" on to the screen very quickly rather than see a continuous roll of scrolling text. When one considers diagrams, this probably makes even more sense. In some hardware arrangements, scrolling diagrams may be difficult to arrange. Note that there are always exceptions to a rule. One area where you might want to scroll diagrammatic information is in map reading. One can envisage a map held in memory, too large to fit on the screen at its full resolution, the screen could then be a rectangular window onto any designated region of the map. Under user control the window could be moved across the surface of the map to any desired location. The result is that the diagram on the screen is scrolled in some direction.

Closely allied to the page concept, it is also important to provide the user with some graphical means of knowing where in the book or paper he currently is. With a physical book this information is provided by the wad of paper that lies either side of the current page.

Proposals and aims of the study

The main purpose of the current study is to investigate methods for the representation and reproduction of pictorial information. However, because this concerns electronic journal systems, the methods will be subject to constraints imposed by how a journal system should perform. This has been discussed in general terms earlier. Such constraints, for example fast reproduction speeds and image fidelity will require methods to be tailored in a systematic way to the diagram type and even perhaps the output device.

When proposals for this study were submitted[4], one feature of hard-copy journals that had not been replicated in the BLEND experiment, except in a very rudimentary manner, was the provision of a graphics facility. The only graphics that have been provided were obtained by direct mapping onto the standard ASCII character set. This is obviously a very severe limitation on the development of an electronic journal. There has been some work on the development of extended character sets including enhanced facilities for graphics. Such work has been carried out by various bodies including PIRA and ECMA.

Alternative approaches will be made possible with the advent of the TELETEX terminals and distribution systems. When the proposals were put forward, neither of these

systems were available. It was also thought that the Teletex system was likely to be expensive or of limited market penetration initially. In addition, the extended code solution might still not have a sufficiently versatile code set for the reproduction of diagrams.

As a consequence of these considerations, it was proposed to investigate solutions based on equipment currently available or shortly to become available to the would be user, (i.e. the microcomputer with communications software and hardware). In fact many of the participants involved in the BLEND/LINC electronic journal experiment used these devices to communicate with the Birmingham computer science centre's DEC-2060. However, the local processing power these micros provide has been very much under utilized, except for certain automatic logging procedures and off line file/document preparation.

The solutions mentioned, (teletex and extended character sets), rely primarily on transmitting "screen fulls" of data which are then simply displayed on the user's device. In this case, the level of sophistication is entirely within the transmitted data and the local decoding, (i.e. the presentation). These methods do not make very much use of the processing power that is usually available at the user's terminal. In addition,with many micros that provide hardware graphics facilities, there is usually quite a comprehensive software library available to support that hardware.

To investigate some possibilities and pitfalls the proposals suggested a comparative study of alternative methods of utilizing this local processing power and accompanying software in order to reconstruct source diagrams on a variety of microcomputers. The information transmitted from a host to some local device would be further processed by local software to for example, separate graphics from text and reproduce the, "screen" or "page", complete with diagrams. It was suggested the comparative study should consider the following three methods for representing graphics:-

(a) Bit map techniques, (this is effectively gray scaling the image).

(b) Reproduction in terms of a picture description language to be interpreted by some locally running process.

(c) Representation in terms of a high level language code, the same code as the locally running process, so that it is able to make use of the local graphics

subroutine library in a general manner.

One final word on the subject of original proposals. In order to direct the project toward a tangible goal, the suggested subject area for use in the pilot study was, "Computer Applications in Archaeology". There are several reasons for this, first, there is interest in the department in this subject. In addition archaeology was considered likely to provide a wide variety of material, with a minimal of bias toward computer techniques.

Diagram Classification Survey

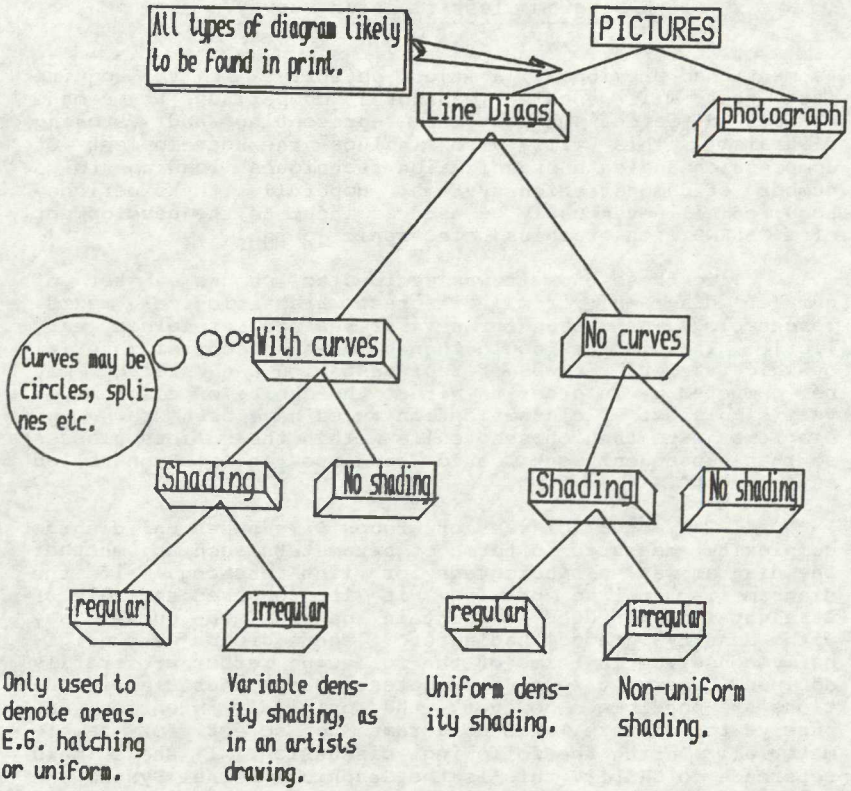
As mentioned previously, a major objective is to acquire data that will enable meaningful comparisons to be made between different methods of representing and storing diagrams. This will also include the development of compression and data organization techniques leading to a number of demonstration systems. Hopefully, the experience acquired will eventually be used as input to the development of a "BLEND with graphics" electronic journal.

To these ends it was decided to choose a set of sample diagrams from different archaeological texts, (archaeology was chosen for several reasons explained earlier). The diagrams could then be used as control material by which various methods of representation and storage may be compared. In order to direct the choice of diagrams, a very simple set of classification rules have been drawn up. Diagrams were then chosen to lie within the various classes so that subsequent work could continue in a "controlled environment".

The actual classes or groups were based on diagram complexity measured in terms of parameters such as, whether the diagram was a photograph or line drawing. If the diagram is a line drawing, is it composed entirely of straight lines or does it contain generalized curves say with density graded shading etc? These groups were originally conceived in terms of the following rather arbitrarily defined binary tree of attributes. Many other classifications are possible, however, the original intention was purely to dictate what diagrams to select for sample material. During the following discussion, I shall make reference to GKS[1], this is the Graphical Kernel System, an ISO standard for the representation of computer graphics.

Types of diagram found in sample texts

There a few simple perhaps obvious guidelines that can be used to categorize diagrams found in journals. These are illustrated in the following tree diagram.



From this tentative diagram the following list of attributes has been drawn up.

Type (1)	Line diagrams without curves, without shading.
Type (2)	Line diagrams without curves, with regular shading.
Type (3)	Line diagrams without curves, with irregular shading.
Type (4)	Line diagrams with curves, without shading.
Type (5)	Line diagrams with curves, with regular shading.
Type (6)	Line diagrams with curves, with irregular shading.
Type (7)	Photographs.

This section contains a random selection of diagrams from several journals these are used primarily to show the range of pictorial information that has to be considered[7]. Figures 1 to 4 below, all lie within category (1) because they are essentially line diagrams with curves and no shading. This is obviously so with figures 1 to 3, figure 4 appears more complex only in that the line segments are shorter, it is however a borderline case and might perhaps be better represented by one or more spline functions.

I have included two examples of the type (2), (i.e. no curves but with regular shading), in figures 5 and 6. Figure 5 exhibits regular shading in the GKS sense of defining a pattern with which to fill areas bounded by irregular polygons. The same is true of figure 6 in which certain boundaries of this excavation map appear curved from a distance but were in fact represented by straight line segments.

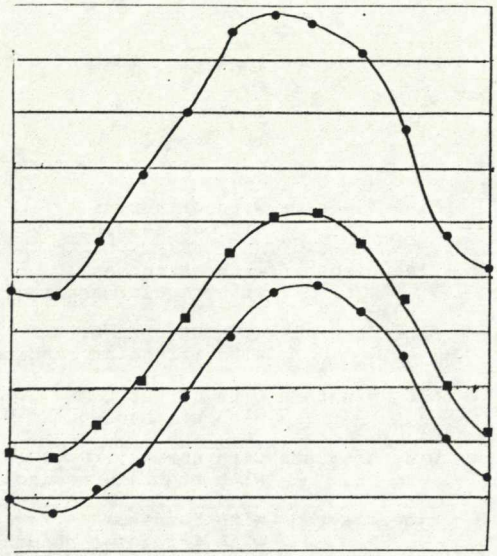


Figure 1

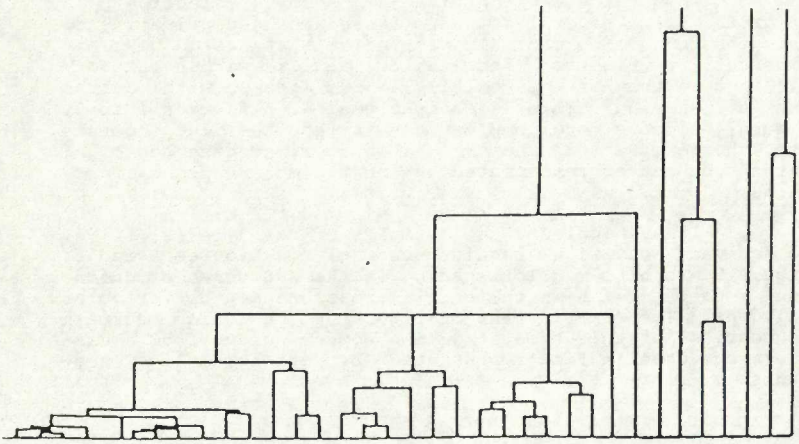


Figure 2

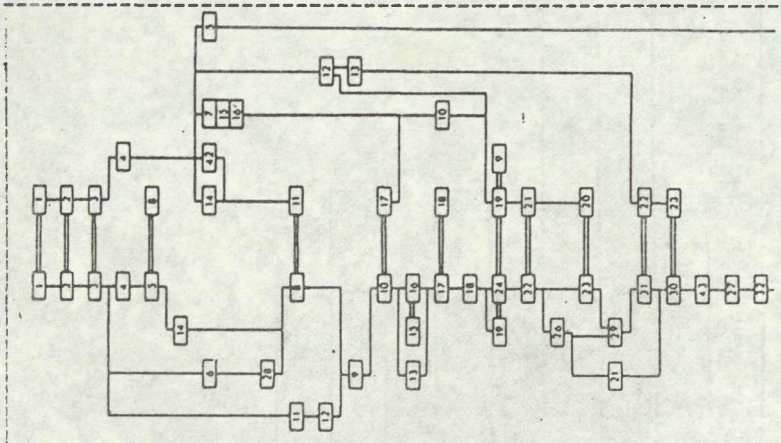


Figure 3

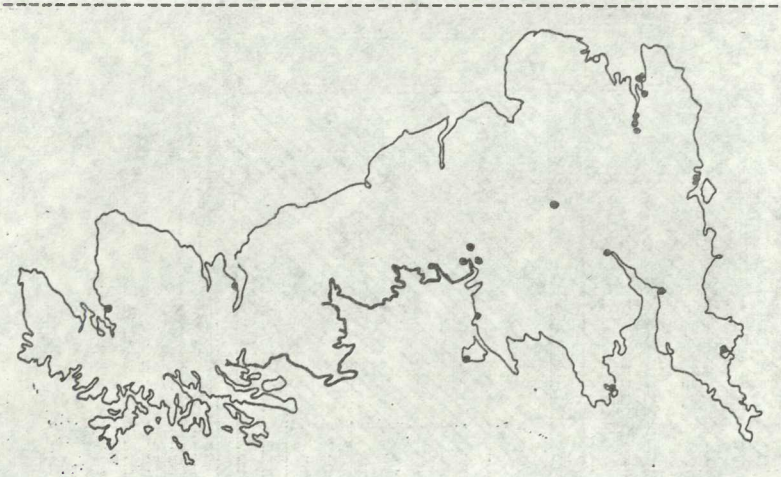


Figure 4

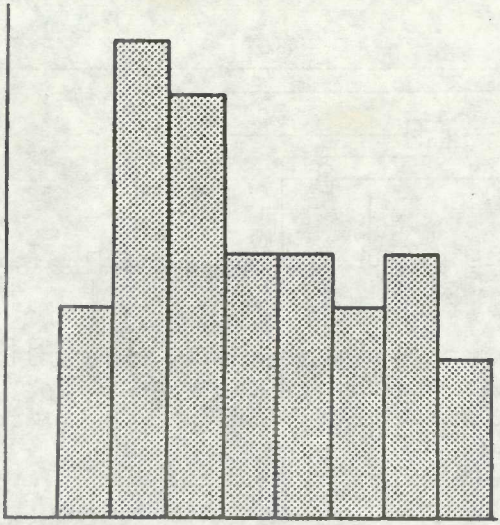


Figure 5

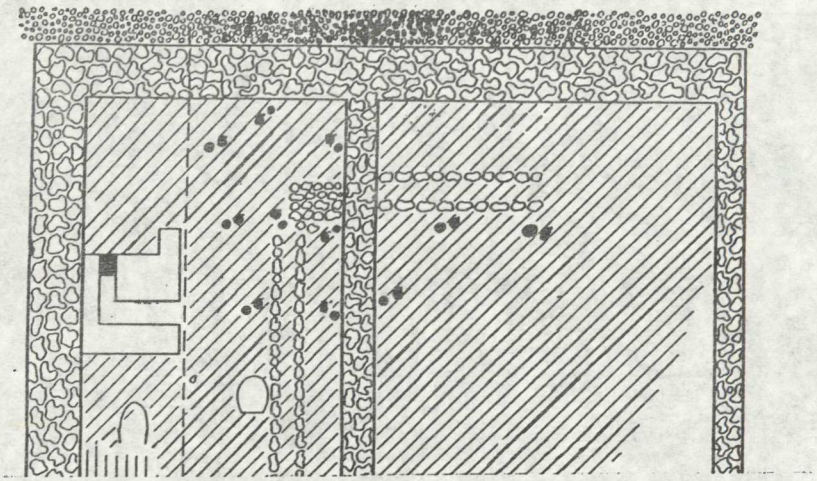


Figure 6

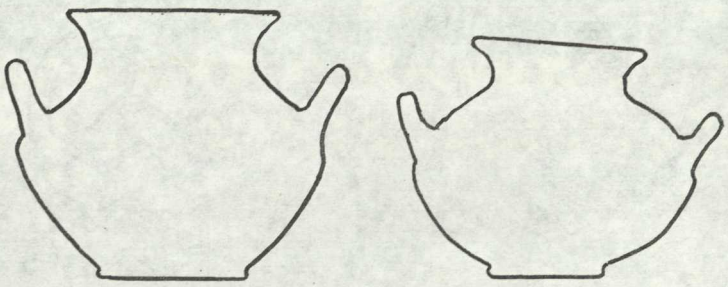


Figure 7.

Type (3) has been shelved as I have been unable to find any suitably convincing examples but no doubt they do exist.

Figure 7 is an example of type 4, (i.e. curves but with no shading in the sense mentioned above). There is perhaps little to say here, except to note that those diagrams might be represented in GKS terms as splines within the generalized drawing primitive.

Figures 8 and 9 overleaf, contain examples of what we have labelled type (5). They are line diagrams but contain general curves and uniform density shading. Although shaded areas in figure 9 appear somewhat complicated, it is uniform in the sense that it could be represented by the GKS area-fill as explained in connection with category (2). In this case however, the actual mechanism for shading would either be more complicated or take longer because the areas are bounded by curves rather than straight lines.

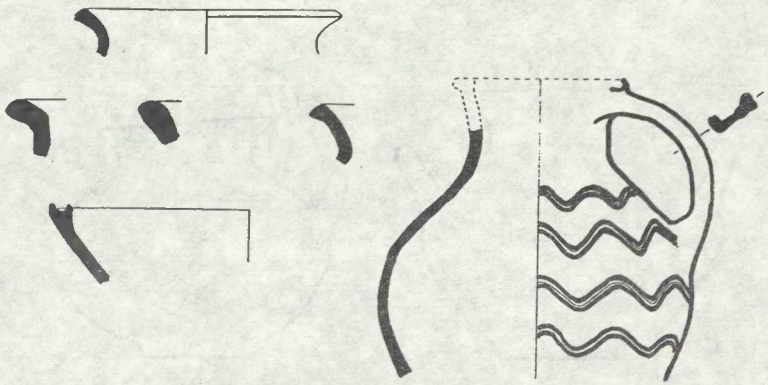


Figure 8

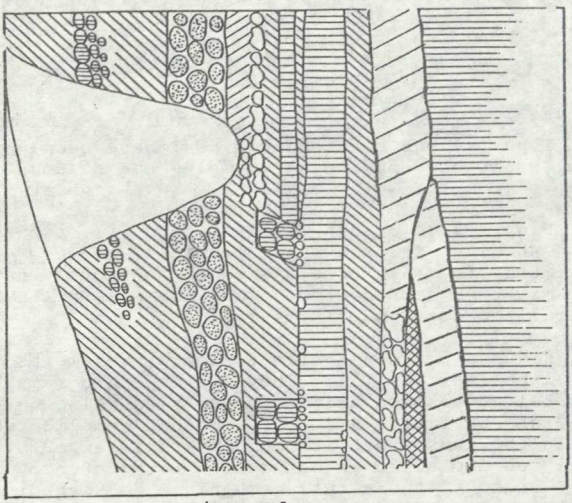


Figure 9

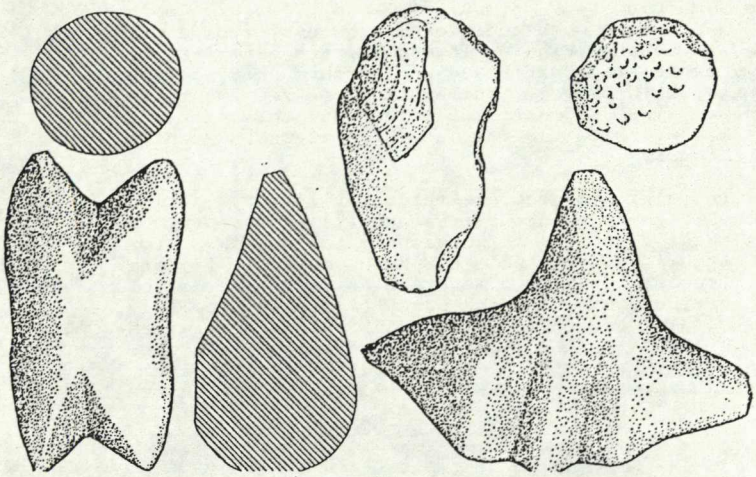


Figure 10

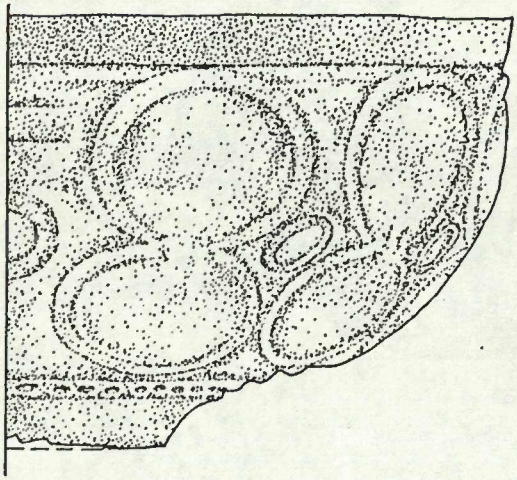


Figure 11

The remaining two groups, (6 and 7), are more interesting and could pose additional problems, from a GKS point of view at least. Figures 10 and 11, are examples of category (6), exhibiting curves and non-uniform shading. As an aside, these would be very cumbersome to represent in terms of pure line segments but more about this shortly. Figure 12 is a copy, albeit a bad one, of a black and white photograph (category 7).

Before talking of representation it is worth noting the relationship between groups (6) and (7). Category (6) will contain material which is obviously a drawing, for example see figure 10. On the other hand, category (7) are photographs, (see figure 12). Nevertheless, in terms of intensity distribution across the page, figure 10 type (6), is almost as complex as figure 12 type (7). Indeed detailed drawings, may be almost or just as complicated as photographs. Consequently, in terms of how to represent them, the distinction between groups (6) and (7) may become fuzzy.

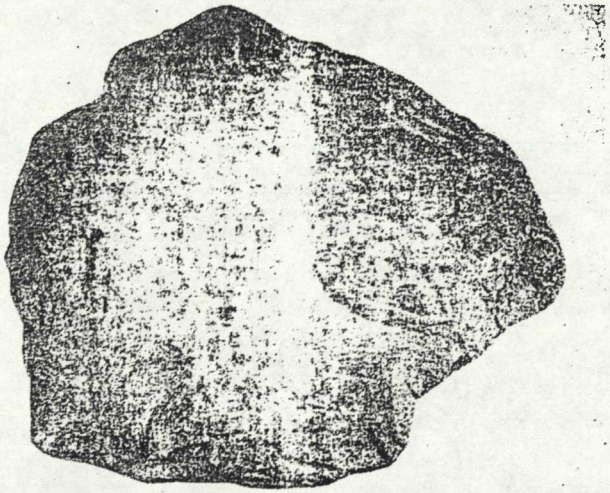


Figure 12

Possible Methods for Representing Diagrams from the Classification

I have described the loose guide that has been used to categorize diagrams into different types, from the simplest at type 1, to what I have called the most complicated at type (7). It has been mentioned, that this grouping was drawn up somewhat arbitrarily in terms of how one might represent the diagrams in a GKS biased environment. It should therefore be remembered, that this definition of complexity would be different as seen from a different perspective. With the above provisos in mind, this section will briefly describe ways of representing figures 1 to 12.

Type (1) and "polyline" in GKS

Diagrams in figures 1 to 3 are relatively simple straight line constructions. It would therefore make sense to represent them in terms of the discontinuities or vertices they contain, with a two dimensional co-ordinate pair for each, relative to some origin and at a specified scale. In GKS terminology, each unbroken, (not in the dashed sense), line in the picture would be represented by a "polyline" primitive with scale and origin being determined by the selected window. The diagrammatic representation could be stored symbolically for example, and reproduced quite quickly by software designed to interpret the symbols and call appropriate GKS primitives to reproduce the diagram. This should be qualified by noting that if the original diagram happens to contain a high degree of stylization, e.g. it is constructed solely say of square boxes, a more compact representation will be afforded by describing the positions of the boxes rather than the co-ordinates of every vertex. There are many examples of this in both category (1) and (2) diagrams, it is a consequence of the type of information they usually represent.

Figure 4, which is also of type (1) could be represented in this manner but the number of points in the polyline would be large, in the thousands, (i.e. the coastline in this map looks curved because it is constructed of many short line segments). This will only lead to problems if space is at a premium or if one is planning to transmit our "digital map" along a slow transmission line.

Type (2) and "area-fill" in GKS

Figures 5, and 6 cited as examples of type (2) diagrams could be dealt with in much the same fashion. In this case however, to specify the picture in terms of GKS, requires the, "area fill" primitive, (which looks very similar to polyline). Thus the diagrams in this group could be stored as aggregates of one or more polyline and fill area functions. The particular shading pattern that we wish to use for a given region can easily be described by a single parameter in the corresponding fill area primitive. Figure 6 could be represented by a more complicated example of the same method. In this case however, the shading pattern is more intricate, presumably to provide information about what it is intended to represent.

Type (4) and curve representation

I have not included any examples from the type 3 diagram. Figure 7 is of type (4), curves, but no shading, and could be represented by collections of GKS polyline primitives in the same manner as described for type (1) diagrams. In this case however, like figure 4 in type (1), the number of points along one of the curves would have to be large for the lines to appear smooth. This is especially so for the example in figure 7.

It would make more sense in terms of storage space to represent the constituent curves in memory mathematically. Thus one could represent figure 7 as a set of cubic splines[2], Bezier, or B-spline curves[3]. Doing this would require a relatively small number of defining co-ordinates to be stored. These co-ordinates, generally called, "control points" are used by the mathematical function or functions to compute a much larger set of points that represent the actual curves we wish to display. The price one has to pay for recovering these from control points is in the computation time required by the spline algorithm. Depending on the installation, the algorithm used, and of course complexity of the picture, this can vary from fractions of a second to hours. Clearly a trade off exists between the amount of information one has to store and the work required in order to retrieve the original picture. In GKS a spline function could be included as a generalized drawing primitive. It would take the control points, calculate the many actual co-ordinates and perhaps hand these to the familiar

polyline primitive to produce the shapes for display.

Type (5) and the GKS area fill

Figures 8 and 9 both lie within type (5), and may be treated in a manner analogous to that described for type 4, noting that in this case, uniform shading has to be included. If the shapes are stored in terms of control points for spline functions as mentioned above, the resulting calculated co-ordinates representing boundaries of shaded areas will be handed to the GKS area-fill function along with a specified filling or shading pattern. It is also worth noting here, that in addition to the calculation time required by the spline function, depending on the hardware, it may also be a considerable task for GKS to implement the actual shading operation and so this may also take a while.

Types (6) and (7) and Bit mapped images.

I shall again consider types (6) and (7) together. Figures 10 and 11, are of type (6), in both there are instances of variable density shading. In such circumstances, it is not easy to locate a polygon that defines the boundary of any area to be shaded. In these examples most of the important detail is implicit within the shading alone, i.e. certain important features are not bound by lines at all. How in the GKS sense could this be represented?

The photograph, (figure 12), that has been used as an example of type (7) illustrates more poignantly the same problem. In this case all that exists is an intensity distribution across the surface of the picture. Clearly, these types of illustration are not ideally represented in terms of GKS area-fill and polyline primitives.

The general method adopted in cases such as these is to divide the picture into square-ish cells or pixels. Each pixel will then have an associated level of gray determined by where on the picture it lies, this is usually represented as an integer. In the case of a colour photograph, one method assigns every pixel an associated value for each of the three prime colours, red blue and green, these are then combined to give some composite colour.

Obviously, the more intensity/colour values one allows, and the smaller the pixels, the higher will be the quality of the picture. The price for this is, a greater amount of information that has to be stored, manipulated and transported.

GKS provides for this general type of representation with the so called "cell array" primitive, this allows the programmer to define virtual pixels so that he may map these to an actual display in the manner desired.

Many microcomputers feature a bit mapped display, in which every physical pixel of the screen is mapped via hardware and software to a particular memory location. In order to change the colour of a given pixel in such a system, it is only necessary to write an appropriate digit to the corresponding memory location and the change is effected practically instantly. Thus, for a picture that consists solely of intensity values, it is in principle just a matter of setting up the memory map, (or display buffer as it is sometimes called), with those values in some way and the picture will appear very quickly. Since no calculations or interpretations have to be carried out, and the technique is able to represent all types of picture, it looks quite an attractive method. However, as the following example will show, a bit mapped image contains a lot of data, (but not necessarily information), which occupies considerable space. This is a big disadvantage if one intends to transmit such an image between installations A and B, or if in our journal system library there are an appreciable number of photographs to be archived. On the other hand, there is the advantage that once all image data has arrived at the user's microcomputer, provided it has not been compressed, little or no processing need be done to display it.

A simple bit map example

A simple example will put this in perspective, consider a picture having a spacial resolution of 512 by 512 pixels. Each pixel has an associated 256 different possible levels represented as an 8 bit byte. Depending on size of course, this will have about the same image quality as a newspaper photograph. Thus the picture has a total of, 262,144 pixels, with eight bits per pixel that means it will need, 2,097,152 digital data bits. In terms of computer graphics, this is regarded as medium resolution and is in no way exotic. The IBM PCAT is a popular personal computer, when equipped with the professional graphics hardware, it is capable of providing similar resolution, at 640 by 480

pixels with 8 bits per pixel, (incidently, this is slightly higher resolution than our example with a total of 307,200 pixels).

In terms of storage it is interesting to compare the space occupied by the image with that provided by familiar storage devices. With the advent of IBM's PC, the 360K byte five and a quarter inch diskette became a standard. Our typical image above, would occupy almost three quarters of these disks. In fact we would only be able to fit 38 such images on a 10Mbyte IBM PC winchester and have very little space for any accompanying text and nothing left for programs or operating systems! It can be seen that this would provide a very limited electronic photograph album, so how could we build a journal system that contains potentially many photographs and diagrams hidden among text?

To make these figures tangible, a page of densely typed A4 text contains approximately 4680 characters, (remember, most pages of A4 will contain a great deal less). Thus, 1Mbyte will hold about 214 such pages, and our image would occupy the same space as about 56 pages of A4.

It would therefore appear that not only does a picture tell the story of a thousand words, it also occupies the space of about 33 thousand of them. Obviously, there are techniques for compressing the amount of information that must be stored in order to represent a picture. Again, the applicability of these techniques depends on the diagram. Just as the method for diagram representation has to be chosen to suit the particular picture, so it is with compression methods. It is possible for a compression technique to reduce the amount of storage required for one type of diagram and actually increase it for another to which it is not suited. Again the price one pays for compression is the overhead in terms of the amount of processing the computer must perform in order to recover the original image.

Bit Map and Vector representation

a brief Comparison

I would like to summarize this section by making a few comparisons between the different methods for diagram/picture representation in terms of storage space and type of diagram. Figure 2 of type (1), represents one of the simpler examples we are likely to encounter or want to reproduce in an electronic journal.

Consider 2 extreme cases, representation as a set of GKS primitives by vectors, and also as a bit map. It must be emphasized here that this diagram has been chosen purely to illustrate its pictorial content and that it is of type (1).

The choice of example is somewhat unfortunate since this is a diagram that represents an actual data structure, the Harris matrix, but the Harris matrix is probably also one of the most compact ways of representing the diagram. This however will not affect the discussion. It must be appreciated that figure 2 is a special case, in general many other examples exist of diagrams that represent concepts rather than a particular data structure, and these have to be represented using more generally applicable methods. In reality figure 2 conforms to one particular type of subject dependent stylization which could be incorporated into the software machinery of an electronic journal, however, doing this would lower it's general usefulness to other disciplines. With these points in mind, let us consider two methods of representing figure 2.

Figure 2 contains 659 vertices, these are spread among 132 continuous lines constructed of straight segments, (the connecting lines), plus another 87 continuous lines that make up the boxes. Note we have considered the boxes to be separate lines with four vertices each.

From a GKS point of view, this diagram could easily be represented as 219 polyline primitives, each of which corresponds to a particular line. We may estimate the space requirements of the diagram by assuming that we store the co-ordinates for each vertex, along with the additional information required to determine what vertices lie within which lines, i.e. the polylines. Assuming a display, such as the professional hardware option available for the IBM PC-AT with a resolution of 640 by 480 pixels, it is most convenient to use two 8 bit bytes for each number in this range. Each co-ordinate will be a pair of integers and thus occupy 4 bytes of memory. Each of the separate polylines can be introduced as a single character. Consequently, figure 2 may be represented as 219 bytes, (for the polylines), plus 4 x 659 bytes for the co-ordinates of all the vertices in the diagram. The diagrammatical information alone neglecting the labels in figure 2 would then occupy a space of 2855 bytes plus any housekeeping information we might have to include. There are 87 separate labels in the diagram, which collectively contain 156 characters. Since each character is represented by one byte, the additional storage for these will be 156 bytes plus the 87 x 4 bytes of co-ordinate pairs required to position them on the display.

It is also necessary to identify these 87 separate character strings as text, one character for each will suffice giving another 87 bytes. This adds up to an additional 591 bytes, bringing the total storage requirement of this diagram to 3,446 bytes.

The same diagram may of course also be represented as a bit map, the method discussed above. Doing this would be a very wasteful exercise and occupy 38,400 bytes in monochrome or 307,200 bytes if we include colour on a non compressed bit map. At very best, the pure bit map will occupy more than ten times the volume of the pure GKS description. Clearly, in terms of space this type of diagram is much better represented in terms of its constituent co-ordinates. On the other hand, it would not be practical to represent an illustration such as figure 11 by co-ordinates in GKS in this manner, and there is no choice but revert to bit mapping with compression techniques. In this section, I have attempted to highlight the contrasts between two very different methods as applied to two very different types of picture. The vector based method using GKS has a certain range of applicability, the bit map technique is completely general and may be used to represent all types of diagram listed but can be hungry in terms of storage. The method has to be chosen carefully, to suit the particular circumstance, this is especially so in the case of an electronic journal. Not only are we interested in image fidelity, but also in the time it takes a system to reproduce that image, and the space it occupies. The latter two points are particularly important constraints, the first is imposed by how we would want to use an electronic journal, the second will determine how many pictures we can store on a given device or library.

Future intentions

Some of the basic considerations have been outlined. The current study will compare reproduction speeds and storage requirements for different methods of representing various diagrams.

To support this, methods for internal representation of graphics of different types will be investigated. Particular attention will be paid to compression techniques for both gray scale i.e. bit mapped images as well as line diagrams represented as vectors.

A major objective will be to minimize the amount of graphical information that has to be exchanged between a

terminal and host computer or microcomputer and host. One can imagine someone browsing through an article using an electronic journal. In general the article will contain text and diagrams which must appear quickly because our reader may be flicking through the "pages" or screens of information. If the document material is not located within the user's device, it makes sense to transport the minimum amount of information as this will ensure transmission time is as low as it can be on a given communications channel.

One technique which will be studied at least for bit map images,will be to transmit or display degraded versions. Degraded in the sense of having reduced resolution, (perhaps in terms of colour and number of pixels), so that pictures will appear quickly but at the same time remaining recognizable. A previous section has illustrated just how much data there is in an average resolution image. There is evidence to suggest[6] that provided the presentation is correct, users are able to identify images as either, of interest, or not of interest to them, after having received only 2% of the original image data.

If it is possible in many cases to get away with transmitting only a small fraction of the available picture data, a valuable saving of time is achieved. Thus in a "flicking mode", the system could support browsing by sending degraded images every time a picture is encountered. When the user wishes, he may stop on a page and interactively upgrade the diagram to whatever level he desires, up to the resolution of the original image if necessary. This will ensure that only the information actually required gets transmitted so less time is likely to be wasted displaying irrelevant pictures one would rather skip past.

Brief notes about CD-ROM Storage

Having spoken about storage requirements of computer graphics, and the archiving problems this is likely to cause, it is perhaps appropriate to finish with a few remarks about a relatively new storage device the CD-ROM. A previous section has indicated just how intensive on space graphical information can be. In particular, as the quality or complexity of the picture is increased from say pure line diagrams to photographs, the space requirements rise by orders of magnitude. This can cause enough problems at the display end, because ideally, one requires sufficient video memory to represent an image to its full resolution.

However, storage problems are compounded enormously by the need to archive many such images in some computer readable form. Examples have been given comparing the size of non-compressed bit map images with the IBM PC standard 360K Byte floppy diskette. An electronic journal system that accommodates pictures as well as text will have to archive this information on a medium that is convenient to access.

There are a wide range of reliable magnetic media available for storing data. These range from tape cassettes which essentially provide serial access to the information stored, to large capacity winchester disks that support random access. Most applications in the computing world require a storage medium to allow reading, erasing, and also writing of data. There are however certain storage devices that offer potentially very large capacities and random access, but are read only. One by now famous example of this is the CD-ROM, (the acronym stands for Compact Disc Read Only Memory). The data is stored optically and read by detecting the changes in intensity of a laser beam reflected from the "surface" of the disk. In practice the actual "active" layer is sandwiched between transparent plastic for protection.

It is anticipated the disks will have a high degree of permanence, actual physical wear during reading is likely to be negligible because the read head does not touch the surface. The disks are also likely to be more robust than magnetic ones, being less critical of their environment. In addition, because the active surface on which the data is stored lies below a layer of plastic, the laser is not focused on any surface in contact with the "outside world" but on a small distance below it. This affords an extra degree of durability in that if the disk becomes scratched during handling the data will still be readable. Thus in an office environment, in which they are continually being taken out of, and inserted into drives, CD-ROMs are likely to last in excess of ten years without irrecoverable reading errors. However, in a library environment where the disks are archived lifetimes likely to be almost limitless.

One big advantage of the CD-ROM is its capacity, storage volumes as large as 600M Bytes have been quoted. This is roughly equivalent to a space occupied by the character information on 200,000 pages of A4 text. This figure would probably be reduced somewhat by the need to store directory information on the disk and various other pieces of housekeeping data. Even so, assuming one can only use an effective 400M bytes it can be seen that a four and three quarter inch CD-ROM has enormously more capacity than our standard five and a quarter inch floppy diskettes. These

devices are as random access as winchesters, and from the software point of view could be read in much the same manner.

An example will attach more meaning to these figures. Consider the information content of a standard text, I have chosen, "Report on the Excavations at Wroxeter, 1923-1927", by D.Atkinson. The main body of text consists of 370 pages, plus 17 pages of index, 4 pages of contents information, and another 4 pages listing figures and plates. There are also 50 diagrams/drawings plus 73 photographs. Wanting to over, rather than underestimate the information this contains, I have assumed each of the 370 pages of main text are full, (in fact there are gaps containing diagrams). From an approximate, but generous estimate, the contents section has 125 lines with 50 characters per line, the figures list has 180 lines with the same number of characters on a line, and the index is 17 pages long, each page contains about 58 lines of around 64 characters. Itemizing these figures there are:-

contents	125 lines, 50 char's/line
figures-list	180 lines 50 char's/line
index	17 pages, 58 lines/page, 64 char's/line
Main-text	370 pages, 39 lines/page, 60 char's/line

Adding this up gives an approximate figure of 944,154 characters in the whole of Atkinson. In order to quickly allow for the space occupied by the diagrams and photographs assume each is represented by a non-compressed bit-map. Following the discussion above, a very acceptable resolution might be 512 x 512 pixels with 256 possible shades of gray for each. Every diagram and photograph will therefore occupy 262,144 bytes of space and collectively there are 123 of them requiring 32,243,712 bytes. Overestimating its storage requirements, this volume of Atkinson could be stored within about 34 M bytes. Even without compression ten volumes would easily fit on a CD-ROM, (600M bytes) with plenty of room to spare for indexing data, and any related software the suppliers might consider it necessary to provide!

From the computer applications point of view, CD-ROMs have one very big handicap, as the name implies they are read only. A serious problem if one requires a medium whose contents are likely to change over short periods. There are optical technologies that extend this slightly

such as the so called WORM, (for write once read many), and others under development. Such devices can only be written once and so have limited application. However in the short term, CD-ROMs lend themselves to other uses. In an electronic journal system that needs to archive items purely for reading, the read only character is not a handicap and its permanence and capacity, (especially if we want to store pictures), are points very strongly in its favour. In fact, a CD-ROM would be useful in any circumstance that requires large volumes of reasonably static data to be held locally for reference purposes. Currently, work is commencing at Hatfield Polytechnic looking at uses for these devices in the context of electronic journal systems.

It is worth noting that other optical storage technologies exist, such as the optical video disc. These also have potential for storing large volumes of information but do so in analogue form. Unlike a CD-ROM, the video disc cannot be read directly using a computer, although a computer can be used to access the various images for subsequent display on a monitor. The main difference to note is that information on the video disc cannot be directly processed whereas the information on a CD-ROM can.

Referrence list

1 .....Introduction to the Graphical Kernel System
G.K.S. .
F.R.A.Hopgood, D.A.Duce, J.R.Gallop, and
D.C.Sutcliffe.
(Aric Studios in Data Processing, No 19)
Acedemic Press.

2 .....Applied Numerical Analysis, P90.
C.F.Gerald.
Addison Wesley series in Mathematics.

3 .....Mathematical Elements for Computer Graphics.
D.F.Rogers, and J.A.Adams.
McGraw-Hill.

4 .....A proposal for the investigation of
"THE PROVISION OF A GRAPHICS FACILITY FOR
COMPUTER-BASED ELECTRONIC JOURNALS"
W.P.Dodd and Mrs S.LaflinBarker,
Centre for Computing and Computer Science,
University of Birmingham.

5 .....Experiments With Reading Articles On-Screen
D.J.Pullinger

6 .....Interactive Image Query System using
Progressive Transmission.
F.S.Hill,Jr, S.Walker,Jr, F.Gao.
P323 Computer Graphics, Vol' 17, No' 3 July 83.

7.....The diagrams in figures 1 to 12 are examples
from a number of archaeological journals.
They have been used only to illustrate the
range of pictorial information likely to occur
in the subject.