MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo No. 162

July 1968

Remarks on Visual Display and Console Systems

by Marvin Minsky

Part 1

Preliminary Draft of a
Deluxe Picture Maintenance System
June 1963

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge 39, Massachusetts

Project MAC

Technical Memorandum No. 1

Remarks on Visual Display and Console Systems

by Marvin Minsky

Part 1

Preliminary Draft of a

Deluxe Picture Maintenance System

June 14, 1963

Remarks on Visual Display and Console Systems

by Marvin Minsky

Introduction

Vision is the most complex and comprehensive of our sensory modalities. Consequently, we can expect to obtain from visual display environments the most powerful techniques of transmitting information to the user of a complex computer system. In this memo we discuss some aspects of construction and use of visual display systems that are of concern in our effort to obtain more accessible computer systems.

Computer usage techniques are already essentially visual; input and output take the form of the printed page. Punched cards and paper tapes serve only as an intermediate form, except in the hands of the most bit-oriented programmers. Use of the spoken word for computer I-O is technically feasible (though more O- than I-) today, but would not be considered a great advantage by users.

While on the subject of vision in general, I would like to point out that the human visual mechanism includes also what may turn out to be our most comprehensive output channel as well! The speed and precision of the oculomotor channel

is very large. The innervation of the eye muscles is very fine and thorough, and it reflects a chief concern of a large part of the forebrain--the frontal eye fields. This system is probably closely concerned with the input pattern-recognition system, but we will propose later that by suitable computer techniques we may be able to use it also in a motor control mode of possibly unprecedented effectiveness. This potential effectiveness would derive from the circumstance that oculomotor control is more directly connected with the mechanisms of conscious thought, and through a better information channel, than are the other motor modalities.

The immediate problem that faces Project MAC is this. We need a system that will give the user a very intimate channel interaction with his computation. There is no question but that the best system technically feasible today would be based on a high-quality cathode-ray display surface, backed-up by a powerful combination of special features--character generators, line generators, etc. This in turn must be backed up by a powerful picture-construction system--software that can convert a reasonable form of picture-description expression into the elements of the picture to be displayed. In addition, one would want light-pen and other input devices plus the appropriate tracking programs. If expense were no concern, one could specify and obtain displays of near photographic quality.

The trouble is one of expense. The expense is partly

located in the cost of the high-quality display hardware, but this is only a part. The other part, and the more serious at present, is the enormous amount of computation capacity that would have to be rented to maintain a moving, flicker-free, display scene of high- or even of television-quality. To see the scale of the difficulty observe that the full high-speed output rate of the 7094's memory bus could maintain just one television picture (using every bit) provided that it did not have to stop for computation. If computation be taken into account, the 7094 would be exhausted by the requirements of maintaining much lower-quality optical presentations.

It is then as much for computational reasons as cost reasons that we must compromise, and this memo explores directions of compromise. We have observed that the full 7094 is unable to maintain certain kinds of displays; on the other side, there are many kinds of useful displays easily within its reach. Let us list here some of the more important factors in the compromise.

1. <u>There is an important place for a large number of low quality displays.</u>

   Each programmer working with an active program could use a display system that can maintain a legible display of printed characters in a simple text format. A display matrix of, say, 50 x 10 characters would be quite useful. If the character selection is large, this

would be preferable in most respects to the teletype or typewriter output, except where printed output is desired.

2. We can tolerate display quality far below "military" specifications.

Our fairly extensive experience with TX-0 and PDP-1 displays shows that a flickering display, while unpleasant, is very useful. Manufacturers today offer display systems whose generators will put up characters at rates of the order of 400,000 per second. We are now used to rates like 1500 per second (e.g., on PDP-1). Clearly such generator equipment could be shared with a large number of display tubes. This is important because the essential individual display tubes and hardware costs are not much larger than that of a typewriter!

3. Moving (motion-picture) displays are not essential for every program.

For many systems the information can be displayed in fixed-frame format. Here the picture need be regenerated only every one or a few seconds. This means that in such cases it is not necessary to consume computation capacity, provided one can buffer the picture in some other medium.

4. Many display computations do not need the full 7094 facilities.

In many situations the PDP-1 computer could do the required computations at 1/2 to 1/3 the speed of the 7094, and hence at a great deal less expense. The most promising method seems to me to be this; descriptions of pictures are transmitted to the PDP-1 in a picture-language format; these are buffered and handled from then on by the PDP-1. In particular it would be fairly difficult to get the '90 to handle tracking programs, unless they were given a high-priority status in the monitor core.

5. For special advanced projects, time-sharing is incompatible with adequate displays.

We recognize that the MAC 7094 system is marginal in many respects, and cannot be expected to meet computational needs that would tax the machine non-shared. For example, research on animation techniques, which I regard as very feasible today, might run into such limitations. Another example; it is technically feasible to implement hand-printed input (through light-pen) recognition, but this might overtax the computer, unless it had a very special status. We can now do this, more or less marginally, with the PDP-1. Obviously, anything like on-line television analysis is beyond today's computer, though not tomorrow's.

6. Continuously changing displays can be maintained economically only by an auxiliary computer, while maintained, but unchanging displays can be depended from 190 channels.

But the cost of using a channel is not low. One can "steal" just so many cycles before overall system deterioration is noticeable.

7. Restriction to typewriter-class printed output will seriously restrict some on-line applications.

Unfortunately, this probably includes most attempts to bring on-line mathematical techniques into the hands of scientists who would like to use the facility without learning a new and basically restricted and inconvenient format. All technical fields use special notational techniques, whose loss would seriously interfere with the goal of improved interaction.

The Light-Pen Problem

The use of the "light-pen" has become standard in the MIT area, wherever a computer has a scope display console. It is important to know that the light-pen facility has, in itself, a completely trivial cost. All it requires is a photocell-amplifier connected to a flip-flop whose state the computer can read--a "sense-light" or a "status bit". Tracking programs based on simple interrupt or "sequence-break" operations need consume no more than a few percent of com-

puter capacity; they could become a problem with a great many users.

Programming the use of the light pen is quite simple, normally. If one wants a certain spot on the CRT to have a certain program significance, then one simply inserts in his program a test instruction following the operation that causes that point to be lighted. If the status-bit has been changed (indicating that the light-pen has "seen" the spot just displayed) the test instruction causes a jump to an appropriate subroutine which does what is to be done in such a case.

A second mode of light-pen operation is this: the photocell is connected to an interrupt line so that the computer is interrupted when the pen decks a light-pulse. The interrupt, which must store the contents of the program counter as in any other interrupt, transfers the machine to a program which has access to a programmed table of light-pen interrupt conditions. Inspection of the interrupt location then tells the light-pen subroutine what to do then.

There are really two distinct families of applications of the pen. In the tracking situation, one uses the pen to supply some sort of continuous input information to the computer. In the display-sensing mode, the computer asks whether the pen has seen a certain character or picture element: it has in effect asked the user a yes-or-no question. Tracking would be used, say, in drawing a graph. Sensing would be used in a text-editing program, to designate a text-location to be

modified. There is no great difference in the programming appropriate to these because the tracking itself is done by putting up a special set of picture elements. But the difference becomes important when one considers the design of a system in which picture frames are buffered in a separate buffer store.

With the buffer-store, the trouble is this; the appearance of a picture point is not synchronous with the execution of the part of the program responsible for the display of that point! It will do little good to use the interrupt system then, because the current program counter value will be irrelevant. It would be more adequate, however, if the interrupt subroutine could interrogate the buffer system to find out which buffer location was just displayed. But then another serious programming problem intervenes; there has to be a table, somewhere else, giving the relation between buffer display operations and the subroutines which are concerned with the meanings of different light-pen locations.

The solution usually proposed here is that we abandon the light-pen in favor of a non-optical position sensing device. The result is that the computer can interrogate the device, at any time, to find the X-Y coordinates of a hand operated "cursor". This raises the same serious programming problem in an even worse form, because one has then to make a test for the proximity of the pointer to the various significant areas of the pictures; this requires not only a

table but a comparison routine. While this would not be too much trouble where the picture surface can be divided into a meaningful coarse mesh, or where one wants to draw a curve or graph, it would be fairly deadly for applications like Sketchpad, and a problem even for text-editing. While a table could eliminate a search, it would have a high cost in memory reservation and in set-up time. The light-pen in its present form is so natural and convenient that it is worth a lot to try to preserve it. All the mechanical substitute proposals I have seen are relatively clumsy, both manually and in their programming requirements.

A more satisfactory but somewhat expensive solution to this would be a system in which the buffer stores, along with the deflection information, an occasional address or index number for each light pen. When a pulse is detected by the pen, the machine transmits the current (or next) pen index number back to the computer, and this number is used to call the appropriate subroutine. For character sensing this would be quite adequate (provided that the additional buffer space is afforded) but it would not solve all the problems of the drawing-tracking requirement. A deluxe display-maintenance console could incorporate its own tracking feature, moved between frames by simple DDA circuitry.

Below, we propose a system--a deluxe, expensive one. While there is some question of whether the current 7094 system would justify the large system, it can serve as a starting point for discussing the system to be acquired.

A Deluxe System

The system described below might cost between $100,000 and $200,000 if purchased entirely from manufacturers. It has 3 deluxe scope-keyboard consoles, with extras, 6 medium quality smaller scope-keyboard consoles, and a number of "cheap" scopes. The display is maintained by a special-purpose core-memory "computer" with a very restricted order-code. This is an expensive technique, because in many respects the core memory is treated as though it were a sequential drum memory. The advantages are that it can be loaded random-access by the main computer, and can be programmed to repeat fields without going through the whole store. The value of these advantages have to be examined carefully. It is important to recognize that there would be little saved by using a drum here if it were then necessary to hold also a core image of the data in the main computer. One could argue that the display core should be part of the computer core memory. This would be of value, provided that the display channel does not tie up the computer main frame (since it will be operating more or less continuously. In the system below variable storage fields are assigned to the separate scopes, rather than fixed sectors; this means that many "crude" displays can operate while a few are requiring higher data rates. The numerical specifications below are for example purposes only, and are probably not particularly well-balanced.

3 large high-resolution scope-keyboard stations

6 medium quality scope keyboard stations

n≈16 cheap (5") scope keyboard stations

These are all deflected in parallel (cheap!) with simple z-axis blanking selection under program control. This means they will all have to be located near the driver computer. Longer distances will require special lines and (probably) compensated amplifiers.

The store is a 4,000 word, 24 bit, 8 microsecond core memory. For 8-bit character address this means we have up to 12,000 characters, or (mixed 4000 points, lines, etc. This is enough to supply many scopes, provided they do not all require very elaborate pictures at the same time.

The entire memory can be scanned in 32 milliseconds, if necessary. This means we can maintain several flicker-free displays of high complexity. We assume a character time of around 4 μs, and line time of the same order.* By programming, one or more scopes can get preferred service.

Display Features

Characters:	128 fixed, 4 sizes,	4 microseconds
Characters:	128 programmed, 4 sizes,	4 microseconds
Points:	(x,y)	4 microseconds
	(Δx, Δy) (two per word)	

*With the current technology, this means something like 7.5 megacycle video. Things would be somewhat cheaper and simpler with 8μs picture elements.

Lines: two points 4 to 30 microseconds

next point (incremental)

(r, θ) ?

For characters, one wants an incremental scheme like a typewriter.

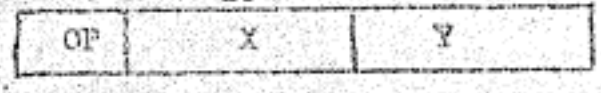
Operation

The standard operating mode is cyclic scanning through a string of user programs linked each to the next. The cycle is interrupted only by user request and by light-pen traps. All traps are of the halt-and-proceed type, because pen-interrupts will usually require immediate attention. The computer operating the Buffer must keep tables to interpret the interrupt transfer vectors supplied by user programs. It also keeps a map of user locations, so that it can re-assign areas and inserts jumps around programs that are being modified or re-loaded. Finally, the computer has on file a compiler to translate picture description expressions coming to it as input.

As will be seen, the Buffer's order code is rich enough to provide many complex services without bothering the computer at all.

Basic Order Code

Class A.



LPR X,Y	Load Position Register with (X,Y).
LRR X,Y	" Relocation " " "
APR X,Y	Increment PR " " "
ARR X,Y	" RR " " "
LRI	Add PR into RR.
DPP X,Y	Display point at (X,Y) and LPR.
DPR X,Y	" " " PR + (X,Y) and LPR.
DPN X,Y	" " " " " . Don't reload PR.
DLP X,Y	Draw line from PR to (X,Y). LPR.
DLI X,Y	" " " PR to PR + (X,Y). LPR.

Class B.

TYP A,B,C — Enter Typewriter mode for next A words, 3 8-bit characters/word. Brightness B, size C. Size and Brightness will hold until reset. Automatic carriage return.

Class C.

WTV N	Select scope N, reset pen flag, PR and RR.
JPF N	Jump on pen flag to N. Reset PF.
JPF' N	" on no pen flag to N.
JUN N	Unconditional jump.
JUM'=NOP	
IPH N	Interrupt computer and halt. Computer can read N directly. Conditional on PF.
IPH' N	Interrupt, etc., on no pen flag.
IUN N	Interrupt unconditional

Class D. Same format as class C, except that bit 4 is a 1.

SPA N	Deposit c[PR] in N.
SRA N	Deposit c[RR] + c[PR] in N.
LRA N	Load RR from N.
SJP N	Complement bit 12 of register N.
SJD N	Jump to N + 1 and deposit ILC in Address part of N.

The order code contains a variety of relocation features. A Relocation Register is provided for shifting a whole picture. Furthermore, things are usually incremental; a base Position Register serves to define connected polygons with one point

address per segment. (One could increase efficiency by adding "near line" and "near point" modes, at two units per word.)

The class B instruction TYP provides for automatic unpacking of BCD text at 3 8-bit characters per word. One might add a 4 6-bit mode. Size and Brightness are selected here for everything (by using TYP 0,B,C). Characters are automatically spaced, with automatic carriage return and incrementing of the PR.

The class C instructions are concerned with light-pen control. The XPH N instruction causes a computer interrupt, and then the number N is available for interpreting the pen action. Bit 12 reverses branch decisions.

Class D instructions are distinguished by the fact that they make data memory references. They are luxuries that make possible fancy automatic conditional displays and pen-tracking, at the price of 2-cycle operation and corresponding complication.

The SJP instruction, in conjunction with class C instructions, allows for fully automatic teaching machine programs without any computer interruption!

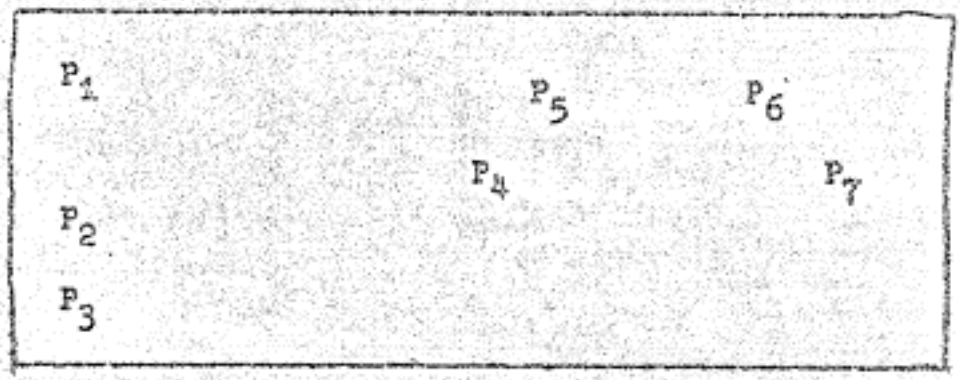
The SJD instruction allows for use of common system subprograms, i.e., useful parts of pictures, graphs, etc.

Example 1.

Here is a display with some light-pen control, and the

program to maintain it.

Scope 17




```
WTV 17          Set-up.

LPR X,Y
TYP 3,2,3       Brightness 2, size 3
"Thi            "This is a"
s□i
s□a"
DPP X2Y2
IPH page 1      Return if pen sees P2.
TYP 4,2,3       "Trapezoid"
"□□□
Tra
Pez
oid"
DPP X3,Y3       Return for pen on P3
IPH page 2
TYP 6,2,3       "Parallelogram"
```

```
"000
par                 "Parallelogram"
all
elo
gra
m00"
LPR X4,Y4           Draw the trapezoid
DLI X5,Y5
DLI X6,Y6
DLI X7,Y7
DLI X4,Y4
ZUN                 Unconditional return.
```

The picture and all takes only only 30 registers. The comment zones show, I think, that the tasks of the compiler and of the original picture-description language are quite straightforward.

Example 2.

Suppose we want to make it possible for the Buffer to provide the console user with branch pictures. We can do this with the SJP instruction as follows. Let ANS1 and ANS2 be the entry location of two programs that are appropriate to the two responses to the frame of example 1. We modify the program above as follows:

```
Q     WSV 17
Q+1   JPF ANS1
Q+2   JPF ANS2
      ...           <---- write "this is a".
      DPP X2,Y2           Switch to ANS1 program.
      JPF' Q+2
      SJP Q+1
      ...           <---- write "trapezoid".
      DPP X3,Y3
      JPF' Q+2            Switch to ANS2 program.
      SJP Q+2
      ...           <---- write "parallelogram".
      ...                 draw trapezoid
```

```
      JUN END
ANS1  LRR 0,0
      LPR X2,Y2
      TYP 5,2,3
      "That          "That is correct".
      t?i
      s[c
      orr
      ect"
      JUN END
ANS2  LRR 0,0
      LPR X3,Y3
      TYP 1,2,3
      No,"           "No,"
      DPP X1,Y1
      JPF" Q+2       Return to original display, if subject
      SJP Q+2        desires.
      "TYP 2,3
      Try            "Try Again?"
      Uag
      ain
      ?QQ"
END   JUN Next
```

This shows how a simple but complete teaching program is maintained without any computer interrupt at all. The locations Q+1, Q+2 serve as a switch which is actuated by light-pen signals from corresponding selection picture elements. An incorrect answer produces a "NO. TRY AGAIN?" picture, and this new picture is also pen-sensitive and can return the display to its original state. Note that only 23 registers were consumed by the two additional simple texts, and that the programming is still simple.

Pen-tracking

We see in the above examples that pen-sensing works very

simply. (We did not attend to resetting the pen flag. One way to do this without wasting instructions is to use a bit in the non-class A instructions.) Tracking is done very easily by using STA. Let $(X_1,Y_1),(X_2,Y_2)\ldots(X_K,Y_K)$ form a suitable "tracking pattern", e.g.,

Supose that (X_j,Y_j) are given in incremental form $(X_x,Y_y)=(0,0)$. Then the following program will track automatically.

```
START   WTV
        LRR X,Y     (X,Y are initial "ink" coordinates)
        DPP X1,Y1
        JPF END-1   test first point
        DPP X2,Y2
        JPF END-1   test second point
        ...
        DPP XK,YK
        JPF END-1   test last point
        LPR 0,0
        STA START+1
END     JUN Next
```

The computer can find where the pen is at any time simply by reading START+1. If interrupt is permissible, then the END instruction is IUN END, and the computer can read the RR. Otherwise, once started, the tracking program requires absolutely no attention, and is called at the overall frame-repetition rate. The program is quite short and can be modified (by using LRA instead of LRR) to be shared by several users.

It turns out that one can very easily program a logarithmic pen-position search in a similar manner.

Some Other Features

1. Use of Common Subroutines

We can economize storage and time in 2 ways, when several users want the same display. (This may occur in many non-obvious cases once things get going; graph formats, common bits of text, instructions for operating the systems, complex relocatable objects and figures, etc. Access can be obtained by

A. Subroutine Calls

SJD A Jump to A+1 and deposit location counter in address part of A.

Then a subroutine ends by jumping back to an instruction

A JUN ---

which returns control to whence it came. Again some protection may be needed between users.

B. Common Select

The SSC, select scope command, can be made to add in another scope but not disconnect the ones already in. Then we need a Disconnect SCope command too.

One disadvantage of this is that the compiler has to worry about interaction between users. This may be worth it.

2. Brightness

It is very useful to have the pen-trigger raise the brightness of the picture element that excited the pen.

this can be done by an instruction

SBR A,B — Deposit B in the B-part of register A (where A contains the class B instructions that set the relevant brightness level.

Discussion

A 24-bit word would allow for three 8-bit characters, or two 12-bit X-Y increment pairs. For text, the typewriter mode is quite efficient. When text can be confined to within some standard 64 character alphabet, then one could use 4 characters per word through another TYP instruction that sets the thing to take 6-bit bytes.

If 9-bit coordinate selection were acceptable, then one could do quite well with another code based on 18-bit (PDP-1 compatible) words. The 9-bits would not compromise character and line quality, but would slightly affect curves.

There is no great advantage in using a class B format for lines here. The DLI is just as good, in the 24-bit version. But in an 18-bit machine a class B mode would be necessary.

The display buffer computer sketched above is a kind of I-O channel computer. It is not quite a general purpose computer, and need not be terribly expensive, but it is complicated enough. It would make a good Master's thesis or two. Except for class D instructions, the registers are loaded in the "immediate mode" (e.g., like the LAW instruction on the PDP-1 or the AXT instruction on the 7090). Therefore, the instruction cycle takes one memory cycle, rather

than the usual two in a typical single address computer. The two cycles required by most of the proposed class D instructions could be omitted if we were prepared to supply a separate RR register for each scope, as selected by the SSC instruction, but this might be expensive.

Other features of the system involve expenses no longer attributable to the buffer computer itself, but rather to necessary scope accessories. Thus the expense of line generation should not be charged to the innovations here. Similarly, the SIZE and BRIGHTNESS options are part of the character generator, as are the 8-bit characters. The typewriter feature is not expensive, since it can be done by treating the X-Y of the position register as a couple, adding an increment to the X part, and carrying out the X part by incrementing the Y part.

Instructions in the Computer Operating the Buffer

SBR X	Start buffer operating at Y.
RBC	Read buffer's instruction register (the address for IFH and IWH).
LBR X	Load buffer register X.
BBT X,Y,Z	Block transfer Y words into buffer starting at X (in computer) and Z (in buffer).
PRC	Proceed (after Halt).
RPR	Read Position.
RRR	Read Relocation.

Reading a buffer register is optional, since one should be able to regard picture output as "write-only". But for diagnostic purposes it would be valuable. Also, for tracking, etc., it can be useful to be able to read the position register.

Probably the most sensible configuration would be to have the buffer core be on a channel from the computer. Our picture here, for MAC, is that this channel is on the PDP-1, which is in turn attached to the 7090. The PDP-1 itself could not maintain the service proposed here. Things might come out extremely well, however, if the buffer core is directly addressable by the PDP-1 as an extra section of memory. In that case, the 24 bits would be bad, and we should consider an order code more suitable for (ugh!) 18 bits.

In the computer, things are not too complicated. Each user is assigned a zone for his transfer vectors (assuming 1-channel interrupt) and the picture compiler keeps track of these; they are symbolic in the user's picture language. The compiler can be quite simple, and still allow the user great flexibility in specifying his text, points, lines, etc., and page options. There are a few points about the interaction of computer and buffer. The computer should be able to load the buffer without slowing it too much. There is a slight danger of the loader catching up with the buffer program, but this could be programmed around. Except for the S-type instructions, there is no user-interaction danger, because the programs don't modify memory. For S-instructions we must be sure that S in in the same user's program, and that an S-operation does not create a new, illegal S-operation.

Since the dangerous case is so restricted, we can make

it fool proof by trickery in the operation-code numbering. For example, it seems likely that S-types will not modify class B or class A D-types, so we can make all S-instructions such bit-4 be 0. If we have the CJB kind of [illegible] protection gyrations are quite serious.

Cost Estimate: Based on no investigation at all.

Memory	30 K	
Character and	15 K	
Line Generators	10 K	
Control and	30 K	
Registers		
Video Amps	15 K	
and cables		
Slave Scopes	30 K	
Consoles		15 consoles at $2000 each with teletype and misc. features.
	130 K	

Continuation

I plan another memo describing a lower-performance system and discussing the oculomotor input scheme; its promise, and the associated human instrumentation problem. This memo will shortly be incremented by an order-code for an 18-bit version more appropriate to the PDP-1, 709⅓ complex.