Data structures and techniques for remote computer graphics

by IRA W. COTTON

Sperry Rand Corporation
Philadelphia, Pennsylvania

and
FRANK S. GREATOREX, JR

Adams Associates
Bedford, Massachusetts

INTRODUCTION

It has been adequately demonstrated[1,2,3] that computer graphics systems need not require the dedication of a large scale computer for their operation. Computer graphics has followed the trends of computing in general, where remote access, time sharing, and multiprogramming have become the key phrases. The problems involved in providing a remotely accessed, interactive computer graphics system are more formidable than for a dedicated system, or even than for a local time shared system. The requirement is basically to obtain a real time response for the display console operator, while at the same time minimizing the overhead imposed on the central computer facilities. Also present, of course, are the classic graphical problems such as that of providing refresh data for cathode ray tube displays, and of relating the appearances of a picture on the tube face to its description in the data structure.

Figures 1 and 2 illustrate a typical configuration as it

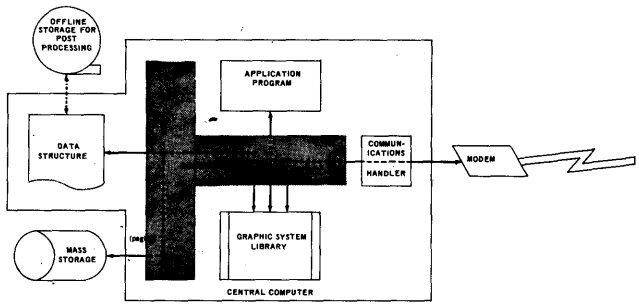
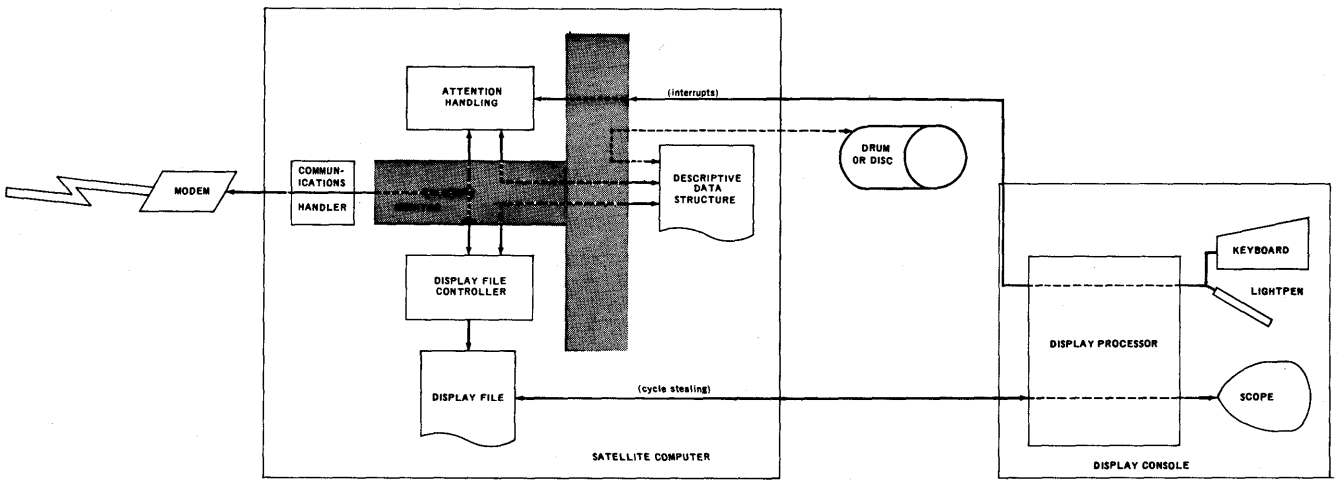


FIGURE 1—Graphics system organization in the central computer

FIGURE 2—Graphics system organization in the satellite computer



would appear to a single user. Actually, the central computer is time-shared by a number of users, both graphic and non-graphic, and there may be a number of other special features. For example, if the graphic system library is reentrant, the same copy may be used by all application programs. There may also be a large number of remote stations, and it may be possible for one satellite processor to drive more than one display. A number of configurations have been described[1,2,4] and it is not the intent of this paper to describe a particular hardware implementation. Such an implementation is currently in progress and will be described at a later date, but the system has been designed to be essentially hardware independent in the sense that the implementation either in the central or satellite computers may be recoded for different or improved hardware, while still maintaining the same interface with each othre and with the application program. The structure of the system is also sufficiently modular so that enhancements such as the ability to support additional input devices may be added without great difficulty. This paper will describe this system, show how the data structure is an integral and dynamic part of it, and focus on how interactions with the data structure occur both at the central and satellite computers.

System software requirements

Interactive graphics systems should provide the means for the basic functions of computing, data management, image generation, and attention handling. Certain software is required to provide these facilities in a remotely accessed environment (central site software for time-sharing is assumed):

1) There must be an *application program* which provides ultimate control of the work being done by the user at the display console.
2) There should be a *data base* with sufficient structure to provide the applications program, the display presentation software, and any post processing routines with the means of defining and integrating the actions of the user in formulating valid images.
3) There should be a group of *service routines* with the ability to manipulate the central site data structure under the direction of the application program.
4) There must be *telecommunication routines* for the control of messages between the remote graphics processor and the central site computer.
5) There must be a *monitor program* in the remote computer to build and maintain the display file from the description of the picture in the data base and to direct attention handling at the satellite so as to provide acceptable real-time response to operator actions at the display console.

Application program

The application program is provided by the user. Several points are significant. First, the user is completely free to perform whatever computations are required as an adjunct to his display requirements. Second, the user may have an independent data base apart from the graphics system, or it may be integrated into the graphics system. Normally. the interface between application programs and the system is desired via FORTRAN-type subroutine calls. Application programs will not be considered further in this paper.

Data base

The system data base, referred to as the *Entity Table*, is a hierarchically organized data file, dynamically expandable and modularized to permit paging. Structures in this list are indexed by a hash-coded directory, also modularized, for the purpose of rapid retrieval. The creation, manipulation and retrieval of structures in the Enitity Table are performed by system supplied subroutines. The list processing technique of ring chaining[4] is extensively employed to improve the efficiency of scanning the data base. Since the length of each entry varies, depending on its type, garbage collection routines are embedded in the system unbeknownst to the user, to prevent excessive fragmentation and minimize storage requirements.

At this point, it might be asked why a data structure is required at all for computer graphics applications. The main reasons for using a hierarchically organized data structure such as will be described may be summarized as follows:

1) It appears to arise normally from the organization of the class of pictures normally considered in computer graphics;[5]
2) It allows the system to take advantage of the graphical subroutining capability being built into current display hardware;[6]
3) Such a data structure is important not only to the display, but also for the use of the applications program presumably directing the course of the graphical processing (for example, the same data structure used to store the description of an electronic circuit for display could also contain the data used in the analysis of the circuit).

A viable data structure is crucial to the success of any computer graphics system, especially if it is resolved that the system is to serve as a useful tool, rather than just an expensive toy.

The basic philosophy of the data structure is to embed the facility for hierarchical picture definition within a list processing scheme with sufficient power to allow easy manipulation and retrieval of the pictorial elements, and sufficient flexibility to be essentially hardware independent both in its own design and in the description of pictorial elements.

This goal of hardware independence has a number of virtues. First of all, hardware independence means that the applications programmer need not be burdened with worrying about hardware idiosyncrasies of the display device. Secondly, it means that the system can be employed to drive a number of output devices, including digital plotters for hard copy, and is amenable to post-processing of various sorts (the area of numerically controlled tools comes immediately to mind). Finally, hardware independence at the central computer means that a consistent application program interface could be maintained if that computer were replaced; this also applies to the satellite processor if hardware independence can be achieved there also. Eventual replacement of at least a portion of the hardware is almost a certainty, given the rapid rate of new developments in computer technology. A sufficiently flexible, hardware independent system guarantees that technological advances will not make the system prematurely obsolete.

Service routines

The service routines constitute the library of list-processing subroutines which allow the user to build and manipulate his data structure in the central computer. The data structure is accessible only through these routines. Since the system is implemented in a time-shared environment, it is desirable that these routines be re-entrant, so that a number of application programs may "simultaneously" make use of them.

A number of list-processing schemes have been proposed for computer graphics.[7] The subroutines provided in the system are naturally determined by the requirements of the particular data structure selected. A representative set of routines for the data structure to be described are listed in the appendix, with a summary of their function.

Communications

Telecommunications routines are of two types: physical and logical. "Physical" routines are required to mediate the transmission of all required messages between the central and satellite computers. They are assumed to be provided by the executive system which permits the satellite processor to remotely access the central processor. "Logical" routines provide the user with a means of sending and receiving information to and from each of the processors, possibly in an encoded or parametized or metalinguistic form. Clearly, such routines must be provided in both computers.

An interactive remote graphics system faces three major problems with respect to communications: 1) the time required to transmit data could be excessively long, 2) the overhead lead on the central computer would be high if it is required to service all interrupts, and 3) as a result of the above, the reaction time to operator actions at the display would be very slow. Experience with graphic systems indicates that some feedback to operator actions should be provided almost immediately (such as intensifying a light-pen detected image), with the effects of his actions apparent within seconds, at most.

It is assumed that because of line-rental costs, transmission is restricted to a single linkage such as a 210-B modem, which would make available, at most, a transmission rate of 2400 bits per second.

Obviously, the transmission time for lines and points cannot be improved. However, the transmission times for conics (circles, ellipses, parabolas, hyperbolas) can be decreased by including in the remote processor sufficient power to generate the straight line approximations to these conics from a parametric representation. To minimize the number of expansion routines in the remote processor, the parametric homogeneous mathematics for conics[8], has been selected.

An additional reduction can be achieved in the quantity of data which must be sent when pictures are to be displayed with a high degree of shape repetition. The description of a shape can be transmitted only once, with subsequent instances of that shape requiring only a reference to that shape as previously sent. This is one further argument for a hierarchically organized data structure.

Remote monitor

Software in the remote processor includes a routine to store the logical organization of a picture sent to it from the central site. Another routine builds from this data a display file, containing the actual vector and character hardware commands for that picture's display. This routine contains the logic required to expand the parametric representation of conics to a short vector approximation. Sufficient structure is present in the organization of the data and sufficient power is provided in the monitor to permit the identification and manipulation of elements within the picture being displayed. The ability to respond locally to operator actions is a very powerful feature which can be built into remote systems containing a satellite processor. This raises the interesting question of the division of labor:

how much work is the central site processor to do and how much is the remote processor to do?

Interactive graphic systems are characterized by large numbers of short-burst program executions associated with scope operator actions such as pushing a button, tracking a light target, etc. Seconds can elapse between these bursts, each of which might only consist of retaining the identity of a button pushed or of moving a tracking cross. In a time-shared environment, these short bursts can overload the system if the application program has to be retrieved from auxiliary storage; also, since the system operates remote displays at the end of relatively slow transmission lines (as a 201–B modem), response times could be too slow.

An interpretive language has therefore been designed to allow the user to program his own strategy for attention handling and other functions to be performed in the remote computer. Because these programs in some sense reflect the logic of the particular application, they are referred to as Logic Tables. Logic Tables are executed (interpreted) in the remote processor independently of application program execution at the central site. The intent is to allow the application program complete freedom over the method of system controls and to minimize the interactive communication between the resident programs in the satellite computer and the Worker in the central processor. Individual Logic Table commands may be thought of as subroutine calls that respond to attentions (e.g., keystrokes), conditionally transfer control as a function of input from the operator, request light-pen tracking and pointing, perform insertions and deletions to the satellite data structure, and initiate the transmission of internal messages to the Worker Program.

This approach has three important features not yet available in other graphic systems: 1) it allows the application to design its own control program (not forced to accept a rigid scheme), 2) it provides for remote-console real-time responses, but 3) it frees the application program at the central site of the requirement to provide such responses, thus lowering the time-shared overhead. In a sense, the user has the ability to "fine-tune" the system for his own particular requirements.

Data structures

The data structures in the system are of two main types: a data structure for the definition and organization of graphic structures in the central and satellite computers, and a data structure for image presentation and logical attentional handling. These data structures have an intimate relationship, the second being constructed directly from the first. Each data structure, however, may be independently modified.

Entity table elements

To facilitate the organization and manipulation of the graphic representation being defined or manipulated by the application program, a central data structure called the Entity Table has been defined, which serves as the repository for all the data within the system. Associated with the Entity Table is a modularized, hash-coded reference table called the Directory, which provides rapid access to prime structural entries in the Entity Table. A third table serves as an External Directory, correlating user defined alphanumeric names with their internal numeric identification codes in the Directory and Entity Table.

The prime structural element in the Entity Table is the *group*. Composed of graphical entities called *items* and *uses* (instances) of other groups, it provides the basic "subroutining" capability in the data structure. All sub-structures within groups are defined in terms of the group's coordinate system and are collectively referenced by the identification associated with the group.

A group by itself is essentially an unpositioned organizational structure which may not itself be displayed. A group may be used, however, so that the graphical sub-entities in its organization be displayed.

The concept of group use is basically one of taking a group structure, complete except for its orientation relative to the user's drawing coordinates, and applying an affine transformation relative to the coordinate system of the group receiving the use entry (called the parent group). If a link is formed by an iterative chain with the universal or master group, the group use is then defined relative to the drawing coordinates and its position can be determined for display purposes.

This system allows a structure (of groups and uses) to be defined of considerable complexity with no duplicative effort, since the same group definition can be used repeatedly in different contexts. With this ability to define groups in terms of other groups, it is entirely possible to cause a recursive definition. Since no recursive constraints have been provided, such recursion would be infinite, and hence cannot be allowed. A violation would be discovered when the data structure was searched in preparation for transmission to the satellite, and an error code would be returned to the application program.

Thus far we have described a picture-defining structure with nothing that is displayable (actually we have defined a directed graph with no terminals). Imen entities are logical combinations of points, lines, and test(called components) which form a picture and which are primitive elements within groups. Items provide the actual graphical information which may be displayed. They respond to freely positionable graphic sub-

routines which are repeatedly "called" when the group of which they are a member is used. Items themselves may not be multiply used, but are defined once within a particular group. A group may be defined, however, with only a single item entry, and be repeatedly used.

Groups are represented in the Entity Table as a header with a pointer to the chain of entities (uses and items) in the group. The group is identified by its unique internal identification code. Pointers are provided to associated data (managerial data) and to a chain of all the uses of this group. The Directory serves as an index to all groups in the Entity Table. A group's internal identification code serves as a virtual pointer to the group header which is resolved through the Directory.

A use entry has a pointer to the next entry within the parent group structure, a pointer to the next use of the same dependent group, a pointer to the component ring for the use (examples of components for uses are external name, positioning vector and transformation matrix), and a pointer to associated managerial data. The Logic Table associated with this use is also identified if one has been defined. The group referenced by the use is linked by its unique internal identification code via the Directory).

Item entries require a pointer to their component ring (components for items include the actual text, vectors and points displayed on the tube face), but naturally do not include any group references.

Components themselves contain a count of the number of parameter words following and a pointer to the next entry in the item component chain. The order of components in the ring is significant (the only case where this is so), since they may contain polystrings of relatively positioned points and vectors.

Figure 3 presents a simple schematic meta-language for describing a structure of groups, uses, and items, and Figure 4 illustrates how a portion of such a structure would be represented internally at the central site.

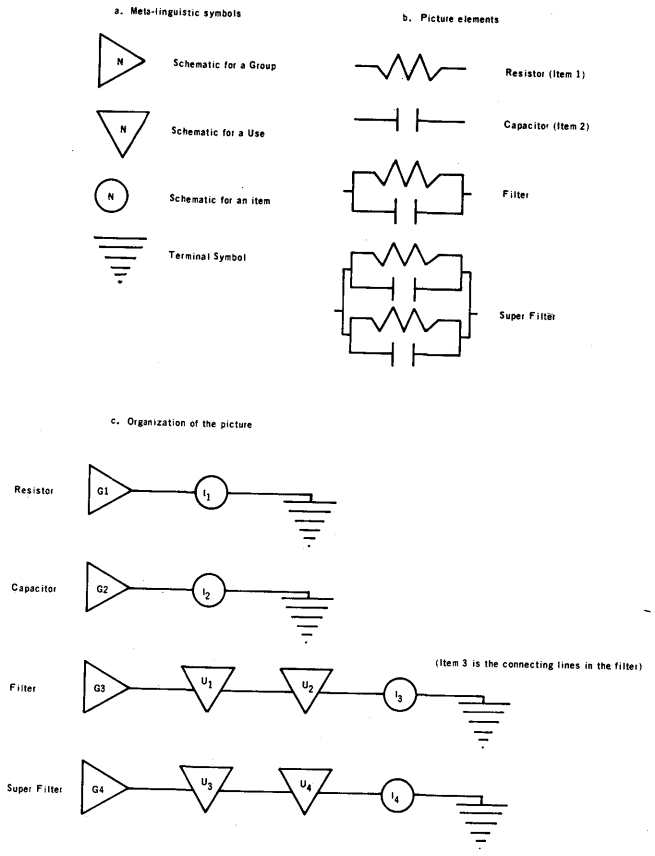


FIGURE 3—Schematic representation of the organization of a picture

Attributes

Associated with each entry in the Entity Table are a number of logical and functional attributes which serve as modifiers or descriptors for that entry. Not all attributes are applicable to all entities. The concept of attribute is a general one, and the system could easily be modified to accommodate additional attributes in reponse to additional user requirements or new hardware capability (such as color, for example). The attributes following are representative, and are taken from the implementation currently in progress.

First of all, an entity may have an external name associated with it for the user's convenience in referencing it. However, the system works with internal identification codes which are unknown to the user, and does not require all entities to have external names in order to be referenced; if the graphics terminal has a light-pen, entities may be identified by pointing to them (an example of "the positive power of grunting"[9]). If a light-pen is not available, one may be simulated using the keyboard. An attribute is provided which specifies if the entity is eligible for light-pen detection.

Other attributes similarly refer to the graphic characteristics of the entity. One attribute defines whether or not the entity is displayable, and another defines if it is to be displayed; thus an entity can be temporarily deactivated without being removed from the data structure. Attributes define line type (for example, solid, dashed or end point, intensity (most displays allow at least two settings in addition to off), and thickness (if hardware permits). Color is an obvious extension which can easily be accommodated when such hardware becomes generally available. Attributes can also be defined to specify interest, threshold, and security level. Interest and security levels are self-explanatory; threshold level is simply a function of magnification and the resolution capability of the display device: as magnification increases, more detail can be shown.

With respect to the graphic attributes of displayable

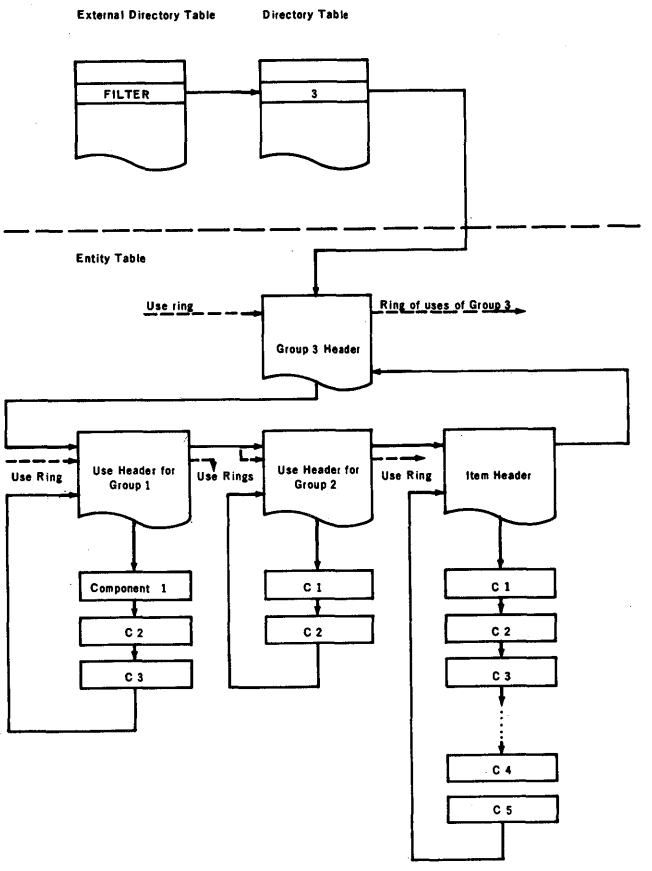


FIGURE 4—Example of a portion of the data structure in the central computer for the representation of Group 3 in Figure 3

items, the hierarchical structure of the picture definition is employed to resolve conflicts which may arise in the data structure. The attributes of a group use override the attributes of the referenced group, which in turn override the group uses within the definition of that group. The rule is that higher level definitions take precedence.

Three functional attributes have special significance to the efficient operation of the remote graphics system. When a picture description is to be sent to the satellite computer, its entire structure must be retrieved, scanned, and transmitted (at one time or another). An attribute is provided to aid in the efficiency of this scan. Whenever the scan of a group is completed, it is marked as having been searched. If this entity is encountered again in the scan of the whole structure (such as a group used repeatedly), it need not be scanned again, but may ben referenced by its internal ID. Similarly, when an entity is transmitted to the remote computer, it is marked as having been processed. If the need later arises to transmit such an entity to the remote, only its identification code need be sent, since a copy of it is known to exist in the data base of the satellite.

The third attribute of special significance arises from the ability within the system to specify the display of only a segment of the picture which has been defined, and to later dynamically alter the extent of that segment.

The extent of the coordinate system within which pictures may be defined is limited only by the magnitude of the largest floating point number which the central computer can represent. Pictures defined in this universe will be said to be in "user coordinates." A segment of this universe is selected for display by the definition of a drawing mask, which identifies a rectangular section of the universe. The picture is scissored to lie within this section, and appropriately scaled for display so as to map onto a selected display window on the face of the CRT. The size of this coordinate system is limited by the CRT raster itself. Figure 5 illustrates this concept. There may be a number of masks simultaneously mapped onto associated display windows and simultaneously displayed. Each window's content and display is independent, however. Both masks and windows may overlap.

Specifying a segment or region of the user defined picture to be displayed on the scope involves scissoring pic-

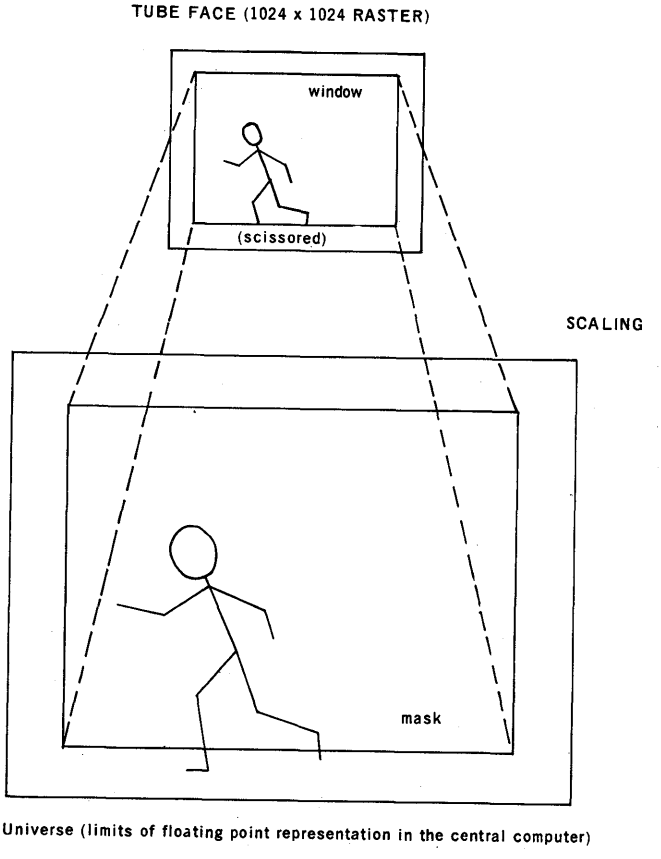


FIGURE 5—Defining and displaying an image. The Universe constitutes the domain of definition for pictures. The mask selects a segment to be mapped onto a particular window for display

ture elements which lie partially or entirely outside of the mask. For reasons of transmission efficiency and storage economy at the satellite, scissored data are eliminated from the picture's description wherever possible, thus altering its description. The description which is sent is marked as having been scissored, so that programs in the satellite may be aware that the picture's description is incomplete. If a complete description is later required for one reason or another, it may be necessary to refer back to the central computer to obtain it.

The problem of scissoring assumes somewhat greater proportions when there are a number of windows displayed on the tube face of less than full screen size, and when interactive work by the user is likely to result in the movement of pictures on the tube face. Hardware assistance can usually be obtained for the problem of full screen scissoring,[3] but this is not generally the case when arbitrary boundaries are to be established beyond which images will not be permitted to extend. It becomes the responsibility of the software to provide this service, within the framework of the graphics system and the data structure provided. This service must be provided whenever a call is given to display an image.

Initiating a display

A call to initiate or alter a display sets off a complex chain of events. Starting with the master group for each window, the Entity Table must be searched and all the appropriate graphic elements extracted. This requires essentially a canonical scan of all the branches of the structure. The value of the "Processed" attribute is evident here. When an entity marked as processed and "not scissored" is encountered later in the scan, only its identification need be recorded.

The next step, actually performed when the entity is being extracted, is to convert all the floating point values in its definition to fixed point, with scissoring performed according to the drawing mask and with appropriate scaling according to the relationship of the drawing mask to the display window. Entities which are scissored are marked. Entities lying entirely outside of the mask are discarded, and their parent entity marked as scissored.

The resulting structure, in fixed point form, is then transmitted to the remote processor, where a display file is constructed and the image displayed. The organizational structure is retained in the remote processor for the purposes of relating this structure to attention handling. If a drum or disc is available to the remote processor, this *structure file* may be stored there, and retrieved as required.

Satellite data structure

The data structures in the satellite processor are of three types. First, there is a description of the picture to be displayed in each window in terms of groups, group uses and items. This is just the structure file, or picture organization which is sent from the central site, and does not contain anything which is interpreted by the display presentation hardware.

The second type of structure contains the actual vector and character commands to be displayed. Since the data to be displayed must be associated with an item use (actually a use of the group in which the item resides), the entites which contain the display commands are called item blocks. These blocks are essentially freely positionable graphic subroutines, and are part of the display file.

The ordering, positioning for display, and determination of eligibility for light pen-detection of the item blocks is controlled by the third type of structure. Called the ordered display list, it is constructed by extracting the group and item uses implied by the hierarchical organization of the picture.

Group occurrences within the ordered display list are accomplished through control blocks which contain a jump instruction (interpreted by the display hardware) to lower structures (other groups or item control blocks), and then a jump to the next structure in the parent group. These jumps are not return jumps (subroutine calls), since the display list is explicit for all groups and uses. However, lower level structures contain jumps back to their parent structures, so that the display decoding hardware executes a complete pass through the picture organization on each refresh cycle. Control blocks also contain a command word for the control of the light-pen and trap interrupts, and a chain pointer to the ring of all repetitive occurrences of the entry. This pointer is located in the block so that it is not seen by the display decoding hardware.

Item control blocks contain positioning vectors and return jumps to item blocks. In addition they contain a control word for light pen and trap interrupts, and a chain pointer to the ring of all instances of this item. The jump command exiting from this block may be to another item or group control block or back to a parent group control block.

Item blocks themselves are the graphic subroutines containing the actual vector and character instructions which are displayed.

The descriptive structure is related to the display structure through the rings of groups and uses. Figure 6 gives an example of the organization of this data structure for the same graphic structures used previously in Figure 4.

Logic tables

Logic Tables provide a novel approach to the problem of defining and accomplishing the processing to be performed in the satellite computer. Most remote systems are limited in their flexibility since it is difficult, if not impossible, for the ordinary user to specify the program in the satellite computer. Besides the function of basic attention handling, it is often desirable for the satellite to be able to analyze and change both its data base and display file, and to be able to send and request information to and from the central computer as required.

Typical graphic applications vary widely in the type of response required of the satellite processor. For example, one application might be to display intermediate results of a large data reduction process, and interactively direct the course of the analysis[10] while another common application is to provide the console user with a sketching facility.[11] In the first case, the satellite would be required to transmit key depression information back to the central computer to direct the calling of different subroutines, while the second case requires that the satellite be programmed to analyze the actions of the console operator, define new structures, alter existing structures, and perhaps notify the central computer of such additions and deletions.

The Logic Tables are a hardware independent, interpretively executed, interactively oriented language in which processing to be performed by the satellite can be easily programmed by the user. The applications program initially specifies the Logic Tables as strings of mnemonic commands, and associates them with use or item structures in the data base. Not all structures must have Logic Tables associated with them, except that the master or universal group must have a Logic Table associated, since it is always the first Logic Table interpreted. During execution, Logic Tables are translated and transferred to the satellite with their associated structure. The Logic Table interpreter portion of the satellite resident monitor will execute one pass through the master Logic Table whenever one of the following occurs: the light-pen picks a display target, the tracking symbol is moved, a software timer expires, or a key is depressed on the alphanumeric or function keyboard. Based on its analysis of the action to which it responds, the master Logic Table may initiate execution of sub-Logic Tables; such nesting may continue arbitrarily deep. Logic Table commands include the following:

1) Arithmetic and logical capability using software registers for the retention of results.
2) Ability to transfer control conditionally within a Logic Table and to other Logic Tables.
3) Ability to decode keystrokes and to react to timer expirations.
4) Control of light pen tracking and picking identification and feedback.
5) Definitional capability for points, lines, text and conics.
6) Facilities for obtaining the complete genealogy

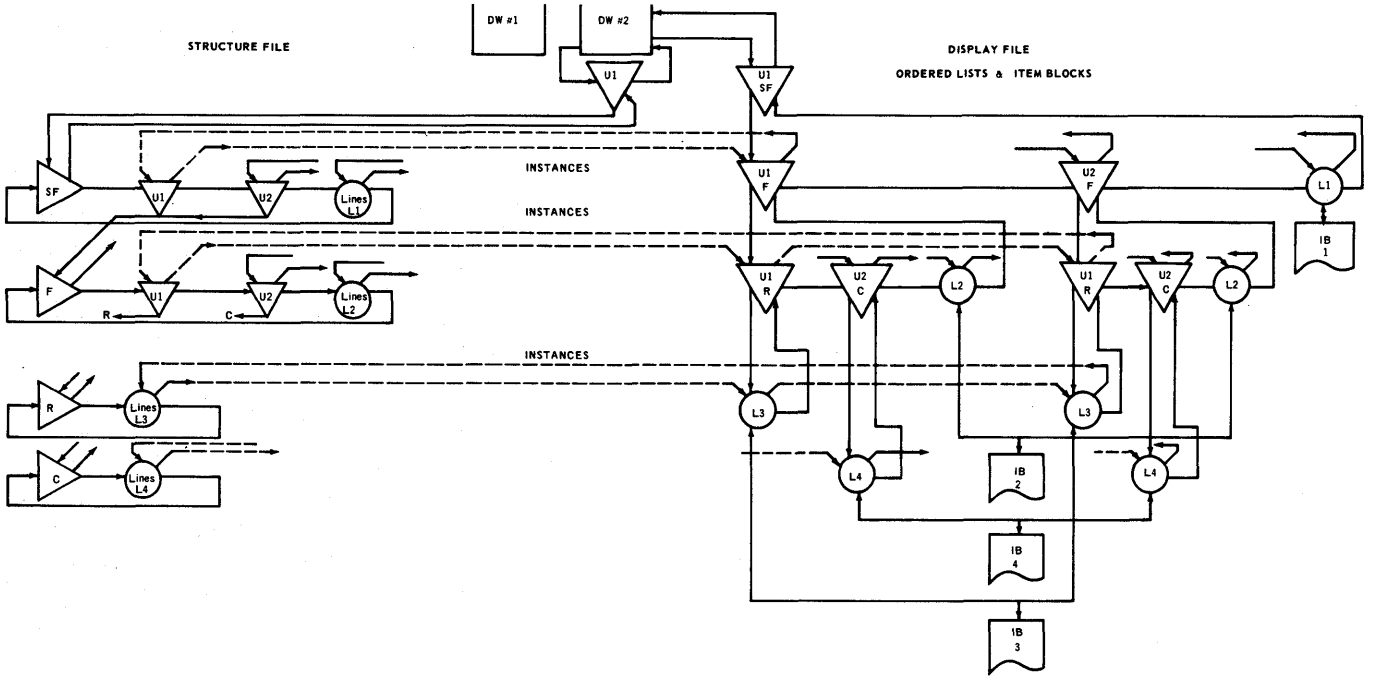


FIGURE 6—Example of the data structure in the satellite computer for the same picture used in Figures 3 and 4

(hierarchical identification) of any entity in the data structure, as well as the identification of its type.

7) Ability to search the data structure either horizontally (within a group) or vertically (from parent to dependent entity or vice versa).
8) Ability to define new entities and alter existing entities, including the ability to reposition entities and to turn them on and off.
9) Ability to record data into report blocks for transmission to the central computer, and to initiate this transmission.

Some examples of the interaction of the Logic Tables with the system's data structures, both in the satellite and in the central computer, follow.

Light-pen identification

In the past, light-pen picking was limited to a one-to-one relationship of the item block hit to a single level of identification. While simple in approach, this did not always satisfy the needs of the graphic system. The data structure design allows the Logic Tables to identify for the scope user either an item, particular use of a group, or all uses of a group.

When the light produced by an item block which is enabled for pen detection is sensed by the light-pen, an interrupt is triggered by the hardware, causing the resident monitor to activate the identification sequence. This sequence may vary depending on the processor in which the system is implemented. For the processor in the present implementation, only the address in core of the display command which generated the light which was sensed is available. Starting with this address, the item block is scanned to find its end, where may be found a pointer back to its head. Just as subroutines may be traced back to back to a calling program by return addresses, so may item blocks and group uses be traced back through this parent structure to the master group.

A Logic Table command is provided to make this genealogy available by entering it on a push-down list which may be manipulated by other Logic Table commands. The push-down is constructed so that dependent structures are on the top of the stack, and parent structures deeper in the push-down. Push and Pop commands are provided for searching the data structure in the vertical sense. A Ring command is also provided for traversing the data structure in the horizontal sense. Feedback (blinking) is also under the control of the Logic Table, which may select a particular instance or all instances of entities at any level in the structure for feedback, depending on the contents of the top entry in the push-down list at the time of the request for feedback.

Light-pen tracking

Light-pen tracking presents a somewhat different problem. While the display is ordinarily refreshed at a rate somewhere between 30 and 60 times per second, this is too slow to detect all but the slowest movements of the light-pen across the tube face. Provision must therefore be made to display the tracking symbol more often—about 200 times per second has proved to be satisfactory.

This is accomplished by the cooperative utilization of trap commands in the display file and logical interrupts occuring every five milliseconds from a real-time clock in the satellite processor. Clock pulses stimulate immediate display of the tracking symbol if display file presentation is not currently in progress. If display file presentation is in progress, display of the tracking symbol is delayed until the next trap command, in order that the display file be resumed at the proper location. Traditional tracking methods[12] can thus be applied with no detriment to the picture being displayed.

Logic Table commands are not required to insert the trap commands, to monitor the real-time clock, to perform the actual tracking calculations, nor to reposition the tracking symbol. The logical effects of pen tracking, however, are all under the control of the Logic Table. An approach is taken which seeks to provide sufficient tracking information without overloading the system. The approach is to require the Logic Table to establish a range around the tracking symbol prior to enabling tracking. The Logic Table will be executed as a result of tracking only when the tracking symbol is moved outside this range. Thus, the user may obtain either very fine or very coarse tracking, depending on his particular requirements.

When a Logic Table is invoked as a result of tracking, the new X and Y position of the tracking symbol is made available. The Logic Table may do with these as it wishes. The rubberbanding of lines and the inclusion of vertical or horizontal constraints are particularly easy with other Logic Table commands which allow lines to be defined on the basis of data in software registers. Thus, Logic Tables represent a very powerful tool for the construction of sketching facilities.

Local modification of structures

Since the Logic Tables provide the ability to modify the data structure in the satellite computer both by altering existing structures and by defining new entities, the satellite's data file may be used as a scratch-pad for changes to be made to the central site data file. All changes made by the Logic Tables are strictly local to

the satellite, and have no effect on any structures in the central site. In fact, the communications link could be disconnected between central site and satellite during the period in which the console operator wished to try out his modifications, without degrading the console response. Of course, the link would have to be re-established before these changes could be incorporated into the central site data base. The facilities of the report block would be used to inform the application program of console operator actions or of modification performed by the Logic Tables in response to console operator actions. The application program would then instruct that changes be made to the Entity Table paralleling the changes made in the satellite.

Report block

Report blocks provide for the transmission of information about the console user's actions from the satellite to the central computer. Report blocks are built in response to Logic Table commands, and only transmitted to the application program at the direction of the Logic Table. Report blocks allow accumulation of data to form a logical, comprehensive set of parameters for application program execution. By deferring application program activation until a complete message is ready, this approach should keep conversational overhead to a minimum.

Actually, report block information is not simply allowed to accumulate in the satellite until the report block is terminated and sent. Since core is at a premium in the satellite, and in order to decrease the amount of transmission which must actually be performed when the completed report block is requested to be sent, a report block buffer is provided in the satellite which is flushed and transmitted to the central site whenever it fills. A complete report block, however, implies a report block header and terminator, indicating a complete logical message. While the report block may be transmitted in fragmented physical segments as it accumulates, it is not made available to the application program until a complete logical report block is built and sent by the Logic Table.

The following types of information may be recorded into a report block by the Logic Tables: constant values, software register contents, push-down list contents (genealogy of an entity), coordinates of a particular point, and new use and item entries. This information becomes available to the application program in a general parametized form which the worker can employ to reconstruct the actions of the console user which the Logic Tables were programmed to record. Based on this information, the application program may alter the data structure in the central computer to reflect the current status of the image on the tube face.

Scissoring

As has been described, all picture elements are defined within a user universe defined in floating point user coordinates. Scissoring first became necessary when a segment of this universe was selected to be displayed on the tube face. When the call to display an image was given, the data structure was searched to determine what lays within the defined limits. Items which lay completely outside the limits were discarded, but their parent structures marked as scissored. Items which lay partially out were converted to a scissored representation. This may have involved discarding certain components and altering others (for example, shortening a line). In this case the parent structure was also marked as scissored. The scissored picture was converted to fixed point representation so as to map onto the display window, and transmitted to the satellite for display.

Scissoring (or unscissoring) may be required at the satellite processor when entities are moved under the direction of the Logic Tables. These are two possible cases: entities may be moved outside the defined limits so as to require scissoring, or scissored entities may be moved so as to lie more with the limits. The first case the satellite processor can handle by itself it has an unscissored version with which to work. Sufficient power is built into the resident monitor to perform the required scissoring of the entity as it is moved outside the limits. The second case will require the assistance of the central computer. Recall that entities which lay entirely outside the limits were not transmitted to the satellite. The satellite is not aware of even which these entities are; all it knows is that the parent structure was scissored. When the parent structure is moved so that more of it may be displayed, a request must be made back to the central computer to provide the additional data required to display structures coming into view.

Since the central computer is aware of which entities are stored in the satellite's data base, it may be able to save on transmission by only sending identification codes for entities which it knows are available at the satellite, and permitting the satellite to perform the appropriate scissoring. In the case of partially scissored entities, it may be that the satellite has an unscissored version in its data base. If it can be determined that only this entity is required to be moved (Logic Table commands provide the ability to reposition particular entities), then the satellite can use the unscissored version without the necessity of a request back to the central computer.

Logic table extensions

Once the basic interpretation cycle for Logic Tables

has been implemented new Logic Table commands may be added (by a systems programmer) without great difficulty. Since the full power of the satellite computer is available, rather elaborate Logic Table commands may be designed to satisfy particular user requirements. For example, a problem in the effective use of the light-pen has been the inability to determine the exact position of the pen when a detect is recorded on a line generated by a single vector instruction. A Logic Table command has been designed which can provide this information by automatically displaying the locus of addressable points on the line in a search for the pen.

Another example of the imaginative use of Logic Table capabilities are in the display of labeled *construction points*. These are points displayed under Logic Table control which are not part of the data structure. Other Logic Table connamds may associate such points with the parameters of particular graphic structures, such as conics, for their automatic definition and inclusion into the data structure.

Finally, Logic Table commands may be designed which make advantageous use of the computing power and information available at the central site. For example, in mapping user coordinates onto raster coordinates, a certain degree of resolution is usually lost. Reconverting raster coordinates back to user coordinates then results in values differing from the original. Facilities can be included in the graphics system to insure that reconversion results in values within the original locus. Such facilities may also be employed to insure that a newly defined point which lies visually on a structure lies actually on it.

CONCLUSION

In summary, the system described is a synthesis of the already popular methods for hierarchical picture definition and the technique of interpretive specification of programs to be performed in a small satellite processor. This approach results in a system with a higher degree of flexibility than that found in most graphics systems. The system is designed to satisfy the requirements of dynamically defining and altering graphical images, within a remotely accessed environment. It is an interactively oriented system, such as would be required for design automation. It is only fair to point out that providing this capability occasionally results in additional overhead beyond that which might be required for systems oriented towards output only. Certainly there must be an output-oriented system embedded in the interactive system, but this system may at times be less efficient that it might need to be, were display its only function. In a sense, the display of an image is an afterthought; the system is primarily concerned with the description organization of the picture. As has been pointed out, this approach lends itself well to a hardware independent interface.

APPENDIX

A representative set of service routines for the central site are outlined. Only routines required by the system are included. Other routines could be included in a working system for application program convenience or economy.

WINDOW—define a CRT window and associated mask delimiting the display of a given data structure.

GROUP —create a group-header entry in the data structure.

USE —place a use-header for a specified group into a specified parent group.

ITEM —place an item-header entry into a specified group.

COMPE —add components to the specified item or use header.

DELETE —delete the specified entity from the data structure.

DISPLAY —display any specified graphic structure contained in the Entity Table, subject to attribute logic plus mask and window limitations.

ASSOC —enters arrays of managerial information into the Entity Table.

GET —retrieves any specified data from the Entity Table.

LOGIC —performs Logic Table conversion and enters the Logic Table into the Entity Table associated with a specified entity.

WAIT —suspends worker program activity until a report block is sent from the remote.

REPORT —retrieves report blocks entries.

ACKNOWLEDGMENT

The authors wish to acknowledge R. Ladson, N. Fritchie, and G. Halliday of UNIVAC, who were responsible for initiating and directing the work described, and D. Cohen and R. Baust of Adams Associates who made significant contributions to the system design.

The design of the Graphics System was developed by Adams Associates, under contract to UNIVAC. The implementation currently in progress employs UNIVAC 1557 Display Controllers and 1558 Display Consoles remotely accessing UNIVAC 1108 computers.

REFERENCES

1 D T ROSS et al
The design and programming of a display interface system integrating multi-access and satellite computers
ACM/SHARE 4th Annual Design Automation Workshop Los Angeles California June 26–28 1967

2 C CHRISTENSEN E N PINSON
Multi-function graphics for a large computer system
Proc of the FJCC Vol 31 Thompson Book Co Wash DC 1967 pp 697–711

3 W H NINKE
A satellite display console system for a multi-access central computer
To be presented at 1968 IFIP Congress

4 W R SUTHERLAND
The CORAL language and data structure
PhD Dissertation MIT Cambridge Mass 1966

5 A VAN DAM D EVANS
A compact data structure for storing retrieving and mainipulating line drawings
Proc of the SJCC Vol 30 Thompson Book Co Wash DC 1967 pp 601–610

6 M H LEWIN
An introduction to computer graphic terminals
Proc IEEE Vol 55 No 9 Sept 1967 pp 1544–1552

7 J C GRAY
Compound data structure for computer aided design
Procedures of the ACM National Conference 1967 pp 355–365

8 L G ROBERTS
Homogeneous matrix representation and mainpulation of n-dimensional constructs
Notes for the Engineering Summer Conf Univ of Michigan Ann Arbor June 14–18 1963

9 A VAN DAM
Computer driven displays and their use in man/machine interaction
Advances in Computers Academic Press New York 1966

10 BOWMAN & LICKHALTER
Graphical data management in a time-shared environment
Proc of the SJCC Vol 30 Thompson Book Co Wash DC 1967 pp 353–362

11 I E SUTHERLAND
Sketchpad a man-machine graphical communication system
Proc of the SJCC Vol 32 Thompson Book Co Wash DC 1968 pp 329–346

12 ADAMS ASSOCIATES
Computer Display Review
Bedford Mass