MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence Project
Vision Memo. No. 120 February, 1967.

Marvin Minsky

This Memo. proposes a set of systems programs for vision work. Please comment immediately as we should start on it at once.

Picture Arrays

All p-arrays are to be considered as sub-arrays of the "full picture" whose coordinates have the range

$$0 < x < 37777$$
$$0 < x < 37777$$

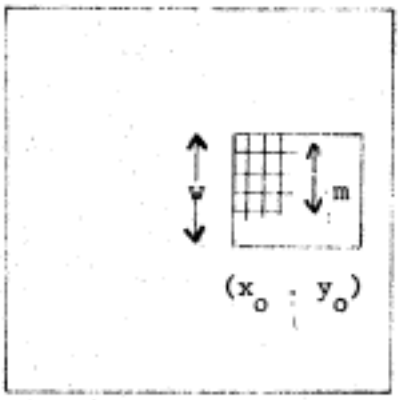
This covers the full circular field of view: the inscribed square will have a range of approximately

$$5000 << 33000$$

Values stored outside an array range should have no effect, but set an overflow flag: values read outside a range are zero and also should set a flag.

Coordinates normally occur as a dotted pair (x . y) in half words. For display purposes, normally the 10_{10} most significant bits are used, but higher resolution options will be available.

To specify a sub-array we have to state its size, location and mesh. All sub-arrays will be square. (Generalizing to rectangles is unwise because the natural generalization for later systems will be projective). A sub-array has the form:



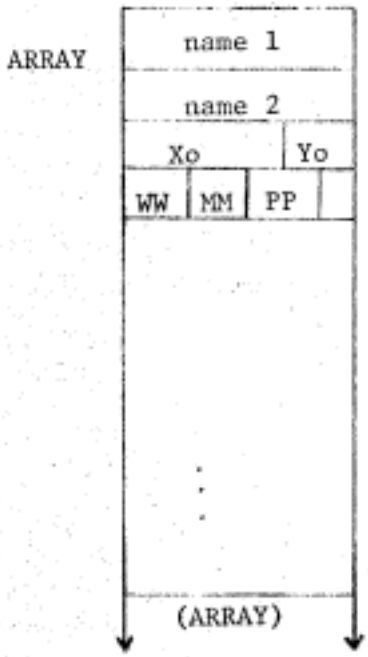
full picture

$(x_o . y_o)$ = lower left corner

w = width = 2^{ww}

m = mesh size = 2^{mm}

An entry to the array is a block of 2^{PP} words. An array is headed by a 16 word identification block, that can even contain an indexing program.



In any case calling ARRAY(POINT) should get the address.

ARRAY + 20_8 + 2^P ($(x - x_o)2^{-m}$ + $(y - y_o)2^{w-m}$) with the m low order bits of x and y masked out.

Array-Handling Functions

We want an extremely versatile system function for moving, reading and storing picture arrays. The system should be able to:

Read from secondary storage a sub-array of any array.

Write arrays out on storage.

Move within core memory.

To do this one should be able to say

AMOVE ARRAY, NAME, DEVICE, MODE

which finds the array called NAME in the secondary store called DEVICE and brings it into core at ARRAY. ARRAY is presumed to contain the x_o, y_o, m,w,p parameters, and AMOVE must extract this sub-array. If

NAME=0, AMOVE gets the name from ARRAY and ARRAY+1. If DEVICE is CORE, then NAME must point to another array in core. If the Destination Array is denser, i.e. its m is smaller than the source's, it should still load properly. The same conditions should apply

AMOVEM

except that here the source is core and the destination is DEVICE. If NAME=0 a file is created with the name in ARRAY, ARRAY+1; if such a file exists it is overwritten.

THE DEVICE MODES

These are a collection of important operating and debugging system programs. MODEs control devices and intra-array formats.

For TVC

MODE is a pair of pointers

P1 points to a word that sets the blur, threshold, and dark cut-off of TVC.

P2 is a format word that specifies what part of TVC's output goes in which byte of the array block. Zero means right ten bits.

For tape, disc

MODE is again a pair of pointers that specify

P1 Bytes in the record

P2 Bytes in the core array

Some thought should be given to what happens when the p-parameters are different for the source and destination blocks.

For the 340 Display

This is most important. Mode should specify

1. Is display compiled? If so, is it optimized?
2. What bytes do what? We need to be able to plot intensity, numeric, displacements, etc.
3. Normally, scale is same as data. But one needs to be able to specify a new CENTER and SCALE FACTOR.

4. A superposed coordinate scale should be available, with calibrated axes.

For the Printer

Numeric printouts, with good labelled margins. Should print alternate lines to give 6:5 layout - much better than normal 3:5 distortion. 4 digits per entry. MODE should be able to call a data function for making the entries: normally a two digit number.

For the Plotter

We should be able to plot whatever is compiled for the display. MODE should allow for insertion of new modes, such as Krakauer's contour plots.