A DIGITAL VIDEO DISPLAY SYSTEM IMPLEMENTED ON A KIM-1 MICROCOMPUTER†

N. Solntseff and M.D. Drummond
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S 4K1

ABSTRACT

The "microelectronic revolution" and the accompanying decrease in the cost of semiconductor memory has increased the availability of raster-scan graphical displays, yet, as pointed out in a recent survey [BAE79], the implementation of graphics software for raster-scan systems has lagged behind that for random-scan ones. The aim of the work described in the present paper has been to apply random-scan techniques to a system employing a relatively inexpensive raster-scan device. The system, incorporating a display-file processor, is implemented on a KIM-1 microcomputer. The display device is composed of a Micro Technology Unlimited video board and a standard TV monitor.

Keywords and phrases: digital video graphics, random-scan graphics, low-cost graphics, display file processor, display file, computer animation.

1. INTRODUCTION.

Information can be presented pictorially in two distinct ways: The first is by means of line drawings as in the case of a newspaper cartoon, where the picture is made up of an ordered set of lines; the second is exemplified by a hooked rug, where the picture is made up of collections of points. This distinction is also present in computer graphics, where the first approach is used in vector-display devices and the second in raster-scan devices [NEW79].

The aim of the work described in the present paper has been to apply the techniques used in the case of vector-display devices to the display of structured (or segmented) pictures on a relatively inexpensive raster-scan device.

In his comprehensive survey of raster-scan devices, Baecker [BAE79] makes the point that "There is an increasing body of opinion which holds that systems software for raster-scan devices can be constructed in the same manner as that for vector-scan graphics " [SPR75]. Although this has been known for some time, few systems have in fact been constructed (see [BAE79]). Thus, the present work represents an exploratory effort to assess the feasibility of using vector-display techniques for computer graphics on raster-scan devices.

A picture can be broken up into elements, such as, the foreground, the background, and a number of "figures" which together form what may be called a "scene." In their turn, the figures can themselves be built up from a number of smaller components which may be called "parts." When a digital video display system is used to display a picture on a raster-scan cathode ray tube (CRT), the representation of the image stored in digital form must reflect the structure possessed by the original picture [GIL78]. The digital representation may be a "frame buffer," where contiguous sections of memory represent contiguous pixels (discrete points in the original picture). Alternatively, the representation may be in encoded format, in which case it must be translated into frame-buffer format (scan converted) by the display system [BAE79].

For the encoded picture representation to mirror the structure of the picture it defines, the digital representation must also be structured and it is customary to build it up from segments corresponding to the figures in a scene. The latter are built up from primitives corresponding to the lines, points, and text characters of a picture [GIL78]. A digital video display system can thus be pictured in the manner shown in Fig. 1, where process P1 is responsible for receiving encoded picture definitions generated in a host CPU and storing them as a structured "display file" in display-processor memory. Process P2 performs the scan conversion and creates the bit map of the original picture within the frame buffer. The contents of the latter can now be directly displayed by process P3 which represents the display hardware of the complete display processor.

----

†Supported in part by Grant No. A8327 of the National Science and Engineering Research Council of Canada.

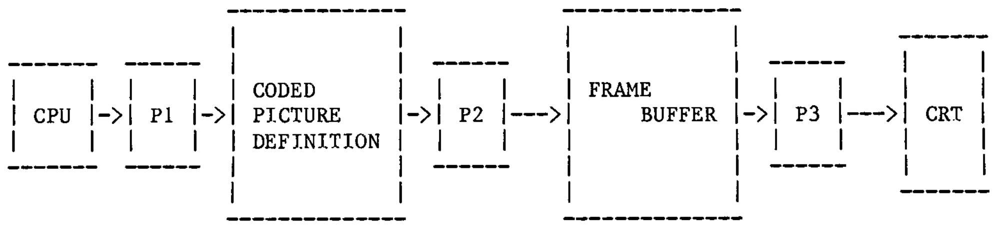

Figure 1. A Digital Video Display System [BAE79].

The generation of an image on a CRT will thus involve the translation of a sequence of picture elements within the display file into actions resulting in the drawing of lines, points, or text characters on the CRT screen. The coded picture definition (display file) may thus be considered as a hierarchical collection of instructions to be executed or interpreted by the display processor. A single instruction, thus, generates the image of a primitive entity on the screen.

Section 2 provides an overview of our system. Section 3 describes the stucture of the display file and the manner in which its instructions are interpreted by the "display-file" interpreter. Section 4 deals with the structure and operation of the "communications" process which accepts coded picture definitions from the host and manages the display file. Finally, Section 5 discusses the uses of the system for the production of animated pictures and outlines directions for future work in the area of low-cost raster-scan devices.

2. OVERVIEW OF THE SYSTEM.

The system described in this paper is of the type called "systems with coded picture definitions" by Baecker and is closest to that shown in his Figure 10 [BAE79]. This is shown in Figure 2 in a somewhat modified form in order to exhibit the system components as implemented on the Commodore/MOS Technology KIM-1 microcomputer.

The host is an Ohio Scientific, Inc. Challenger III, although any computer with an RS232 serial communication link employing an Asynchronous Communications Interface Adaptor (ACIA) [OSB77] can be used for this purpose. Synchronization of the communication between the host CPU and process P1 is achieved by means of one of the modem control signals available with the ACIA. This is described in greater detail below in Section 4.3.

Picture data are generated in the host CPU and transmitted to the KIM-1 in a coded format which will be described below in Section 3. These data are read by process P1 (described in Section 4) and stored as the display file.

The display file is a structured collection of "segments," each of which describes one part of a picture, i.e., the picture is represented in a condensed form which has to be interpreted and converted into a bit map. Process P2, called the display-file interpreter (DF interpreter for short) performs the scan conversion and transforms the coded picture definition into a bit map, where each bit represents a picture pixel. The bit map is stored in the memory provided by the "Visable Memory" video board manufactured by Micro Technology Unlimited [MTU77]. The video-board memory acts as a "frame buffer" and allows the display of 320x200-point pictures. The video board contains the display generator (process P3) which transforms the bit map into a television signal which is then fed into a standard T.V. monitor.

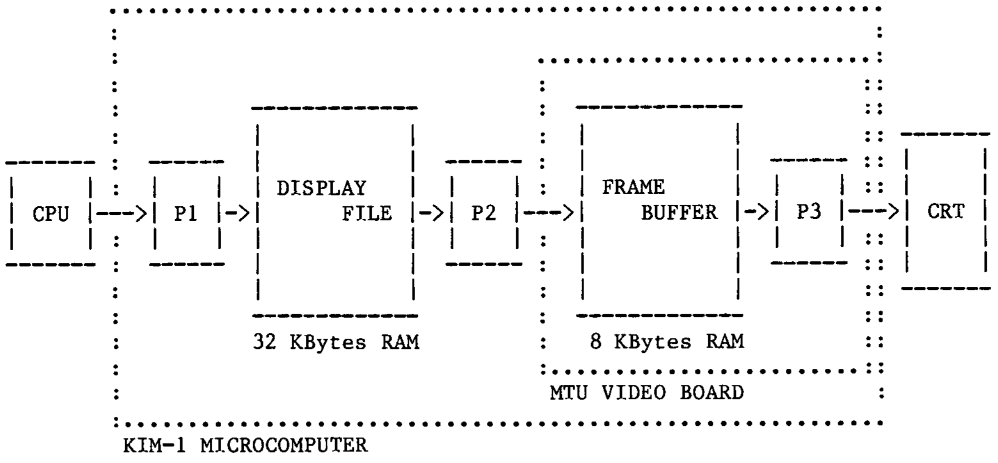

Figure 2. The Digital Video Display System as Implemented on a Commodore/MOS Technology KIM-1 Microcomputer.

3. THE DISPLAY-FILE INTERPRETER.

3.1 Display-File Instructions.

As mentioned in Section 1, the representation of a scene is broken up into a numner of parts to reflect the underlying picture structure. The display file is likewise subdivided into segments, where a segment represents a part of the complete scene and contains the instructions needed to draw it on a CRT screen. These instructions may be categorized into "primitive-generation," "file-organization," and "communications" instructions. A list of available instructions is given in Table 1.

TABLE 1. (a) Primitive-Generation Instructions.

MNEMONIC	INSTRUCTION					SEMANTICS
	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	
LSETPIX	91	abs X Coord		abs Y Coord		move cursor to (X,Y) and set pixel ON
SSETPIX	81	DX	DY	unused		move cursor by (DX,DY) and set pixel ON
LCLRPIX	11	abs X Coord		abs Y Coord		move cursor to (X,Y) and set pixel OFF
SCLRPIX	01	DX	DY	unused		move cursor by (DX,DY) and set pixel OFF
LLINE	92	abs X Coord		abs Y Coord		draw line from current position to (X,Y)
LERASE	12	abs X Coord		abs Y Coord		erase line from current position to (X,Y)
SLINE	82	DX	DY	unused		draw line from current position to (X+DX,Y+DY)
SERASE	12	DX	DY	unused		erase line from current position to (X+DX,Y+DY)
PRINT	04	character string follows				print character string terminated by ETX

TABLE 1. (b) File-Organization Instructions.

MNEMONIC	INSTRUCTIONS			SEMANTICS
	BYTE 1	BYTE 2	BYTE 3	
JMPTO	43	seg#	unused	interpretation continues with first instruction of named segment
JMPSUB	23	seg#	unused	interpretation continues with first instruction of named segment; current seg# & offset within segment are saved
RETURN	13	00	unused	information saved by JMPSUB is restored; interpretation resumes in calling seg.
TEST	07	DX	DY	test pixel at relative position (DX,DY); skip next instruction if pixel is ON
EXIT	05	address		exit the interpreter to given address

TABLE 1. (c) Communications Instructions.

MNEMONIC	INSTRUCTION BYTE	SEMANTICS
SETDFLAG	06	the dynamic flag is set and the PAINTED bit in tne status byte is complemented (see Sect. 3.2)

The routines needed to implement the primitive drawing operations were adapted from the graphics package supplied by Micro Technology Unlimited with their video board [MTU77]. It should be noted that text characters are generated by software and can only be displayed horizontally.

The file organization instructions are five in number and are two or three bytes in length. They provide jumps to and returns from subpictures and are conventional in nature (e.g., by comparison with those given in [NEW79] and [GIL78]).

The third type of instructions are those concerned with "dynamic" segments (see the discussion in Section 5 for details), i.e., segments designed to assist in the generation of animated pictures. Currently, there is only one instruction of this type which is listed in Table 1 (c).

3.2 Control of the Appearance of a Picture Part.

The appearance of a picture part drawn as the result of the interpretation of a segment by the display-file interpreter depends on the values of the various attributes that a picture part can possess. The part may be visible or invisible, it may be static (forming the foreground, the background, or any other immutable part of the picture), or it may be dynamic in order to represent a moving object. If colour or intensity variations can be produced by the display processor, then colour as well as intensity has to be specified.

In the case of the system described in this paper, the attributes are all binary and are primarily used to minimize the regeneration of the display as suggested by Newman and Sproull [NEW79]. Only segments which are changed need to be redrawn and this is indicated by the PAINTED bit [NEW79] which describes the current state of the display.

If the PAINTED bit is ON, then the segment has been drawn on the CRT screen at some time in the past. On the other hand, if the PAINTED bit is OFF, then the segment has been erased, it has never been drawn, or it needs regeneration because it has been changed.

Whether a segment has to be drawn or undrawn (erased) is controlled by a DRAW/ERASE bit which specifies the colour to be used in painting the picture part. Whenever the display is to be updated the following algorithm is followed by the display-file interpreter:

```
if DRAW = ON & PAINTED = OFF
     then interpret segment
          (draw picture part);

if DRAW = ON & PAINTED = ON
     then ignore segment
          (do not redraw;
           picture already visible);

if DRAW = OFF & PAINTED = ON
     then interpret segment
          but set all PIXELS to OFF
          (erase by "undrawing");

if DRAW = OFF & PAINTED = OFF
     then ignore segment
          (do not undraw;
           picture already invisible)
```

In addition, a segment can be deleted and the memory occupied by it returned to the system whenever a picture part is removed from a scene by changing a DELETE bit from OFF to ON.

The three attributes described above are implemented by means of a STATUS byte associated with each segment. For convenience, the STATUS byte is included in the "segment table" (called the name table in [NEW79]), which allows the display-file interpreter to access a segment once it has been given a segment number.

3.3 The DF-Interpreter Algorithm.

The algorithm embodied in the display-file interpreter can be expressed by the following PASCAL-like program:

```
procedure DISPLAYFILEPROCESSOR ;

type DISPLAYFILE : array [ 0..N ] of MEMORY ;
     TABLE       : array [ 0..85 ] of SEGMENTDATA ;
     SEGMENTDATA : record
                      STATUS, ADDRESS : integer
                   end ;
var PC,OPCODE,SEGMENT : integer ;
    DF                : DISPLAYFILE ;
    SEGTABLE          : TABLE ;

procedure DECODE (var DELTA : integer) ;

begin (* decode *)
  PC := PC + DELTA ;
  OPCODE := DF [ PC ] ;
  case OPCODE of
    NOP    : DECODE (1) ;
    POINT  : SETPOINT ;
    VECTOR : DRAWLINE ;
    JUMP   : NEXTSEGMENT ;
    PRINT  : DRAWTEXT ;
    EXIT   : goto DF [ PC+1 ] ;
    SETDFL : SETFLAG ;
    TEST   : READPIXEL
  end
end ; (* decode *)

begin (* update *) ;
  SEGMENT := 0 ;
  NEXTSEGMENT ;
  while SEGTABLE [ SEGMENT ].STATUS <> 0 do
    DECODE (DELTA) ;
  if (any segment was erased) then
     (* fill any holes in the display *)
    begin
      SEGMENT := 1 ;
      while SEGTABLE[ SEGMENT ].STATUS <> 0 do
        begin
          if SEGTABLE[ SEGMENT ].STATUS = POSTED
            then
            SEGTABLE[ SEGMENT ].STATUS
                                     := UNPAINTED ;
          SEGMENT := SEGMENT + 1 ;
        end ;
      DISPLAYFILEPROCESSOR ;
    end
end (* display file update *) ;
```

The six procedures invoked by DECODE comprise the heart of the display-file interpreter. POINT provides the basic capability for setting individual pixels. It calculates the cursor position and calls the appropriate routine from the MTU package to set or clear the memory bit indicated.

VECTOR provides the line-drawing capability and, after calculation of the new cursor position, calls the MTU line drawing routine. For line segments drawn with relative cursor positioning, successive cursor increments may be grouped sequentially following a single opcode to increase efficiency. The sequence is terminated with a null increment. Upon exit from the line drawing routine, the cursor position remains at the end of the last line drawn.

The procedure DRAWTEXT allows the full ASCII character set to be drawn on the display. The character string which appears following the PRINT opcode is drawn by the MTU text drawing routine. The raster matrix for each character is contained in the MTU package. "Undrawing" of characters is accomplished by replacing each character to be drawn with a blank. The character string is terminated by an ETX (04) character.

The NEXTSEGMENT procedure is provided to allow some control over the sequence in which segments are executed and to provide linking from one segment to another. The JUMP opcode may specify one of three types of linkage. The simplest is the JUMPTO (segment number) which retrieves the STATUS and starting address of the specified segment from the SEGTABLE. In addition, JUMPTO also determines from the status whether or not the segment requires interpretation using the algorithm of section 3.2. Segments may also be interpreted as subroutines if the JMPSUB option is specified. Subroutine segments are interpreted and control is RETURNed to the calling segment at the instruction following the JMPSUB call. Any segment may be terminated with a RETURN instruction which simply transfers control to the next segment entered in the SEGTABLE if no JMPSUB call is pending.

The interpretation sequence may also be altered through the use of the READPIXEL procedure. This procedure causes the two-byte instruction which follows to be skipped if the pixel specified (using relative coordinates) is ON. In this manner, dynamic program control can be achieved.

The final procedure causes a DYNAMIC flag to be set so that continuous interpretaion of a segment can be achieved. As well, the PAINTED bit in the segment status byte is complemented so that the JUMPTO procedure recognizes the need to repaint the segment.

The simple instruction set of the display-file interpreter is designed to achieve maximum flexibility with a minimum amount of program overhead so that the device is able to perform at a reasonable speed. Simple displays are created easily yet more complicated, dynamic drawings are not impossible.

4. THE COMMUNICATIONS PROCESS.

4.1 Introduction.

The functions of the communications process P1 (see Fig. 2) are:

(a) to receive picture data from the host,
(b) to receive commands from the host for displaying a picture at any given time,
(c) to send information back to the host concerning the state of the display processor.

A block diagram of the routine implementing P1 is shown in Figure 3. Here, arrows represent subroutine calls and returns. Details of the implementation of the communications process are given in the next sections.

4.2 The Supervisor.

Communication between the host CPU and the KIM digital video display system is achieved by means of a serial line. An additional control line

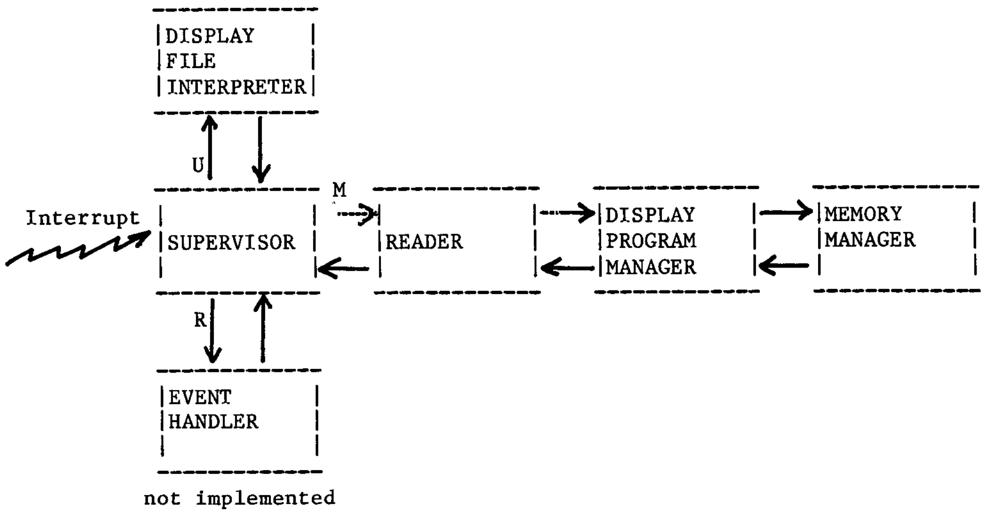

Figure 3. The Structure of the Communications Process in the KIM-1 Microcomputer.

connected to the interrupt request line of the KIM-1 CPU is used to permit the host to interrupt the KIM-1 processor whenever the host requests one of a number of actions to be performed by the KIM-1. These requests are initiated by single-character commands which are summarized in Table 2.

TABLE 2. Commands to the KIM-1 Display Unit from the Host.

COMMAND	FUNCTION
C	CLEAR entire display
M	RECEIVE information for DF Manager
R	SEND information about display status
U	UPDATE (redraw) displayed segment

Note: The R command has not yet been implemented.

The SEND information command, when implemented, will be used by the host to obtain cursor position, light-pen pick coordinates, and other graphical input data.

The RECEIVE command is used to instruct the KIM-1 to read data which represents new segment status information or picture-data elements describing a new segment or additions to an existing segment. The actions which can be performed by the KIM-1 as the result of it receiving the M command are:

(a) change the status byte of a specified segment,
(b) delete a specified segment,
(c) create a new segment,
(d) move an existing segment to a new position,
(e) append picture elements to an existing segment.

The UPDATE command is used to make the display reflect the current state of the display file. On receipt of a U command, the SUPERVISOR calls the display-file interpreter which will scan the segment table from the beginning and regenerate the displayed picture. The effect of the command is that any UNPOSTED/PAINTED picture parts are erased and any picture parts that have to be redrawn at a new screen location will be appropriately moved.

On getting a RECEIVE command, the SUPERVISOR calls the READER routine which handles reading of information sent to the KIM-1 by the host. The operation of the READER is described in detail in the next section.

4.3 The Reader.

The READER is called by the SUPERVISOR whenever data or instructions have to be received by the KIM-1. The synchronization of data transmission between the host and KIM-1 is controlled by the KIM-1. A one-bit control port is used to provide a not-RTS signal to the not-CTS input of the host serial-port ACIA. The KIM is thus able to inhibit data transmission by the host whenever the READER is not ready to receive information.

On being called by the SUPERVISOR as the result of an M command issued by the host, the READER expects to receive a four-byte message of the form

NUMBER OF BYTES	SEG. NO.	INSTRUCTION

The first two bytes give the number of data bytes that follow to specify a new segment or additions to an existing segment. If only status information is being transmitted by the host to the KIM-1, then the number of bytes is zero.

When the READER receives the above message, it calls the display-file (DF) MANAGER to obtain the address in the display-file area into which subsequent data bytes are to be stored. Having obtained this address, the READER stores incoming data until it receives the end-of-transmission character (an ASCII "."). A two-byte checksum is then transmitted, and the READER will respond with the ASCII character ACK if the checksum is verified, otherwise it will send the NAK character back to the host, which will attempt to retransmit the data. The READER returns to the SUPERVISOR on completion of the above cycle of operations.

4.4 The Display-File Manager.

The DF Manager is called by the READER to create space for new segments, to extend existing segments, or to change the STATUS byte in the segment table. The manner in which the DF MANAGER handles its task in response to a request made by the host is briefly described below. These requests are listed in Table 3.

The Append Command causes the addition of picture elements to the specified segment. On receiving this command, the DF MANAGER examines the segment table to find the starting address in KIM-1 memory of the block originally occupied by the segment. The DF MANAGER then calls the MEMORY MANAGER to allocate space sufficient to contain the existing segment data plus the additional data. The DF MANAGER relocates the existing segment and moves its picture elements to the beginning of the newly allocated memory block and calls the MEMORY MANAGER to reclaim the memory block originally occupied by the segment. The DF MANAGER finally returns to the READER with the starting address of the locations into which the additional picture elements should be read.

The Change Status Command causes a change in the status byte of the specified segment. On receiving this command, the DF MANAGER replaces the status byte of the specified segment by the new byte provided by the READER. If the change involves the deletion of the segment, then the MEMORY MANAGER is called to release the memory block occupied by the deleted segment.

TABLE 3. Format for the Commands to the Display Program Manager.

NO. OF ADDITIONAL BYTES		SEG. NO.	STATUS

(a) Command for Appending Picture Elements to an Existing Segment.

0	0	SEG. NO.	NEW STATUS

(b) Command for Changing the Status Byte of a Segment.

0	5	SEG. NO.	MOVESEG

(c) Command for Moving a Segment to a New Screen Position.

NUMBER OF BYTES		SEG. NO.	NEWSEG

(d) Command for Creating a New Segment.

The Move Segment Command causes the display of the specified segment in a new location on the CRT screen. In order to do this, the DF MANAGER examines the first instruction of the segment to determine whether it is an absolute cursor positioning instruction (LSETPIX or LCLRPIX in Table 1). If this is the case, the DF MANAGER calls the READER immediately with the address in KIM-1 memory of the first instruction so that the READER can replace it with a new cursor positioning instruction.

If the first instruction is not an absolute cursor positioning instruction, the DF MANAGER calls the MEMORY MANAGER and requests it to allocate space in KIM-1 memory to hold the segment data prefixed by an absolute cursor positioning instruction. The DF MANAGER then relocates the segment into its new block, leaving five bytes at the beginning into which the new absolute cursor positioning instruction will be placed. The MEMORY MANAGER is next called to reclaim the space previously occupied by the segment. Finally, the DF MANAGER returns to the READER with the new starting address.

The New Segment Command causes the creation of a segment. On receiving this command, the DF MANAGER calls the MEMORY MANAGER to request allocation of space. The DF MANAGER then returns to the READER with the starting address of the newly allocated memory block into which display-file instructions should be read by the READER.

4.5 The Memory Manager.

The function of the MEMORY MANAGER is to reserve blocks of KIM-1 memory for the storage of display-file instructions and to release for reuse any blocks that are no longer needed. The MEMORY MANAGER is a conventional memory management routine employing Knuth's algorithm ([KNU68], Section 2.5). It is described in detail elsewhere ([DRU80]).

5. DISCUSSION AND CONCLUSIONS.

The design and implementation of the system described above was suggested by the remark in [NEW79] that " Experience with raster-scan graphics is simply too limited to justify offering an authoritative design. " Baecker [BAE79] also points out that few systems for raster-scan graphics have been constructed on the same principle as vector-scan systems. The present work should therefore be viewed as an exploratory project and represents only one of many possible approaches to the adaptation of random-scan techniques to raster-scan graphics.

The segmented display file, in particular, is an adaptation of a structure which evolved for use on vector-scan displays. The utility of organizing picture descriptions into segments is well established by experience with vector-scan devices; it allows the user to structure the picture definition into logically related parts as well as to selectively manipulate portions of the picture. As pointed out by Newman and Sproull [NEW79], "...a major requirement for dynamic computer graphics is ... the ability to make selective modifications to the picture..." This is easily achieved through the use of segmented display files. The KIM-1 system described above demonstrates that segmented display files can also be used effectively to organize, generate and manipulate a raster-scan display.

Not all refresh display techniques however, can be used on a raster-scan device. Several important differences require special attention. Erasures on a vector-scan device, for example, are accomplished simply be deleting the particular segment from the refresh list; raster-scan devices require the segment being erased to be "undrawn" by repainting it in the background colour. This leads to complications when some pixels form part of more than one segment. The intersecting pixels must be redrawn, which on a vector-scan display is automatic but on a raster-scan display, explicit regeneration is required. On the KIM-1 system, erasure of any segment causes all posted segments to be re-interpreted immediately. This brute-force approach works satisfactorily and appears to be an effective way of dealing with the problem as the alternative requires a sophisticated, time-consumming algorithm to detect and fill "holes".

A novel feature incorporated into the KIM-1 graphics system is the notion of a "dynamic" attribute associated with a segment. This allows automatic repositioning of the image generated by a "dynamic" segment without the intervention of the host CPU. As an example, it is possible to generate the image of an object which, once started, will move by itself across the display.

The KIM-1 system can serve as the test-bed for future exploration of the implementation of a graphics system on a relatively inexpensive microcomputer. For example, planned future developments include the addition of graphics input devices such as a light pen (easily added to the MTU Video board), joysticks, and a graphics tablet. The addition of line-clipping software for windowing is already under way.

In conclusion, the project described in this paper can be judged to be a highly sucessful one and it will doubtlessly form the basis of a useful and sophisticated raster-scan graphics terminal.

REFERENCES

[BAE79] Ronald Baecker, "Digital video display systems and dynamic graphics," _Computer Graphics_, _13_, No. 2 (August 1979), pp. 48-56.

[DRU80] M.D. Drummond, "A dynamic free storage management package for the 6502", Computer Science Technical Note 80-CSTN-01, Unit for Computer Science, McMaster University, Hamilton, Ontario L8S 4K1 (1980).

[GIL78] W.K. Giloi, _Interactive Computer Graphics, Data Structures, Algorithms, Languages_, Prentice-Hall, Inc. (1978).

[MTU77] Micro Technology Unlimited, Documentation for the K-1008 Visable Memory Board, MTU, Manchester, New Hampshire 03108 (1977).

[NEW79] William M. Newman and Robert F. Sproull, _Principles of Interactive Computer Graphics_, 2nd. Edition, McGraw-Hill (1979).

[OSB77] A. Osborne, S. Jacobson, and J. Kane, _An Introduction to Microcomputers_, _Vol. 3 -- Some Real Products_, June 1977 Revision, Adam Osborne and Associates, Berkeley, California 94702 (June 1977).

[SPR75] R.F. Sproull and W.M. Newman, "The design of gray-scale graphics software," _Procs. of the Conf. on Computer Graphics, Pattern Recognition, and Data Structures_, IEEE Computer Society (May 14-15, 1975), pp. 18-20.