Message orientated interactive graphics

D. J. Grover*

3 Ducketts Mead, Roydon, Essex

The paper discusses the implications of interactive work with a remote stand-alone graphics display using dynamic storage. The display was designed to minimise cost consistent with sensible capability. ISO compatible message formats for communicating between the display and a central computer, via a telephone line, are considered in relation to the means of defining and restricting the message to permit a fast response and economy of line time.

(Received June 1971)

1. Introduction

The paper considers the problem of providing a graphic display interface to a timesharing computer over a telephone line. Previous systems have used a satellite computer for such features as display refresh, line protocol, data concentration, interrupt analysis, etc. The bibliography lists some of the systems.

The particular problem addressed by this paper is that of organising the system and procedures of a remote graphics hardware terminal to permit useful and economic interaction with a central processor without entailing the cost of a satellite computer. By this means the terminal cost is reduced below £2,000.

A limited degree of interaction over a telephone line is possible using a stand-alone storage tube display, however interaction can be enhanced if a dynamic store is incorporated at the terminal and the system so designed as to utilise the often neglected interaction capability of the Operator/Display keyboard interface.

The graphic display under discussion resembles an alphanumerics Video display unit in that it incorporates a recirculating store to control hardware character and vector generators. Features such as addressing and editing, etc., are actioned at the refresh rate as the data is presented in the 'window' of the recirculating store.

The operations are of a kind where an action or a state is operable, on subsequent serial data, until cancelled. Accordingly the logic hardware required to perform these simple operations is minimal. The operations, though simple, can be exploited with effect as will be described in the paper.

The serial nature of operations is illustrated by considering the sequence of events for addressing a message to store. The message from a telephone line enters the buffer queue; on recognising File Separator the system waits for correlation between the next word (label) in the message queue and the stored data appearing serially in the store window; on achieving correlation the system commences overwriting the data in store until stopped by an EOM (end of message) code or another file separator. Such operations are simple to implement at a speed compatible with a telephone line.

The cost of hardware to implement these systems features is much smaller than would be achieved by incorporating a minicomputer.

The facilities which have been incorporated are those which can be implemented at low cost. The procedures which these low cost facilities permit may differ in some cases from those used in conventional local interactive graphics due partly to the constraints of the low cost hardware and partly to the limited bandwidth of a telephone line. Procedures are described which overcome many of the difficulties.

Facilities which are discussed are the means of limiting and identifying messages to reduce communications time; the use of hardware garbage collection and windowing to reduce the degree of servicing required from the central computer. Simple light pen and menu facilities are also mentioned.

The system was designed within the framework of an ISO compatible 7-bit character code set to ensure conformity to accepted communications standards. Accordingly the calculations of communications times are based on vectors, points and labels consisting of two characters, i.e. a 16-bit word inclusive of parity bits. In the discussion of message formats message header characters are ignored so that only the data occurring within the STX-ETX envelope† are considered.

Message formats will be described using the following abbreviations

Abbreviation	*Meaning*	*Properties*
$\underline{P}$	Absolute Position	It permits rapid spatial positioning. The history of the store, prior to *P*, does not affect the visual position of data following *P*.
$\underline{V}$	Relative vector (Visible)	A subpicture may be duplicated at different positions. The position is determined by the history of the store.
$\underline{U}$	Relative vector (Unseen)	
α	Alphanumeric	Condensed code for controlling a hardware character generator.
$\underline{L}$	Label	Identifies a sequence of data. It can permit topological and hierarchical association between sets of data.
$\underline{A}$	Associative address	The address is independent of store position and is not modified by garbage collection. It can aid the selection of pertinent data from a stream of general data.
$\underline{F}$	File Seperator	Used in the present context to indicate that the following data is to be sent to a new address which is specified by the word following $\underline{F}$.

*Formerly with Cossor Electronics Ltd., now with the National Research Development Corporation.
†i.e. data between 'Start of Text' and the 'End of Text' control codes.

Abbreviation	*Meaning*	*Properties*
E	End of Medium	Used in the present context to indicate the end of a file or sequence of data. An outcome of this being that the remainder of an overwritten file, or sequence of data in store, is erased.
S	Storage Space generator	The converse of garbage collection for generating space in store between existing data.

The paper now discusses the advantages of associative addressing and the ways in which it can be implemented. The implications of associative addressing with regard to security are considered together with some means of retrieval from corrupted data. The use of a form of hardware garbage collection for conserving store with the minimum of computer and communications line usage is described, savings are also apparent in the methods of communicating with the CPU. The hardware window can be used to permit local storage of data which can be called on to the display screen with minimal commands; it is shown that the facility allows animation over a low speed telephone line. A method of implementing a low cost recirculating store as an offline menu is discussed together with its additional uses for security aspects.

2. Associative addressing

The relative advantages of position and associative addressing are dependent on the application and system organisation, but the advantages of associative addressing for remote graphics are sufficient to warrant concentrating the discussion on this mode of operation. Associative addressing, though desirable for computer systems, often carries an impossible overhead. However the interfacing of a low speed communications line to a recirculating store with serial access (such as are economic for use on remote refreshed displays) permits associative addressing in as fast a manner as position addressing and with some hardware economy. The addressing is achieved simply by observing the data as it is presented in the 'window' of a recirculating store so that on achieving correlation, between the message address characters and the equivalent characters, the system can commence entering data at that store position. The position in store at which action is occurring is identified by a cursor. On entering a character into store the cursor is incremented by one store position so that the next character is written into the subsequent position. The description of an associative address could either be a code set which is reserved for that purpose only or by use of a modifier control code which identifies following characters as being an address. These forms of addressing require the entry of the address characters into store before a message can be addressed to it; accordingly the system requires a means of distinguishing between characters in a message to be entered into store for future addressing purposes and characters which define the address of the message. The use of a modifier control code permits more flexibility here since any code sets can then be used for addressing purposes. 'F' will be used as the modifier in the following discussion.

Given the flexibility of a modifier control code, one can now consider two types of associative address. Using the definitions of Section 1, $A::=\langle F\rangle\ \langle L\rangle/\langle F\rangle\ \langle P\rangle$. The first type ($A::=\langle F\rangle\ \langle L\rangle$), which is essential, permits unique addressing irrespective of the remaining data in store. For this purpose a reserved code set (or further modifier in store) must be used and will be referred to as a label since it has the further ability of uniquely identifying a set of data for interactive purposes. An example of a label might be one or more ISO lower case alphabetic character codes (if they are reserved for this purpose) where only upper case ISO character codes are used for displayed text. Procedures incorporating this facility are discussed in Appendix 1.

The second type of associative address may not permit uniqueness, particularly if it is a toponym or name associated with a place or spatial position. Such a situation would occur if absolute position were used as an address, i.e. $A::=\langle F\rangle\ \langle P\rangle$ the address is now associated with a part of the screen. It is pertinent here to distinguish between

1. Addressing a position on the screen.
2. Addressing data associated with a position on the screen.

In (1) new data may be written anywhere in store to appear visually at a point on the screen and appear coincident with data already there.

In (2) the new data may overwrite (and therefore modify or delete) the data in store which appears visually on the screen, from the point in question.

The situation is apparent in Appendix 4 **Figs. 2 and 3**. In Fig. 2 the different triangles have been placed at several different spatial positions. To maintain unique addressing the sequence at P_0 must be removed before placing a second sequence at P_0. The situation in Fig. 3 is not amenable to this toponymic addressing since only two positions are used and duplication of address would occur to create an ambiguous situation. It is accordingly necessary to use unique addresses incorporating labels which at different times are associated with positions P_5 or P_6.

The addressing discussed so far has permitted addressing an identity (or label) and an absolute screen position (absolute point or absolute vector). The addressing of a relative vector (i.e. $A::=\langle F\rangle\ \langle V\rangle/\langle F\rangle\ \langle U\rangle$) permits us to address directional information, in fact the address is composed of both direction and magnitude. Such a facility is of use in animation, though the higher probability of duplication requires care in software. Hierarchical addressing is an aid in giving a unique address.

Hierarchical addressing is available if the system is constrained to look for a new address from its previous address. It is of use, when addressing, if a particular element, which appears in different sets, is required to have a similar address. The particular element of interest is then addressed by first addressing the set.

3. Message security

There are two aspects of security to consider. The first embraces the procedures used for line protocol for which various handshake procedures of varying degrees of security have been devised. The procedures handle those erroneous conditions which are spotted by error-checking systems monitoring character parity and longitudinal check characters in a message. They deal quite well with those situations where the complete store is created but they are not so secure when the store is addressed.

This paper is not concerned with the handshake procedures used to check and correct message errors, since these have been amply discussed elsewhere. However, it is concerned with the errors which are a function of the message content within the STX-ETX envelope (i.e. between the start and end of text.)

Good security in communication, while giving a useful guarantee of message validity, is not 100 per cent. The degree of security being a compromise between that desired and the cost of software and hardware. Even supposing the message checking procedures were 100 per cent, the displayed information may be less valid if addressing is incorporated since an error in an address is liable to corrupt other parts of store from which there is normally no retrieval. The problem differs

depending on whether position or associative addressing is used. The system is more vulnerable to a position addressing error since if every position in store can be addressed then there is a high probability that an erroneous address will correspond to a valid address, so that the data in store at the erroneous address will be corrupted. Prohibition of writing to store in the event of a parity error on the address characters improves the situation but does not guarantee it, since more than one bit in a character may be subject to corruption. Security is further improved by duplicating the address characters and prohibiting writing if they differ.

The situation regarding associative addressing is dependent on the flexibility of the system and security tends to vary conversely with flexibility. The probability of corrupting other parts of store due to a misaddress is less when compared to position addressing, since in typical applications there are relatively fewer parts of the store which can be addressed, in fact those points of store which are addressable could be under the control of the applications program.

The facility to change an associative address, by addressing and overwriting it, presents a problem if an error is detected since the handshake procedures cannot cause an identical message to be regenerated if the address differs. It is discussed in Appendix 2.

The facility to address the cursor position in the absence of an address, i.e. $A::=\langle$Absence of File Separator$\rangle\langle$cursor position$\rangle$ is liable to error if an address is defined by a modifier character (e.g. file separator) which is corrupted. A solution to this problem is to reject the facility, if it is considered troublesome, by addressing the cursor to a vacant area at the end of store when concluding each message.

Solutions which are available for security in a corrupt environment (though not without additional cost) are

(*a*) In the event of an error in an addressed message the complete remote store is recreated. It is necessary for the CPU to maintain, or be able to generate, an up-to-date display file. The solution excludes an operational mode where remote data entry at the terminal is not necessarily known to the CPU.

(*b*) The inclusion of a buffer store, at the terminal, in which the message is verified before overwriting the main store. A maximum block length for messages, which is a fraction of the main store, permits economy of buffer store. The solution restricts the continuity of animation that would be possible in a direct (error free) system.

(*c*) Duplication of picture so that duplicated data is entered into store and displayed twice per refresh cycle. The result is to double the refresh rate for correct data. On the assumption that it is improbable that two similar but separate messages will collect identical errors, then those errors will be apparent on the screen if the corrupted data causes separate picture segments of lower brightness. The solution essentially halves the available store.

4. Garbage collection

The economy offered by message addressing ability in interactive work is enhanced by a form of hardware garbage collection at the remote terminal whereby the store is repacked so as to delete areas containing erased data and generate a continuous block of vacant store.

The latter technique does not correspond to the erudite garbage collection within software data structures where dependent functions are deleted in sympathy with the deletion of a parent unless the display file can be organised so that dependent functions are in the same sequence as the parent.

A simple form of garbage collection can be achieved by hardware in association with the labelling of sequences of data within the terminal store. A sequence will be defined as data which occurs between two successive labels in store A need for repacking the store arises if a sequence is modified or changed and the new sequence is of a different length to the previous one. The case of a new sequence of greater length than its predecessor can be catered for by space generator control codes which cause blocks of vacant storage to be generated within the existing sequence until it is of greater length than the new one. Subsequently the same procedure can be used to cater for any new sequence whether longer or shorter than its predecessor.

A message containing an end of sequence indication (e.g. the ISO 'End of Medium' control code) may now be entered into store; the EM code causing all data up to the next label (i.e. the end of the former sequence) to be converted to redundant codes which are recognised by the hardware, for garbage collection purposes, and deleted. The procedure is described in Appendix 3.

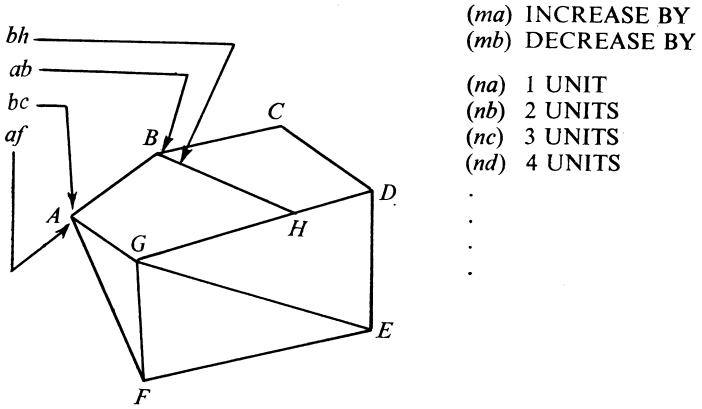
5. Reply to the central processor

The discussion so far has centred on messages addressed to the terminal. Replies to the Central Processor, in interactive work, are normally for identification of data with instructions for processing. Although there are times when a complete screen of data is to be entered, a reply must normally be short and easy to implement.

Various techniques are available for identifying data on a remote display screen. One technique commonly employed, of moving a spatial cursor to the vicinity of data on the screen and thereby advising the approximate X–Y coordinates to the computer, carries a software overhead to correlate the approximate X–Y data with the data nearest that position. This technique, usually employing a rolling ball, joystick or tablet might be termed a secondary mode of interaction.

Direct interaction with the data in store can be achieved using a light-pen or a store positioned cursor. Use of garbage collection at the terminal disqualifies the use of store position for interaction as is normal on computer refreshed displays. It is however feasible to copy the label describing a sequence, in which data is selected by the light-pen or cursor, and send it as the descriptor of a short message. The mode of operation is suitable for light-pen picking, and permits some economy in software if the correlation map can be dispensed with by carrying the label, as a name, through to the data structure; hierarchical considerations excepted. In the event that a picture has hierarchical structure various possibilities can be envisaged. A multi-character label would permit the individual characters to define sets and subsets for interpretation by software. A more ambitious scheme might incorporate hierarchical labels whose status is determined by modifier codes which the display hardware can act upon. We then become intimately involved in ergonomics and the operational procedures required to indicate association and status to both operator and software.

Fig. 1 illustrates a hypothetical example of interaction used to modify a network or structure. The problem is one where a topological description is useful and a two character label lends itself to this mode of operation. The label can be used to describe the nodes between which a branch occurs. Thus when assembling the display file the label *bh* would precede the vectors composing the branch between nodes *B* and *H*. Light-pen picking of this branch would then generate a short message containing the label *bh* as a descriptor to define the branch under consideration. Visual display of labels is subject to problems due to allocation of visual space and a degree of confusion when mixed up with vectors; they are only shown on this diagram to indicate their relative place in store. The labels *ma* and *mb* correlate with the instructions in the text, which would be used as light buttons. The numeric values may be extracted directly from their labels since when using the ISO code set, the five least significant bits of *a* are binary 1; those of



To increase branch '*AB*' by 3 units the message formats would be:

STX, *, *a*, *b*, ETX.
STX, *, *m*, *a*, ETX.
STX, *, *n*, *c*, ETX.

where '*' is a control code which defines the message as a light-pen format.

Fig. 1. Network example of light-pen interaction

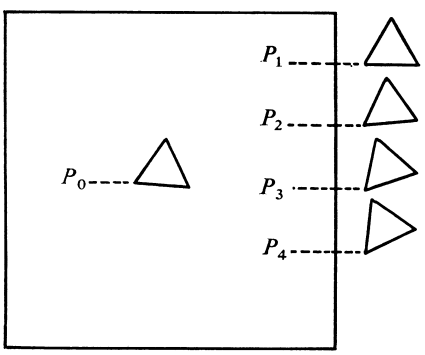


Fig. 2. Rotation at remote store

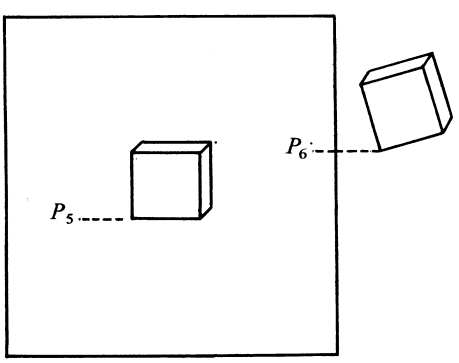


Fig. 3. Rotation from updated remote store

b are binary 2; etc. An alternative mode of operation is to send the sequence prefaced by its label. The use of absolute positioning, in the sequence, advises the actual *X*–*Y* coordinates of the data if necessary.

6. Animation and the use of a hardware window

The procedures for animation are discussed since they encompass and are more illustrative than the procedures for static applications.

The discussion will be divided into

(*a*) Procedures for displaying discrete blocks of store such as are required for the rotation of objects.

(*b*) Procedures for a quasi-continuous translation such as the scanning of waveforms.

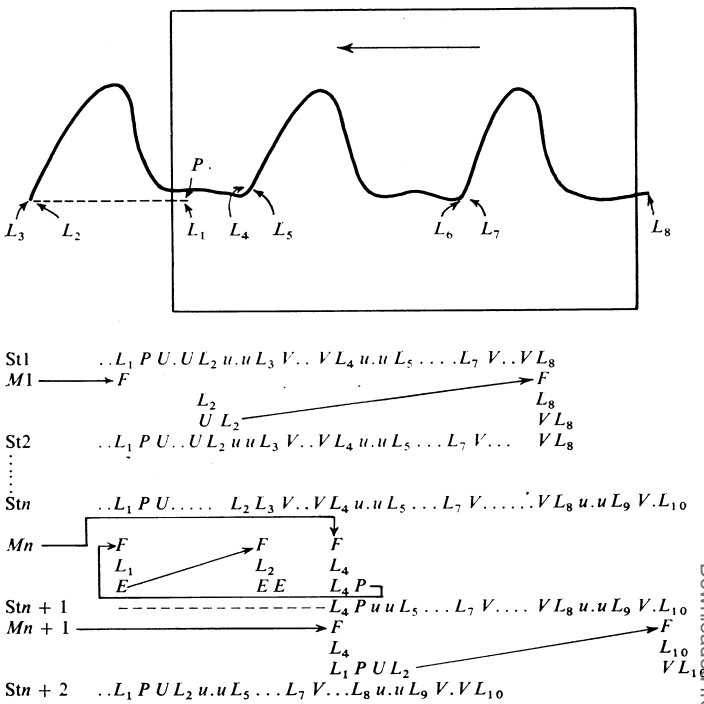


Fig. 4. Waveform translation from continuously updated remote store

6.1. *Rotation by discrete blocks*

Fig. 2 illustrates a procedure for rotating a figure with the minimum communications requirement, assuming there is adequate store. The hardware window provides a means of storing data in memory in its normal form but positioning it off-screen where it is not on view. A hardware window implies that wraparound does not occur immediately a vector is drawn off the screen.

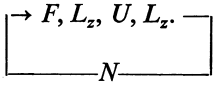
Restriction of absolute position to a point on the screen is consistent with optimum code description. The triangles of Fig. 2 are shown positioned off-screen by a blank vector from an absolute position at the edge. Rotation is now achieved by positioning each picture onto the screen in sequence.

The procedure is discussed in Appendix 4. It is shown that rotation at 10° intervals permits a complete rotation in 6 seconds using a 600 baud communications line. The size of picture is limited to a fraction of the display store size.

Fig. 3 illustrates the case of rotation of a detailed picture, where it is necessary for the display memory to be updated. The cube at P_5 is on display while the picture at P_6 is updated over the communications line; when complete the pictures at P_5 and P_6 are exchanged and the cycle repeated. The rate of rotation is necessarily much slower than in the previous example and is dependent on the picture content. The procedures are discussed in Appendix 5. It is shown that a rate of rotation of one increment per second permits 140 words of picture over a 2400 baud telephone line, (i.e. if an increment equals 10° then a complete rotation is 36 seconds).

6.2. *Waveform scanning*

The continuous scanning of waveforms, which vary with time, presents the problem of continuous translation, updating and deletion. The illustration in **Fig. 4** shows three cycles of a waveform which extend beyond the window on both sides. The right off-screen area is used for updating the waveform and the left off-screen area is used for translating the waveform and deleting spent data. Translation may be achieved by continually updating a translation vector or by repeating a translating sequence *N* times;

viz: 

Continuity of movement is achieved by updating the waveform alternately with translation. It is necessary for the translation procedure to be reset at intervals to conserve store. The procedures are discussed in Appendix 6. It is shown that translation at a rate of 1 cycle in 1–5 seconds is possible over a 2400 baud communications line. Rolling of text is the vertical equivalent of waveform scanning, but is simpler to implement due to the resetting action of carriage return.

7. Offline menu

The associative addressing described in this paper permits addresses to be created, in the display memory, from one data source ready to be addressed by another data source. The technique finds application in the use of a low cost recirculating tape store containing a menu of symbols or other data. Each symbol in menu, if prefaced by an associative address, will enter the display memory if, and only if, its address is found there. The assembly of a picture at a remote display could therefore be created with the minimum of CPU and communications line time by transmitting only the addresses of picture segments ready to be addressed by the recirculating menu offline. Such a facility is of further interest if the addresses can be created offline from a display keyboard, since, in addition to offline problem assembly, it permits an operator to define the display's address. The technique lends itself to privacy considerations since the operator can enter a particular format of addresses from tape, or any other medium, thus providing a key for unravelling a message sequence.

8. Conclusion

The procedures described permit message orientated interactive graphics from a remote station to one, or several, processing facilities over a telephone line in a manner which allows economy of line and computer time. Features can be incorporated which permit rapid change in a visual display which is controlled over a restricted line. Associative addressing and labelling give an efficient means of interaction, and are of further interest for offline problem assembly and privacy considerations.

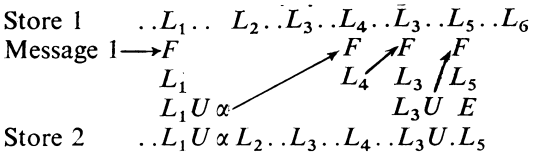
Acknowledgements

The paper is a revised version of a paper presented at Datafair 71 while the author was employed by Cossor Electronics Ltd. The system was implemented in hardware by M. Middleditch and D. Cross.

Appendix 1

A procedure for associative addressing of labels is illustrated below. Store 1 indicates the state of the area of store of interest prior to message 1. Store 2 indicates the state between messages 1 and 2, etc. A dot in the store area indicates any data such as vectors, alphanumerics, etc. which is not of direct interest in the discussion. Message 1 consists of the following string of data: $F L_1 L_1 U \alpha F L_4 F L_3 L_3 U F L_5 E$. The action is illustrated in relation to the store position which corresponds to the address. On receiving the associative address $F L_1$ the system awaits the appearance of the label L_1 and then commences writing into store from and including the address L_1 until the file seperator F is again seen. Hierarchical addressing is illustrated by the sequence $F L_4 F L_3$. The L_3 which is addressed is one which follows L_4. Deletion of the complete L_5 sequence is achieved by addressing L_5 and overwriting it with the End of medium code E, as discussed under garbage collection.

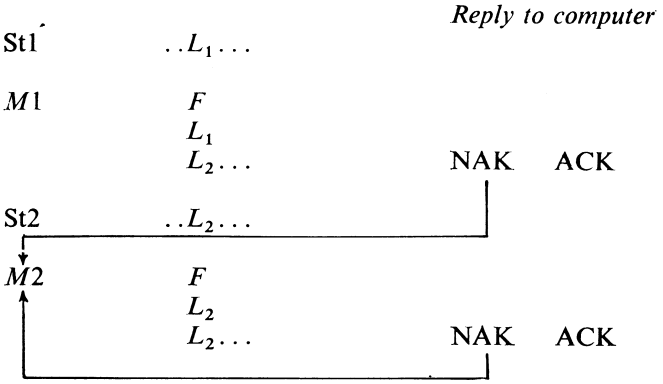
Updating of remote store, by message from computer



Appendix 2

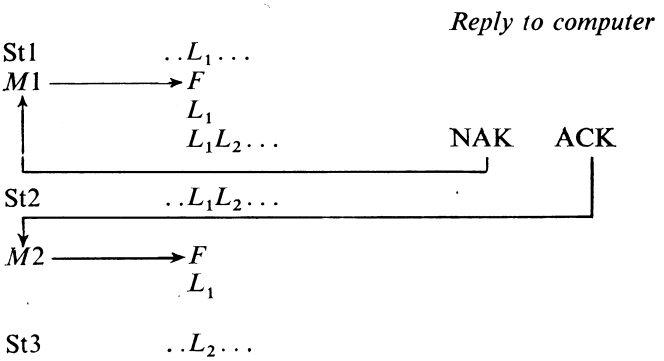
Direct change of address of remote store

The procedure for directly changing an associative address by overwriting it is shown. Since the address L_1 is no longer available a NAK response must cause L_2 to be addressed and requires the new message M_2.



Indirect change of address at remote store

An alternative procedure retains the existing address until the message is validated by an ACK response; so that $L_1 L_2$ are both contained in store 2. L_1 is then deleted by addressing it and overwriting with a null word (or EM) which is subject to garbage collection to yield store 3.



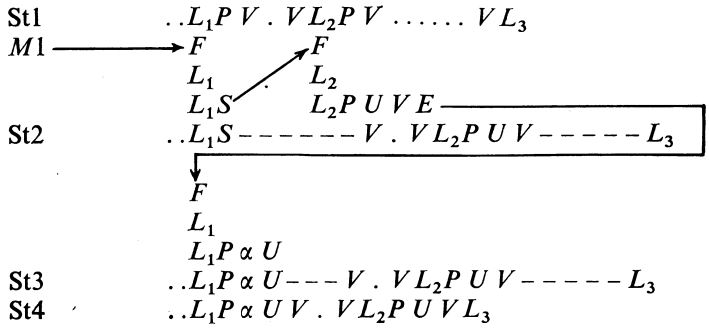
Appendix 3

The procedure for garbage collection and space generation is illustrated. Store 1 shows the area of store under discussion which contains two sequences labelled L_1 and L_2. It is required to insert further data into the L_1 sequence, thereby making it a longer sequence, and to replace the L_2 sequence with one which is known to be shorter. Message 1 addresses L_1 and enters a store space generator S, which in this example overwrites P and generates six vacant store positions including its own. Message 1 also addresses L_2 and overwrites three data positions before ending the sequence with E. The effect of E being to delete all data up to the next label L_3.

The time required to generate storage space or collect garbage is dependent on the hardware used and a slow repacking rate is consistent with hardware economy. Accordingly it may be necessary to delay readdressing L_1, until the required status of store 2 is achieved, and this may involve a seperate message. In the event that repacking is achieved within the message

period then Message 1 may continue and readdress L_1 to overwrite on S and the adjacent vacant store. The erasure of S causes the remaining vacant store to become vulnerable to garbage collection to reach the status of store 4.

Garbage collection and space generation in store

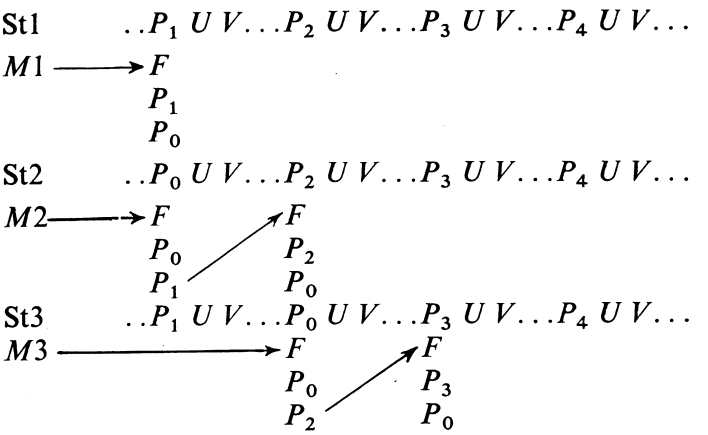


Appendix 4

Rotation by discrete blocks

Rotation of an object by calling memory resident discrete pictures onto the screen is illustrated in Fig. 2. The offscreen discrete pictures are stored from points $P_1\ P_2\ P_3 \ldots$. Store 1 represents the status for a blank screen. The problem is amenable to associative addressing of positions, accordingly message 1 addresses P_1 and overwrites it with P_0 which positions the 'P_1' triangle onto the screen. Message 2 addresses P_0 to reposition the picture to P_1 and then addresses P_2 etc.

Rotation at remote store



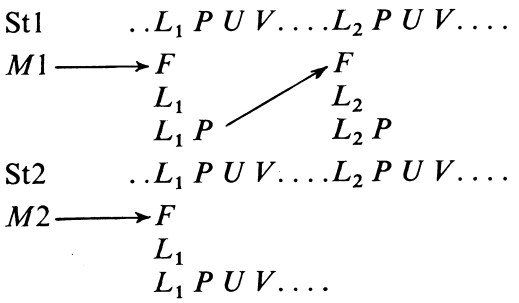
It is instructive to consider what can be achieved over a 2400 baud telephone line when using a 1 Kiloword store. If rotation by display at 10° intervals is considered acceptable then 36 pictures are required. Each picture with its positioning data is therefore limited to 27 words.

A 2400 baud line transmits 150 16-bit words/second. 6 words in the message are required to cater for one picture so that the communications line can control 25 pictures/second giving one complete rotation in under $1\frac{1}{2}$ seconds, which is rather fast and a 600 baud line would be adequate. The foregoing discussion assumes that $M_1\ M_2\ M_3$, etc. are incorporated in one continuous message over an error free line.

Fig. 3 illustrates the case of detailed picture rotation, for which the store would have to be updated. It is necessary for the pictures to be labelled to permit discrete addressing if only two screen positions are used. Store 1 shows the status for the L_1 cube on the screen. M_1 exchanges the cube positions and now while it is off the screen, L_1 is ready to be updated with the next view in the rotation sequence.

The rate of rotation is necessarily much slower than in the previous example and is dependent on the picture content. A picture of 140 words with addressing overheads requires 150 words, i.e. one second on a 2400 baud line. Rotation at 10° intervals therefore requires 36 seconds for a complete rotation; assuming a continuous message and an error free line.

Rotation from updated store



Appendix 5

Waveform scanning

Scanning of the waveform shown in Fig. 4 can be achieved by the message formats shown below (it is assumed that absolute positioning is not available offscreen). L_1 identifies the positioning sequence, for the waveform, consisting of P followed by blank vectors U in the $-X$ direction. L_2 identifies a vacant area of store consisting of zero length blank vectors (say, shown as 'u'. L_3 identifies vectors composing the first cycle of the waveform. L_4 identifies an area of vacant store to be used for translational purposes; etc. L_8 identifies the end point of the waveform at which new data is to be added.

Translation of the waveform is shown by using the repetitive sequence F, L_2, U, L_2, where $N \times U$ equals the translation for one cycle of the waveform (Overwriting of the translation vector, with successively longer vectors, may be preferred if the maximum increment is sufficient). L_2 is addressed and replaced by U and L_2 is re-entered for future addressing.

Continuity of movement is achieved by updating the waveform at L_8 alternately with translation:

It is necessary for the CPU to reset the translation procedure at intervals in order to conserve store. The procedure illustrated in Fig. 4 attaches the translating sequence to a constant height in a new cycle (which happens to be the lowest point) and then deletes the previous cycle and translation vectors. The procedure requires the CPU to note the geometry of the waveform for three cycles. As shown it is assumed that when L_5 reaches the position P then the status of the store is represented by St*n*-*Mn* addresses L_4 and defines its position as P_1; it then deletes the sequences $L_1\ L_2\ L_3$. The next message recreates L_1 (at L_4) and inserts a translation vector and L_2 into the vacant store previously identified by L_4. The procedures are now similar to M_1.

The message content requires eight words per increment of translation. If an increment is chosen to be 1 per cent of a cycle then a 2400 baud lines would permit translation of 1 cycle in approximately 5 seconds, or alternatively nearly 1 cycle per second with a 5 per cent increment.

References

GROVER, D. J. (1971). Low Cost Graphics display with serial access store. *The Computer Bulletin*, Vol. 15, No. 1, January 1971, pp. 33-37.

ISO Recommendation R646 (1967). 6 and 7-bit Coded Character Sets for Information Processing Interchange.

RICHARDS, R. K. (1966). *Electronic Digital Systems*, John Wiley & Sons.

ROSS, H. M. (1970). The British Standard Data Code and how to exploit it. *The Computer Journal*, Vol. 13, No. 3, August 1970, pp. 223-229.

Bibliography

COTTON, I. W. and GREATOREX, F. S. (1968) Data structures and techniques for remote computer graphics, *Proc. FJCC*, 1968, pp. 533-544.
COTTON, I. W. (1970). Languages for Graphic attention handling, *Computer Graphics 70*, Brunel University, April 1970.
ELLIOT GREEN, R. (1970). Computer Graphics, *Computer Aided Design* supplement Spring 1970, pp. 29-48.
ELLIOTT, W. S., JENKINS, A. P., and JONES, C. B. (1970). An Interactive Graphical System using computers linked by a voice grade line. *Computer Graphics* 70, Brunel University, April 1970.
GRAY, J. C. (1967). Compound data structure for CAD. *Proc. ACM National Conference*, 1967, pp. 355-365.
INGRAM, D. G. W. (1970). Man-Computer Interaction in the Design of Telecommunications Systems. IEE Conference on Man-Computer Interaction, September 1970.
LEWIN, M. H. (1967). An introduction to Computer Graphic Terminals. *Proc. IEEE*, Vol. 55, No. 9, September 1967.
MCKINLEY, J. T. (1970). Further Conversational Techniques developed for Remote Access Computer Aided Design. IEE Conference on Man-Computer Interaction, September 1970.
NEWMAN, W. M. (1969). Interactive graphical response and its effect on display system performance. *International Symposium on Man-Machine systems*, Vol. 4, September 1969.
NINKE, W. H. (1965). Graphic 1—A Remote Graphical display console system. *Proc. FJCC* 1965, pp. 839-846.
STOTZ, R. H. and CHEEK, T. B. (1967). A low cost graphic display for a computer timesharing console. Project MAC ESL, MIT July 1967 (EL-TM-316).
SUTHERLAND, I. E. (1963). Sketch pad—A man machine graphical communication system. *Proc. SJCC* 1963, 329-346.
SUTHERLAND, I. E. (1966). Computer Graphics—10 unsolved problems. *Datamation*, May 1966.
VAN DAM, A. (1970). Storage tube graphics: Comparison of terminals. *Computer Graphics 70*, Brunel University, April 1970.
VAN DAM, A. Lecture Notes.
WISEMAN, N. E. (1968). A Note on compiling display file from a data structure. *The Computer Journal*, Vol. 11, No. 2, August 1968, pp. 141-147.

Book review

Approximate Linear Algebraic Equations, by I. B. Kuperman, 1971; 225 pages. (*Von Nostrand–Reinhold*, £6·00)

The term 'Approximate linear algebraic equations' means that the values of the coefficients of the matrix of the equations and also the right hand sides are not known precisely, instead it is supposed that these quantities are contained in given small intervals. Moreover it is supposed that all arithmetic calculations are exact, so the subject of this volume is quite different from the usual error analysis of linear equations that occurs in most books on numerical analysis. Indeed the main problem that is considered is to estimate the range of each component of the solution of the equations that is consistent with the data.

This problem is not straightforward because the set of possible solution vectors that are admitted by the data need not be convex. However the intersection of this set with each orthant of solution space is convex, so it is possible to solve the problem by a finite number of linear programming calculations. This technique is described very well. However applying the technique can require much computation, so some simpler methods are presented for estimating the range of the solution vector.

No less than eight methods of estimation are described of various degrees of complexity and accuracy. Of course the methods that require less computation tend to be less accurate, but, with the exception of Method 1 that is derived from vector and matrix norms, all of the methods give realistic estimates when the data intervals are small. Moreover some of the methods provide actual bounds on the components of the solution of the linear equations.

The style of writing is rather unusual, for the main results are emphasised by repetition. Indeed the author has taken a great deal of trouble to communicate his ideas unambiguously. Most of these ideas are illustrated by numerical examples, but there are no exercises for the reader. The mathematical accuracy of the book is of a particularly high standard.

Except for an overlap with Dr. E. Hansen's work on interval arithmetic and a chapter on statistical methods, the subject matter is the research of the author. This subject is interesting from an academic point of view, and also it is relevant to practical computation. Therefore I recommend this book very warmly.

M. J. D. POWELL (Harwell)