# ESCHER: A Block-Oriented Graphics Language for Microcomputers

Dr Harvey A. Cohen
Computer Science Department
La Trobe University
Bundoora, Victoria 3083

## ABSTRACT

ESCHER is an interpretive graphics language tailored to the pictorial capabilities of the relatively coarse grained memory mapped video output available in today's personal computers, and so-called "video computers". The language is block oriented, rather than line oriented. The user creates and merges rectangular subpictures called motifs, to form whole screen views, which in their turn may be stored and combined together. Motif manipulation and definition is via the mediation of a screen creature called a NAKI. A unique screen arithmetic enables the immediate undoing of any motif dumping or whole screen addition despite superimposition. The language has a simple command structure, with immediate execution of completed commands. The programmer may define run-time macros that include conditional branching dependent on the state of a number register or on NAKI location. Using ESCHER, it is possible to readily program a range of video games involving repeated patterns and shapes, and numeric scoring. This use of ESCHER is illustrated in the compact implementation of a preschool video calculator called the PLUSMINUS.

## INTRODUCING ESCHER

ESCHER was conceived to creatively exploit the graphic capabilities for relatively coarse TV graphics with the smallest video dot, a pixel, visually resolveable. In ESCHER the TV screen is interpreted as composed of small rectangular blocks of pixels, called cells. The contents of a cell is termed a graphic character. Underlying ESCHER is the notion of a screen object called a NAKI. This screen robot has rather more elaborate behavior than the "TV Turtle" used in the line drawing language LOGO (Papert, 1973; Goldberg, 1976). The NAKI when made visible appears as a flashing asterisk that alternates with the graphic character that occupies the same cell. The NAKI can erase the cell it occupies, or replace the contents of that cell by another graphic character. The NAKI also controls a rectangular array of cells with the NAKI located at the upper left hand corner. As the NAKI moves from cell to cell about the screen, its rectangle moves with it.

ESCHER commands, and user defined macros, are of one character, which may be preceeded by a number (of repetitions or auxiliary parameter), and sometimes require a following digit or upper case character serving as an index for their completion.

The NAKI can move about the TV screen from cell to cell in obedience to the basic movement commands, n for north, e for east, s for south, and w for west. ( "north" on the TV screen is upwards as in maps). The number of cells moved on (but not off) the screen in these four directions is specified by the number preceeding the movement command. There are special cells on the screen that are "homes" for the NAKI. On the command Nh, where N is some integer, the current location of the NAKI becomes home number N. Then on the command Nj, the NAKI will jump to home number N.

## MOTIF DEFINITION AND SCREEN PICTURES

The contents of the NAKI rectangle are stored as a named motif with single character name <char> on the command z<char>. Thus, for instance, one might define a complete set of motifs to provide format alpha-numeric characters, with each appropriately named. The NAKI will dump motif <char> into the NAKI rectangle on the command @<char>, adding each pixel of the stored motif <char> to each pixel of the rectangle according to the picture addition rules:

BLACK + BLACK = BLACK
BLACK + WHITE = WHITE
WHITE + WHITE = BLACK

Hence performing a command @<char> twice in succession leaves the NAKI rectangle as it was. This is one of several ways in which ESCHER is kind and forgiving to the user; the effect of any injudicious screen addition is erased by repeating the incorrect operation.

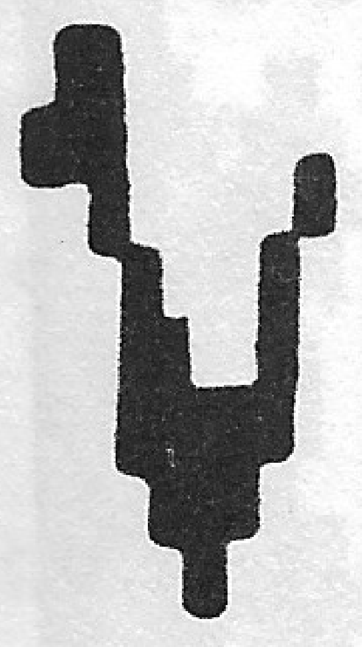
The whole screen may be saved as picture number <num> on the command t<num>. A <num> greater than 9 has to be enclosed with brackets. The contents of screen picture <num> are added to the screen on the command v<num>, the addition being performed pixel by pixel in accord with the picture arithmetic formula presented above.

There are two whole screen reflection commands, m<num> for reflection in a horizontal mirror, and l<num>for reflection in a vertical line. These commands leave the drawing screen unchanged, but store the reflected whole screen as picture number <num>.

In some circumstances one wishes to simply copy a motif exactly into a NAKI rectangle. The command to so dump motif <char> into the NAKI rectangle is #<num>, where <num> equals the ASCII value of <char> minus the ASCII value of "0". (So #<digit> dumps the motif <digit> exactly).

The command to clear the entire drawing screen is just c (for clear), whereas the NAKI rectangle is blanked on the command b. Another command to be used in an example below is the command r, which video reverses (black to white and vicevera) all pixels within the NAKI rectangle.

Example 1: Wall Paper Design



The simpler graphic features of ESCHER are well demonstrated in the following account of how a wall paper design is produced. Suppose that the motif d (for duck) displayed to the left has been defined. The screen may be cleared on the command c (for clear). Moving the NAKI to the top left of the screen, the motif can be dumped to form a row of eight on the command

8(@d8e)

Then following the commands

```
4s4w7(@d8w)
```

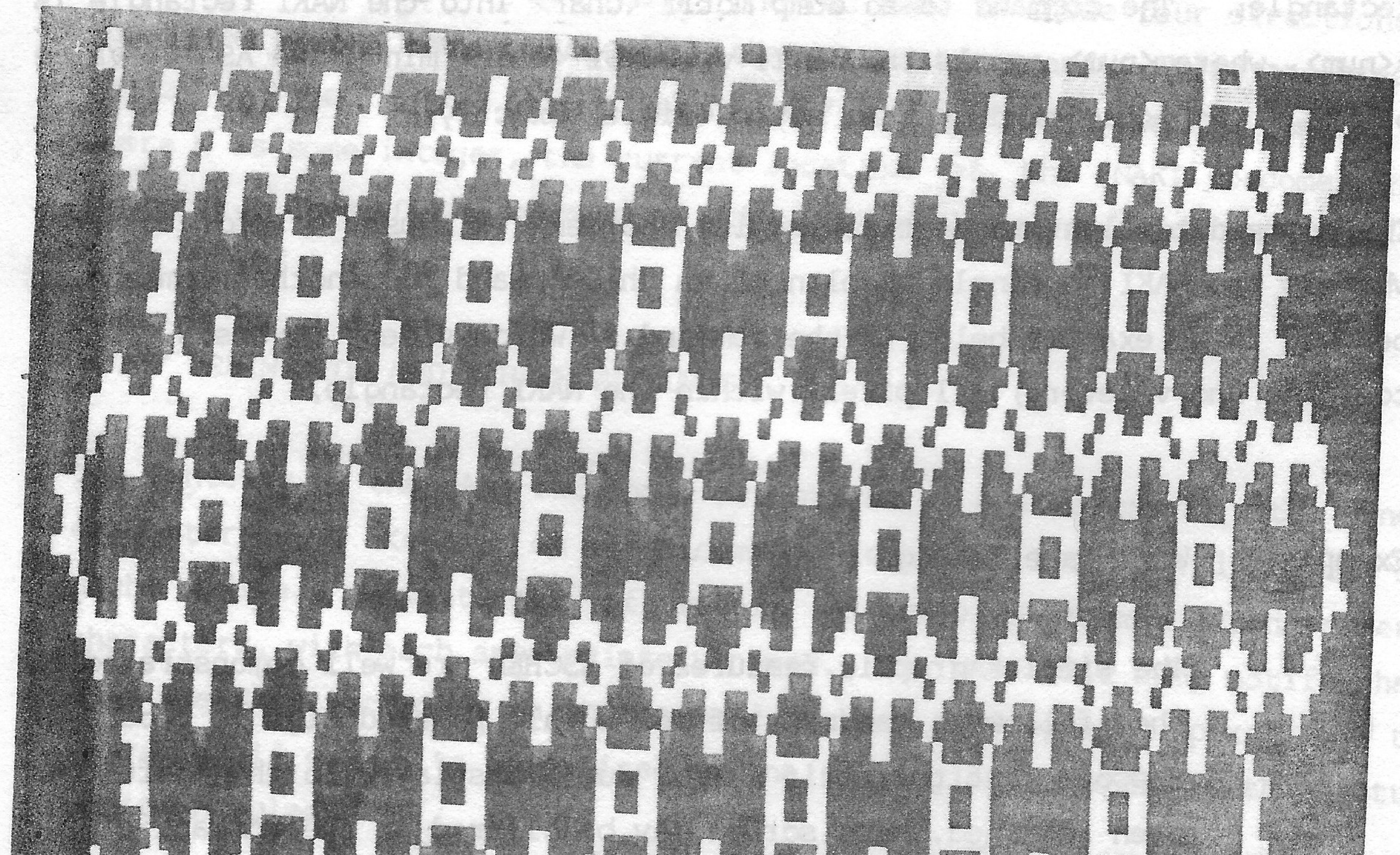
a second row, staggered with respect to the first may be produced. Continuing on in this manner, the whole screen may be covered with staggered rows of ducks all facing the same way. On the command,

```
l1
```

a copy of the screen with the ducks facing the opposite way, will be stored as picture 1. Adding this stored picture to the screen yields a simple fret-work like design with vertical symmetry. Now making the futher reflection in the horizontal and storing in 2, by the commamnd

```
m2
```

we can now add the picture 2 to the screen on the command v2 yielding



THE COMMAND SCREEN

So far we have only described the drawing screen, the TV picture on which a NAKI roams. However, when ESCHER is used as an author language, there is a second video output to the command screen, which shows the input command buffer, used defined macros, and two number registers called A and B.

```
A=007   B=010
U=(0j#(A)Aj#(B)! UPDATE)
V=(9S( )(+U)! "Plus")
W=(0S( )(b-U)! "Minus")
D=(%A-11+%G! "All dogs")
E=(%A-10+%G! "All trains")
F=(%A-12+%G! "All birds")
G=(1jA(#(B)7e)Aj! REMAKE ROW)
```

2V

The ESCHER command screen, as shown in the printout above, is very similar to that used in the educational macro language WIZ77 (Cohen, 1978,9). The left hand side of the command screen is reserved for an yet unimplemented system for drawing motifs pixel by pixel, using pseudo-pixels as large as the cells on the drawing screen. On the bottom right is the input buffer. A macro of name <char>, where <char> is upper case and not a reserved character, may be defined as textually equivalent to the completed command stream "listl" using the command

Z<char>(listl)

Following its definition, the macro named <char> appears in the macro list in the form <char>=(listl) On the above printout, the defined macros are those used in the PLUSMINUS implementation described below.

The command <number>+ adds <number> to register A, while <number>- subtracts one from A <number> times, with the proviso that A does not decrease when its zero. The contents of these registers are referenced by name, so that A- sets register A to zero. The command % swaps the contents of A and B registers.

The conditional commands have the form

```
        <num>Y(list1)(list2)
```

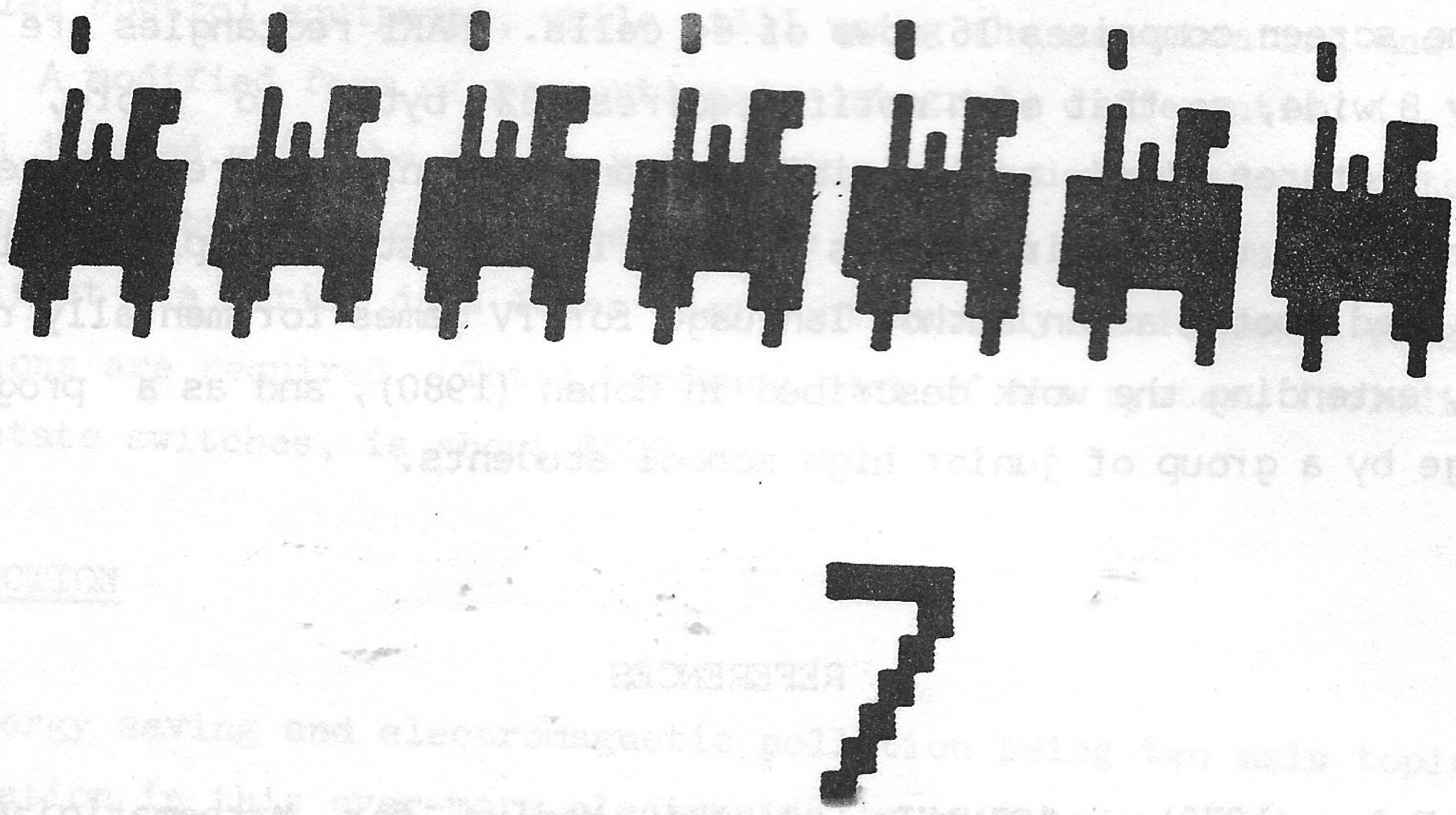
where the brackets about the command lists are essential. If the (Boolean) condition Y is true, then the command list list1 is executed, otherwise list2. An example of such a conditional is <num>T, the condition that A is equal to <num>. The conditional <num>S tests whether the NAKI is on home cell number <num>.

Example 2: The PLUSMINUS

Using the conditionals and macro facilities it is possible to use ESCHER to readily implement a range of TV games of educational value. This capablility will be demonstrated via ESCHER implementation of the PLUSMINUS, a program used to introduce preschoolers to elementary number concepts. (See Cohen, 1979).

In PLUSMINUS, the child user commands the screen using a set of large keys appropriately labeled by its teacher. For clarity, we shall talk as though these labels were multi-character as, for instance, "All trains", whereas the actual Ascii value of the key in this instance is E. But the macro E is defined to include the comment ! "All trains", so that the relation of user keys to command characters is clear. However the keys marked with digits on the child's keyboard do in fact output the corresponding Ascii digit to the microcomputer controller.

In PLUSMINUS the (drawing) TV screen displays a set of objects, together with a large format display of the number count. On pressing the "Plus" key, the count increases by one, up to a maximum of nine objects. On pressing the "Minus" key, one object disappears, provided there is one object at least on the screen. On pressing the "All trains" key, all the objects displayed become trains, the "All dogs" command makes for all dogs, the "All birds" key makes for all birds. The digit keys work as on a reverse Polish calculator, so that, for instance, the command sequence 3 "Plus" adds three to the display of object and corresponding count.



The first stage in implementing PLUSMINUS is to produce the screen animals. We suppose the BIRD motif is #(12), the train motif is #(10), and the dog motif is #(11). Large format numbers 0 to 9 are naturally labelled 0 to 9. There are two homes to be formed, home zero, where the NAKI will update the number display, and home 1, where the NAKI will dump the first screen "animal" at the left hand edge of the screen. In addition, in order to demonstrate macros constructed using the j command, we require homes 2 to 9 to be also formed at the same row, at 7 cell spacing.

In the implementation of PLUSMINUS specified by the above macros shown on the command screen, the NAKI sits where it has dumped the Nth screen animal, at home number N. On the "Plus" command, it checks whether it is located at home 9, and if so, does nothing, otherwise A is increased by those used in the PLUSMINUS implementation described below.

The command <number>+ adds <number> to register A, while <number>- subtracts one from A <number> times, with the proviso that A does not decrease when its zero. The contents of these registers are referenced by name, so that A- sets register A to zero. The command % swaps the contents of A and B registers.

The conditional commands have the form

```
        <num>Y(list1)(list2)
```

where the brackets about the command lists are essential. If the (Boolean) condition Y is true, then the command list list1 is executed, otherwise list2. An example of such a conditional is <num>T, the condition that A is equal to <num>. The conditional <num>S tests whether the NAKI is on home cell number <num>.

Example 2: The PLUSMINUS

Using the conditionals and macro facilities it is possible to use ESCHER to readily implement a range of TV games of educational value. This capablility will be demonstrated via ESCHER implementation of the PLUSMINUS, a program used to introduce preschoolers to elementary number concepts. (See Cohen, 1979).

In PLUSMINUS, the child user commands the screen using a set of large keys appropriately labeled by its teacher. For clarity, we shall talk as though these labels were multi-character as, for instance, "All trains", whereas the actual Ascii value of the key in this instance is E. But the macro E is defined to include the comment ! "All trains", so that the relation of user keys to command characters is clear. However the keys marked with digits on the child's keyboard do in fact output the corresponding Ascii digit to the microcomputer controller.

In PLUSMINUS the (drawing) TV screen displays a set of objects, together one, and the row updated using the G macro. On the "All dogs" command, the number in the B register becomes 11, and then the row of animals is remade using the G macro.

## IMPLEMENTATION AND USE

The current implementation of ESCHER is for 8080 based personal computers using Polymorphic Systems VDM (memory mapped video rams) for video output. When ESCHER is used as an author language two VDM boards are required, in order to display both the command screen and the drawing screen. To match the hardware used for video rams, cells are three pixels deep by two wide, and the screen comprises 16 rows of 64 cells. NAKI rectangles are 4 cells deep by 8 wide, so that each motif requires 32 bytes to store, whereas screen pictures (and an invisible command screen) require 1K bytes. The minimum useful ram requirement is 16K. The existing implementation is being used both as an author language for TV games for mentally retarded pupils, extending the work described in Cohen (1980), and as a programming language by a group of junior high school students.

## REFERENCES

COHEN, H.A. (1978): "OZNAKI: A New Medium for Mathematicians", in D. Williams (Editor), "Learning and Applying Mathematics", Published by the Australian Association of Mathematics Teachers, pp 274-283.

COHEN, H.A. (1979): "OZNAKI and BEYOND", in D. Harris (Editor), Proceedings of NECC 1979 National Education Computing Conference, The University of Iowa, Iowa, pp 170-178.

COHEN, H.A. (1980): "Expanding the Child's Concept of Number, Space and Operation", in M. Poole (Editor), "From Creativity to Curriculum", Allen and Unwin, Sydney, pp 147-162.

GOLDBERG, A., and KAY, A. (1976): "Personal Dynamic Media", Xerox Palo Alto Research Center Report, 1976.

PAPERT, S. (1973): "Uses of Technology to Enhance Education", LOGO Memo No. 8, M.I.T. Artificial Intelligence Laboratory.