A compact data structure for storing, retrieving and manipulating line drawings

by ANDRIES Van DAM
Brown University
Providence, Rhode Island

and

DAVID EVANS
University of Pennsylvania
Philadelphia, Pennsylvania

INTRODUCTION

The field of graphical man/machine interaction is customarily split into hardware and software areas. The former can be considered to have come of age: there are over twenty-five brands of off-the-shelf consoles with all the requisite input devices, and new techniques and improvements are constantly being developed. Many consoles are also provided with primitive supporting software which allow one to draw points, lines, arcs, etc., in a symbolic language of some sort. Less well understood and developed, however, is that aspect of display software concerned with representing and manipulating the problem model from which these primitive point/line/arc pictures are derived. The "data structure" is the machine representation of the often complex and hierarchical problem model. It must be judiciously derived from the model on the one hand and, on the other, lead readily to the reduced console display file of points, lines and arcs which cause the actual visual display. Furthermore, the data structure must be efficiently stored and processed (usually contradictory requirements).

Historically, pictorial (line drawing) data structures have been at one of two extremes: the obvious, first-approximation, single-level structure already on the point/line/arc-level of the console display file, or a highly complex, hierarchical and interconnected list structure (Ross's "plex,"[1] Sutherland's "ring,"[2] Roberts' CORAL lists,[3] or even Feldman's[4] and Rovner's[5] hashing schemes for simulating an associative memory). The second method, in our opinion, is more elegant and powerful in terms of storage and processing economy for hierarchical and repetitive pictures. However, the degree of interconnectedness of the list structures (forward pointers, backward pointers, head-of-list and tail-of-list pointers, etc.) may considerably inflate the size of the block of storage (the "item") which represents a picture. In addition, most of the above schemes rely on fixed position of the various pointers within blocks, so that the growth of an item is relatively constrained, and updates are intricate. Special list processing languages (such as CORAL) in which one can write picture processors have been written to manipulate these data structures and pointers.

An alternate approach is described below. The data structure is purposely kept as free of pointers as possible (without losing any topological or geometrical information) to reduce the size of a given item to the absolute minimum, and picture processing "primitives" are written as "worker programs" in assembly language or are bootstrapped from simpler ones. The subpicture hierarchy is recursively built into every "master" item by describing it as being composed of points, lines, and "instances" of lower-level subpicture masters, to which affine transformations have been applied. The MULTILANG system* provides automatic linking and retrieval of those items using instances of the same subpicture master, and of the primitives' worker programs. The IBM 7040 implementation described below accepts card input, and uses high-speed printer output since a vector scope is not available. Input of all commands is therefore in a digital rather than a light pen/function key language. Pictorial ENCodIng Language (PENCIL) primi-

*"THE MULTILANG on-line programming system", R.L. Wexelblat and H.A. Freedman. Proceedings of the 1967 Spring Joint Computer Conference.

Table I

PENCIL Primitives*

A. DEFINITION

* POINT/NAME/X,Y/OP — where OP = N, for NAME, and T, for TEXT

* LINE/NAME/PT1/PT2

ARC/NAME/PT1/PT2/PT3

PARC/NAME/F/V/PT1/PT2

EARC/NAME/F1/F2/PT1/PT2

B. MANIPULATION (level 1)

* COIN/PT1/PT2

* MERGE/PT1/PT2

* COPY/NAME/L/PT

* INTPT/NAME/L/X,Y

SPIN/L1/L2/PT/DEGREE

C. TRANSFORMATION

* MOVE/NAME/OP/PT/QUAL — where OP = P, for point; N, for node; S, for subpicture; D, for display, i.e., the screen level picture

* ROTATE/NAME/OP/PT/DEGREE/QUAL

* SCALE/NAME/OP/PT/DEGREE,XX,YY/QUAL

D. CONTROL

* CLEAR

* DELETE/NAME/OP/QUAL — where OP = P, for point; N, for node; L, for line; S, for subpicture; U, for uniformly

* ASSIGN/NAME/NODE1/.../NODEn

* USE/NAME/QUAL

* SHOW/NAME

LOCATE/X,Y; ENDLOC

TEXT/PT/OP1/H/W/OP2/---...-$--...-$...$$ — where OP1 = TL, for top left; TR, for top right; BL, for bottom left; BR, for bottom right

OP2 = L, for left; R, for right

*Those primitives included in the present version of PENCIL are identified by *. This set provides all basic capabilities, including hierarchical assembly of pictures.

```
XTRANS PROC    BUFIN
       CLEAR
       USE/DIODE/1
       USE/DIODE/2                    LINE/L4/P4/P6
       USE/DIODE/3                    LINE/L5/P7/P8
       USE/RESIS/4                    POINT/P10/=15,=0
       USE/RESIS/5                    LINE/L6/P8/P10
       USE/RESIS/6                    LINE/L7/P10/P9
       USE/RESIS/7                    MOVE/D01/N/P1/1
       USE/TRANS/8                    MOVE/D02/N/P2/2
       USE/CAPAC/9                    MOVE/D03/N/P3/3
       USE/GROUND/11                  MOVE/A4/N/P3/4
       POINT/P1/=-21,=24              MOVE/A5/N/P9/5
       POINT/P2/=-21,=12              MOVE/D28/N/P9/8
       POINT/P3/=-21,=0               MOVE/B6/N/C8/6
       POINT/P4/=-15,=12              ROTATE/RESIS/S/A7/=90./7
       POINT/P5/=-15,=0               MOVE/A7/N/P5/7
       POINT/P6/=-12,=12              MOVE/D29/N/P6/9
       POINT/P7/=12,=12               MOVE/A11/N/C8/11
       POINT/P8/=15,=12               ASSIGN/BUFIN/IN1/IN2/IN3/OUT
       POINT/P9/=21,=0         END
       POINT/OUT/=57,=12       XBEGIN
       POINT/IN1/=-48,=24
       POINT/IN2/=-48,=12
       POINT/IN3/=-48,=0
       LINE/L1/P1/P3
       LINE/L2/P3/P5
       LINE/L3/P5/P4
```

Figure 1 – Procedure BUFIN

tives (see Table I) are specified as MULTILANG statements, and picture processing "procedures" composed of these statements are processed by MULTILANG. (Figure 1, procedure BUFIN, is a sample procedure† for the highest-level drawing indicated schematically in the exercise of Figure 2; Figure 3 is its result on the printer. Figure 4 shows the same circuit drawn from a points-and-lines model on the IBM 2250). The PENCIL picture processor itself is not dependent on these digital modes of I/O: only small pre-and post-processing I/O routines need be altered to accommodate a vector scope, while the digital inputs were chosen to be as similar as possible to scope-mode inputs.

A complete specification of the system described below may be found in Reference 7.

PENCIL Items

Assumptions

In the PENCIL approach several fundamental assumptions were made. One was that the hierarchical subpicture structure is natural for the class of two- or three-dimensional line drawings under consideration (flowcharts, block diagrams, electrical circuit schematics, trusses, pipe diagrams, etc.). This led to the SKETCHPAD[2] master/instance technique. It was also assumed that picture items should be stored and retrieved in the same manner as are any other types or data or program in the MULTILANG

†Two small discrepancies between the format of POINT here and POINT in Table 1 exist: x,y values are passed with a Hollerith =sign, and OP = N for NAME is assumed when left blank.

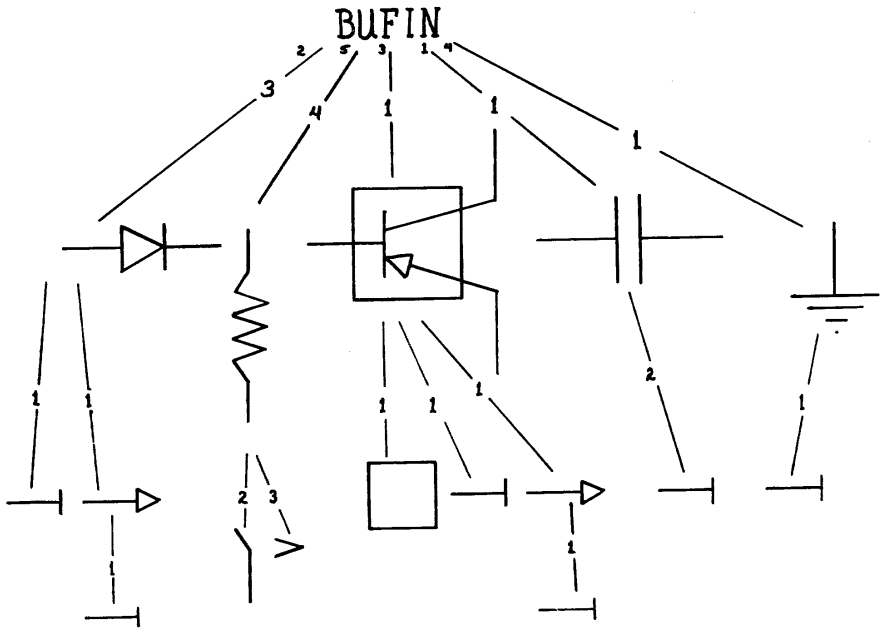

Figure 2 – Assembly tree of BUFIN

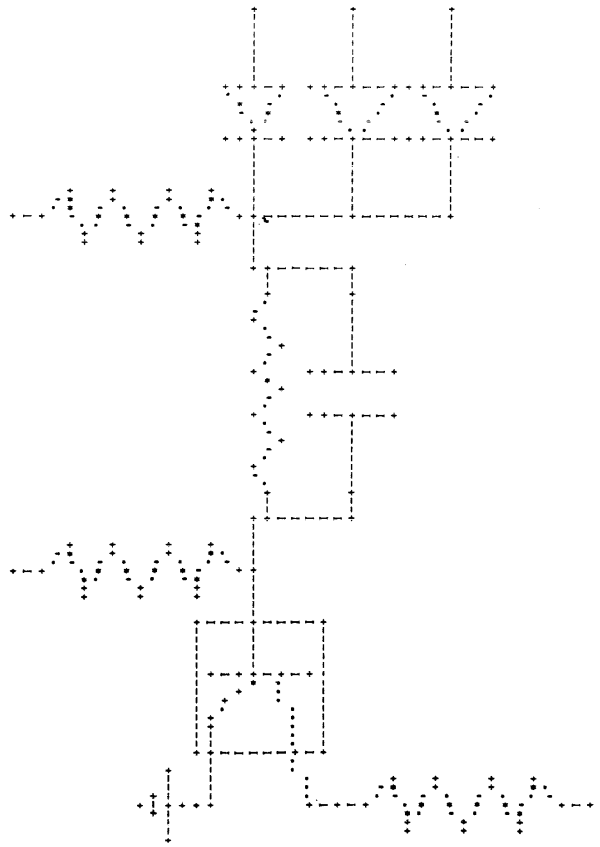

Figure 3 – Buffer inverter drawn on 1403

system: by "description," i.e., by an essentially Boolean function of key words (keys) or ranges of keys. Thus browsing is possible, since a picture or pictures can be identified by a variable configuration of keys pertaining to topological, geometric or merely descriptive information (e.g., number or type of components used, date on which it was drawn, manufacturer(s), power rating of a specific component, etc.). The third (and easily accepted) assumption was that core and disk space were at a premium, especially if realistically large files of items containing much "descriptive" information were to be handled.

These assumptions led to the use of MULTILANG threading and retrieving as a basis for PENCIL, since it is feasible to avoid redundancy and still get all the necessary information from the data structure without undue sequential searching.

Control item and line item

The basic unit of data is a picture, composed of points, lines and/or other pictures. The PENCIL primitives operate on these pictures to DEFINE, MANIPULATE, TRANSFORM or CONTROL them (see Table 1). Any picture may be used recursively as a component in a "higher-level" picture, and at such time is designated a "subpicture."

A picture (subpicture) is represented in the file by a "Control Item" and associated "Line Items," "Text Item," and "Information Item." Each item is a MULTILANG data item[6] and as such has the control information which MULTILANG requires. Figure 5 shows schematically a completed Control Item, the principal picture item containing point and subpicture information. The MULTILANG interpretation of the blocks is listed on the left and the PENCIL interpretation on the right (see below).

The Control Item contains a first, "primary" key (and its link) which is initially set to a non-printable internal code and is replaced with the assigned name of the picture when the user completes it. Following four special-purpose keys and their links (see Appendix A, section 1), keys occur which are the names of the subpictures composing the picture. The geometrical disposition of each subpicture within the picture is represented in the Control Item by a data element which contains a cumulative record of all transformations which were applied to the subpicture. This "Transformation Matrix" data element (TM) contains a rotation/scale matrix and a translation vector for each occurrence (instance) of the given subpicture. There is also single "Proper Transformation Matrix" data element (PTM) which accumulates the transformations applied to the screen (current display) as a whole (including all subpictures, points and lines of which the screen is composed).

Points explicit on the screen are represented in the Control Item in a "Points Table" data element (PT) in which the name of a point and its x,y,(z) values on the screen are listed consecutively. Lines defined on the level of the screen are not explicitly contained in the Control Item: they are encoded in separate MULTILANG Data Items called "PENCIL Line

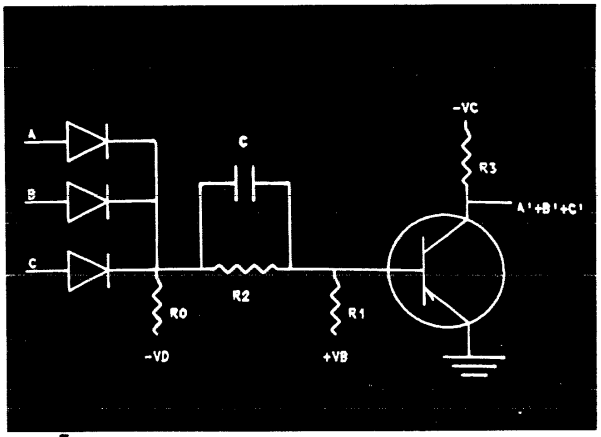

Figure 4 – Buffer inverter drawn on 2250

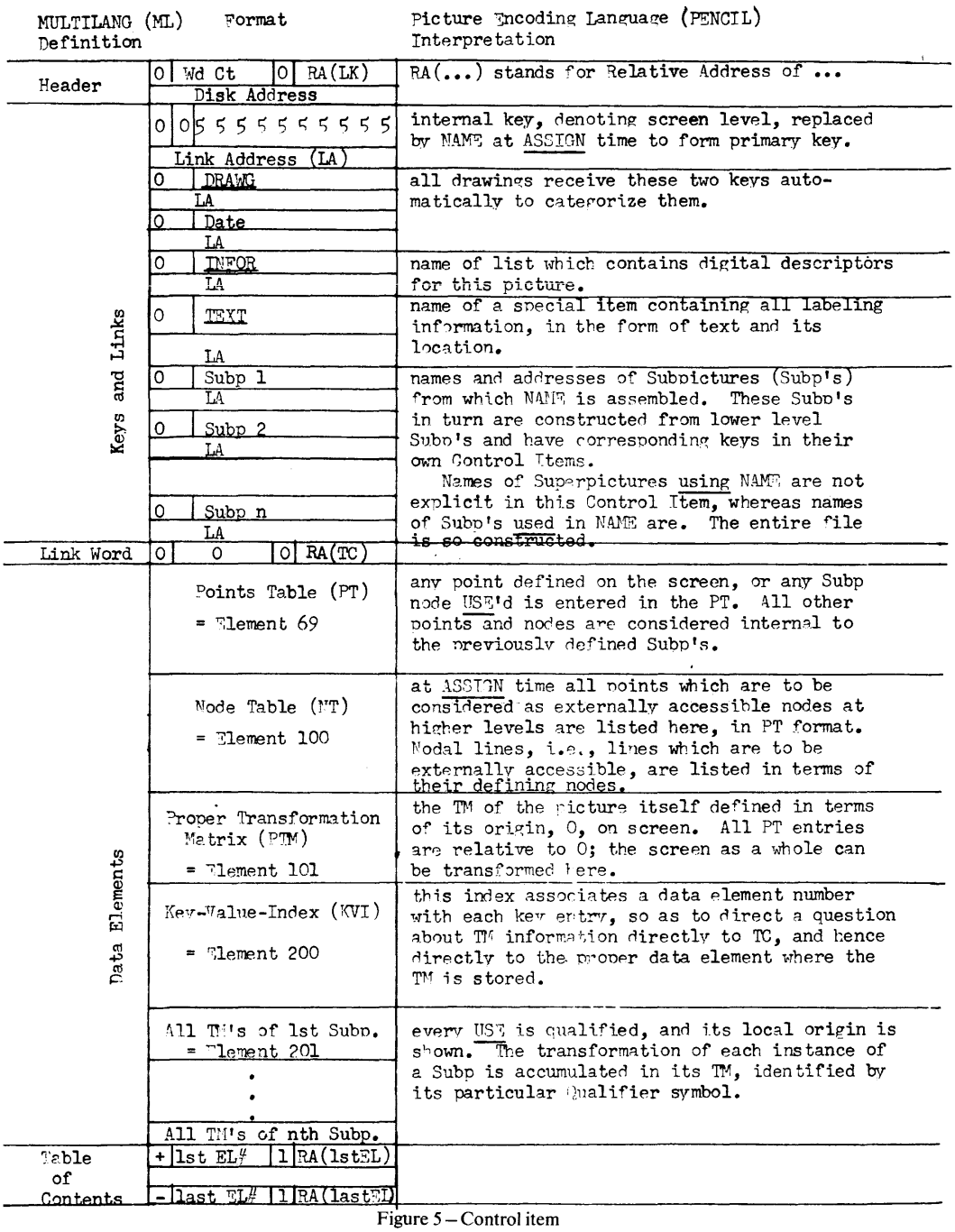

Figure 5 – Control item

Items"* (Figure 6) in which they are defined in terms of their endpoints. These endpoints are keys in the Line Item and also entries in the Points Table of the super Control Item. The Control Item is linked to all its Line Items through the appearance of its picture name as a key in all of them. The primary key of a Line Item is the key *LINE*†† and its name is a second key. Thus, data items can be uniquely categorized: if a picture name appears as primary key in the item, it is the Control Item for that picture. If the picture name is not the primary key and if the key *LINE* is present, it is a Line Item belonging to the

*In what follows "line" or "Line Item" denotes both a conventional straight line (*LINE*) and a conic section: circular arc (*ARC*) parabolic arc (*PARC*) or elliptical arc (*EARC*). The corresponding Items are similar to the Line Item.

††Italized words of capitals such as *LINE* or *CLEAR* denote the identical alphanumeric characters strings in PENCIL Items and primitives. Words in capitals but not italized denote variable names.

ML	Format				Picture Encoding Language (PENCIL) Interpretation
Header	0	17	0	12	
	Disk Address				
Keys & Links	0	LINE			LINE as primary key identifies this as a Line item, as opposed to a picture control item.
	Link Address (LA)				
	0	5 5 5 5 5 5 5 5 5 5			replaces at ASSIGN time by name of picture in which this line is used.
	LA				
	0	NAME			the line's own NAME
	LA				
	0	Endp 1			endpoints in terms of which the line is defined
	LA				
	0	Endp 2			
	LA				
Link Word	0	0	0	15	
Data Elements	RA(endp 1) in Control Item's PT				
	RA(endp 2) in Control Item's PT				
Table of Contents	+	1	1	13	
	-	2	1	14	

Figure 6 – Line item

picture; if the key *LINE* is absent, the item is the Control Item of a superpicture using the subpicture with the specified name.

Keys and subpicture structure

The choice of the function or interpretation of keys is determined by the method of storing the hierarchical structure of pictures. First of all, it is usually advantageous to assemble as much as possible of the drawing on a subpicture level. Hence subpictures should be included in Control Items very efficiently and compactly. A threaded list structure using the names of subpictures of a given picture as keys would seem the obvious answer, but the utility of this type of a file should be compared with the advantages of an "inverted" file in which each subpicture would contain, as keys, the names of superpictures which use it. In terms of space required the two kinds of files are identical. In terms of processing time, they are not: the question "where is this picture used, and how many times?" (a typical parts inventory question) is asked much less frequently than "of what is this picture composed?" For instance every time a change is made on the screen, the entire component tree, or at least a branch of it, must be traversed to yield the subpicture structure explicitly; this disassembly process would be much less efficient for the inverted filestructure.

To produce the console display file with the present scheme, a canonical scan with pushdown is performed of the tree of subpictures of the current Control Item. Each node of the tree corresponds to a picture instance: the terminal nodes represent pictures of points and lines only ("level 1") and the root node represents the current screen. Each picture is printed recursively, by printing its subpictures and then its screen level points and lines. Therefore, the first item to be completely printed is the left-most and down-most "level 1" subpicture of the tree.

Matrix pre- and post-multiplication takes place at every node in order to adjust for the current geometrical disposition of the corresponding subpicture within the higher level. Subpicture Control Items are retrieved by MULTILANG from disk or core, as is appropriate. The item found is then placed in a pushdown stack for further reduction, while a copy is maintained in core (if space is available) to avoid another

disk retrieval in case another instance of this subpicture master item is required elsewhere in the tree.

With this choice of key structure the problem of locating a specific line or lines for output or for manipulation is handled by initiating a MULTILANG retrieval on the parent picture name, *LINE*, and the name of the point(s), locating all appropriate lines in sequence; given a line, finding all connecting lines involves initiating a retrieval on the parent picture name, *LINE*, and the names of the endpoints in the first line, etc. Such properties as topological intersection and common use of elements are easily handled in a similar manner.

In addition to the output sequence advantage described above, another good reason for not choosing the "inverted" file is that updating (and hence retrieving and manipulating) of the used subpicture would be required every time the using subpicture added or deleted it. With the built-in component tree, on the other hand, changes are made only to the Control Item and neither to its subpictures nor to its superpictures. Duplication is wasteful of storage, and so it was decided neither to incorporate both files simultaneously nor to store one file structure in a special data element of a picture item. The "inverted" file can be quickly obtained from the component file by retrieving all items with the subpicture's name as key: a single retrieval request in MULTILANG produces sequentially all such items, since they are linked. For each of these items all their super items can be retrieved, etc.

Appendix A contains additional information on the structure of the Control Item.

PENCIL primitives and the assembly process

PENCIL processes take the names of subpictures and lines as data and manipulate the corresponding Control Items and Line Items. The processes can be considered as primitives in an open-ended, growing language, in much the same way that IPL-V's[8] primitives, the "Js," are considered. The intent is to provide a handful of efficient and basic primitives which constitutes a sufficiently complete set to allow bootstrapping, again in the same manner as that in which the IPL-V "J"s are bootstrapped. Calling PENCIL primitives by MULTILANG statements makes it possible to form MULTILANG "procedures" to assemble pictures, or in fact to build higher-level primitives, just as higher-level pictures are built. The eventual capability of MULTILANG to make hierarchical procedure calls and to perform subroutine calls with substitutable parameters will allow the indefinite nesting of PENCIL primitives for efficient bootstrapping.

The set of primitives listed in Table 1 is divided into four subsets, roughly according to similarity of function. The DEFINITION verbs serve to establish new data in the form of points, straight lines or conic sections. The MANIPULATION verbs move lines and points on the screen by making points coincident or identical through merging, and by rotating lines to make specified angles with other lines. TRANSFORMATION verbs are used to apply affine transformations to subpictures used on the screen. The CONTROL verb *CLEAR* erases the PENCIL working area, initializes the screen, and assigns blocks of unused storage to form a skeletal Control Item (and a Text Item which stores digital annotation/legend information for the drawing). *ASSIGN* names the subpicture which has been assembled on the screen and sets up MULTILANG and PENCIL control words in the Control Item prior to calling on MULTILANG to store the item on the disk. *USE* retrieves a picture stored on the disk and displays it on the screen as a subpicture by putting the relevant control information in the Control Item of the screen picture. *SHOW* simply displays the specified picture. *DELETE* erases points, lines and specific subpictures from a given picture, and also can be used to delete a picture uniformly in the file. LOCATE models the pointing feature of light pens.

Appendix B contains additional information on some of these primitives.

CONCLUSIONS

The 7040 PENCIL system has run satisfactorily with only 7000 36-bit words (42K characters) available for both programs and pictorial data, exclusive of MULTILANG. Real time response with a light pen and a buffered scope would require more core for reasonably complex pictures. Pictures frequently used should probably be reduced to points and lines to eliminate tree scanning.

Some other properties of the system are listed below:

(1) Subpictures may be added and deleted easily, in any order; items are of variable size, and there is no order among keys or data elements.

(2) Pictures are intermixed in memory with other types of data and programs. A picture can be treated exactly as is any other block of information, or can be singled out and subjected to special picture processing. In particular, browsing through the pictures with even rather vague criteria is very easily (and economically) done, whereas in most other systems only retrieval by identification number only is possible.

(3) Since pictures are manipulated and drawn by executing statements and procedures in an interpretive mode, picture-processing statements may be mixed

with retrieval or computation statements in the same procedure.

(4) The overhead per subpicture is as low as possible, no more than twelve words for the two-dimensional case: key and link, identifier plus six-word transformation matrix, and at most three additional control words. [Nodes (see Appendix A.2), of course, are optional.] With a reasonable amount of core, most of PENCIL and the entire current picture could be kept in core, especially if it were kept in its reduced points-and-lines form. Only light pen pointing would make this reduction impractical. Some schemes for keeping part of the picture in tree form and part of the picture in reduced form are being considered for the System/360 implementation.

(5) Pictures might be stored more compactly as the PENCIL procedures which drew them than as the equivalent Items (providing, of course, that they were drawn efficiently and without too much trial and error).

ACKNOWLEDGMENTS

The work described in this paper was carried out at the University of Pennsylvania, primarily through support from the Research Division of the Naval Bureau of Supplies and Accounts, and the Information Systems Branch of the Office of Naval Research, under Contract NOnr 551(40).

REFERENCES

1 D T ROSS and J E RODRIGUEZ
Theoretical foundations for the computer-aided design system
Proceedings of the 1963 Spring Joint Computer Conference Vol 23 p 305 1963

2 I E SUTHERLAND
Sketchpad A man-machine graphical communication system
Lincoln Laboratory Technical Report No 296 1963

3 W R SUTHERLAND
The on-line graphical specification of computer procedures PHD dissertation submitted to MIT January 1966. See also
W. KANTROWITZ
CORAL macros-reference guide
Lincoln Laboratory Technical Memorandum No 23L-0003 1965

4 J A FELDMAN
Aspects of associative processing
Lincoln Laboratory Technical Note 1965-13 1965

5 P D ROVNER
An investigation into paging a softwares–simulated associative memory system
University of California (Berkeley) Document No 40.10.90 1966

6 R L WEXELBLAT and H A FREEDMAN
The MULTILANG on-line programming system
Proceedings of the 1967 Spring Joint Computer Conference

7 A van DAM
A study of digital processing of pictorial data
Ph D dissertation Department of Electrical Engineering University of Pennsylvania 1966

8 *Information processing language–V*
Prentice-Hall 1964

APPENDIX A–Additional Information on the Control Item

A.1 Keys and auxiliary information storage

In addition to the primary name key and the subpicture keys, there are four standard keys entered initially in each Control Item (see Figure 5). *DRAWG* is a key which all Control Items share to distinguish them from other MULTILANG items in the MULTILANG file. *DATE* is obtained from the IBSYS 7040 Operating System and allows retrieval of all drawings made within a certain time span. *INFOR* is a key of a list of standard MULTILANG data items, exactly one such list being keyed to any one picture by that picture's name. An *INFOR* item contains all manner of digital descriptive information which one may wish to store in order to be able to retrieve subpictures not merely by their "pictorial" (i.e., component) structure. It also contains physical component characteristics, test data, etc. Applications programs would use MULTILANG item definition programs to establish these items. Thus the Control Item contains only structural (pictorial) information.

The fourth key is *TEXT* and links a Text Item to each picture. This item contains all annotation information in the form of text "boxes": all text is considered as written on successive lines in a rectangular enclosure (never externally apparent) which is "tacked" to the picture at one of its corner points. If the corner point is defined within a subpicture, the text box will be moved with its corner point as the subpicture is moved. The internal format of a Text Item is a set of data elements, one per text box, containing (1) the segmented text and (2) a control word giving the dimensions of the text box, the corner point which serves as the anchor, and whether the text is to be left- or right-justified within the box. Individual points on screen level may also have associated with them a single label, their name, as specified in the option of *POINT*.

A.2 Points and node table (PT and NT)

On the current display level there exist two types of points: those defined at that level and those at subpicture level. Those at subpicture level are in the current display because they were declared "nodes" at the time the subpicture was *ASSIGN*ed. Only such points are considered accessible (in the sense of connections) on the current level; all others are considered internal to the given subpicture. The same point may be declared a node in as many as five different levels. In both tables, point names alternate

with their x,y (z) values. Flags are used to indicate whether the point is to have its name displayed or whether it is the anchor point of a text box.

A.3 Proper transformation matrix (PTM) and sub-picture transformation matrices (TM's)

After a display has been partially filled it may be desirable to move, rotate or scale it in its entirety. The screen has a Transformation Matrix of its own, a "Proper" Transformation Matrix (PTM), which accumulates affine transformations of the entire screen. Furthermore, to each subpicture in a Control Item there corresponds one Transformation Matrix data element in which are stored the Transformation Matrices (TMs), one per instance, of the given subpicture. The entire display or individual subpictures may be transformed in an arbitrary sequence. The results are accumulated in the appropriate elements.

A.4 Key value index (KVI) and table of contents (TC)

It is often necessary that primitives be able to manipulate Transformation Matrices and subpicture-name keys. There must therefore be a quick way to map from a key to the data element containing the corresponding Transformation Matrices. The Key Value Index (KVI) associates an element number from a strictly increasing integer sequence with each new subpicture name key.

The Table of Contents is a list of pairs; the first part of each pair is a data element number, and the second part is the Relative Address (RA) of the first word of that data element. The Table of Contents of an item containing subpictures always contains the four standard entries 69 (PT), 100 (NT), 101 (PTM), 200 (KVI), and as many other entries as there are subpicture name keys in the Control Item. The Table of Contents and the KVI are used to find the Transformation Matrix data element for a given instance of a given subpicture-name key.

APPENDIX B – Control Primitives

The pictorial assembly process using PENCIL primitives is initialized with a *CLEAR*. The drawing of points, lines or subpictures on the blank screen may then proceed.

As soon as the picture is *ASSIGN*ed, it is ready to be used as a subpicture on a higher-level screen. In assembling a complicated drawing, the user must compromise between the tediousness of constructing the picture entirely on the points-and-lines level, and the comparative ease of constructing it on the subpicture level (which makes the drawing more compact but slower to retrieve and process). For example, standard electrical symbol "macros" such as resistors and capacitors are best drawn on the points-and-lines level to avoid retrieval of one or two additional levels of subpictures. An entire buffer inverter, however, is more easily and economically drawn using at least one level of macros.

Since a given subpicture may be *USE*d a number of times on the screen, particular instances must be uniquely identified. This is accomplished by "qualifying" each separate instance of a subpicture. Each *USE* (one per instance) must specify a unique one-or two-character qualifier to be suffixed to all labeled points of the given subpicture. Each picture is assembled on screen level with respect to the screen orgin (0,0) which becomes its "local origin" if the picture is used as a subpicture. Since this local origin of the subpicture instance, properly qualified, always appears with it on the screen and in the Points Table of the screen level Control Item, there is a unique differentiation between instances. Each instance, furthermore, has a separate entry in the Transformation Matrix data element of the subpicture in the screen level Control Item.

During the assembly of a picture, manipulations defined on lines and points are intermixed with those on retrieved subpictures, and there is often occasion to connect subpictures by screen level lines. When the picture is finished and is to be stored (*ASSIGN*ed), the user may specify a list of those screen level points (level 1 and/or subpicture level) which are to be considered as nodes, to allow attachment in higher level usage. He may then "attach" to lines by attaching to the nodes which define them. When the subpicture is *USE*d, only points designed as nodes and the local origin will appear explicitly and in qualified format on the screen.

CLEAR initializes the screen level Control Item with an internal primary key of $(5555555555)_8$. The user names a finished picture with *ASSIGN*, which replaces this "screen" key by the NAME operand in all screen level items. There is never more than one screen level picture, and hence never more than one set of related items with the screen key as a key. As long as the screen key remains, components may be freely manipulated; once the screen is *ASSIGN*ed (and therefore stored), it becomes an immutable entity with a fixed topology, and a geometry which may be changed only by affine transformations on the higher-level screens utilizing this picture. If a user wants to change the structure of a stored picture, he must retrieve it by *SHOW*, which is the inverse of *ASSIGN* in that it replaces the primary key of the Control Item and the picture-name key in all corresponding Line Items with the screen

key, and in effect reinitializes the assembly process on that picture.

An important feature of the assembly process is that it may be mixed with non-pictorial manipulations and computations—the primitives are completelv context-free due to the nature of MULTILANG procedures and their execution.