ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

IN THIS ISSUE:

HARDWARE NOTES by Larry Stewart

A third disk drive has arrived. It will probably stay in its present position under the sign-up board. This is so that people desiring additional machine time may open the door suddenly and dislodge their competitors magnets. The disk will run on the old 7/32 whenever the controller arrives.

One of the recently arrived printers has been stashed in the 79-B closet (across from "Flash Gordon and the 'Purple Death'"). It was necessary to be somewhat ruthless with the other closet contents to make it fit. Be cautious! The other printer now reposes near the old printer. You can tell them apart because they face in opposite directions. Efforts on the part of Steve Lang should soon lead to the printed word. Precisely in what manner the printer will connect to the system is as yet unknown. Probably it will be standalone on the old 7/32 or just possibly, shared.

An Imlac is due shortly from Imlac. Bob Hoffman has built the Imlac end of the Interim2 link. Hopefully, x'3E' should be up before long.

Mike Slezak, with some small assistance, has been working on re-mounting the mechanicals of the 85 display system for improved access and cooling, and reduced grumpiness on the part of maintenance personnel. The mechanicals are complete and should be operating by Monday or so. Users of the 85 should be warned that Andy Lippman and myself will be poking at the 85, the memory, and the display much of the coming week in order to undermine the foundations of Seth's quote of last week. We will try green screwdrivers on Wednesday and Duodecaplylatomate on Thursday.

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

THE ANNALS OF SOFTWARE RESEARCH -- A DESIGN FOR AN ANIMATION LANGUAGE

by Seth Steinberg

Paul Pangaro and I have decided that it is time to think seriously about developing a new animation language for use here at the Architecture Machine. We have done a bit of shopping around and are currently thinking about a node-network-oriented language. Animation languages, like simulation languages can be thought of as node-network-oriented (IBM's GPSS for example) or as "multitasking" systems (SIMSCRIPT for example). There are obviously systems in the middle.

In a "multitasking" system, the action which takes place on the screen is broken into linear sequences of drawings. For example, a "script" could contain instructions to draw a moving cart. This "script" can be run in a task in a given local time scale and at a given position so that the cart (or a number of carts) could appear to be moving at different speeds and in different directions, even though there is only one "script". This is much like the use of a re-entrant editor on a time-sharing system; each user edits a different file.

In a node-network system, the picture can be thought of as a "signal" passing through wires. The wires can connect various nodes, each of which has some effect or another on the signal. Thus, there could be a moving cart source node, the output of which can be fed into a number of displacement, rotation, and the like nodes, to produce a number of different carts moving around on the screen.

Taken to one extreme, the node-network system can be used in conjunction with busy boxes (dials, buttons, etc.) to produce interactive real-time animation sequences. (For example, the Chicago Circle Graphics Habitat can generate simple pictures that can be controlled in real time.) The actual approach we are taking will not be real-time. Images will be generated as fast as they can be, and the next frame will be flashed on the screen as soon as possible.

The system will probably resemble PAINT in some ways. Most functions can be performed using just a tablet and a control screen menu. (We were thinking of 8 - 383x384 planes for the picture and 3 - 384x384 planes for the network editor.) There will be modes and submodes, and the network can be manipulated graphically to a certain extent. Debugging can be accomplished by moving around a "probe" which allows the various intermediate points of the node to be examined.

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

THE ANNALS OF SOFTWARE RESEARCH -- A DESIGN FOR AN ANIMATION SYSTEM
continued

The basic "types" which the system will know about are as follows:

numbers	simple values
points	positions on the screen
colors	colors
lines	lines are determined by a number of points along them, and may form closed shapes which can then have "flood points" and colors in which they will be displayed

In addition, groups of things can be passed from node to node. So, if we wish to rotate a set of lines around a given point, the lines could be grouped together and passed to a rotation node. The output would in turn be a group of the same order as the input group. Similarly, if a set of lines are to be rotated around a set of points, the lines could be grouped, and the points could be grouped and the result would be a group of lines of the same order as the input group. If this troubles you, think of the nodes as functions which are "mapped" (in the LISP sense) over their arguments, or think of how the Multics command processor handles parenthesized strings.

There will be a large number of basic nodes, at least fifty, and maybe more. Some of these are for very simple things such as multiplying two numbers, or constructing a point out of two numbers. Other nodes will allow lines to be rotated around various points, parts of lines to be extracted, and points to be extracted from along lines. This latter allows a line to be used as a path for another object. In addition, it will be possible to extract intermediate (or extended) images which are defined by the initial and final lines.

We will make it as easy as we can to add new nodes so that the cost of experimentation can be kept low. This entire project is partially an experiment in its own design. A number of very important features have been left out. A good example of something missing is the lack of bit patterns as a basic type. These could be "colorized", rotated, scaled, and so on, but would involve grossly increasing the size (though probably not the complexity) of the prototype.

In addition to the basic nodes, will be the user defined nodes. A user can define a node by drawing a network of more primitive nodes

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

THE ANNALS OF SOFTWARE RESEARCH -- A DESIGN FOR AN ANIMATION SYSTEM
continued

and specifying the inputs and outputs to the node. Thus, interesting transforms can be saved, edited, and used in other sequences. There could be a graphic node compiler which could convert ASCII text descriptions to and from node networks to help provide printed documentation of the networks.

Now, for an example. We will take the rotating lines example from Paul's article last week. The program he used was a simple recursive program which went one level deeper each time tne number of line segments is doubled.

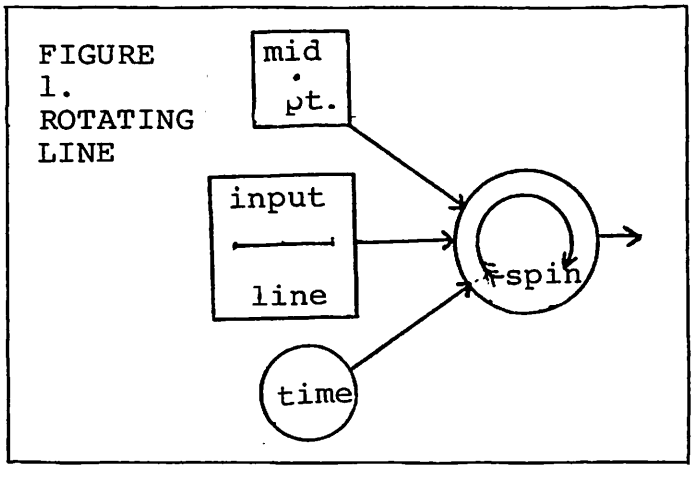
So to state the programming problem:

a) the basic state is a simple rotating line.
b) the line can be broken into halves each of which rotates about its own midpoint
c) the midpoints maintain their original rotation derived from the basic spinning line
d) the rate of rotation is multiplied by some factor as the length of the displayed lines shorten.

The basic state is pretty simple. A line can be drawn in using a graphic editor, or described by its two endpoints. (We are using the word line in the "shortest distance twixt two points" sense, as well as the more complex primitive data-type sense.) The line can be fed into a rotation node along with a time value, and a center point.

Figure 1 (to the right) shows how to set up a rotating line. We now want to divide the line into two halves, each rotating around its own midpoint. To do this, we split the line into two pieces (nodes A1 and A2 in Figure 2), and then group the two pieces together (node B). Node C finds the midpoints of each of its input lines and sends a group of points into the rotator (node D). The rotator receives a group of lines and a group of points (and

FIGURE 1. ROTATING LINE



ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

THE ANNALS OF SOFTWARE RESEARCH -- A DESIGN FOR AN ANIMATION SYSTEM
continued

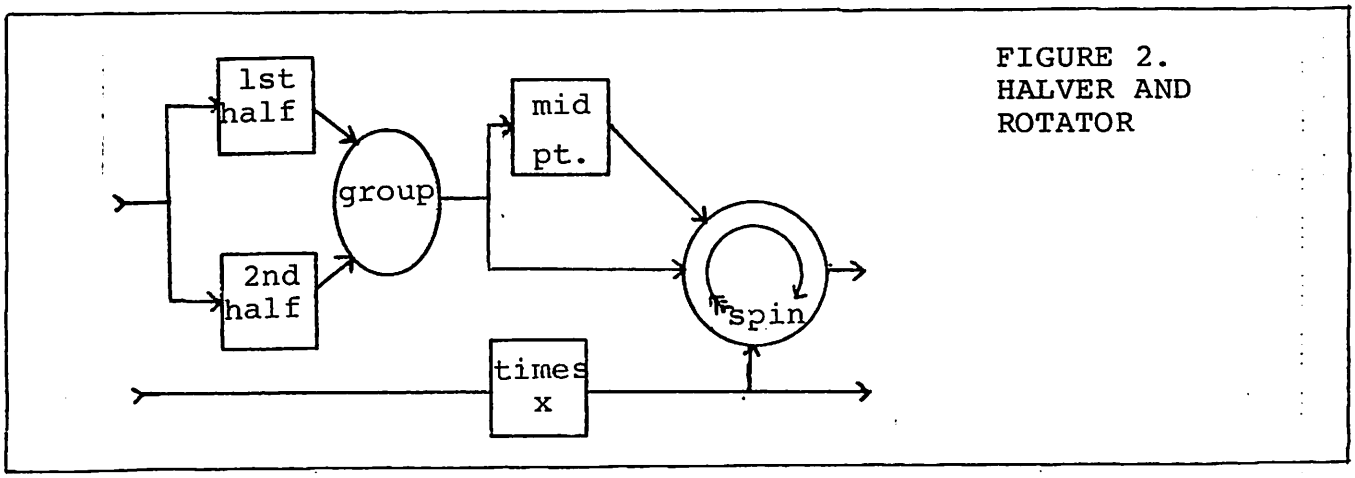


FIGURE 2.
HALVER AND
ROTATOR

the original time value times some x), and puts out a number of rotated lines.

The mechanism in Figure 2 above could be turned into a user-defined node simply by replacing its inputs with argument input nodes, and its outputs with argument output nodes, and thus forming an interesting node in its own right, a "break it in half and rotate each half around its own midpoint" node. The rotating line from Figure 1 could be hooked up to it, or the output from Figure 2 could be hooked up to it and so on. Each time this node is connected to a previous node's output, the number of lines doubles, and the complexity of the motion doubles (see Figure 3 on the next page).

The animation language should be easy to learn since once the mechanics of network editing are worked out, any new combination of nodes can be tested out simply. In general it would be nice if the network editor and the interpreter could fit in the same overlay. Perhaps they can, or perhaps the basic editing and running functions can fit together, but the more complex editing functions and running functions will require overlays.

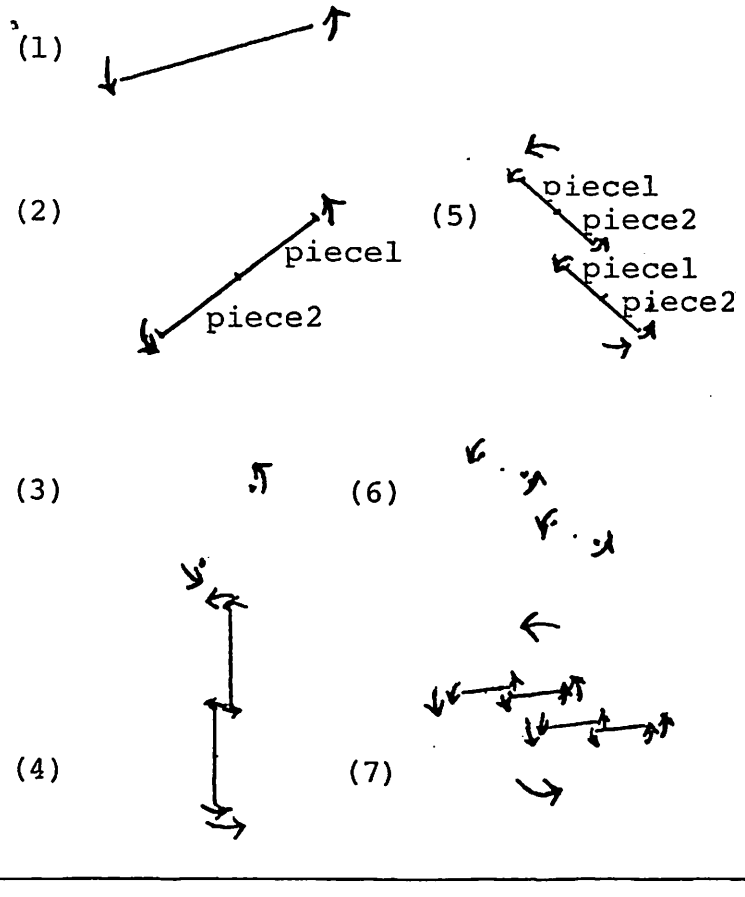
This design is in no way a final one. Many changes will occur, perhaps even in the basic design. (We could use the extra three windows the editor needs for overlay storage, and edit the network on the Imlac). Any suggestions you may have will be appreciated.

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

THE ANNALS OF SOFTWARE RESEARCH -- A DESIGN FOR AN ANIMATION SYSTEM
continued



(1) original input
(2) two pieces
(3) midpoints of pieces
(4) pieces rotating around midpoints
(5) like (2)
(6) like (3)
(7) like (4)

FIGURE 3. INTERMEDIATE STEPS

HELP WANTED

CUSTOMER SPECIALS PROGRAMMERS Positions available in the development of special purpose software for a mini-computer based graphics system. Jobs involve close customer contact, program development and test. Knowledge of Fortran, assembly language, data communications, interactive graphics, I/O handling, operating systems, and real-time control a definite plus.

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

HELP WANTED continued

N.C. PROGRAMMERS Positions available in the development of N/C postprocessors for machine tool control. Knowledge of Fortran, Apt and machine tool control desirable.

Computervision is a leader in mini-computer based interactive systems for major industries, including IC, PC, mapping, mechanical design, numerical control machining.

Please submit resume and salary history, in confidence, to Personnel Department. Computervision, 201 Burlington Road, Bedford, Mass.

TSD NOTES by Steve Mann

The TSD Software Development Effort (Lippman, Herot, Stewart, Lang, and, lately, myself) brought you the the TSD.SGIM which supports the POINT/PPLINE package, TABLET, and CLRTAB subroutine calls. The current TSD.SGIM can be ATTACH'ed by the abbreviation FG (remember Finger Graphics) and is available now on the fixed-head system only. I have become default Keeper of the Sgim, so see me about problems. What follows are notes of general interest for people who will be using the TSD.

The TSD.SGIM program currently works continually as a background process using the interrupt mechanism. The TSD generates an interrupt at a fixed length of time after being given a command to pulse any transducer. When an interrupt from the device occurs, MAGIC suspends the current program operation and calls the TSD interrupt handler. The TSD interrupt handler stores the information received from the pulse which generated the interrupt, sends a new pulse command, and returns. In this way, the TSD runs continuously, stealing a certain percentage of time away from the user program.

All application programs, which use the TSD in the MAGIC environment, use the TSD.SGIM (Static Graphics Interface Module) either directly or indirectly to get coordinate and touch information. As mentioned in recent A.M. issues, the TSD senses on six channels, three in the x direction and three in the y direction. Each channel has eleven

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

TSD NOTES continued

transducers, which send and receive. On each axis, the three sets of transducers are interleaved.

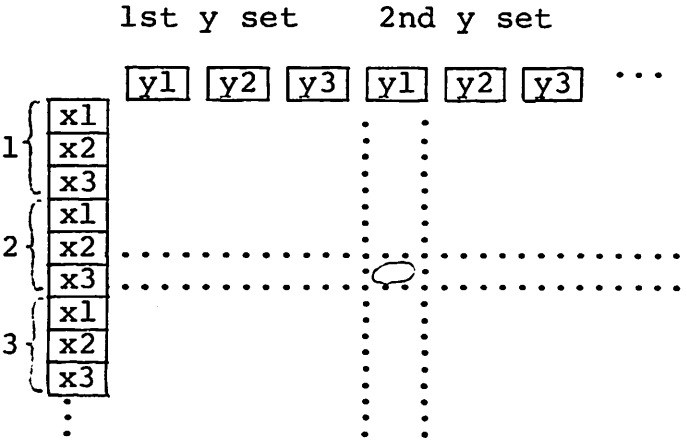


Figure 1. Transducer Locations: Upper left corner of TSD

The design of the hardware interface to the TSD allows you to command send pulse/ receive on any combination of x1, x2, x3, y1, y2, y3 channels. This allows a great deal of freedom in designing how the SGIM will work.

There are two ways in which application programs can use the TSD as a tablet which returns (x,y) coordinates and touching information. The TABLET subroutine, which uses the POINT/PPLINE static graphics system, requires the TSD.SGIM to return only x,y,z information when called. The overhead involved in a TABLET call is the need to link the large POINT/PPLINE package (about 2K bytes). The advantage is that automatic coordinate transformations are done. The other alternative is to call the TSD.SGIM more directly, getting back raw device coordinates. The disadvantage to this is that there is no method for transforming the coordinates to user coordinates. The advantage is that access to the touch information is quick and requires little additional space in the program.

In cases where only touch information is needed, using the direct call to the SGIM is more efficient.

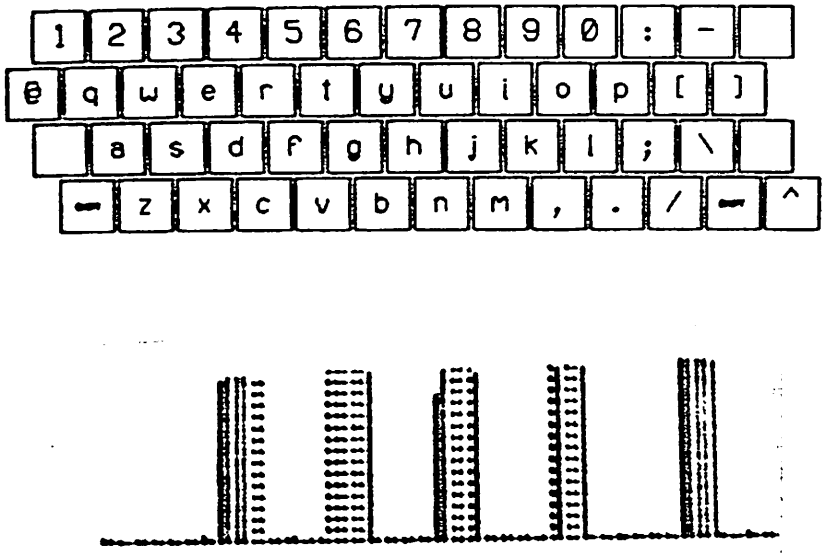
To get a general idea of how well the TSD software operated I wrote a program that relied heavily on the ability to discern where on the screen a person pointed. Using the Imlac to display the keyboard and the TSD to supply the touching information, I took the "virtual keyboard" of Knowlton's experiment one step further in flexibility. Not only can the symbols on the keys be changed instantly to reflect different functions, but the keys themselves are virtual rather than

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

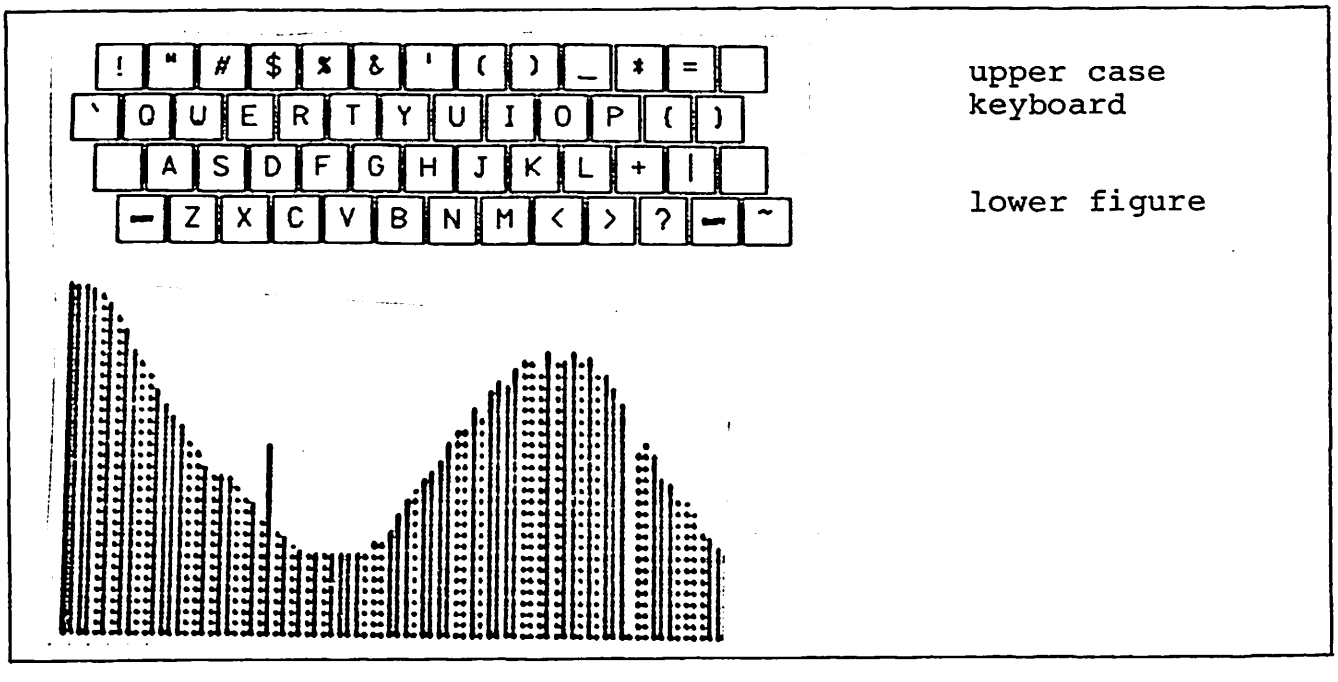
Vol. II., No. 29. July 18, 1976

TSD NOTES continued



lower case keyboard

upper figure about 2 seconds



upper case keyboard

lower figure

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

TSD NOTES continued

physical.

Experience using the keyboard program showed the need for better low-level strategies to reduce the uncertainty in the positional information. Since the TSD routines run continuously as a background process to the user process, standard on-line debugging methods cannot reflect back the dynamics of the situation.

To evaluate the performance of the SGIM/TSD, I'm using a simple graphical output program that plots and samples the TSD output continuously. When the TSD and its interface are moved to the 7/32, precise timing information will become available. The figures on the previous page are histogram snapshots of two seconds of the raw coordinate output of the TSD. The figures show solid lines for touch indicated, dotted lines for no touch indicated. The height of the line indicates the value of the y coordinate returned. The sample rate is about 20 milliseconds. The stimulus for the upper figure is simply the sliding of the finger up and down the screen. The stimulus for the lower figure is a finger tapping five times on the screen.

DOCUMENTATION

Major revisions to MAGIC SYSTEM MAINTENANCE COMMANDS and MAGIC COMMANDS have been incorporated in the new manuals. Also, thanks largely to Ben McCann, MAGIC PL/1 SUBROUTINES has recently been reissued with many corrections and expansions. Currently in the works we have a new manual entitled MAGIC INTERNAL DESCRIPTIONS which consists of a series of memos on obscure Steinbergian points. It should be available by the end of the week.

QUOTE OF THE WEEK by Steve Lang, Hardware Whiz

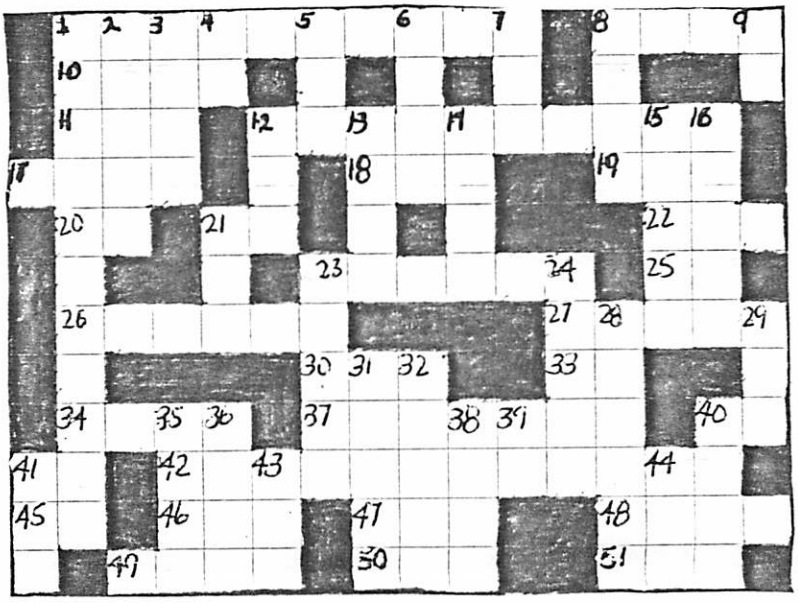
"Let's get out of here before it falls off."

ARCHITECTURE MACHINATIONS

A weekly newsletter of the Architecture Machine Group, Department of Architecture, M.I.T., Room 9-518, Lee Nason, editor.

Vol. II., No. 29. July 18, 1976

CROSSWORD PUZZLE



ACROSS

1. When the hackers come in
8. A, B, and C for example
10. Signup riot time
11. Base for binary
12. One way of doing arch.
17. What happened to OSLOAD
18. Have
19. Function for checksums
20. Createuser to ATTACH
21. Possessive pronoun
22. Habraken's design methodology
23. Type of display on the 85
25. Start of MAGIC
26. Good MAGIC word with chango
27. One metal in BMW's
30. One way to be held
33. A 39 down he ain't
34. M.I.T.
37. MAGIC does this often
40. Start of a constitution
41. About
42. 4
45. Lee Nason, abbr.
46. Ringer
47. Base for decimal
48. Time spent in the linker
49. Something to do with do's
50. Block balancer
51. A part of the ACM

DOWN

1. The arrival of things shipped
2. _____ of Hanoi
3. Word with foul or ring
4. Opposite of OUT
5. Opposite of AYE
6. Programs tend to do this
7. _____ and feathers
8. Acme
9. Start for short
12. Back _____
13. Dr. Friedman
14. One Unitarian
15. What 16 does with dirty heads
16. Word with key or up
21. Mr. Nagashima
23. Word for a system crock
24. Initialize
28. Units of Machinations
29. Compass Point
31. Type of ring
32. Happened to naughty paper tapes
35. Holy media
36. Makes a mistake
38. Oft done to BOX
39. _____ 2314
40. Will _____
41. Korf
43. The 9-514 couch
44. AL in France

THE END

2314

MMCCCXIV